GROVES, Brenton Robert, 1930-
THE USE OF COMPUTER-GENERATED MOTION PICTURES IN THE ANALYSIS OF ELECTRICAL ENGINEERING PHENOMENA.

The Ohio State University, Ph.D., 1966
Engineering, electrical

THE USE OF COMPUTER-GENERATED MOTION PICTURES IN THE ANALYSIS OF ELECTRICAL ENGINEERING PHENOMENA

DISSERTATION

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Brenton Robert Groves, B.S.E.E., M.S.E., P.E.

* * * * * * *

The Ohio State University
1966

Approved by

[illegible]

Adviser
Department of Electrical Engineering

ACKNOWLEDGMENTS

I wish to acknowledge the helpful assistance and suggestions given me by my adviser, Professor John D. Cowan, Jr., without whose help this dissertation would not have been possible. I also wish to thank David M. Shupe and Edward M. Kennaugh for their help in developing the basic idea of the spheroidal radiator. The Numerical Computation Laboratory of The Ohio State University granted much of the necessary computer time required. Albert Farha of the Columbus Division of North American Aviation supplied the FORTRAN subroutines for the SC 4020 that were used as a model for development of the SCATRAN programs and also allowed time on North American's SC 4020 to check out the many subroutines that were developed for OSU's computer-operating system. Above all I wish to thank my wife, Lois Ann Groves, for her support, encouragement, and typing of this dissertation.

The work reported in this dissertation was supported in part by the National Science Foundation through the National Committee for Electrical Engineering Films of Educational Services, Incorporated of Watertown, Mass. whose technical services were also appreciated.

VITA

May 28, 1930	Born - Chicago, Illinois
June 1952	B.S.E.E., Massachusetts Institute of Technology, Cambridge, Mass.
1952 - 1953	Ordnance Engineer, Boston Naval Shipyard, Boston, Mass.
1953 - 1957	Technical Representative, Link Aviation, Inc., Binghamton, N. Y.
1957 - 1960	System Design Supervisor, Link Division, General Precision Corp., Binghamton, N. Y.
1960 - 1963	Engineering Specialist, Goodyear Aerospace Corp., Akron, Ohio
Feb. 1963	Completed requirements for Professional Registration, State of Ohio
May 1963	Elected to Senior Membership in the Institute for Electrical and Electronic Engineers
June 1963	M.S.E., University of Akron, Akron, Ohio
1963 - 1966	Research Associate, Department of Electrical Engineering, Ohio State University, Columbus, Ohio
May 1966	Elected to Chapter Membership in The Society of Sigma Xi, Ohio State Chapter

FIELDS OF STUDY

Major Field: Electrical Engineering

Studies in Electrical Engineering

Professors	John D. Cowan, Jr.
	William C. Davis
	Claude E. Warren
	Frank C. Weimer

Studies in Electromagnetics (Physics)

Professor	Albert L. Prebus

Studies in Applied Mathematics

Professor	Henry D. Colson

CONTENTS

CONTENTS (Continued)

CONTENTS (Continued)

LIST OF TABLES

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Continued)

CHAPTER I

INTRODUCTION

The purpose of this work was to see if computer-generated motion pictures could be applied to the analysis of electrical engineering phenomena. This subject was broken down into four parts which are described later in this chapter and which comprise Chapter II through Chapter V.

What are computer-generated motion pictures? They are a small branch of computer-graphics, one of the fastest growing fields in electrical engineering today. Computer-graphics, or the science of producing pictures with computers, is part of the attempt to cope with the most serious problem affecting these machines today. As computers become faster and larger, communication with them on the part of the human being becomes more difficult. Not only do the rules for programming inputs become more involved, but the mass of output tends to overwhelm the researchers when they try to interpret what they have created.

The solution to this problem is to use a graph to show the various relationships that are hidden in the mass of figures that constitute the average program, and computer-graphics uses the machine to draw these graphs directly instead of plotting them by hand.

The usual output device used for this purpose is the microfilm plotter. It consists of a computer-controlled cathode ray tube and a camera that can record the face of the tube on film. The computer causes lines or characters to be plotted at any desired location on the tube. When a complete picture has been drawn, the film is advanced to a new frame and the next picture is constructed.

Generally, each frame is independent, but when a computer-generated movie is produced, the information plotted in each frame is incremented from the one before it, so that the final result when the film is run through a projector is an animated motion picture. These movies differ from a regular one in one respect. A regular movie is exposed in "real-time" and run at the same rate to give the proper motion, while computer-generated movies are created frame by frame. This means that the computer can take as long as it wants to calculate each picture in a sequence, and it is the calculated increment in position that causes the motion.

The first problem considered in this work was how computer-generated motion pictures could be applied to increasing understanding in electrical engineering. Chapter II discusses how the advantages of the technical training film could be combined with the computer's ability to handle large masses of data in order to illustrate physical events that are usually expressed mathematically. Many of the subjects in electrical engineering are of this type and the equations explaining what is happening are usually accompanied by a still drawing. The tremendous advantages of the computer-generated motion picture are that the motion in either time or space can be shown and the limitations of the amount of data that can be handled are removed. The exact solution can be plotted and described as easily as the approximate one. The main requirements are that the subject be mathematically complex and that motion be an essential component to understanding.

A number of different topics are discussed and it was felt that electromagnetic fields are undoubtedly one of the best subjects that can be found to which these techniques can be applied. The degree of abstraction of the mathematical concepts contained in Maxwell's equations, for example, make it a difficult field to understand from the student's point of view. When solutions are known to the field equations in a certain case, the complex equations and mass of numerical output detract from complete understanding of the problem. This is particularly true when other than steady-state fields are considered and it is important to see time as an element of understanding of the problem.

Therefore, a number of topics in electromagnetic fields are discussed with regards to how they might be helped by the production of a computer-generated movie.

The next problem encountered was how to give Ohio State

University the capability to make movies using their IBM 7094 computer. It was decided fairly early to center this effort around the SC 4020 microfilm plotter made by the Stromberg-Carlson Corporation. This machine is an off-line plotter that uses magnetic tape to drive a cathode ray tube that is matched to an automatic 35 mm camera. It was chosen because it was almost the only machine available on the market at the time this work was begun and because a body of subroutines was available from North American Aviation in Columbus, Ohio for use on their SC 4020. Although OSU's computer center is not equipped with an SC 4020, there are enough of them at other colleges and in industry that time can be obtained fairly easily, either for research purposes or by contract. Also, the machines that are coming on the market at the present time generally follow the same organization as the SC 4020 so that when OSU obtains a microfilm plotter, the capability demonstrated in this work can be transferred to the new machine with very little effort.

There were, however, a number of problems that had to be solved before a movie could be produced. These included determining the capabilities of the SC 4020, organizing the available software to find out what was useful and what was not, overcoming a computer language difficulty between OSU and the outside world, and finally, writing subroutines that would permit any programmer at OSU to produce movies of electrical engineering phenomena merely by adding the right subroutine calls to a main program describing this phenomena.

In order to solve these problems, the subject of computing software had to be considered in relation to the requirements of the SC 4020. A general description of how these problems were solved is given in Chapter III. This turned out to be a very complex subject with a great many details that would be of interest only to a programmer who was attempting to actually use the subroutines developed in the course of this work to produce a movie. Therefore, the details are described in the first three appendixes of this work covering the differences between FORTRAN and SCATRAN computing languages, the SCATRAN operating system, and the FORTRAN system respectively. Chapter III ends with an example of how a programmer would use the system to produce a scene in a movie from a very simple program.

Chapter IV describes the details of a motion picture that was produced using the technique developed in the preceding chapter. A number of different subjects were considered from the field of electromagnetics, such as plane waves and wave guides, in addition to the field pattern that was finally chosen, but it was felt that a coherent movie covering one subject thoroughly would be more interesting to the viewer than a number of bits and pieces. Therefore, a script outline was written and from this a three-minute motion picture entitled "The Resonant Spheroidal Radiator" was produced. The movie pictures the antenna that was chosen, and then demonstrates several facets of the field produced by this antenna.

This chapter covers the mathematical theory behind the programming, describes each scene and discusses a number of problems that were encountered and solved. The biggest problem was what to show the viewer in order to get the point across. Therefore, the description of each scene also attempts to outline its purpose. The final effect can only be judged by seeing the movie itself, rather than the illustrations in this work.

For the sake of completeness, the program listings for the movie are included in Appendix D.

In the course of making the movie example, it became apparent that although the availability of the SC 4020 dictated the use of this device, the use of a computer-driven plotter expressly designed for producing computer-generated movies would ease the task of the programmer. Therefore, Chapter V examines the plotting devices that are now on the market and also makes a survey of the various types of computer-graphic equipment now in the laboratory stage to see what trends might be of help in this area. The main problem would seem to be one of cost. The hardware and programming techniques are available today so that the actual programming and computing of a movie could be almost completely automated, but so far they have not been put together.

The chapter concludes with some suggestions that could be considered should the effort be made to reduce movie-making to where the human being can communicate with the machine on his terms rather than on the computer's. This would free the programmer to apply himself completely to the ideas behind a particular movie instead of getting bogged down in the computer programming details.

CHAPTER II

COMPUTER GRAPHICS IN ELECTRICAL ENGINEERING

Training films are considered to be very effective in the educational process, but they are usually used only to extend the teacher's effectiveness. This is especially true in such fields as electrical engineering where the subject matter is mainly expressed in mathematical terms. One solution to this problem is the use of computer-graphics to make movies of mathematical representations. Several subjects in the field of electromagnetics are discussed.

In the last twenty years, movies have come into their own as a training device. When one looks at the catalogs of available films, the number seems limitless. A representative list put out by the Instrument Society of America has almost two hundred titles that would interest engineering students ranging from "Accelerometer Calibration" to "X-ray Crystallography".[1] These films come from universities, industrial corporations, laboratories, advertising agencies, and professional educational film companies. When all the sources are added up, the sheer volumne indicates how helpful films are considered to be in the educational process.

[1] ______, Selected Films on Instrumentation (Pittsburgh: Instrument Society of America, 1956).

However, ninety-nine per cent of these movies have one thing in common. The film is used only to extend the range of the teacher. They save a great amount of time and effort, particularly when an experiment needs large and complicated equipment, but in general movies are used only to bring the instructor to the student and not in their own right to increase the student's knowledge and understanding.

Taking all of the available films and determining which ones would be suitable for graduate instruction, movie after movie is dropped until very little remains. This is particularly noticeable in research, with the notable exceptions of time lapse photography and Harold E. Edgerton's high-speed strobe motion pictures. It is obvious that the motion picture camera is a valuable instrument, especially where the motion is impossible to observe, as in the above cases, but until now it could only be applied to the real world. The camera saw only the action, and philosophical approaches were handled by letting the expert lecture to the camera instead of the class. The only advantage was that the audience could be far greater and a good lecture could be given many times.

The main difficulty seems to be that although education experts are unanimously agreed that training films are effective teaching aids, no one seems to know why or even how effective they are.

An indication of this lack of knowledge is shown by a current study being financed by Britannica Films and Bell and Howell, and conducted by the School of Education at Ohio State University. Two schools, one in a poor area of Washington, D. C., and another in Shaker Heights, Ohio, have projectors in every classroom and have been given a complete library of 2000 movies and filmstrips. The hope of the study is to determine whether the quality of education can be raised by unlimited use of the educational movie.[2]

The pat answer is that movies are effective because if one picture is worth a thousand words, then a thousand pictures are worth that much more. However, before one can really answer the question, "How effective are movies in education?", one must answer the question, "How do people learn?". In the education library at OSU, there are over 500 titles under the heading of "Psychology of Learning" and yet as one reads through this massive body of literature, the impression is given that we really know very little about the learning process. Even the definition of a good teacher seems to cause lots of difficulty, although it is easy to pick out such individuals on the campus. The learning and teaching process does not seem to be one that can be specified on a rating form.

Electrical engineering as a field suffers particularly

---

[2]The National Observer, September 11, 1965, p. 15.

because most of the communication is in the form of mathematics. The information to be presented is contained in the form of symbols and conventions that must be understood before the information can be transmitted. As the user of mathematical tools gains more familiarity with his subject, he finds it harder to communicate with those who do not have the same depth of background that he has. Tensor analysis is a particularly good example of this in that by the time the notation is thoroughly understood, many problems are easily solved, but the results can be explained only to someone who also knows the notation.

This already complex problem is intensified when one considers the change brought about by the introduction of the digital computer into the educational process. It is becoming a dominant force in shaping the trends of the future, particularly in the sciences.

The digital computer represents one of the great revolutions of our time. It has already affected our daily life to a far greater extent than most people realize and the changes that it is responsible for are occurring at a faster and faster rate as time passes. Nowhere have more changes occurred than in the academic and research fields so that now one man, aided by the digital computer, can find solutions to problems that would have taken a lifetime of effort without it.

At the present time, an undergraduate student can easily solve problems assigned for homework in the course of a week that could not be solved by hand over several months. The solutions are more likely to be correct because of the lack of human error, and the number of calculations can be far greater. This is because the machine can execute a program a hundred times as easily as once with the same amount of effort on the part of the human programmer. In fact, these features of the digital computer can also lead to difficulties that are unique to this new age. The use of the computer has a tendency to guide research in the directions where computer solutions are the easiest to obtain and therefore some promising avenues of exploration are not pursued. Along with this, the output of a computer is tremendous and because anything as expensive as this has to be kept busy at all times, we may end up buried in our own paperwork with pages of figures that have no meaning because there is no time to have a human being analyze the forest for the trees.

Computers are becoming bigger and faster at an increasing rate every year. However, the advantages in this increase cannot be utilized to their fullest extent because one area has lagged far behind the rest in computer technology. The input-output area where the human must communicate with the machine is still approximately the same as it was when digital computers were invented. At the start, the already invented teletypewriter was used to print out numbers stored electrically in the machine. Almost the same method is used today with an increase of about an order of magnitude in the speed at which the numbers can be printed. The storing of numbers in machine codes on magnetic tape has helped in increasing the rate of flow from the computer, but the information stored is in a form that is inaccessible to the user until the tape is printed out on paper.

If the statistician or engineer wishes to make sense out of a jumble of figures, he draws a graph. It seems to have occurred to people only fairly recently that this is tedious and time-consuming and if the computer could take over the work of drawing graphs rather than presenting numbers, a dent could be made in the vast output that pours from these machines every day. The area of computer graphics has just started to move. At the present time, one cannot pick up a technical journal in this field without seeing at least one article on the subject. And yet very little hardware exists for putting these ideas into practice. Although many analog X-Y plotters have been adapted to accept computer output and translate it into visual material, these plotters are even slower than the printer. What is needed is a device that can work at computer speeds and at the same time reduce a mass of numerical data into an easily understood picture.

With the advent of computer controlled display systems that can be recorded on film and combined with high speed plotting devices, the result is that time can be added to the parameters of space already available. Better understanding will result even for those who already have a thorough knowledge of the problem when a motion picture can be made easily of a process that could be described only mathematically.

There are many phenomena particularly in the field of electrical engineering which, although they are physical events, can only be explained on the basis of mathematical theories. As "The Ad Hoc Committee for Demonstration Films in Electrical Engineering" stated:

> Electrical engineering teachers have a peculiarly difficult task because electrical phenomena arises from invisible submicroscopic charged particles that can interact through empty space, in contrast with mechanical, chemical, and other physical phenomena normally involving visible matter undergoing changes of state......Films which show well-designed demonstrations of physical phenomena and relate the phenomena to the applicable electromagnetic field or circuit theory will enhance and broaden the comprehension of those who are inclined to view the world as a geometric structure covered by a set of equations. Such film can point up where simplifying or restrictive mathematical assumptions must be made in order to have a tractible mathematical field or circuit theory. The emphasis here should be on the natural phenomena and the visual images of the abstract concepts of theory which interpret it.....[3]

[3] J.G.Brainerd et al., "Summary Report", Conference on Motion Pictures for Instruction in Electrical Engineering, University of Illinois, September 23-25, 1962, pg. A-5.

It is in this area that computer graphic movies can serve as an instrument for explaining and understanding electrical engineering phenomena so efficiently that every university will have its own means of producing these movies. The users will wonder how they ever got along without them in the same way that it would now be impossible to get along without the computer itself.

The number of subjects in electrical engineering to which this equipment can be applied is very large. The main requirements are that the subject can be expressed mathematically and the element of motion has to be important enough so that a movie can show concepts that would be missed in ordinary text book pictures.

Among the many topics that could be investigated even with the limited equipment at hand are convolution, root locus methods, energy levels in quantum mechanics, and Fourier analysis. But the overwhelming choice would seem to be electromagnetic fields.

Maxwell's equations consist of a set of coupled first-order partial differential equations relating the various components of electrical and magnetic fields. They can be used to explain all electrical phenomena from antennas and wave guides to circuit theory and Ohm's Law. Since Maxwell's equations have both time and position as basic parameters and represent a very abstract concept of physical events that are hard to illustrate, they are the ideal subject to use to demonstrate the ideas expressed in this work.

Transmission lines

The equations for propagation of waves on a transmission line are well known.[4] However, in the classroom, the student is generally concerned with the behavior of a transmission line when the voltage and current present are a steadily applied single frequency. Transient phenomena is usually considered only for an open or short circuited lossless line.

The use of computer generated motion pictures to show the signals on a transmission line opens up a whole new region where the student can accurately see what happens to a pulse or step applied to the line under various conditions of terminal impedance and different parameters of the transmission line itself. For example, consider a pulse traveling doen a line terminated by an inductance. When the pulse first hits the inductance, it acts as an open circuit because no current can flow through the coil. Therefore, the pulse is reflected, but, as time passes, the inductance changes to a short circuit because of the current it is drawing. The problem can be further complicated by having

[4] H. H. Skilling, Electrical Transmission Lines (New York: McGraw-Hill, Inc., 1951).

a termination that is part resistance and part inductance or capacitance and by having the pulse reflected at both ends of the line.

The transient behavior of a transmission line can become extremely complicated if it is not distortionless, and as a consequence, most texts treat only that class of line. The differential equations of a line with distortion can be solved,[5] but the solutions are usually in the form of infinite series that make it hard to understand exactly what is happening to the shape of a wave traveling down the line. Here is an ideal subject for a computer-generated movie. The computer can be used to solve the differential equations in series form, with enough terms to give good answers, at different points on the line at different times and then the movie producing subroutines can complete the job by showing what is actually taking place.

Plane electromagnetic waves

The study of time varing electromagnetic fields usually starts with the plane wave. The E and H fields are derived from Maxwell's equations, and a drawing (Figure 1) attempts to show the relationship between these two vector fields.

---

[5] V. Bush, Operational Circuit Analysis (New York: John Wiley and Sons, Inc., 1937).

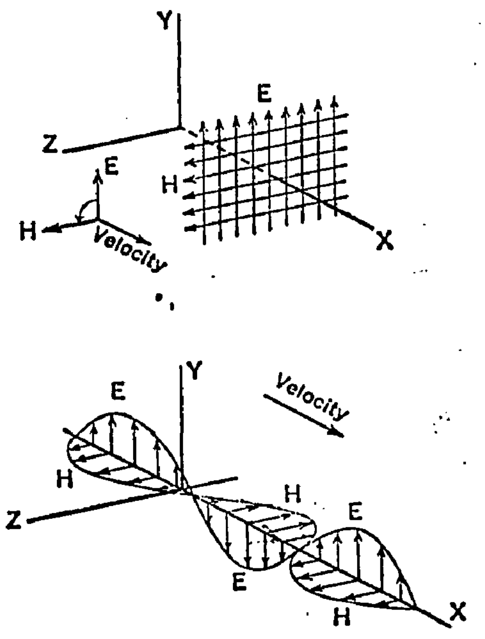


FIGURE 1

STATIC DRAWING OF A PLANE WAVE

As one author puts it, when referring to this drawing:

> To show the wave as it exists in three-dimensional space would require a combination of these diagrams, with the distribution of the electric and magnetic fields shown throughout many planes. Even this would be only an instantaneous picture of the wave and would fail to show its motion; for a complete picture the imagination must be called upon to visualize what cannot be drawn on paper.[6]

This sums up very neatly the usefulness of computer generated motion pictures. The limitations of drawings on paper are released and what can be shown is limited only by the programmer's ingenuity.

[6] H. H. Skilling, Fundamentals of Electrical Waves (2d. ed. rev.: New York: John Wiley and Sons, Inc., 1948),P.124.

An interesting movie could be produced covering the following subjects as a unified topic: plane waves radiating in space, the concept of parallel and perpendicular polarization, waves hitting an infinite dielectric interface, Brewster's angle and total internal reflection, waves propagating through a dielectric slab, wave propagations in a semi-conductor, and finally, wave behavior at the surface of a conductor.

A program has already been written that calculates the field of a perpendicularly polarized plane wave hitting an infinite dielectric interface as an example of what can be done. The standing wave due to reflection and the waves propagating in the dielectric are shown by means of the stream function. Lines of constant potential in the electric field are calculated, and the final result can show the variations in the field due to changing both the angle of incidence and the dielectric constant.

Wave guides

Wave guides are another area where the drawings showing the electromagnetic field in the guide have become stereotyped. Generally, a cross section and a longitudinal section are shown with solid lines for the electric field and dashed lines for the magnetic field. Almost any reference on the subject of wave guides will give an example of these drawings.[7]

Using a rectangular guide as an example, the requirements that there be no tangential electrical field at the sides of the conducting wave guides, in addition to satisfying Maxwell's equation, causes the field to assume a certain form. The drawings show this form but do not explain why this should be so.

The movie-making technique described in this work can be used to great advantage. A number of things could be shown that could not be demonstrated in any other way.

The requirements to be met in the wave guide can be filled by showing two plane waves traveling in the guide at the same time following a zigzag path with multiple reflection from the walls of the guide. These two waves combine in such a way to give a zero electric field at the walls at all times while propagating down the guide.

This effect could be explained in a movie by showing a cross section of the guide. The differences between wave and group velocity could be shown due to the zigzag path and the dimensions of the guide could be changed to show the effect of cut off at a particular frequency.

The ultimate example would be a cut away perspective

[7]T. Moreno, Microwave Transmission Data (New York: Dover Publications, Inc., 1958) pp. 113-119.

view of the wave guide in which the components of the wave could be shown one at a time and then the combined effect of all of them. Occasionally, drawings of this type appear in texts, but without motion, much of the effect is lost.[8] A perspective view would be much easier to program if a subroutine were written that would take the three x, y, and z coordinates and convert them into the two-dimensional x-y coordinates of the perspective view. Such a subroutine would not be difficult to write and it would allow the programmer to think and calculate in the three-dimensions of the wave guide. One of the features of this subroutine could be to suppress any points or lines outside of the cut out in the wave guide.

The extension of these ideas to higher order modes and wave guides of different shapes is obvious. Another application in research could be the determination of fields in a wave guide that had minor distortions of the shape, such as a stud inserted in one side that would alter the field. Of course, this method is limited by the fact that the mathematics of such a field would have to be known before the field could be computed, but it is possible that this could be handled by a perturbation method where the basic field would be corrected by the distortion before movie subroutines were called.

[8]Skilling, loc. cit.

Antennas and radiators

Antennas would seem to be one of the best candidates for a movie of the field pattern in so far as their mathematical properties are known. A complete description of one such field is given in the spheroidal radiator example which was produced as a demonstration of a computer generated movie. The discussion will be limited in this section and the reader is referred to Chapter IV for further details.

In general, the field of a radiator can be divided into three zones: the surface of the antenna, the near or induction field, and the far or radiated field. An interesting movie could be made showing the difference between these two fields for the same antenna. The scale would be different because the induction field is only up to about a wave length away from the radiator, while the far field exists to infinity. In the example of Chapter IV, only the far field is shown because at resonance all of the power is radiated and there is no inductive near field.

The solution to a radiated field problem that is going to be used as a subject can be handled either as a boundary value solution to Maxwell's equations, as in the example, or as a diffraction and scattering of optics. In either case, once the decision has been made as to how the field is going to be represented, the movie subroutine can be used to calculate the picture on the screen.

Only one area in electrical engineering has been discussed, and that only briefly. Still, it can be seen that the number of areas even in this one field is great and a lot of work is waiting for someone who can translate mathematical calculations into motion pictures, especially for the benefit of the electrical engineering student. However, as a glance into Appendix D covering the program for the spheroidal radiator example will show, there is a large amount of work between selecting a subject and having the film in hand. The ideal researcher would be one who combines a thorough knowledge of the mathematics of the chosen subject with an excellent grasp of computer programming, and such people are hard to find.

CHAPTER III
THE SC 4020 MICROFILM PLOTTER

Summary

The SC 4020 Microfilm Plotter was the main device used to produce the motion pictures described in the dissertation. This chapter describes the construction and operation of this plotter and explains that there are eighteen command words that can control the available operations when they appear on a magnetic tape used to feed data to the plotter.

The concepts of software and programming languages are briefly discussed and the importance of software in expanding the above eighteen codes is explained. The languages in use at OSU are defined and the steps taken to overcome a conflict between the original SC 4020 programming system and the OSU operating system are outlined. Finally, the place of the SC 4020 plotter in the overall system is explained and the subroutines available to the programmer for making movies are given. An example of how these calls would be added to a main program defining a movie scene is given so that the actual mechanics of placing the desired SC 4020 command words on tape is accomplished with very little effort on the part of the programmer.

The actual details of the SC 4020 operating system are expanded at great length in Appendix B for those who eould be interested in actually using the subroutines developed for making movies. This chapter covers only the general outlines of the work done in programming the SC 4020.

SC 4020 Microfilm Plotter description

Most of the visual materials used as examples for this dissertation were produced with an SC 4020 microfilm computer recorder manufactured by the Stromberg-Carlson Corporation. The SC 4020 is representative of a number of devices that are used to translate digital commands into visual images, such as graphs or printing. It is important to understand that this device is not an analog machine where the characteristics of the input signal are related directly to the output. In a television receiver, the magnitude of the video signal is directly proportional to the degree of brightness seen on the screen. In contrast to this, a digital machine such as the SC 4020 uses the numerical information contained in its input commands as the visual material to be produced. The input signal can be represented as a number which, if it is within the set representing the machine's operating instructions, tells the machine to perform a particular operation. For example, a single number will instruct the SC 4020 to plot a dot at a certain location on its screen.

Because the SC 4020 was the prime research tool used in this study, it is desirable to go into some detail as to the construction and operation of this device.

The heart of the SC 4020 is a seven-inch cathode ray tube of a special design called a CHARACTRON. This tube consists of an electron gun, a set of deflection plates, a matrix consisting of a metal plate with sixty-four alpha-numeric characters cut in it, a second set of deflection plates, and finally a phosphor screen for viewing the image.

The first set of deflection plates locate the electron beam so that any desired character cut in the matrix can be illuminated. As the electron beam passes through the small hole in the shape of a particular character, it assumes the shape of the hole. For this reason, the CHARACTRON tube is often known as a beam shaping cathode ray tube. The second set of deflection plates positions the shaped electron beam on the face of the tube where it can be photographed. In general, two options are available for the output of the SC 4020, either a nine by nine inch hardcopy is produced on oscillograph paper or a thirty-five millimeter camera is used to produce microfilm. This camera is special in that it has to be fitted to the face of the cathode ray tube and controls must be provided to operate the shutter and to advance the film frame by frame upon receipt of an electrical command.

So far, the cathode ray tube can be considered as an analog device much like a television tube. However, it should be noted that the first set of deflection plates must be energized in such a manner that only sixty-four precise locations corresponding to the sixty-four holes in the matrix are illuminated by the beam. This converts the CHARACTRON tube to a discrete device in which a digital signal consisting of six bits processed through the proper logic circuitry can select any one of sixty-four different characters. Fourty-eight of the sixty-four available characters correspond to the standard IBM BCD alphanumeric code.[1] In each case, the octal number used by IBM for each character in the fourty-eight character set is identical with the one used by the SC 4020 in the sixty-four character set. There are sixteen codes not used in the IBM set that are used by the SC 4020. They are given in Table 1.

The second set of deflection plates determine the position of the beam on the face of the tube. These can be energized in both a discrete or continuous manner. In order to translate the location of the beam into a finite number of digital bits, the face of the tube is divided into a matrix of 1024 cells down by 1024 cells across. Each address of an X location (across) or Y location (down) can be specified by ten bits and therefore with twenty bits of a thirty-

[1] ______, _FORTRAN II Assembly Program Manual, File No. 7090-21_ (New York: IBM Corporation, 1963) p. 68.

TABLE 1

CHARACTERS AVAILABLE IN ADDITION
TO THE STANDARD IBM BCD SET

OCTAL CODE	SYMBOL	NAME
12	∂	Partial Derivative
14	"	Quote
16	δ	Delta
17	α	Alpha
32	π	Pi
35	β	Beta
36	$\pm$	Plus or Minus
37	?	Question Mark
52	•	Plotting Dot
55	γ	Gamma
56	$\sim$	Similar (Tilde)
57	d	Differential
72	$\circ$	Degree
75	$\int$	Integral Sign
76	Σ	Sigma
77	$\square$	Alignment Frame

six bit control word 1,048,576 different locations on the face of the tube can be specified. The addition of two more bits used for this purpose in the control word would multiply the number of addressable locations by four, but this is not necessary nor desirable.

The size of the face of the tube compared with the size of the beam will show that all of the available resolution is gained with the 1024 divisions. Two dots plotted in adjacent cells will overlap, two plotted one cell apart will just touch. In order to plot discrete vectors or points, they should be at least three cells apart. The second set of deflection plates can also be used in a continuous manner to plot either vectors or axes. A discrete location signal or address will determine the start of a vector and the beam will be swept in the desired direction to form a line. Vectors are limited in length but axes can be swept for any length.

Associated with the CHARACTRON shaped beam tube is a certain amount of logical hardware that is used to translate the input commands into the necessary voltage signals used by this tube. The complete SC 4020 package also includes an input buffer for matching the data transmission rates to the logical hardware. Although this buffer is generally driven by a magnetic tape unit, it is perfectly possible to use the SC 4020 on-line with a large computer. Strictly speaking, the SC 4020 is not a computer but is only a peripheral device which produces data in response to input commands. A computer has the power to alter its operation based on past responses to its program; the SC 4020 does not. It can only take each command word and perform the operation specified by that command.

SC 4020 hardware commands

It is important to understand that in the discussion which follows the commands referred to are hardware commands and not part of the SC 4020 software. A hardware command can be equated to the throwing of a switch to turn on a light, while a software program can be equated to a routine that calculates when and how the switch is to be thrown. There are two reasons why the programmer should be able to utilize the machine commands even though he will not refer to them again once all the required subroutines are operational. The first is that the available SC 4020 subroutines were developed by North American Aviation, Inc. to meet a specific need and unfortunately the production of a movie was not one of the desired goals. Although liberal use was made of these subroutines as a model, many new subroutines had to be written. In order to write subroutines for the SC 4020, it is essential to understand the capabilities of the machine, and this can best be obtained by analyzing the machine commands available to the programmer. The second reason is that as programs are written to produce scenes in a movie it is necessary to have a convenient check on the output. The best way is to make a tape of a particular scene and then run it on an SC 4020. However, this may not be convenient. A subroutine is available that prints out the control words as data from the IBM 7094 computer and knowledge of the codes used by the SC 4020 will enable the programmer to check his work directly.

The SC 4020 uses eighteen different operation codes which can be conveniently divided into four categories: performance, vector, plotting, and typing commands. Each command occupies a thirty-six bit word even if not all the bits are used for a particular function. A thirty-six bit word can be represented in octal notation by twelve digits with three bits to a digit.

For example, the command word to advance the frame of either the film or hardcopy can be expressed in octal notation as:

460000000000

These eighteen codes are expressed in some detail in Appendix B of this work. Also given in this appendix are two examples, the output of two different tests, showing what the SC 4020 command words look like when they are printed out by the computer.

It is the software programs associated with the SC 4020 that make it possible for the programmer to produce movies. Since the command words are in a one to one correspondence with machine operations and some movie frames contain over sixty thousand points to be plotted, it would be impossible for the programmer to use these hardware commands directly even if he had a method of determining what numbers to use in the control word from a series of points expressed in X and Y coordinates. With the software, the single call

```
CALL SUBROUTINE()=APLOTV.(N,X,Y)-
```

will do all of the work necessary to place N points at different X and Y locations of a programmer's coordinate system on magnetic tape in the form of command words matched to the SC 4020's raster grid. The only limit is the size of the computer's memory.

Purpose of software

At this point, it is necessary to digress somewhat and discuss the general concepts of software as opposed to hardware. A data processing glossary definition of these two terms is:

> Hardware - the physical equipment or devices forming a computer and peripheral equipment. Contrasted with (software).
>
> Software - the totality of programs and routines used to extend the capabilities, such as compilers, assemblers, routines, and subroutines. Contrasted with (hardware).[2]

[2] H. E. Kosters _et al._, _Automatic Data Processing Glossary_ (Washington: U. S. Government Printing Office, 1962).

Hardware has long been with us. It is all that is really necessary in order to compute with a machine as anyone can attest who has used a desk calculator. Since the birth of the digital computer in 1944 (a relay machine called the Mark I), a great deal of effort has gone into making computers bigger, better, faster, and as a consequence, far more complicated.

The term software has a relatively more recent origin and the importance of software is newer still. For several years, many computer experts were against any form of what has come to be called software. To quote one such author as late as 1958:

> There are, on the other hand, people who will have nothing to do with any form of conversion. (A very crude form of software - BRG.) They believe that a programmer does best to write his orders in a form as near as possible to that which they will take inside the machine - in the case of a binary machine this means the use of the scale of 8, 16, or 32 - the input operation consisting of a simple transfer of digits to the store to form words . . . the establishment of such a scheme absorbs a great deal of programming effort and they suggest that this effort could better be directed towards solving problems. No sooner, they say, does someone get good at programming than he stops doing real programming and starts making programs to make other programs.[3]

[3]M. V. Wilkes, Automatic Digital Computers (New York: John Wiley & Sons, Inc., 1958) p. 273.

To show how far we have come since 1958, in 1965 IBM spent more money on having people "make programs to make other programs" than they did to manufacture all of the hardware that was sold in that year. One of the reasons that IBM is the giant of the computing world is that their software has generally been ahead of that of their competitors.

It is software that makes the digital computer such a powerful tool. The key word in the definition of software is the word "extend." Without software, the digital computer is nothing more than a glorified adding machine. With software, each person's efforts add to the foundations of already solved problems and techniques so that the final result is a complex structure that no one person can hope to understand completely. The total of all software for a particular computer installation is known as the operating system. The tremendous extension in the computer's capabilities comes about by the fact that it is not necessary to understand how the operating system works. It is only necessary to understand how to enter the system and record the desired results. The software keeps track of the literally thousands of details that are required to process the program in addition to calculating the results of a problem from the input data. For example, when the programmer specifies that a number is to be used, the software not only has to convert that number into a binary number and store it in the computer's memory, but in addition, it has to give the number a name and then record where the number was stored so that it can be found again. This of course is a trivial example, but when it is multiplied a million times, it gives an idea of the magnitude of the operating system. The main job of software is to shift as much of the work in programming as possible to the machine and make it as easy as possible for the programmer to do his work. This requires that the machine take the programmer's language and convert it into the ones and zeros that are the only language the machine understands. The ultimate operating system would allow a person to communicate with the computer as he would with another person, but this seems far in the future.

It was stated above that the main purpose of software was to shift the burden of necessary details to the machine as far as possible so that the programmer would not have to know how they were done. Unfortunately, when writing an extension of an operating system, the exact reverse is true. It is a requirement that the software programmer know as much as possible about the system that he is going to extend, because his subroutines must dovetail with all the other programs that have already been written and that are going to be used to support the extension. For example, the proper calls must be made to the already existing input-output subroutines.

The following discussion in this chapter covers a body of software that was developed to write magnetic tapes for the SC 4020 and BL 120 microfilm plotters. These subroutines were written to operate under the Ohio State University Numerical Computation Laboratory operating system and in their entirety they are suitable for running only at that location. However, the foundation for these subroutines comes from a set written by North American Aviation for the IBM 7090 series computer. Therefore, they could be adapted to any system using this series of computers with a smaller amount of effort than would be required to write a new set of subroutines for this purpose. The tapes that are produced can be run on any SC 4020 available to the programmer subject to the input buffer limitations mentioned in Appendix B under program running operations.

Software languages

In the discussion on software which follows, the word "language" can have several different, although precise, meanings. This is because a "language" as applied to computers can operate on one of several different levels depending on how close the programmer wishes to be to the machine. In general, the "language" is the system of rules for communicating with the computer and for combining various statements to build up a program that can be executed. The two main levels of "language" are object language and source language. The source language is the language that the programmer uses to prepare the program. The object language is the output language of some routine after the machine has operated in the source language. Generally, the object language is very close to the language that the machine uses in actually solving the problem (usually a series of binary numbers), but it is entirely possible to have the object program from one step of processing become the source program for another step. Unfortunately for programmers, as for the world, one language is not enough. Someday there may be a universal computing language (a new language called PL-1 is said by its writers to be all things to all men), but at the present time there are several hundred well known computer languages. If this were not enough, every operating system grows away from its original source so that there are thousands of different varieties, all called by the same name. Just as an example, at a computer conference, tha following languages were discussed:

ALGOL	FORTRAN	MIMIC
APT	GPSS	PERT
AUTOCODE	HELP	PLACE
COBOL	ICES	QUICKTRAN
COGO	INTERCOM	SAINT
CPM	JOSS	SCATRAN
DYSAC	MATHLAB	SIMSCRIPT
FAP	MIDAS	SNOWBOL

However, this is not the place to go into great detail on computer language. There are many references that cover areas in this field but none that cover it completely or adequately.[4]

Before discussing the language used at OSU, it is necessary to explain the difference between an assembler and a compiler. An assembler operates on an object program and changes it into a machine program. The programmer, in writing an object program, usually follows the general outlines of the machine operation, but he specifies the various commands and variables by means of a relocatable program: i.e., one in which a piece of data is not assigned a definite place but is assigned only in relation to all the other pieces of data in the program. Because each piece of information is given a name, this is often called a symbolic program. The assembler is a software program added to the programs placed in the computer by the programmer that makes certain that everything has a place. The assembly program is <u>not</u> part of the program that is used to compute the desired output and is used only the first time a program is run. As a general rule, the object program is assembled by the assembler and then punched on cards in machine lan-

[4]Bernard A. Galler, <u>The Language of Computers</u> (New York: McGraw-Hill, 1962).

Ivan Flores, <u>Computer Software</u> (Englewood Cliffs, N. J.: Prentice Hall, 1965).

guage. This is known as an object deck and is the most basic form used. When the object deck is placed in the machine, an additional small software program called the "loader" gives each machine instruction the right address so that it can be found again. It also checks all subroutine names called in each program. If any name is missing, this subroutine checks the "library" of subroutines that are on call in the operating system. The library consists of subroutines that are so popular that it would waste time if the programmer had to write his own each time he wanted to use one. A good example of this is the square root subroutine (SQRT) which is a subroutine in its own right and not a machine function like addition.

All of the above remarks apply to a compiler as well as an assembler, but a compiler resembles an assembler as a Model A Ford resembles this year's Cadillac. The machine language is the Model T. At the start (in 1950), all programs were written in machine instruction with each operation located in the machine by the programmer as he came to it. The assembler was born when the machine operations were coded but the written program was still tied to a particular machine. The great advantage in using a compiler is that not only does the programmer not have to know how a particular computer works, but he does not even have to know what computer is going to be used to run his program.

The programmer writes in a source language that is procedure oriented rather than in an object language that is machine oriented. Each machine will have a different compiler that translates the source program into an object program for that particular machine. The compiler is an extremely complicated piece of software because it effectively replaces the skilled programmer by the computer. In fact, when IBM began to write the first compiler, many people felt that it would prove to be an impossible job. With a good source language, a person who has never run a problem on a computer can do so after several hours of instruction; after several weeks, they are able to routinely use the computer in their work. A good source language resembles the mathematics that the user is accustomed to and all the rules are in a book. One can construct a program by looking up the various pieces. The most difficult thing about a source language is to have it flexible enough to be really useful and at the same time have few enough rules that they can be easily understood. As the two goals of flexibility and simplicity are met, the compiler grows until it is the dominant part of the operating system. At the present time, it is almost impossible to sell a computer without compilers operating in the more popular languages because there are not enough programmers around who can write programs in machine language for a new machine.[5]

[5]For that matter, the "third generation" of computers has

It would be perfectly possible to have a universal source language except for two influences. The most important one is the fact that a source language is written in the terminology of the field in which it is going to be used. Most computer programs can be divided into two classes: scientific and data processing (business), and so at least two languages are needed. The most popular ones are FORTRAN II (Formula Translator) and COBOL (Common Business Oriented Language) respectively.

The other reason is that a language is never fixed. At some stage, a name is given to a language, hence the II in FORTRAN II, but people keep changing the language under the same name or write a completely new one that has the features they want.

SCATRAN is a good example of this. It is a new language that was developed at OSU to overcome a serious deficiency present in FORTRAN II under the type of computing load expected at a large university. The computing load consists of a large number of small student problems with a very short execution (calculating) time. Most of the time on the machine is spent in compiling the incoming programs so a language with a very fast compiler speed is needed.

---

been delayed by several years because there are not enough programmers who can write compilers. The only person harder to find is the system analyst, a man who not only knows all there is to know about a particular computer, but also knows the field to which it is going to be applied.

SCATRAN is at least an order of magnitude faster than FORTRAN II, so ten times as many programs can be run in the course of a working day. If the execution times for each program were on the order of fifteen to twenty minutes instead of several seconds, there would be no need for the faster compiler. But at OSU, the computing load could not be handled without the SCATRAN language.

Far out in left field are the source languages developed for a special purpose. By developed, I mean that a set of rules have been set up and a compiler written to translate a program according to these rules into machine language or into another source language. An example of this that is available at OSU is SNOWBOL. It was originally written to determine authorship of anonymous passages by word count and analysis of style.

Computing languages at OSU

There are two source languages and two versions of one object language in use at OSU. The two source languages are FORTRAN II and SCATRAN. The object language is FAP (Fortran Assembly Program). I am not going to attempt to describe all of the features of these three languages, to do so would require several books.[6] In the bibliography, there is a list of the essential texts that are needed to use these languages.

[6] See bibliography under SCATRAN, FORTRAN, and FAP.

In general, these books will not show you how to write a good program because that comes only with experience. However, they do list the rules that are to be used and give some examples on how they are to be applied. What the books do not tell you is how to make everything work together, and the purpose of this section is to try to tie up some of the loose ends between the three languages.

Most of the subroutines for producing SC 4020 plotter commands are written in FAP. There are actually two FAP operating programs at OSU. One is the original FAP assembler language used when running main programs under the FORTRAN II operating system. The other is called OS-FAP and is used when running programs under SCATRAN. It can be shown how close the two are by the fact that the object deck output from either assembler can be run under the other operating system. In the near future (1967), both FORTRAN and SCATRAN will be combined into one operating system using one FAP assembler. At the present time, it is immaterial which system is used to assemble a subroutine. The advantage of the FORTRAN assembler is that the diagnostics are better and the statement listings are followed by an index giving the locations of every variable being defined and their locations. This is a help chiefly when trouble shooting a long, complex subroutine or trying to understand one somebody else has written. It is recommended that if a library of subroutines is being constructed, the final version be assembled under the FORTRAN system to obtain the more complete statement listings. This will help the people who are going to use them whether in FORTRAN or SCATRAN. The advantages of the SCATRAN assembler are that it is faster than FORTRAN (of importance only when a large number of subroutines are to be assembled) and programs run under SCATRAN are returned to the programmer much sooner. This is because SCATRAN is the primary language used ar OSU. It is running most of the time while FORTRAN programs are run only at certain times of the day.

However, when writing a main program to produce movies, the programmer will generally use a compiler language and add the calls to the subroutines in that language to make the tapes. There is no choice between FORTRAN and SCATRAN in this case. SCATRAN is the preferred language by far because it will compile more than ten times faster than FORTRAN. Since compiling uses up most of the time in test runs to check out the written programs, almost ten short programs can be run in SCATRAN in the time of one FORTRAN program. Another advantage is that since SCATRAN is the preferred language at OSU, the turn around time is a matter of hours while for FORTRAN it is at least one day.

However, using SCATRAN brought up a very troublesome problem. The foundations for the SC 4020 subroutines were written by North American Aviation as a corporate effort involving the work of many people. At North American, the main languages are FORTRAN II and FAP. This meant that these subroutines were designed to be called from the FORTRAN language. SCATRAN calls in a main program resemble FORTRAN calls, but the operating procedure of the SCATRAN compiler makes it impossible to call a FAP program that was originally written under the FORTRAN system. Therefore, one of the major jobs that had to be done was to write new subroutines, using the old ones as a guide, that could run in SCATRAN. The results of this conversion are covered in great detail in Appendix A.

It is apparent from the foregoing discussion that the subject of computing languages is an extremely complex one. A review of which languages are used where in the course of producing an SC 4020 tape may be helpful. Figure 2 shows the process in block diagram form.

The programmer writes the main program in a source language such as SCATRAN. The computer uses the system compiler to reduce that into an assembly language called FAP which is very close to the IBM 7094 machine language.

The SC 4020 subroutines called by the main program are originally written in FAP. This is because the programmer-oriented languages have a very extensive set of rules, but if a case comes along that does not fit these rules, there is no way the job can be done in that language. One sacrifices flexibility for the advantage of easier programming.

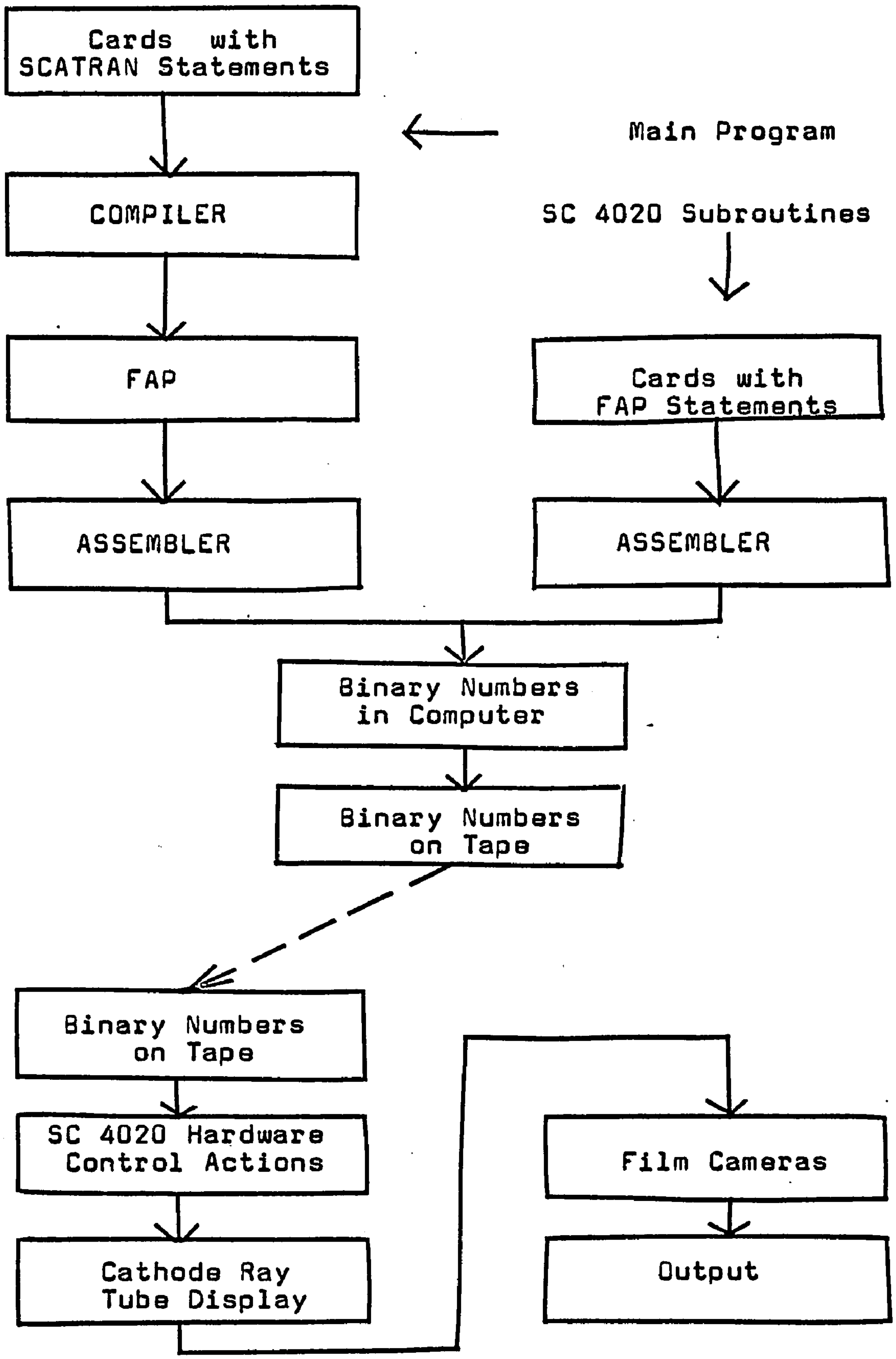


FIGURE 2 - COMPUTING LANGUAGE RELATIONSHIPS IN THE SC 4020 OPERATING SYSTEM

FAP has an entirely different structure from this. The language consists of approximately four hundred different operation codes such as CAL, SXA, etc. Each one of these is completely independent and it is up to the programmer to string them together to form a program that will do what he wants. There are no rules other than the fact that the program has to have continuity so that it can be entered, executed, and left. In fact, the programmer does exactly the same job that the compiler does in the machine but because human programmers are not limited like the compiler,there is no limit in flexibility in writing a FAP program.

The command words for the SC 4020 are constructed bit by bit from the input data. This requires that a very large number of logical operations be performed for every arithmetical operation. In FAP, it is very easy to operate on each individual bit in a thirty-six bit word. In SCATRAN it would be very difficult to operate on the individual bits because the language is not machine oriented. Also, the SCATRAN compiler is very inefficient and a compiled subroutine would have many more words than a hand-written one, thus requiring more room in the memory for any given program.

One subroutine was originally written in SCATRAN and the compiled FAP version is used. This is the subroutine that writes the magnetic tape and handles the tape file in general. This is because the input-output functions are so intimately tied in with the SCATRAN operation system that it is better to use SCATRAN statements to open and close files and set up the output buffers. These calls could be written in FAP but not much would be saved by doing so.

When the program is actually run, all the FAP programs are converted into IBM 7094 machine language by the assembler. The program is then executed. All machine language instructions can be considered as thirty-six bit binary words (usually written as twelve digit octal words), but a very limited subset of these instructions can be considered as an SC 4020 machine language. This language has only eighteen words in it and most of these will not be used in making movies.

The reason that the SC 4020 language is so limited is that the SC 4020 does not have software to extend its capabilities. Each of the eighteen words in the language correspond to a hardware function and if a word is put on tape that is not recognized, the only thing the machine can do is record that an input error has occurred.

In summary, this very large mass of software has only one purpose: to enable the programmer to state in relatively simple terms, "plot a point or draw a line here, using my coordinate system." The computer does the rest, ending up with one of a few words that the SC 4020 will understand, which are repeated millions of times on magnetic tape.

<u>The overall computing and plotting system</u>

The SC 4020 microfilm plotter is only one link on the end of a long chain that is necessary if one is to have a successful motion picture result from the programmer's efforts. Figure 3 briefly shows the elements in this chain in a block diagram. The diagram and the description of the over-all system can be divided into two sections. One is labled "IBM 7094" and the other "SC 4020". The magnetic tape containing the information necessary to generate the movie makes a good place to separate the system, not only because the two machines are actually separated, but also because two different approaches are being used on each side of the dividing line. The IBM 7094 computer is used to create the necessary information, the SC 4020 to transfer this information onto film or hardcopy.

The first, and by far the hardest, job to be done by the programmer is to decide what should be shown in a particular scene. This is because the relationship between the programmer, the subject, and the audience is greatly interdependent. Once the human factors aspects have been satisfied, the main program can be written.

In this case, one scene from the spheroidal radiator will be used as an example to illustrate the overall system. It was desired to show that a particular antenna is in the shape of an ellipse. Most of the viewers of this movie would have seen an ellipse drawn with two focal points and a

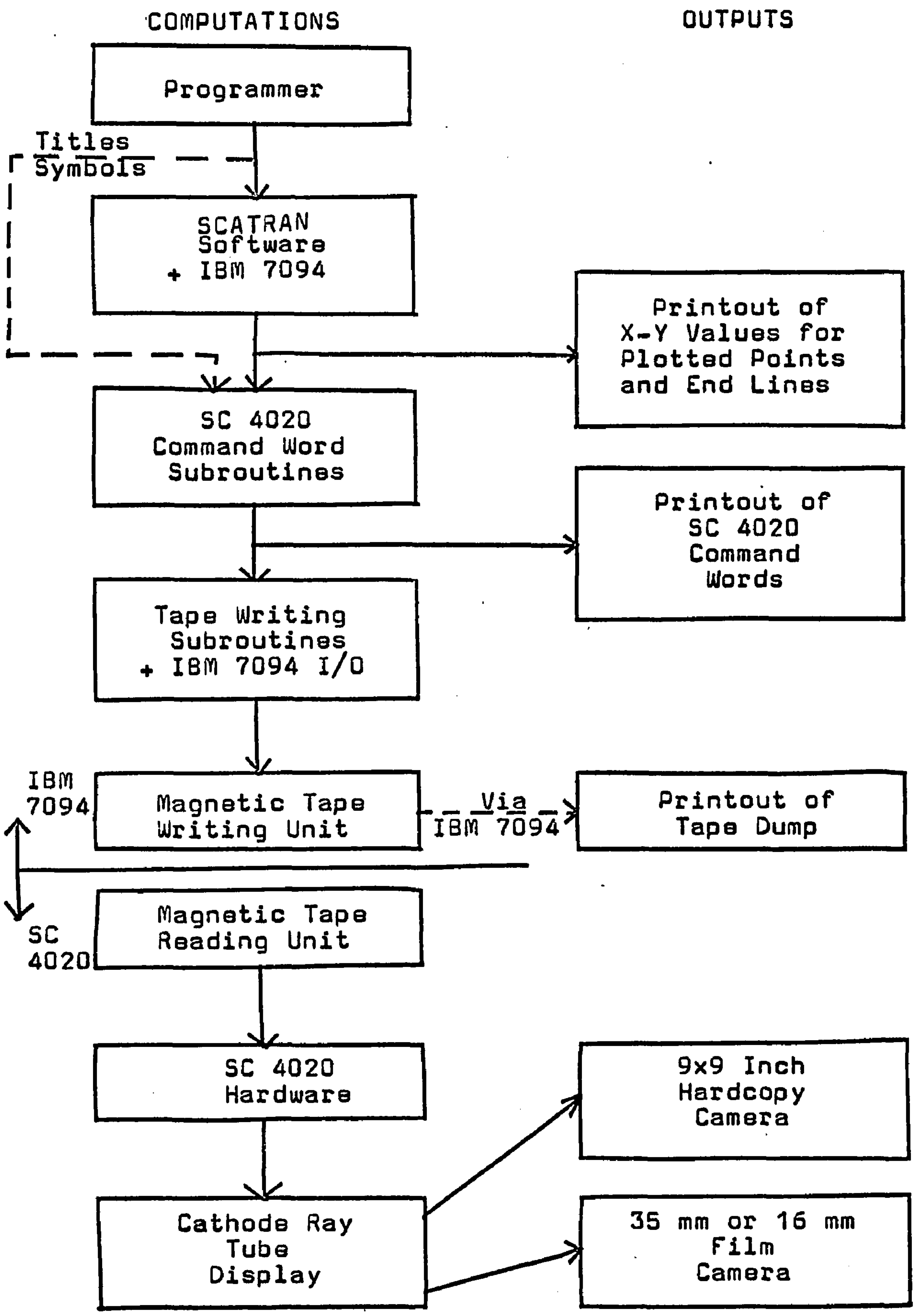

COMPUTATIONS
OUTPUTS
Programmer
Titles
Symbols
SCATRAN Software + IBM 7094
Printout of X-Y Values for Plotted Points and End Lines
SC 4020 Command Word Subroutines
Printout of SC 4020 Command Words
Tape Writing Subroutines + IBM 7094 I/O
IBM 7094
Magnetic Tape Writing Unit
Via IBM 7094
Printout of Tape Dump
SC 4020
Magnetic Tape Reading Unit
SC 4020 Hardware
9x9 Inch Hardcopy Camera
Cathode Ray Tube Display
35 mm or 16 mm Film Camera

FIGURE 3 - OVERALL SYSTEM BLOCK DIAGRAM

loop of string, so it was decided to draw the ellipse by first showing the focal points and then connecting them to a point that would proceed to circle around, drawing the ellipse after it. A coordinate system was scaled and 196 equally spaced points were calculated on the desired ellipse. Up to this point, all calculations were done on the computer using SCATRAN. A check was run by printing out the X and Y values of the points and this is the first possible output in the block diagram.

In the description which follows, it must be remembered that the SC 4020 has a limited number of commands. In fact, in this scene, only two will be used, the drawing of a line between two points and the command to advance the film.

The next thing is the forming of the commands to draw the focal points. The points consist of three short, straight lines that form an asterisk. These are created when the programmer follows the dotted line in the block diagram and uses the SC 4020 subroutines directly to form a symbol. Without going into details, the programmer can use the subroutines to specify that a symbol (the asterisk) should be drawn to a certain size in a certain location in his coordinate system. The subroutines look up the symbol in a table, determine that three lines are to be drawn in a relationship that will look like an asterisk, check the size and location, and then form three control words, one for each line, that will be understood by the SC 4020 hardware. These words are placed in an output buffer reserved in the computer memory and the program proceeds to calculate the second focal point in the same manner. These control words are described in Appendix B. They will eventually be placed on the magnetic tape, one at a time, in the order in which they were placed in the output buffer. These words can be printed out in the course of a run and can be used to check that a tape is being generated which the SC 4020 will recognize. This is the second possible output in Figure 3.

The next thing is to draw the string. A LINE subroutine is used to draw lines from one focal point to the other, and from each focal point to a specified point on the outside of the ellipse. This point is the first location in an array containing the 196 X and Y locations around the ellipse. Since the SC 4020 can only draw lines that are limited in length for each control word, the subroutine may have to generate several words which, when interpreted by the plotter, will draw short lines end to end to cover the desired distance. These words are placed in the output buffer. This completes the first scene, so the programmer specifies that an advance film command is to be placed in the buffer and the magnetic tape writing subroutine is then called.

This subroutine takes all of the words that have been placed in the output buffer and writes them on a magnetic tape as thirty-six bit binary words. Six bits are placed in one row with six rows containing a complete word. When all of the words have been recorded, the tape is moved ahead three-quarters of an inch to form a record gap. The record gap can be used to find where information was placed on a tape.

After writing the tape, the subroutines return control to the main program which calculates the same scene over again with one difference. The second point on the ellipse is used for the lines from the focal point and a line is added from the first point to the second point. This is repeated point by point with an advance film command after each set of calculations. The final effect when these commands are recorded is the string swinging around the two focal points, leaving the ellipse behind it. With the proper use of DO loops in the SCATRAN program, the number of statements the programmer has to write is very small.

Most of the work is done by the software. The SCATRAN subroutines calculate the desired X and Y values and also write the magnetic tape. The SC 4020 subroutines translate the information generated by the SCATRAN program into control words that can be recognized by the SC 4020. The main program, which is the only one written by the programmer, merely serves as a control that keeps track of the jobs to be done and transfers control from one subroutine to another.

Another form of output can be obtained at this point. After the magnetic tape is returned to the programmer, he can have a "dump" printed out. This will show exactly what is on the tape before it is sent to the SC 4020.

The SC 4020 reads the magnetic tape word by word and does the job specified by each one. In this case, each word will first position the cathode ray tube beam at a certain location. The beam will then be turned on (unblocked) and moved in the desired direction for the desired length. This process is repeated for each word on the tape. The advance frame command moves the film at the proper times and the programmer can control either one of two cameras to receive back either film or 9 x 9 inch hardcopy.

SC 4020 subroutine library

The subroutines mentioned above form the SC 4020 library. The purpose of a library in an operating system is to make life easier for the human programmer. He can concentrate on writing his main program and leave the details to the library. He can use calls that will preform a specified operation without further effort on his part. For example, there is no reason why sines and cosines have to be calculated by the main program each time they are needed. A call to the SIN subroutine will not only save the programmer's time needed to write the statements for the calculations, but he also has the assurance that the SIN subroutine will work every time while his own program might have mistakes in it.

The operating system library can be extended by the programmer himself through the use of a subroutine deck. This deck will contain any subroutine written by the programmer and will be placed in the back of the main program to take care of the calls in that program. The operating system is very accommodating in that if the programmer wishes to use his own subroutines rather than the system library, the system will choose the same name in the deck over the one in the library. Another reason for using a subroutine deck is that there may be very little call for a particular subroutine. Since each subroutine occupies space on the system tapes, it is better to have the few people using the SC 4020 use the necessary subroutine decks rather than to occupy this space permanently. When OSU acquires a microfilm plotter, these subroutines will become a permanent part of the system library.

The available SC 4020 subroutines are described and listed in Appendix B, but a brief review will be given here. There are four groups of subroutines that are of interest in the production of movies. They are as follows:

1. Setup routines to make tape
2. Routines to plot points
3. Routines to draw lines
4. Routines to construct tables for characters and symbols

The setup subroutine for producing SC 4020 tapes will open up a tape file, start a clock running to time the program, and write an ID frame to identify the run with only one call in the main program. Another call at the end of the program writes a closing ID frame, either rewinds or dismounts the finished tape, reads the clock and prints out the time taken by the program, and calls ENDJOB to close the run. This subroutine also handles the necessary buffers to store information that is to be written on tape and generally keeps track of all the details such as when to write an "advance frame" command on the tape.

The routines to plot points are very useful to draw smooth curved lines by plotting a series of touching points along the path of the line. This can be done with a single call if the X and Y locations of each point have been calculated and placed in an array before the call.

The LINE subroutine will draw a line between any two specified points. This is done by using limited length end-to-end vectors to fill up this space. Routines are also provided to draw either an X or Y axis with a single call.

Some of the most useful subroutines for generating computer animation are the ones used to generate variable sized symbols. They allow the programmer to specify the shape, size, and location of any symbol completely independent of each other. In addition, a writing routine allows the programmer to use variable sized letters, one after the other, for titles and subheadings in his movies by only specifying the text that is to be used. Several examples of the use of these subroutines are given in this work. They include the titles, the generator symbol, and the ruler discussed in the spheroidal radiator movie example and complete details are given in Appendix B.

Use of the SC 4020 subroutine calls in producing a movie

As an illustration, we will consider a very simple program and then add the required calls.

```
          DIMENSION (X(6),Y(6))-
    START DO THROUGH S1, J=0,1,J.L.50-
          DO THROUGH S2, K=0,1,K.L.6-
          X(K)=COSF.(THETA)-
          Y(K)=SINF.(THETA)-
      S2  THETA=THETA + 1.0472-
      S1  THETA=THETA + .1256-
          CALL SUBROUTINE()=ENDJOB.()-
          END PROGRAM (START)-
```

This program has two DO loops. The inner one, using K as an index, calculates the position of six evenly spaced locations on a unit circle. The outer one, using J as an index, calculates fifty different sets of the six points after moving them one-fiftieth of the way around the circle between each set. The final result, if made into a movie, would be one cycle of six equally spaced dots moving completely around a circle.

The statements needed to call the SC 4020 subroutines are now added to the above program in order to translate the point locations into command words on magnetic tape. The new program is listed below with the original statements underlined.

```
        DIMENSION (X(6),Y(6))-
START   CALL SUBROUTINE()=OUT.(4)-
        CALL SUBROUTINE()=CAMRAV.(9)-
        DO THROUGH S1,J=0,1,J.L.50-
        CALL SUBROUTINE()=FRAMEV.(-1)-
        DO THROUGH S2,K=0,1,K.L.6-
        X(K)=COSF.(THETA)-
        Y(K)=SINF.(THETA)-
   S2   THETA=THETA + 1.0472-
        CALL SUBROUTINE()=APLOTV.(-6,X(0),Y(0),
        1,1,1,44,ERROR)-
   S1   THETA=THETA + .1256-
        CALL SUBROUTINE()=CLOSTP.()-
        END PROGRAM (START)-
```

The first thing to notice is that the statements for calculating each scene (in this case, the inner DO loop with K as an index) are unchanged.

Two statements have been added at the top of the program and one at the bottom outside of any of the DO loops. Subroutine OUT. determines the form of the output (printed or magnetic tape) for this program. Subroutine CAMRAV. opens the tape file and writes the ID frame on it. At the bottom, subroutine CLOSTP. prints out the time used, closes the tape file, and transfers control to the subroutine ENDJOB.. This call was eliminated from the original program.

The two subroutine calls FRAME. and APLOTV. are added inside the beginning and end of the DO loop that generates each frame. The FRAMEV. call writes an advance frame command on the tape and closes the record. The APLOTV. call generates the necessary command words to plot the dots from the X and Y data arrays.

In addition to adding these statements to the main program, it is necessary to add eight subroutine decks to the back of the main program deck. Also, certain data cards have to be added at the end of the complete deck in order to supply the necessary information to scale the X and Y points into SC 4020 command words and fill out the ID frame with the date, programmer's name, etc. Complete details are given in Appendix B.

The final result is a reel of magnetic tape containing the necessary control words to write each specified scene and advance the frame between each one. If this tape was run on an SC 4020, the programmer would receive back fifty frames of hardcopy containing six asterisks each plus an opening and closing ID frame.

CHAPTER IV

THE SPHEROIDAL RADIATOR, AN EXAMPLE

Summary

It was decided to make a full length (three-minute) movie utilizing all of the available computer techniques. Any difficulties that might show up in using the SC 4020 software could be corrected before the subroutines were released, but, more importantly, it would be a demonstration of the effectiveness in using the computer to generate a mathematically complex moving picture both in research and in the classroom.

When properly presented, a motion picture presentation should be understood even by viewers without a full mathematical grasp of the solution being presented. It is hoped that this example will illustrate these concepts.

The subject matter chosen was the resonant spheroidal radiator. This antenna consisted of a surface formed by rotating an ellipse about its major axis. A certain current distribution was assumed to exist both on the surface and in the interior of the antenna. The choice of these particular currents then determined a radiated field which satisfied both Maxwell's equations and the values on the boundaries of the system. The field produced by such an antenna is described in the next section of this chapter.

Some of the considerations that went into making the movie are discussed. This is followed by a brief outline of the script used for the movie.

The mathematical basis of the electromagnetic field lies in both the coordinate system chosen and the solutions to the differential equation generated by Maxwell's equations. Both of these are discussed and a derivation of the actual formulas used for the movie is given.

The programming and a description of what was being attempted for each scene is included along with some of the problems encountered. Finally, a short criticism of the finished product ends the chapter.

The resonant spheroidal radiator

One of the advantages in using the resonant spheroid as an example is that the field it produces is axially symmetric. Therefore, all of the information available from a pictorial field representation can be shown in two dimensions instead of three. A typical display of the field pattern for the resonant spheroid is shown in Fig. 4. If the entire set of lines is rotated about a vertical axis, the result will be a series of curved toroids that move outwards from the center spheroid. Each toroid has a nest of smaller toroids inside it.

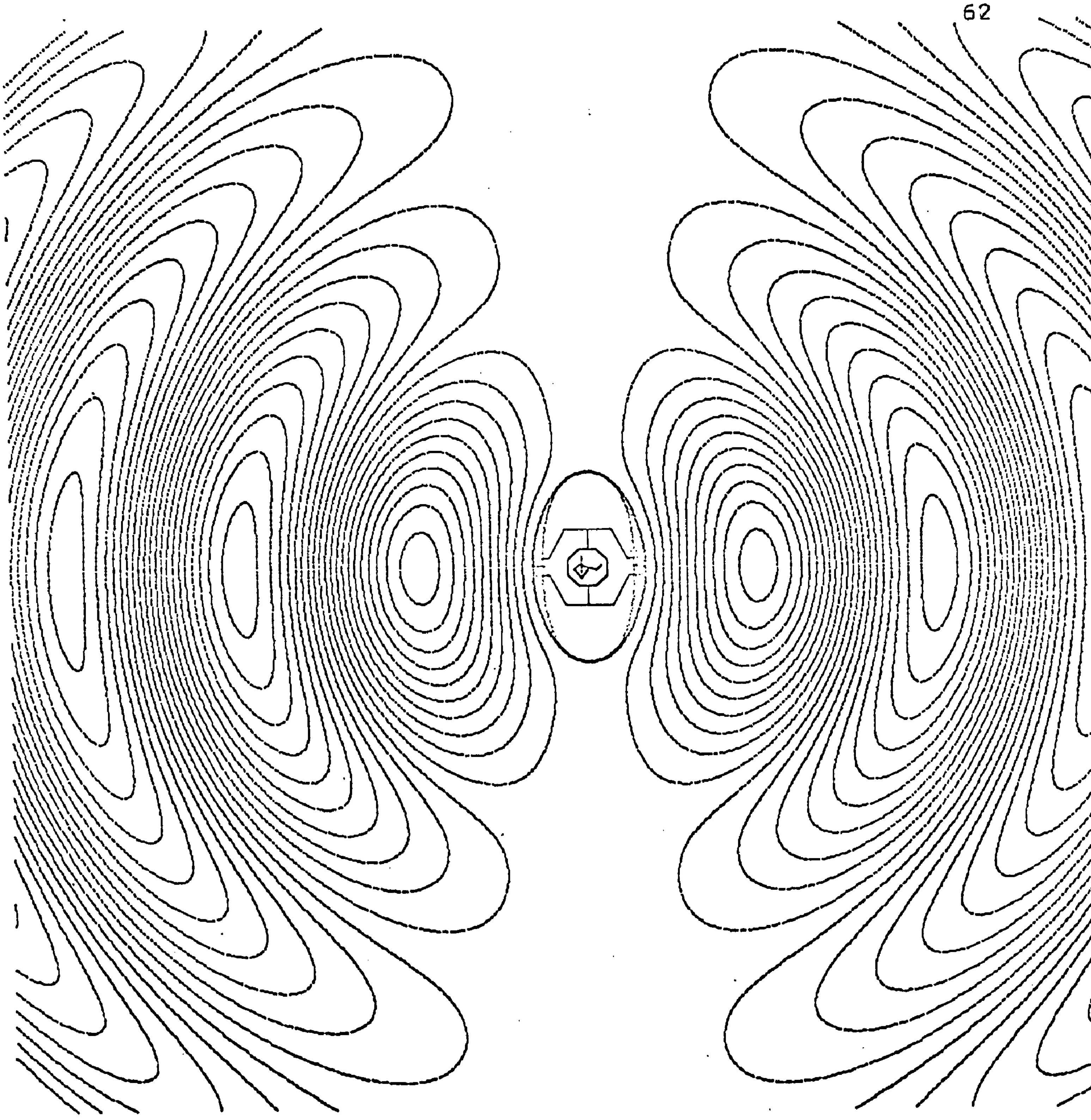


FIGURE 4

THE RESONANT SPHEROIDAL RADIATOR

A three-dimensional axially symmetric field is ideal when one is limited to a two-dimensional display because any cut made along a plane containing the axis is identical with any other cut. Therefore, all of the information available from a pictorial field representation can be shown in two dimensions instead of three.

Each one of these toroidal surfaces represents a stream-surface which has no current flowing across it. This can be determined from Ampere's Law:

$$\oint \bar{H} \cdot dl = I_d + I_c$$

where I_d and I_c are the displacement and conduction currents enclosed by the path of integration. If, due to rotational symmetry, the H field is independent of the angle about the vertical axis and the integration path is a circle of radius ρ about this axis, then

$$\oint \bar{H} \cdot dl = 2\pi \rho H_\phi = I_d + I_c$$

If I_d and I_c are set equal to a constant, then the above equation describes a surface of revolution. Each of the streamlines in Fig. 4 were calculated from a different constant. It can be shown by applying Apmere's Law to a surface area that the net current flow across a streamsurface is zero. Therefore, if no conduction currents are present, the displacement current across the surface must be zero.

Assume that a plane cuts a family of streamsurfaces normal to the axis of symmetry. Any two surfaces that intersect this plane form two concentric annular areas. Since no displacement current flows across a surface, the current in one annular area must be equal to the current flowing in the other area but opposite in direction. Therefore, the motion of the displacement current is a rolling one around the center of each nest of toroids. The motion reverses between each set so that there will be no discontinuities in the field. It can be seen that the current density across an area normal to the flow between two toroids depends on the position of the area. Obviously, the current density is greater on the X-Z plane than at the top of the toroid and this is shown by the density of the streamlines at that point.

I wish to give full credit to David M. Shupe of the Department of Electrical Engineering, OSU, for his work in the development of the streamline model that was used to represent the resonant spheroidal radiator from the electromagnetic field equations.[1]

Technical decisions and limitations

Once it was decided to produce a movie with continuity, rather than separate scenes that would just demonstrate the

[1]David M. Shupe, "Streamline Representation of Radiating Electromagnetic Fields" (unpublished Master's thesis, Dept. of Electrical Engineering, Ohio State University, 1966).

computer generating techniques that have been developed, a plan of action was needed. Before starting, two limitations were imposed.

In view of the fact that it was basic policy to have the computer and display device do as much of the work as possible and to have as little photographic work required other than the developing of the basic film, it was natural to reject sound as one of the techniques that could be used to explain the pictorial context of the film. Therefore, the movie would have to be complete in itself without the aid of narration to explain the motion.

The other limitation was that since this film was to be only a prototype and not a finished product, a length limitation would be set to use one hundred feet of sixteen millimeter film. The film comes in hundred foot lengths with the next larger size being 400 feet. The nominal speed of sixteen millimeter is twenty-four frames a second with each frame being three-tenths of an inch long. Therefore, a hundred feet of film will result in a movie that has approximately four thousand frames and runs for 167 seconds or two and three-quarters minutes.

It was decided not to use photographic stretching for any part of the movie. Photographic stretching is a method whereby only one frame is generated by the computer for each scene. Photographic means are then used to generate, in the case of titles, etc., a large number of frames in a row to give the viewer time to read the text. Any animated motion that is cyclic in nature can be extended by producing only one cycle on the computer and running the one cycle over and over again to generate photographically the required length of film. This method has great advantages in that costs are far less expensive per minute of film due to the difference in cost between computer time and the cost of having the photographic work done. In fact, up to this time, all of the computer generated movies that have been produced used this technique in one form or another.

However, after consulting with the animation experts at Educational Services, Inc.,[2] who had produced several computer generated movies in another field, it seemed better to absorb the greater computer costs and not use photographic stretching. The main problem is in the registration of the film in the camera. When a movie is produced, each frame on the film has to be exactly in the same position as the last one when the shutter opens or the motion will flicker or jump when the movie is shown. This is no problem when ordinary movies are filmed because the film is always moving smoothly through the camera at a constant rate.

An entirely different situation exists during the transport of the film in a microfilm plotter. The film is

[2]Phone calls to Mr. Como, ESI, during May and June, 1966.

moved into position and the shutter is opened. The amount of time that then elapses depends on how much information there is to be plotted in that frame and can be as long as twelve seconds. Since each frame is independent, there is no requirement for accurate registration and therefore the average camera in use on these plotters has very poor registration from frame to frame.

When attempting to use the average camera for making movies, the picture jumps so much that the result is practically useless. In the demonstration movie, the SC 4020 was equipped with a "snap action camera" built in the 1920's. This camera moved the film with a sprocket mechanism that hit a plate at the end of its stroke so each frame was very precisely positioned before the shutter was opened.

Once a cycle is photographed (50 to 90 frames) by this type of camera, it would be possible to lengthen the film by photographic stretching as mentioned above. But the same problem of registration is present after the first cycle has been copied. The original film then has to be moved back to its starting point and the registration with the copy is lost. This causes a jump each time the film is cycled in making the copy. This is not a problem, of course, if the complete run is generated on the computer. The fact that the registration of the snap action camera is good enough to give good animation from frame to frame guarantees that the animation from cycle to cycle will be just as good.

One other decision was made which, at the time, was thought to increase the amount of computing time needed to run the example, but proved, in the long run, not to have increased the necessary time at all. There are two ways of drawing a line with the SC 4020 software, either using the LINE subroutine which draws a straight line between two specified points, or by plotting dots along the path of the line. The position of the dots are calculated so that each dot touches the one next to it to form the line.

The LINE subroutine breaks up the distance between the two specified points into vector commands. If the points are more than sixty-four raster counts apart in either the X or Y direction, a separate vector command is required for each sixty-four count section. Of course, if the distance is less than this, only one command is needed. Since dots plotted in every other cell will make a continuous line due to the size of the dots, the maximum distance that can be covered by a single vector command would require at least thirty-two dot plotting commands. In addition to the time required to place these commands on tape, time is also required to calculate the positions of each dot so it would appear that to draw lines by this method would increase the computing time required by quite a bit. In fact, all of the computer plotted drawings appearing in the literature have used the lines drawn between points as the method used to create the desired figures.[3]

However, the decision to use a dot-plotting technique to draw lines turned out not to require any excessive amount of time over the line-drawing method. As long as the subject matter is composed primarily of straight lines, the line-drawing method would be faster, but the spheroidal radiator example contains only curved lines. In order to obtain smooth enough curves, the end points of each line would have to be close enough together that almost as many points would have to be calculated as in the dot-plotting method. Since a certain amount of time has to be spent in the LINE subroutine to generate the vector command, the final result is that the two methods use approximately the same amount of computer time.

Because the plotting dots are round and do not overlap completely, the line drawn is not completely smooth but has a minor roughness along the edges. This is offset by the fact that a continuous curve can be drawn without any of the sharp breaks that appear when using line segments, no matter how short.

---

[3] Ivan L. Finkle, "Recording Lissajous Figures," Science, Vol. 148, No. 3577, June 1965, pp. 1541-1542.

G. A. McCue and J. D. O'Keefe, "Computer Stereography," Science, Vol. 151, No. 3712, Feb. 1966, pp. 839-840.

Script outline

Although each scene will be described in turn later in this chapter, something should be said about the overall goals of the example movie. First of all, it should be remembered that this is a prototype showing the use of the SC 4020 plotter and is not a finished product. It undoubtedly has as many things wrong with it as right, but at the same time, it is only supposed to show the way forward.

The first efforts to achieve a coherent text in the time allowed resulted in a few short major scenes with the great bulk of the time being spent in text that tried to explain what was going on. By the time the viewer had been given enough time to read a complete text, most of the time had been used up and very little was left for any action. One of the main purposes of this work was to demonstrate the usefulness of short movies in the electrical engineering classroom. The result of so much text and so little activity would have been an effort that could better be duplicated by an instructor explaining from a book.

Professor W. H. Huggins of Johns Hopkins University has suggested that the classroom animated film will be fully effective only when the dependence on the text is removed.[4] The concepts expressed should be demonstrated using

[4] Talk prepared for the Symposium on the Human Side of Computing Machines at Bell Telephone Laboratories, Inc., Murray Hill, N. J., June 20, 21, 1966.

only pictorial images that would have some meaning to anyone who viewed them regardless of their background.

It is easy enough to have the computer generate text, and several examples are included in the example for completeness. However, the results are a waste of the computer's abilities especially in view of the fact that much better effects can be achieved photographically. It is the computer's capability to display large amounts of mathematical data pictorially that is going to justify the use of expensive computer time and make this method such a useful tool in the understanding of electromagnetic field theory.

Therefore, the spheroidal radiator example has very little text, only one scene in fact other than the title and credits, and tries to show the development of an idea by means of pictures. The film starts out by showing only two points (asterisks) that will be the focal points of an ellipse. An ellipse is drawn much in the same manner as using a piece of string and two thumbtacks. The ellipse is rotated by changing the perspective to show that a three-dimensional object is being shown. It is then shrunk to a smaller size and a generator is added in the interior. This is followed by a dark fringe on the surface of the ellipse that varies both in position and time in proportion to the magnitude of the current that would appear on the actual antenna. The electric field streamlines begin to radiate from the spheroid until they fill the screen. The motion is stopped and a ruler is used to compare the distance between the focal points with the distances for one-half cycle of the radiated wave. The final scene shows the direction of motion of the displacement current by means of moving segments added to the streamlines. The motion reverses for each half cycle. A more complete description of each of these scenes will be given when the programming is described.

I wish to thank Professors Cowan and Kennaugh for their help in developing a script for this movie. Many of the scenes were the result of suggestions by them. Some additional suggestions that were not incorporated into the movie due to lack of programming time but which would have resulted in an improved version of this example are described later in this chapter.

Solutions to the prolate spheroidal antenna at resonance

The main sequences in the film example show the radiating field from a prolate spheroidal antenna at resonance by means of the stream function. Maxwell's equations are used in a boundary value problem to determine the differential equations that specify the conditions of the radiated field. In this case, the internal structure of the spheroidal antenna is unimportant. It is assumed that all generators are completely enclosed by the surface of the antenna and the only thing of interest is the current flowing within and on the surface of the spheroid.

The coordinate system is very important in solving the boundary value problem. The details of the spheroidal coordinate system are as follows.[5]

A family of confocal spheroids are located on the Y axis at the points $\pm f$. Points on these surfaces are defined by the orthogonol coordinate system η, ξ, ϕ where η is the radial distance outward, ξ is the angular coordinate upward, and ϕ is the rotational coordinate about the vertical axis. This system can be defined in terms of a cylindrical system r, ϕ, z by the parametric equations:

$$r = f\cdot\sqrt{(\eta^2 - 1)(1 - \xi^2)}$$
$$\phi = \phi$$
$$z = f\cdot\eta\cdot\xi$$

This transformation is very useful in making the movie. When ϕ is set to zero, the r-z coordinates become the X-Y coordinates of the standard cartesian set used by the SC 4020 software. The development of the equations will show that there is no dependence on ϕ and therefore the field can be

[5] L. J. Chu and J. A. Stratton, "Forced Oscillations of a Prolate Sphero'd," Journal of Applied Physics, Vol. 12, March, 1941, p. 242.

expressed in two dimensions instead of three. Figure 5 shows the two systems used when ϕ is set to zero.

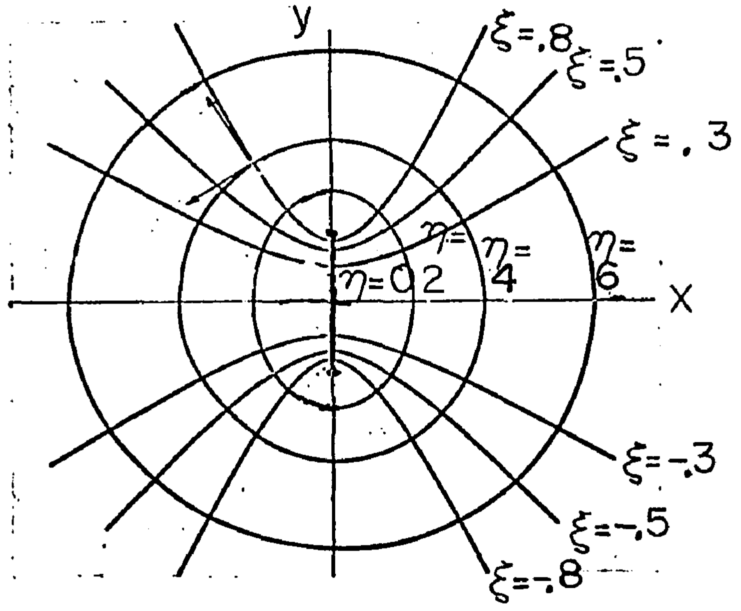


FIGURE 5

$\eta-\xi$ COORDINATE SYSTEM

The fact that the coordinate system and the field both have rotational symmetry can be used to advantage in solving Maxwell's equations.[6]

Since the field has the same symmetry as the coordinate system, the components are independent of ϕ. The two equations:

$$\nabla \times \overline{E} - i\omega\mu\overline{H} = 0$$
$$\nabla \times \overline{H} + (i\omega\epsilon - \sigma)E = 0$$

[6] J. A. Stratton, Electromagnetic Theory (New York: McGraw-Hill Book Company, Inc., 1941) pp. 422-423.

can be broken up into their components and when the differentials of ϕ are set to zero, the following differential equation results:

$$(\eta^2-1)\frac{\partial^2\Psi}{\partial\eta^2}+(1-\xi^2)\frac{\partial^2\Psi}{\partial\xi^2}+f^2k^2(\eta^2-\xi^2)\Psi=0$$

where

$$\Psi=f\sqrt{(\eta^2-1)(1-\xi^2)}\,H_\phi=\rho H_\phi$$

It is obvious that the differential equations can be attacked by means of the separation of variables into two equations that are functions of η and ξ respectively:[7]

$$\Psi=Q_1(\eta)\,Q_2(\xi)$$

$$(\eta^2-1)Q_1''+(f^2k^2\eta^2-C)Q_1=0$$

$$(1-\xi^2)Q_2''+(-f^2k^2\xi^2+C)Q_2=0$$

where C is a separation constant. If we let $C=t^2k^2$, the two

[7]This equation was dealt with extensively in: Robert M. Ryder, "The Electrical Oscillations of a Perfectly Conducting Prolate Spheroid", Journal of Applied Physics, Vol. 13, May 1942, pp. 327-339. This paper suggested a solution to the above differential equation. However, Ryder assumed that a plane wave was incident on a perfectly conducting spheroid and then solved for the current induced in it. The present work assumed a current present on the spheroid that will lead to the desired solution. This current is only a theoretical concept and does not represent any particular configuration of the interior of the spheroid.

equations reduce to ones that have the following solution:

$$Q_1 = A_1 e^{jfk\eta} + A_2 e^{-jfk\eta}$$

$$Q_2 = B_1 e^{jfk\xi} + B_2 e^{-jfk\xi}$$

From the boundary values of the field, $Q_2(\xi)$ must vanish when $\xi = \pm 1$ and $Q(\eta)$ must have the form $e^{jfk\eta}$ for large η, therefore[8]

$$fk = \frac{\pi}{2} p \quad , \quad B_1 = -B_2$$

and

$$\Psi = AB \cos\left(\tfrac{\pi}{2} p \xi\right) e^{j\frac{\pi}{2} p \eta}$$

$$H_\phi = \Psi/\rho = AB \frac{\cos\left(\frac{\pi}{2} p \xi\right) e^{j\frac{\pi}{2}\eta}}{f\sqrt{(\eta^2-1)(1-\xi^2)}}$$

where p is an integer corresponding to a particular harmonic. The equation that was used set p=1 where the interfocal distance (2f) is equal toone half of the wave length of the radiated signal. The value of AB is determined from the boundary conditions of this particular problem.

[8]Ryder, loc. cit., pp. 328-329.

The assumption is made that the spheroid has a surface current flowing on it equal to

$$I_{\eta_0}(\xi) = \cos\left(\frac{\pi}{2}\xi\right)$$

and a current flowing internally equal to

$$I_i(\eta,\xi) = -\cos\left(\frac{\pi}{2}\xi\right)\cos\left(\frac{\pi}{2}\eta\right)e^{j\frac{\pi}{2}\eta_0}$$
$$1 < \eta < \eta_0$$

These currents were chosen in order to give a simple solution for the constant AB and do not represent any particular configuration of generators in the spheroid. In the movie, the interior current was replaced with a generator symbol just to indicate that an active source is required to energize the antenna.

Using Ampere's Law

$$I_i = \iint J\cdot dA \qquad = 2\pi\rho H_\phi$$

for this field. Therefore:

$$H_\phi^i = I_i/2\pi\rho = -\frac{\cos\left(\frac{\pi}{2}\xi\right)\cos\left(\frac{\pi}{2}\eta\right)e^{j\frac{\pi}{2}\eta_0}}{2\pi F\sqrt{(\eta^2-1)(1-\xi^2)}}$$

At the interior surface of the spheroid

$$H_\phi(\eta_0,\xi) = -\frac{\cos(\frac{\pi}{2}\xi)\cos(\frac{\pi}{2}\eta_0)\, e^{j\frac{\pi}{2}\eta_0}}{2\pi f\sqrt{(\eta_0^2-1)(1-\xi^2)}}$$

The current on the surface can be divided by the circumference to give the current density

$$K_{\eta_0} = \frac{\cos(\frac{\pi}{2}\xi)}{2\pi f\sqrt{(\eta_0^2-1)(1-\xi^2)}}$$

One of the boundary conditions states that

$$\hat{n} \times \left[\bar{H}^i - \bar{H}^o\right] = \bar{K}$$

where the two magnetic fields are on either side of a surface current density K. The magnetic field at the outside of the spheroid surface is equal to

$$H_\phi^o = \Psi/\rho = AB\frac{\cos(\frac{\pi}{2}\xi)\, e^{j\frac{\pi}{2}\eta_0}}{f\sqrt{(\eta_0^2-1)(1-\xi^2)}}$$

AB can now be determined:

$$AB \frac{\cos\left(\frac{\pi}{2}\xi\right) e^{j\frac{\pi}{2}\eta_0}}{f\sqrt{(\eta_0^2-1)(1-\xi^2)}} - \frac{\cos\left(\frac{\pi}{2}\xi\right)\cos\left(\frac{\pi}{2}\eta_0\right)e^{j\frac{\pi}{2}\eta_0}}{2\pi f\sqrt{(\eta_0^2-1)(1-\xi^2)}}$$

$$= \frac{\cos\left(\frac{\pi}{2}\xi\right)}{2\pi f\sqrt{(\eta_0^2-1)(1-\xi^2)}}$$

Therefore,

$$AB = \frac{1}{2\pi}\left[e^{-j\frac{\pi}{2}\eta_0} - \cos\left(\frac{\pi}{2}\eta_0\right)\right] = -j\,\frac{\sin\left(\frac{\pi}{2}\eta_0\right)}{2\pi}$$

The requirement that the tangential components of the E field be continuous is used as a check. The tangential E field can be determined from the potential Ψ.[9]

$$E_\eta = -\frac{j\omega\gamma}{\rho h_2 k^2}\frac{\partial\Psi}{\partial\xi}$$

From above:

$$\left.\frac{\partial\Psi_i}{\partial\xi}\right|_{\eta_0} = \frac{\pi}{2}\,\frac{\sin\left(\frac{\pi}{2}\eta_0\right)}{2\pi}\cos\left(\frac{\pi}{2}\xi\right)e^{j\frac{\pi}{2}\eta_0}$$

$$\left.\frac{\partial\Psi_o}{\partial\xi}\right|_{\eta_0} = -\frac{\pi}{2}\,\frac{e^{j\frac{\pi}{2}\eta_0}}{2\pi}\cos\left(\frac{\pi}{2}\xi\right) - \sin\left(\frac{\pi}{2}\eta_0\right)$$

and these are obviously equal.

---

[9]Stratton, loc. cit.

Therefore, the equations that were used for the field are:

$$\Psi = -j\frac{\sin(\frac{\pi}{2}\eta_0)}{2\pi}\cos\left(\frac{\pi}{2}\xi\right)e^{j\frac{\pi}{2}\eta} \quad \eta_0 < \eta < \infty$$

$$I(\xi) = \cos\left(\frac{\pi}{2}\xi\right)$$

Multiplying these equations by $e^{j\omega t}$ to give the time dependence and taking the real part:

$$R_e(\Psi) = \frac{\sin(\frac{\pi}{2}\eta_0)}{2\pi}\cos\left(\frac{\pi}{2}\xi\right)\sin\left(\frac{\pi}{2}\eta+\omega t\right)$$

$$R_e(I_{\eta_0}) = \cos\left(\frac{\pi}{2}\xi\right)\cos\omega t$$

For ease of computing the following transformation was made:

$$\xi = U \quad , \quad \eta = V \quad , \quad \frac{2}{\pi}\omega t = DTME$$

$$\cos\left(\frac{\pi}{2}\xi\right) = \sin\frac{\pi}{2}(1-U) \qquad \text{(for the current)}$$

and the final equations were:

$$\Psi(U) = \sin\frac{\pi}{2}(1-U)\cos\omega t$$

$$\Psi = \frac{\sin(\frac{\pi}{2}\eta_0)}{2\pi}\cos\left(\frac{\pi}{2}U\right)\sin\frac{\pi}{2}(V+DTME)$$

In the calculations for the movie, the term

$$\frac{2\pi\Psi}{\sin(\frac{\pi}{2}\eta_0)} = K = .05, .15, \cdots\cdots, .95$$

for each nest of streamlines. Therefore, V can be determined from U by the formula

$$V = \frac{2}{\pi}\left[K\Big/\cos\left(\frac{\pi}{2}U\right)\right] - DTME$$

and these in turn can be transformed into the X-Y coordinates by the relation[10]

$$X = f\sqrt{(1-U^2)(V^2-1)}$$
$$Y = f\cdot U\cdot V$$

[10]This is the same transformation as for r and z on page 73.

Detailed description of the spheroidal radiator example

The movie starts with a single frame containing a conventional identification frame which serves to both identify the run and to check the correct operation of the SC 4020 plotter.

The next ninety or so calls produce the titles, credits, and text. These four scenes are shown in Figure 6. The lettering is generated by the use of the variable sized symbol subroutines.

Most of the lettering is centered on the screen when producing titles, so equations were developed to make it easier to determine some of the calling parameters in the calls:

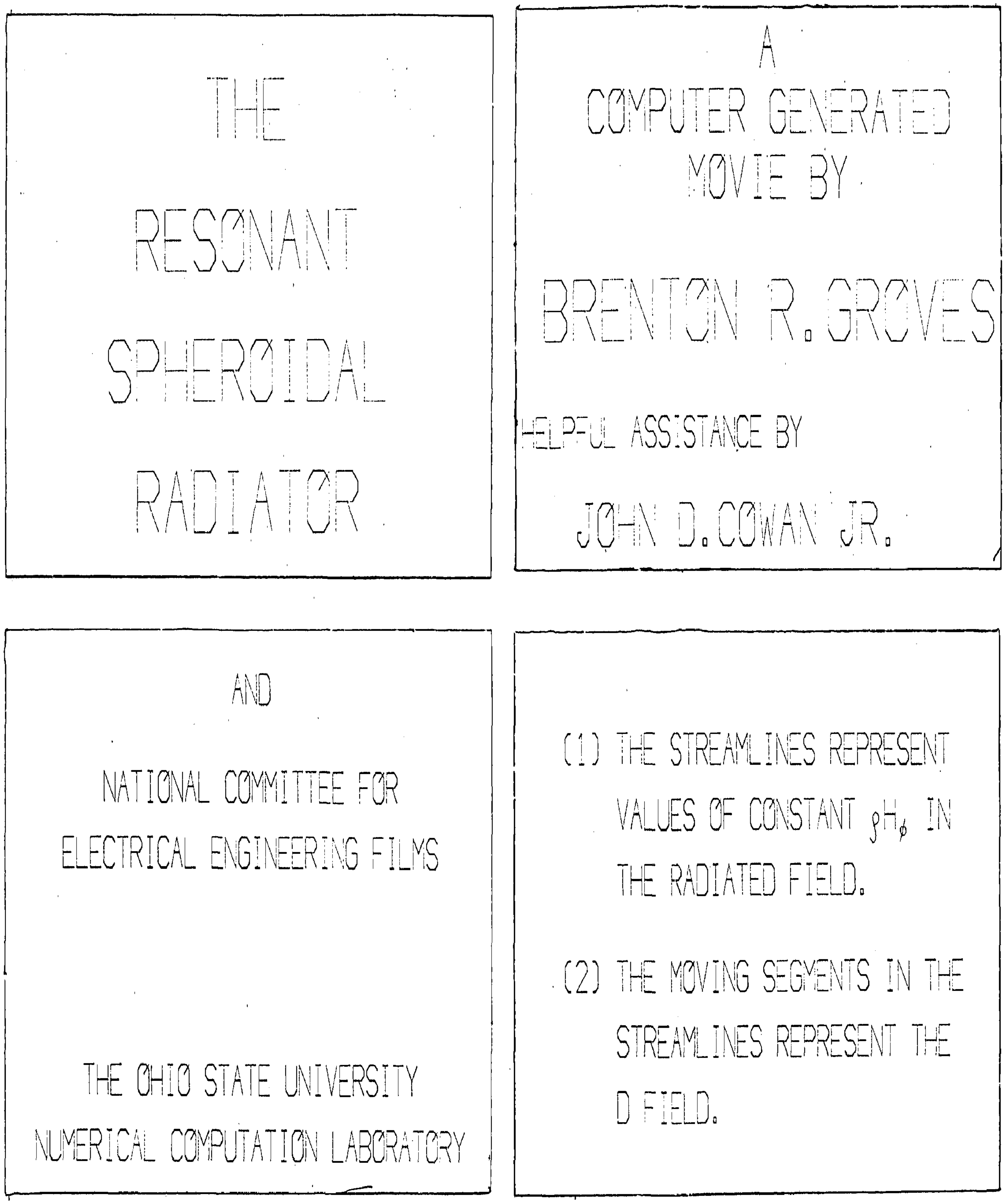
```
CALL SUBROUTINE()=CHSIZS.(IX,IY)-
CALL SUBROUTINE()=RITSTS.(ISPACE,IROW,TABLIV.)-
CALL SUBROUTINE()=RITE2S.(X,Y,IEND,90,1,N,-1,
                                    WORD, IRE)-
```

that are used for titles. ISPACE, which generates the space between letters, can be determined from IX, the lattice space used for tables, by the equation:

$$ISPACE = 5 \times IX + K$$

The addition of K prevents crowding the letters when IX is reduced to the smaller values and is usually set at three.

X and IEND can be calculated from the number of letters in a line when centered lettering is desired.



THE

RESØNANT

SPHERØIDAL

RADIATØR

A

CØMPUTER GENERATED

MØVIE BY

BRENTØN R. GRØVES

HELPFUL ASSISTANCE BY

JØHN D. CØWAN JR.

AND

NATIØNAL CØMMITTEE FØR

ELECTRICAL ENGINEERING FILMS

THE ØHIØ STATE UNIVERSITY

NUMERICAL CØMPUTATIØN LABØRATØRY

(1) THE STREAMLINES REPRESENT VALUES OF CONSTANT ρH_ϕ IN THE RADIATED FIELD.

(2) THE MØVING SEGMENTS IN THE STREAMLINES REPRESENT THE D FIELD.

FIGURE 6. - TITLES, CREDITS, AND TEXT OF THE SPHEROIDAL RADIATOR MOVIE

$$X = (N-1)(2.5 \times IX + .5K)(SCALE)$$

$$IEND = 512 + (N-1)(2.5 \times IX + .5K) + 3 \times IX + K$$

where N is the number of characters in a line including blanks. SCALE is the value of each cell expressed in the programmer's grid. This method of determining position can be used with offset lines if the right number of blanks are supplied at the beginning and end of each line.

TABL1V is used because it contains the standard alphanumeric set. As long as a standard keypunch is used to punch the cards for the format statements (F statements), then TABL1V will match the desired letters or numbers. CHSIZS and RITSTS do not have to be called each time if the size of the letters is not going to be changed between calls. The starting position of a group of letters is set by the call RITE2S so if it is desired to center a line of text, separate calls have to be made for the lines containing an even and odd number of letters. This is why separate calls had to be made for THE and RESONANT SPHEROIDAL RADIATOR in the title scene.

The titles show the largest size letters that can be produced with TABL1V. Larger letters could be produced but a table would have to be written to break up each vector in these letters into a number of pieces.

The SC 4020 tapes were produced with 1024 words in a record. Since the number of words required for each of these scenes was less than this, it was necessary only to calculate each scene once. The following order of calls was used over and over again with minor variations:

```
                    CALL SUBROUTINE()=STORE.(X,4)-
                    DO THROUGH (LABEL),N=0,1,N.L.M-
                    CALL SUBROUTINE()=STORE.(X,3)-
              LABEL CONTINUE-
                    CALL SUBROUTINE()=STORE.(X,1)-
```

The call to subroutine STORE with a calling parameter of "4" adds the advance film command. The DO loop writes the scene on tape M number of times. The last statement writes the scene one more time and then resets the buffer to zero. The amount of text determines the number of times the scene should be called. Enough time should be allowed to read the text through easily for each scene at twenty-four frames per second.

An interesting effect was created with very little programming effort when the names at the bottom of scene two were calculated. One of the calling parameters in RITE2S determines the number of letters in the text to be printed. Even if there are more letters in the text, the subroutine ignores any after this number is reached. Instead of fixing this number at fifty-six, the number of characters in these names, a variable N was used for the calling parameter. This same variable was used in the DO loop that called the scene. The effect is that the first time the scene is written, only one letter will appear. An inner DO loop writes this scene four times to give four frames and then control is returned to the outer loop. This time N is 2 so two letters are written in the next four frames. This is repeated until N reaches fifty-six and the DO loop is terminated. When the movie is run, the names will appear as if written letter by letter across the screen.[11]

The text scene shown in Fig. 6 has one error in it. It should have said,"The moving segments in the streamlines represent the displacement current.", rather than the D field. This could be corrected merely by changing one final statement in the program that produced the titles.

The ellipse

The movie starts with two asterisks being shown near the top and bottom of the center line of the screen. The program then calculates fourty-eight evenly spaced points that lie along the path of the ellipse that is to be drawn. The position of these points are all in the first quadrant. The rest of the points are calculated by first interchanging -X with X and the -Y with Y so that the final result is 192 points positioned evenly around the ellipse.

---

[11]The complete listings for the titles, etc., shown in the four scenes given in Fig. 6 are listed from the start through the statement just after L106 in Appendix D, PP. 354-355.

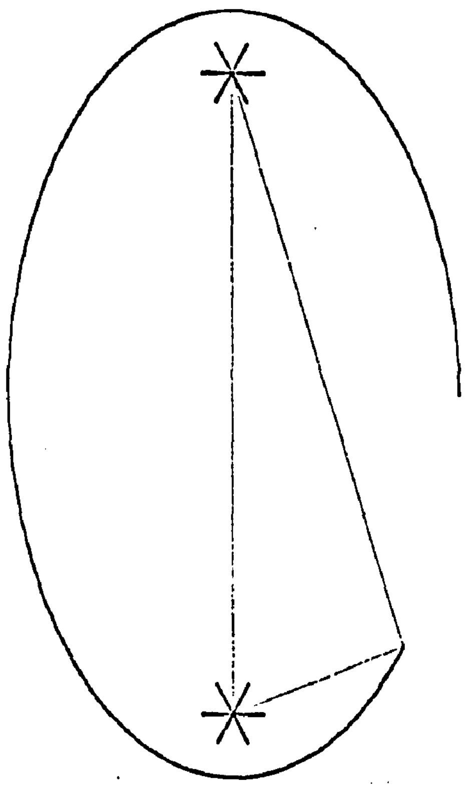


FIGURE 7

DRAWING THE ELLIPSE

The ellipse is drawn by using the LINE subroutine to draw a line from one focal point to the other and from each focal point to point M when M runs from 0 to 191 by means of a DO loop. Another DO loop inside of this loop draws a line between each point to the one next in line up to point M. M is increased by one between each frame and the total effect is the same as that of a string looped around the two focal points with a pencil stretched to draw the ellipse. The motion starts at the right hand center side and swings all the way around counter-clockwise to return to the starting point. The complete ellipse and the two focal points are shown for several more scenes.[12]

The ellipse is next rotated by means of a perspective view through the vertical center line. Each of the X distances of the points on the ellipse is multiplied by the use of the linear increment of time and the ellipse is then drawn using the new points. For each complete cycle the ellipse will shrink in the X direction only to a thin line and then expand to full size again. The effect is that of looking head on to an ellipse of wire that can be rotated into the third dimension about the vertical axis. The idea behind this scene is to show the viewer that what he is seeing is not a two-dimensional figure, but is a plane section of a three-dimensional figure. One of the improvements

[12]The statement listings run from L106 through L112, p.357.

that could be made is to keep each ellipse as it is drawn and to use overstroking of certain ellipses to draw a shaded three-dimensional perspective figure of the spheroid. However, this would add more complexity to an already large program and the rotation and shrinking described next were accomplished with only thirty-three statements.[13]

In all of the above scenes the ellipse almost fills the screen. When the ellipse is used as a radiator, it is much smaller and occupies only the center of the screen with the field pattern filling most of the rest of the space.

Therefore, each point on the ellipse is multiplied by a reducing factor and the ellipse is redrawn with the LINE subroutine. The reducing factor is incremented downward between each frame so that when the film is run, the ellipse will appear to shrink from its original size to the size it assumes for the rest of the movie.

Generator and current symbols

Up to this point, the ellipse representing the spheroidal radiator has been drawn by calculating equally spaced, widely separated points along the path of the ellipse and then connecting each point by using the LINE subroutine. In the main scenes, lines are drawn by plotting touching dots along the path of the line, and to make this program compatible with

[13] Statement listings L112 + 2 to L120 + 1, p. 357.

the ones to follow, the drawing of the ellipse is converted from line segments to dots. The position of each dot is calculated by an iterative process that calculates each position, along a line determined by the equation of the desired ellipse, from the last calculated position. The details are not given here because the program used is fully explained in the section on how the field pattern is calculated.[14]

The last twenty statements in this program (the movie was run in sections of which this is one) form the current generator symbol in the interior of the spheroid and place a dark fringe on the outside surface in proportion to the current distribution. An explanation of the program is given here but it is much easier to see the final result in Figure 8. The current varies according to the equation

$$I(U) = \sin\left[\pi/2\cdot(1-U)\right]\cos\omega t$$

where U is the angular coordinate in the confocal spheroidal coordinate system being used. Since the surface of the spheroid is fixed, the above equation can be converted into one that uses Y as a single variable rather than U. The cartesian coordinate Y is related to the spheroidal coordinates by the equation

[14] Statement listings are from just below L120 to L399 along with a call to APLOTV just below statement L121, pp. 357-358.

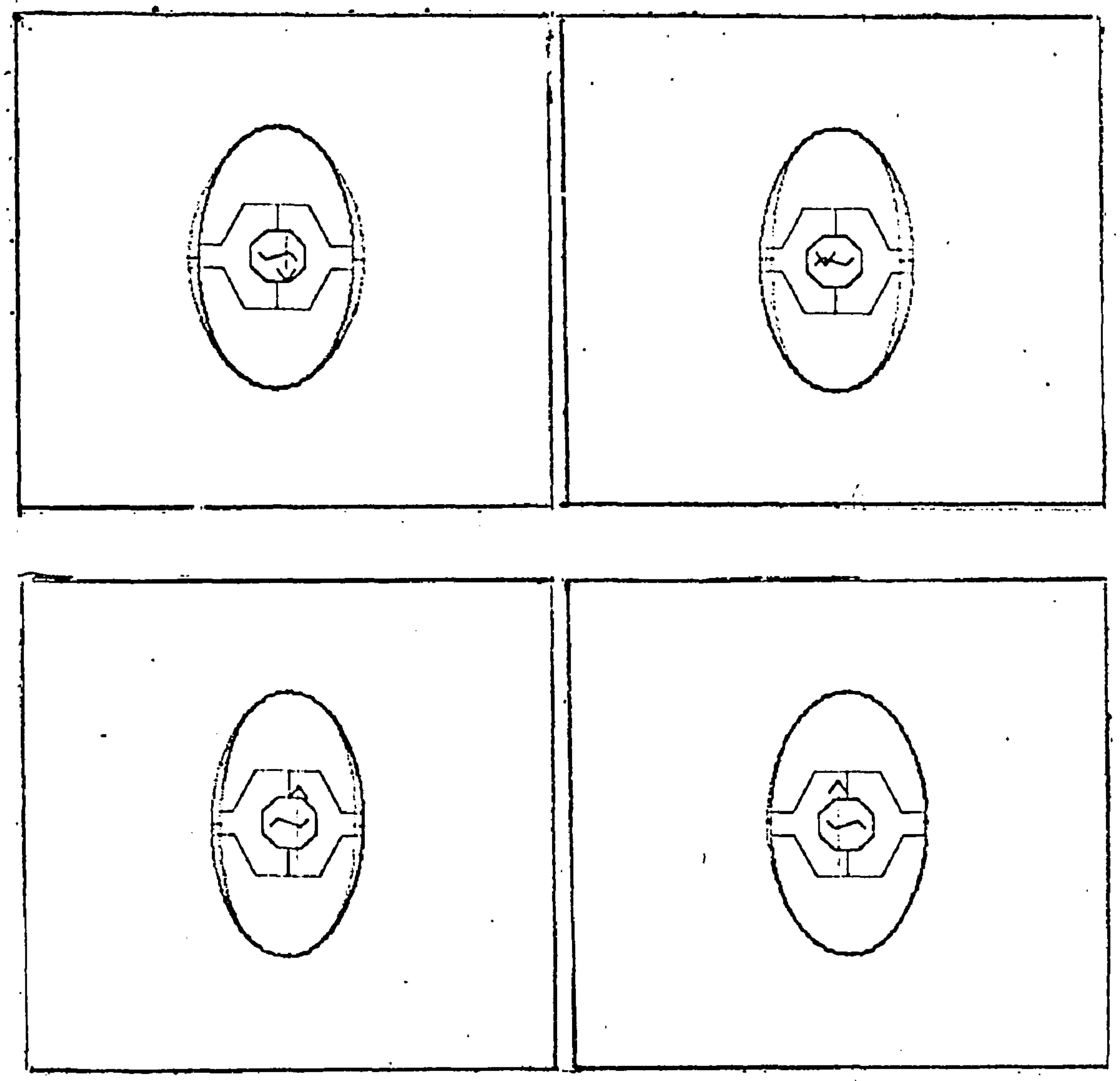


FIGURE 8

FOUR VIEWS - GENSYM AND CURRENT DISTRIBUTION

$$Y = f \cdot U \cdot V$$

where $f = .25$ is the focal distance and $V = 1.2$ is the radial dimension of the spheroid. Therefore,

$$Y = .3U$$

$$U = 3.3333Y$$

and the current on the spheroid can be expressed as

$$I(Y) = \sin\left[\pi/2\ (1-3.3333Y)\right]\cos\omega t$$

This value is calculated for a particular value of t and for each value of Y corresponding to the points calculated along the ellipse. This value is scaled so that it will occupy a distance of ten raster counts when Y and t are zero. This increment is added onto the X value of each point to form a new point and the subroutine LINE connects the two points. Four calls to LINE have to be made in order to cover the four quadrants. The points on the ellipse are touching so all of the lines drawn are side by side and form a fringe on the ellipse. A call to GENSYM with the proper parameter to give the correct magnitude and direction of the current arrow completes the scene. The current symbols and current distribution complete several cycles to show that the antenna is going to be driven by an alternating source.

This scene brings up a far more difficult problem to answer than how a particular movie is going to be programmed. In a nutshell, the question is, "Under the limitations of the computing techniques used, what should be shown to communicate the desired idea to the viewer?". The chapter on conclusions will discuss this question in some detail but it is asked here because this scene caused more discussion than any other over what the spheroidal radiator example is trying to show. The result is an example of what can be done and is not a finished product by any means.

The only way to resolve this problem, especially with regards to movies that are to be used in the classroom, would be to try out as many ideas as possible and then show each one before a typical audience. A test given to the viewers afterwards would determine the most successful presentation.

The radiated field

Most of the remainder of the movie used the same basic program to produce variations on the radiated field.

The program starts by reading in a number of variables for the data card on the back of the input deck. This gives the program a good deal of flexibility because, although the basic equations being used remain the same, the result can be altered quite a bit for any run. The number of cycles, the number of lines in each cycle, and the time between each frame are among the factors that can be preset without changing the program.

Only the first quadrant of the radiated field is calculated. The PLOT subroutine is constructed to fill in a mirror image for the other three quaddants after all the points have been determined. The first quadrant is shown below. All of the lines appearing in the final result are calculated in exactly the same manner so a detailed explaination of how one line is calculated will serve for all the others. Only the input data as determined by a number of DO loops is different.

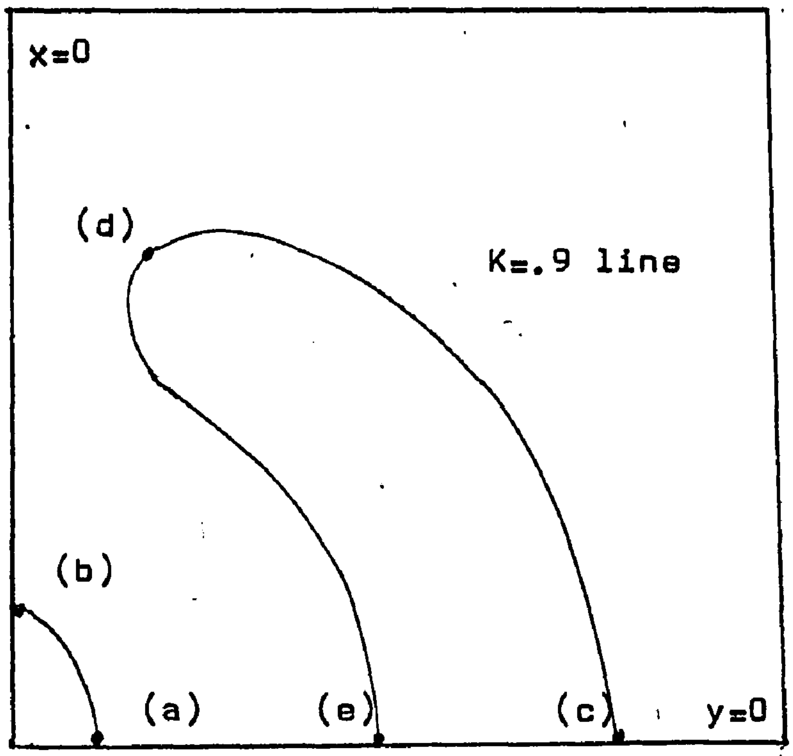


FIGURE 9

1st QUADRANT PLOT OF SPHEROIDAL RADIATOR

The coordinate system used for the calculations is a set of confocal ellipses for the variable V = constant, and two sets of confocal hyperbolas for the variable U = constant. The calculations are executed in this coordinate system, then a transformation is made to cartesian coordinates before the points are actually plotted. The transformation is

$$X = f \cdot \sqrt{(1-U^2)(V^2-1)}$$
$$Y = f \cdot U \cdot V$$

where f is the focal distance from the origin. At large distances from the origin, the U-V system approaches a polar coordinate system with a nonlinear angular measurement.

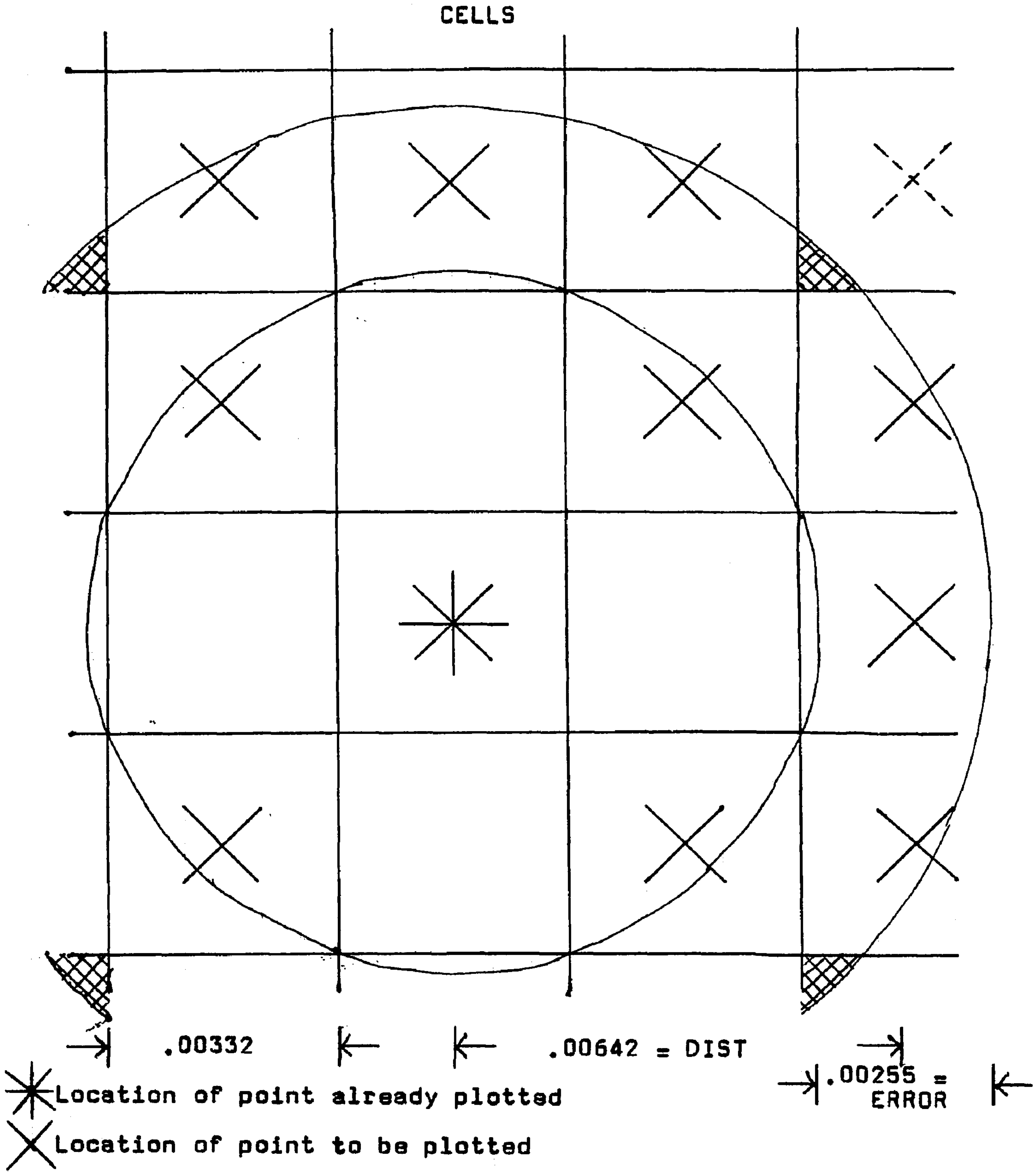
The program determines the ellipse by setting the parameter V to a constant. This constant is called the spheroid index and it can be changed by means of a data card. If the spheroid index is set equal to one, the resulting field pattern would be that of a dipôle, and as the index is increased to larger values of V, the pattern approaches that of a sphere.

The positions of the dots plotted to make a line are all calculated by the same method, but it is easier to see how this is done for the ellipse than for the field.

The variable V_o is set to a constant value, as determined by the spheroid index, and the variable U is set to zero. This determines the first point to be plotted ("a" in Figure

9). The next point is calculated by incrementing U and then calculating the distance back to the first point. If this distance is within preset limits, the X and Y values are stored; if not, the increment in U is adjusted by the amount of the error in distance and the point recalculated. When the value of U reaches one, equivalent to being on the Y axis (point "b"), the program jumps to the next section.

The spacing of these points is very important. A series of plotting dots will overlap to form a line if they are plotted in every other cell. The scale the programmer is using determines the distance between the dots and the size of each cell determines the amount of error in this distance that should be allowed. The eoror should be as large as possible because the iterative process used will require less passes to be within the error. At the same time, if the error is too large, it is possible to plot a point that is far enough away to have a gap in the line being drawn. Figure 10 was used to determine how big the error should be. The asterisk is the center of the dot that has been plotted and the cross shows the centers of any dots that will be plotted if the calculated point falls anywhere in the particular cell. The error shown in the figure was chosen so that two cells side by side would not be plotted. There is a small chance that a cell three spaces away on the disgonal will be plotted, resulting in a small gap in the


CELLS
.00332
.00642 = DIST
.00255 = ERROR
Location of point already plotted
Location of point to be plotted

Note: DIST is approximately equal to plotting dot SIZE

FIGURE 10

ERROR ANALYSIS OF PLOTTING DOTS

line, if the calculated point falls in the crosshatched area. Assuming that the next calculated position can fall anywhere in the error band with equal probability, approximately three per cent of the dots will fall in this category.

The rest of the program can be divided into two steps that are repeated as many times as necessary. In Figure 9 point c is calculated and U is incremented upwards until point d is reached. At that point, a sine term in the field equation is shifted by 180 degrees and U is incremented downwards to calculate the points lying on the line from "d" to "e". When U reaches zero again at "e", the input data is changed and a new line is calculated.

In the section on the mathematical basis of field patterns, the equation for the streamlines was developed in terms of the U-V coordinate system. The final result for the streamline calculations was

$$K = \cos\left(\frac{\pi}{2}U\right)\sin\frac{\pi}{2}\left(V + DTME\right)$$

For the purpose of computing the points along a streamline, K is set to a constant and the above equation is solved for V.

$$V = \frac{2}{\pi} \mathrm{Arcsin}\left[K/\cos(\tfrac{\pi}{2}U)\right] - DTME$$

The value of K determines the particular streamline in a cycle. The different cycles are computed by adding 2n (n = positive integer) onto the value of V for the first cycle. n is the number of cycles away from the origin. V also contains the information as to the time in the cycle of radiation. The position of a point on a streamline at any time in the cycle can be determined by adding the time increment to V before the transformation to cartesian coordinates. The time variable in the computer program is labled DTME and appears in the expression for V.

Point "c" is calculated by setting U to zero, calculating V, adding the time to it, and then calculating X. Points are calculated in the same manner as described for the ellipse. It is easy to tell when "d" is reached because at that point

$$\cos(\tfrac{\pi}{2}U) < K$$

and the arcsine cannot be calculated. If the last point calculated is close to the maximum point of the curve, the program switches to the next cycle of the arcsine. If there is enough distance left on the curve, a few more points are calculated to fill in the top before the switch is made.[15] The rest of the program increments U downwards and calculates that the next line is ready to be computed.

If a streamline is being computed that is close to the spheroid, it may enter the spheroid rather than returning to U = 0. In this case, the program immediately jumps to the next line.

When all of the points have been computed for a particular scene, a call to APLOTV computes the necessary SC 4020 commands to plot each dot. Subroutine PLOT generates four plotting commands to fill the four quadrants for each set of X's and Y's and places them on tape.

The basic program described above was used for the next four scenes in the movie.

Starting streamlines

The scene demonstrates how the streamlines are formed. Only the generator symbol and current distribution on the spheroid shows. The streamlines emerge, one by one, until the screen is filled. It was computed by generating four cycles (enough to fill the screen) but by starting at a negative time. This negative time was large enough when added to V that even for the fourth cycle the total V was smaller than the spheroid index. Therefore all four cycles fall

---

[15]This occurs at statement L17 on page 364.

inside the ellipse and the streamlines are not computed. As time is increased between each frame, V gets large enough to let the streamlines appear one by one. As they flow outwards, more are generated until all four cycles appear.

Ruler

In this scene, time is not incremented between frames. Therefore, the field pattern does not move. Instead of the generator symbol appearing intthe center of the spheroid, two rulers appear. One is vertical and measures the distance between the focal points, the other is horizontal and measures one cycle of streamlines at the right of the screen. The idea is to show the viewer that there is a relationship between the size of the spheroid and the wave length and that the wave length is in fact greater near the spheroid.

The concept of resonance is another area where ground has just been broken. Several suggestions were made to have clocks to show the idea of frequency and wave length in this example but there was not time enough to try out these ideas. Again, the hard part is not the programming but how to get the concept across to the viewer.

Steady state

The main program was run with the time limits set to give a complete cycle moving out from the spheroid. This section was actually computed in two sections, one with five cycles of streamlines, the other with four. By doing this, it was possible to cut down on the computing time while still filling up the complete screen. In the moving segment part of the movie, the switch in the number of cycles used was made automatically by the program so that a whole cycle of motion could be made at once and time would still be saved by having as few off scale points as possible to compute. If this scene were to be done again, it is suggested that this feature be added.

Moving segments for the displacement current

The displacement current moves in a direction always tangent to the streamlines. The motion was implemented by placing segments in the streamlines that moved around a center in each group of lines. This motion is independent of the outward motion of the radiated field as far as the program is concerned.

The missing segments in the streamlines were created by omitting certain dots when the plotting commands were placed in the buffers in the STORE subroutine prior to placing them on tape. As each line is computed, the dots are counted. Twenty-four dots are plotted and the next twelve are skipped so that each streamline consists of a section followed by a gap that is half as long. Motion is generated by properly positioning the gaps between each frame so that the breaks appear to move along the streamline. The problem is complicated by the fact that the motion has to reverse direction between each group of lines.

The basic program computes the spheroid for each frame and then computes each set of lines. Between each set, a call to APLOTV puts the dot commands on tape. When the specified number of groups of lines have been computed, the program returns to the start to compute a new frame. As the X and Y values of each point are computed, it is stored in an array until the call to APLOTV.

The key to the moving segment program lies in the fact that all points that are computed always lie in the first quadrant.

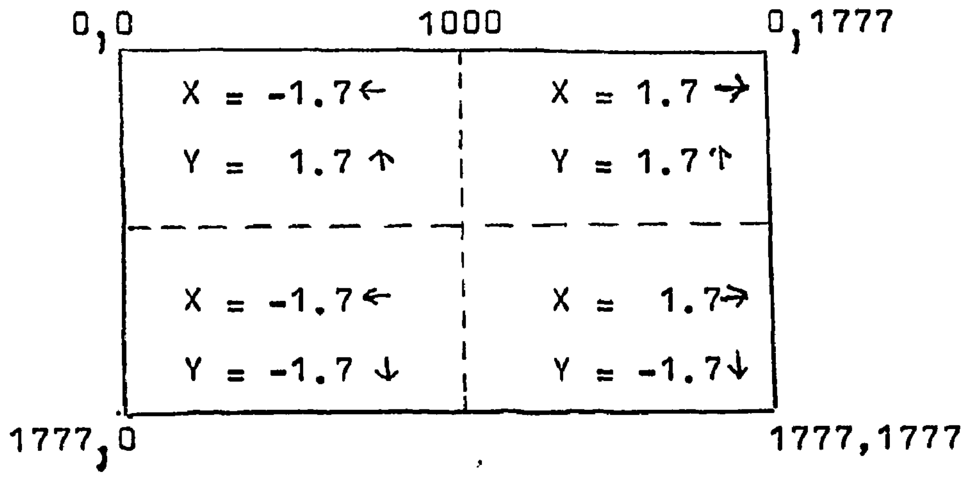


FIGURE 11

PROGRAMMER'S GRID FOR RADIATOR

The value of X and Y is always positive so that when the array is called by APLOTV, the control word sent to the (PLOT) subroutine has the following characteristics:

```
          00   1xxx   52   1xxx
        Plot    X    Dot   Y
     Op Code  Value        Value
```

The two zeros are the operation code for plotting and the "52" is the code for a dot at the specified location. The call positions run from 0000 to 1777 (octal) with 1000 falling on the centerline. The (PLOT) subroutine manipulates the X and Y values to give command words that will plot the other three quadrants.

It was decided to let (PLOT) count the dots that had been plotted to determine if a line or a gap was present and therefore if the dot should be plotted or not. Since the number of dots required for each line depends on its length which is an unknown quantity, the main program has to determine when a line starts. It would have been possible to call (PLOT) directly to start the counting, but another method was used instead. In addition to the values of the position computed by the main program, some values were included that resulted in points plotted in the second quadrant. The (PLOT) subroutine was rewritten so that instead of plotting these words, they would serve as controls to start the counting of the dots along the line. Therefore, this problem can be broken up into two sections. The first is the additions to the main program to create control words and the second is the use of these words in the (PLOT) subroutine.

A brief outline of the main program is:

```
        START
        DO EACH FRAME TO L14
        COMPUTE SPHEROID
        DO EACH SET OF LINES TO L16
        DO EACH LINE TO L9
        COMPUTE LINE
L 9     CONTINUE
        CALL APLOTV
L16     CONTINUE
        CALL FRAMEV
L14     CONTINUE
        ENDJOB
```

The array that stores the data as it is computed is labled Q(N) for the X values and R(N) for the Y's. The following statements were added to the main program.

```
        START
        DELDOT = -.00332
        SET1 = -.00166
        DO EACH FRAME TO L14
        Q(0) = -1.69502
        R(0) = -.066406
        SET = -1.69834
        SET2 = -.00664
        COMPUTE SPHEROID
        DO EACH SET OF LINES TO L16
        DO EACH LINE TO L9
        Q(N) = SET
        R(N) = SET 1
        COMPUTE LINE
L 9     CONTINUE
        CALL APLOTV
SET=    SET - SET2
        SET2 = -SET2
L16     CONTINUE
        CALL FRAMEV
        SET1 = SET1 + DELDOT
L 9     CONTINUE
        ENDJOB
```

The original statements are underlined. Each one of the values above were chosen so that when APLOTV was called the command word produced by XSCALV through APLOTV would have a certain format that can be recognized by (PLOT).

The first X and Y values to be read into the array at the beginning of each frame are

$$Q(0) = -1.69502$$

$$R(0) = -.066406$$

This will result in a command word as follows:

00 0001 52 1012

The 00 and the 52 are constant throughout. However, notice that the third digit from the left is a zero. Since any command word that is to be plotted always has a "1" in this position, the presence of a zero indicates that this word is to be used by (PLOT) to set counters and not plotted.

The 1 in the sixth position from the left is used as an indicator that all the dots in a line are to be plotted. If a zero appears in this position, it indicates that the segments should appear and move counter clockwise in the first and fourth quadrant. The presence of a two reverses the direction of motion. The reason for plotting all the dots at this point is that the spheroid should not have any segments in it.

At the start of the first line, the following values appear:

Q(N) = SET = -1.69834

R(N) = SET1 = -.00166

These two values will be combined to form the word

00 $\underline{0}00\underline{0}$ 52 10$\underline{00}$

The first underlined zero indicates that this is a control word. The second zero indicates that the motion of the segments should be counter clockwise. The last two zeros are used to set the counter and will cause the segments to move between each frame.

The above values are in the innermost loop so the control word appears at the start of each line.

When all of the points in the first line have been computed, the above word is repeated for the next line. When all of the lines in a set have been computed and the values read into APLOTV, the value of SET is changed by the statement

SET = SET - SET2

The sign of SET is changed for the next cycle and a new set of lines is computed. This time the value of

Q(N) = SET = -1.6917

R(N) = SET1 = -.00166

will result in the word

00 $\underline{0}00\underline{2}$ 52 10$\underline{00}$

The zero has been changed to a two so the motion of the segments will reverse. For each set of lines, the sixth figure from the left will alternate between a zero and a two for as many cycles as there are in a frame.

When a new frame is to be computed, the value of SET1 is changed by adding DELDOT to it. SET1 has a value of -.00166 to start. DELDOT changes this by a negative increment of -.00332 which results in one being added to the last two digits of each control word. For example

```
00   0002   52   1000
```

should become

```
00   0002   52   1001
00   0002   52   1002
00   0002   52   1003
```

and so on for each succeeding frame.

These controls are interpreted by the (PLOT) subroutine.[16] This subroutine divided the plotting area into a top and a bottom. Two index registers are used to count the number of dots that have been plotted using modulus "thirty-six" as a base. If the tested index register contains zero to eleven, the dot is not plotted. If it contains twelve to thirty-five, the dot is plotted. If it contains thirty-six, it is reset to zero. After each set is plotted, the index register is increased by one.

[16]Called 4-QUAD(PLOT)#2 in the subroutine list.

The subroutine tests the incoming control word,and if the two highest order digits (six bits) are not zero, the word is not a plotting command and it is placed directly in the buffer contained in STORE.

If the two highest order digits are zero, index registers IR1 and IR2 are loaded with the numbers they held when this call was last made. The incoming word is next tested for the presence of a one in the third highest order bit. If a one is found, the dot command is to be plotted. IR2 is tested and if higher than eleven, a dot is plotted in quadrants I and II. IR1 is tested to see if quadrant III and IV should be plotted. Both index registers are raised by one, reset to zero if they read thirty-six, and the contents stored for next time. Control then returns to the calling program. This portion of the program occupies about the first half of the statements on the subroutine.

If the incoming word starts with two zeros and the third digit is also a zero, the plotting determination is skipped and the second half of the subroutine comes into play.

Figure 12 shows how the dots would look near the center line of the field pattern for ten frames in a row if the right-hand segments are moving counter-clockwise. The small numbers are the magnitudes that have to be placed in the index registers at the start of each line in order to make the segments move properly. For example, in the fifth

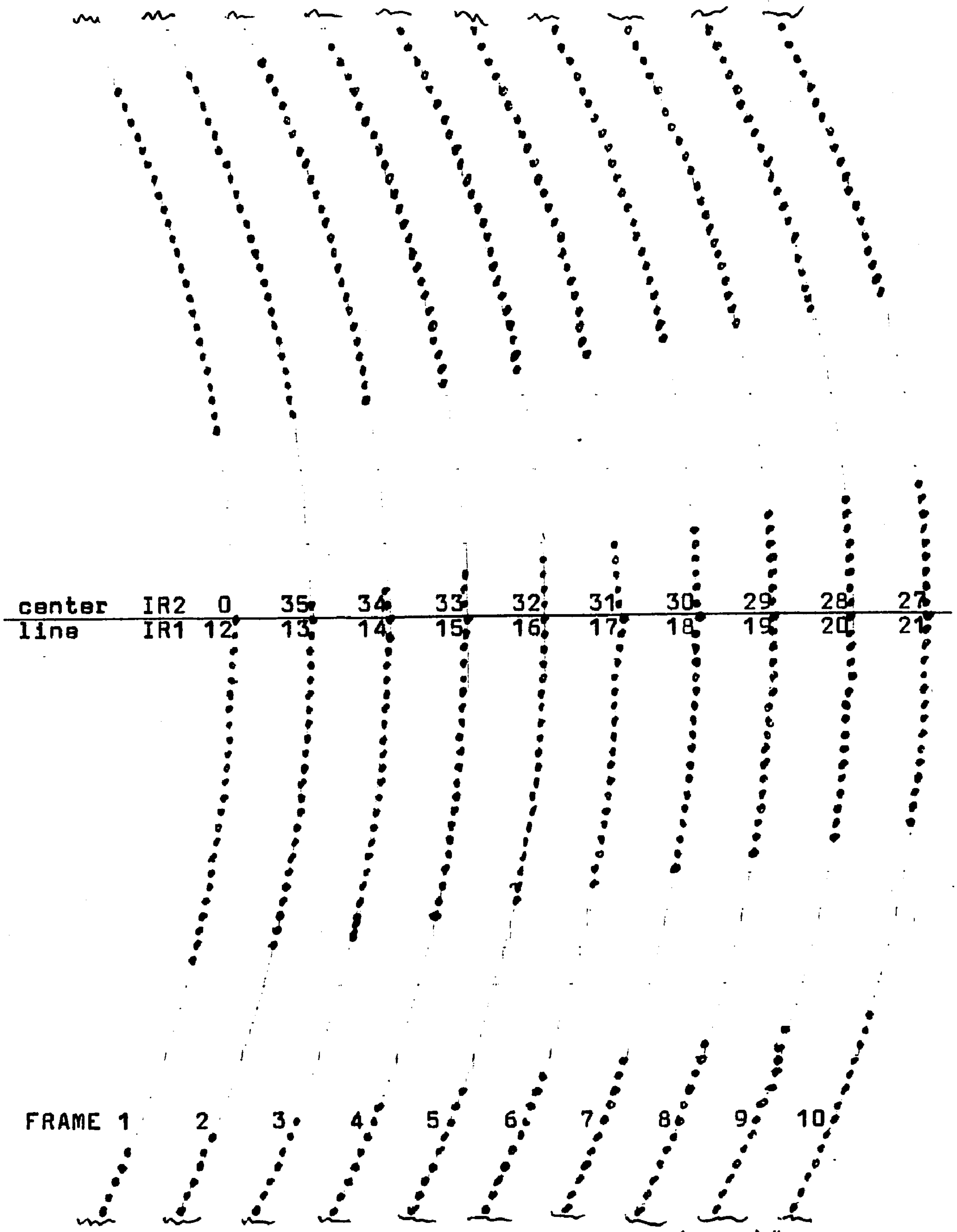


FIGURE 12 - DOTS PLOTTED BY 4-QUAD(PLOT)#2

frame a "32" is placed in IR2. This causes four dots to be plotted and then the next twelve are omitted. IR1 has a "16" placed in it to start. The twenty dots plotted before the gap match the four plotted above the line to fill out the twenty four dot line segment crossing the center line axis. Another graph identical to Figure 12 can be made up in which the segment moves down instead of up.

The subroutine places the correct starting count in the index registers by analyzing the incoming words that start with three zeros. If the sixth number is a "1", the indexes are set to twenty and not incremented. This causes all of the dots to be plotted for the spheroid. A zero or a two in the sixth position in combination with the last two digits are manipulated to give the proper number in each index register. All of the segments in any one group of lines move together and the left-hand side of the pattern is a mirror image of the right.

Off scale dots have to be counted also or the segments would jump as the lines move off the edge of the screen. The subroutine XSCALV has been modified to call (PLOT) and insert the word

```
00  0000  60  0000
```

into the (PLOT) subroutine. Under normal conditions, this word has no effect because it plots a blank at zero-zero. However, the segment generating (PLOT) subroutine will increase the count in each index register by one upon receipt of this word.

One complete cycle of outward motion requires twice as much time as the steady state scene because two complete cycles have to emerge from the spheroidal radiator instead of one. This is due to the fact that the motion of the segments reverses between every group of lines. The group of lines has been referred to as a cycle several times in this work but actually the complete cycle of a spheroidal radiator is represented by two groups of streamlines.

Another timing factor has to be considered if the one computed cycle is to be run through the SC 4020 several times in order to make several minutes run time for this scene. The position of the lines in the last frame has to match those of the first, but also the segments have to have just finished moving across the center line to match the start. This requires that the number of frames in a cycle be divisible by thirty-six to make this match. Figure 13 shows the segments inserted in the spheroidal radiator.

END scene

TABL5V was written to place the block letters

THE END

in the center of the screen by using RITE2S. The program for this scene changes the size of the letters from minimum to maximum in steps between each frame. The effect when the

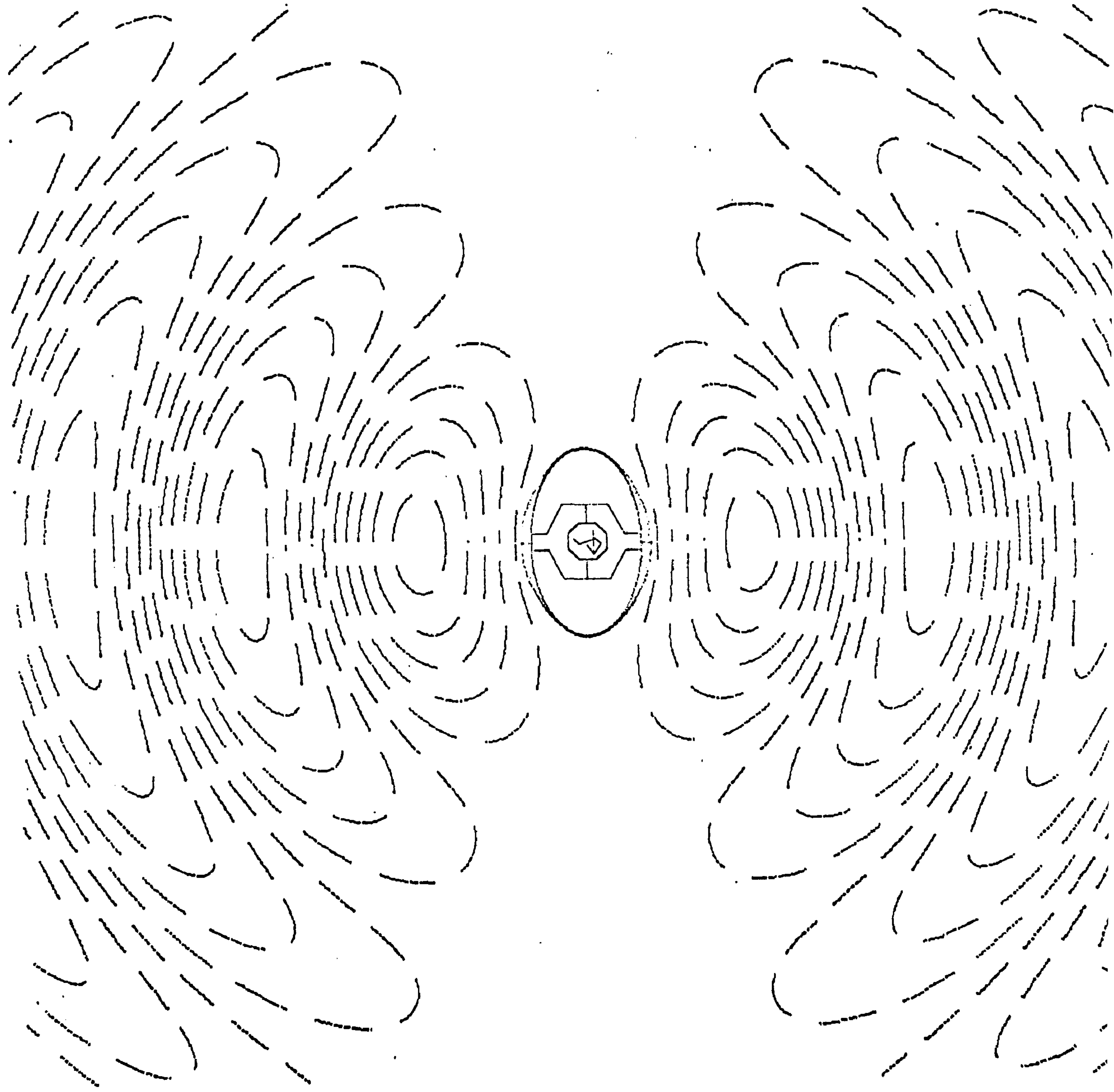


FIGURE 13

SPHEROIDAL RADIATOR WITH MOVING SEGMENTS movie is run is that of the letters coming towards the viewer out of the screen.

Table 2 summarizes the computer time required, length of magnetic tape, number of frames, and running time for each scene in the spheroidal radiator example. The movie was calculated in several segments which were then combined during the copying process. The final result was three reels of tape.

The finished product had a few minor faults due to programming errors and a major difficulty during the running of the magnetic tape on the SC 4020.

The minor errors were of a sort that can be corrected easily. For example, the ellipse did not shrink properly from the large drawn one to the center of the radiator. A change of one variable in a calling parameter could correct this. However, this points up the fact that each scene should be checked by hardcopy production before the computer time is invested in the final run. Also, the hardcopy gives the programmer an actual representation of his work that cannot be visualized from the numbers alone.

The major difficulty was that even though great care was taken in recording the magnetic tape (see the section on "program running operations" in Appendix B) there was still false information seen by the SC 4020 that resulted in many blank frames in the finished movie. Also, some of the information was lost resulting in a spotty picture.

TAPE	FILE	SCENES	TOTAL 7094 TIME	TOTAL WORDS	RECORDS	FEET OF TAPE	FRAMES	TIMES TO RUN	TOTAL FRAMES	MOVIE RUNNING TIME
I	1	Titles, Credits, Text Ellipse, Rotation, Generator	4.3	598,084	1869	655	1869	1	1869	1.29
	2	Transient	14.8	1,338,889	1351	1289	88	1	88	.06
	3	Ruler	3.6	241,746	245	246	35	7	245	.17
II	1	First half of one complete cycle steady-state	12.7	887,937	880	854	25	12	600	.42
	2	Second half of one complete cycle steady-state	11.2	884,396	872	850	25			
III	1	Moving segments for "D" field	17.5	880,359	884	848	72	12	864	.60
	2	THE END	.7	8,581	132	16	132	1	132	.09
		(Totals)	64.8 min.	4.84×10^6		4758			3798 = 96 feet	2.63 min.

TABLE 2

MOVIE PRODUCTION TAPE DATA

This was particularly evident in the closing scene when many of the letters in the words "THE END" were missing so that the complete title flickered on the screen. The only solution to this problem is to either do a careful alignment procedure on the SC 4020 just before the movie is to be made or to make a copy of the information on the magnetic tape at the installation where the SC 4020 is located. If OSU obtains a microfilm plotter, this should cease to be a difficulty.

Upon first viewing, it seemed that the movie was trying to do too much in too little time. A number of difficult concepts are being expressed in the movie, and before the viewer has had a chance to sort out his impressions, the scene has passed and a new idea is being presented.

However, closer examination showed that the timing of the scenes was badly out of balance, not because they were too short, but because the files on the magnetic tape were not processed according to the instructions sent with them.

The timing of the scenes of the spheroidal radiator was adjusted by computing one complete cycle and having this portion of tape run through the SC 4020 the required number of times for the desired length. However, due to the length of time required to process the more involved scenes on the SC 4020, the number of passes was drastically cut. The main scene of the radiator was run twice instead of twelve times and the scene containing the moving segments was run once instead of twelve times. The obvious solution to this problem is to make certain that when the magnetic tapes are processed, the instructions to give the correct timing are followed.

Some of the introductory scenes are a bit short. To lengthen these scenes, particularly the rotation of the ellipse, it will be necessary to recompute the first file of the movie tape. This is a small problem because the computer time used for these scenes was only about four minutes and a repeat should only take a little longer.

CHAPTER V

PRESENT AND FUTURE COMPUTER GRAPHIC EQUIPMENT

Summary

There are a large number of companies that make visual output devices to be used as peripheral equipment associated with a digital computer. However, most of these are not of interest with regards to this work because they have no provision for permanently recording their output.

The devices described below do produce permanent records and therefore can be used to make a movie of computer-generated output. They were chosen because they are representative of what is actually commercially available and of what is being developed in the computer manufacturer's laboratories. They cover the range of available tools that can be used to produce movies today.

Present equipment

The CALCOMP digital incremental plotter manufactured by California Computer Products, Inc. is a mechanical plotter that produces paper output in the form of graphs.[1] Like

[1]Bulletin No. 1758/February, 1966, California Computer Products, Inc., Anaheim, California.

most computer-graphic equipment, it was designed to create engineering output and not pictures. It is limited to drawing lines under computer control although quite a bit of software programming has been developed to create symbols, label graphs, and scale plotted points. The OSU computation center is equipped with one of these plotters. A body of subroutines is available as part of the computer library.

The CALCOMP plotter is useful when it is desired to plot out single frames of electromagnetic fields. It was originally planned to use this type of plotter to make the spheroidal radiator example using the following technique. The mathematical description of the radiated field would be used to calculate the field streamlines at one particular time. The commands for the particular plot would be punched on cards by the IBM 7094 and these cards would be used to drive the plotter. The plotting would be repeated at different times to complete the cycle and the total number of scenes would be animated using regular film techniques. In fact, a length of film was produced in this manner but it was not satisfactory for several reasons.

Because of the limitations on mechanical plotting speeds, up to forty-five minutes of computer time was required for each picture. The time increment to give smooth motion without jumps was in a range that meant at least thirty-two different plots had to be made for each scene. The film animation techniques proved to be very time consuming and difficulties were encountered in accurately placing each frame as it was photographed to give proper register when the movie was shown.

Another disadvantage is that this is an "incremental plotter" and therefore the commands generate a specified step length in only eight directions at forty-five degree angles. This puts a severe limitation on the amount of resolution that can be obtained unless the drawings produced are very large. Even with the first two by two and one half foot drawings, this resolution limitation could be clearly seen. The newer plotters are better in this respect (twenty-foursteps instead of eight) but it seems difficult to imagine producing any volume of film with a mechanical plotter.

The second commonly available device is the SC 4020 microfilm plotter, manufactured by Stromberg-Carlson, Inc., which reads magnetic tape or computer output to produce graphical, pictorial, or alphanumeric output directly on film. Since a complete description of this device is given in this work, details are not covered here. The heart of this device is a cathode ray tube that can be photographed directly with both the display on the face of the tube and the camera being controlled by computer output.

Because a cathode ray tube, whose beam is without inertia for all practical purposes, is used to perform the actual plotting, the limitations in plotting speed are not determined by a mechanical movement. Instead, the speed limitation is set by the rate at which the computer can transmit information from one place to another. As this transmission is being performed at computer speeds, the increase in plotting rates over a mechanical recorder is appreciable. In the spheroidal radiator example, some of the drawings took approximately forty-five minutes each. When the same plots were duplicated using the SC 4020, the total time to complete each scene was about five seconds.

Two other machines have recently come onto the market, one of which is a serious contender for the spot now held by the SC 4020.

The CALCOMP 835 electronic digital plotting system is a new plotter that is meant to complement their mechanical plotters rather than replace them.[2] The digital logic has been constructed to make the cathode ray tube operate as an incremental device with the direction and position specified for a preset step size. The interesting thing about this plotter is that it is compatible with the current software programs available at O.S.U. for the mechanical plotter. The only difference is in speed, up to 100,000 steps per second instead of 200 steps per second achieved at present. This unit has another advantage in that it is the least ex-

[2]Bulletin No. 188A/Januray 1966, California Computer Products, Inc., Anaheim California.

pensive unit now available (\$76,000) that could be considered for making movies. Unlike the other plotters described, it does not produce hardcopy, so the 35 mm film has to be developed before a print can be made.

The B-L 120 plotter manufactured by the Benson-Lehner Corp. is almost a direct replacement for the SC 4020.[3] Its design was obviously influenced by the other because magnetic tapes made for the SC 4020 can be run on the B-L 120 providing one minor precaution is observed.[4] According to the manufacturer, this plotter has approximately twice the speed of the SC 4020 and in addition, logic is provided to check for tape errors. This is the ability to run a record over again if the machine hits a command that it does not recognize. The SC 4020 under similar conditions puts an error mark on the edge of the film and then skips to the next record. This error can occur if the magnetic tape was produced in the presence of noise so that the bit levels on the tape are obscure. All of the description in the chapter on the SC 4020 also applies to this unit. One detail is interesting, the announced price for the B-L 120 (\$100,000) is approximately one-fourth that of the SC 4020. This may be a reflection of the fact that the SC 4020 has been

---

[3]Bulletin No. IDS-14/July 1965, Benson-Lehner Corporation, Van Nuys, California.

[4]A record gap (a blank space on the tape) has to follow an advance film command on tapes that will be run on the B-L 120. This is not necessary with the SC 4020.

the only machine of its type in the market and if so, a drop in price should occur.

The DD 80 microfilm plotter manufactured by Control Data Corp. of Minneapolis [5] is a good example of a "third generation" machine. The SC 4020 and the BL 120 can be considered as a "second generation," and the CALCOMP plotter as a "first generation." As such, the DD 80 has all of the features of the other plotters, but it has in addition several advance options that make it especially attractive for making movies. Its control logic operates on a different principle that results in much less data being transmitted for the same amount of information to be recorded. This is significant in movie making where the computed data is liable to be voluminous and the cost of the computer time is far greater than that of the plotter.

In the SC 4020, a single input number of thirty-six bits is used to represent both the operation and the location where this operation is to take place on the screen. In a third generation device, the operation code and the position codes are separated. An operation code is sent to the machine that tells it to perform this operation until another operation code is received. For example, the code might be to plot a dot on the screen. Thereafter, until the receipt of another operation code, all information

[5] Model DD 80H Operation and Programming Manual (St. Paul, Minn: Data Display, Inc., 1965).

fed to the display device will be location only. Some idea of the time that is saved by using this method can be gained from the fact that in the spheroidal radiator movie some scenes contained fifteen thousand dots plotted in sequences. In the SC 4020 the status, namely that a dot was to be plotted, had to be determined for each of the fifteen thousand commands. On the other hand, where the operation code is used to set a status, three locations can be stored in each of the following command words. This compression of information plus the fact that the operation code does not have to be interpreted each time will result in a four-fold increase in speed at a minimum using the same hardware. Actually due to increased machine speeds and better information transmission handling methods, the increase in speed is more likely to be twenty times as fast. The SC 4020 will plot about twelve thousand characters a second, and the BL 120 about twenty thousand characters a second. In contrast to this, the DD 80 will plot up to two hundred thousand characters per second.

Another very important feature of the DD 80 is that it has a dynamic memory rather than an input buffer. An input buffer is a temporary store that only holds information for a short period of time. If the information is not recorded immediately, it is crowded out by new information coming in and is lost as far as the recorder goes. On the other hand, the dynamic memory allows the programmer to accept a certain amount of information, depending on the size of the memory, and hold it in store while writing it out on the face of a cathode ray tube display. Each command in the memory is cycled at a rate to keep a continuous picture on the CRT display. By placing several frames in the memory and cycling the full amount of information present, a scene from a movie can be checked before the film is recorded.

If an error is discovered or the programmer wishes to make a change in a scene, there are three methods available to insert changes once the data has been placed in the dynamic memory. The first is a teletypewriter that can be used to modify any command word in the memory. The effect of any changes made can be seen immediately on the large CRT display. The other two methods are similar and only the means of actuating the display are different. The display comes equipped with both a light pen and a trackball. The light pen has a two position detent switch mounted on it. The first position turns the light on to form a spot on the screen. When the switch is pressed into the second detent, the position of the spot is inserted into the memory and the last action that was typed in takes place at that location. The light pen can be used to delete or replace characters, draw lines, and do general editing before the scene is recorded. The trackball is a phenolic ball mounted in the desk in front of the display screen. The top of the ball sticks slightly out of the top of the desk and is connected to two digital encoders that keep track of the X and Y movement of the ball. When a switch is thrown, a circle with a point in the center appears on the screen corresponding to the X and Y encoder positions. As the ball is moved, the circle will follow. Another switch serves the same purpose as the second detent in the light pen.

It is obvious how helpful these features would be in the production of a movie. If the dynamic memory were big enough, not necessarily the case with the DD 80, each scene of the movie could be edited before it was recorded without the time and expense of having to compute the entire scene over to correct a few minor mistakes.

Future trends in computer graphics

The trends of the future can be clearly seen in the computer graphics equipment that now exist in prototype form at both the university and the computer manufacturer. Table 3 summarizes some of the efforts in this area. In each case, these have to be considered as prototype efforts because, unlike the DD 80, they are not being offered as a commercial item by the people involved. There are undoubtedably other systems in existence but these have received the most publicity.

NAME	FULL NAME	SPONSOR	COMPUTER INPUTS	MANUAL INPUTS	FILM OUTPUT
DAC-1	Design Augmented by Computer	General Motors	ON LINE (2-7094's)	Light Pen, Electric Pencil Key Board, Card Reader	Yes
Graphic 1	Graphic 1	IBM-Bell Labs	ON LINE	Light Pen Key Board	No
MAGIC	Machine for Automatic Graphics Interface to a Computer	NBS-NASA	ON LINE	Light Pen, Switches Key Board	No
GPDS	General Purpose Display System	System Development Corp.	ON LINE	Light Pen, Rand Tablet Key Board, Teletypewriter	Yes
COED	Computer Operated Electronic Display	Bendix	Magnetic Tape	Track Ball Switches	No
Sketchpad	Man-Machine Graphical Communication System	MIT-Project MAC	ON LINE (Special purpose computer) Paper Tape	Light Pen, X-Y Knobs, Many Switches (not key board)	Yes

TABLE 3

EXPERIMENTAL COMPUTER GRAPHIC SYSTEMS

Two trends are evident. The first is that efficient systems, whether they are just for research or can also make movies, will have to be run ON-LINE in real time with a computer that has a very large memory in order to try the effects of different inputs as the results are being produced. One of the main difficulties to date is that with even the largest scale systems (GM's two 7094's, for example), the response is very slow because of the vast amounts of information that have to be moved from place to place in the computer.

Some idea of the difficulty can be obtained from BEFLIX,[6] a computer language that has been developed by Bell Telephone Laboratories for the production of animated movies by a computer. In this particular language, the cells on the screen of the SC 4020 are treated as a television raster with a location reserved for each cell in the IBM 7090 being used. Two complete frames take up the entire storage, so a large disk file had to be added where up to 440 frames can be stored. When peripheral equipment such as disk or tape is involved, the needed time is increased by a far greater amount.

The solution to this problem may be found in the concept of parallel processing now under investigation at the

[6]K. C. Knowlton, "A Computer Technique for Producing Animated Movies," Proceedings of the Spring Joint Computer Conference, 1964, Spartan.

University of Illinois. Instead of having only one central processor which does one arithmetical or logical operation at a time, this computer would have a hundred or more processors so that large increases in throughput speeds can be obtained without a corresponding increase in hardware speeds.

The other item common to most of these systems is the light pen used to edit and modify displays on a cathode ray tube. A control of this type would seem to be a necessity when making movies to save on the amount of programming effort required to make changes.

An interesting variation on the light pen is the Rand Tablet. This is a plate with 1024 x 1024 wires imbedded in it with just the ends showing. When the circuit is completed by running an electric probe over the surface, the computer stores the circuits which were completed and controls the cathode ray tube to follow the path of the probe. Therefore, it is very easy to insert lines and curves into a display, but they will not have the linearity of a programmed line. In addition to the Rand Tablet, software is needed to make lines straight and circles round after they have been drawn.

The spheroidal radiator example was produced by very conventional methods of programming. A program for each scene was written, typed on cards, debugged by several runs through the computer, and then joined to the larger program being constructed. In each run, the feedback to correct a program was delayed by hours spent waiting for it to be returned.

As the Sketchpad engineers have pointed out,[7] a display system with a cathode ray tube allows one to communicate with the machine by means of line drawings instead of written words. It is obvious how much time can be saved when an object can be drawn rather than described. Since computer generated movies involve highly complex and abstract pictures, a visual display would seem to be almost essential in order to determine the program being made.

By using a type of input-output where there is very close cooperation between the man and the machine, the burden is placed on the programmer to write software that is man oriented rather than machine oriented. This, of course, will aid the movie producer because he can now concentrate on what the scene should show and not on the programming or debugging details.

However, there are still many unsolved problems. The one of speed was mentioned above. Although the display device is essentially a two dimensional one, several people are experimenting with three-dimensional movies, either by

[7]I. E. Sutherland, "Sketchpad: A Man-Machine Graphical Communication System," Proceedings, AFIPS, Spring, 1963.

using perspective views or a stereoptican process where the viewer wears polarized glasses. The newer SC 4020's are equipped to plot dots with eight different shades of gray. This means that the programmer could get away from the type of drawing that can be produced on a blackboard and into drawing where halftones are important. For example, the direction of motion of an object could be shown by leaving a trace that slowly fades out after the object has passed.

Another area that has not been touched is that of color. One way that this could be handled would be to have a wheel in front of the movie camera with filters containing the primary colors in it. The camera would be loaded with color film and each scene would consist of several exposures, one for each color. The program would control the position of the color wheel and the result would be like that of color printing.

At the present time, movies are produced by having one computer do the necessary calculations and another expose the film. Consideration should be given to have the camera as an integral part of the computer. It would serve as the final storage for information being generated. Particularly in the case of recurring or cyclic motion, the scenes for one cycle could be stored in a disk memory associated with the computer and then recorded over and over again to produce the required footage of film. Both disk and tape have very fast access times if the information is going to be read in serial form so that the unit does not have to go through a search procedure. The disk would be preferable where modifications were going to be made via a CRT and light pen because the response time to place changes on the disk would be faster then that of tape.

Unfortunately, the cost of implementing these ideas would be very high. The hardware, in large parts, exists today, but the programming effort would incur an amount of time and money that is beyond the reach of most organizations. Perhaps the cost of such a machine could be justified by the production of movies in many fields and not just electrical engineering. In any event, a university would seem to be the ideal place in which to have such an endeavor considering that experts would be available in all necessary fields from antennas to human factors.

CHAPTER VI

SUMMARY AND CONCLUSIONS

Summary

The use of microfilm plotters to produce computer-generated motion pictures has opened a new area in explaining electrical engineering phenomena. The technique described in this work will be especially useful in the classroom in helping the student understand the reasoning behind the mathematics placed on the blackboard, and it can also be of assistance in research where the movies produced would serve to reduce pages of figures to a few clear pictures.

The computer is beginning to play a dominant role in electrical engineering education and research, and while the labor that the computer replaces allows for more complicated problems to be handled, the output is in a form that tends to overwhelm the researcher who is trying to interpret the results. It is hoped that this work will help reverse the trends that the computer has set in motion and that computer-generated movies will find common use in the classroom and laboratory. It is especially applicable to electromagnetic fields and the applications described were primarily in that area.

The technique consists of using a large computer (such as an IBM 7094) to calculate the desired results, such as the points of constant potential in the electromagnetic field produced by an antenna, and then converting these points into commands that can be interpreted by a high speed microfilm plotter. The parameter "time" is varied and the data is plotted frame by frame for each time increment. The final result is a series of plots on sixteen millimeter film that can be run as a movie to show the motion involved. Up to now, the researcher had two choices: either cartoon animation, or a still frame showing the subject at one instant in time while the associated text described the motion. As some of these motions are complicated, there are times when even a lengthly description does not get the point across. It is in these cases that being able to produce a motion picture would be invaluable to the student.

The number of subjects in electrical engineering to which this method could be applied seemed to be greater in the area of electromagnetic fields, so the possibilities were considered mainly from that field. These included transients on transmission lines; plane waves of different polarizations hitting a dielectric slab at various angles; fields in wave guides; and the radiation from antennas, diffraction, and scattering.

In order to study the concept of computer-generated movies, a three-minute example entitled "The Resonant Spheroidal Radiator" was programmed. This was a complete movie from the title to "THE END" and it was an attempt to show both what the subroutines could do and more importantly, what the techniques could do for electrical engineering.

In the course of writing programs that resulted in the movie example when run on the SC 4020, many limitations were encountered that were due to the machine and not the programming. For example, it was impossible to edit any pictures before committing them to film. Therefore, the computer-graphic equipment now in prototype form was investigated to determine what future trends might help the computer movie programmer.

One point should be made. The particular programmer cannot use these subroutines to support his own lack of understanding of the problem on which he is working because the input data to the subroutines must be the correct solutions to the problem being considered. However, under this limitation, it should be able to accomplish a helpful trend in electrical engineering. At the present time, most effort is directed towards determining the mathematical model from the physical theory until the reality of the system is lost in the mathematical details. When one is going to program a movie of physical action from the mathematics, one is forced to think back to what one is actually trying to explain. This should help greatly in eliminating the confusion that can easily arise between mathematical analysis and experimental evidence.

Conclusions

The first conclusion reached in the course of this study is that computer-generated motion pictures have a definite place in electrical engineering. The mass of formulas and figures presented to the student in the classroom may be meaningless, but if a movie can be shown that illustrates the physical aspects of the mathematics, a greater understanding and appreciation of the reasons behind them can be gained. The main requirements are that the subject can be expressed mathematically and that the motion be important enough that a movie will show things that would be missed in still pictures.

The importance of motion can be expressed in two ways. The first of these is motion in time. For example, this motion can be used to express the form of electromagnetic fields far more effectively than by any other method. In fact, this seems to be one of the best subjects for these computer-generated motion pictures.

The second consideration is to use the motion to explain the effect of a parameter other than time in a system. A good presentation would be to show the effect that varying the position of the poles and zeros in a root locus diagram has on the stability of the system under consideration.

In electrical engineering research, the methods developed in this work can be used to gain time in the processing and understanding of results that can be checked by the computer. For example, a motion picture of a plasma field might be compared to a motion picture of the mathematical model of the same field to see if the analog is correct. Hand plotting would take far longer and would not be nearly as effective.

The second problem that was solved in this work was how to give programmers at OSU a working system to produce motion pictures from computer output. Although the future holds great promise in the developments in computer-graphics, at the time the movie example was being produced, the available graphic output equipment was pratically limited to one device: the SC 4020 microfilm plotter by Stromberg-Carlson. This device was originally designed to produce engineering graphs or tabulated data in the form of thirty-five millimeter microfilm or 9 x 9 inch hardcopy. Therefore, an extension of the Ohio State University Numerical Computation Laboratory operating system was written so that the SC 4020 could be used to produce movies at OSU. This extension consists of about fifty subroutines that enable a programmer to concentrate on programming of the movie itself and not on the details of how the SC 4020 operates.

The main reason that such a device can be considered for making movies is the speed with which it plots. It can draw vectors or plot points at 12,500 a second, so in the course of about fifteen minutes an amount of film can be produced frame by frame that would take days of animation by any other technique.

The SC 4020 has a limited range as a computing device and in fact the only thing it can do is either plot a character at a specified location or draw a vector from that location in a specified direction. Since it was assumed that the people using this device to produce movies would know how to program the desired results but would not necessarily know how to use the SC 4020, software was written and debugged to remove this chore from the programmer's hands. Therefore, one of the accomplishments of this work is a body of subroutines that will accurately create the desired pictures. All that the programmer has to do is calculate touching points along any line to be drawn or, if straight lines will suffice, the end points of these lines. The subroutines will take care of the rest from calculating the proper plotting commands to writing the magnetic tape used by the SC 4020.

One difficulty overcome in writing these subroutines was a computing language conflict. A subroutine library existed in FORTRAN II for producing engineering output and it was desired to use this as a model for the movie-making subroutines. Using this library saved several man-years of programming, but a conversion had to be made to SCATRAN, the language used at OSU. The input-output routines were completely rewritten due to the difference in handling magnetic tape between the two systems and conversion programs were written so that the available FORTRAN subroutines directly useful in movie making could be used. In addition, details are given so that the programmer can construct any desired symbols or figures out of straight lines without the bother of having to calculate the position of each break-point in the drawing.

The final result was extremely successful. At the present time, OSU not only has the capability of producing a magnetic tape that can be processed by any SC 4020, but several other plotters are coming on the market that use the same command code words. The BL 120 by the Benson-Lehner Corp. will run SC 4020 tapes subject to the minor restriction that any "advance film" command be followed by a record gap.

The complete programming and subroutine system for this plotter is described in great enough detail in Appendix B of this work that it would be possible for a programmer to duplicate the system without difficulty.

During the programming of the example described before, a number of problems were encountered that could not be foreseen. To save grief and difficulty on the part of those who might use these subroutines, a section on "program running operations" was included in this appendix.

In addition to the operating system described in Appendix B, the listings for the necessary changes so that these subroutines could be run under FORTRAN have been included in Appendix C. This would be a great convenience to someone trying to use them outside of OSU. Finally, there will undoubtedly be plotters coming on the market that use a different code than the SC 4020. Even though these could not be used directly in that case, they can still be used as a useful guide as to what is required for making movies.

The next problem to be solved was to show first that the body of subroutines was capable of actually producing a movie and, more importantly, proving that computer-graphics have a definite place in the production of movies of electrical engineering phenomena. The final result was a three-minute movie entitled "The Resonant Spheroidal Radiator" which was made to serve as a demonstration of both the use of these programs and of the basic ideas behind them.

The spheroidal radiator as a radiating antenna has been investigated in some detail by Stratton, Chu, and Ryder, but their final results left something to be desired for the average reader.[1] Therefore, this seems to be an ideal subject: namely, it was mathematically complex, time was a basic parameter, and, at resonance, the streamline equations gave an interesting picture without an excessive amount of programming. Incidentally, all of the programming was done by

[1]Chu and Stratton, loc. cit. Ryder, loc. cit.

one person, and therefore it is also a good example of what a researcher interested in a particular function of antenna design could do without a team effort.

Up to now, this effort has resulted in endeavors that are primarily of interest in the classroom, but the above example is just a start on something that could show the usefulness of computer-generated movies in electrical engineering research. The resonant case was picked for the spheroidal radiator because the solutions that were available were reasonably tidy in closed form. However, the authors who have investigated this problem also give series solutions to the differential field equations for all frequencies and it would be extremely interesting to see what the near field, in particular, looked like as the antenna was driven by a range of frequencies passing through resonance.

The last scene of the movie was particularly interesting because it shows the results of two different motions at the same time. The lines indicating the field are calculated from lines of constant potential and also show when the magnetic field has constant phase. In addition, the displacement current is shown by means of segments that move along the streamlines. They reverse for each radiated cycle to indicate the alternating field. It is felt that this concept could be extended to a number of different cases in electromagnetic fields where it would be desired to show both the electric and magnetic fields by means of the displacement current streamlines.

The movie was successful except for some difficulties in processing. These were the fault of the particular SC 4020 used and not the programming, but it would be necessary to make certain that all errors and noise are removed before a perfect copy could be achieved.

When the scenes in the movie are compared with the mathematical formulas used to develop them, it is felt that this demonstration definitely proves the value of computer-generated movies in the human interpretation of exceedingly complex electrical engineering phenomena.

The fourth problem arose as a result of the work on the demonstration movie example. It was found that the SC 4020 is extremely limited in concept because it was originally meant only to make microfilm documents from computer tapes. Even with the addition of extensive software tailored for producing movies, there were a number of features that could have been desired to help the programmer. Therefore, a number of computer-graphic display systems were examined to see what was in the future as far as the present state-of-the-art is concerned. The conclusion reached was that it would be possible to build a device that could accept programs, display the result frame by frame, and then immediately produce an animated movie, but such a device would have to have both a bigger and faster memory than any machine on the market today.

Display equipment is improving rapidly, but no displays have yet been offered that are expressly made for producing movies. It was found in the course of this work that the ordinary microfilm plotter would not be sufficient, principally because of the registration problems with the film camera. It was felt, however, that even with the small survey that was made, computer-graphics will become one of the fastest growing fields in the area where computers and education meet.

In conclusion, the computer is swamping electrical engineering researchers and students with its output, and the production of computer-generated movies would seem to be one of the best ways that a balance can be restored. The technique developed and demonstrated in this work will serve as a valuable accessory in promoting understanding in both electrical engineering classrooms and research.

APPENDIX A

THE FAP LANGUAGE AND DIFFERENCES BETWEEN FORTRAN AND SCATRAN

Summary

This appendix gives the details for solving a conflict in programming languages that occurred when SC 4020 subroutines written to be called from FORTRAN main programs were used as a model for subroutines that were to be called in the OSU SCATRAN language. Both FORTRAN and SCATRAN use FAP (FORTRAN Assembly Processor) as a basic machine language but there are enough differences between the two systems that a subroutine written to be called in one language cannot be called in the other. A brief description of the FAP language is given covering only those areas that are affected by the above differences. Two programs are then compared so that the calling structure to the subroutines can be determined. Finally, a sample conversion program is described to enable the programmer to run subroutines written outside of OSU under the SCATRAN system.

The FAP language

The FAP language is a basic machine language used by the IBM 7094 computer. No matter what source language is used, the compiler will eventually write the program in FAP so that it can be loaded into the computer. Because it is an assembly language and not a compiler language, it is extremely flexible and therefore preferred for writing subroutines that are going to become a part of the operating systems library. Also, the average programmer who is thoroughly familiar with FAP can write a tighter program than is possible by using a source language and having the computer do the compiling into FAP. This is especially important where there are a great many subroutines to be loaded and the core space may be limited by the size of the main program.

A detailed discussion of the FAP language will be left to the programming manuals,[1] but some of the features of this language must be discussed in order to understand both the operation and the difficulties encountered in writing the SC 4020 subroutines. Two features will be discussed in some detail, namely how an entrance and exit is made from a FAP subroutine and the principle of indirect addressing.

At the beginning of each FAP subroutine there is a list of entry names to that subroutine. Any of these names can be used in another program to call the subroutine.

The next item in the statement listing lists any subroutine names called by this subroutine. Every subroutine

[1] _7094 Principles of Operation, File No. 7094-01_ (New York: IBM Corporation, 1962).

name has six letters (including enough blanks) that are referred to as a transfer vector. The loader takes all of the transfer vectors and compares them one by one with the BCD name associated with an entry point address. When a match is found, the transfer vector is replaced with a transfer command to the address in that particular subroutine. For example the transfer vector 74 41 46 22 45 34 equals (JOBN). This BCD twelve digit number represents one 36 bit word when the program is loaded in the computer, but it is not the number that appears in that location when the problem is ready to be run. Instead a transfer to the current entry point appears. If the subroutines are shuffled into new locations, the next time the program is run the loader will supply the new addresses as easily as it did the old.

The programmer does not have to specify the transfer vectors for each subroutine the way he does for the entry points. He refers to the subroutine name by either one of two calls:

```
CALL    Subroutine name - 6 letters or less
TSX     $Subroutine name - 6 letters or less
```

The word CALL is equivalent to the word TSX (transfer and set index) and it is a matter of individual preference as to which one the programmer uses. However, when the TSX is used, the subroutine has to be prefixed with a dollar sign so that the assembler will know that the name is not an undefined variable in the program on which it is working.

The assembler will automatically set up the correct transfer vectors at the start of the program.

It was stated above that the loader attempts to match the transfer vectors with the entry points on a one-to-one basis. If the loader does not find an entry point in the list obtained from the input deck, it looks in the OSU system library and loads the subroutine it needs from there.

Once the entry point has been called, it is necessary to be able to return to the called point after leaving the subroutine. This is accomplished by means of storing the calling address in an index register, namely index register IR4.[2]

It is obvious that a subroutine can be left by calling another subroutine and then exiting through the transfer vector, but a more common method is to return to the next instruction in the main program after the subroutine call. This is made possible by the fact that either CALL or TSX,4 stores the address of the transfer command in index register four before the transfer is made.

Therefore, IR4 can be used to determine the return from the subroutine. One of the powerful features of the FAP language is the ability to tag an address. Consider the FAP

[2]There are three index registers available in the IBM 7090 computer: IR1, IR2, and IR4. IR3 is used for a bookkeeping function. The IBM 7094 computer has twice as many available, but these subroutines will run on either machine because they were first written with only three index registers in mind.

statement that occurs quite frequently at the termination of any FAP subroutine:

```
TRA     1,4
```

The one refers to the number that is going to be placed in the address of this instruction when the subroutine is loaded, and the four is placed in what is known as the tag bit. Since TRA is a transfer, this statement would be read as "transfer to address one tagged by four". This means that the contents of IR4 are added to the address "one" before the transfer is made. Assume that the subroutine was called from location 15455. When the above statement is executed, the transfer is made to location 15456. Notice that the subroutine had no knowledge of where it was called from. Address 15455 could have been any location in the computer.

Very often it is desired to use index register four in a subroutine when calling yet another subroutine, and the calling address may be lost. Since the address in IR4 is the only way the subroutine has of knowing where to return in the main program, it is necessary to save this address before using IR4. Therefore, very often a typical subroutine will have the following form:

```
ENTRY  SXA     (X4),4
- - - - - - - - - - - -
subroutine statements
including use of IR4
```

```
to call other sub-
routine
- - - - - - - - - - -
(X4)   AXT     **,4
       TRA     1,4
       END
```

ENTRY is the name of this subroutine. The first operation (SXA) saves the contents of IR4 in the address (X4).[3]

At location (X4) there are two asterisks in the position for specifying the address. This means that the address is to be supplied by a statement earlier in the program, namely the SXA.[4]

The subroutine is then executed including any calls to other subroutines that will alter the contents of IR4. When (X4) is reached, the address in this location is placed in IR4.[5] Since this is the address of the calling statement, the TRA then transfers to the next statement after this. Both the subroutine and the calling statement can be at any space in the computer store and neither has to know the location of the other in order to communicate between the two.

---

[3]This location could have any name up to six BCD characters. (X4) is just a convenient one.

[4] When troubleshooting a program, the address stored in the AXT statement of a subroutine is very helpful. If the subroutine was never reached, it will still contain zeros and if it was, the location of the last call can be determined.

[5]AXT = address to index, true.

In order to understand how the data of one subroutine or program can be used in another subroutine, it in necessary to understand how indirect addressing works. Approximately half of the operations permitted on the IBM 7094 (about 200) are of a type that requires the machine to do something with the data that is stored either in a particular location in storage or in the accumulator or multiplier registers. For example, the instruction

```
CLA     ALPHA
```

in a program clears the accumulator and then places the information in location ALPHA into the accumulator.

When it is desired to use the indirect addressing feature, any of the above commands can be modified by the addition of an asterisk to the three letter operation code. For example,

```
CLA*    ALPHA
```

performs an entirely different operation than the previous command. In this case, the computer instead of taking the data in ALPHA and placing it in the accumulator looks for the address stored in ALPHA. It then performs the "clear and add" operation, not in ALPHA, but on the data stored in the address stored in ALPHA. Hence the name "indirect addressing." This characteristic extends the capabilities of the computer by a tremendous amount because very often it is not known where a piece of information is stored but the location of the address of the data is known. A perfect example of this is in a data array where a series of numbers are stored in the computer's memory. By placing the address of the first word of the array in a specified location, any word in the array can be located merely by adding the number of the word to the address. Assume that the first word of the array is in address 14401. If this address plus the number of the desired word are placed in address 14400 (i.e., for word 22, 14423 would be placed in 14400), it is only necessary to apply an operation code followed by the asterisk to address 14400 to use data in 14423. Without indirect addressing it would be very difficult to write a compiler that could handle subscripted variables in the source language. It is this capability and that of being able to modify addresses with a tag that separate the large computer from the small one. Indirect addressing and index register tagging are two ways to increase program speed by an order of magnitude without increasing hardware speed.

However, indirect addressing is of vital concern to the programmer in another area when he is writing subroutines that either use or supply data to the main program. Needless to say, most of the SC 4020 subroutines are of this type. In order to explain how indirect addressing is used to transfer data into a subroutine, it is necessary to explain the use of "calling parameters." They are also called the "arguments" of a subroutine because they are written in the same fashion as the arguments of a function. Consider the following statement in a SCATRAN source program:

```
CALL SUBROUTINE()=EXMPLE.(X,N,1,TABLE.)-
```

The name of the subroutine is EXMPLE.. The period after the subroutine name is the SCATRAN method of keeping track of subroutine names for the transfer vectors between programs. Any symbol in the brackets is a calling parameter; i.e., X, N, 1, TABLE. are all arguments of the subroutine EXMPLE.. It will be noticed that they can be floating point numbers (X), fixed point numbers (N), literals (1), or even subroutine names (TABLE.) subject to the limitation that a subroutine cannot call itself. This notation for calling parameters is quite general for different computing languages as will be shown when SCATRAN and FORTRAN are compared.

In order to understand why indirect addressing is important in this case, it is necessary to look at how the compiler would handle the above statement. It would appear in a core dump of the storage as the following set of words:

```
0074 00 4 14415
0000 00 0 27540
0000 00 0 27541
0000 00 0 14401
0000 00 0 14414
```

The 0074 is the operation code for "transfer to the address in this word (14415) and set an index register to the address of this word" (TSX). The index register to use is given by the tag of 4 for IR4. Address 14415 is the location of the entry point for subroutine EXMPLE. and the following four addresses specify the locations of the variables X, N, the location of a word containing the literal "000000000001" and the location of the entry point to the subroutine TABLE. respectively.

Note that only the address of the data is given, not the data itself. Since only the address is given and since the address of the calling statement is not known but has been placed in IR4, it can be seen that both tagging and indirect addressing will have to be used to obtain the required data once the subroutine has been entered. To continue this example, the subroutine EXMPLE. might be written as follows:

```
Address in      Machine Word         FAP Language
Computer                             Statement

14415           0500 60 4 00001      EXMPLE     CLA*      1,4
14416           0601 00 0 14425                 STO       ALPHA
14417           0500 60 4 00002                 CLA*      2,4
14420           0601 00 0 14426                 STO       ALPHA+1
14421           0500 60 4 00003                 CLA*      3,4
14422           0601 00 0 14427                 STO       ALPHA+2
14423           0500 60 4 00004                 CLA       4,4
14424           0621 00 0 14430                 STA       ALPHA+3
14425           0020 00 4 00005                 TRA       5,4
14426           0 00000 0 00000      ALPHA      PZE
14427           0 00000 0 00000                 PZE
14430           0 00000 0 00000                 PZE
14431           0020 00 0 00000                 TRA       **
```

Each of the CLA*'s operate in the same manner. The tag

4 means that the number in the address of the instruction is added to the address in IR4. The asterisk means to clear and add the word whose address is in the address specified above. For example, the first instruction is CLA* 1,4. A transfer is made from the main program to the address of EXMPLE.. The data in the address given by the next machine instruction (27540) is placed in the accumulator. The STO stores it in location ALPHA. The fourth CLA is not used for indirect addressing because it is the address itself, i.e., the address of the entry point to TABLE, that is desired. The following STA stores this address in the third location after ALPHA. The last command in the subroutine is a transfer (TRA) that will jump back to the address in IR4 plus five locations. This is the next instruction in the main program when the four calling parameters are accounted for. The final result after going through the subroutine is:

ADDRESS	CONTENTS
14426	Value of X
14427	Value of N
14430	000000000001
14431	0020 00 0 14414

and control has been returned to the main program. An additional set of commands could operate on the stored data with the aid of another subroutine call. The reasons for the emphasis on this particular area of programming will be shown when the differences between SCATRAN and FORTRAN are discussed.

Reasons for converting to SCATRAN

There were three main difficulties with using the North American subroutines at O.S.U. The first one was that the North American operating system has an extensive bookkeeping program that is tied into all of their subroutines. When their computer executes an SC 4020 program, in addition to generating the desired output data, it counts the number of frames of hard copy made, the number of frames of film, the programmers name, the date, department number, time on the computer, etc. Since all of these items were handled in different places in North American's "library," when an attempt was made to run them at O.S.U. over forty loose ends showed up that prevented the program from being run at all. Therefore, the first job was to uproot the bookkeeping functions from each subroutine while making sure that the removal did not destroy the continuity of the subroutine.

The second difficulty involved the input-output equipment available at North American versus that at O.S.U. Since input-output equipment is very expensive, the tendency is to buy just what is needed at a particular installation. In this case, North American has an IBM 7090 computer and O.S.U. has an IBM 7094. The 7094 is a larger machine but it is also compatible with the 7090 so that any program written for the 7090 will run on the 7094. However, this is not true with regards to the available input-output equipment. Although North American has a similar computer, the nature of their computing load requires that their input-output equipment be far more extensive than that of OSU's larger computer.[6] The final result was that when the basic subroutines were run at OSU, the computations were performed, but the first time a piece of data was to be written on an output tape, a tape unit was called for that did not exist at OSU. The program would halt at that point and because the computer had entered the IOS (input-output supervisor) subroutine of OSU's operating system library and not in one of the subroutines that had been supplied to it by the programmer, the reason for the machine halt was very well hidden. This was one of the many troubles encountered in this area.

The third difficulty was by far the most serious one. It was stated above that two compilers are in use at OSU, namely SCATRAN and FORTRAN. As the subroutines had been written originally for FORTRAN calls, it was decided to use this language to develop the operational software for pro-

---

[6] The SC 4020 itself is a good example of this. Each North American Aviation operating division is equipped with one at a cost of $400,000, while OSU's only plotter is a slow mechanical one at a cost of approximately $8000.

ducing tapes at O.S.U. This objective was accomplished very successfully from the point of view of producing a tape that would run on any SC 4020 but certain obstacles appeared that detracted from this success.

In theory, the two available operating systems are equal in computing capacity with SCATRAN noted for the flexibility of its program statements and FORTRAN for running programs written elsewhere. However, in practice, SCATRAN is the language preferred by the Numerical Computation Laboratory and the difficulties in running FORTRAN programs were found to be almost insurmountable. The many minor ones include the fact that only a limited amount of FORTRAN runs are made each day and that very few consultants in the computing department are interested in FORTRAN or able to give advice on problems that the individual programmer may be having.

The minor difficulties could have been ignored but a major obstacle led to the decision to abandon FORTRAN as a language and to write subroutines that could produce SC 4020 tapes under the SCATRAN operating system. In addition, it was decided to keep the FORTRAN features of the subroutines as far as possible so that the programs could be run in FORTRAN if it was desired to do so in the future.

This major obstacle was the time that it took to process a program under FORTRAN. A test program was developed in order to check out the operating system when using the SC 4020 subroutines. It consisted of an ID frame and a series of frames (usually four) of asterisks in concentric circles that rotated between each frame. This program was run under identical conditions in both FORTRAN and SCATRAN using the identical FAP object decks for the SC 4020 subroutines. Incidentally, these FAP decks were assembled under the FORTRAN operating system so that if there were any incompatabilities they should favor the FORTRAN system. In the case of SCATRAN, the program took 5.46 seconds total execution time. The same program when run under FORTRAN took from a minimum of five and one-half minutes to a maximum of just over seven minutes. In SCATRAN the computer keeps track of the time being used for each separate stage of processing a problem and this is printed out after the problem is completed. In FORTRAN, this item was recorded manually by the operating personnel at the computer center and no breakdown could be obtained as to what was taking up the time in the FORTRAN as opposed to the SCATRAN run.[7]

[7]Although it could not be determined definitely as to why FORTRAN programs should take so much longer than SCATRAN, I believe that the great bulk of the time is used in two areas. First, the FORTRAN compiler is extremely slow compared to SCATRAN. This does not matter at most computer installations where a program will only be compiled once and then loaded from an object deck afterwards, but at a university installation most jobs take considerably less than a minute to run and several minutes of compiling time for each program cannot be tolerated. SCATRAN was written with the most emphasis

The seriousness of this situation can be shown by comparing the IBM 7094 computer time needed to make a hundred feet of movie film. In 100 feet of 16 mm movie film, there are 4000 frames. If under SCATRAN five frames took 5.46 seconds, then the 4000 frames could be computed in about 1.2 hours of IBM 7094 time. These five frames would take about five minutes under FORTRAN. For 100 feet of film, this would require approximately 67 hours. This would be impossible from both the point of view of obtaining time on the computer and of the cost incured in making only small amounts of film. For this reason, although the subroutines were operational under FORTRAN, the decision was made to run all future programs in SCATRAN.

---

on compiling speed and as a consequence, SCATRAN will compile an amount in one second that would take FORTRAN about one minute.

The other reason is due to the way a "non-system file" (the SC 4020 tape returned to the programmer) is handled under FORTRAN as opposed to SCATRAN. Under each system, a message is printed out for the computer operator to dismount a tape and return it with the program. At this point SCATRAN immediately goes into the next program and stops charging to the previous job. In effect, having thetape dismounted is free under SCATRAN. In FORTRAN however, the computer stops after printing out the message. You are being charged while the operator reads the message, finds another tape to replace yours, locates which tape unit is being used, replaces the tape and then starts the computer again manually to accept the next job. In each case, this procedure can take several minutes but in the case of SCATRAN it does not show up as very expensive IBM 7094 time charged to the job.

FORTRAN to SCATRAN conversion

The major task in the conversion was to determine the differences between the FORTRAN and the SCATRAN operating systems. These differences are due to the way the compiler is written and cannot be removed. The compiler operates as a total system and any changes that would be required had to be made in the subroutines under the programmers control and not in the operating system.

Although there are many differences between the two operating systems, fortunately there are only two areas that will effect the operation of the FORTRAN based SC 4020 subroutines when they are called in SCATRAN. Both systems use FAP as an object language so that the problem is, one: finding where the input and output data is stored in the main program and, two: the form of this data, particularly integer numbers.

Two identical programs were written for the two systems that have no function other than showing the differences that must be considered when using a FORTRAN object deck with a SCATRAN program.

Figures 14 through 17 give a SCATRAN test program and lower core dump followed by the same program and core dump for FORTRAN. Each program has four different subroutine calls that show the use of different calling parameters. By combining the two core dumps for the respective program

SOURCE LANGUAGE STATEMENTS

```
C               TEST PROGRAM FOR SCATRAN        @
                DIMENSION(X(3))@
  START         NR=1@
                IR1=1@
                IR2=496@
                XR4 = 35.367@
                X(NR)=3.5@
                CALL SUBROUTINE()=EXAMP1.(  )@
                CALL SUBROUTINE()=EXAMP2.(X(NR).NR.TABLE.)@
F  NAME         BRENTON R. GROVES@
F  DEPT         ELECTRICAL ENGINEERING@
F  DATE          7/15/66@
                CALL SUBROUTINE()=EXAMP3.(NAME.DEPT.DATE)@
                CALL SUBROUTINE()=EXAMP4.(IR1.IR2.IR3.XR4)@
                CALL SUBROUTINE()=ENDJOB.()@
                END PROGRAM(START)@
```

FIGURE 14

TEST PROGRAM FOR SCATRAN

```
GROVES, B. R.          JOB  FGJ500       09/13/66   089010  PAGE     7

SYSTEM DUMP 14400  TO    14536       AC OV OFF   MQ OV OFF   DV CK OFF   M TAG CN    FTRAP ON    ITRAP OFF

AC 0+000000014426  MQ +000000000000   SI 266621642360                                  SSW6 U       LITI 04123  LDDI 00000

  IR1 77777    IR2 00001    IR3 76510*  IR4 63250    IR5 61641*  IR6 62225*  IR7 61641*  (0) 00000      REL REG 00000
    -00001       -77777       -01270*      -14530       -16137*      -15553*      -16137*                PROT REG 77400 14400

14400  202700000000  206432737166  000000000760  000000000001  002100014532  002100014533  002100014536  002100014534

14410  002100014535  002100014554  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000

14420  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  202700000000  000000000000

14430  000000000001  000000000001  000000000760  206432737166  000000000000  225125456346 -056051336027 -114665256260

14440  254325236351  312321436025 -052731452525 -113145276060 -200761010561  060660606060  002000014472  000000000000

                                CELLS  14450  TO   14467      ALL CONTAIN       000000000000

14470  000000000000  000000000000  050000014403  060100014430  050000014403  060100014431  050000014402  060100014432

14500  050000014401  060100014433  053500114430  050000014400  060100114425  007400414404  002000014510  000000014425

14510  050000014430  040000014507  062100014514  007400414405  000000014426  000000014430  000000014406  007400414407

14520  000000014435  000000014440  000000014444  007400414410  000000014431  000000014432  000000014434  000000014433

14530  007400414411  007400414411  002000400001  002000400004  002000400004  002000400005  076100000000  002100000153
```

FIGURE 15

CORE DUMP OF SCATRAN TEST PROGRAM

```
*       XEQ
C       TEST PROGRAM FOR FORTRAN II
        DIMENSION X(3)
F       TABLE
        NR = 1
        IR1=1
        IR2=496
        IR3=0
        XR4 = 35.367
        X(NR)=3.5
        CALL EXAMP1
        CALL EXAMP2(X(NR),NR,TABLE)
        CALL EXAMP3(17HBRENTON R. GROVES,22HELECTRICAL
       X ENGINEERING,8H 7/15/66
        CALL EXAMP4(IR1,IR2,IR3,XR4)
        CALL DUMP
        END
```

FIGURE 16

TEST PROGRAM FOR FORTRAN II

```
            AC                MG               SI               KEYS          XR1       XR2       XR4
   - 075400100274    -206060606060    -000000300200     000000000000     00001     00000     77554
     -.33897956-20    -.06095237+02                                       -77777    -00000    -00224

TRAP  DCT  IOT  OFL     SENSE LIGHT    1    2    3    4     SENSE SWITCH   1    2    3    4    5    6
 OFF  OFF  OFF  OFF                   OFF  OFF  OFF  OFF                  OFF  OFF  OFF  OFF  OFF  OFF


00000  000000000000  000000000000  000000000000  000000000000  000000000000  000000004470  000000000000  000000003354

00010  002100000275  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000

                                   CELLS  00020  TO  00127 ALL CONTAIN       000000000000

00130  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  000000000000  001101060606

00140 -000000300201  000000000103  000000000000  000665077461  002000400001  002000400004  002000400004  002000400005

00150  076100000000  002100000275  002100000150  002100000144  002100000145  002100000146  002100000147  002100000326

00160  050000000002  060100000264  050000000151  060100000010  060000077462  050000000236  060100000266 -053400100266

00170  050000000236  060100000271  050000000237  060100000270  050000000240  060100000267  050000000255  060100000265

00200  050000000256  060100100275  007400400153  075400100275  040200000203  062100000207  007400400154  007400000274

00210  007400000266  007400000152 -053400100266  007400400155  007400000241  007400000245  007400000252  007400400156

00220  007400000271  007400000270  007400000267  007400000265  007400400157  076200001321  054000000232  054400000000

00230  003000000233  002000000001 -100003000000  076000000005  042000700001  002000000234  000001000000  000760000000

00240  000000000000  225125456346 -056051536027 -114665256260 -377777777777  254325236351  312321436025 -052731452525

00250 -113145276060 -377777777777 -200761010561  060660606060 -377777777777  206432737166  202700000000  233000000000

00260  000000377777  000000000000  000001000000  000000000000  000000000000  206432737166  000001000000  000000000000

00270  000760000000  000001000000  000000000000  000000000000  202700000000  063400400322  053400400000  063400400323
```

FIGURE 17 .

CORE DUMP OF FORTRAN II TEST PROGRAM

statements, we can see the changes required to convert a subroutine:

FORTRAN		SCATRAN	
CALL EXAMP1		CALL SUBROUTINE()=EXAMP1.()-	
ADDRESS	CONTENTS	ADDRESS	CONTENTS
00202	0074 004 00153	14505	0074 004 14404
00153	0021 000 00144	14404	0021 000 14532
00144	0020 004 00001	14532	0020 004 00001

In each case, the CALL statement results in a TSX (0074) tagged by IR4. Control passes through the transfer vector (0021) to the subroutine EXAMP1 (a one line transfer (0020) back to the next statement after the call). Therefore without calling parameters, the two subroutine statements are exactly the same. The next statement requires four locations in succession:

```
F TABLE
  CALL EXAMP2                          CALL SUBROUTINE()=EXAMP2.
  (X(NR),NR,TABLE)                     (X(NR),NR,TABLE.)
```

ADDRESS	CONTENTS	ADDRESS	CONTENTS
00206	0074 004 00154	14513	0074 004 14405
00207	0074 000 00274	14514	0000 000 14426
00210	0074 000 00266	14515	0000 000 14430
00211	0074 000 00152	14516	0000 000 14406

The first difference is that although the calling parameter addresses follow the TSX tagged by IR4, FORTRAN fills the first four characters with 0074's, while SCATRAN uses zeros. As a general rule, this does not matter because that part of the word is never used, but some subroutines use the presence of the 0074 as a flag to tell how many calling parameters there are in a call statement. The conversion example will show how to handle this. Let us look at the contents of each of the addresses of the first of the above words:

```
00154      0021 000 00145        14405      0021 000 14533

00145      0020 004 00004        14533      0020 004 00004
```

Location 00154 has a transfer (0021) to location 00145. Location 00145 has a transfer (0020) back to the main program. The identical listings can be seen in SCATRAN.

The next address contains

```
00274      202700000000          14426      202700000000
           = 3.5 = X(NR)                    = 3.5 = X(NR)
```

This shows that a floating point number declared as a calling parameter is treated in exactly the same way in both programs. Thereofre, floating point numbers need no conversion, such as the call to XSCALV. The next address contains

```
00266      0 00001 0 00000       14430      0 00000 0 00001
           = NR                             = NR
```

This shows the fundamental difference when the calling parameter is an integer number.[8] FORTRAN places the integer in the decrement of the word and SCATRAN places it in the

[8] A computer word can be divided up as follows: the highest order integer is the prefix, the next five are the decrement, the next one is the tag, and the last five are the address.

address. This is fundamental because many FAP fixed point (integer) operations operate either on the decrement or the address of the word. Since the integers are naturally found in the decrement in FORTRAN, decrement operations are used almost exclusively. When it is desired to have two pieces of information in a word, the integer is brought into the decrement and then shifted into the address. Therefore, a conversion program has to take in the words representing any integers and then shift the integer into the decrement before calling the subroutine. Without the shift, the subroutine would think that all incoming integers were zero. The last address contains

```
00152    0021 000 00150        14406    0021 000 14536
00150    0761 000 00000        14536    0761 000 00000
```

In each case the transfer (0021) is the transfer vector for subroutine TABLE. This shows that subroutine entry points can be called by name in a calling parameter in either system. However, one should note that in SCATRAN the name has to be followed by a period to designate it as a subroutine and not as a variable. In FORTRAN, a letter F in column one followed by a list of names separated by commas starting in column seven has to be used. This is a statement separate from the subroutine calls and it can be placed anywhere in the FORTRAN program.

EXAMP4. is merely a repeat example of the above with both fixed and floating point numbers in the calling parameters. The core dump representation in the two cases are as follows:

```
SCATRAN - CALL SUBROUTINE()=EXAMP4.(IR1,IR2,IR3,XR4)-

ADDRESS      CONTENTS           ADDRESS       CONTENTS

14523     0074 004 14410        14410      0021 000 14535
14524     0000 000 14431        14431      0 00000 0 00001
14525     0000 000 14432        14432      0 00000 0 00760
14526     0000 000 14434        14434      0 00000 0 00000
14527     0000 000 14433        14433      206432737166
```

```
FORTRAN - CALL EXAMP4(IR1,IR2,IR3,XR4)

ADDRESS      CONTENTS           ADDRESS       CONTENTS

00217     0074 004 00156        00156      0021 000 00147
00220     0074 000 00271        00271      0 00001 0 00000
00221     0074 000 00270        00270      0 00760 0 00000
00222     0074 000 00267        00267      0 00000 0 00000
00223     0074 000 00265        00265      206432737166
```

Reference to the above text will show the differences and similarities between the two compilations.

In addition to data and subroutine names, a calling parameter can refer to a Hollerith field. A Hollerith field is a set of machine words containing BCD coded alphanumeric text. It is variable in length, containing enough words to hold the designated text at six characters to a word. In FORTRAN, a Hollerith field is specified by a number equal to the number of characters followed by the letter H, followed by the text. In SCATRAN, an F in column one of a statement will identify the statement as a Hollerith field with the statement name containing the address of the first word in the field. This is not to be confused with the F card in FORTRAN although both have an F in column one.

SCATRAN - CALL SUBROUTINE()=EXAMP3.(NAME,DEPT,DATE)-

```
                    ADDRESS      CONTENTS

                    14517      0074 004 14407
                    14520      0000 000 14435
                    14521      0000 000 14440
                    14522      0000 000 14444
```

Looking at the address specified by the calling parameters we find

```
14407     0021 000 14534         14534     0020 004 00004

14435     22 51 25 45 63 46      14436     -05 60 51 33 60 27
          B  R  E  N  T  O                  N     R  .     G

14437    -11 46 65 25 62 60
          R  O  V  E  S

14440     25 43 25 23 63 51      14441     31 23 21 43 60 25
          E  L  E  C  T  R                 I  C  A  L     E

14442    -05 27 31 45 25 25      14443    -11 31 45 27 60 60
          N  G  I  N  E  E                 R  I  N  G

14444    -20 07 61 01 65 61      14445     06 06 60 60 60 60
             7  /  1  5  /                 6  6
```

The above shows how the three Hollerith fields would appear in the core dump. The minus sign on a BCD word means that a four is to be added to the first number to obtain the BCD character.

The FORTRAN statement is

```
CALL EXAMP3(17HBRENTON R. GROVES,22HELECTRICAL ENGINEERING,
            8H 7/15/66)
```

The equivalent core dump is

```
ADDRESS          CONTENTS

00213            0074 004 00155
00214            0074 000 00241
00215            0074 000 00245
00216            0074 000 00252
```

If cells 00155, 00241, 00245 and 00252 and the corresponding ones following them are examined, it will be found that they contain exactly the same BCD information as in the SCATRAN call above. Since it is the address of the Hollerith field that is important to the subroutine, the conversion program should handle these in exactly the same manner as the address of the subroutine name.

The other main difference between SCATRAN and FORTRAN is in the way each language stores data words in arrays. SCATRAN stores data in a forward array or one that has the first word in the lowest storage address while FORTRAN uses a backward array in that the first word goes into the highest address.

Fortunately, the SC 4020 subroutines were originally written to accept either a forward or backward array. Generally, one of the calling parameters will have a minus sign attatched to it for the forward array. When a conversion program is written, the proper "clear and add, store" combination should be used to insure that the minus sign is carried along with the data word.

Sample conversion program

A sample conversion program follows to convert the SCATRAN statement

```
CALL SUBROUTINE()=EXAMP.(X(NR),NR,TABLE.)-
```

so that a subroutine that uses the FORTRAN statement

```
CALL EXAMP2(X(NR),NR,TABLE)
```

can be called. The 2 was dropped in the subroutine name because the conversion program has to have its own name. In the SC 4020 programs, the ending "V" was usually changed to "S". Fig. 18 will give a sample conversion program.

This sample program shows how to handle three types of calling parameters so that a FORTRAN-FAP subroutine can be called from a SCATRAN main program. The value of X(NR) is brought into the subroutine by indirect addressing and stored in cell A1. The value of NR, which is in the address of the data word, is brought in by indirect addressing through the calling parameter and stored in cell A2 after being shifted into the decrement. This is accomplished by having a "ALS 18" (accumulator left shift eighteen bits) between the "clear and add" which places the whole word into the accumulator and the "store" which places the word into the proper cell. When a word is printed out in octal notation, it has twelve digits which represent three bits each of the thirty-six bit word in the machine. Therefore, to

```
***    FAP
       COUNT    20
*    CONVERSION PROGRAM FOR EXAMP2 FROM EXAMP
*    THE CALL TO THIS PROGRAM IS
*    CALL SUBROUTINE()=EXAMP.(X(NR).NR.TABLE.)@
EXAMP  SXA      (X4).4        ENTRY POINT. SAVE IR4 IN (X4)
       CLA*     1.4           BRING IN X(NR)
       STO      A1            STORE X(NR) IN A1
       CLA*     2.4           BRING IN NR
       ALS      18            LEFT SHIFT 18 BITS
       STO      A2            STORE NR IN DECERMENT OF A2
       CLA      3.4           BRING IN ADDRESS OF TABLE
       STA      C3            STORE ADDRESS OF TABLE IN C3
       TSX      $EXAMP2.4     CALL EXAMP2
       TSX      A1.0          (X(NR).
       TSX      A2.0          NR.
C3     TSX      **.0          TABLE)
(X4)   AXT      **.4          RESTORE IR4
       TRA      4.4           RETURN TO MAIN PROGRAM
A1     PZE                    STORAGE FOR X(NR)
A2     PZE                    STORAGE FOR NR
       END
```

FIGURE 18

SAMPLE CONVERSION PROGRAM

shift an integer from the address (the right five numbers) to the decrement requires a shift of eighteen bits.

Since it is the address of the subroutine TABLE we are interested in and not the TABLE itself, indirect addressing is not used. The "clear and add" without the asterisk brings in the third calling parameter word which has the address of the TABLE transfer vector in it. All the other positions are zeros. The address part of this word is placed in cell C3 which simulates a FORTRAN calling parameter for TABLE.

When the four TSX's are assembled and loaded in the computer, they will look like this:

OP CODE	TAG	ADDRESS
0074	004	Address of Transfer Vector EXAMP2
0074	000	Address of A1
0074	000	Address of A2 (integer in dec. of A2)
0074	000	Address of Transfer Vector to TABLE

It can be seen that these four calls are identical to the FORTRAN compilation for the statement

```
CALL EXAMP2(X(NR),NR,TABLE)
```

Since calling parameters are indifferent about whether the called item is a variable or a constant, constants can be used just as easily.

So far, the discussion in this chapter has centered on the use of SCATRAN and FORTRAN II as the programming languages. Should it ever be desired to run these subroutines under a FORTRAN IV operating system that uses FAP as a basic machine language, very few changes would have to be made. A reference is given that lists all of the changes between FAP in FORTRAN II and FORTRAN IV.[9]

---

[9] ______, *IBM 7090 - 7094 IBSYS Operating System Version 13, Macro Assembly Program (MAP)* (New York: IBM Corporation, 1965), p. 56.

APPENDIX B

SC 4020 OPERATING SYSTEM

Summary

This appendix contains the details of the SC 4020 operating system. It is an extension of the SCATRAN operating system in use at the Ohio State University. The system will produce magnetic tape that can be used on any SC 4020 or BL 120 installation and is indifferent to the computing languages used at that location. The appendix can be divided into five sections.

1. A description of the hardware operating commands and their function used by the SC 4020 Microfilm Plotter.
2. A description of the calls and uses of the software written to produce the above commands on tape with the addition of a few calls added to any main program.
3. Tests to demonstrate and check out the subroutines of the operating system.
4. Suggestions to aid the programmer in running programs that produce SC 4020 tapes.
5. Statement listings of the available sub-

routines with comments as to the exact calls to use.

This body of work is complete and there should be no difficulty in producing satisfactory output under the SCATRAN operating system. However, it is useful to have a copy of the SC 4020 Programmer's Manual in support of the subroutine descriptions because where this work would merely repeat the manual, reference was made to the manual rather than giving the complete description.

Performance commands

The performance commands can be represented in a twelve digit octal notation representing a thirty-six bit binary number by a two octal digit code that always appears in the two highest order (left-hand) positions of the command word. These codes are given in Table 4.

The Select Camera commands are used to open the shutter on the selected camera and close the shutter on the other one. Camera 1 corresponds to either the thirty-five millimeter or the sixteen millimeter camera, depending on which is mounted on the SC 4020, and Camera 2 corresponds to the nine by nine hardcopy output.

The use of Advance Film is obvious and will be used most by the programmer when making movies. When this command is used with the earlier SC 4020's or with the BL 120, it should be placed at the end of a record on the magnetic tape to give time for the film to advance. The newer SC 4020's lock out the processing when this command is received and therefore it can be placed anywhere in a record.

The Expand Image and Reduce Image commands will probably not be used in the course of making a movie. The Expand Image is used when it is desired to run one frame into another as when plotting a strip chart. The Reduce Image shrinks the frame area to about one-fourth its normal size. In both cases, normal operation will be restored by calling the opposite command.

TABLE 4

PERFORMANCE COMMANDS

COMMAND	OCTAL OP CODE
Select Camera 1	41
Select Camera 2	42
Select Both Cameras	43
Advance Film	46
Expand Image	44
Reduce Image	45

Vector and Plotting commands

The second set of commands are used to draw lines, axes, and vectors. They are given in the table on the next page.

TABLE 5

VECTOR COMMANDS

COMMAND	OCTAL OP CODE
Draw Vector	6X or 7X
Generate X-Axis	30
Generate Y-Axis	32

The third set is used to plot a single character at any desired point on the face of the screen. They are given in the following table.

TABLE 6

PLOTTING COMMANDS

COMMAND	OCTAL OP CODE
Plot	00
Expose Heavy	02
Expose Light	04

The reason for including these two groups of commands in one area is that they use many of the same bits of the thirty-six bit control word. The bits are assigned as follows:

0 - 5	Operation Code
6 - 7	Not Used
8 - 17	X Deflection
18 - 23	Character Code
24 - 25	Not Used
26 - 35	Y Deflection

Each location of the X-Y grid can be specified by a ten bit word representing 1024 different cells in each direction. In the case of vector and axis commands, the specified location is the start of the line to be drawn, and in the case of the plot commands the specified location is the center of the character to be plotted. The cells are numbered in octal notation from zero to 1777 with the zero-zero position being in the upper left-hand corner of the grid. X locations are read across and Y locations down with 1000, 1000 being in the center of the screen.

The other bits of the control word are utilized quite differently depending on the particular operation code at the beginning of the control word. When the first two bits are ones, the SC 4020 will draw a vector from the location specified by the deflection bits. The length of this vector in the X direction is determined by bits two through seven and the Y length by bits twenty through twenty-five. The direction of each of these components is determined by the sign bits in eighteen and nineteen. A one in bit eighteen indicates a plus X component, a zero indicates a minus X component. Likewise, bit nineteen indicates the sign of the Y component.

Because six bits are alotted for each component, the maximum length that a component can cover is sixty-four cells. As there are 1024 cells for full scale, this means that the maximum length of a vector is about 1/16 of full scale. Since the nine-by-nine inch hardcopy examples use about seven and one-half inches for full scale, the longest vector can be approximately one-half inch long. To draw lines longer than this between two arbitrary points requires that vectors be formed end to end. An example is shown by the arrow in the ID Frame on page 195. The point of the arrow is a maximum length vector and the lines forming the tail are constructed of three vectors end to end. A subroutine has been written to construct automatically vector control words that will join any two arbitrary points merely be specifying the points to be joined.

When the GENERATE X-AXIS or GENERATE Y-AXIS operation codes are used, bits zero to five specify the operation, eight through seventeen the X deflection, and twenty-six through thirty-five the Y deflection. The remaining ten bits -- namely, six, seven, and eighteen through twenty-five -- are used to specify a stop point. The X command causes a line to be drawn parallel to the X axis starting at the point designated by the X and Y deflections extending towards the right to the stop point. The Y command causes a line to be drawn parallel to the Y axis upwards to the stop point. The stop point has to be at least sixty-four locations away from the starting point in each case. If the stop point bits all contain zero, the axis will be swept to the far right or upper limit of the picture area. The only difference between the two commands is that the Y-AXIS stop point is expressed directly as a binary number while the X-AXIS stop point is expressed as the ones complement of the location. This means that to calculate the stop point from the X-AXIS command word, it is necessary to subtract the octal number formed from the stop point bits from 1777 in order to determine the true stop point.

The PLOT, EXPOSE HEAVY, and EXPOSE LIGHT commands are identical except for the operation code. Bits six, seven, twenty-four and twenty-five are not used. Bits eighteen through twenty-three are used for a BCD code that corresponds to the desired character that is to be plotted. This format is very easy to interpret when the thirty-six bit word is printed out as a twelve digit octal base word. An example is given below:

```
          0 0 1 4 0 0 5 2 0 4 0 0

           OP       X     CHAR     Y
         CODE      DEF            DEF
```

In this case, the 52 is the BCD code for the plotting dot. The X deflection of 1400 would be three-quarters of the way across the screen and the Y deflection of 0400 would be one-quarter of the way down. Of course, these are both octal numbers where the 1024 cells correspond to an octal number of 2000. Therefore, the plot for this command word would look like this:

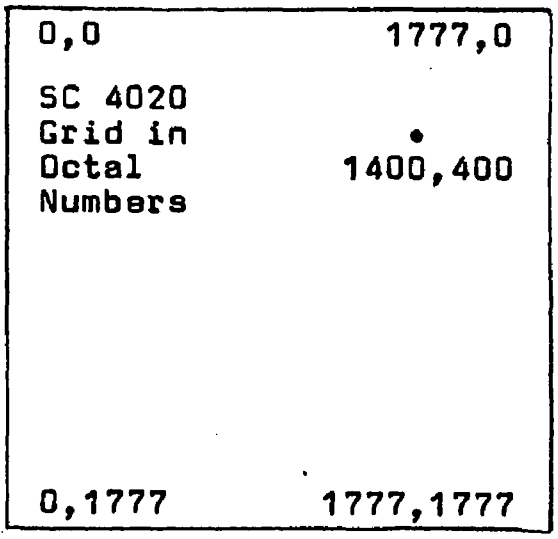


FIGURE 19

COMMAND WORD "001400520400" PLOTTED

The EXPOSE HEAVY and EXPOSE LIGHT commands allow for the plotting of two different densities. When either of these calls has been made, all plotting of characters will be in the same density until the opposite mode is called. Usually EXPOSE HEAVY is called at the start of a frame and all plotting is done in this mode. There is no density mode selection when drawing vectors or generating axes, but the intensity can be controlled by repeating the same call several times to increase the shading of a particular line.

Typing mode commands

The fourth group of command words are concerned with what is referred to as the typing mode. They are as follows:

TABLE 7

TYPING MODE COMMANDS

COMMAND	OCTAL OP CODE
Type Specified Point	20
Type Current Point	22
Carriage Return	52
Stop Type	12
Reset	56

These commands are used to produce microfilm from text and will generally not be needed when making a movie because the letters (actually the characters on the matrix in the CHARACTRON tube) can be neither rotated nor changed in size. However, these commands are used for the ID Frame that opens and closes each job run on the SC 4020.

The typing mode is initiated by either the TYPE SPECIFIED POINT or TYPE CURRENT POINT command. The face of the tube can be considered as a page capable of holding sixty-four lines of printing of one hundred twenty-eight characters each. Any word following the above commands will be translated as BCD text and will be typed in order until the one hundred twenty-eighth position is reached in a line. The logical hardware will then cause the printing to start at X = 0 and sixteen cells below the Y deflection specified for the last line.

The TYPE SPECIFIED POINT command uses the deflection bits to give the location of the first character to be typed.

The TYPE CURRENT POINT command uses only the first six bits of the word. The other bits can be used for BCD characters. In this case, typing starts at whatever location was stored in the X and Y deflection registers for the last operation.

The SC 4020 logical hardware does not have any provision for determining whether or not a complete word will fit at the end of a line. Therefore, typed words will be broken whenever the one hundred twenty-eighth location is reached unless the CARRIAGE RETURN command is used. The BCD character "52" will not be printed but will cause the next character to be printed on the line below the present line at X = 0.

When the sixty-fourth line has been reached, it is necessary to advance the film or the following characters will be overlayed in the same frame that has already been plotted.

The typing mode can be halted by calling either a STOP TYPE command ("12") or a RESET command ("56"). The appearance of either a "12" or a "56" in a BCD word returns the SC 4020 to the plotting mode and all of the rest of the BCD characters in that word are lost. In addition to stopping the typing mode, the RESET command performs two other useful functions. It advances the film and sets the SC 4020 to the plot heavy mode.

One additional command is available to the SC 4020 programmer. This is PROJECT FORM using operation code "50" in the first six bits of the command word. The other bits of this word are not used. This command allows a prepared slide to be projected onto the film along with the data to be displayed.

This completes the number of operations that are available on the SC 4020. I wish to repeat again that these operations are hardware functions (as oppose to software) and are the end result of the computations that are made to produce a movie. For example, to draw a line either a series of touching dots could be plotted or a series of vectors could be drawn end to end. In either case, the one line would consist of many repeated command words on magnetic tape using only two of the eighteen available operation codes. Although the programmer could specify these commands word by word, all of the advantages of computer-generated pictures would be lost. It is the capability of the software in using the eighteen available commands that turn the SC 4020 into a useful tool.

<u>Various types of subroutines</u>

There are two kinds of subroutines, open and closed. An open subroutine is one that is entered at the top in the normal course of events as the computer progresses from command to command. After the subroutine has finished its computations, the program is left at the bottom and takes the next machine instruction. The open subroutine has to be placed where it is to be used and can only be entered once. It must be replaced at each point that it is needed. The main use of an open subroutine is to check out parts of a main program. When each part is running, they can be placed end to end to form the main program. A good example of this is the program used for the first tape of the spheroidal radiator movie.

A closed subroutine is one that is entered by a jump from a location in the main program that is completely independent from the location of the subroutine. The jump can be made to any location in the subroutine. When the transfer is made, the original location is stored in an index register so that it is possible to return to the main program from any location in the subroutine. From this it can be seen that a closed subroutine can be stored anywhere in the computer memory, have as many entrances and exits as desired, and be called whenever it is needed. For these reasons, most subroutines used in making an SC 4020 tape are of the closed type.

SC 4020 subroutines

These subroutines can be divided into seven different categories:

1. Setup routines to make tape
2. Routines for plotting points or characters
3. Routines for typing, making variable size letters, and tables for symbols
4. Routines for drawing lines, axes, and vectors
5. Miscellaneous routines that do not fit in above categories
6. Routines specifically developed for the spheroidal radiator movie example
7. Tests to check out the above routines

The statement listings for these subroutines are given, along with enough information to use the subroutine calls, in Appendix B. This appendix will briefly describe the operation of the various subroutines, the calls that can be made, and any idiosyncrasies that would be of interest to a programmer attempting to use them. The routines that are covered in categories two through five follow very closely the original routines developed by North American Aviation. Except for the SCATRAN conversion subroutines, much additional information can be found in the SC 4020 Programmers' Reference Manual. In addition to the subroutines listed below, there are a number of other programs available in FORTRAN for use on the SC 4020. These programs are covered in the Reference Manual but are not described herein for two reasons.

The first reason is that they form a set of subroutines that are higher in the software hierarchy than the basic subroutines described below. Because of this, they generally use the basic subroutines to perform extremely complex but limited functions for the programmer. GRIDIV is an example of this in that it draws linear and nonlinear graphs and then plots data points and labels them. This subroutine could be considered as the main program to most engineering production on the SC 4020 but it would have almost no use in the making of a movie.

The second reason is that these programs are generally written in FORTRAN and not in FAP. This means that a conversion program cannot be written to utilize them directly under the SCATRAN operating system at O.S.U. In order to do so, they would have to be rewritten line by line to conform to the SCATRAN language rules. Since SCATRAN is a more flexible language than FORTRAN, this could be done without too much effort except for the time involved, but in view of the limited usefulness of these subroutines, this has not been done. Since a successful FORTRAN operating system extension was developed at O.S.U. prior to the conversion to SCATRAN, if the programmer desired to use these additional capabilities, the program could always be run under the FORTRAN system as outlined in Appendix C.

Setup routines to make tape

This category of subroutines[1] was written to produce an SC 4020 tape under the SCATRAN operating system. It has two features that were especially designed with the programmer in mind. As much of the setup routine required as possible was shifted from calls in the main program to calls in the subroutines. This means that the tape non-system file can be opened and the various input parameters read in with only one call. In the same fashion, the file can be closed and the run ended with only one call.

By dividing the job to be done up into a number of different routines instead of one larger package, the programmer can be given the flexibility of having a number of different options with the same call as above. Should any new options be desired, the programmer has to rewrite and debug only one small section instead of the whole package.

[1]Listings on pp. 243-247.

The following subroutines must be used for every SC 4020 tape run.

CAMRAV	XSCALV
(PLOT)	ID FRAME
JOBNO	END FRAME
TAPE	

This set is self-contained in the sense that for every transfer vector required, there is a matching entry point either in the subroutines themselves or in the SCATRAN system library. The main calls are:

```
CALL SUBROUTINE()=OUT.(N)-
CALL SUBROUTINE()=CAMRAV.(N)-
CALL SUBROUTINE()=FRAMEV.(N)-
CALL SUBROUTINE()=CLOSTP.()-
```

The main program can start with a call to OUT.. The program enters the TAPE. subroutine and sets the variable OUTPUT to the value of N in the calling parameter. N can be any integer from one to four and will have the following effect on the output of the program to be produced.

N = 1 The program is executed and an SC 4020 tape is made but no output is returned.

N = 2 The SC 4020 tape is made but not returned with the program. The output consists of the command words printed in 170 word blocks in the form of a core dump.

N = 3 The SC 4020 tape is made and returned with the program. No printed output results.

N = 4 Both the tape and the printed output are returned.

Each one of these forms has a definite use. "1" is used when it is desired to check the timing of a program to determine time estimates and the output is known or does not matter. "2" is used for checking out programs so that the programmers can analyze the output of a tape without the bother of obtaining a tape dump. It also has the great advantage that the program does not have to be submitted at the Numerical Computation Laboratory but can be run at a remote terminal. A word of caution is necessary when obtaining printed output. The programs to be checked should be very short because in spite of the compact format used to display the output (450 words to a page), it is possible to generate over a hundred pages of output in the course of a one minute program. In the case of a long program, it is better to generate only the tape and then use a special dump routine that was written to check small sections in the middle of a large block of data.

"3" and "4" can only be used at the Numerical Computation Laboratory because they result in the physical tape being returned with the program. "3" is the normal operating output parameter. "4" can be used to check the printed output against a test tape that is actually going to be run on the SC 4020.

The next call in the main program should be to CAMRAV.. This call should be made before any attempt to write information on the SC 4020 tape because it opens the non-system file. The following takes place. The call to CAMRAV. immediately calls IDFM., part of the TAPE subroutine. The ZCLOCK., a library subroutine, is called to start the clock running. This is done so that a more precise interval will be available to determine the time estimates for longer programs than can be obtained from the bookkeeping routines. Several commands open the non-system file and reserve the necessary space for the buffers. Next, the job number is read off of a data card and stored in JOBNO.. Eight scaling factors are read from a second data card and XSCALV. followed by YSCALV. is called.in order to read in the data.

XSCALV. is a very important subroutine that allows the programmer to use his own scale for positioning on the SC 4020 screen rather than the cell location numbers the machine uses. In order to scale the programmer's floating point data, it is necessary to know the boundaries of the working field and the width of the margin on the four sides of the field. This information is obtained from a data card that contains the X and Y boundaries and the margin widths.

The next statement in the TAPE subroutine reads in ten six-character alphanumeric words from a third data card. This card contains the title of the program, the programmer's name and department, and the date. This information is stored in the subroutine IDINFO. that is going to write the identification frame as the first frame of the program. The transfer back to the TAPE subroutine ends on a NORMAL EXIT statement which brings the program back to one statement after CALL IDFM in CAMRAV.. CAMRAV. then determines the correct command word to be written at the start of the SC 4020 tape to produce either 9 X 9 hardcopy, 35 mm film, or both. This is accomplished by the calling parameter N in the CAMRAV. call statement.

CALL SUBROUTINE()=CAMRAV.(N)-

N = 9 The operation code for camera two (9 X 9 hardcopy, OP CODE 42000000000)

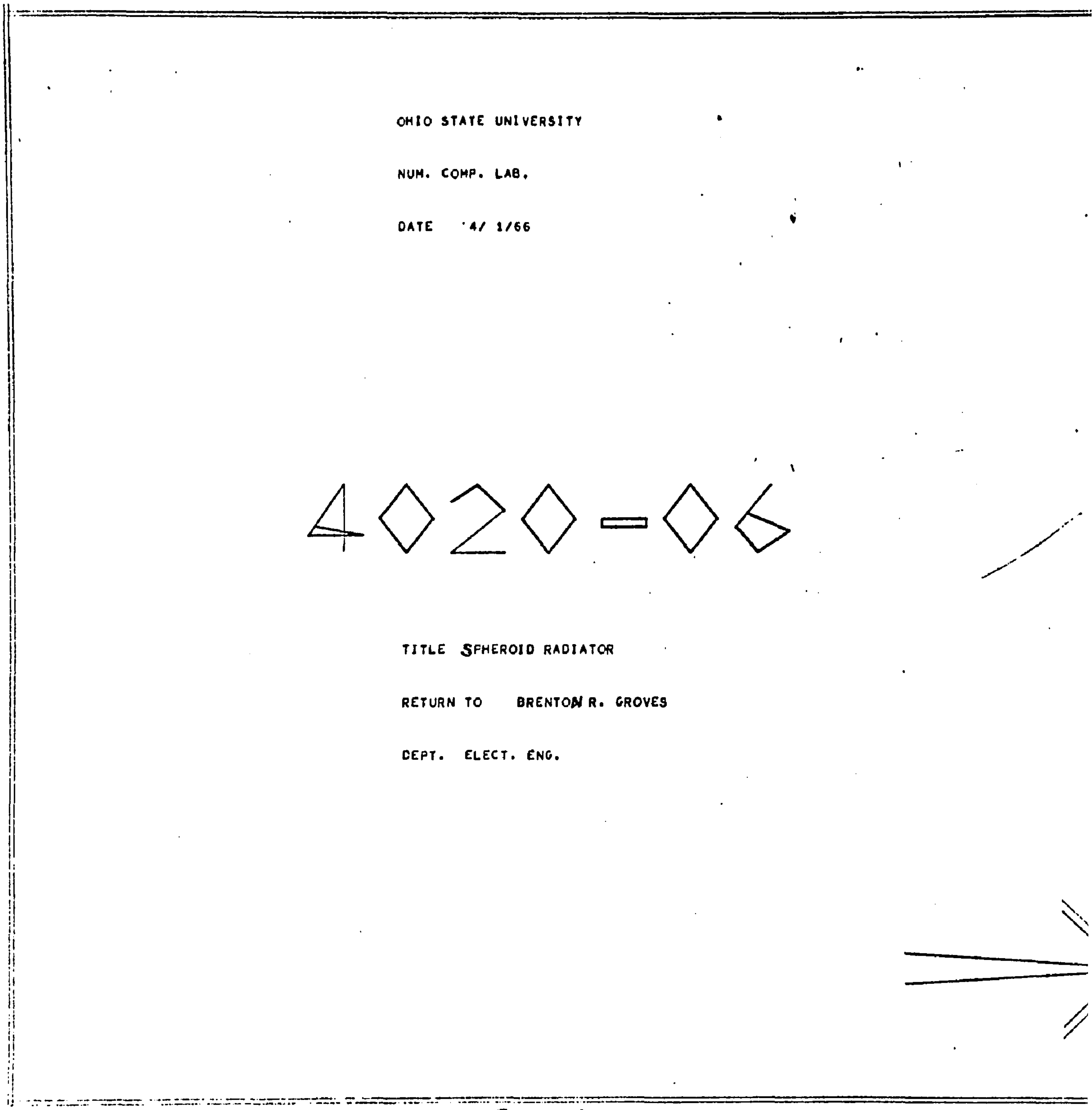
N = 35 The operation code for camera one (35 mm film, OP CODE 41000000000)

N = Any other number. The operation code for both cameras (OP CODE 43000000000)

The actual writing of the command word on the tape is done by the CALL (PLOT) statement. This calls the (PLOT) subroutine which in turn calls STORE, a part of TAPE. STORE could be called directly to write on tape except that almost all of the original subroutines called (PLOT) as a subroutine name for their output. The TAPE subroutine is written in SCATRAN which does not allow the use of brackets as part of the name. Therefore, rather than change (PLOT) in the many places where it appeared, the small subroutine (PLOT) was written to make the call (PLOT) equivalent to STORE. The STORE call has two calling parameters associated with it, the word to be written on tape and an integer that is either 0 or 1. When STORE is called through (PLOT), this integer is always zero. The zero tells STORE to place the incoming word into a 170 word buffer. The buffer is then checked to see if it is full. If not, control passes back from STORE to (PLOT) to CAMRAV..

The next statement in CAMRAV is CALL IDFIN in the ID FRAME subroutine. This subroutine calculates the commands, one word at a time, to create the identification frame (Fig. 20) with the programmers name, date, etc., and the job number in big letters. These are stored in the buffer in STORE through (PLOT) but because the total number of words for the identification frame does not reach 170, no tape has been written yet. After calculating approximately sixty words and placing them in the output buffer, IDFIN. returns control to CAMRAV. which then returns control to the next statement in the main program.

So far, the programmer has made only two statements in his program, namely calls to OUT. and CAMRAV.. The output form is set up, bookkeeping data has been read into the pro-



OHIO STATE UNIVERSITY

NUM. COMP. LAB.

DATE 4/ 1/66

4020-06

TITLE SPHEROID RADIATOR

RETURN TO BRENTON R. GROVES

DEPT. ELECT. ENG.

FIGURE 20

FILM IDENTIFICATION FRAME

gram, a non-system file has been opened, and the command words for the ID frame are in the output buffer.

Incidentally, the ID frame serves another purpose in addition to identifying the job. It combines most of the SC 4020 machine codes in one frame so that the performance of the SC 4020 can be checked merely by looking at the first frame. If it is correct, then it is likely that any difficulties encountered are on the tape and not in the SC 4020 itself.

When the main program is constructed for the generation of a movie, the general procedure is to calculate one frame in a DO loop, using the DO loop limits to change the data at each pass. The patterns generated in the spheroidal radiator example are a good example of this. This DO loop should start with the statement:

```
CALL SUBROUTINE()=FRAMEV.(N)-
```

The calling parameter N permits the following actions:

N = 0 or missing - the job number and frame count will be displayed in the upper right hand corner of the frame.

N = Anything except zero - the above printing will be suppressed.

N = Negative - when producing hardcopy, the working area is seven by seven inches while the paper is nine inches wide. Four corner marks normally show the smaller working area for each frame.

When N is negative, these marks are suppressed.

The first thing the call to FRAMEV. does is to place operation code for advancing the film in the output buffer. A call to (CLEAN results in the same call to STORE as (PLOT) except that the second STORE calling parameter is now a one instead of a zero. This causes control to first check to see whether there are more than three words in the buffer and then calls in the SCATRAN library subroutine WRNSFR. (Write non-system file under count control.)

The usual statement for writing tape is the SCATRAN statement WRITE BINARY. The SCATRAN manual in giving the instructions to use this statement neglects to mention that each written record on the tape is preceded by a control word written by the system that contains information as to the number of words in the particular record. This word is used to tell the WRITE BINARY subroutine when the buffer is empty, but unfortunately this word also corresponds to an SC 4020 operation code. The result is that an arbitrary character would be plotted at an arbitrary location on the SC 4020 plot at the start of each record. Since each frame can consist of several hundred records, WRITE BINARY is impossible to use. WRNSFR. is exactly what is needed and is in the SCATRAN library but none of the available manuals or memos covering SCATRAN give instructions for its use or even a hint of its existence. It was discovered by discussing the problem of the control word with one of the senior programmers, Mr. Edward Lassiter, at the computer center who knew of its existence.

Since WRNSFR. does not use a leading control word to determine the number of words in the record, the count has to be supplied to it as a calling parameter. This is easily done by counting each word as it is placed in the output buffer. This is the reason for the "under count control" in the name.

WRNSFR. will not accept a record of fewer than three words because of tape drive limitations. If the program attempts to write a one-word or two-word record, the job will be forced off the computer with a message that the program attempted to write a short record. Since it may be desired to attempt to write with only one word in the buffer, at the end of a frame for instance, the STORE program checks this at the start of each write command. If there are fewer than three words in the buffer, the blanks are filled in with a .0.120000000000 which is equivalent to a "no operation" for the SC 4020. In the course of placing words into the output buffer, each time 170 is reached WRNSFR. is called to write a new record.

The calling parameter that follows the call to OUT. at the start of the program is checked to see whether output should be printed as well as being written on tape. If not, control exits back to FRAMEV..

The rest of FRAMEV. keeps track of the frame count and determines if the printing and corner marks should be written on the new frame.

The reason for placing the call to FRAMEV. at the beginning of the DO loop and not the end is because it is this command that writes the operation code for film advance and then closes the record. If it were placed at the end of the DO loop, the ID frame would be printed over the first computed frame instead of being separate.

After all the frames have been calculated the call

CALL SUBROUTINE()=CLOSTP.()-

is made as the last statement of the main program. A call to the subroutine CLOSTP. performs the actions necessary to end the run. The LASTFM. call is the same as (CLEAN and it empties any remaining data in the buffer onto the tape before placing an advance frame code on the tape and ending the record. The next few statements in CLOSTP. are used to determine whether or not the SC 4020 tape is to be returned with the job. The calling parameter from OUT. determines if CLOSE REWIND or CLOSE UNLOAD is called. The first merely rewinds the tape while the second prints out a message to the computer operator to save the tape that was just being used and return it with the program. The rewind operation can be given in a program sent in from a remote terminal, but when a tape is to be returned, the program has to be filed at the Numerical Computation Laboratory.

The last operation to be performed before calling ENDJOB. is to read the clock that was zeroed at the start of the program. The time used by the program is printed out as the last piece of output and is expressed in both minutes and seconds to the nearest one-hundredth of a minute. Subroutine ENDJOB. then removes the program from the computer.

One of the major advantages of using a compiler language such as SCATRAN for the TAPE subroutine lies in the handling of data in the form of input and output. Input-output speeds are extremely slow compared with the rate of operation within the computer, and they constitute a bottleneck to the amount of work that can be processed in any unit time. To get around this, either the operating system or the programmer can set up a series of buffers and the internal commands to move data in blocks in and out of the computer. If the programmer desires to do this using FAP, a very lengthly, complicated program will result. This trouble can be avoided by the simple commands OPENFILE and READ or WRITE when the language in the software hierarchy above FAP is used. The compiler reserves space for the buffers, sees that the computed information gets to the right place, and then writes tape while the rest of the computer is still calculating on the remainder of the data. As long as the time used to calculate each piece of information exceeds the time to write that piece of information on tape, which is the usual case in SC 4020 programs, but not so in most programs that require tape storage, the program will be as efficient as possible. However, if the tape writing time is longer than the computation time, then the program has a fixed time to run that cannot be decreased. Since input-output difficulties were encountered due to hardware differences in the two operating systems, these subroutines were rewritten using the compiler language rather than FAP.

<u>Options to tape writing subroutines</u>

A number of other subroutines have been written that can replace certain sections in this area in order to give the programmer increased flexibility.

TAPE II - This subroutine replaces the TAPE subroutine and is called in exactly the same manner. The first difference is that the open file commands have been rewritten so that instead of writing 170 word records on tape, the number in each record is increased to 1024. This allows much more information to be placed on a single tape, but requires that the microfilm plotter have a 1024 word input buffer.

The other change has been to rewrite the STORE call to allow information already in the output buffer to be changed and to obtain additional calls that are extremely useful when producing a movie. The call is

```
CALL SUBROUTINE()=STORE.(X,N)-
```

X is a dummy calling parameter that is used only when N is zero. N has the following meanings:

N = 0	Add word X to the buffer
N = 1	Write record - reset buffer to zero
N = 2	Write record - drop last word in buffer
N = 3	Write record - keep all words in buffer
N = 4	Add .0.460000000000 (advance film) to words already in buffer
N = 5	Print out the number of words and records put on tape up to this time

N = 0 and N = 1 are the same as TAPE. The additional calls can save a large amount of computer time when writing frames of less than 1024 words such as titles. All of the words in a frame are computed once. Then, instead of calling FRAMEV, a call to STORE.(X,4) will add the command to advance the film. Repeated calls to STORE.(X,3) will write as many frames as desired without having to recompute the words in the buffer each time. STORE.(X,2) will remove the advance film command so that additional information can be placed in the buffer between two frames without losing the information already available in the buffer. The call to OUT is not used with this subroutine because the only output is magnetic tape.

TAPE III - This subroutine was written to permit a quick check by means of printed output. The OUT call, the non-system file statements, and the computation of the ID frame have all been removed. This program should not be used with subroutines ID FRAME or END FRAME because calls to these programs have been replaced by "no ops" in the TAPE III program.

CLEAN UP - When a program is run that represents the center section of a movie, the ID frame is not needed and the job number is not required on each frame. This subroutine was written not only to replace the subroutines ID FRAME, CAMRAV, FRAMEV, and JOBNO but it also frees a large amount of room in the computer memory for the main program. The four subroutines above take five hundred octal words to space. The CLEANUP takes twenty-three.

END FRAME II - When a movie is computed in sections, the end identification frame is not wanted except on the last segment. This subroutine allows the CLOSTP call to be made without writing this frame on the tape.

Subroutines for plotting points and characters

There are three subroutines that can be used for plotting points and characters. One of the most useful calls in this subroutine library is given on the following page.

CALL SUBROUTINE()=APLOTV.(-N,X(M),Y(M),IX,IY,NC,CHAR,ERROR)-

Since APLOTV. operates in the manner described in the SC 4020 programmers' manual it will not be described here. An example of the use of the call APLOTV. is given in both the subroutine test description and in the spheroidal radiator example.

The other two subroutines are PLOTS. and POINTS.. The PLOTS. subroutine is used for plotting a single character using the CHARACTRON matrix characters. The call for this subroutine is

CALL SUBROUTINE()=PLOTS.(X,Y,N)-

X and Y represent the locations where the character is to be plotted. N represents the BCD code for the particular character. This call is useful primarily when it is desired to call a character present in the SC 4020 BCD set that is represented by a number not in the IBM BCD code.

The call to POINTS. is identical to that of PLOTS. except for the use of the calling parameter N. In this call, if N is minus, a plotting dot will be called in the center of the character. Also, the values of N do not correspond to the standard BCD code, but to a code that places the most useful plotting characters equivalent to the smaller integers. The table for this code can be found in the programmers' manual.[2]

~~2~~ ______, _Programmers' Reference Manual, SC 4020 Computer Recorder_ (San Diego: Stromberg-Carlson Corp., 1965) p. 7-3.

Ibid., 7-9.

<u>Printing subroutines</u>

These subroutines consist of a set that was originally used to produce microfilm from text, but it has proven to be extremely useful in creating desired animation effects with very little effort on the part of the programmer. In this group are the following subroutines:

PRINTS	or	RITEZS
PRINTV		RITEZV
		CHSIZV
		HOLLV
		TABL*V (*=integer)

The typewritten mode of operations was explained in the section on the SC 4020. In order to utilize this mode, one of two calls, PRINTS or PRINT6, can be used. The format is

```
                CALL SUBROUTINE()=PRINT6.(-N,WORD,X,Y,)-
    F WORD      PLAIN ENGLISH TEXT-
```

N is the number of letters in the text and X and Y are the floating point values specifying the location of where the typing is to start. PRINT6 will automatically supply the carriage return at the end of a line but will not advance the frame at the end of a page. A higher order subroutine, SCOUTV, is available to simulate the action of an output printer but it is not available in SCATRAN.

Notice that although the SCATRAN FORMAT statement is used to insert the text into the subroutine, the usual brackets and H or Q field indicators are <u>not</u> required. If they are used, the subroutines will treat themselves as part of the text and will drop off the same number of characters at the end of the text.

The call to PRINTS is exactly the same as PRINT6 except that integers equivalent to the raster count on the SC 4020 grid must be used in place of X and Y for the start printing location.

The other block of subroutines is used for variable size letters and to produce animated symbols with only a small amount of programming effort.

The use of these subroutines to produce animated symbols will be explained in the section on special subroutines developed for the spheroidal radiator example. The calls that are used to produce variable sized letters for titles are as follows:

```
CALL SUBROUTINE()=CHSIZS.(N,M)-
```

N and M are integers that control the width and heighth respectively of each character.

```
CALL SUBROUTINE()=RITSTS.(N,M,TABL*V.)-
```

N and M are integers that control the spacing between characters and the spacing between lines. The third calling parameter is the name of the table being used for the characters being used. The usual case is to specify TABL1V. as this contains the BCD set common to both SCATRAN and FORTRAN.

```
CALL SUBROUTINE()=RITE2S.(X,Y,I,J,K,L,-M,WORD,ER)-
```

All of these calling parameters have the same meaning as for RITE2V in the SC 4020 programming manual[3] except for X and Y. When RITE2V is used, X and Y are raster counts, but in the RITE2S. SCATRAN call, they are floating point locations on the programmer grid. The SC 4020 programmers' manual gives a set of instructions for the values of M and N that should be understood and followed when making the above calls.[4] Both the sections on subroutine tests and the spheroidal radiation example give examples of the use of these calls to produce titles in a movie.

Line and axis subroutines

Since the SC 4020 hardware commands can only determine that a vector is to be drawn up to sixty-four raster counts in the X and Y increments, it is necessary to have a subroutine that will draw a line between any two specified points by using these vector commands. This is accomplished by using the subroutine LINE. with the call

```
CALL SUBROUTINE()=LINE.(X1,Y1,X2,Y2)-
```

The calling parameters are the floating point X's and Y's for the two end points of the line to be drawn. An example of the use of LINE. to draw a desired figure is given in the spheroidal radiator example when the ellipse is

---

[3] *Ibid.*, 9-7. [4] *Ibid.*, 9-10.

drawn. If an accurate curved line is desired, dots can be plotted side by side along the path of the line, but for general work any curved line can be drawn by means of short vectors. Most of the computer-generated pictures to date have used the technique of calculating points and then using subroutine LINE. to fill in the spaces.

Two subroutines are available to draw the axes in the X and Y directions. They are VXAX I and VXAX II. They are mainly used for drawing grids for engineering graphs and their operation is self-evident.

Programmer-generated TABLES and special subroutines developed for the spheroidal radiator example

Some of the most useful subroutines for generating computer animation are the ones used for generating variable size letters. The SC 4020 programmer's manual gives instructions for producing titles using the subroutine RITE2V and VCHARV. A slightly different method of writing a TABL*V will be given here.[5]

The programmer constructs any desired symbol on an eight-by-eight lattice using straight lines to connect any two points. There is no limit to the number of lines that can be employed. The only restriction is that any one line

[5]The name of the table can be any six alphanumerics. The format TABL1V, TABL2V, etc., is used merely for continuity with the existing programs.

cannot have a required distance of more than sixty-three raster counts between its ends in either the X or Y direction. The longest line that can be drawn is one that goes from one corner of the lattice to the other where the lattice has sixty-three counts between each point. On the 1024 raster, this would span 441 counts in each direction or approximately forty percent of the distance across the working field.

The GENERATOR SYMBOL (GENSYM) subroutine will be used as an example. It was desired to place a symbol of a generator in the middle of an ellipse. Leads were drawn to the ellipse on both sides and an arrow showed the direction of current flow. The sine symbol was placed in the center of the generator and was tied to the arrow in that it flipped upside down when the arrow changed direction. A drawing was made of the symbol and was broken down into its various parts. This is shown in Figure 21. The reason that the symbol was not generated all at once was that the different parts required different sizes (or counts) between the points of the lattice. A table was then constructed by listing the points to be connected, three lines to a word. The table starts at the bottom and is written upwards. The operation code OCT is used for octal numbers. The table is given on the page following Figure 21.

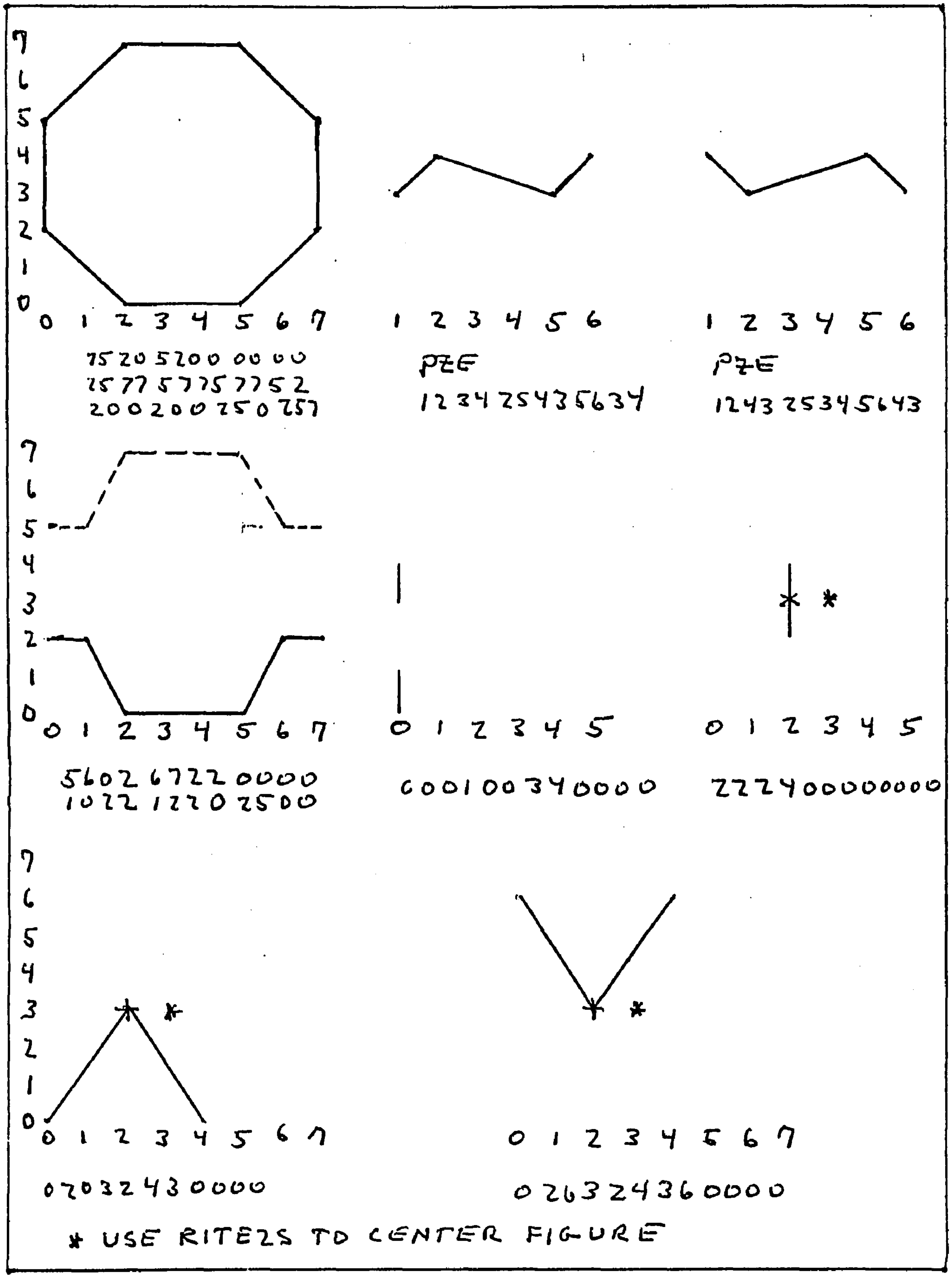


FIGURE 21- GENSYM WORK SHEET #1

```
***        FAP
           COUNT 20
           VFD 9/9, 9/10, 9/11, 9/12
ENTR4V     VFD 9/0, 9/3, 9/5, 9/7
           OCT 026324360000
           OCT 020324300000
           OCT 222400000000
           OCT 000100340000
           OCT 560267220000
           OCT 012212202500
           PZE
           OCT 123425435634
           PZE
           OCT 124325345643
           OCT 752052000000
           OCT 257757757752
TABL4V     OCT 200200250257
           PZE ENTR4V
           END
```

Each of the octal numbers represents the X beginning lattice point followed by the X ending followed by the same for Y expressed as a four digit number. Therefore, each twelve digit number represents three lines. The first three OCT numbers are the eight lines for the octagon figure, the next two lines are for one of the sine symbols, and so on. A symbol ends with four zeros in a row so that when an even multiple of three lines is expressed, they have to be followed by a PZE (all zeros) in order to shut the symbol generating subroutine off.

The above TABL4V could be used with calls to RITE2S., but because all the pieces formed a single symbol with a preset location, it was decided to continue the subroutine GENSYM which would have the table in it and would make all of the required calls with a single call.

Since the calls were going to be made to CHSIZV and VCHARV rather than to RITE2S, the coordinate positions had to be expressed in raster counts in the decrement of the calling parameter rather than in a floating point number that expressed the location in the programmer's coordinate system. RITE2S is usually called to perform this conversion through XSCALV, but in this case the design has a fixed location and the calls to the basic subroutines were easier. Also, RITE2S and RITE2V are not needed which leaves more room for the main program.

A large figure of the design was drawn (Figure 22) to determine the number of counts required between the points of the eight by eight lattice and the location to center each part of the design properly. It should be kept in mind that a call to VCHARV requires that the location of the lower left-hand corner of the eight by eight lattice be given whereas when calling RITE2S or RITE2V, the center of the design should be given.[6]

Details of the calls to draw the octagon will be given. Referring to the subroutine GENSYM one finds the following calls about two-thirds of the way down the page. A section of this page is written out on the next page after Figure 22.

[6]*Ibid.*, 9-22.

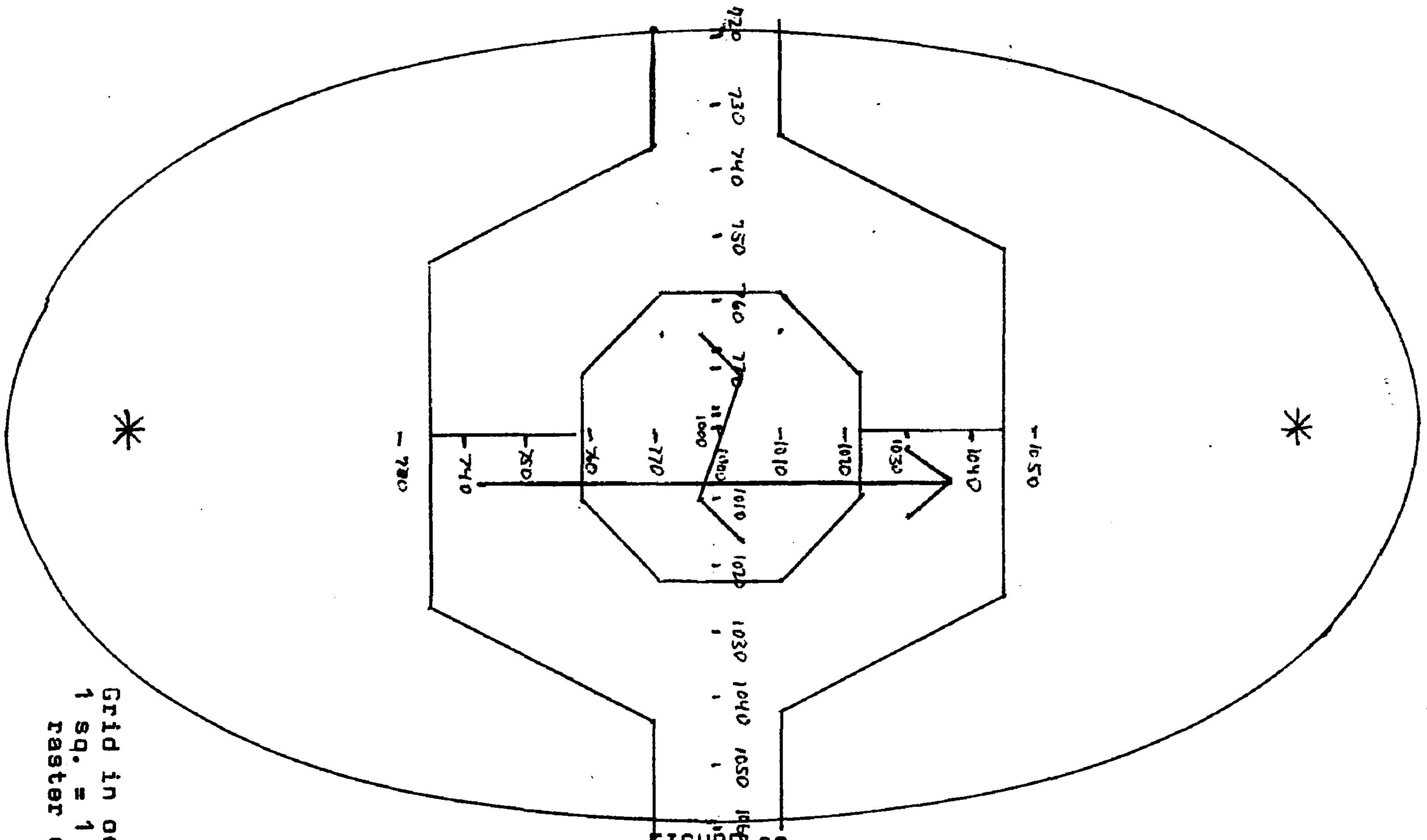


FIGURE 22

GENSYM WORK SHEET #2

Grid in octal
1 sq. = 1
raster count

```
TSX        $CHSIZV,4
TSX        =Ø5000000,0
TSX        =Ø5000000,0
TSX        $VCHARV,4
TSX        =Ø132000000,0
TSX        =Ø1000000,0
TSX        =Ø757000000,0
TSX        =Ø757000000,0
TSX        =0,0
TSX        TABLE,0
```

This is the FAP language notation for

CALL SUBROUTINE()=CHSIZV.(5,5)-

CALL SUBROUTINE()=VCHARV.(90,1,495,495,0,TABLE.)-

in SCATRAN. The calling parameters for CHSIZV indicate that the eight by eight lattice will have five cells between each point. The call to VCHARV will have the following specifications:

132	=	figure will be upright
1	=	figure will be plotted once
757,757	=	lower left-hand corner of figure will be at raster counts 757 in X, 757 in Y
0	=	figure should be constructed from first item in the table
TABLE	=	the table used is part of this subroutine

All of these numbers are in octal notation due to the basic subroutine called. Each integer calling parameter should be followed by six zeros so that the subroutine is assembled with the integer in the decrement and not in the address. This is a reflection of the fact that the subroutines were originally written for the FORTRAN compiled programs which have integers in the decrement.

The SC 4020 coordinate system has zero,zero at the lower left-hand corner of the grid and uses the octal notation in the calling parameters. Therefore, the drawing of designs using these subroutines is facilitated by using this coordinate system and eight by eight per inch graph paper rather than ten by ten.

Although most of the GENSYM subroutine is made up of the above described table and the calls to VCHARV and CHSIZV, the rest is used to invert the sine symbol in the center of the octagon and to draw an arrow whose size and direction is related to current magnitude. The actual call to this subroutine from the main program is

```
CALL SUBROUTINE()=GENSYM.(M,N)-
M = 1     Current arrow points up
          First sine symbol used
M = 2     Current arrow points down
          Second sine symbol used
N = The length of arrow in integer raster
          counts.
```

Subroutine RULE

Subroutine RULE is almost identical in its construction with GENSYM. It was used to illustrate the concept of resonance in the spheroidal radiator example. Two rulers placed on the radiating pattern indicated that the inter-focal distance of the ellipse was the same as the distance of one-half cycle of the radiating field. Figure 23 shows the computer generated output of the RULE subroutine.

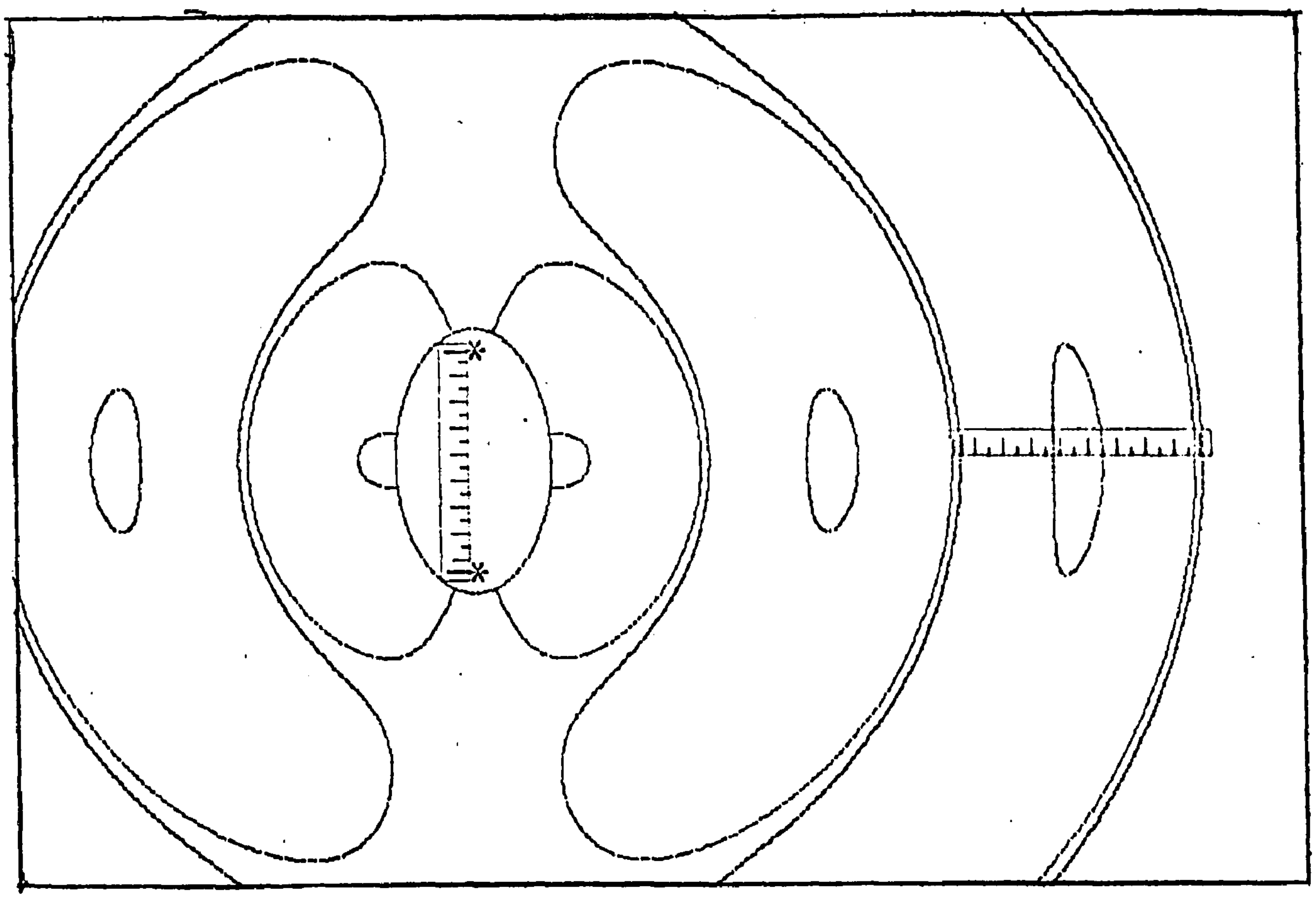


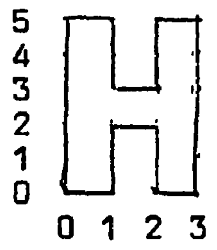
FIGURE 23

RULE SUBROUTINE OUTPUT

A table was constructed for the edge of the ruler, the heavy end marks, two sets of light marks, and the asterisks used for the focal points. A large drawing was made with all these pieces fitting together on the SC 4020 grid so that the location calls could be determined. As in GENSYM, this table was made part of a larger subroutine so that one call to RULE. would serve to draw the entire symbol.

Subroutine TABL5V

TABL5V was written to show the words THE END in block letters for use in closing a movie. A three by five section of the eight by eight matrix was used to form the letters. For example



equals

```
0000 0000 0000
2122 1120 0100
3350 3200 2202
1233 2235 2355
0005 0155 1153
```

The other letters are formed likewise and a free standing table is written. The entry to this table is the subroutine name TABL5V.

The use of this table gives an example of how any table constructed by the programmer can be used. The first call

```
        CALL SUBROUTINE()=CHIZS.(IXS,IYS)-
```

gives the number of raster counts between the eight by eight lattice of each letter. The call

```
        CALL SUBROUTINE()=RITSTS.(IX,IY,TABL5V.)-
```

gives the spacing between each letter, between the rows, and identifies which table is to be used.

```
F WORD    0125234-

          CALL SUBROUTINE()=RITE2S.

                              (X,Y,750,90,1,7,-1,WORD,IR)-
```

is the call that actually calculates the required vectors and writes them on the output tape. The use of these calling parameters can be determined from the programmer's manual except for the use of the variable WORD.[7] It will be noticed that WORD is the address of a SCATRAN format statement consisting of a series of integers. The format statement compiles these numbers into BCD words as follows:

```
                    000102050203
                    046060606060
```

Since T equals item zero in the table, H equals item one, blank equals item five and so on, it can be seen that each BCD character starting at the address WORD is interpreted as an equivalent item in the specified TAB*L. This is fine as long as the table has ten or less items that are labled with the number zero through nine in the format statement. But if you called for item ten at "10", this would be interpreted as an item one followed by an item zero. This difficulty can be avoided by the use of an octal literal instead of the format statement. An example is given on the following page.

---

[7] *Ibid.*, 9-7.

```
WORD = .0.00102050203-
WORD(1) = .0.04606060606O-
```

is equivalent to the format statement provided WORD has been dimensioned properly.

If only six items are to be called at a time, any place in the table can be reached, including the illegal BCD characters, by replacing the variable WORD in the call to RITE2S with an octal literal representing the desired six items.

It is now apparent why TABL1V can be used with straight text. The construction for A is at the seventeenth place in the table, B is at the eighteenth, and so on through the alphabet. Text in the format statement is converted to BCD words which automatically draw the correct letters if TABL1V is used.

Table TABL2V and TABL3V cover the lower and upper case Greek letters. Figure 24 gives the characters generated by these three tables along with the decimal location for each character. All of the characters can be generated by use of the octal literal as described above.

4-QUAD(PLOT)

This subroutine was designed to replace the regular subroutine (PLOT) to perform a special function when making the spheroidal radiator movie. It cut the IBM 7094 computer time required to a quarter when the antenna patterns were

TABL1V

	0	1	2	3	4	5	6	7	8	9
00	0	1	2	3	4	5	6	7	8	9
10	∂	=	"	'	δ	α	+	A	B	C
20	D	E	F	G	H	I	π	.	)	β
30	±	?	–	J	K	L	M	N	O	P
40	Q	R	•	$	*	γ	~	d		/
50	S	T	U	V	W	X	Y	Z	°	,
60	(	∫	Σ	□	&	%	¢	[	]	{
70	}	>	<	≥	≤	∞				

TABL2V

	0	1	2	3	4	5	6	7	8	9
00	α	β	γ	δ	ε	ζ	η	θ	ι	κ
10	λ	μ	ν	ξ	ο	π	ρ	σ	τ	υ
20	φ	χ	ψ	ω						

TABL3V

	0	1	2	3	4	5	6	7	8	9
00	A	B	Γ	Δ	E	Z	H	Θ	I	K
10	Λ	M	N	Ξ	O	Π	P	Σ	T	Υ
20	Φ	X	Ψ	Ω						

FIGURE 24

CHARACTERS GENERATED BY

TABL1V, TABL2V, AND TABL3V

calculated. These patterns were constructed mainly of dots plotted one after the other and were identical in all four quadrants. The first step was to calculate the X and Y points required in one quadrant. Then, instead of calculating the other X's and Y's required for the other three quadrants, the SC 4020 command word was manipulated to give the three other commands to plot in the other three quadrants. Mixed in with the plot commands were other operation codes, such as frame advance. Repeating these would have been very undesirable so a test was made on the six highest order bits of each word. Since the plot command has six zeros for the high order bits, these words were repeated three more times through the proper logic. All other words were sent to STORE, unaltered, one time only.

4-QUAD(PLOT) Version II is an extension of the above which not only plots the four quadrants but also decides whether or not to plot each dot at all. The output gives an appearance of moving segments in the lines constructed for the plotted dots. This subroutine requires a number of key words that interlock it with the main program with which it is used and the subroutine operation can best be explained in the section of the spheroidal radiator example covering the implementation of the D electric field.

Subroutine tests

During the course of this work, a number of small programs were written to check out the subroutine. These test programs are included here for two reasons. The first is that they give an idea of the output to be expected from the different subroutines so that when these subroutines are used with a main program, the programmer will know what to expect. The second reason is that they give the programmer an example to follow. It is far easier to show how software works than to try to explain it in words.

The first test is a program that was written not only to check out all the subroutines, but also to check out the entire operating system including the microfilm plotter that will be used to run the job. The statement listings are given on page 320. The output consists of a series of asterisks plotted on concentric circles following an identification frame. The entire pattern rotates between each frame. The number of frames, the number of circles, the number of asterisks on each circle, and the degree of rotation between each picture can be specified by the input parameters. An example of the output of one of these frames along with the corner and cut marks is shown in Figure 25.

After calling OUT. and CAMRAV. to set up the output and open the non-system file, the input variables which are called in with the read input statement are listed on the next page after Figure 25.

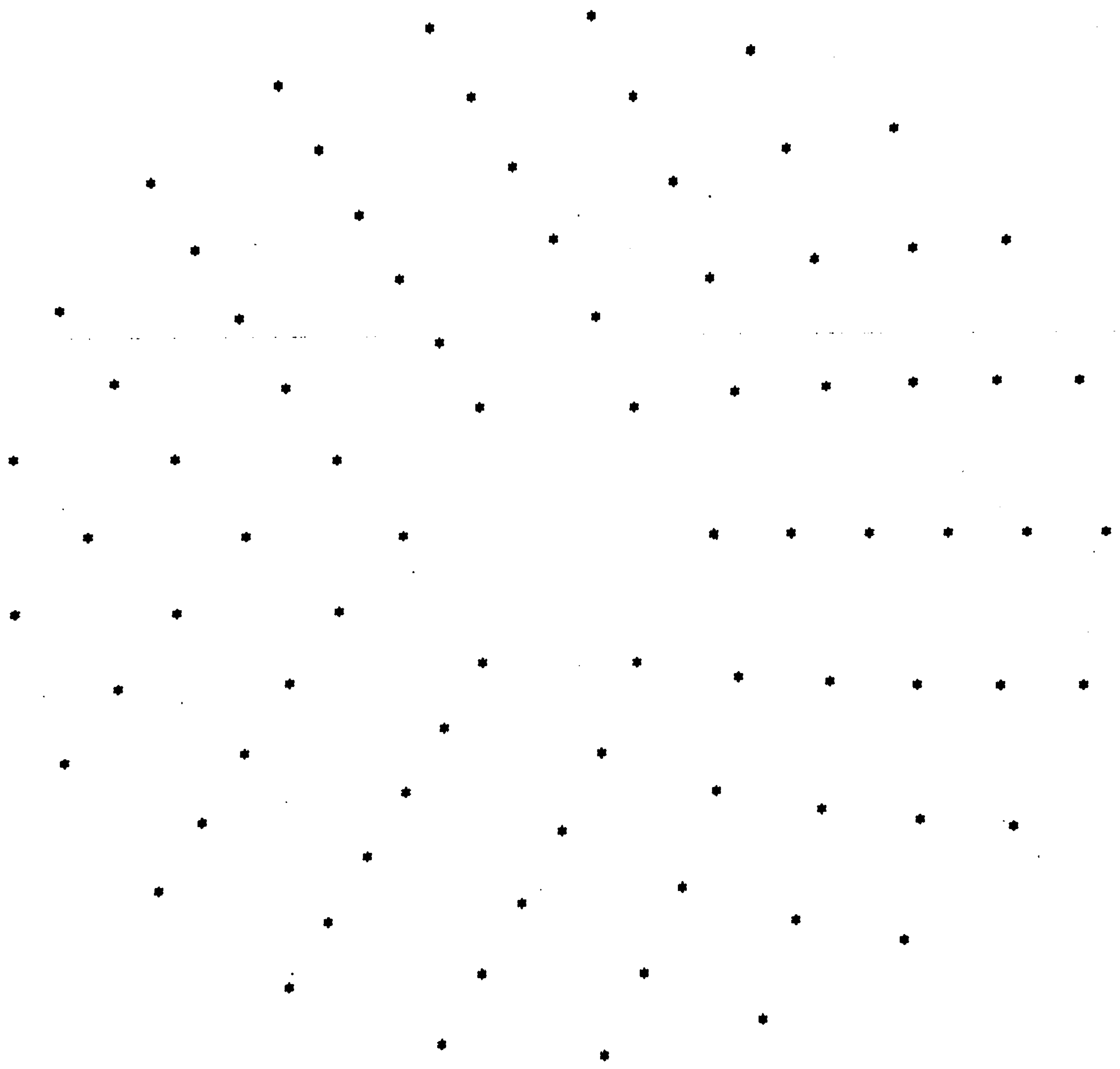


FIGURE 25

DOT OUTPUT OF TEST PROGRAM

RL	=	Radius of inner circle
RI	=	Increment of radius between each circle
THETA	=	Angle of rotation of first frame in radians
SD	=	Distance between each asterisk along the circle
TP	=	Increment of rotation between each frame
NP	=	Number of frames
NC	=	Number of circles

The circumference of each circle is divided by SD, the average distance between each asterisk, to give the number on each circle. SD is then adjusted to space the asterisks evenly around each circle. The various DO loops utilize the input data to generate the right amount of data. Of course, for more than 350 asterisks the DIMENSION statement would have to be changed. This test shows the use of
CALL SUBROUTINE()=APLOTV.(NR,X(L),Y(L),1,1,1,44,ERROR)-
to plot out only part of an array. "L" is preset at the start of the FRAMEV DO loop by setting it equal to "JL" and this supplies the starting address of the array through X(L) and Y(L). The number of asterisks in each frame is summed up through the statement "NR = NR - N - " after NR has been set equal to zero at the start of the frame. NR gives APLOTV. the number of points to be plotted from the array and the minus sign indicates the forward storage common to SCATRAN.

A printed output of this program is given on page 321.

A number of tests were run to check out the subroutines LINE, AXIS, PRINTS, RITE2S, GENSYM, RULE, and TABLSV. The statement listings are given on pages 322 to 323. The resulting output drawn from the printed output is shown in Figure 26. The printed output is given on page 323.

The results of these tests were deliberately drawn by hand rather than run on an SC 4020. This was to indicate how a subroutine might be debugged if the plotter was not available to run a tape. The command words to a small run are printed out. Each word is broken down into its component parts of operation code, location, etc., and the result plotted on an eight by eight to the inch graph paper. This will seem like very tedious work at first, but with a little practice the words can be interpreted almost by eye and a plot made in a reasonable time. Very often only key words have to be plotted, like the beginning and end letters of a line of print, to give a clear picture of what the finished product is going to look like or whether the subroutine is working.

SC 4020 special tape dump check

A special tape dump has been written to selectively dump tape in order to check the output of a long SC 4020 tape for correct data without obtaining an excessive amount

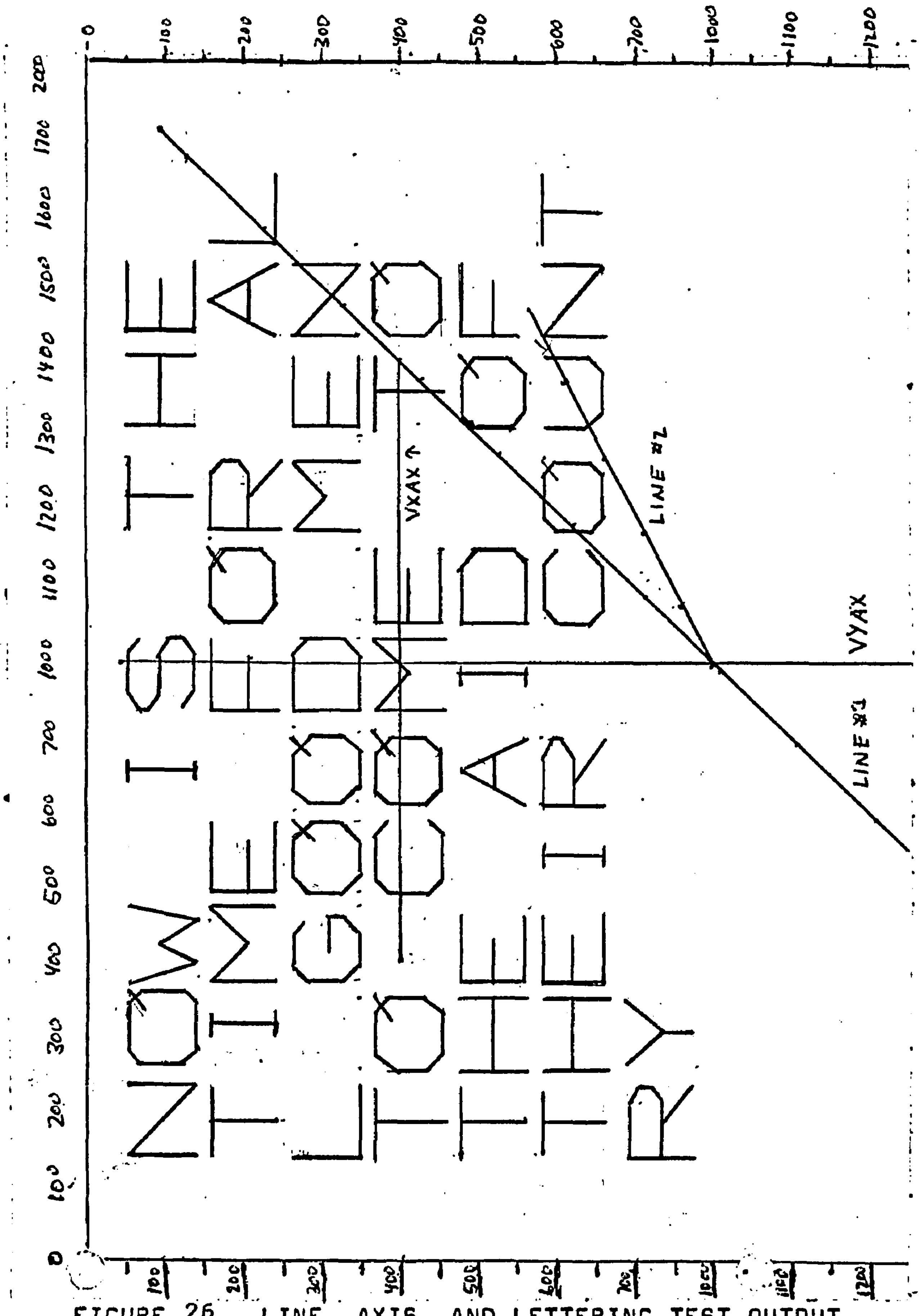


FIGURE 26 - LINE, AXIS, AND LETTERING TEST OUTPUT

of printed data. Normally, the IBM 1410 computer is used to dump tape because it is far cheaper to operate than the IBM 7094. However, if it was desired to check out each frame in a complete tape, the printed output could amount to several hundred pages, most of which would be completely useless. If the output can be checked with only a small fraction of the data printed out, as is often the case, the IBM 7094 can be used to great advantage. The subroutine SC 4020 TAPE DUMP was written to search every record on the tape for those that end in command word "460000000000." When this word is found, the next ten records are printed out. Since this is the advance film command, the final result is that only the first ten records of every frame are printed out. Since this many records usually have enough data to tell what is in each frame and since each frame can easily have over a hundred records, the advantage can be seen over printing out all the data on the tape and throwing most of it away.

Miscellaneous SCATRAN subroutines

Several additional subroutines have been checked out in SCATRAN. They are CUTIV, MARKV, BRITEV, XPANDV, and STOPTV. They are given in the statement listings but they are not described here because they are only used during the production of 9 X 9 hardcopy. Calls to these subroutines are listed in the SC 4020 Programmer's Manual.

Program running operations

A brief review of the steps necessary to run an SC 4020 tape program is helpful to the programmer to avoid having a run forced off the machine. It is a good idea to use a check list of both the necessary control cards and the subroutines needed for any call in the main program. This is particularly true if an SC 4020 run is attempted after an interval of time because some of the many details can easily be forgotten.

The first item after the usual ID cards and before any three-star control cards is the file control card. This card is green and has the information needed to determine the tape unit to be used and the writing density on it. The SCATRAN programmer's manual gives information as to how these cards are to be punched.[8] To run an SC 4020 tape, the card should be filled out as follows:

Column No.	Punch
2-3	**
7-10	FILE
17	*
19-21	B01 (zero, not letter o)
27	N
28	P
30	L
31	B
55-60	SC 4020

If the tape dump subroutine is to be used, the N and P in columns 27 and 28 should be replaced with an I in column

[8] R.Reeves, SCATRAN Reference Manual Section IV (Columbus: Ohio State University, 1964) PP. 82 - 83.

28 and the SC 4020 beginning at column 55 with the word REEL NUMBER followed by the number of the reel to be dumped.

This card is followed by the usual three-star SCATRAN control cards depending on the wishes of the programmer.

The next deck of cards is the main program. This can be either an original SCATRAN statement listing or it can be a symbolic binary deck headed by a READ SYMBOLIC BINARY card (a SCATRAN statement, not a three-star card) and any changes or modifications to the main program.

Subroutine deck assembly

The next task is to assemble the subroutine decks that are necessary to run the program. The first block is needed for every job.[9] They are

CAMRAV
(PLOT)
JOBNO
TAPE
XSCALV(MOD)
ID FRAME
END FRAME

Each one of these decks has to have a three-star LOAD

[9]The original XSCALV will run perfectly well with any job, but it has to have associated with it a small subroutine (SMXV) due to additional transfer vectors.

card preceding it but the decks themselves can be put in any order. This is the basic package for running tape.

There are several options available in this set of subroutines. They are listed below:

NORMAL	NO ID FRAME	NO EFRAME	PRINTED OUTPUT ONLY
Same as above	Substitute CLEAN UP for CAMRAV JOBNO ID FRAME	Substitute EFRAME II for EFRAME	Substitute TAPE III for TAPE ID FRAME END FRAME

In the PRINTED OUTPUT ONLY section, CAMRAV can be removed if the main program has a direct call to subroutine IDFM.

<u>Other subroutines</u>

To plot characters or dots - APLOTV

To plot lines - LINE and LINEV

To plot standard CHARACTRON printing - PRINTS and PRINTV

To plot variable sized letters and symbols such as titles, etc. - RITE2S, CHSIZV, RITE2V, HOLLV, TABL*V (any necessary table)

The following two subroutines were developed for the spheroidal radiator example and require some of the above routines even if they are not called directly:

GENSYM	RULE
requires	requires
CHSIZV	CHSIZV
RITE2V	
HOLLV	

Data cards

The last deck is followed by a three-star ***DATA card. The data cards themselves will depend on the main program but each program will have three additional data cards required by the call to CAMRAV. The first card contains the job number in the first six columns of the card. The rest of the card is blank. If no number at all is desired, a blank card can be used but the card, blank or not, must appear in the data deck. Because of the way the subroutine ID FRAME handles this information, only the number zero through nine, or a blank, can be used. Letters and symbols are not permitted.

The second card contains the information for the scaling ratios used in XSCALV. The entire card uses a (8F9.6) format to fill columns 1 - 72. The floating point field for each number covers nine columns for seven figure accuracy.

The other two columns are used for the decimal point and the sign, if necessary. The eight "nine column" fields have the following representation:

FIELD	COLUMNS	USE
1	1 - 9	Value of X at left boundary
2	10 - 18	Value of X at right boundary
3	19 - 27	Size of left margain*
4	28 - 36	Size of right margain*
5	37 - 45	Value of Y at bottom boundary
6	46 - 54	Value of Y at top boundary
7	55 - 63	Size of bottom margin*
8	64 - 72	Size of top margin*

*The numbers for the margin sizes have to be floating point integers between zero and 1024 followed by a decimal point. They are never negative and are usually "0." when producing movies.

The third data card contains the information to be printed on the ID frame. It has four alphanumeric fields, each of which is separated by a zero punch on the card. The format is as follows:

FIELD	COLUMNS	USE
1	1 - 17	Name of program
	18	Punch zero
2	19 - 35	Programmer's name
	36	Punch zero
3	37 - 47	Programmer's department
	48	Punch zero
4	49 - 56	Date in form of MO/DAY/YEAR
	57	Punch zero

The zero in columns 18, 36, 48, and 57 are required even if the rest of the card is blank. They serve as flags in the ID FRAME subroutine to reset the SC 4020 from the typing mode after reading in each of the above data items.[10]

With regard to the data deck, the CAMRAV statement should be treated as a READ INPUT statement. In other words, these three data cards are placed in relation to the other data cards in the same relationship that the CAMRAV call has to any READ INPUT statement in the main program.

Program core space requirements

SCATRAN programs can be run in two different modes, "time share" and "non-time share." When referring to core space, it is better to calculate in octal notation than decimal because both subroutine lengths and the core dump addresses are in octal notation. In this case "time share" addresses run from 00000 to 37777 and "non-time share" from 00000 to 77777.

Five items have to be allowed for to determine the size of a program. The SCATRAN monitor (input-output controls, etc.) always occupies all addresses up to 14400. The main program size is determined by the programmer and is therefore unknown. However, experience will give the programmer a feel for the size of any particular program. The SC 4020 subroutines follow the main program. As an aid, the space required for each subroutine listed is given in Appendix B. Subroutines from the SCATRAN library can use quite

[10]This card is omitted when using subroutine TAPE III.

a bit of room. Their length can be determined either from the memory map, if the program has been run once, or from the SCATRAN programmer's manual.[11] Finally, a small amount of room has to be allowed for a literals table that uses one word for each literal number in the main program.

Running time estimates

One of the most difficult decisions the programmer has to make is the time estimate to be used for each run. As long as check out work is being done, a time estimate of one minute is completely adequate and almost no job will run over this estimate.

However, it is when the programmer is running the final tapes for a movie that a dilemma has to be faced. Each run can be up to a half hour long. If the run takes longer than the time estimate, the job will automatically be forced off the computer and all the tape made up to that time will be lost. This is because the tape is rewound without a DISMOUNT message being given to the computer operator. The answer to this problem is to make certain how long a problem will run and then put down some excess time to cover unforseen contingencies.

The other horn of the dilemma is that if an excessive

[11] Ibid., Section V.

amount of time is put down and the program has a loop in it, all the time will again be lost due to the loop. Therefore, the time should be as short as possible in case a loop is encountered. This problem troubled the spheroidal radiator example run because the inherent loop in the way the successive points were calculated would not show up in some of the runs but would in others when the same problem was run with slightly different input data.

The only way around this is to make many different runs at different times, each covering some part of the input data spectrum, and each taking less than a minute. Ten minutes of computer time spent here could save much more when the longer runs are attempted because the timing can be more exact based on the times of the short runs and the programmer can be assured that looping will not occur.

Magnetic tape details

There are two means of dividing up the data that is placed on a reel: a minor division called a record, and a major division called a file.

A record consists of the amount of information that is written or read at one time. It can be of any length but in this case the size depends on the size of the input buffer of the particular SC 4020 that is going to be used. If the record is too long for the size of the buffer, it will be read until the buffer is full and the tape will then be moved to the start of the next record. All successive words in the record will be lost. Buffer lengths in the SC 4020 have three different sizes: 170 words, 1024 words, and 4096 words. Before a tape is produced, it is necessary to determine the buffer size of the SC 4020 that will be used to run the job. If this size cannot be determined, the 170 word length should be used. The longer buffers will accept the short records but at the cost of very poor efficiency both when writing the tape and when reading it with the SC 4020.

The beginning and ending of each record can be identified by a record gap. This is a 3/4 inch gap in which no recording appears. Its purpose is to allow time for the tape drive to accelerate and reduce speed when reading a record. The tape can only stop at a record gap which is why only complete records can be read. When more words are to be computed than the length of the buffers, by far the usual case, the write subroutine will automatically close the record at the right count, write a record gap, and then open a new record as the information continues to be computed.

The major division of a tape is the file. A file mark consists of a one word record that is used to tell the computer when a section of the tape is computed. It is used when it is desired to repeat a scene over several times. The SC 4020 will halt on a record gap only long enough to perform operations contained in that record, but when a file mark is reached, the machine stops and returns to operator control. The operator can then either go on to the next file or back-space and repeat the file already recorded. In this way, a small amount of tape can result in a great number of frames to increase the running time of a scene in a movie.

The tape density determines the capacity of the tape. At the present time, there are three densities in standard use: 200 characters per inch, 556 characters per inch, and 800 characters per inch. So little work is being done with 200 characters per inch that it can almost be considered to be obsolete. However, it is necessary to know that this density exists when defining low and high density tape. As an example, at OSU low and high density refer to 556 BPI (bits per inch) and 800 BPI respectively. At North American Aviation, low and high density refer to 200 BPI and 556 BPI. Therefore, a low density tape at OSU is a high density tape at North American.[12] At the present time, the SC 4020 will accept only 556 BPI tapes.

The formula given by IBM for a density of 556 BPI for the number of inches per record is:

$$\text{Inches per record} = .75 + .0108N$$

[12]A character is six bits of information. Since magnetic tape records six bits in one location across the tape, the terms "characters per inch" and "bits per inch" are interchangeable when refering to tape density.

where N is the number of words per record and the ".75" is the 3/4 of an inch allowed for the record gap. Since a reel of tape is 2500 feet long or 30,000 inches, we can calculate the number of words that can be placed on one reel of tape, assuming either 170 words per record or 4096 words per record.

At 170 words per record:

I = .75 + .0108 X 170 = 2.586 inches

Records = 30,000/2.586 = 11600

Words = 11600 X 170 = 1,972,000

At 4096 words per record:

I = .75 + .0108 X 4096 = 44.987 inches

Records = 30,000/44.987 = 667

Words = 667 X 4096 = 2,732,000

In the case of 4096 words per record, the number of words that can be placed on a reel of tape is approximately forty per cent greater than that which can be placed on the reel at 170 words per record. Therefore, tape should be written using the largest size records possible allowing for the SC 4020 input buffer size. In general, the output buffer size of the IBM 7094 computer will not be a problem.

One difficulty was encountered when producing tape at O.S.U. for running elsewhere. The original tape computed by the IBM 7094 seemed to be noisy in that non-repeatable errors were encountered when these tapes were run. The symptoms were that the SC 4020 would occassionally fail to recognize a valid command word on the tape and would give an error signal while advancing the tape to the end of the record. Naturally, all command words from that point to the end of the record were lost. This meant that the picture was spotty or if one of the missing words happened to be an advance film command, the following frame would be overlaid on the one already recorded. The SC 4020 does not have as high a degree of error checking as the IBM 7094 which is why the difficulty showed up only when the tapes were being actually run. After some investigation, it was found that the problem could be avoided entirely if the IBM 7094 tapes were copied on a small computer (IBM 1410) designed for this purpose. A perfect copy is made without any noise that might have been present in the original. The tape is only being copied rather than being computed so the costs of the copying operation are negligible compared to that of the original program.

A word of warning is necessary when copying tape with the IBM 1410. There are two copy routines available, CP 1 and CP 2. It is essential that CP 2 be used for this purpose and that this information be given to the computer operator before the tape is to be copied.

If CP 1 is used to copy a tape, the input buffer is hardware limited to accepting one hundred and five words per record. After the first one hundred and five words are copied, the tape is moved to the end of the record and all the succeeding words are lost. This means that if a tape has been written with 170 word records, about fourty per cent of the information is lost in making the copy. When copying tape that is written with 1024 word records, the result is disastrous.

SC 4020 subroutine listings

This section contains the operating subroutines used with the SC 4020 and BL 120 computer plotters. They are listed alphabetically below with the amount of octal core space required for each subroutine. The subroutines are not filed alphabetically but by groups according to their interlocking requirements and frequency of use. Also included are two tests for checking the subroutines and a tape dump for checking magnetic tape output.

A number of these subroutines will run under either a FORTRAN or SCATRAN operating system. They have asterisks in front of the name. They are of two types, ones that are not called from a main program, such as TABL-V, and ones that have an associated conversion program so that they can be called in SCATRAN. RITE2V and CHSIZV are of the second type and need RITE2S to be convected to the SCATRAN operating system. The preceding section on program running operations gives the data on which particular subroutines must be run together. Should it be desired to modify any of these subroutines, the section on the differences between FORTRAN and SCATRAN should be thoroughly understood before any changes are attempted.

The table on the following page gives the core space requirements for these subroutines.

Pages 243-325 contain the statement listings for the SC 4020 subroutines that were written to produce movies at OSU.

TABLE 8

SUBROUTINE CORE SPACE REQUIREMENTS

	NAME	OCTAL SPACE REQUIREMENTS	PAGE
	APLOTV	224	266
	CAMRAV	211	248
*	CHSIZV	251	282
	CLEANUP	023	265
	CUT MARKS	046	317
	EFRAME I	122	262
	EFRAME II	011	264
	GENSYM	222	306
*	HOLLV	037	286
	ID FRAME	252	258
	JOB NO	012	253
	LINE	043	299
*	LINEV	133	300
*	MARKV	042	319
	(PLOT)	030	252
*	PLOTV	026	269
*	POINTV	133	270
	PRINTS	050	272
*	PRINTV	133	273
	4 QUAD(PLOT)I	126	312
	4 QUAD(PLOT)II	226	314
	RITE2S	113	276
*	RITE2V	242	278
	RULE	174	309
*	TABL1V	321	287
*	TABL2V	115	293
*	TABL3V	110	296
*	TABL5V	031	298
	TAPE I	2377	243
	TAPE II	7777	245
	TAPE III	2155	247
	VXAX	063	302
*	VXAXV I	053	303
*	VXAXV II	024	305
	XSCALV	213	254

```
 ***    SCATRAN
                          TAPE1000
C               THESE SUBROUTINES OPEN THE FILE AND WRITE TAPE@
C               THIS IS TAPE I @
                SUBROUTINE()=OUT.(M)@
                MOUT = M@
                NORMAL EXIT@
                SUBROUTINE()=IDFM.()@
                CALLSUBROUTINE()=ZCLOCK.(MTIME)@
  LIST          FILE LIST(BETA.7.4.1.$FILES)@
                DEFINE POOL.ALPHA.4.172@
                ATTACH FILES.ALPHA.BETA.1@
                OPEN REWIND.$FILES@
                READ INPUT.5.(NUM)@
                CALL SUBROUTINE()=JOBNO.(NUM)@
                READ INPUT.SCALE.(XL.XU.BM.TM.YL.YU.GM.RM)@
                CALL SUBROUTINE()=XSCALV.(XL.XU.BM.TM)@
                CALL SUBROUTINE()=YSCALV.(YL.YU.GM.RM)@
F SCALE         (8F9.3)@
                READINPUT.5.(NAN.NSN.NDN.NFN.NGN.NHN.NJN.NKN.NLN
                   .NPN)@
                CALL SUBROUTINE() = IDINFO.
                (NAN.NSN.NDN.NFN.NGN.NHN.NJN.NKN.NLN.NPN)@
                NORMAL EXIT@
                END SUBPROGRAM@
                SUBROUTINE()=STORE.(ACC.N)@
                DIMENSION (WORD(172))@
                EXTERNAL(MOUT)@
                TRANSFER(AHEAD)PROVIDED(N.NE.0)@
                WORD(I)(=)ACC@
                I = I+1@
                TRANSFER(AHEAD)PROVIDED(I.G.169)@
                NORMAL EXIT@
  AHEAD         TRANSFER(EXTRA)PROVIDED(I.L.3)@
  PLOT          CALLSUBROUTINE()=WRNSFR.($FILES.WORD.I)@
                TRANSFER TO (END)PROVIDED(MOUT.E.1.OR.MOUT.E.3)@
                WRITEOUTPUT.S1.((WORD(K).K=0.1.K.L.I))@
  END           I = 0@
                NORMAL EXIT@
  EXTRA         WORD(I)(=).0.120000000000@
                I=I+1@
                TRANSFER(EXTRA)PROVIDED(I.L.3)@
                TRANSFER(PLOT)@
                NORMAL EXIT@
 F S1           (5T.8(2X.K12))@
                END SUBPROGRAM@
                SUBROUTINE() =CLOSTP.()@
                EXTERNAL(MOUT)@
                TRANSFER TO (S4) PROVIDED (MOUT.E.1.OR.MOUT.E.2)
                   @
                CLOSE UNLOAD.$FILES@
```

```
              TRANSFER (S5)@
   S4         CLOSE REWIND.$FILES$@
   S5         CALLSUBROUTINE(MTIME)=RCLOCK.()@
              NTIME=MTIME*60@
              WRITENOHEADING.OUT4.(NTIME.MTIME)@
F OUT4        (///10T.Q*EXECUTION TIME USED BY THIS PROGRAM IS
                 *.
               3PB6.2.Q* SECONDS*/46T.Q*OR*.-3PB6.3.Q* MINUTES
                 *)@
              CALLSUBROUTINE()=ENDJOB.()@
              NORMAL EXIT@
              END SUBPROGRAM@
              END SUBPROGRAM@
         END  END PROGRAM@
```

```
 ***    SCATRAN
C              THESE SUBROUTINES OPEN THE FILE AND WRITE TAPE@
C              THIS IS TAPE II @
               SUBROUTINE()=IDFM.()@
               CALLSUBROUTINE()=ZCLOCK.(MTIME)@
  LIST         FILE LIST(BETA,7,4,1,$FILES)@
               DEFINE POOL,ALPHA,4,1030@
               ATTACH FILES,ALPHA,BETA,1@
               OPEN REWIND,$FILES@
               READ INPUT,5,(NUM)@
               CALL SUBROUTINE()=JOBNO.(NUM)@
               READ INPUT,SCALE,(XL,XU,BM,TM,YL,YU,GM,RM)@
               CALL SUBROUTINE()=XSCALV.(XL,XU,BM,TM)@
               CALL SUBROUTINE()=YSCALV.(YL,YU,GM,RM)@
F SCALE        (8F9.3)@
               READINPUT,5,(NAN,NSN,NDN,NFN,NGN,NHN,NJN,NKN,NLN
                   ,NPN)@
               CALL SUBROUTINE() = IDINFO.
               (NAN,NSN,NDN,NFN,NGN,NHN,NJN,NKN,NLN,NPN)@
               NORMAL EXIT@
               END SUBPROGRAM@
               SUBROUTINE()=STORE.(ACC,N)@
               DIMENSION(WORD(1030))@
               TRANSFER(AHEAD)PROVIDED(N.NE.0)@
               WORD(I)(=)ACC@
  A2           I = I+1@
               TRANSFER(PLOT)PROVIDED(I.G.1023)@
               NORMAL EXIT@
  AHEAD        TRANSFER(EXTRA)PROVIDED(I.L.3)@
  A1           TRANSFER(CLOSES)PROVIDED(N.E.4)@
  PLOT         CALLSUBROUTINE()=WRNSFR.($FILES,WORD,I)@
               TRANSFER(END)PROVIDED(N.E.1.OR.I.E.1024)@
               TRANSFER(OUT)PROVIDED(N.E.2)@
               NORMAL EXIT@
  OUT          I=I-1@
               NORMAL EXIT@
  END          I = 0@
               NORMAL EXIT@
  EXTRA        WORD(I)(=).O.120000000000@
               I=I+1@
               TRANSFER(EXTRA)PROVIDED(I.L.3)@
               TRANSFER(A1)@
  CLOSES       WORD(1)(=).O.460000000000@
               TRANSFER(A2)@
               END SUBPROGRAM@
               SUBROUTINE() =CLOSTP.()@
               CALL SUBROUTINE()=LASTFM.()@
               CLOSE UNLOAD,$FILES@
  S5           CALLSUBROUTINE(MTIME)=RCLOCK.()@
               NTIME=MTIME*60@
               WRITENOHEADING,OUT4,(NTIME,MTIME)@
```

```
F OUT4        (///10T.Q*EXECUTION TIME USED BY THIS PROGRAM IS
                 *.
               3PB6.2.Q* SECONDS*/46T.Q*OR*.-3PB6.3.Q* MINUTES
                 *)@
              CALLSUBROUTINE()=ENDJOB.()@
              NORMAL EXIT@
              END SUBPROGRAM@
        END   END PROGRAM@
```

```
 ***    SCATRAN
                   TAPE1000
C               THESE SUBROUTINES OPEN THE FILE AND WRITE TAPE@
C               THIS IS TAPE III @
C               DO NOT USE ID FRAME OR EFRAME @
                SUBROUTINE()=IDFM.()@
                CALLSUBROUTINE()=ZCLOCK.(MTIME)@
                READ INPUT.5.(NUM)@
                CALL SUBROUTINE()=JOBNO.(NUM)@
                READ INPUT.SCALE.(XL.XU.BM.TM.YL.YU.GM.RM)@
                CALL SUBROUTINE()=XSCALV.(XL.XU.BM.TM)@
                CALL SUBROUTINE()=YSCALV.(YL.YU.GM.RM)@
F SCALE         (8F9.3)@
                NORMAL EXIT@
                END SUBPROGRAM@
                SUBROUTINE()=STORE.(ACC.N)@
                DIMENSION (WORD(172))@
                TRANSFER(AHEAD)PROVIDED(N.NE.0)@
                WORD(I)(=)ACC@
                I = I+1@
                TRANSFER(AHEAD)PROVIDED(I.G.169)@
                NORMAL EXIT@
  AHEAD         WRITEOUTPUT.S1.((WORD(K).K=0.1.K.L.I))@
  END           I = 0@
                NORMAL EXIT@
F S1            (5T.8(2X.K12))@
                END SUBPROGRAM@
                SUBROUTINE() =CLOSTP.()@
  S5            CALLSUBROUTINE(MTIME)=RCLOCK.()@
                NTIME=MTIME*60@
                WRITENOHEADING.OUT4.(NTIME.MTIME)@
F OUT4          (///10T.Q*EXECUTION TIME USED BY THIS PROGRAM IS
                   *.
                 3PB6.2.Q* SECONDS*/46T.Q*OR*.-3PB6.3.Q* MINUTES
                   *)@
                CALLSUBROUTINE()=ENDJOB.()@
                NORMAL EXIT@
                END SUBPROGRAM@
                SUBROUTINE()=IDFIN.()@
                NORMAL EXIT@
                END SUBPROGRAM@
                SUBROUTINE()=EFRAME.()@
                NORMAL EXIT@
                END SUBPROGRAM@
          END   END PROGRAM@
 ***    FAP
                      CMRV0000
        COUNT    150
        ENTRY    CAMRAV
        ENTRY    FRAMEV
        ENTRY    RESETV
        ENTRY    FORMV
*  CRT  HARDWARE MANIPULATION ROUTINES
*       CALL     CAMRAV(N)
*         N      =9 FOR 9*9 CAMERA
*                =35 FOR 35 MM CAMERA
*                =OTHER FOR BOTH (ARG MAY BE DELETED)
*
**************************************************************
                 ************
        SPACE
*       CALL     FRAMEV
*    ADVANCES  FILM FRAME
*
**************************************************************
                 ************
        SPACE
*       CALL     RESETV
*    ADVANCES  FILM FRAME
*    LEAVES  TYPE MODE
*    SETS HARDWARE COUNTER TO (0,0)
*    SETS INTENSITY HIGH
*       CALL     FORMV
*    CAUSES FORM TO BE DISPLAYED
        SPACE
*       CALL FRAMEV(N) - IF (N) IS ZERO OR MISSING THE JOB NU
                 MBER
*       AND FRAME COUNT WILL BE DISPLAYED IN THE UPPER RIGHT
                 HAND
*       CORNER OF THE FRAME. IF (N) IS NON-ZERO THE PRINTING
                 WILL
*       BE SUPPERSSED. IF (N) IS NEGATIVE CORNER MARKS WILL B
                 E
*       SUPPRESSED.
*
**************************************************************
                 ************
        SPACE
*       CALL FRAMEV(N) - IF (N) IS ZERO OR MISSING THE JOB NU
                 MBER
*       AND FRAME COUNT WILL BE DISPLAYED IN THE UPPER RIGHT
                 HAND
*       CORNER OF THE FRAME. IF (N) IS NON-ZERO THE PRINTING
                 WILL
*       BE SUPPERSSED. IF (N) IS NEGATIVE CORNER MARKS WILL B
                 E
```

```
*       SUPPRESSED.
*
M****************************************************************
                     ************
        SPACE
FRAMEV  SXA      X4,4
        CAL      FRAME
 F1     CALL     (PLOT)
 FO     CALL     (CLEAN                  FINISH PRECEDING RECORD
 F2     TRA      FIRST
        SXA      X1,1
        CLA      CAMERA
        CALL     FMCT
        LYA      X4,4
        CAL      1,4
        ANA      MASK1                   LOC 777777700000
        STO      EXTEST                  EXIT TEST
        TNZ      AXT
        CLA*     1,4
        TMI      NOCOR              NO CORNER MARKS
 AXT    AXT      4,1
 COR    CAL      CORNER,1                    PLOT CORNER DOTS
        CALL     (PLOT)
        CAL      CORNR2+1,1
        CALL     (PLOT)
        TIX      COR,1,1
        LXA      X4,4
 NOCOR  ZET      EXTEST
        TRA      PRNT
        ZET*     1,4
        TRA      X1             DONT PRINT ID
 PRNT   STZ      DBCD
        CAL*     $(CTJB)
        STA      *+1
        CLA      **
        STA      DBCD
        ARS      18
        STO      DBCD+1
        AXT      2,4
 PRN1   AXT      18,1                    CONVERT FRAME COUNTS
        LDQ      DBCD+2,4
        STZ      DBCD+2,4
 PRN2   PXD      0
        DVP      TENAD
        ALS      18,1
        ACL      DBCD+2,4
        SLW      DBCD+2,4
        TIX      PRN2,1,6
        ACL      BLANKS                  LOC 606060000000
        SLW      DBCD+2,4
        TIX      PRN1,4,1
```

```
       AXT     0,1                    SET UP TO TYPE
PRN4   CAL     CORDJ,1
       CALL    (PLOT)
       TXI     *+1,1,1
       TXL     PRN4,1,6
X1     AXT     **,1
X4     AXT     **,4
       ZET     EXTEST
       TRA     1,4
       TRA     2,4
FIRST  CLA     NOOP                   DO ONCE
       STO     F2
       CLA*    $(JOBN)
       STA     *+1
       CLA     **                     JOB NO.
       LDQ     STOPT                  STOP TYPE 1200
       LGR     12
       ALS     6
       ACL     BDASH                  INSERT DASH
       LGL     6
       SLW     JOB
       STQ     JOB-1
       TRA     F2
*
RESETV SXA     X4,4
       CAL     RESET
       TRA     F1
FORMV  SXA     XX4,4                  DISPLAY FORMS
       CAL     FORM
       CALL    (PLOT)
       CALL    (CLEAN
XX4    AXT     **,4
       TRA     1,4
*
       EJECT
CAMRAV SXA     CX4,4
       CALL    IDFM
       LXA     CX4,4
       CLA*    1,4
       PAX     ,4
       CAL     SBC
       LDQ     QSBC
       TXH     CX,4,35
       TXH     D35,4,34
       TXL     CX,4,8
       TXH     CX,4,9
       CAL     SC2
       LDQ     QSC2
       TRA     CX
D35    CAL     SC1
       LDQ     QSC1
```

```
 CX     STO     CAMERA
        CALL    (PLOT)
        CALL    IDFIN
 CX4    AXT     **,4
        TRA     2,4
        VFD     2/3,6/15,10/0,2/0,6/0,10/0
        VFD     2/3,6/15,10/0,2/0,6/0,10/1023
        VFD     2/3,6/15,10/1023,2/2,6/0,10/0
        VFD     2/3,6/15,10/1023,2/2,6/0,10/1023
CORNER  VFD     2/3,6/0,10/0,2/1,6/15,10/0
        VFD     2/3,6/0,10/0,2/0,6/15,10/1023
        VFD     2/3,6/0,10/1023,2/1,6/15,10/0
CORNR2  VFD     2/3,6/0,10/1023,2/0,6/15,10/1023
 RESET  OCT     560000000000
 FRAME  OCT     460000000000
 FORM   OCT     500000000000
 SBC    OCT     430000000000
 SC1    OCT     410000000000
 SC2    OCT     420000000000
 QSBC           1,,1
 QSC2           0,,1                    9*9
 QSC1           1,,0                    35 MM
CAMERA          0,,1                   9X9 IF NOT SELECTED
 STOPT  OCT     120000000000
 DBCD   PZE     0
        PZE     0
 CORDF  OCT     201610600014             COORD FOR FRAMES
        PZE     0                       DASH NO AND STOP TYP
 JOB    PZE     0
 CORDJ  OCT     201642600000             COORD FOR   JOB. NO.
EXTEST  PZE     0                       EXIT TEST
 NOOP   NOP     0
 BDASH  OCT     000000000040
 MASK1  OCT     777777700000
BLANKS  OCT     606060000000
 TENAD  PZE     10                      10 BI. SCALE 35
        END
```

```
***     FAP
                          PLOT0000
        COUNT    25
*       USED TO PUT SCALED DATA INTO BUFFERS
        ENTRY    (PLOT)
        ENTRY    LASTFM
        ENTRY    (CLEAN
(PLOT)  SXA      (X4),4
        SLW      ACC
        CALL     STORE,ACC,N
 (X4)   AXT      **,4
        TTR      1,4
(CLEAN  SXA      (X4),4
        CALL     STORE,ACC,N1
        TTR      (X4)
LASTFM  SXA      (X4A),4
        CALL     EFRAME
        CALL     STORE,ACC,N1
 (X4A)  AXT      **,4
        TTR      1,4
 ACC    PZE
 N      PZE
 N1     OCT      1
        END
```

```
***     FAP
                           JOBN0000
        COUNT   10
*               JOB NUMBER              FRAME COUNT
        ENTRY   JOBNO
        ENTRY   (JOBN)
        ENTRY   FMCT
        ENTRY   (CTJB)
JOBNO   CLA*    1,4
        STO     (JOBN)+1
        TRA     2,4
(JOBN)  PZE     *+1
        BCI     1,
 FMCT   ADD     CTJB
        STO     CTJB
        TRA     1,4
(CTJB)  PZE     *+1
 CTJB   PZE
        END
 ***    FAP
        COUNT    100
        ENTRY    XSCALV
        ENTRY    YSCALV
        ENTRY    NXV
        ENTRY    NYV
        ENTRY    SCERRV
        ENTRY    SERSAV
        ENTRY    SERREV
*       CALL     XSCALV(XL,XU,ML,MU)
*       CALL     YSCALV(YL,YU,ML,MU)
*            XL =LOWER LIMIT OF X VALUES (LEFT)
*            XU =UPPER LIMIT OF X VALUES (RIGHT)
*            YL =LOWER LIMIT OF Y VALUES (BOTTOM)
*            YU =UPPER LIMIT OF Y VALUES (TOP)
*            ML =LEFT OR BOTTOM MARGIN (FLOATING INTEGER)
*            MU =RIGHT OR TOP MARGIN (FLOATING INTEGER)
*
**************************************************************
                 ************
XSCALV  SXA      X1,1
        AXT      0,1
        TRA      *+3
YSCALV  SXA      X1,1
        AXT      1,1
        SXA      X4,4
        SXA      X2,2
        CLA*     3,4
        STP      TEST
        NZT      TEST
        TSX      FLOAT,2
        STO      LOW,1                  ML
        STO      LO
        CLS*     4,4
        STP      TEST
        NZT      TEST
        TSX      FLOAT,2
        FAD      =1023.
        STO      HIGH,1                 MU
        STO      HI
        CLA*     1,4                    VL
        STO      VL
        CLA*     2,4                    VL
        STO      VU
        TSX      (AB),2                 COMPUTE LIN. TRAN. COEFFS
                 .
        LDQ      VL
        FMP      A,1
        FRN
        FAD      B,1
        FRN
        FSB     LOW,1
        FRN
        CHS
        FAD     LOW,1
        STO     LOW,1                   CORRECTED ML
        LDQ     VU
        FMP     A,1
        FRN
        FAD     B,1
        FRN
        FSB     HIGH,1
        FRN
        CHS
        FAD     HIGH,1
        STO     HIGH,1
        TSX     (AB),2                  RECOMPUTE A,B
        CLA     LO
        FSB     =,5
        STO     LOW,1
        CLA     HI
        FAD     =,5
        STO     HIGH,1
 X1     AXT     **,1
X2      AXT     **,2
X4      AXT     **,4
        TRA     5,4
(AB)    CLA     VU                      VU
        FSB     VL                      -VL
        STO     A,1
        CLA     HIGH,1
        FSB     LOW,1                   1023-MU-ML
        FDP     A,1
        STQ     A,1
        FMP     VL
        CHS
        FAD     LOW,1
        STO     B,1
        TRA     1,2
VL
VU
 FLOAT  ARS      18
        ORA     =155B8
        FAD     =155B8
        TRA      1,2
 TEST
* SCALING DATA BLOCK

 LOW

 HIGH
 B

 A
 HI
 LO
NXV     SXA       NX1,1                   X ENTRY POINT
        AXT       0,1                     INDICATE X
        TRA       N.
NYV     SXA       NX1,1                   Y ENTRY POINT
        AXT       1,1                             INDICATE Y
*  LINEAR SCALING AND FLOAT-FIX CONVERSION
 N.     PXD       0
        STD*      ERRORX,1
        LDQ*      1,4
N.1     FMP       A,1                     REENTRY FROM N.-L. SCALIN
                  G
        FRN
        FAD       B,1
        FRN
        CAS       LOW,1
        CAS       HIGH,1
        TRA       ERR
        TRA       ERR
 N.2    UFA       =0233000000000
        FRN
        ANA       =01777
        ALS        18
NX1     AXT       **,1
        TRA       2,4
ERR     CLA       =1817
        STD*       ERRORX,1
        SXA        (X4),4
        CAL        =0000000600000
        CALL       (PLOT)
 (X4)   AXT       **,4
        PXD       0
        TRA       NX1
SCERRV  CLA       1,4
        STA       ERRORX
        CLA       2,4
        STA       ERRORY
        TRA       3,4
SERSAV  CLA       ERRORX
        STA*      1,4
        CLA       ERRORY
        STA*      2,4
        TRA       3,4
SERREV  CLA*      1,4
        STA       ERRORX
        CLA*      2,4
        STA       ERRORY
        TRA     3,4
ERRORY          *                           ADDRESS=LOC (ERROR CELL)
ERRORX          *
        END
```

```
 ***   FAP
                     IDFR0000
       COUNT    300
*   THIS SUBROUTINE WRITES THE ID FRAME
       ENTRY    IDINFO
       ENTRY    IDFIN
IDINFO CAL*     1,4
       SLW      TITLE
       CAL*     2,4
       SLW      TITLE+1
       CAL*     3,4
       ORS      TITLE+2
       CAL*     4,4
       SLW      NAME
       CAL*     5,4
       SLW      NAME+1
       CAL*     6,4
       ORS      NAME+2
       CAL*     7,4
       SLW      DEPT
       CAL*     8,4
       ORS      DEPT+1
       CAL*     9,4
       SLW      DATE
       CAL*     10,4
       ORS      DATE+1
       TRA      11,4
 IDFIN SXA      ID4,4
       SXA      ID2,2
       SXA      ID1,1
       CAL      IMAGE                      NON-EXPANDED IMAGE
       TSX      BUFR,4
       CAL      RST                   NEW FILM FRAME, AND
       TSX      BUFR,4                STOP TYPE AND SET BRITE
       CLA*    $(JOBN)
       STA      *+1
 JBNO  LDQ      **                    FETCH JOB NO.
* CONVERT JOB NUMBER AND PRINT IN LARGE LETTERS
* WITH AXES INSERTED BETWEEN LETTERS.
       AXT      7,2
 (TAB) BSS
 I.1   PXD      TAB1
       LGL      6                     FETCH NEXT CHARACTER
 I.2   STQ      TEM
       CAS      ABLNK                LOC 000000000060
       TRA      I.34
       TRA      I.34
       ALS      2                     DIGIT *4, PLUS
       ACL      (TAB)                 TABLE ORIGIN
       STA      I.3                   =TABLE ENTRY
       AXT      4,1
```

```
* LOOK UP VECTOR COMMANDS FOR THIS DIGIT
 I.3    CAL     **.1                        VECTOR COMMAND
        ACL     OFFSET                      INCREMENT X-ORIGIN
        TSX     BUFR.4                      TRANSMIT VECTOR
        TIX     I.3.1.1                     COMPLETE DIGIT
 I.34   CAL     AXIS.2                      OUTPUT AXIS
        TSX     BUFR.4
        CAL     OFFSET
        ACL     S64B17               LOC 64B17
        SLW     OFFSET
        SPACE
        LDQ     TEM
        TNX     I.4.2.1
        TXH     I.1.2.3
        TXL     I.1.2.2
        CLA     ATEN                 TEN IN ADDR.
        TRA     I.2
 I.4    AXT     WORDS-INSTAL.1
        CAL     WORDS.1
        TSX     BUFR.4
        TIX     *-2.1.1
*   OUTPUT EIGHTH AXIS TO END ID FRAME
        CAL     AXIS-8
        TSX     BUFR.4
 ID1    AXT     **.1
 ID2    AXT     **.2
 ID4    AXT     **.4
        TRA     1.4
 BUFR   SXA     BX4.4
        TSX     $(PLOT).4
 BX4    AXT     **.4
        TRA     1.4
*    TABLE OF DIGITS FOR JOB NUMBER   .........
 TAB0   VFD     2/3.6/24.10/8.2/3.6/32.10/32
        VFD     2/3.6/24.10/32.2/2.6/32.10/0
        VFD     2/3.6/24.10/56.2/0.6/32.10/32
        VFD     2/3.6/24.10/32.2/1.6/32.10/64
 TAB1   VFD     2/3.6/16.10/16.2/3.6/16.10/16
        VFD     2/3.6/0.10/32.2/0.6/62.10/0
        VFD     2/3.6/16.10/32.2/0.6/2.10/62
        VFD     2/3.6/32.10/16.2/2.6/0.10/64
 TAB2   VFD     2/3.6/24.10/8.2/3.6/16.10/16
        VFD     2/3.6/24.10/32.2/2.6/24.10/0
        VFD     2/3.6/48.10/56.2/0.6/40.10/24
        VFD     2/3.6/48.10/8.2/2.6/0.10/64
 TAB3   VFD     2/3.6/48.10/8.2/2.6/8.10/0
        VFD     2/3.6/24.10/56.2/0.6/24.10/8
        VFD     2/3.6/24.10/32.2/2.6/16.10/32
        VFD     2/3.6/48.10/56.2/0.6/24.10/48
 TAB4   VFD     2/3.6/0.10/40.2/1.6/63.10/64
        VFD     2/3.6/32.10/40.2/0.6/48.10/0
        VFD     2/3,6/48,10/8,2/2,6/0,10/48
        VFD     2/3,6/40,10/56,2/1,6/8,10/48
 TAB5   VFD     2/3,6/40,10/48,2/0,6/0,10/0
        VFD     2/3,6/8,10/8,2/0,6/24,10/0
        VFD     2/3,6/40,10/0,2/2,6/28,10/24
        VFD     2/3,6/40,10/40,2/0,6/28,10/52
 TAB6   VFD     2/3,6/32,10/40,2/0,6/40,10/0
        VFD     2/3,6/20,10/8,2/2,6/24,10/40
        VFD     2/3,6/28,10/28,2/3,6/20,10/64
        VFD     2/3,6/36,10/56,2/1,6/16,10/44
 TAB7   VFD     2/3,6/48,10/8,2/2,6/0,10/0
        VFD     2/3,6/24,10/56,2/0,6/63,10/0
        VFD     2/3,6/16,10/32,2/3,6/56,10/64
        VFD     2/3,6/40,10/48,2/1,6/8,10/8
 TAB8   VFD     2/3,6/48,10/8,2/2,6/0,10/0
        VFD     2/3,6/48,10/56,2/0,6/63,10/0
        VFD     2/3,6/48,10/8,2/2,6/0,10/64
        VFD     2/3,6/48,10/56,2/1,6/63,10/64
 TAB9   VFD     2/3,6/27,10/35,2/1,6/23,10/43
        VFD     2/3,6/24,10/8,2/3,6/20,10/20
        VFD     2/3,6/24,10/32,2/2,6/20,10/0
        VFD     2/3,6/40,10/56,2/0,6/44,10/20
 DASH   VFD     2/3,6/40,10/16,2/3,6/0,10/32
        VFD     2/3,6/0,10/56,2/2,6/8,10/32
        VFD     2/3,6/40,10/56,2/0,6/0,10/40
        VFD     2/3,6/0,10/16,2/1,6/8,10/40
*  TABLE OF AXES TO OUTLINE ID FRAME
        VFD     06/30,30/0
        VFD     06/30,30/4
        VFD     06/30,30/1019
        VFD     06/30,30/1023
        VFD     06/32,12/0,18/1023
        VFD     06/32,12/4,18/1023
        VFD     06/32,12/1019,18/1023
        VFD     06/32,12/1023,18/1023
 AXIS   BES
*     JOB ID BLOCK
INSTAL  OCT     200536460144,303146606263,216325606445            O
                .S.U.
        OCT     316525516231,637012000000                         O
                .S.U. CONT.
        OCT     200536450226,644433602346                         N
                .C.L.
        OCT     444733604321,223312000000
        OCT     200536240310,216325606060                         D
                ATE
 DATE   OCT     0000000000- ,000012000000                         S
                PACE FOR DAT
        OCT     200536631130,316343256060                         T
                ITLE
 TITLE  OCT     606060606060,606060606060,000000000012            S
                PACE FOR TIT
        OCT     200536511212,256364514560,634660606060       R
                ETURN TO
 NAME   OCT     606060606060,606060606060,000000000012       S
                PACE FOR NAM
        OCT     200536241274,254763336060                    D
                EPT.
 DEPT   OCT     0000000000   ,000000000012                   S
                PACE FOR DEP
*  TABLE OF VECTORS .... ARROW INDICATING THIS JOB
        VFD     2/3,6/63,10/1000,2/1,6/63,10/900
        VFD     2/3,6/63,10/1000,2/0,6/63,10/900
        VFD     2/3,6/53,10/990,2/1,6/53,10/900
        VFD     2/3,6/53,10/990,2/0,6/53,10/900
        VFD     2/3,6/63,10/986,2/0,6/4,10/902
        VFD     2/3,6/63,10/986,2/1,6/4,10/898
        VFD     2/3,6/63,10/923,2/0,6/4,10/906
        VFD     2/3,6/63,10/923,2/1,6/4,10/894
        VFD     2/3,6/63,10/860,2/0,6/4,10/910
        VFD     2/3,6/63,10/860,2/1,6/4,10/890
 WORDS  BES
 RST    VFD     06/56,30/0
 TEM
OFFSET          444,,256
IMAGE   OCT     450000000000          STANDARD IMAGE
 ATEN   PZE     10
 APLNK  OCT     000000000060
S64B17  OCT     000100000000
        END
```

```
***     FAP
                   EFRM0000
        CCUNT    100
*       CLOSING  ID FRAME
        ENTRY    EFRAME
EFRAME  SXA      (X4),4
        SXA      (X1),1
        CAL      FMADV
        CALL     (PLOT)
        CALL     (CLEAN
        AXT      WORDS-INSTAL,1
        CAL      WORDS,1
        TSX      $(PLOT),4
        TIX      *-2,1,1
        CALL     (CLEAN
        AXT      3,1
        CALL     WEF
        PZE      =HFILE
        PZE      *+2
        TIX      *-3,1,1
 (X1)   AXT      **,1
 (X4)   AXT      **,4
        TRA      1,4
 FMADV  OCT      460000000000
*  TABLE OF VECTORS FOR (END) IN LARGE LETTERS
INSTAL  VFD      2/3,6/00,10/431,2/0,6/18,10/516
        VFD      2/3,6/00,10/422,2/0,6/63,10/480
        VFD      2/3,6/39,10/422,2/2,6/00,10/480
        VFD      2/3,6/39,10/422,2/2,6/00,10/543
        VFD      2/3,6/00,10/431,2/0,6/18,10/489
        VFD      2/3,6/22,10/431,2/2,6/00,10/507
        VFD      2/3,6/22,10/431,2/2,6/00,10/516
        VFD      2/3,6/30,10/431,2/2,6/00,10/489
        VFD      2/3,6/30,10/431,2/2,6/00,10/534
        VFD      2/3,6/00,10/461,2/0,6/09,10/480
        VFD      2/3,6/00,10/453,2/0,6/09,10/507
        VFD      2/3,6/00,10/461,2/0,6/09,10/534          E
                 ND OF E     1
        VFD      2/3,6/00,10/480,2/0,6/63,10/480
        VFD      2/3,6/29,10/489,2/2,6/44,10/480
        VFD      2/3,6/00,10/518,2/0,6/44,10/480
        VFD      2/3,6/00,10/527,2/0,6/63,10/480
        VFD      2/3,6/09,10/480,2/2,6/00,10/480
        VFD      2/3,6/09,10/518,2/2,6/00,10/480
        VFD      2/3,6/00,10/489,2/0,6/44,10/499
        VFD      2/3,6/29,10/489,2/2,6/44,10/499
        VFD      2/3,6/09,10/480,2/2,6/00,10/543
        VFD      2/3,6/09,10/518,2/2,6/00,10/543          E
                 ND OF N     2
        VFD      2/3,6/00,10/546,2/0,6/63,10/480
        VFD      2/3,6/29,10/546,2/2,6/00,10/480
        VFD     2/3,6/19,10/575,2/2,6/19,10/480
        VFD     2/3,6/00,10/594,2/0,6/25,10/499
        VFD     2/3,6/19,10/594,2/0,6/19,10/524
        VFD     2/3,6/29,10/575,2/0,6/00,10/543
        VFD     2/3,6/00,10/555,2/0,6/45,10/489
        VFD     2/3,6/15,10/555,2/2,6/00,10/489
        VFD     2/3,6/15,10/570,2/2,6/15,10/489
        VFD     2/3,6/00,10/585,2/0,6/15,10/504
        VFD     2/3,6/15,10/585,2/0,6/15,10/519
        VFD     2/3,6/15,10/555,2/2,6/00,10/534                   E
                ND OF D     3
*       TABLE OF VECTORS    ARROW INDICATING THIS JOB
        OCT     776030777604
        OCT     776030577604
        OCT     752042553604
        OCT     752042753604
        OCT     776046411606
        OCT     776046611602
        OCT     776145411612
        OCT     776145611576
        OCT     776244411616
        OCT     776244611572
*       TABLE OF AXES TO OUTLINE ID FRAME
        OCT     300000000000
        OCT     300000000004
        OCT     300000001773
        OCT     300000001777
        OCT     320000001777
        OCT     320004001777
        OCT     321773001777
        OCT     321777001777
        OCT     201122600042
        OCT     606330316260
        OCT     475146275121
        OCT     444431452760
        OCT     227060606060
        OCT     223360275146
        OCT     652562120000
WORDS   BES
        END
```

```
 ***    FAP
                          ENFM1000
        COUNT    3
*   USE THIS EFRAME SUBROUTINE FOR NO END FRAME ON THE TAPE
        ENTRY    EFRAME
EFRAME  SXA      (X4),4
        CAL      FMADV
        CALL     (PLOT)
        CALL     (CLEAN
 (X4)   AXT      **,4
        TTR      1,4
 FMADV  OCT      460000000000
        END
 ***    FAP
                            EXT 0000
        COUNT    50
*        CLEAN UP SUBROUTINE
*      USED TO REPLACE ID FRAME,JOBNO,CAMRAV IN MIDDLE RUNS
        ENTRY    CAMRAV
        ENTRY    FRAMEV
        ENTRY    JOBNO
        ENTRY    IDINFO
CAMRAV  SXA      (X4),4
        CALL     IDFM
        CAL      =0430000000000
        CALL     (PLOT)
 (X4)   AXT      **,4
        TTR      2,4
FRAMEV  SXA      X4,4
        CAL      =0460000000000
        CALL     (PLOT)
        CALL     (CLEAN
 X4     AXT      **,4
        TTR      2,4
 JOBNO  TTR      2,4
IDINFO  TTR      11,4
        END
 ***   FAP
                           APLT0000
        COUNT    150
*       ARRAY PLOTTING FOR X,Y POINTS
*      CALL APLOTV(N,X,Y,IX,IY,NC,CHAR,IERR)
        ENTRY    APLOTV
APLOTV  SXA      X4,4                  SAVE X REG
        SXA      X2,2                  X
        SXA      X1,1                  X
        TRA      A1                    TRA TO START
 EXIT   TSX      $SERREV,4             RESTORE ERROR
        TSX      SX                    CELLS
        TSX      SY
 X1     AXT      **,1                  RESTORE X REG
 X2     AXT      **,2                  X
 X4     AXT      **,4                  X
        TRA      9,4                   RETURN
 A1     CLA      2,4                   STORE BASE
        STA      AX                    ADDRESSES OF
        CLA      3,4                   X,Y,CHAR
        STA      AY                    ARRAYS
        CLA      7,4                   X
        STA      ACHAR                 X
        CLA      8,4                   SAVE IERR CELL
        STA      ERR                   ZERO IERR CELL
        STZ*     ERR                   X
        TSX      $SERSAV,4             SAVE PREVIOUS
        TSX      SX                    ERROR CELLS
        TSX      SY                    X
        TSX      $SCERRV,4             SET NEW ERROR
        TSX      XERR                  CELLS
        TSX      YERR                  X
        LXA      X4,4
        CLA*     4,4                   SAVE IX, IY
        ALS      18
        STD      IX                    X
        CLA*     5,4                   X
        ALS      18
        STD      IY                    X
        CLA*     1,4                   PICK UP N
        TZE      EXIT                  OUT IF ZERO
        TMI      A3                    FORWARD ARRAY
        ALS      18
        LXD      IX,1                  POSITIVE VALUES
        LXD      IY,2                  FOR IX, IY
        SUB      1B17                  N-1
        TRA      A2
 A3     ALS      18
        PDC      0,1                   COMPL. N
        PXD      0,1                   X
        LDC      IX,1                  COMPL. VALUES
        LDC     IY,2             FOR IX, IY
        ACL     M1               MAKE TXL
A2      STD     A15              STORE TEST DECR
        STD     A16              X
        ACL     M2               FILL IN OP
        STP     A15              STORE OPS
        STP     A16              X
        SXD     A11,1            STORE INCR
        SXD     A12,2            VALUES
        TXH     A4,1,0           TEST FOR INCR BOTH
        TXH     A4,2,0           ZERO
        TPL     A5               YES, TEST OP
        AXC     1,1              AND MAKE TEST
        TRA     A6               DECR 1 OR -1
A5      AXT     1,1              X
A6      SXD     A15,1            X
A4      CLA*    6,4              PICK UP NC
        ALS     18
        TNZ     A7               IF ZERO MAKE
        CLA     1B17             IT 1
A7      TPL     A8               BACKWARD ARRAY
        PDC     0,1              COMP OF NC
        PXD     0,1              X
        AXC     1,1              -1 FOR INCR
        TRA     A9               X
A8      SUB     1B17             NC -1
        AXT     1,1              +1 FOR INCR
        ACL     M1               MAKE OP TXL
A9      ACL     M2               SET IN OP
        STD     A14              STORE TEST DECR
        SXD     A10,1            STORE INCR VALUE
        STP     A14              STORE OP
        AXT     0,1              ZERO ALL XR
        AXT     0,2              X
        AXT     0,4              X
A13     STZ     P                ZERO PLOT COMMAND
AX      CLA     **,1             GET X
        STO     V
        SXA     A17,4
        TSX     $NXV,4           SCALE X
        TSX     V
        ANA     M3               X
        ORS     P                STORE IN PLOT COM
AY      CLA     **,2             GET Y
        STO     V
        TSX     $NYV,4           SCALE Y
        TSX     V
        LXA     A17,4
        CHS                      X
        ADD     M3
        ANA     M3               X
        ARS     18                      STORE IN PLOT
        ORS     P                       COMMAND
        CAL     XERR                    ACC SCALING
        STZ     XERR
        ORA     YERR                    ERRORS
        STZ     YERR
        TZE     ACHAR                   X
        ADD*    ERR
        STD*    ERR
        TRA     A10
ACHAR   CLA     **,4                    GET PLOTTING
        ALS     18
        STA     T                       CHARACTER AND
        STP     T                       STORE IN PLOT
        ZET     T                       COMMAND
        ARS     12                      X
        ANA     M4                      X
        ARS     6                       X
        ORS     P                       X
        CLA     P                       X
        TSX     $(PLOT),4               GO TO BUFFER
        LXA     A17,4
 A10    TXI     *+1,4,**                INCR CHAR
 A14    TXL     A11,4,**                OK
        AXT     0,4                     CYCLE CHAR
 A11    TXI     *+1,1,**                INCR X
 A15    TXH     EXIT,1,**               TEST FOR END
 A12    TXI     *+1,2,**                INCR Y
 A16    TXH     EXIT,2,**               TEST FOR END
        TRA     A13                     BACK FOR NEXT PT.
*               CONSTANTS AND TEMPORARY CELLS
 ERR    HTR     **                      L(IERR)
 SX     PZE     0,0,**                  PREVIOUS ERROR
 SY     PZE     0,0,**                  CELLS
 XERR   PZE     0,0,**                  NEW ERROR
 YERR   PZE     0,0,**                  CELLS
 IX     PZE     0,0,**                  IX
 IY     PZE     0,0,**                  IY
 1B17   PZE     0,0,1                   ONE IN DECR
 M1     OCT     400000000000            MASKS FOR
 M2     OCT     300000000000            TXH, TXL
 P      HTR     **                      PLOT WORD
 M3     OCT     001777000000            MASK FOR NX, NY
 T      HTR     **                      TEMP FOR CHAR
 M4     OCT     000077000000            MASK FOR CHAR
 A17    HTR     **
 V      HTR
        END
 ***    FAP
        COUNT   40
        ENTRY   PLOTV
*  MAIN CRT POINT PLOTTING ROUTINE
        SPACE
*       CALL    PLOTV(NX.NY.K)
*   PLOTS CHARACTER NO. (K) AT (NX.NY)
*   K MAY BE DECIMAL INT OR BOOLEAN VARIABLE
*   IF ADDRESS FIELD OF (K) IS NON-ZERO.
*   K IS ASSUMED TO BE OF THE FORM (1HK)
*   AND IS SHIFTED ACCORDINGLY.
*
        EJECT
 PLOTV  SXA     X4.4
        CAL*    3.4                     CHAR
        STA     T                       TEST FOR BCD FORMAT
        ZET     T
        ARS     12                      YES. POSITION CHARACTER
        ANA     =63B17                  REMOVE BLANKS ETC.
        ARS     6                       POSITION IN CRT WORD
        SLW     =HERE                   CHARACTER CODE
        CLS*    2.4
        ADD     =01777001777            CORRECT SIGN CONVENTION
        ARS     18
        ACL*    1.4
        ANA     =01777001777
        ORA     =HERE
        CALL    (PLOT)
 X4     AXT     **.4
        TRA     4.4
 T
        END
```

```
***    FAP
       COUNT   100
       ENTRY   POINTV
       REM     CALL POINTV (X•Y•NS)
       REM     OFF SCALE POINTS WILL NOT BE PLOTTED
       REM     SCERRV SUBROUTING WILL CONTAIN ERROR
       REM     INDICATORS
       REM     NS NEGATIVE MEANS NO CENTER DOT
       REM     INTEGER•NS•MAY BE 0•1•••••21
       REM     OR   CALL POINTV (NX•NY•NS•1)
POINTV SXA     ENDPT+1•2
       SXA     ENDPT+2•1
       CLA*    3•4
       PDX     0•1
       STO     NS
       CLA*    1•4
       STO     IX
       CAL*    2•4
       SLW     IY
       CAL     4•4
       ANA     =0777777700000
       SUB     =0007400000000
       TNZ     *+2
       TXI     *+1•4•-1                HAS 4 TH  ARGUMENT
       SXA     ENDPT•4
       TZE     P2
P1     TSX     $NXV•4
       TSX     IX
       STO     IX
       TZE     ENDPT
       TSX     $NYV•4
       TSX     IY
       STO     IY
       TZE     ENDPT
       REM     COULD• IN FUTURE• PROVIDE FOR VECTOR SYMBOLS
P2     TIX     P2•1•1•100
       TXL     P3•1•99
       AXT     0•1
P2•1   SXD     NS•1
       TSX     $PLOTV•4            POSITION BEAM TO
       TSX     IX                  * AVOID POSITION
       TSX     IY                  *ERROR
       TSX     BLANK
P3     TXL     P4•1•0
       TXL     P4•1•0
       TIX     *•1•48
       CLA     NS
       TMI     P5
       TXL     P4•1•14
       TXL     P5•1•24
P4     TSX     $PLOTV•4
```

```
        TSX     IX
        TSX     IY
        TSX     DOT
        TXL     ENDPT,1,0
P5      CLA     ADDR1                   LOL TABL1
P6      STA     P8
        SXA     P10,1
        AXT     0,2
P7      TNX     P8,1,6
        TXI     P7,2,1
P8      LDQ     **,2                    TABL 1 OR TABL2
P9      PXD     0
        LGL     6
        TIX     P9,1,1
        PAX     0,2
        SXD     SYMB,2
        TSX     $PLOTV,4
        TSX     IX
        TSX     IY
        TSX     SYMB
P10     AXT     **,1
        CLA     ADDR2                   LOC TABL2
        TXH     ENDPT,1,24
        TIX     P6,1,14
ENDPT   AXT     **,4
        AXT     **,2
        AXT     **,1
        TRA     4,4
ADDR1   PZE     TABL1
ADDR2   PZE     TABL2
NS      PZE     0
BLANK   PZE     0,0,48
DOT     PZE     0,0,42
        OCT     404061406060            32,32,49,32
TABL2   OCT     676720206161            55,55,16,16,49,49
        OCT     515502030710            41,45,2,3,7,8
        OCT     212225264447            17,18,21,22,36,39
        OCT     563445631324            46,28,37,51,11,20
        OCT     312762665074            25,23,50,54,40,60
        OCT     467761000067            38,63,49,0,0,55
        OCT     757146774677            61,57,38,63,38,63
        OCT     436400302365            35,52,0,24,19,53
TABL1   OCT     466777702054            38,55,63,56,16,44
IX      PZE     0
IY      PZE     0
SYMB    PZE     0
        END
        END
```

```
***     FAP
                        PRNT0000
        COUNT   50
*
*       SCATRAN TO FORTRAN CONVERTER FOR PRINTV
        ENTRY   PRINTS
        ENTRY   PRINT6
PRINTS  SXA     X4A,4
        CLA*    1,4
        ALS     18
        STO     A1
        CLA     2,4
        STA     BCD
        TSX     $PRINTV,4
        TSX     A1
 BCD    TSX     **
 X4A    AXT     **,4
        TTR     3,4
PRINT6  SXA     X4B,4
        CLA*    1,4
        ALS     18
        STO     A1
        CLA     2,4
        STA     BCD1
        CLA*    3,4
        STO     A2
        CLA*    4,4
        STO     A3
        TSX     $NXV,4
        TSX     A2
        STO     A2
        TSX     $NYV,4
        TSX     A3
        STO     A3
        TSX     $PRINTV,4
        TSX     A1
 BCD1   TSX     **
        TSX     A2
        TSX     A3
 X4B    AXT     **,4
        TTR     5,4
 A1     PZE
 A2     PZE
 A3     PZE
        EJECT
        END
```

```
*       FAP
        COUNT    150
        LBL      4PRTV000,L
        ENTRY    PRINTV
*   CRT TYPING ROUTINE
*       CALL     PRINTV(N,BCD)
*    OR CALL     PRINTV(N,BCD,NX,NY)
*           N   =NO. OF CHARACTERS TO PRINT
*           BCD =NAME OF BCD ARRAY
*                OR HOLLERITH ARG IF N IS NEGATIVE
*           NX  =ORDINATES IF SPECIFIED
*           NY   CURRENT POINT IF OMITTED
*    STOP TYPE WILL BE INSERTED AFTER FINAL CHARACTER.
*************************************************************
                 ************
        SPACE    2
PRINTV  SXA      X4,4
        SXA      X2,2
        SXA      X1,1
        CLA*     1,4                    NO. OF CHARACTERS
        LRS      53
        DVH     =6                      REDUCE MOD 6
        PAX      ,2                     NO. CHARS IN LAST WORD
        XCA                             NO. WORDS BEFORE LAST
        AXT      1,1                    SET UP MAIN LOOP
        TPL      *+3
        AXC      1,1
        COM
        ORA     =3B20
        LGR      18
        SLQ      L2                     TXH,,N OR TXL,,-N-1
        SXD      L1,1                   + OR-1
        AXT      0,1                    SET FOR 1ST WORD
        CLA      2,4
        STA      BCD
        SPACE    1
        CAL      3,4
        ERA      0,4                    TEST FOR SPECIFIED POINT
        ANA      OP
        TNZ      T1                     NO NX,NY GIVEN
        TXI      *+1,4,-2               NX,NY.
        SXA      X4,4
        CLS*     2,4                    NY
        ADD     =01777001777            CORRECT SIGN CONVENTION
        ARS      18
        ACL*     1,4
        ANA     =01777001777
        SLW      LOC
*   PREPARE 1ST CHARACTER
        CAL*     BCD
        LGR      30
        STQ     LOC2                      SAVE CHARS 2-6
        ALS     12
        ORA     TSP
        ORA     LOC                       TSP-X-CHAR-Y
        CALL    (PLOT)
        CAL     STOP
        TRA     T5
        SPACE   1
T1      BSS                               CURRENT POINT
        CAL*    BCD
        LGR     30
        STQ     LOC2
        ALS     24
        ORA     TCPST                     TCP-CHAR-STOP
T5      CALL    (PLOT)
        CAL     LOC2                      CHARS 2,3,---,N
        ARS     6
        ORA     TCP
        LXD     L2,4
        TXL     T12,4,0                   ONLY ONE WORD
        TXH     T12,4,-2
        TRA     L1                        TCP-CH-CH-CH-CH-CH
        SPACE   2
BCD     CAL     **,1
L1      TXI     *+1,1,**                  +1 OR-1
L2      TXH     T,2,1,**                  TXH,,N OR TXL,,-N-1
        CALL    (PLOT)
        TRA     BCD
*
T12     TXL     X4,2,1                    ONLY ONE CHARACTER
T,2     LDQ     STOP                      LAST WORD
        XEC     RSH,2
T,21    XEC     LSH,2                     SHIFT IN STOP TYPE CODE
        CALL    (PLOT)
*
X4      AXT     **,4
X2      AXT     **,2
X1      AXT     **,1
        TRA     3,4
STOP    OCT     120000000000
TSP     OCT     200000000000
TCP60   OCT     221200000000
TCPST   OCT     220012000000
TCP     OCT     220000000000
LOC
LOC2
OP      TXL     ,,-1
        SPACE   1
* TABLES OF SHIFTS
        ARS     6
        ARS     12
        ARS     18
        ARS     24
        ARS     30
 RSH    NOP
*
        LGL     6
        LGL     12
        LGL     18
        LGL     24
        LGL     30
 LSH    CAL     STOP
        END
```

```
***     FAP
                          PRVT0000
        COUNT    100
*
*       SCATRAN TO FORTRAN CONVERTER SUBROUTINE FOR PRINTING
        ENTRY    RITSTS
        ENTRY    RITE2S
        ENTRY    CHSIZS
RITSTS  SXA      X4,4
        CLA*     1,4
        ALS      18
        STO      A1
        CLA*     2,4
        ALS      18
        STO      A2
        CLA      3,4
        STA      A15
        TSX      $RITSTV,4
        TSX      A1
        TSX      A2
 A15    TSX      **
 X4     AXT      **,4
        TTR      4,4
RITE2S  SXA      X4A,4
        SXA      X1A,1
        CLA      8,4
        STA      WORDS
        CLA*     1,4
        STO      A1
        CLA*     2,4
        STO      A2
        AXT      5,1
 B1     CLA*     3,4
        ALS      18
        STO      A7+1,1
        TNX      *+2,1,1
        TXI      B1,4,-1
        TSX      $NXV,4
        TSX      A1
        STO      A1
        TSX      $NYV,4
        TSX      A2
        STO      A2
        TSX      $RITE2V,4
        TSX      A1
        TSX      A2
        TSX      A3
        TSX      A4
        TSX      A5
        TSX      A6
        TSX      A7
```

```
WORDS  TSX     **
       TSX     LAST
       ARS     18
X4A    AXT     **,4
X1A    AXT     **,1
       STO*    9,4
       TTR     10,4
CHSIZS SXA     X4B,4
       CLA*    1,4
       ALS     18
       STO     A1
       CLA*    2,4
       ALS     18
       STO     A2
       TSX     $CHSIZV,4
       TSX     A1
       TSX     A2
X4B    AXT     **,4
       TTR     3,4
A1     PZE
A2     PZE
A3     PZE
A4     PZE
A5     PZE
A6     PZE
A7     PZE
LAST   PZE
       EJECT
       END
```

```
***     FAP
        COUNT    300
*       SUBROUTINE RITE2V
        ENTRY    RITE2V
        ENTRY    RITSTV
        ENTRY    RITXYV
        REM      GENERAL WRITE USING VCHARV
        REM      CALL RITE2V(IX,IY,LIMIT,K,INT,NTOTAL,
        REM      NTH, WORDS, NLAST)
        REM      NTH SHOULD BE MINUS IF WORDS HOLLERITH
        REM      ********
        REM      MUST DEFINE TABL-V WITH F CARD
        REM      CALL RITSTV(ISPACE,IROW,TABL-V)
        REM      OR CALL RITSTV(ISPACE,IROW,A(1),0)
        REM      RITSTV IS A SET UP ROUTINE.
        REM      IF RITSTV IS NOT CALLED, STANDARD WILL BE USE
                 D
        REM      CALL RITXYV(IX,IY)
        REM      THIS RETRIEVES THE LAST USED IX,IY
RITSTV  CLA*     1,4
        STO      ISPACE
        CLA*     2,4
        STO      IROW
        CLA      3,4
        STA      V6
        CAL      4,4
        SLW      V7
        ANA      MASK2                 LOC777777700000
        SUB      MASK3                 LOC TSX 0,0,0
        TZE      5,4
        CLA      NOP                   LOC NOP
        STO      V7
        TRA      4,4
RITXYV  SXA      SAV2,2
        LXD      KPRIME,2
        CLS      IX
        XEC      SBHH,2
        STD*     1,4
        CLS      IY
        XEC      ADHW,2
        STD*     2,4
 SAV2   AXT      **,2
        TRA      3,4
RITE2V  SXA      IR4,4
        SXA      IR2,2
        SXA      IR1,1
        CLA      8,4
        STA      WORDS
        AXT      7,1
 R10    CLA*     1,4
        STO      NTH+1,1
        TNX     *+2,1,1
        TXI     R10,4,-1
        CLA     K
        ANA     MASK1                   LOC 000007000000
        ARS     1
        PDX     0,2                     0,1,2,3
        STD     KPRIME
        TSX     SRTSIZV,4
        TSX     INCRW
        TSX     INCRH
        AXT     2,1
R12     CLA     INCRH+1,1               INCRW,INCRH
        ALS     1
        TXH     *+2,1,1                    NOP,ADDINCRH
        ADD     INCRH
        STO     HALFH+1,1               HALFW,HALFH
        ALS     1
        STO     HIGH+1,1                WIDE, HIGH
        ADD     L4
        CAS     IROW+1,1                ISPACE,IROW
        STO     IROW+1,1
        NOP
        TIX     R12,1,1
        AXT     2,1
R13     CLA     IY+1,1
        XEC     SBHH,2                  SUBHALFH,SUBHW,ADDHH,ADHW
        STO     IY+1,1
        TMI     ERROR
        LDQ     L1023
        TLQ     ERROR
        XEC     ADH,2
        TMI     ERROR
        LDQ     L1023
        TLQ     ERROR
        TNX     *+2,1,1
        TXI     R13,2,-1
        TXI     *+1,2,1
        CLA*    IYS,2                   IY,IX,IY,IX
        STO     ISTART
        LXD     NTOTAL,1                NTOTAL
        LDQ     SBW,2                   SUB,ADD,ADD,SUB
        SLQ     R40
        SLQ     R50
        LDQ     SBHH,2                  SUB,SUB,ADD,ADD
        SLQ     R70                     FOR OPERATION
R30     TSX     $HOLLV,4
        TSX     NTH
WORDS   TSX     **
        TSX     L
        TSX     $VCHARV,4
        TSX     K
```

```
V2      TSX      INT
V3      TSX      IX
V4      TSX      IY
        TSX      L
V6      TSX      **,0
V7      NOP      **
        CAL      NTH
        ACL      L1
        SLW      NTH
        CLA*     IYS,2                    IY,IX,IY,IX
R40     SUB      ISPACE                   SUB,ADD,ADD,SUB
        STO*     IYS,2                    IY,IX,IY,IX
R50     SUB      WIDE                     SUB,ADD,ADD,SUB
        SUB      LIMIT
        TZE      R80
        TXL      TPL,2,0
        TXH      TPL,2,2
        TPL      NEWRW
        TMI      R80
TPL     TPL      R80
NEWRW   CLA      ISTART
        STO*     IYS,2                    IY,IX,IY,IX
        CLA*     IXS,2                    IX,IY,IX,IY
R70     SUB      IROW                     SUB,SUB,ADD,ADD
        STO*     IXS,2                    IX,IY,IX,IY
        TMI      ERROR
        LDQ      L1023
        TLQ      ERROR
R80     TIX      R30,1,1                  BACK FOR NEXT CHAR
OUTOK   PXD      0                        CLEAR
IR4     AXT      **,4
IR2     AXT      **,2
IR1     AXT      **,1
        STO*     9,4
        TRA      10,4
ERROR   CLA      NTH
        TRA      IR4
        ADM      HALFW
        ADM      HALFH
        SBM      HALFW
SBHH    SBM      HALFH
ADHW    ADM      HALFW
        SBM      WIDE
        SBM      HIGH
        ADM      WIDE
ADH     ADM      HIGH
SBW     SBM      WIDE
        PZE      IY
        PZE      IX
        PZE      IY
IXS     PZE      IX
```

```
 IYS    PZE      IY
 HALFW  PZE      0
 HALFH  PZE      0
 INCRW  PZE      0
 INCRH  PZE      0
 WIDE   EQU      INCRW
 HIGH   EQU      INCRH
ISTART  PZE      0
 L      PZE      0
 IX     PZE      0
 IY     PZE      0
 LIMIT  PZE      0
 K      PZE      0
 INT    PZE      0
NTOTAL  PZE      0
 NTH    PZE      0
 L1     PZE      0,0,1
 L4     PZE      0,0,4
 L1023  PZE      0,0,1023
ISPACE  PZE      0,0,18
 IROW   PZE      0,0,24
 MASK1  OCT      000007000000
KPRIME  PZE      0
 MASK2  OCT      777777700000
 MASK3  TSX      0,0,0
 NOP    NOP      0
        END
```

```
***     FAP
        COUNT    800
        ENTRY    CHSIZV
        ENTRY    VCHARV
        ENTRY    RTSIZV
        REM      NO TESTS MADE FOR INSUFFICIENT SPACE
        REM      STANDARD SIZE. IF CHSIZV NEVER ENTERED
CHSIZV  CLA*     1.4                  IX MULTIPLE
        STO      IWM
        CLA*     2.4                  IY MULTIPLE
        STO      IHM
        TRA      3.4
RTSIZV  CLA      IWM
        STO*     1.4
        CLA      IHM
        STO*     2.4
        TRA      3.4
        REM      CALL VCHARV(IDEG.INT.IX.IY.NCODE.TABLE)
        REM      OR CALL VCHARV(IDEG.INT.IX.IY.0.A(1).0)
VCHARV  SXA      OUT2.2
        SXA      OUT3.1
        CLA*     3.4
        STD      ROOT
        CLA*     4.4
        STO      ROOT2
        CLA*     1.4                  0.90.180.270.360
        ANA      MASK1                LOC000006000000
        ARS      1                    0.1.2.3.0
        STD      SWICH
        CLA*     2.4
        ARS      18
        STA      VC046
        CLA*     5.4
        PDX      0.2                  CHARACTER SELECT
        CAL      7.4                  ARE THERE 7 ARGS.
        ANA      MASK2                LOC 777777700000
        SUB      MASK3                LOC TSX 0.0.0
        TNZ      VC006
        CLA      6.4
        STA      VC020
        CLA      LZERO
        TXI      VC007.4.-1
 VC006  CLA*     6.4
        STA      VC020
        ADD      ONEA
        STA      *+1
        CLA      **
 VC007  STA      VC015
        SXA      OUT1.4
 VC008  AXT      0.1
        TXI      VC010.2.1            GET STARTED AT NON-ZERO
```

```
VC010 TNX       VC015,2,4
      TXI       VC010,1,1
VC015 LDQ       **,1                   ENTRY
      XEC       RQLS,2
      PXD       0
      LGL       9
      PAX       0,2                    MOVE THROUGH TABLE
VC019 AXT       3,4
VC020 LDQ       **,2                   TABLE
      SXA       SAV2,2
      LXD       SWICH,2
VC021 PXD       0
      LGL       12
      TZE       OUT1
      LGR       9
VC022 LXD       IWM,1
      STZ       WORD
      ALS       15
      LGL       3
      STO       ADDR
      TNX       VC032,1,1
VC030 ADD       ADDR
      TIX       VC030,1,1
VC032 STD       XTEMP
      ALS       21
      ARS       3
      SUB       XTEMP                  X2-X1
      XEC       MOVE1,2                ARS8,ALS10,ARS8,ALS10
      XEC       TMIPL,2
      XEC       MOVE5,2                BIT 19,18,19,18
VC035 ORS       WORD
VC036 LXD       IHM,1
      PXD       0
      LGL       3
      ALS       15
      LGL       3
      STO       ADDR
      TNX       VC042,1,1
VC040 ADD       ADDR
      TIX       VC040,1,1
VC042 STD       YTEMP
      ALS       21
      ARS       3
      SUB       YTEMP                  Y2-Y1
      XEC       MOVE3,2                ALS10, ARS8, ALS10, ARS8
      XEC       MOVE2,2                TMI,TMI,TPL,TPL
      XEC       MOVE4,2                BIT 18,19,18,19
VC045 ORS       WORD
      XEC       MOVE6,2                CLA XTEMP,CLS YTEMP
      REM                              CLSXTEMP,CLA YTEMP
      SUB       ROOT2
```

```
        ADD     L1023
        ARS     18
        ORS     WORD
        CAL     ROOT
        XEC     MOVE7,2             ADD YTEMP,XTEMP,SUB YTEMP
                ,XTEMP
        ACL     WORD
        STQ     SAVQ
        SXA     SAV4,4
        SXA     SAV1,1
VC046   AXT     **,1                INT FACTOR
        PAI
        PIA
        TSX     $(PLOT),4
        TIX     *-2,1,1
SAV1    AXT     **,1                RESTORE
SAV4    AXT     **,4                RESTORE
        LDQ     SAVQ                RESTORE
        TIX     VC021,4,1
SAV2    AXT     **,2
        TXI     VC019,2,1
OUT1    AXT     **,4
OUT2    AXT     **,2
OUT3    AXT     **,1
        TRA     7,4
ROOT    OCT     600000000000
ROOT2   PZE     0
WORD    PZE     0
ADDR    PZE     0
YTEMP   PZE     0
XTEMP   PZE     0
SAVQ    PZE     0                   SAVE QUOT.
BIT18   PZE     0,4,0
BIT19   PZE     0,2,0
L1023   PZE     0,0,1023
ONED    PZE     0,0,1
ONEA    PZE     1
        TRA     VC010               SHOULDNT GET HERE
        RQL     27
        RQL     18
        RQL     9
        NOP
RQLS    NOP
IHM     PZE     0,0,3               STANDARD
IWM     PZE     0,0,3
SWICH   PZE     0                   0 OR 1
        SUB     XTEMP
        SUB     YTEMP
        ADD     XTEMP
MOVE7   ADD     YTEMP
        ALS     10
        ARS     8
        ALS     10
MOVE1   ARS     8
MOVE3   ALS     10
        TPL     VC045
        TPL     VC045
        TMI     VC045
MOVE2   TMI     VC045
        ORA     BIT19
        ORA     BIT18
        ORA     BIT19
MOVE4   ORA     BIT18
MOVE5   ORA     BIT19
        CLA     YTEMP
        CLS     XTEMP
        CLS     YTEMP
MOVE6   CLA     XTEMP
        TPL     VC035
        TMI     VC035
        TMI     VC035
TMIPL   TPL     VC035
ZERO    PZE     0
LZERO   PZE     ZERO
MASK1   OCT     000006000000
MASK2   OCT     777777700000
MASK3   TSX     0,0,0
        END
```

```
***     FAP
        COUNT    100
        ENTRY    HOLLV
        REM      CALL HOLLV (IJ, WORDS, L)
        REM      +J FORTRAN (DESCENDING) WORDS BLOCK
        REM      -J ASCENDING WORDS BLOCK
        REM      J NO. OF THE CHARACT.
        REM      L. LOCATION IN WHICH CHAR IS STORED
HOLLV   SXA      H5,4
        SXA      H6,2
        SXA      H7,1
        CLA*     1,4
        PDX      0,1                        NO OF THE CHAR
        AXT      1,2                        START WORD COUNTER
        TPL      H1
        AXC      1,2
H1      SXD      H3,2
        CLA      2,4
        STA      H4
        AXT      0,4
H2      TNX      H4,1,6
H3      TXI      H2,4,**                    INCREASE OR DECREASE
H4      LDQ      **,4
        PXD      0
        XEC      RQL+6,1
        LGL      6
        ALS      18
H5      AXT      **,4
H6      AXT      **,2
H7      AXT      **,1
        STO*     3,4
        TRA      4,4
RQL     RQL      30
        RQL      24
        RQL      18
        RQL      12
        RQL      6
        NCP
        NOP
        END
```

```
 ***    FAP
        COUNT    220
*       TABLE FOR ENGLISH CHARACTERS
        ENTRY    TABL1V
        REM      TABLE 1 TO BE USED WITH VCHARV
        REM      MIGHT ADD FURTHER ENTRIES
        PZE      0
        VFD      9/180,9/181,9/183,9/185
        VFD      9/171,9/173,9/176,9/179
        VFD      9/159,9/162,9/166,9/169
        VFD      9/151,9/153,9/155,9/157
        VFD      9/142,9/144,9/146,9/149
        VFD      9/136,9/138,9/139,9/141
        VFD      9/129,9/130,9/131,9/135
        VFD      9/119,9/121,9/124,9/126
        VFD      9/105,9/109,9/112,9/115
        VFD      9/94,9/96,9/98,9/102
        VFD      9/87,9/88,9/91,9/93
        VFD      9/76,9/78,9/82,9/84
        VFD      9/68,9/70,9/72,9/74
        VFD      9/57,9/60,9/62,9/64
        VFD      9/47,9/48,9/50,9/54
        VFD      9/38,9/39,9/40,9/44
        VFD      9/23,9/29,9/33,9/37
        VFD      9/12,9/14,9/17,9/21  4,5,6,7
ENTR1V  VFD      9/0,9/3,9/5,9/8      0,1,2,3
        REM      MIGHT ADD MORE CHARACTERS
        VFD      3/4,3/3,3/3,3/2,3/3,3/2,3/2,3/3,12/0
        VFD      3/1,3/0,3/2,3/3,3/2,3/3,3/3,3/4,3/3,3/4,3/4,3
                 /3
 INFIN  VFD      3/0,3/1,3/3,3/4,3/1,3/2,3/4,3/3,3/2,3/1,3/3,3
                 /2
        PZE      0
LESEQU  VFD      3/0,3/4,3/4,3/2,3/0,3/4,3/4,3/6,3/0,3/4,6/0
        PZE      0
GRTEQU  VFD      3/0,3/4,3/2,3/4,3/0,3/4,3/6,3/4,3/0,3/4,6/0
 LESS   VFD      3/0,3/4,3/3,3/1,3/0,3/4,3/3,3/5,12/0
 GRATR  VFD      3/0,3/4,3/1,3/3,3/0,3/4,3/5,3/3,12/0
        VFD      3/4,3/3,3/3,3/2,24/0
        VFD      3/3,3/2,3/5,3/6,3/2,3/1,3/6,3/6,3/3,3/4,3/4,3
                 /3
BRACEC  VFD      3/1,3/2,6/0,3/2,3/3,3/0,3/1,3/3,3/3,3/1,3/5
        VFD      3/0,3/1,3/3,3/2,24/0
        VFD      3/1,3/2,3/5,3/6,3/2,3/3,3/6,3/6,3/1,3/0,3/4,3
                 /3
3RACEO  VFD      3/3,3/2,6/0,3/2,3/1,3/0,3/1,3/1,3/1,3/1,3/5
        PZE      0
BRAKTC  VFD      3/2,3/4,6/0,3/4,3/4,3/0,3/6,3/4,3/2,3/6,3/6
        PZE      0
BRAKTO  VFD      3/2,3/0,6/0,6/0,3/0,3/6,3/0,3/2,3/6,3/6
        VFD      3/1,3/4,6/0,24/0
        VFD     3/0,3/1,3/3,3/2,3/1,3/3,3/2,3/2,3/3,3/1,3/7,3
                /0
 CLV    VFD     3/3,3/1,3/5,3/5,3/1,3/0,3/5,3/4,6/0,3/4,3/3
        VFD     3/3,3/2,3/0,3/1,24/0
        VFD     3/2,3/3,3/1,3/2,3/3,3/4,3/2,3/1,3/4,3/3,3/1,3
                /0
        VFD     3/1,3/0,3/4,3/5,3/1,3/4,3/6,3/6,3/4,3/0,3/6,3
                /0
PRCNTV  VFD     3/0,3/1,3/5,3/6,3/1,3/2,3/6,3/5,3/2,3/1,3/5,3
                /4
        VFD     3/1,3/1,3/5,3/4,3/1,3/4,3/4,3/0,12/0
        VFD     3/0,3/3,3/2,3/5,3/3,3/2,3/5,3/6,3/2,3/1,3/6,3
                /5
 ANDV   VFD     3/4,3/3,3/3,3/0,3/3,3/1,6/0,3/1,3/0,3/0,3/2
        VFD     3/4,3/0,3/1,3/1,24/0
 SQUAR  VFD     6/0,3/1,3/5,3/0,3/4,3/5,3/5,3/4,3/4,3/5,3/1
        VFD     3/0,3/3,6/0,24/0
 SUM    VFD     3/0,3/3,3/6,3/6,3/0,3/2,3/6,3/3,3/2,3/0,3/3,3
                /0
        VFD     3/2,3/1,3/1,3/0,3/1,3/0,6/0,12/0
 INTEG  VFD     3/4,3/3,3/6,3/6,3/3,3/2,3/6,3/5,3/2,3/2,3/5,3
                /1
        VFD     3/3,3/2,3/0,3/0,3/3,3/2,3/6,3/6,12/0
 PRENO  VFD     3/2,3/1,3/0,3/1,3/1,3/1,3/1,3/5,3/1,3/2,3/5,3
                /6
        VFD     3/2,3/2,3/2,3/1,3/2,3/1,3/1,3/0,12/0
 COMMA  VFD     3/2,3/1,3/2,3/2,3/1,3/1,3/2,3/1,3/1,3/2,3/1,3
                /1
        VFD     3/2,3/1,3/3,3/3,3/1,3/0,3/3,3/4,12/0
        VFD     3/2,3/3,3/6,3/5,3/3,3/3,3/5,3/4,3/3,3/2,3/4,3
                /3
 DGREE  VFD     6/0,3/4,3/5,3/0,3/1,3/5,3/6,3/1,3/2,3/6,3/6
        PZE     0
 Z      VFD     3/0,3/4,3/6,3/6,3/0,3/4,3/0,3/6,3/0,3/4,6/0
        PZE
 Y      VFD     3/0,3/2,3/6,3/3,3/2,3/2,3/0,3/3,3/2,3/4,3/3,3
                /6
 X      VFD     3/0,3/4,3/6,3/0,3/0,3/4,3/0,3/6,12/0
        VFD     3/3,3/4,3/0,3/6,24/0
 W      VFD     3/0,3/1,3/6,3/0,3/1,3/2,3/0,3/4,3/2,3/3,3/4,3
                /0
 V      VFD     3/0,3/2,3/6,3/0,3/2,3/4,3/0,3/6,12/0
        VFD     3/3,3/4,3/0,3/1,3/4,3/4,3/1,3/6,12/0
 U      VFD     6/0,3/6,3/1,3/0,3/1,3/1,3/0,3/1,3/3,6/0
 T      VFD     3/2,3/2,3/0,3/6,3/0,3/4,3/6,3/6,12/0
        VFD     3/3,3/1,6/0,3/1,3/0,3/0,3/1,12/0
        VFD     3/3,3/4,3/3,3/2,3/4,3/4,3/2,3/1,3/4,3/3,3/1,3
                /0
        VFD     6/0,3/5,3/4,3/0,3/1,3/4,3/3,3/1,3/3,3/3,3/3
 S      VFD     3/4,3/3,3/5,3/6,3/3,3/1,3/6,3/6,3/1,3/0,3/6,3
                /5
```

```
SLASH VFD       3/0,3/4,3/0,3/6,24/0
BLANK PZE       0
      VFD       3/3,3/3,3/1,3/5,24/0
      VFD       3/3,3/2,3/2,3/1,3/2,3/1,3/1,3/1,3/1,3/0,3/1,3
                /2
DERIV VFD       3/0,3/1,3/2,3/3,3/1,3/2,3/3,3/3,3/2,3/3,3/3,3
                /2
      PZE       0
TILDE VFD       3/0,3/1,3/3,3/4,3/1,3/3,3/4,3/2,3/3,3/4,3/2,3
                /3
      PZE       0
      VFD       3/2,3/1,3/0,3/1,3/1,3/1,3/1,3/2,3/1,3/4,3/2,3
                /6
GAMMA VFD       3/0,3/3,3/5,3/2,3/3,3/3,3/2,3/1,3/3,3/2,3/1,3
                /0
      PZE       0
ASTRK VFD       3/0,3/4,3/3,3/3,3/1,3/3,3/5,3/1,3/3,3/1,3/5,3
                /1
      VFD       3/2,3/2,3/0,3/7,3/1,3/1,3/0,3/7,12/0
      VFD       3/3,3/2,3/2,3/1,3/2,3/1,3/1,3/1,3/1,3/0,3/1,3
                /2
      VFD       6/0,3/5,3/4,3/0,3/3,3/4,3/3,3/3,3/3,3/3,3/2
DOLAR VFD       3/3,3/2,3/5,3/6,3/2,3/1,3/6,3/6,3/1,3/0,3/6,3
                /5
      PZE       0
      VFD       3/2,3/1,3/2,3/3,3/2,3/2,3/4,3/2,3/3,3/1,3/3,3
                /3
DOTT  VFD       3/1,3/2,3/3,3/4,3/2,3/3,3/4,3/3,3/3,3/2,3/3,3
                /2
      VFD       3/2,3/4,3/3,3/0,24/0
      VFD       3/4,3/4,3/5,3/4,3/4,3/3,3/4,3/3,3/3,3/0,3/3,3
                /3
R     VFD       6/0,3/0,3/6,3/0,3/3,3/6,3/6,3/3,3/4,3/6,3/5
      PZE       0
      VFD       3/3,3/1,6/0,3/1,3/0,3/0,3/1,3/2,3/4,3/2,3/0
      VFD       3/3,3/4,3/6,3/5,3/4,3/4,3/5,3/1,3/4,3/3,3/1,3
                /0
Q     VFD       6/0,3/1,3/5,3/0,3/1,3/5,3/6,3/1,3/3,3/6,3/6
      PZE       0
      VFD       3/4,3/4,3/5,3/4,3/4,3/3,3/4,3/3,3/3,3/0,3/3,3
                /3
P     VFD       6/0,3/0,3/6,3/0,3/3,3/6,3/6,3/3,3/4,3/6,3/5
      PZE       0
      VFD       3/3,3/1,6/0,3/1,3/0,3/0,3/1,3/4,3/2,3/6,3/4
      VFD       3/3,3/4,3/6,3/5,3/4,3/4,3/5,3/1,3/4,3/3,3/1,3
                /0
O     VFD       6/0,3/1,3/5,3/0,3/1,3/5,3/6,3/1,3/3,3/6,3/6
      PZE       0
N     VFD       6/0,3/0,3/6,3/0,3/4,3/6,3/0,3/4,3/4,3/0,3/6
      VFD       3/4,3/4,3/6,3/0,24/0
M     VFD       6/0,3/0,3/6,3/0,3/2,3/6,3/3,3/2,3/4,3/3,3/6
```

```
L       VFD     6/0,3/0,3/6,3/0,3/4,6/0,12/0
        PZE     0
K       VFD     6/0,3/0,3/6,3/0,3/4,3/3,3/6,3/2,3/4,3/4,3/0
        VFD     3/2,3/4,3/6,3/6,24/0
        VFD     3/1,3/2,6/0,3/2,3/3,3/0,3/1,3/3,3/3,3/1,3/6
J       VFD     3/0,3/1,3/2,3/2,6/0,3/2,3/1,3/0,3/1,3/1,3/0
MINUS   VFD     3/0,3/4,3/2,3/2,24/0
        VFD     3/2,3/2,3/1,3/0,3/2,3/2,3/1,3/0,12/0
        VFD     3/4,3/4,3/5,3/4,3/4,3/2,3/4,3/3,3/2,3/2,3/3,3
                /2
QUSTN   VFD     3/0,3/1,3/5,3/6,3/1,3/3,3/6,3/6,3/3,3/4,3/6,3
                /5
        PZE     0
PLMIN   VFD     3/0,3/4,3/4,3/4,3/2,3/2,3/2,3/6,3/4,3/0,6/0
        VFD     3/3,3/2,6/0,3/2,3/1,3/0,3/1,12/0
        VFD     3/3,3/4,3/3,3/2,3/4,3/4,3/2,3/1,3/4,3/3,3/1,3
                /0
        VFD     3/3,3/4,3/6,3/5,3/4,3/4,3/5,3/4,3/4,3/2,3/4,3
                /3
BETA    VFD     3/0,3/1,3/0,3/5,3/1,3/2,3/5,3/6,3/2,3/3,3/6,3
                /6
        VFD     3/1,3/2,3/0,3/0,3/1,3/2,3/6,3/6,12/0
PRENC   VFD     3/2,3/3,3/0,3/1,3/3,3/3,3/1,3/5,3/3,3/2,3/5,3
                /6
        VFD     3/2,3/1,6/0,24/0
PEROD   VFD     3/1,3/1,3/0,3/1,3/1,3/2,3/1,3/1,3/2,3/2,3/1,3
                /0
        VFD     3/1,3/3,3/3,3/3,3/3,3/4,3/3,3/4,12/0
PI      VFD     3/1,3/1,3/0,3/3,3/3,3/3,3/0,3/3,3/0,3/1,3/2,3
                /3
        PZE
I       VFD     3/1,3/3,6/0,3/2,3/2,3/0,3/6,3/1,3/3,3/6,3/6
        PZE     0
H       VFD     6/0,3/0,3/6,3/0,3/4,3/3,3/3,3/4,3/4,3/0,3/6
        PZE     0
        VFD     3/3,3/4,3/0,3/1,3/4,3/4,3/1,3/3,3/4,3/2,3/3,3
                /3
        VFD     6/0,3/5,3/1,3/0,3/1,3/1,3/0,3/1,3/3,6/0
G       VFD     3/4,3/3,3/5,3/6,3/3,3/1,3/6,3/6,3/1,3/0,3/6,3
                /5
        PZE     0
F       VFD     6/0,3/0,3/6,3/0,3/4,3/6,3/6,3/0,3/3,3/3,3/3
        VFD     3/0,3/3,3/3,3/3,24/0
E       VFD     3/4,3/0,6/0,6/0,3/0,3/6,3/0,3/4,3/6,3/6
        PZE     0
        VFD     3/4,3/4,3/5,3/1,3/4,3/3,3/1,3/0,3/3,3/0,6/0
D       VFD     6/0,3/0,3/6,3/0,3/3,3/6,3/6,3/3,3/4,3/6,3/5
        VFD     3/3,3/4,3/0,3/1,24/0
        VFD     6/0,3/5,3/1,3/0,3/1,3/1,3/0,3/1,3/3,6/0
C       VFD     3/4,3/3,3/5,3/6,3/3,3/1,3/6,3/6,3/1,3/0,3/6,3
                /5
        VFD    3/3,3/0,30/0
        VFD    3/3,3/4,3/3,3/2,3/4,3/4,3/2,3/1,3/4,3/3,3/1,3
               /0
        VFD    3/4,3/4,3/5,3/4,3/4,3/3,3/4,3/3,3/0,3/3,3/3,3
               /3
B       VFD    6/0,3/0,3/6,3/0,3/3,3/6,3/6,3/3,3/4,3/6,3/5
        PZE    0
A       VFD    3/0,3/2,3/0,3/6,3/2,3/4,3/6,3/0,3/1,3/3,3/3,3
               /3
PLUS    VFD    3/0,3/4,3/3,3/3,3/2,3/2,3/1,3/5,12/0
        VFD    3/2,3/4,3/4,3/0,24/0
        VFD    6/0,3/1,3/3,3/0,3/1,3/3,3/4,3/1,3/2,3/4,3/4
ALPHA   VFD    3/4,3/2,3/4,3/0,3/2,3/1,6/0,3/1,3/0,3/0,3/1
        VFD    3/1,3/1,3/1,3/2,3/1,3/2,3/2,3/3,12/0
        VFD    3/4,3/3,3/1,3/0,3/3,3/2,6/0,3/2,3/1,3/0,3/1
        VFD    3/1,3/3,3/4,3/3,3/3,3/4,3/3,3/2,3/4,3/4,3/2,3
               /1
DLTA    VFD    3/3,3/2,3/5,3/6,3/2,3/1,3/6,3/5,3/1,3/1,3/5,3
               /4
PRIME   VFD    3/2,3/2,3/5,3/6,24/0
QUOTS   VFD    3/1,3/1,3/4,3/6,3/3,3/3,3/4,3/6,12/0
EQLS    VFD    3/0,3/4,3/2,3/2,3/0,3/4,3/4,3/4,12/0
        VFD    3/1,3/2,3/3,3/3,3/2,3/3,3/3,3/2,12/0
        VFD    3/1,3/0,3/0,3/1,6/0,3/1,3/2,3/0,3/1,3/2,3/3
        VFD    3/4,3/3,3/5,3/1,3/3,3/2,3/1,3/0,3/2,3/1,6/0
PARTL   VFD    3/1,3/2,3/5,3/6,3/2,3/3,3/6,3/6,3/3,3/4,3/6,3
               /5
        VFD    3/1,3/3,3/3,3/3,3/3,3/4,3/3,3/4,12/0
        VFD    3/1,3/0,3/6,3/5,6/0,3/5,3/4,3/0,3/1,3/4,3/3
        VFD    3/4,3/4,3/1,3/5,3/4,3/3,3/5,3/6,3/3,3/1,3/6,3
               /6
NINEN   VFD    3/0,3/1,3/1,3/0,3/1,3/3,6/0,3/3,3/4,3/0,3/1
        PZE    0
        VFD    3/4,3/3,3/1,3/0,3/3,3/1,6/0,3/1,3/0,3/0,3/1
        VFD    3/3,3/1,3/3,3/3,3/3,3/4,3/3,3/2,3/4,3/4,3/2,3
               /1
        VFD    3/3,3/4,3/6,3/5,3/4,3/4,3/5,3/4,3/4,3/3,3/4,3
               /3
        VFD    6/0,3/4,3/5,3/0,3/1,3/5,3/6,3/1,3/3,3/6,3/6
EIGHT   VFD    6/0,3/1,3/2,3/0,3/1,3/2,3/3,3/1,3/0,3/3,3/4
        PZE
SEVNN   VFD    6/0,3/5,3/6,3/0,3/4,3/6,3/6,3/4,3/1,3/6,3/0
        VFD    3/3,3/1,6/0,3/1,3/0,3/0,3/1,12/0
        VFD    3/3,3/4,3/3,3/2,3/4,3/4,3/2,3/1,3/4,3/3,3/1,3
               /0
        VFD    3/3,3/4,3/6,3/5,3/0,3/1,3/2,3/3,3/1,3/3,3/3,3
               /3
SIXN    VFD    6/0,3/1,3/5,3/0,3/1,3/5,3/6,3/1,3/3,3/6,3/6
        VFD    6/0,3/4,3/6,3/0,3/4,3/6,3/6,12/0
        VFD    3/4,3/4,3/1,3/3,3/4,3/3,3/3,3/4,3/3,3/0,3/4,3
               /4
```

```
 FIVEN VFD      3/0,3/1,3/1,3/0,3/1,3/3,6/0,3/3,3/4,3/0,3/1
       VFD      3/0,3/4,3/2,3/2,24/0
 FOURN VFD      3/2,3/4,6/0,3/3,3/3,3/0,3/6,3/3,3/0,3/6,3/2
       VFD      3/3,3/1,3/6,3/6,3/1,3/0,3/6,3/5,12/0
       VFD      3/3,3/4,3/3,3/4,3/4,3/4,3/4,3/5,3/4,3/3,3/5,3
                /6
       VFD      3/4,3/4,3/1,3/2,3/4,3/3,3/2,3/3,3/3,3/2,3/3,3
                /3
 THREN VFD      3/0,3/1,3/1,3/0,3/1,3/3,6/0,3/3,3/4,3/0,3/1
       PZE      0
       VFD      3/4,3/4,3/5,3/4,3/4,3/0,3/4,3/0,3/0,3/4,6/0
 TWON  VFD      3/0,3/1,3/5,3/6,3/1,3/3,3/6,3/6,3/3,3/4,3/6,3
                /5
       PZE      0
 ONEN  VFD      3/1,3/3,6/0,3/2,3/2,3/0,3/6,3/1,3/2,3/5,3/6
       VFD      3/3,3/2,3/0,3/0,3/2,3/1,3/0,3/1,12/0
       VFD      3/3,3/4,3/6,3/5,3/4,3/4,3/5,3/1,3/4,3/3,3/1,3
                /0
TABL1V VFD      3/1,3/1,3/1,3/5,3/1,3/2,3/5,3/6,3/2,3/3,3/6,3
                /6
       PZE      ENTR1V
       END
```

```
***     FAP
        COUNT   150
        ENTRY   TABL2V
        REM     TABLE OF GREEK CHARACTERS
        PZE     0
        VFD     9/57,9/61,9/63,9/65
        VFD     9/47,9/50,9/53,9/55
        VFD     9/36,9/38,9/42,9/45
        VFD     9/27,9/29,9/31,9/33
        VFD     9/14,9/17,9/20,9/23
ENTR2V  VFD     9/0,9/3,9/7,9/10
        PZE     0
        VFD     3/4,3/4,3/1,3/3,3/4,3/3,3/3,3/4,12/0
        VFD     3/1,3/2,3/0,3/1,3/2,3/3,3/1,3/0,3/3,3/4,3/0,3
                /1
 OMEGA  VFD     3/1,3/0,3/4,3/3,6/0,3/3,3/1,3/0,3/1,3/1,3/0
        VFD     3/2,3/4,3/1,3/3,3/2,3/1,3/5,3/0,12/0
 PSI    VFD     6/0,3/3,3/2,3/0,3/1,3/2,3/1,3/1,3/2,3/1,3/1
        VFD     3/4,3/0,3/5,3/0,24/0
 CHI    VFD     3/0,3/1,3/5,3/5,3/1,3/3,3/5,3/1,3/3,3/4,3/1,3
                /0
        PZE     0
        VFD     3/3,3/1,3/1,3/1,3/1,3/0,3/1,3/2,3/2,3/1,3/6,3
                /0
        VFD     3/3,3/4,3/5,3/4,3/4,3/4,3/4,3/2,3/4,3/3,3/2,3
                /1
 PHI    VFD     6/0,3/2,3/4,3/0,3/1,3/4,3/5,3/1,3/3,3/5,3/5
        VFD     3/2,3/3,3/0,3/1,3/3,3/3,3/1,3/4,12/0
UPSILN  VFD     6/0,3/4,3/1,3/0,3/1,3/1,3/0,3/1,3/2,6/0
        VFD     3/3,3/4,3/3,3/4,24/0
 TAU    VFD     3/2,3/1,3/3,3/0,3/0,3/1,3/2,3/3,3/1,3/3,3/3,3
                /3
        VFD     3/3,3/3,3/1,3/3,3/3,3/2,3/3,3/4,12/0
        VFD     3/0,3/1,3/1,3/0,3/1,3/2,6/0,3/2,3/3,3/0,3/1
 SIGMA  VFD     3/4,3/1,3/4,3/4,3/1,3/0,3/4,3/3,6/0,3/3,3/1
        VFD     3/3,3/2,3/2,3/2,3/2,3/1,3/2,3/3,12/0
        VFD     3/3,3/4,3/5,3/4,3/4,3/4,3/4,3/3,3/4,3/3,3/3,3
                /2
 RHO    VFD     3/0,3/1,3/0,3/3,3/1,3/2,3/3,3/5,3/2,3/3,3/5,3
                /5
        VFD     3/1,3/3,3/3,3/3,3/3,3/4,3/3,3/4,12/0
 PI     VFD     3/1,3/1,3/1,3/3,3/3,3/3,3/1,3/3,3/0,3/1,3/2,3
                /3
        VFD     3/2,3/1,6/0,3/1,3/0,3/0,3/1,12/0
        VFD     3/3,3/4,3/4,3/3,3/4,3/3,3/3,3/1,3/3,3/2,3/1,3
                /0
OMICRN  VFD     3/0,3/1,3/1,3/3,3/1,3/2,3/3,3/4,3/2,3/3,3/4,3
                /4
        PZE     0
        VFD     3/1,3/2,3/2,3/1,3/2,3/3,3/1,3/1,3/3,3/2,3/1,3
                /0
        VFD    3/1.3/2.3/4.3/3.3/2.3/3.3/3.3/3.3/2.3/1.3/3.3
               /2
  XI    VFD    3/2.3/3.3/6.3/5.3/3.3/2.3/5.3/5.3/2.3/1.3/5.3
               /4
        VFD    3/2.3/3.3/1.3/2.3/3.3/4.3/2.3/4.12/0
  NU    VFD    3/0.3/1.3/4.3/3.3/1.3/0.3/3.3/1.3/0.3/2.3/1.3
               /1
        PZE    0
        VFD    3/2.3/3.3/1.3/2.3/3.3/3.3/2.3/4.3/3.3/4.3/2.3
               /1
  MU    VFD    3/0.3/1.3/0.3/4.3/1.3/1.3/4.3/2.3/1.3/2.3/2.3
               /1
        PZE    0
LAMBDA  VFD    3/0.3/1.3/5.3/5.3/1.3/4.3/5.3/0.3/2.3/0.3/3.3
               /0
        PZE    0
 KAPPA  VFD    3/1.3/0.3/4.3/0.3/3.3/1.3/4.3/2.3/2.3/3.3/3.3
               /0
        VFD    3/2.3/3.3/0.3/1.12/0
 IOTA   VFD    3/2.3/1.3/4.3/2.3/1.3/1.3/2.3/1.3/1.3/2.3/1.3
               /0
        PZE    0
        VFD    3/2.3/1.6/0.3/1.3/0.3/0.3/1.3/3.3/1.3/3.3/3
        VFD    3/3.3/4.3/6.3/5.3/4.3/3.3/5.3/1.3/3.3/2.3/1.3
               /0
 THETA  VFD    3/0.3/1.3/1.3/5.3/1.3/2.3/5.3/6.3/2.3/3.3/6.3
               /6
        VFD    3/3.3/3.3/1.3/0.24/0
        VFD    3/2.3/3.3/4.3/4.3/3.3/4.3/4.3/3.3/4.3/3.3/3.3
               /1
 ETA    VFD    3/0.3/1.3/4.3/4.3/1.3/0.3/4.3/1.3/0.3/2.3/1.3
               /4
        VFD    3/3.3/2.3/1.3/0.3/2.3/1.6/0.12/0
        VFD    3/1.3/1.3/4.3/2.3/1.3/2.3/2.3/1.3/2.3/3.3/1.3
               /1
 ZETA   VFD    3/2.3/3.3/6.3/5.3/4.3/3.3/5.3/5.3/4.3/1.3/5.3
               /4
        PZE    0
        VFD    3/0.3/1.3/1.3/0.3/1.3/3.6/0.3/3.3/0.3/2.3/2
 EPSLN  VFD    3/3.3/1.3/4.3/4.3/1.3/0.3/4.3/3.6/0.3/3.3/1
        VFD    3/1.3/1.3/1.3/2.3/1.3/2.3/2.3/3.12/0
        VFD    3/4.3/3.3/1.3/0.3/3.3/2.6/0.3/2.3/1.3/0.3/1
        VFD    3/1.3/3.3/4.3/3.3/3.3/4.3/3.3/2.3/4.3/4.3/2.3
               /1
 DELTA  VFD    3/3.3/2.3/5.3/6.3/2.3/1.3/6.3/5.3/1.3/1.3/5.3
               /4
        PZE    0
        VFD    3/2.3/1.3/0.3/1.3/1.3/1.3/1.3/2.3/1.3/4.3/2.3
               /6
 GAMMA  VFD    3/0.3/3.3/5.3/2.3/3.3/3.3/2.3/1.3/3.3/2.3/1.3
               /0
        VFD     3/3,3/2,6/0,3/2,3/1,3/0,3/1,12/0
        VFD     3/3,3/4,3/3,3/2,3/4,3/4,3/2,3/1,3/4,3/3,3/1,3
                /0
        VFD     3/3,3/4,3/6,3/5,3/4,3/4,3/5,3/4,3/4,3/2,3/4,3
                /3
 BETA   VFD     3/0,3/1,3/0,3/5,3/1,3/2,3/5,3/6,3/2,3/3,3/6,3
                /6
        VFD     3/2,3/4,3/4,3/0,24/0
        VFD     6/0,3/1,3/3,3/0,3/1,3/3,3/4,3/1,3/2,3/4,3/4
TABL2V  VFD     3/4,3/2,3/4,3/0,3/2,3/1,6/0,3/1,3/0,3/0,3/1
        PZE     ENTR2V
 ALPHA  EQU     TABL2V
        END
```

```
***     FAP
        COUNT   80
        ENTRY   TABL3V
        REM     TABLE OF GREEK CHARACTERS
        VFD     9/51,9/55,9/58,9/61
        VFD     9/41,9/44,9/47,9/49
        VFD     9/31,9/33,9/35,9/38
        VFD     9/21,9/23,9/26,9/28
        VFD     9/10,9/13,9/15,9/17
ENTR3V  VFD     9/0,9/2,9/6,9/8
        PZE     0
        VFD     3/4,3/4,3/5,3/3,3/4,3/3,3/3,3/0,3/3,3/4,6/0
        VFD     3/0,3/1,3/5,3/6,3/1,3/3,3/6,3/6,3/3,3/4,3/6,3
                /5
OMEGA   VFD     3/0,3/1,6/0,3/1,3/0,3/0,3/3,6/0,3/3,3/5
        VFD     3/2,3/2,3/0,3/6,3/1,3/3,3/6,3/6,12/0
        VFD     3/3,3/4,3/1,3/2,3/4,3/4,3/2,3/3,3/1,3/3,6/0
 PSI    VFD     6/0,3/3,3/2,3/0,3/1,3/2,3/1,3/1,3/3,3/1,3/1
        PZE     0
        VFD     3/1,3/0,3/6,3/6,3/0,3/4,3/6,3/0,3/3,3/4,6/0
 CHI    VFD     3/1,3/0,6/0,3/0,3/4,3/0,3/6,3/4,3/3,3/6,3/6
        VFD     3/2,3/2,3/0,3/6,3/3,3/1,3/6,3/6,12/0
        VFD     3/4,3/3,3/2,3/1,3/3,3/1,3/1,3/1,3/3,3/1,6/0
        VFD     3/1,3/3,3/5,3/5,3/3,3/4,3/5,3/4,3/4,3/4,3/4,3
                /2
 PHI    VFD     3/1,3/0,3/1,3/2,6/0,3/2,3/4,3/0,3/1,3/4,3/5
        VFD     3/3,3/4,3/6,3/5,3/2,3/2,3/4,3/0,12/0
UPSILN  VFD     3/0,3/1,3/5,3/6,3/1,3/2,3/6,3/4,3/2,3/3,3/4,3
                /6
        VFD     3/2,3/2,3/6,3/0,24/0
 TAU    VFD     6/0,3/5,3/6,3/0,3/4,3/6,3/6,3/4,3/4,3/6,3/5
        PZE     0
        VFD     3/2,3/0,3/3,3/6,3/0,3/4,3/6,3/6,3/4,3/4,3/6,3
                /5
 SIGMA  VFD     3/4,3/4,3/1,3/0,3/4,3/0,6/0,3/0,3/2,3/0,3/3
        VFD     3/3,3/0,3/3,3/3,24/0
        VFD     3/3,3/4,3/6,3/5,3/4,3/4,3/5,3/4,3/4,3/3,3/4,3
                /3
 RHO    VFD     3/0,3/1,6/0,6/0,3/0,3/6,3/0,3/3,3/6,3/6
        PZE     0
        VFD     3/4,3/3,6/0,3/3,3/3,3/0,3/6,3/4,3/3,3/6,3/6
 PI     VFD     3/0,3/1,6/0,3/1,3/1,3/0,3/6,3/0,3/1,3/6,3/6
        VFD     3/4,3/3,3/1,3/0,3/3,3/1,6/0,12/0
        VFD     3/1,3/3,3/6,3/6,3/3,3/4,3/6,3/5,3/4,3/4,3/5,3
                /1
OMICRN  VFD     6/0,3/1,3/5,3/0,3/1,3/1,3/0,3/0,3/1,3/5,3/6
        PZE     0
 XI     VFD     3/0,3/4,3/6,3/6,3/1,3/3,3/3,3/3,3/0,3/4,6/0
        VFD     3/4,3/4,3/0,3/6,3/3,3/4,3/6,3/6,12/0
 NU     VFD     3/0,3/1,6/0,6/0,3/0,3/6,3/0,3/4,3/6,3/0,
        PZE     0
        VFD     3/2,3/4,3/2,3/6,3/4,3/4,3/6,3/0,3/3,3/4,6/0
 MU     VFD     3/0,3/1,6/0,6/0,3/0,3/6,3/0,3/2,3/6,3/2
        VFD     3/4,3/3,6/0,24/0
LAMBDA  VFD     3/1,3/0,6/0,3/2,3/0,3/6,3/0,3/4,3/2,3/0,3/6
        VFD     3/4,3/3,6/0,24/0
        VFD     3/4,3/1,3/6,3/2,3/3,3/4,3/6,3/6,3/2,3/4,3/3,3
                /0
 KAPPA  VFD     3/1,3/1,3/0,3/6,3/0,3/1,6/0,3/0,3/1,3/6,3/6
        PZE     0
 IOTA   VFD     3/3,3/1,3/6,3/6,3/3,3/1,6/0,3/2,3/2,3/6,3/0
        PZE     0
        VFD     3/4,3/3,3/1,3/0,3/3,3/1,6/0,3/4,3/0,3/3,3/3
        VFD     3/1,3/3,3/6,3/6,3/3,3/4,3/6,3/5,3/4,3/4,3/5,3
                /1
 THETA  VFD     6/0,3/5,3/1,3/1,3/0,3/6,3/5,3/1,3/0,3/0,3/1
        PZE     0
 ETA    VFD     6/0,3/6,3/0,3/4,3/0,3/3,3/3,3/4,3/4,3/0,3/6
        VFD     3/4,3/0,6/0,3/4,3/4,3/1,3/0,12/0
 ZETA   VFD     6/0,3/5,3/6,3/4,3/0,3/6,3/6,3/4,3/0,3/6,3/0
        VFD     3/0,3/3,3/3,3/3,24/0
        VFD     3/4,3/4,3/1,3/0,3/4,3/4,3/6,3/5,3/3,3/3,3/4,3
                /2
 EPSLN  VFD     6/0,3/0,3/6,3/4,3/0,6/0,3/4,3/0,3/6,3/6
        PZE     0
 DELTA  VFD     3/4,3/0,6/0,3/2,3/0,3/6,3/0,3/4,3/2,3/0,3/6
        VFD     3/4,3/4,3/6,3/5,24/0
 GAMMA  VFD     3/1,3/1,3/0,3/6,3/0,3/2,6/0,3/1,3/4,3/6,3/6
        VFD     3/4,3/3,3/1,3/0,3/3,3/0,18/0
        VFD     3/3,3/0,3/3,3/3,3/4,3/3,3/2,3/3,3/4,3/4,3/1,3
                /2
        VFD     3/3,3/4,3/6,3/5,3/4,3/4,3/5,3/4,3/4,3/3,3/4,3
                /3
 BETA   VFD     3/0,3/3,6/0,6/0,3/0,3/6,3/0,3/3,3/6,3/6
        VFD     3/3,3/4,6/0,3/1,3/3,3/2,3/2,12/0
TABL3V  VFD     3/0,3/1,6/0,3/0,3/2,3/0,3/6,3/2,3/4,3/6,3/0
        PZE     ENTR3V
        END
```

```
***     FAP
                           END 0000
        COUNT    50
*       THE END
        ENTRY    TABL5V
        VFD      9/17,9/21,18/0
ENTR5V  VFD      9/0,9/3,9/8,9/13
 BLANK  PZE
        VFD      3/2,3/1,3/2,3/1,3/0,3/0,3/0,3/0,3/0,3/0,3/0,3
                 /0
        VFD      3/1,3/1,3/1,3/4,3/1,3/2,3/4,3/3,3/2,3/2,3/3,3
                 /2
        VFD      3/3,3/3,3/4,3/1,3/3,3/2,3/1,3/0,3/2,3/0,3/0,3
                 /0
        VFD      3/0,3/0,3/0,3/5,3/0,3/2,3/5,3/5,3/2,3/3,3/5,3
                 /4
        VFD      3/1,3/0,3/0,3/0,3/0,3/0,3/0,3/0,3/0,3/0,3/0,3
                 /0
        VFD      3/3,3/2,3/0,3/0,3/2,3/1,3/0,3/3,3/1,3/1,3/3,3
                 /0
        VFD      3/2,3/2,3/2,3/5,3/2,3/3,3/5,3/5,3/3,3/3,3/5,3
                 /0
        VFD      3/0,3/0,3/0,3/5,3/0,3/1,3/5,3/5,3/1,3/2,3/5,3
                 /2
        PZE
        VFD      3/1,3/3,3/1,3/1,3/3,3/3,3/1,3/0,3/3,3/0,3/0,3
                 /0
        VFD      3/2,3/2,3/3,3/2,3/2,3/1,3/2,3/2,3/1,3/1,3/2,3
                 /1
        VFD      3/3,3/1,3/4,3/4,3/1,3/1,3/4,3/3,3/1,3/2,3/3,3
                 /3
        VFD      3/0,3/0,3/0,3/5,3/0,3/3,3/5,3/5,3/3,3/3,3/5,3
                 /4
        PZE
        VFD      3/2,3/1,3/2,3/2,3/1,3/1,3/2,3/0,3/1,3/0,3/0,3
                 /0
        VFD      3/3,3/3,3/5,3/0,3/3,3/2,3/0,3/0,3/2,3/2,3/0,3
                 /2
        VFD      3/1,3/2,3/3,3/3,3/2,3/2,3/3,3/5,3/2,3/3,3/5,3
                 /5
        VFD      3/0,3/0,3/0,3/5,3/0,3/1,3/5,3/5,3/1,3/1,3/5,3
                 /3
        VFD      3/2,3/2,3/4,3/0,3/2,3/1,3/0,3/0,3/0,3/0,3/0,3
                 /0
        VFD      3/0,3/3,3/5,3/5,3/3,3/3,3/5,3/4,3/3,3/2,3/4,3
                 /4
TABL5V  VFD      3/1,3/1,3/0,3/4,3/1,3/0,3/4,3/4,3/0,3/0,3/4,3
                 /5
        PZE      ENTR5V
        END
```

```
***     FAP
                      LINE0000
        COUNT   25
*               LINEV SCATRAN CONVERSION SUBROUTINE
        ENTRY   LINE
LINE    SXA     (X4).4
        CLA*    1.4
        STO     A1
        CLA*    2.4
        STO     A2
        CLA*    3.4
        STO     A3
        CLA*    4.4
        STO     A4
        TSX     $NXV.4
        TSX     A1
        STO     A1
        TSX     $NYV.4
        TSX     A2
        STO     A2
        TSX     $NXV.4
        TSX     A3
        STO     A3
        TSX     $NYV.4
        TSX     A4
        STO     A4
        CALL    LINEV.A1.A2.A3.A4
(X4)    AXT     **.4
        TTR     5.4
A1      PZE
A2      PZE
A3      PZE
A4      PZE
        END
```

```
 ***    FAP
        COUNT    100
        ENTRY    LINEV
*      VECTOR GENERATOR
*  CALL LINEV  (NX0,NY0,NX1,NY1 )
 LINEV SXA      X4,4
        SXA      X2,2
        SXA      X1,1
        CLS*     2,4                  STARTING NY
        ADD     =01777001777          CORRECT SIGN CONVENTION
        ARS      18
        ADD*     1,4
        ANA     =01777001777
        SLW      XX
* COMPUTE DIRECTION AND LENGTH OF VECTOR
        CLA*     4,4
        SUB*     2,4
        XCA
        CLA*     3,4
        SUB*     1,4
*  THIS IS REENTRY POINT FROM LOOP
 V0     STO      DELX
        STD      XINC
        STQ      DELY
        CLM
        ORA     =1B19
        TQP      *+2
        CLM
        TPI      *+2
        ORA     =1B18                 1 = +, 0 = -
        STT      CODE
*  REDUCE INCREMENTS TO 63*63 SQUARE
        LXD      DELX,1
        LXD      DELY,2
        TXL      V12,1,63
        TXL      V3,2,63
 XINC   TXL      V3,2,**
*
 V2     TSX      SCAL,1
                 DELX
                 DELY
                 LX
                 LY
 V4     TSX      PLOT,1
        CLS      LY                   PREPARE INCREMENT OF
        ARS      18                     STARTING POINT
        ADD      XX
        ADD      LX
        STO      XX                   NEW STARTING POINT
        CLA      DELY
        SUB      LY
        XCA
        CLA     DELX
        SUB     LX
 NTN    TXI     V0
*
 V3     TSX     SCAL,1
                DELY
                DELX
                LY
                LX
        TRA     V4
*
 V12    TXH     V2,2,63
 V1     CLA     DELX                 X AND Y LESS THAN 64
        STO     LX
        CLA     DELY
        STO     LY
        TSX     PLOT,1
 X4     AXT     ,4
 X2     AXT     ,2
 X1     AXT     ,1
        TRA     5,4
*
 SCAL   LDQ*    2,1
        CLA     =63B17
        LLS     0
        STO*    4,1
        LRS     18
        DVH*    2,1
        MPR*    1,1
        ALS     18
        STO*    3,1
        TRA     5,1
*
 PLOT   CLA     LY
        ARS     18
        ORA     LX
        ALS     10
        ORA     CODE
        ORA     XX
        CALL    (PLOT)
        TRA     1,1
 CODE   BCI     1, 00000
 XX                                  CURRENT STARTING ORDINATE
                S
 DELX                                SIGNED INCREMENTS
 DELY
 LX                                  SUB-INCREMENTS
 LY
        END
```

```
***      FAP
                       VXAX1000
         COUNT    35
*                 VXAXV SCATRAN CONVERSION SUBROUTINE
         ENTRY    VXAX
         ENTRY    VYAX
VXAX     SXA      (X1)•1
         AXT      0•1
         TRA      *+3
VYAX     SXA      (X1)•1
         AXT      1•1
         SXA      (X4)•4
         CLA*     1•4
         STO      A1
         CLA*     2•4
         STO      A2
         CLA*     3•4
         STO      A3
         CLA*     4•4
         ALS      18
         STO      A4
         TSX      $NXV•4
         TSX      A1
         STO      A1
         TSX      $NYV•4
         TSX      A2
         STO      A2
         TXH      *+4•1•0
         TSX      $NXV•4
         TSX      A3
         TTR      AY
         TSX      $NYV•4
         TSX      A3
AY       STO      A3
         TXH      AX•1•0
         CALL     VXAXV•A1•A2•A3•A4
         TTR      (X1)
AX       CALL     VYAXV•A1•A2•A3•A4
(X1)     AXT      **•1
(X4)     AXT      **•4
         TTR      5•4
A1       PZE
A2       PZE
A3       PZE
A4       PZE
         EJECT
         END
```

```
***    FAP
       COUNT    60
       ENTRY    VXAXV
       ENTRY    VYAXV
       REM      VXAXV  -VARIABLE LENGTH AXES
       REM      VYAXV - VARIABLE LENGTH Y AXIS
       REM      ***********************************************
                ************
       REM      *
       REM      * FORMS FOR VARIABLE LENGTH ARE
       REM      *
       REM      * CALL VXAXV (IX.IY.IXEND.INT)
       REM      * CALL VYAXV(IX.IY.IYEND.INT)
       REM      *
       REM      * INT IS INTENSITY FACTOR
       REM      * IX.IY IS ORIGIN
       REM      * IXEND OR IYEND IS END OF LINE
       REM      *
       REM      * MIN LENGTH MAY BE 16 COUNTS
       REM      * MAX LENGTH MAY BE 1023 COUNTS BUT
       REM      * IXEND (OR IYEND) MUST BE 1023 OR LESS
       REM      *
       RFM      * IF IX(OR IY) IS LARGER THAN IXEND (OR
       REM      * IYEND). IXEND (OR IYEND) IS ORIGIN
       REM      *
       REM      * IF LENGTH IS LESS THAN 16.16 IS USED
       REM      * IF END IS MORE THAN 1023.1023 IS USED
       REM      *
       REM      ***********************************************
                ************
VXAXV  SXA      X1.1                SAVE IR1
       AXT      1.1
       LDQ*     1.4                 IX
       CLA*     2.4
       STO      IY                    IY
       TRA      HV
VYAXV  SXA      X1.1                SAVE IR1
       AXT      2.1
       LDQ*     2.4              IY
       CLA*     1.4
       STO      IX                    IX
HV     SXA      X4.4                  SAVE IR4
       CLA*     3.4
       TLQ      *+2
       XCA
       STQ      IY+2.1                ORIGIN X OR Y
COMPL  SUB      N1023            COMPLEMENT
       TMI      *+2
       PXD      0                     ZERO IF TOO LARGE
       LGR      26
       ALS      10
```

```
        LGL     18
        ORA     IX
        ORA     OPER+2,1
        SLW     IX                  SAVE-WORD HAS ALL BUT Y
        CLA*    4,4
        PDX     0,1                 INTENSITY FACTOR
        CLA     IY
        SUB     N1023
        ARS     18
        ORA     IX                  COMBINE REST OF WORD
        TSX     $(PLOT),4
        TIX     *-5,1,1                 REPEAT FOR INTENSITY
X4      AXT     **,4
X1      AXT     **,1
        TRA     5,4
IY      PZE     0
IX      PZE     0
N1023   PZE     0,0,1023
N16     PZE     0,0,16
OPER    OCT     320000000000        OPERATION FOR Y AXIS
        OCT     300000000000        OPERATION FOR X AXIS
        END
```

```
 ***    FAP
        COUNT    30
        ENTRY    XAXISV
        ENTRY    YAXISV
* AXIS  GENERATOR
        SPACE    2
XAXISV  CAL      XAX
        TRA      *+2
YAXISV  CAL      YAX
        SLW     =HERE
        SXA      X4,4
        CLS*     2,4                   NY
        ADD     =O1777001777           CORRECT SIGN CONVENTION
        ARS      18
        ADD*     1,4                   NX
        ANA     =O1777001777
        ORA     =HERE
        CALL     (PLOT)
 X4     AXT      **,4
        TRA      3,4
 XAX    OCT      300000000000
 YAX    OCT      320000000000
        END
```

```
***     FAP
                          GYSM0000
        COUNT    100
*       GENERATOR SYMBOLS
        ENTRY    TABL4V
        ENTRY    GENSYM
GENSYM  SXA      (X4)•4
        SXA      (X2)•2
        TSX      $CHSIZV•4
        TSX      =016000000•0
        TSX      =016000000•0
        TSX      $VCHARV•4
        TSX      =0132000000•0
        TSX      =01000000•0
        TSX      =0717000000•0
        TSX      =0734000000•0
        TSX      =03000000•0
        TSX      TABLE•0
        TSX      $VCHARV•4
        TSX      =0416000000•0
        TSX      =01000000•0
        TSX      =01060000000•0
        TSX      =01044000000•0
        TSX      =03000000•0
        TSX      TABLE•0
        TSX      $CHSIZV•4
        TSX      =022000000•0
        TSX      =022000000•0
        TSX      $VCHARV•4
        TSX      =0132000000•0
        TSX      =01000000•0
        TSX      =01000000000•0
        TSX      =0734000000•0
        TSX      =04000000•0
        TSX      TABLE•0
        TSX      $CHSIZV•4
        TSX      =05000000•0
        TSX      =05000000•0
        TSX      $VCHARV•4
        TSX      =0132000000•0
        TSX      =01000000•0
        TSX      =0757000000•0
        TSX      =0757000000•0
        TSX      =0•0
        TSX      TABLE•0
        LXA      (X4)•4
        CLA*     1•4
        PAX      •2
        ALS      18
        STO      A1
        CLA*     2•4
        ALS     18
        STO     A2
        ORA     =01000000000
        TXL     ODD1•2•1
        CHS
        ADD     =01777000000
        STO     A3
        CLA     =070000000000
        TTR     ODD2
ODD1    STO     A3
        CLA     =060000000000
ODD2    STO     A4
        TSX     $VCHARV•4
        TSX     =0132000000•0
        TSX     =01000000•0
        TSX     =0757000000•0
        TSX     =0757000000•0
        TSX     A1•0
        TSX     TABLE•0
        TSX     $RITSTV•4
        TSX     =0100000000•0
        TSX     =0100000000•0
        TSX     TABLE•0
        TSX     $CHSIZV•4
        TSX     A2•0
        TSX     A2•0
        TSX     $RITE2V•4
        TSX     =01006000000•0
        TSX     =01000000000•0
        TSX     =01500000000•0
        TSX     =0132000000•0
        TSX     =01000000•0
        TSX     =01000000•0
        TSX     =0-1000000•0
        TSX     =0500000000000•0
        TSX     A10•0
        TSX     $CHSIZV•4
        TSX     =03000000•0
        TSX     =03000000•0
        TSX     $RITE2V•4
        TSX     =01006000000•0
        TSX     A3•0
        TSX     =01500000000•0
        TSX     =0132000000•0
        TSX     =01000000•0
        TSX     =01000000•0
        TSX     =0-1000000•0
        TSX     A4•0
        TSX     A10•0
(X2)    AXT     **•2
(X4)    AXT     **•4
```

```
        TTR     3,4
        VFD     9/9,9/10,9/11,9/12
ENTR4V  VFD     9/0,9/3,9/5,9/7
        VFD     3/0,3/2,3/6,3/3,3/2,3/4,3/3,3/6,3/0,3/0,3/0,3
                /0
        VFD     3/0,3/2,3/0,3/3,3/2,3/4,3/3,3/0,3/0,3/0,3/0,3
                /0
        VFD     3/2,3/2,3/2,3/4,3/0,3/0,3/0,3/0,3/0,3/0,3/0,3
                /0
        VFD     3/0,3/0,3/0,3/1,3/0,3/0,3/3,3/4,3/0,3/0,3/0,3
                /0
        VFD     3/5,3/6,3/0,3/2,3/6,3/7,3/2,3/2,3/0,3/0,3/0,3
                /0
        VFD     3/0,3/1,3/2,3/2,3/1,3/2,3/2,3/0,3/2,3/5,3/0,3
                /0
        PZE
        VFD     3/1,3/2,3/4,3/3,3/2,3/5,3/3,3/4,3/5,3/6,3/4,3
                /3
        PZE
        VFD     3/1,3/2,3/3,3/4,3/2,3/5,3/4,3/3,3/5,3/6,3/3,3
                /4
        VFD     3/7,3/5,3/2,3/0,3/5,3/2,3/0,3/0,3/0,3/0,3/0,3
                /0
        VFD     3/2,3/5,3/7,3/7,3/5,3/7,3/7,3/5,3/7,3/7,3/5,3
                /2
TABL4V  VFD     3/2,3/0,3/0,3/2,3/0,3/0,3/2,3/5,3/0,3/2,3/5,3
                /7
        PZE     ENTR4V
 TABLE  TTR     TABL4V
 A1     PZE
 A2     PZE
 A3     PZE
 A4     PZE
 A10    PZE
        END
```

```
***     FAP
                        RULE0000
        COUNT   100
*       RULER MARKS
        ENTRY   RULE

RULE    SXA     (X4),4
        TSX     $CHSIZV,4
        TSX     =03000000,0
        TSX     =03000000,0
        TSX     $VCHARV,4
        TSX     =0132000000,0
        TSX     =04000000,0
        TSX     =0773000000,0
        TSX     =01105000000,0
        TSX     =03000000,0
        TSX     TABLE,0

        TSX     $VCHARV,4
        TSX     =0132000000,0
        TSX     =04000000,0
        TSX     =0773000000,0
        TSX     =0656000000,0
        TSX     =03000000,0
        TSX     TABLE,0

        TSX     $CHSIZV,4
        TSX     =022000000,0
        TSX     =033000000,0
        TSX     $VCHARV,4
        TSX     =0132000000,0
        TSX     =01000000,0
        TSX     =0752000000,0
        TSX     =0656000000,0
        TSX     =01000000,0
        TSX     TABLE,0
        TSX     $VCHARV,4
        TSX     =0,0
        TSX     =01000000,0
        TSX     =01460000000,0
        TSX     =01025000000,0
        TSX     =01000000,0
        TSX     TABLE,0

        TSX     $CHSIZV,4
        TSX     =03000000,0
        TSX     =031000000,0
        TSX     $VCHARV,4
        TSX     =0132000000,0
        TSX     =03000000,0
        TSX     =0755000000,0
        TSX    =0664000000•0
        TSX    =02000000•0
        TSX    TABLE•0
        TSX    SVCHARV•4
        TSX    =0•0
        TSX    =03000000•0
        TSX    =01465000000•0
        TSX    =01022000000•0
        TSX    =02000000•0
        TSX    TABLE•0

        TSX    SCHSIZV•4
        TSX    =03000000•0
        TSX    =011000000•0
        TSX    SVCHARV•4
        TSX    =0132000000•0
        TSX    =01000000•0
        TSX    =0755000000•0
        TSX    =0674000000•0
        TSX    =0•0
        TSX    TABLE•0
        TSX    SVCHARV•4
        TSX    =0132000000•0
        TSX    =01000000•0
        TSX    =0755000000•0
        TSX    =01004000000•0
        TSX    =0•0
        TSX    TABLE•0
        TSX    SVCHARV•4
        TSX    =0•0
        TSX    =01000000•0
        TSX    =01475000000•0
        TSX    =01022000000•0
        TSX    =0•0
        TSX    TABLE•0
        TSX    SVCHARV•4
        TSX    =0•0
        TSX    =01000000•0
        TSX    =01605000000•0
        TSX    =01022000000•0
        TSX    =0•0
        TSX    TABLE•0

 (X4)   AXT    **•4
        TTR    1•4

        PZE
ENTR6V  VFD    9/0•9/3•9/6•9/7
        PZE
 STAR   OCT    130431040422
 LINES  OCT    050005660000
```

```
       OCT    112010000000
       OCT    016611641142
RULER  OCT    000200240046
       OCT    156635770000
       OCT    353315443555
MARKS  OCT    150035111522
       PZE    ENTR6V
TABLE  TTR    MARKS
       END
```

```
 ***    FAP
                     QDPT0000
        COUNT    200
*       USED TO PUT SCALED DATA INTO BUFFERS
*       4 QUAD (PLOT) REPEAT NUM. 1
        ENTRY    (PLOT)
        ENTRY    LASTFM
        ENTRY    (CLEAN
(PLOT)  SXA      (X4).4                     SAVE XR4 IN (X4)
        TPL      *+5
        XCA
        XCL
        PAI
        TTR      BETA
        PAI                                 SPACES OF WORD
        ANA      =0770000000000
        TNZ      BETA
        PIA                                 X AXIS
        ANA      =01777                     POINTS
        ERA      =01000                     FOR
        TZE      ALPHA                      Y = ZERO
        PIA
        SLW      ACC
        CALL     STORE.ACC.N                PLOT QUAD I
        CAL      ACC
        ANA      =0777777                   MASK Y+CHAR
        SLW      ACC1                       SAVE
        CAL      ACC
        ANA      =01777000000               MASK X
        CHS                                 CHANGE SIGN
        ADD      =01777000000               SHIFT X TO QUAD II
        ORS      ACC1                       SAVE
        CALL     STORE.ACC1.N               PLOT QUAD II
        CAL      ACC                        QUAD IV AND III. GO T
                 O A2
        ANA      =01777770000               MASK X+CHAR
        SLW      ACC1                       SAVE
        CAL      ACC
        ANA      =01777                     MASK Y
        CHS                                 CHANGE SIGN
        ADD      =01777                     SHIFT Y TO QUAD IV
        ORS      ACC1                       SAVE
 ALPHA  PIA
        SLW      ACC1                       IF Y = ZERO
 A4     CALL     STORE.ACC1.N               PLOT QUAD IV
        CAL      ACC1
        ANA      =0777777                   MASK Y+CHAR
        SLW      ACC                        SAVE
        CAL      ACC1
        ANA      =01777000000               MASK X
        CHS                                 CHANGE SIGN
        ADD     =01777000000                SHIFT X TO QUAD III
        ORS     ACC                         SAVE
        CALL    STORE,ACC,N                 PLOT QUAD III
 (X4)   AXT     **,4
        TTR     1,4                         AAWGWH
 BETA   PIA                                 PLOT WORDS
        SLW     ACC                         WITH OP CODES
        CALL    STORE,ACC,N
        TTR     (X4)
(CLEAN  SXA     (X4),4                      CLOSE BUFFER
        CALL    STORE,ACC,N1                AND WRITE ALL WORDS
        TTR     (X4)
LASTFM  SXA     (X4A),4
        CALL    EFRAME                      WRITE CLOSING ID FRAM
                E
        CALL    STORE,ACC,N1
 (X4A)  AXT     **,4
        TTR     1,4
 ACC    PZE
 ACC1   PZE
 N      PZE
 N1     OCT     1
        END
 ***     FAP
         COUNT    200
*        USED TO PUT SCALED DATA INTO BUFFERS
*        4 QUAD (PLOT) REPEAT NUM. 2
         ENTRY    (PLOT)
         ENTRY    LASTFM
         ENTRY    (CLEAN
(PLOT)   SXA      (X4),4                       SAVE XR4 IN (X4)
         PAI                                   SPACES OF WORD
         ANA      =0770000000000
         TNZ      BETA
         SXA      (X1),1                       SAVE XR1 IN (X1)
         SXA      (X2),2                       SAVE XR2 IN (X2)
         LXA      COUNT,1                      ADD TO XR1
         LXD      COUNT,2                      DEC TO XR2
         LNT      1000
         TTR      GAMMA
         PIA                                   X AXIS
         ANA      =01777                       POINTS
         ERA      =01000                       FOR
         TZE      ALPHA                        Y = ZERO
         PIA
         SLW      ACC
         TXL      A1,2,11                      IF XR2 = 0 TO 11 DONT
                   PLOT
         CALL     STORE,ACC,N                  PLOT QUAD I
         CAL      ACC
         ANA      =0777777                     MASK Y+CHAR
         SLW      ACC1                         SAVE
         CAL      ACC
         ANA      =01777000000                 MASK X
         CHS                                   CHANGE SIGN
         ADD      =01777000000                 SHIFT X TO QUAD II
         ORS      ACC1                         SAVE
         CALL     STORE,ACC1,N                 PLOT QUAD II
 A1      TXL      A2,1,11                      IF XR1 = 0 TO 11  DON
                  T PLOT
         CAL      ACC                          QUAD IV AND III, GO T
                  O A2
         ANA      =01777770000                 MASK X+CHAR
         SLW      ACC1                         SAVE
         CAL      ACC
         ANA      =01777                       MASK Y
         CHS                                   CHANGE SIGN
         ADD      =01777                       SHIFT Y TO QUAD IV
         ORS      ACC1                         SAVE
         TTR      A4                           SKIP ALPHA
 ALPHA   LXA      CALLS,1
         TXL      A8,1,11
         PIA
         SLW      ACC1                         IF Y = ZERO
```

```
A4      CALL    STORE,ACC1,N               PLOT QUAD IV
        CAL     ACC1
        ANA     =0777777                   MASK Y+CHAR
        SLW     ACC                        SAVE
        CAL     ACC1
        ANA     =01777000000               MASK X
        CHS                                CHANGE SIGN
        ADD     =01777000000               SHIFT X TO QUAD III
        ORS     ACC                        SAVE
        CALL    STORE,ACC,N                PLOT QUAD III
        NZT     CONT                       CONT = 0, COUNT STAYS
        TTR     *+4
A8      LXA     COUNT,I
A2      TXI     *+1,1,1                    XR1 INC BY 1
        TXI     *+1,2,1                    XR2 INC BY 1
A7      TXL     *+3,1,35                   IF XR1 IS 36
        SXA     COUNT,0                    SET ADD OF COUNT
        TTR     *+2                        TO ZERO
        SXA     COUNT,1                    OTHERWISE TO XR1
        TXL     *+3,2,35                   SAME FOR XR2
        SXD     COUNT,0
        TTR     *+2
        SXD     COUNT,2                    XR2 TO DEC OF COUNT
(X1)    AXT     **,1
(X2)    AXT     **,2
(X4)    AXT     **,4
        TTR     1,4                        AAWGWH
BETA    PIA                                PLOT WORDS
        SLW     ACC                        WITH OP CODES
        CALL    STORE,ACC,N
        TTR     (X4)
(CLEAN  SXA     (X4),4                     CLOSE BUFFER
        CALL    STORE,ACC,N1               AND WRITE ALL WORDS
        TTR     (X4)
LASTFM  SXA     (X4A),4
        CALL    EFRAME                     WRITE CLOSING ID FRAM
                E
        CALL    STORE,ACC,N1
(X4A)   AXT     **,4
        TTR     1,4
GAMMA   RNT     20000
        TTR     A2
        STL     CONT
        PIA
        ANA     =07000777
A20     PAX     ,4
        TXL     *+3,4,35
        SUB     =36
        TTR     A20
        SXA     CALLS,4
        PDX     ,4
        TXL     A5,4,0
        TXI     *+1,4,-1
        TXL     A6,4,0
        LXA     CALLS,2
        PXA     ,2
        SSM
        TXH     *+3,2,11
        ADD     =11
        TTR     *+2
        ADD     =47
        PAX     ,1
        TTR     A7
A5      LXA     CALLS,1                     PUT ADD OF CALLS IN X
                R1
        PXA     ,1
        SSM
        TXH     *+3,1,11
        ADD     =11
        TTR     *+2
        ADD     =47
        PAX     ,2
        TTR     A7
A6      CLA     =024000024                  ADD = 20, DEC = 20
        STO     COUNT                       STORE IN COUNT
        STZ     CONT
        TTR     (X1)
ACC     PZE
ACC1    PZE
N       PZE
N1      OCT     1
CALLS   PZE
COUNT   PZE
CONT    PZE
        END
```

```
***     FAP
                       CUTM0000
        COUNT     100
*                 CUT MARKS
        ENTRY     CUTIV
        ENTRY     CUTRV
        REM       ********************************************
                  ************
        REM       *
        REM       * CALL CUTIV (N) FOR CUT MODE
        REM       * SETS INDICATOR FOR CUTTER MARK.
        REM       * IT IS SENSED WHEN FRAMEV CALLS CUTRV
        REM       * N=0. FOR NO CUT MARK
        REM       * N=1 FOR CUT MARK SUITABLE FOR 11X
        REM       * N=2 FOR CUT MARK SUITABLE FOR 15X
        REM       * WHEN USING FORM SLIDE
        REM       *
        REM       * TESTS INDICATOR AND PUTS ON INDICATED
        REM       * CUT MARK
        REM       *
        REM       ********************************************
                  ************
CUTIV   CLA*      1.4
        STO       CUT
        TRA       2.4
CUTRV   CLA       CUT
        TZE       1.4
        SXA       X4.4
        SXA       X1.1
        SUB       N2
        TNZ       CX15                   TRA TO 15X CUT
CX11    CLA       N1023
        TRA       *+2
CX15    CLA       N710
        STO       IX
        AXT       6.1
MRK     TSX       $MARKV.4
        TSX       IX                     IX
        TSX       IY                     IY
        TSX       LX                     LENGTH IN X
        TSX       LY                     LENGTH IN Y
        TSX       INT                    INTENSITY
        CLA       IX
        SUB       N1
        STO       IX
        TIX       MRK.1.1
X4      AXT       **.4
X1      AXT       **.1
        TRA       1.4
CUT     PZE
N1      PZE       0.0.1
```

```
N2      PZE       1
N1023   PZE       0,0,1023
N710    PZE       0,0,710
IX      PZE       0,0,**
IY      PZE       0,0,1023
LX      PZE       0,0,0
LY      PZE       0,0,63
INT     PZE       0,0,2
        END
```

```
***     FAP
                        MRKV0000
        COUNT    50
        ENTRY    MARKV
        REM      USED IN CUT MARKS
        REM      CALL MARKV(IX,IY,LX,LY,INT)
        REM      PUTS VECTOR AT ANY COORDINATE,IX,IY
        REM      INT.=INTENSITY FACTOR
        REM      LX = X COMPONENT MUST BE LESS 64
        REM      LY = Y COMPONENT MUST BE LESS 64
MARKV   SXA      MEND4,4
        SXA      MEND2,2
        CLA*     5,4              INTENSITY
        PDX      0,2
        CLA*     4,4             Y COMPONENT
        TMI      *+2
        ORA      ONE11
        SSP
        ARS      18
        ACL*     3,4              X COMPONENT
        PBT
        ORA      ONE28
        ALS      10
        STO      TEMP
        CLA*     2,4             IY COORD
        SSM
        ADD      TEN23                     MACHINE REFERENCE
        ARS      18
        ACL*     1,4              IX COORD
        ANA      =0001777001777            001777001777
        ADD      TEMP
        ACL      OPCOD                                OP CODE 600....
        SLW      TEMP
        CAL      TEMP
        TSX      $(PLOT),4
        TIX      *-2,2,1
MEND2   AXT      **,2
MEND4   AXT      **,4
        TRA      6,4
ONE11   OCT      000100000000                    FOR -Y DEFL.
ONE28   OCT      000000000200              FOR  -X  DEFL.
TEN23   PZE      0,0,1023
OPCOD   OCT      600000000000
TEMP    PZE      0
        END
```

```
 ***    SCATRAN
C              DOT TEST PROGRAM FOR SCATRAN @
               DIMENSION (X(350).Y(350))@
  START        CALL SUBROUTINE()=OUT.(2)@
               CALL SUBROUTINE()=CAMRAV.(9)@
               READ INPUT.S25.(RL.RI.THETA.SD.TP.NP.NC)@
F S25          (4F16.8/F16.8/2I8)@
               JL = 1@
               DO THROUGH(S20).K=1.1.K.LE.NP@
               CALL SUBROUTINE()=FRAMEV.()@
               NUM = 0@
               L = JL@
               R = RL@
               DO THROUGH (S15).I=1.1.I.LE.NC@
               FR = 6.28319531*R@
               N = FR/SD@
               SF = FR/N@
               TI = SF/R@
               JLN = JL+N@
               DO THROUGH(S10).J = JL.1.J.L.JLN@
               X(J) = R*COS.(THETA)@
               Y(J) = R*SIN.(THETA)@
  S10          THETA = THETA+TI@
               JL = JLN@
               R = R+RI@
  S15          NUM = NUM+N@
               NR = -NUM@
               CALL SUBROUTINE()=APLOTV.(NR.X(L).Y(L).1.1.1.44.
                  IR)@
  S20          THETA = THETA+TP@
               CALL SUBROUTINE()=CLOSTP.()@
               END PROGRAM(START)@
 ***    DATA
402005
    4.0     4.0        0.0        0.0        -4.0        4.0        0.0
                       .0
SC4020 TEST PRGM.OBRENTON R. GROVESOELECT. ENG.O 1/28/660
      1.0                    .5                    .0                  1.0
      1.5707963
       4        6
       END OF DATA
```

```
777633011612  777633211576  777534011616  777534211572  300000000000  460000000000
001177540777  001077540621  000700540621  000600541000  000700541156  001077541156  001277540777  001222540604
001041540503  000640540531  000513540676  000513541101  000640541246  001041541274  001222541173  001377540777
001335540600  001177540442  000777540400  000600540442  000442540600  000400541000  000442541177  000600541335
001000541377  001177541335  001335541177  001477540777  001444540575  001325540422  001142540317  000736540302
000540540353  000375540504  000307540675  000307541102  000375541273  000540541424  000736541475  001142541460
001325541355  001444541202  001577540777  001550540574  001445540411  001277540263  001102540206  000675540206
000500540263  000332540411  000227540574  000200541000  000227541203  000332541366  000500541514  000675541571
001102541571  001277541514  001445541366  001550541203  001677540777  001653540574  001561540403  001427540242
001243540137  001041540101  000634540113  000440540174  000267540317  000154540475  000105540675  000105541102
000154541302  000267541460  000440541603  000634541664  001041541676  001243541640  001427541535  001561541374
001653541203  460000000000
000777540600  000621540700  000621541077  001000541177  001156541077  001156540700  000777540500  000604540555
000503540736  000531541137  000676541264  001101541264  001246541137  001274540736  001173540555  000777540400
000600540442  000442540600  000400541000  000442541177  000600541335  001000541377  001177541335  001335541177
001377540777  001335540600  001177540442  000777540300  000575540333  000422540452  000317540635  000302541041
000353541237  000504541402  000675541470  001102541470  001273541402  001424541237  001475541041  001460540635
001355540452  001202540333  000777540200  000574540227  000411540332  000263540500  000206540675  000206541102
000263541277  000411541445  000574541550  001000541577  001203541550  001366541445  001514541277  001571541102
001571540675  001514540500  001366540332  001203540227  000777540100  000574540124  000403540216  000242540350
000137540534  000101540736  000113541143  000174541337  000317541510  000475541623  000675541672  001102541672
001302541623  001460541510  001603541337  001664541143  001676540736  001640540534  001535540350  001374540216
001203540124  460000000000
000600541000  000700541156  001077541156  001177540777  001077540621  000700540621  000500541000  000555541173
000736541274  001137541246  001264541101  001264540676  001137540531  000736540503  000555540604  000400541000
000442541177  000600541335  001000541377  001177541335  001335541177  001377540777  001335540600  001177540442
000777540400  000600540442  000442540600  000300541000  000333541202  000452541355  000635541460  001041541475
001237541424  001402541273  001470541102  001470540675  001402540504  001237540353  001041540302  000635540317
000452540422  000333540575  000200541000  000227541203  000332541366  000500541514  000675541571  001102541571
001277541514  001445541366  001550541203  001577540777  001550540574  001445540411  001277540263  001102540206
000675540206  000500540263  000332540411  000227540574  000100541000  000124541203  000216541374  000350541535
000534541640  000736541676  001143541664  001337541603  001510541460  001623541302  001672541102  001672540675
001623540475  001510540317  001337540174  001143540113  000736540101  000534540137  000350540242  000216540403
000124540574  460000000000
001000541177  001156541077  001156540700  000777540600  000621540700  000621541077  001000541277  001173541222
001274541041  001246540640  001101540513  000676540513  000531540640  000503541041  000604541222  001000541377
001177541335  001335541177  001377540777  001335540600  001177540442  000777540400  000600540442  000442540600
000400541000  000442541177  000600541335  001000541477  001202541444  001355541325  001460541142  001475540736
001424540540  001273540375  001102540307  000675540307  000504540375  000353540540  000302540736  000317541142
000422541325  000575541444  001000541577  001203541550  001366541445  001514541277  001571541102  001571540675
001514540500  001366540332  001203540227  000777540200  000574540227  000411540332  000263540500  000206540675
000206541102  000263541277  000411541445  000574541550  001000541677  001203541653  001374541561  001535541427
001640541243  001676541041  001664540634  001603540440  001460540267  001302540154  001102540105  000675540105
000475540154  000317540267  000174540440  000113540634  000101541041  000137541243  000242541427  000403541561
000574541653  460000000000
```

OUTPUT OF DOT TEST PROGRAM

```
 ***    SCATRAN
C              TEST PROGRAM FOR LINE,VXAX,VYAX,PRINT,RITE2S@
  START        CALL SUBROUTINE()=OUT.(2)@
               CALL SUBROUTINE()=CAMRAV.(9)@
               CALL SUBROUTINE()=FRAMEV.(1)@
               CALL SUBROUTINE()=LINE.(-1.5,-1.5,1.5,1.5)@
               CALL SUBROUTINE()=VYAX.(.0,-1.6,1.6,2)@
               CALL SUBROUTINE()=LINE.(1.,.5,.0,.0)@
               CALL SUBROUTINE()=VXAX.(-.85,.85,.85,3)@
               CALL SUBROUTINE()=FRAMEV.(1)@
               CALL SUBROUTINE()=PRINT6.(-68,WORD,-1.3,1.5)@
               CALL SUBROUTINE()=FRAMEV.(1)@
               CALL SUBROUTINE()=CHSIZS.(15,9)@
               CALL SUBROUTINE()=RITSTS.(78,68,TABL1V.)@
               CALL SUBROUTINE()=RITE2S.(-1.3,1.5,1000,90,1,68,
                   1,WORD,IR)@
               CALL SUBROUTINE()=CLOSTP.()@
F WORD         NOW IS THE TIME FOR ALL GOOD MEN TO COME TO THE
                  AID OF THEIR
                COUNTRY@
       END     END PROGRAM(START)@
```

```
636000000000  600000236000  636000001777  600000037777  637777400000  601777236000  637777401777  601777037777
201642600000  040002004000  051200000000  201610600014  606060000102  606060000000  120000000000  776074777703
776173777604  776272777505  776371777406  776470777307  776567777210  776666777111  776765777012  777064776713
777163776614  777262776515  777361776416  777460776317  777557776220  653656652121  320777075741  320777075741
777454100551  777355076611  777256100650  777157100710  743060060750  302400000400  302400000400  302400000400
460000000000
636000000000  600000236000  636000001777  600000037777  637777400000  601777236000  637777401777  601777037777
200170451703  120000000000  224666603162  606330256063  314425602646  516021434360  274646246044  254560634660
234644256063  466063302560  213124604626  606330253151  602346644563  517012000000  460000000000
636000000000  600000236000  636000001777  600000037777  637777400000  601777236000  637777401777  601777037777
600132754127  770132554041  600226754127  600250710116  636250622052  674267600041  636325422041  600344510052
636344022116  674325200127  636267222127  674344044041  636366554041  636405710127  636424510063  636443754127
674641600127  600660754127  674641600041  637034222052  675015200041  636757022041  600740422052  636740422063
674757600074  637015422074  601034422105  637034022116  675015200127  636757222127  601232754127  771174600041
601312754127  771312600074  601406754127  771524200127  601430754127  771430600041  733430600074  600170754233
770132600145  674267600233  600306754233  674267600145  600366754233  674366466145  674424666200  600462554145
770600200233  600504754233  770504600145  732504600200  600740754233  770740600145  732740600200  601056710222
637056622156  675075600145  637133422145  601152510156  637152022222  675133200233  637075222233  675152044145
601174754233  733174600145  637251422145  601270422156  637270022167  733251200200  675232466200  675430754233
675466554145  675447600200  601546754233  771546600233  600132754337  770132600337  636462222262  674443200251
636405022251  600366510262  636366422326  674405600337  636443622337  600462644326  674462200304  600504710326
636504622262  674523600251  636561422251  600600510262  636600022326  674561200337  636523222337  674600044251
600622710326  636622622262  674641600251  636677422251  600716510262  636716022326  674677200337  636641222337
674716044251  600740754337  732740600251  637015422251  601034510262  637034022326  733015200337  601174754337
675174466251  675232666304  601270554251  771406200337  601312754337  771312600251  733312600304  601430754337
771430554251  601524754337  600170754443  770132600355  600250710432  636250622366  674267600355  636325422355
600344510366  636344022432  674325200443  636267222443  674344044355  636600222366  674561200355  636523022355
600504510366  636504422432  674523600443  636561622443  600622710432  636622622366  674641600355  636677422355
600716510366  636716022432  674677200443  636641222443  674716044355  600740754443  674740466355  674776666410
601034554355  771152200443  601056754443  771056600355  733056600410  601350754443  771312600355  601430710432
637430622366  675447600355
637505422355  601524510366  637524022432  675505200443  637447222443  675524044355  600170754547  770132600461
600250754547  770250600514  600344754547  770462200547  600366754547  770366600461  732366600514  674622754547
674660554461  674641600514  674757600547  600776754547  674757600461  601056754547  733056600461  637133422461
601152510472  637152022536  733133200547  601312710536  637312622472  675331600461  637367422461  601406510472
637406022536  675367200547  637331222547  675406044461  601430754547  771430600461  733430600514  600170754653
770132600565  600250754653  770250600620  600344754653  770462200653  600366754653  770366600565  732366600620
674523600653  600542754653  674523600565  600622754653  732622600565  636677422565  600716422576  636716022607
732677200620  674660466620  637152222576  675133200565  637075022565  601056510576  637056422642  675075600653
637133622653  601174710642  637174622576  675213600565  637251422565  601270510576  637270022642  675251200653
637213222653  675270044565  601312532565  637312422642  675331600653  637367622653  601406732642  601430754653
771430554565  601524754653  601604754653  771546600565  600132754757  732132600671  636207422671  600226422702
636226022713  732207200724  674170466724  674250466671  600306666757  674306666724
```

OUTPUT OF TEST OF LINE, VXAX, VYAX, PRINT, RITE2S

```
***     FAP
        COUNT    100
*       SC4020 TAPE DUMP
        CALL     DEFINE
        PZE      POOL
        PZE      =10
        PZE      =170
        CALL     ATTACH
        PZE      POOL
        PZE      =HFILE
        PZE      =1
REED    CALL     READ
        PZE      =HFILE
        PZE      OUT
        PZE      EOF
        PZE      OUT
        PZE      IOCL
        LXD      IOCL,4
        TXI      *+1,4,-1
        PXA      ,4
        ADD      BUFLOC
        STA      *+1
        CAL      **
        LAS      =0460000000000
        TTR      REED
        TTR      FRAME
        TTR      REED
FRAME   AXT      10,1
        CALL     WRSYS,FMT1
        STR      0
REPT    CALL     READ
        PZE      =HFILE
        PZE      OUT
        PZE      EOF
        PZE      OUT
        PZE      IOCL
        CALL     WRSYS,FMT
        LXD      IOCL,4
        TXI      *+1,4,-1
        PXA      ,4
        ADD      BUFLOC
        ALS      18
        STD      *+1
        STR      BUFF,,**
        STR      0
        TIX      REPT,1,1
        TTR      REED
OUT     CALL     CLOSRU
        PZE      =HFILE
        PZE      =0
        CALL     JOBOFF
```

```
 EOF    CALL      CLOSRU
        PZE       =HFILE
        PZE       =0
        CALL      ENDJOB
 IOCL   IORT      BUFF,,**
 BUFF   BSS       170
BUFLOC  PZE       BUFF
 POOL   BSS       1722
 FMT    BCI       3,(5T,8(2X,K12))
  FMT1  BCI       4,(//20T,Q*NEW FRAME*//)
        END
```

APPENDIX C
THE FORTRAN OPERATING SYSTEM

The SC 4020 subroutines that were used as a guide for this work were originally written for a FORTRAN II operating system. Since the subroutines had run in FORTRAN elsewhere, they were debugged at O.S.U. and made operational in that language. Since the SC 4020 FORTRAN extension exists and unlike the original subroutines, is self-contained, the details will be given here. Having operating FORTRAN subroutines could prove very useful in the event that a program was to be written and debugged at O.S.U., but the main runs were to be done elsewhere due to the computer time involved.

The FORTRAN subroutines are given in this appendix. In order to run a program, the following subroutines are needed:

```
CAMRAV - FORTRAN
(PLOT) - FORTRAN
CUTIV  - FORTRAN
MARKV
CONVRT - FORTRAN
```

This forms the basic package to produce SC 4020 tapes under the FORTRAN operating system.

CAMRAV - FORTRAN is almost identical to CAMRAV except that a call is made to MARKV through CUTIV to place cut marks on hardcopy. In SCATRAN, the subroutines CUTIV and MARKV have been separated and require their own calls.

(PLOT) - FORTRAN is an extremely complex subroutine that replaces the SCATRAN (PLOT)-TAPE-ID FRAME combination. One reason for its complexity is that it is two different subroutines written on top of each other. The first time (PLOT) is called, the non-system file is opened and the ID frame is written. The information used for the ID frame fills the space allocated for the output buffers (two 170 word buffers in this case). An entry is made to what will be a data word the next time (PLOT) is called, but when the subroutine is loaded it represents an operation to call in the ID information. The logic for filling in the blanks in the ID frame takes place within the space that is going to be used for the output buffers. The buffer area is left and a write command empties the buffer onto the non-system file tape creating the ID frame. The next time (PLOT) is called the space that was occupied by the ID frame is overlaid with the SC 4020 command words, word by word. When 170 words are reached or (CLEAN is called, the stored information is transferred to tape.

This subroutine also keeps track of the number of frames produced and controls whether or not a cut mark should be written by CUTIV and MARKV.

CONVRT is a small subroutine that serves to simulate the North American Aviation operating system as far as the SC 4020 subroutines are concerned. The ten entry points match the transfer vectors in the FORTRAN subroutines themselves. This subroutine also has provision for reading in the information required by the ID frame and CUTIV. It uses the following calls:

```
    READ INPUT,TAPE 5,1,N1,N2,N3,N4,N5,N6,N7,N8,N9
  1 FORMAT (9A6)
    CALL IDINFO(N1,N2,N3,N4,N5,N6,N7,N8,N9)
```

These calls should be made before CAMRAV is called. CAMRAV can be called without the above information calls and will merely result in an ID frame with blanks in the spaces for the programmer's name, etc. When these calls are used, the following data card is needed:

COLUMN	
1	Blank
2-24	Programmer's name
25-27	Dept.
28-30	Group
31-33	Blank
34-36	Box No.
37-41	Blank
42	Cut Mark Flag 0 = no cut mark 1 = normal cut mark 2 = cut mark for expanded operation Anything else or blank = same as zero
43-48	JOBNO (a dash will be placed between the numbers in 46 and 47. Use 0 through 9 only here)
49-54	DATE (the date should be written as six figures without separation. For example, 7/15/66 = b71566)
55-80	Blank

Figure 27 shows an ID frame that was made using these calls.

The call to FRAME empties the buffer of any remaining information, writes a record gap, and then adds the frame advance command followed by the corner marks and cut mark of the new frame. The placement of the advance frame command would have to be moved to the other side of the record gap if tapes made with these subroutines were to run on a plotter such as the BL 120 that requires a record gap after this command.

XSCALV and YSCALV have to be called explicitly so that the scaling factors can be set up. The programmer's manual covers these calls.[1]

A call to CLEAN will empty the buffer of the last frame and call in EFRAME to close the program. The main program ends with the statements

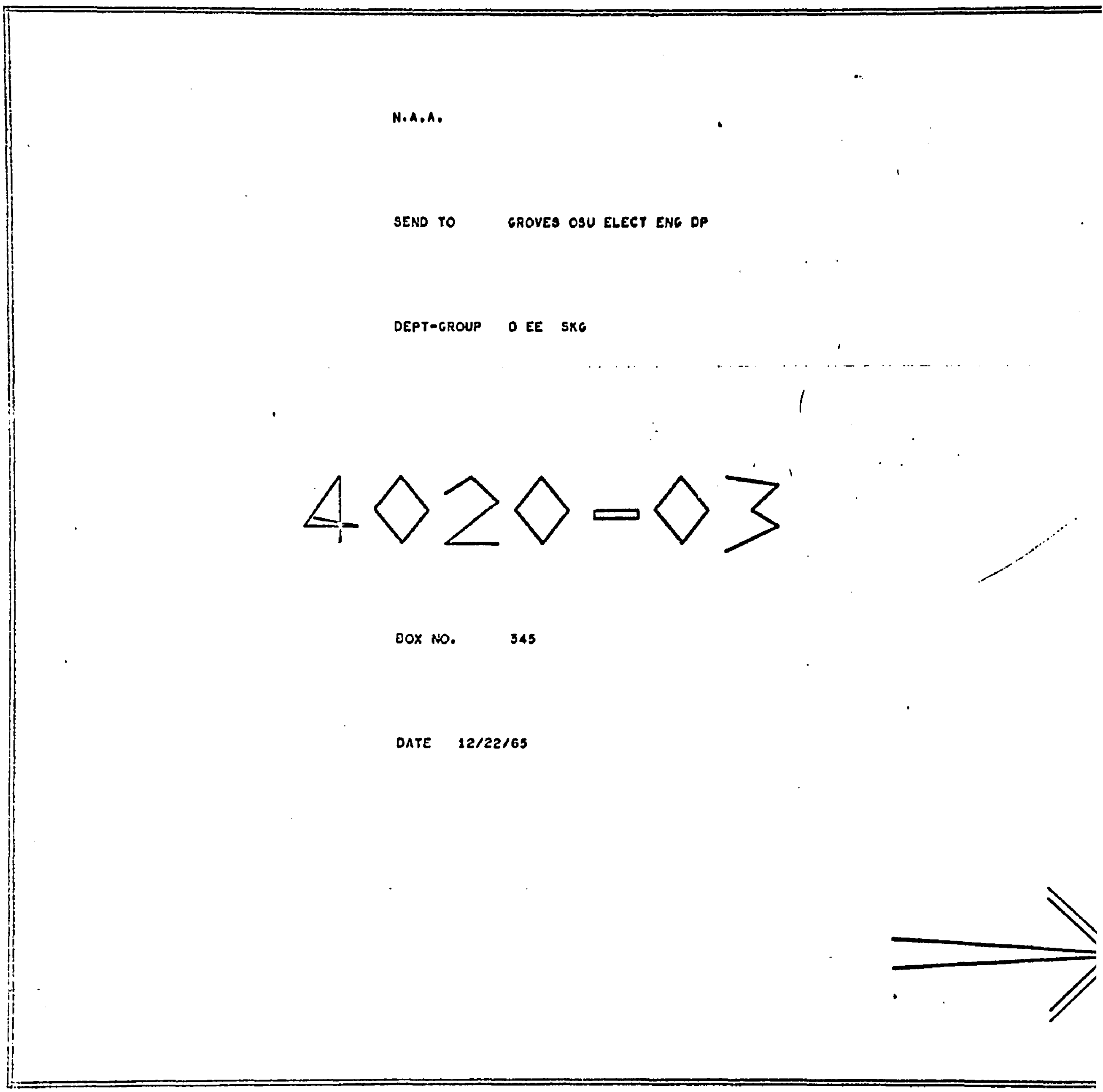
```
CALL TAPE,9
CALL EXIT or CALL DUMP
END
```

The CALL TAPE statement rewinds the non-system file tape and prints out a dismount message for the computer operator to return the tape with the program.[2]

---

[1] Programmers' Reference Manual SC 4020, loc. cit.

[2] If a format statement is used along with END FILE,9, REWIND as in the test program, be sure to include in the dismount message that logical tape 9 is physical tape unit A5. This will save operating time in locating the proper tape unit.



N.A.A.

SEND TO GROVES OSU ELECT ENG DP

DEPT-GROUP 0 EE SKG

4020-03

BOX NO. 345

DATE 12/22/65

FIGURE27

FORTRAN ID FRAME

There are a few subroutines that have the same name in both SCATRAN and FORTRAN but are different in the two systems. They are

APLOTV
EFRAME
CAMRAV
(PLOT)
CUTIV

Outside of these exceptions, any subroutine that ends in V will run under both systems without changes. Subroutines that end in anything but a V will run only under the SCATRAN system. Page 332 gives the statement listings for the dot test program written in FORTRAN that is equivalent to the SCATRAN main test program. This is followed by the above statement listings for the FORTRAN subroutines.

A few subroutines are written in FORTRAN for the purpose of creating a grid with titles and labels for graphs. Since they were useful mainly for engineering purposes and not for making movies, they are available in the SC 4020 library but they will operate only under the FORTRAN operating system and have not been checked out at OSU. Refer to the SC 4020 Programmers Manual for the following subroutines: BNBCDV, GRID1V, LABLV, NONLNV, and DXDYV.

```
C     DOT TEST PROGRAM FOR FORTRAN II
      DIMENSION X(350),Y(350)
      CALL CAMRAV (9)
      CALL XSCALV(-4.0,4.0,0.,0.)
      CALL YSCALV(-4.0,4.0,0.,0.)
      READ INPUT TAPE 5,25,RL,RI,THETA,SD,TP
      READ INPUT TAPE 5,30,NP,NC
   25 FORMAT (4F16.8)
   30 FORMAT (2I8)
      JL = 1
      DO 20 K=1,NP
      NUM = 0
      L = JL
      R = RL
      DO 15 I=1,NC
      FR = 6.283195*R
      N = FR/SD
      DN = N
      SF = FR/DN
      TI = SF/R
      JLN = JL+N-1
      DO 10 J = JL,JLN
      X(J) = R*COSF(THETA)
      Y(J) = R*SINF(THETA)
      THETA = THETA+TI
   10 CONTINUE
      JL = JLN+1
      R = R+RI
      NUM = NUM+N
   15 CONTINUE
      CALL FRAMEV
      CALL APLOTV(NUM,X(L),Y(L),1,1,1,44,IERR)
      THETA = THETA+TP
   20 CONTINUE
      PRINT 101
  101  FORMAT(1H0,32HDISMOUNT LOGICAL TAPE 9, UNIT A5)
      CALL EFRAME
      CALL EXIT
C     ( OR CALL DUMP)
       END
```

```
*       FAP
        COUNT    150
*       ARRAY PLOTTING FOR X,Y POINTS- FORTRAN
*      CALL APLOTV(N,X,Y,IX,IY,NC,CHAR,IERR)
        ENTRY    APLOTV
APLOTV  SXA      X4,4                     SAVE X REG
        SXA      X2,2                     X
        SXA      X1,1                     X
        TRA      A1                       TRA TO START
 EXIT   TSX      $SERREV,4                RESTORE ERROR
        TSX      SX                       CELLS
        TSX      SY
 X1     AXT      **,1                     RESTORE X REG
 X2     AXT      **,2                     X
 X4     AXT      **,4                     X
        TRA      9,4                      RETURN
 A1     CLA      2,4                      STORE BASE
        STA      AX                       ADDRESSES OF
        CLA      3,4                      X,Y,CHAR
        STA      AY                       ARRAYS
        CLA      7,4                      X
        STA      ACHAR                    X
        CLA      8,4                      SAVE IERR CELL
        STA      ERR                      ZERO IERR CELL
        STZ*     ERR                      X
        TSX      $SERSAV,4                SAVE PREVIOUS
        TSX      SX                       ERROR CELLS
        TSX      SY                       X
        TSX      $SCERRV,4                SET NEW ERROR
        TSX      XERR                     CELLS
        TSX      YERR                     X
        LXA      X4,4
        CLA*     4,4                      SAVE IX, IY
        STD      IX                       X
        CLA*     5,4                      X
        STD      IY                       X
        CLA*     1,4                      PICK UP N
        TZE      EXIT                     OUT IF ZERO
        TMI      A3                       FORWARD ARRAY
        LXD      IX,1                     POSITIVE VALUES
        LXD      IY,2                     FOR IX, IY
        SUB      1B17                     N-1
        TRA      A2
 A3     PDC      0,1                      COMPL. N
        PXD      0,1                      X
        LDC      IX,1                     COMPL. VALUES
        LDC      IY,2                     FOR IX, IY
        ACL      M1                       MAKE TXL
 A2     STD      A15                      STORE TEST DECR
        STD      A16                      X
        ACL      M2                       FILL IN OP
        STP     A15                     STORE OPS
        STP     A16                     X
        SXD     A11,1                   STORE INCR
        SXD     A12,2                   VALUES
        TXH     A4,1,0                  TEST FOR INCR BOTH
        TXH     A4,2,0                  ZERO
        TPL     A5                      YES. TEST OP
        AXC     1,1                     AND MAKE TEST
        TRA     A6                      DECR 1 OR -1
A5      AXT     1,1                     X
A6      SXD     A15,1                   X
A4      CLA*    6,4                     PICK UP NC
        TNZ     A7                      IF ZERO MAKE
        CLA     1B17                    IT 1
A7      TPL     A8                      BACKWARD ARRAY
        PDC     0,1                     COMP OF NC
        PXD     0,1                     X
        AXC     1,1                     -1 FOR INCR
        TRA     A9                      X
A8      SUB     1B17                    NC -1
        AXT     1,1                     +1 FOR INCR
        ACL     M1                      MAKE OP TXL
A9      ACL     M2                      SET IN OP
        STD     A14                     STORE TEST DECR
        SXD     A10,1                   STORE INCR VALUE
        STP     A14                     STORE OP
        AXT     0,1                     ZERO ALL XR
        AXT     0,2                     X
        AXT     0,4                     X
A13     STZ     P                       ZERO PLOT COMMAND
AX      CLA     **,1                    GET X
        STO     V
        SXA     A17,4
        TSX     $NXV,4                  SCALE X
        TSX     V
        ANA     M3                      X
        ORS     P                       STORE IN PLOT COM
AY      CLA     **,2                    GET Y
        STO     V
        TSX     $NYV,4                  SCALE Y
        TSX     V
        LXA     A17,4
        CHS                             X
        ADD     M3
        ANA     M3                      X
        ARS     18                      STORE IN PLOT
        ORS     P                       COMMAND
        CAL     XERR                    ACC SCALING
        STZ     XERR
        ORA     YERR                    ERRORS
        STZ     YERR
```

```
         TZE      ACHAR                      X
         ADD*     ERR
         STD*     ERR
         TRA      A10
ACHAR    CLA      **,4                       GET PLOTTING
         STA      T                          CHARACTER AND
         STP      T                          STORE IN PLOT
         ZET      T                          COMMAND
         ARS      12                         X
         ANA      M4                         X
         ARS      6                          X
         ORS      P                          X
         CLA      P                          X
         TSX      $(PLOT),4                  GO TO BUFFER
         LXA      A17,4
A10      TXI      *+1,4,**                   INCR CHAR
A14      TXL      A11,4,**                   OK
         AXT      0,4                        CYCLE CHAR
A11      TXI      *+1,1,**                   INCR X
A15      TXH      EXIT,1,**                  TEST FOR END
A12      TXI      *+1,2,**                   INCR Y
A16      TXH      EXIT,2,**                  TEST FOR END
         TRA      A13                        BACK FOR NEXT PT.
*                 CONSTANTS AND TEMPORARY CELLS
ERR      HTR      **                         L(1ERR)
SX       PZE      0,0,**                     PREVIOUS ERROR
SY       PZE      0,0,**                     CELLS
XERR     PZE      0,0,**                     NEW ERROR
YERR     PZE      0,0,**                     CELLS
IX       PZE      0,0,**                     IX
IY       PZE      0,0,**                     IY
1B17     PZE      0,0,1                      ONE IN DECR
M1       OCT      400000000000               MASKS FOR
M2       OCT      300000000000               TXH, TXL
P        HTR      **                         PLOT WORD
M3       OCT      001777000000               MASK FOR NX, NY
T        HTR      **                         TEMP FOR CHAR
M4       OCT      000077000000               MASK FOR CHAR
A17      HTR      **
V        HTR
         END
```

```
*       FAP
        COUNT   100
*       CLOSING ID FRAME
        ENTRY   EFRAME
EFRAME  SXA     (X4),4
        SXA     (X1),1
        CAL     FMADV
        CALL    (PLOT)
        CALL    (CLEAN
        AXT     WORDS-INSTAL,1
        CAL     WORDS,1
        TSX     $(PLOT),4
        TIX     *-2,1,1
        CALL    (CLEAN
        WEF     9
        WEF     9
        WEF     9
        RUN     9
 (X1)   AXT     **,1
 (X4)   AXT     **,4
        TRA     1,4
 FMADV  OCT     460000000000
*  TABLE OF VECTORS FOR (END) IN LARGE LETTERS
INSTAL  VFD     2/3,6/00,10/431,2/0,6/18,10/516
        VFD     2/3,6/00,10/422,2/0,6/63,10/480
        VFD     2/3,6/39,10/422,2/2,6/00,10/480
        VFD     2/3,6/39,10/422,2/2,6/00,10/543
        VFD     2/3,6/00,10/431,2/0,6/18,10/489
        VFD     2/3,6/22,10/431,2/2,6/00,10/507
        VFD     2/3,6/22,10/431,2/2,6/00,10/516
        VFD     2/3,6/30,10/431,2/2,6/00,10/489
        VFD     2/3,6/30,10/431,2/2,6/00,10/534
        VFD     2/3,6/00,10/461,2/0,6/09,10/480
        VFD     2/3,6/00,10/453,2/0,6/09,10/507
        VFD     2/3,6/00,10/461,2/0,6/09,10/534             E
                ND OF E     1
        VFD     2/3,6/00,10/480,2/0,6/63,10/480
        VFD     2/3,6/29,10/489,2/2,6/44,10/480
        VFD     2/3,6/00,10/518,2/0,6/44,10/480
        VFD     2/3,6/00,10/527,2/0,6/63,10/480
        VFD     2/3,6/09,10/480,2/2,6/00,10/480
        VFD     2/3,6/09,10/518,2/2,6/00,10/480
        VFD     2/3,6/00,10/489,2/0,6/44,10/499
        VFD     2/3,6/29,10/489,2/2,6/44,10/499
        VFD     2/3,6/09,10/480,2/2,6/00,10/543
        VFD     2/3,6/09,10/518,2/2,6/00,10/543             E
                ND OF N     2
        VFD     2/3,6/00,10/546,2/0,6/63,10/480
        VFD     2/3,6/29,10/546,2/2,6/00,10/480
        VFD     2/3,6/19,10/575,2/2,6/19,10/480
        VFD     2/3,6/00,10/594,2/0,6/25,10/499
        VFD      2/3•6/19•10/594•2/0•6/19•10/524
        VFD      2/3•6/29•10/575•2/0•6/00•10/543
        VFD      2/3•6/00•10/555•2/0•6/45•10/489
        VFD      2/3•6/15•10/555•2/2•6/00•10/489
        VFD      2/3•6/15•10/570•2/2•6/15•10/489
        VFD      2/3•6/00•10/585•2/0•6/15•10/504
        VFD      2/3•6/15•10/505•2/0•6/15•10/519
        VFD      2/3•6/15•10/555•2/2•6/00•10/534                E
                 ND OF D     3
*       TABLE OF VECTORS   ARROW INDICATING THIS JOB
        OCT      776030777604
        OCT      776030577604
        OCT      752042553604
        OCT      752042753604
        OCT      776046411606
        OCT      776046611602
        OCT      776145411612
        OCT      776145611576
        OCT      776244411616
        OCT      776244611572
*       TABLE OF AXES TO OUTLINE ID FRAME
        OCT      300000000000
        OCT      300000000004
        OCT      300000001773
        OCT      300000001777
        OCT      320000001777
        OCT      320004001777
        OCT      321773001777
        OCT      321777001777
        OCT      201122600042
        OCT      606330316260
        OCT      475146275121
        OCT      444431452760
        OCT      227060606060
        OCT      223360275146
        OCT      652562120000
  WORDS BES
        END
```

```
*       FAP
        COUNT   150
        ENTRY   CAMRAV
        ENTRY   FRAMEV
        ENTRY   RESETV
        ENTRY   FORMV
*  CRT   HARDWARE MANIPULATION ROUTINES
*       CALL     CAMRAV(N)
*         N      =9 FOR 9*9 CAMERA
*                =35 FOR 35 MM CAMERA
*                =OTHER FOR BOTH (ARG MAY BE DELETED)
*
**********************************************************************
                 ************
        SPACE
*       CALL     FRAMEV
*   ADVANCES  FILM FRAME
*
**********************************************************************
                 ************
        SPACE
*       CALL     RESETV
*   ADVANCES  FILM FRAME
*   LEAVES  TYPE MODE
*   SETS HARDWARE COUNTER TO (0,0)
*   SETS INTENSITY HIGH
*       CALL     FORMV
*   CAUSES FORM TO BE DISPLAYED
        SPACE
*       CALL FRAMEV(N) - IF (N) IS ZERO OR MISSING THE JOB NU
                 MBER
*       AND FRAME COUNT WILL BE DISPLAYED IN THE UPPER RIGHT
                 HAND
*       CORNER OF THE FRAME. IF (N) IS NON-ZERO THE PRINTING
                 WILL
*       BE SUPPERSSED. IF (N) IS NEGATIVE CORNER MARKS WILL B
                 E
*       SUPPRESSED.
*
M*********************************************************************
                 ************
        SPACE
FRAMEV  SXA      X4,4
 F0     CALL     (CLEAN                   FINISH PRECEDING RECORD
        CAL      FRAME
 F1     CALL     (PLOT)
        CALL     CUTRV
 F2     TRA      FIRST
        SXA      X1,1
        CLA      CAMERA
        CALL     (4020)
        LXA     X4,4
        CAL     1,4
        ANA     MASK1                LOC 777777700000
        SUB     TSX             LOC 007400000000
        STO     EXTEST               EXIT TEST
        TNZ     AXT
        CLA*    1,4
        TMI     NOCOR             NO CORNER MARKS
AXT     AXT     4,1
COR     CAL     CORNER,1                    PLOT CORNER DOTS
        CALL    (PLOT)
        CAL     CORNR2+1,1
        CALL    (PLOT)
        TIX     COR,1,1
        LXA     X4,4
NOCOR   ZET     EXTEST
        TRA     PRNT
        ZET*    1,4
        TRA     X1              DONT PRINT ID
PRNT    STZ     DBCD
        CAL*    $(CTJB)
        STA     *+1
        CLA     **
        STA     DBCD
        ARS     18
        STO     DBCD+1
        AXT     2,4
PRN1    AXT     18,1                 CONVERT FRAME COUNTS
        LDQ     DBCD+2,4
        STZ     DBCD+2,4
PRN2    PXD     0
        DVP     TENAD
        ALS     18,1
        ACL     DBCD+2,4
        SLW     DBCD+2,4
        TIX     PRN2,1,6
        ACL     BLANKS               LOC 606060000000
        SLW     DBCD+2,4
        TIX     PRN1,4,1
        AXT     0,1                  SET UP TO TYPE
PRN4    CAL     CORDJ,1
        CALL    (PLOT)
        TXI     *+1,1,1
        TXL     PRN4,1,6
X1      AXT     **,1
X4      AXT     **,4
        ZET     EXTEST
        TRA     1,4
        TRA     2,4
FIRST   CLA     NOOP                 DO ONCE
        STO     F2
        CLA*    $(JOBN)
        STA     *+1
        CLA     **                      JOB NO.
        LDQ     STOPT                   STOP TYPE 1200
        LGR     12
        ALS     6
        ACL     BDASH                   INSERT DASH
        LGL     6
        SLW     JOB
        STQ     JOB-1
        TRA     F2
*
RESETV  SXA     X4,4
        CAL     RESET
        TRA     F1
 FORMV  SXA     XX4,4                   DISPLAY FORMS
        CAL     FORM
        CALL    (PLOT)
        CALL    (CLEAN
 XX4    AXT     **,4
        TRA     1,4
*
        EJECT
CAMRAV  SXA     CX4,4
        CLA*    1,4
        PDX     ,4
        CAL     SBC
        LDQ     QSBC
        TXH     CX,4,35
        TXH     D35,4,34
        TXL     CX,4,8
        TXH     CX,4,9
        CAL     SC2
        LDQ     QSC2
        TRA     CX
 D35    CAL     SC1
        LDQ     QSC1
 CX     STQ     CAMERA
        CALL    (PLOT)
        CALL    (CLEAN
 CX4    AXT     **,4
        TRA     2,4
        VFD     2/3,6/15,10/0,2/0,6/0,10/0
        VFD     2/3,6/15,10/0,2/0,6/0,10/1023
        VFD     2/3,6/15,10/1023,2/2,6/0,10/0
        VFD     2/3,6/15,10/1023,2/2,6/0,10/1023
CORNER  VFD     2/3,6/0,10/0,2/1,6/15,10/0
        VFD     2/3,6/0,10/0,2/0,6/15,10/1023
        VFD     2/3,6/0,10/1023,2/1,6/15,10/0
CORNR2  VFD     2/3,6/0,10/1023,2/0,6/15,10/1023
 RESET  OCT     560000000000
 FRAME  OCT     460000000000
 FORM   OCT     500000000000
 SBC    OCT     430000000000
 SC1    OCT     410000000000
 SC2    OCT     420000000000
 QSBC           1**1
 QSC2           0**1                    9*9
 QSC1           1**0                    35 MM
CAMERA          0**1                   9X9 IF NOT SELECTED
 STOPT  OCT     120000000000
 DBCD   PZE     0
        PZE     0
 CORDF  OCT     201610600014             COORD FOR FRAMES
        PZE     0                       DASH NO AND STOP TYP
 JOB    PZE     0
 CORDJ  OCT     201642600000             COORD FOR   JOB. NO.
EXTEST  PZE     0                       EXIT TEST
 NOOP   NOP     0
 BDASH  OCT     000000000040
 MASK1  OCT     777777700000
BLANKS  OCT     606060000000
 TENAD  PZE     10                      10 BI. SCALE 35
 TSX    OCT     007400000000
        END
```

```
*       FAP
        PCC
        COUNT    300
        LBL      4020V000.L
        ENTRY    (PLOT)
        ENTRY    (CLEAN
        ENTRY    (4020)
        ENTRY    NOFRV
        ENTRY    (CUT
*  BUFFERED TRANSMISSION FOR CRT PLOTS
        REM      JAN. 8 CHANGE      170 WORD BUFFER
        REM                         (CUT HOLDS CUTTER FLAG
*       CALL     (PLOT)
*  STORES WORD IN (P,1-35) IN BUFFER AND CHECKS FOR FULL BU.
*       CALL     (CLEAN
*  EMPTIES BUFFER-USED TO FORCE EOR AFTER CERTAIN COMMANDS
*  AND AT END OF CHAIN OR JOB
**********************************************************
                 ***********
(PLOT)  SXA      (X4),4
 P1     TRA      FIRST                LXA POS ,4
 P2     SLW      (1),4
        TNX      CLEAN,4,1
        SXA      POS,4
 (X4)   AXT      **,4
        TRA      1,4
 CLEAN  SLW      ACC
        TXI      FSX,4,-1             ADJUST INDEX
(CLEAN  SXA      (X4),4
 POS    AXT      (SIZE),4
 FSX    SXD      F1,4
        CAL      CRTAPE
        CALL     (IOS)
 SIZE   AXT      (SIZE),4
        SXA      POS,4
 F1     TNX      (X4),4,**            BUFFER EMPTY. RETURN
        SXD      (IO),4               BUFFER WORD COUNT
        AXT      0,4
        CLA*     P2                   SWITCH BUFFERS
        STA      P2
        PDX      ,4
 FID    SXA      (IO),4
        XEC*     $(SDH)
        XEC*     $(WRS)
        AXC      (IO),4
        PXA      ,4
        STA*     $(WTC)
        XEC*     $(RCH)
        CAL      TES
        SLW*     $(TES)
        CAL      ACC
 FEND   TRA      (X4)
*
 TES    CALL     (WER)
 CTES   TSX      (CLEAN,4
 (IO)   IOCD     **,,**
CPTAPE           16,,9
(SIZE) EQU      170
 ACC
(4020) ADD*     CTJ                     UPDATE FRAME COUNT
        STO*     CTJ
        TRA      1,4
*  CUTTER FLAG SET BY ACCT DATA.  CUTIV CAN OVERRIDE
(CUT    PZE      0,0,1                  PROP.70 PRIME.AS AMENDED
                 2/2/65
 NOFRV  CLA*    $(CTJB)
        STA      *+1
 CTJ    CAL      **
        STZ*     1,4               CLEAR
        STD*     1,4                    9*9 FRAMES
        ALS      18
        STO*     2,4                    35 MM FRAMES
        TRA      3,4
 (B1)   BSS
* THE FOLLOWING CODE IS USED ONLY AT THE BEGINNING
* OF THE JOB, TO CREATE AN ID FRAME AND SET UP THE
* FORTRAN MONITOR TO CLEAN UP AT THE END OF EACH CHAIN LINK.
 FIRST  SLW      *
        LDQ      LXA
        STQ      P1                     PREVENT REENTRY
        CLA*     $(IDCD)
        STA      A
*    PROCESS CUTTER MARK FLAG
        AXC      5,1
        CAL*     A
        ANA      MASK1
        PAX      0,2               CUTTER MARK FLAG
        TXH      *+2,2,2                PROP,70 PRIME, AS AMENDED
                  2/2/65
        SXD      (CUT,2
        LDQ      CTES
        STQ*    $(CTES)
        CLA*    $(CTJB)                 LOC OF FRAME COUNT
        STA      CTJ
        NZT*     CTJ                    CHECK FRAME COUNT
* ID FRAME WAS WRITTEN BY PREVIOUS CHAIN LINK, IF NON-ZERO
        TSX      (IDFRM,4               WRITE ID FRAME
 LXA    LXA      POS,4
        CAL      FIRST
        TRA      P1
*    TABLE OF DIGITS FOR JOB NUMBER   .........
 TABO   VFD      2/3,6/24,10/8,2/3,6/32,10/32
```

```
         VFD      2/3,6/24,10/32,2/2,6/32,10/0
         VFD      2/3,6/24,10/56,2/0,6/32,10/32
         VFD      2/3,6/24,10/32,2/1,6/32,10/64
TAB1     VFD      2/3,6/16,10/16,2/3,6/16,10/16
         VFD      2/3,6/0,10/32,2/0,6/62,10/0
         VFD      2/3,6/16,10/32,2/0,6/2,10/62
         VFD      2/3,6/32,10/16,2/2,6/0,10/64
TAB2     VFD      2/3,6/24,10/8,2/3,6/16,10/16
         VFD      2/3,6/24,10/32,2/2,6/24,10/0
         VFD      2/3,6/48,10/56,2/0,6/40,10/24
         VFD      2/3,6/48,10/8,2/2,6/0,10/64
TAB3     VFD      2/3,6/48,10/8,2/2,6/8,10/0
         VFD      2/3,6/24,10/56,2/0,6/24,10/8
         VFD      2/3,6/24,10/32,2/2,6/16,10/32
         VFD      2/3,6/48,10/56,2/0,6/24,10/48
TAB4     VFD      2/3,6/0,10/40,2/1,6/63,10/64
         VFD      2/3,6/32,10/40,2/0,6/48,10/0
         VFD      2/3,6/48,10/8,2/2,6/0,10/48
         VFD      2/3,6/40,10/56,2/1,6/8,10/48
TAB5     VFD      2/3,6/40,10/48,2/0,6/0,10/0
         VFD      2/3,6/8,10/8,2/0,6/24,10/0
         VFD      2/3,6/40,10/0,2/2,6/28,10/24
         VFD      2/3,6/40,10/40,2/0,6/28,10/52
TAB6     VFD      2/3,6/32,10/40,2/0,6/40,10/0
         VFD      2/3,6/20,10/8,2/2,6/24,10/40
         VFD      2/3,6/28,10/28,2/3,6/20,10/64
         VFD      2/3,6/36,10/56,2/1,6/16,10/44
TAB7     VFD      2/3,6/48,10/8,2/2,6/0,10/0
         VFD      2/3,6/24,10/56,2/0,6/63,10/0
         VFD      2/3,6/16,10/32,2/3,6/56,10/64
         VFD      2/3,6/40,10/48,2/1,6/8,10/8
TAB8     VFD      2/3,6/48,10/8,2/2,6/0,10/0
         VFD      2/3,6/48,10/56,2/0,6/63,10/0
         VFD      2/3,6/48,10/8,2/2,6/0,10/64
         VFD      2/3,6/48,10/55,2/1,6/63,10/64
TAB9     VFD      2/3,6/27,10/35,2/1,6/23,10/43
         VFD      2/3,6/24,10/8,2/3,6/20,10/20
         VFD      2/3,6/24,10/32,2/2,6/20,10/0
         VFD      2/3,6/40,10/56,2/0,6/44,10/20
DASH     VFD      2/3,6/40,10/16,2/3,6/0,10/32
         VFD      2/3,6/0,10/56,2/2,6/8,10/32
         VFD      2/3,6/40,10/56,2/0,6/0,10/40
         VFD      2/3,6/0,10/16,2/1,6/8,10/40
*   TABLE OF AXES TO OUTLINE ID FRAME
         VFD      06/30,30/0
         VFD      06/30,30/4
         VFD      06/30,30/1019
         VFD      06/30,30/1023
         VFD      06/32,12/0,18/1023
         VFD      06/32,12/4,18/1023
         VFD      06/32,12/1019,18/1023
```

```
       VFD      06/32,12/1023,18/1023
 AXIS  BES
*    JOB ID BLOCK
INSTAL VFD      06/20,12/350,H6/N,12/100
       VFD      H30/.A.A..06/12
       VFD      06/20,12/350,H6/S,12/200
       VFD      H18/END,06/60,H12/TO
       VFD      H36/
 PROG  VFD      36/0,36/0,36/0,36/0
       VFD      06/20,12/350,H6/D,12/300
       VFD      H54/EPT-GROUP,H18/
 DEPGR VFD      36/0,18/0,018/120000
       VFD      06/20,12/350,H6/B,12/600
       VFD      H12/OX,06/60,H18/NO.,H36/
 BOXNO VFD      30/0,06/12
       VFD      06/20,12/350,H6/D,12/700
       VFD      H18/ATE,H18/
 DATE  VFD      48/0,024/12000000
*  TABLE OF VECTORS .... ARROW INDICATING THIS JOB
       VFD      2/3,6/63,10/1000,2/1,6/63,10/900
       VFD      2/3,6/63,10/1000,2/0,6/63,10/900
       VFD      2/3,6/53,10/990,2/1,6/53,10/900
       VFD      2/3,6/53,10/990,2/0,6/53,10/900
       VFD      2/3,6/63,10/986,2/0,6/4,10/902
       VFD      2/3,6/63,10/986,2/1,6/4,10/898
       VFD      2/3,6/63,10/923,2/0,6/4,10/906
       VFD      2/3,6/63,10/923,2/1,6/4,10/894
       VFD      2/3,6/63,10/860,2/0,6/4,10/910
       VFD      2/3,6/63,10/860,2/1,6/4,10/890
 WORDS BCS
 RST   VFD      06/56,30/0
 A              **,1
 TEM
OFFSET          444,,256
       BSS      (SIZE)+(B1)-*        PAD TO COMPLETE BUFFER BL
                OCK
 (1)            (2),,(1)-(SIZE)
 (B2)  BSS
(IDFRM SXA      ID4,4
       SXA      ID2,2
       SXA      ID1,1
       CLA*     $(TCRT)           L(BATCH FRAME COUNT)
       STA      *+1
       ZET      **                ANY PRIOR FRAMES
       TRA      *+4               YES, DO NOT REWIND
       CAL      CRTAPE            NO, INSURE THAT
       CALL     (IOS)             CRT OUTPUT TAPE
       XEC*     $(REW)            IS REWOUND
       CAL      STOPR            STOP TYPE , INSURANCE
       TSX      BUFR,4
       CAL      BOTHCM
```

```
        TSX    BUFR,4
        CAL    IMAGE                        NON-EXPANDED IMAGE
        TSX    BUFR,4
        CAL    RST                     NEW FILM FRAME, AND
        TSX    BUFR,4                  STOP TYPE AND SET BRITE
        CLA*   $(JOBN)
        STA    *+1
 JBNO   LDQ    **                      FETCH JOB NO.
* CONVERT JOB NUMBER AND PRINT IN LARGE LETTERS
* WITH AXES INSERTED BETWEEN LETTERS.
        AXT    7,2
 (TAB)  BSS
 I.1    PXD    TAB1
        LGL    6                       FETCH NEXT CHARACTER
 I.2    STQ    TEM
        CAS    ABLNK                  LOC 000000000060
        TRA    I.34
        TRA    I.34
        ALS    2                       DIGIT *4, PLUS
        ACL    (TAB)                   TABLE ORIGIN
        STA    I.3                     =TABLE ENTRY
        AXT    4,1
* LOOK UP VECTOR COMMANDS FOR THIS DIGIT
 I.3    CAL    **,1                    VECTOR COMMAND
        ACL    OFFSET                  INCREMENT X-ORIGIN
        TSX    BUFR,4                  TRANSMIT VECTOR
        TIX    I.3,1,1                 COMPLETE DIGIT
 I.34   CAL    AXIS,2                  OUTPUT AXIS
        TSX    BUFR,4
        CAL    OFFSET
        ACL    S64B17               LOC 64B17
        SLW    OFFSET
        SPACE
        LDQ    TEM
        TNX    I.4,2,1
        TXH    I.1,2,3
        TXL    I.1,2,2
        CLA    ATEN                 TEN IN ADDR.
        TRA    I.2
*  FETCH NAME, ETC.
 I.4    AXC    0,1
        LDQ    AONE
        CAL*   A                       DEPT-GRP
        LGR    18                      DEPT IN AC,GRP IN MQ
        XCL
        ORS    DEPGR+1
        AXC    1,1
        CAL*   A
        ARS    30
        RQL    18
        LGL    30
        ORA     A6060
        SLW     DEPGR                   END OF DEPT-GRP
        CAL*    A
        AXT     3,2                     THREE WORDS OF NAME
A1      TXI     *+1,1,-1                MOVE TO NEXT WORD
        LDQ*    A
        LGL     6
        SLW     PROG+3,2
        LGL     30
        TIX     A1,2,1
        ALS     6
        ORA     ATEN               STOP TYPE OCT 12
        SLW     PROG+3                  END OF PROG. NAME
        AXC     6,1
        CAL*    A                       BOX NO
        LDQ     BLKSTP                 LOC  606012
        LGL     18
        SLW     BOXNO                   NOTE MQ IS CLEAR
        AXT     2,1                     DATE
        CAL*    A
        LGR     24
        ALS     6
        ORA     OCT61
        LGL     12
        ALS     6
        ORA     OCT61
        SLW     DATE
        XCL
        ORS     DATE+1
*   OUTPUT TYPED NAME, DEPT-GRP,BOX NO., DATE.
        AXT     WORDS-INSTAL,1
        CAL     WORDS,1
        TSX     BUFR,4
        TIX     *-2,1,1
*   OUTPUT EIGHTH AXIS TO END ID FRAME
        CAL     AXIS-8
        TSX     BUFR,4
        CAL     CAMSTD                    STANDARD CAMERS
        TSX     BUFR,4
        TSX     KLENE,4
 ID1    AXT     **,1
 ID2    AXT     **,2
 ID4    AXT     **,4
        TRA     1,4
 BUFR   SXA     BX4,4
        LXA     B1,4
        SLW     0,4
        TNX     BCL,4,1
        SXA     B1,4
 BX4    AXT     **,4
        TRA     1,4
```

```
RCL     SLW     ACC
        TXI     BSX,4,-1
KLENE   SXA     BX4,4
B1      AXT     (SIZE),4
BSX     SXD     BF1,4
        AXT     (SIZE),4
        SXA     B1,4
BF1     TNX     BX4,4,**
        SXD     (IO),4
        AXT     BEND,4
        SXA     FEND,4
        CAL     CRTAPE
        CALL    (IOS)
        AXC     (SIZE),4
        TRA     FID
 BEND   AXT     (X4),4
        SXA     FEND,4
        AXC     *+1,4
        XEC*   $(TCO)
        TRA     BX4
 STOPR  OCT     120000000000             STOP TYPE
BOTHCM  OCT     430000000000          BOTH  CAMERAS
CAMSTD  OCT     420000000000        STANDARD CAMERAS
IMAGE   OCT     450000000000         STANDARD IMAGE
 AONE   PZE     1
 ATEN   PZE     10
 ABLNK  OCT     000000000060
 A6060  OCT     000000006060
BLKSTP  OCT     606012000000        2 BLANKS AND STOP
 OCT61  OCT     000000000061
S44B17  OCT     000100000000
 MASK1  OCT     000000000077
        BSS     (SIZE)+(B2)-*        PAD TO COMPLETE BUFFER BL
                OCK
 (2)            (1),,(2)-(SIZE)
        END
```

```
*        FAP
         COUNT    100
*                 CUT MARKS
         ENTRY    CUTIV
         ENTRY    CUTRV
         REM      **************************************************
                  ************
         REM      *
         REM      * CALL CUTIV (N) FOR CUT MODE
         REM      * SETS INDICATOR FOR CUTTER MARK.
         REM      * IT IS SENSED WHEN FRAMEV CALLS CUTRV
         REM      * N=0. FOR NO CUT MARK
         REM      * N=1 FOR CUT MARK SUITABLE FOR 11X
         REM      * N=2 FOR CUT MARK SUITABLE FOR 15X
         REM      * WHEN USING FORM SLIDE
         REM      *
         REM      * TESTS INDICATOR AND PUTS ON INDICATED
         REM      * CUT MARK
         REM      *
         REM      **************************************************
                  ************
 CUTIV   SXA      SAV4,4
         CAL      NULL
         TSX      $(PLOT),4
  SAV4   AXT      **,4
         CLA*     1,4
         STO*     $(CUT
         TRA      2,4
 CUTRV   CLA*     $(CUT
         TZE      1,4
         SXA      X4,4
         SXA      X1,1
         SUB      N1
         TNZ      CX15                  TRA TO 15X CUT
 CX11    CLA      N1023
         TRA      *+2
 CX15    CLA      N710
         STO      IX
         AXT      6,1
 MRK     TSX      $MARKV,4
         TSX      IX                    IX
         TSX      IY                    IY
         TSX      LX                    LENGTH IN X
         TSX      LY                    LENGTH IN Y
         TSX      INT                   INTENSITY
         CLA      IX
         SUB      N1
         STO      IX
         TIX      MRK,1,1
 X4      AXT      **,4
 X1      AXT      **,1
```

```
        TRA     1,4
NI      PZE     0,0,1
N1023   PZE     0,0,1023
N710    PZE     0,0,710
IX      PZE     0,0,**
IY      PZE     0,0,1023
LX      PZE     0,0,0
LY      PZE     0,0,63
INT     PZE     0,0,2
NULL    OCT     001000601000
        END
```

```
*       FAP
        COUNT   50
        LBL     NAOSU000.L
*    OSU NAA CONVERSION SUBROUTINE FOR FORTRAN ONLY.........
        ENTRY   IDINFO
        ENTRY   CLEAN
        ENTRY   (SDH)
        ENTRY   (TES)
        ENTRY   (CTES)
        ENTRY   (JOBN)
        ENTRY   (IDCD)
        ENTRY   (CTJB)
        ENTRY   (TCRT)
        ENTRY   (CNTV)
*
*   THIS PROGRAM TIES UP THE LOOSE ENDS THAT WOULD
*   NORMALLY BE HANDLED BY THE N. A. A. BOOKKEEPING ROUTINES
*
IDINFO CLA*     5.4             BRING IN DEPT
       STO      NAME
       CLA*     1.4             BRING IN NAME
       STO      NAME+1
       CLA*     2.4
       STO      NAME+2
       CLA*     3.4
       STO      NAME+3
       CLA*     4.4
       STO      NAME+4
       CLA*     7.4             BRING IN  CUT MARK FLAG
       STO      NAME+5
       CLA*     6.4             BRING IN BOX NO.
       STO      NAME+6
       CLA*     8.4             BRING IN JOB NO.
       STO      NAME-3
       CLA*     9.4             BRING IN DATE
       STO      NAME-2
       TRA      10.4            TRANSFER BACK TO MAIN PROGRAM
 (SDH) NOP
 (TES) NOP
(CTES) NOP
(CTJB) PZE      *+1
       PZE
(TCRT) PZE      *+1
       PZE
(CNTV) TRA      2.4
*   NAME.JOB NUMBER. DEPT-GROUP. BÕX NO.. DATE
(JOBN) PZE      *+1
       BCI      2.
(IDCD) PZE      *+1
 NAME  BCI      7.
*   USED TO MAKE CLEAN IDENTICAL TO (CLEAN
CLEAN   SXA     (X4),4          STORE IR4 IN (X4)
        TSX     $(CLEAN,4       CALL (CLEAN
 (X4)   AXT     **,4
        TRA     1,4             RETURN TO CALLING PROGRAM
        END
```

APPENDIX D
MOVIE STATEMENT LISTINGS

This appendix contains the statement listings for three programs.

The first one is for the statements used to produce the spheroidal radiator movie up to the small ellipse with the generator symbol inside.

The second one is the basic program for the spheroidal radiator pattern. It was used with various additions for the rest of the movie.

The third one lists the statements that were used to produce "THE END" in growing block letters at the end of the movie.

The @ symbol at the end of each line represents the SCATRAN end symbol on the particular printer used for these printouts.

```
 ***    RUN, DUMP LOWER CORE. SCATRAN
C              THIS PROGRAM PRODUCED FILE 1. TAPE 1. TITLES. CR
                  EDITS. ETC.
  START        CALL SUBROUTINE()=CAMRAV.(40)@
               CALL SUBROUTINE()=FRAMEV.(-1)@
               CALL SUBROUTINE()=CHSIZS.(10.10)@
               CALL SUBROUTINE()=RITSTS.(53.128.TABL1V.)@
               CALL SUBROUTINE()=RITE2S.(-.17598..6375.600.90.1
               .3.-1.WORD1.IRE)@
F WORD1        THE@
               CALL SUBROUTINE()=RITE2S.(-.792..2125.790.90.1.2
               9.-1.WORD2.IRE)@
F WORD2         RESONANT SPHEROIDAL RADIATOR@
               CALL SUBROUTINE()=STORE.(X.4)@
               DO THROUGH(L101).N=0.1.N.L.167@
               CALL SUBROUTINE()=STORE.(X.3)@
  L101         CONTINUE@
               CALL SUBROUTINE()=STORE.(X.1)@
               CALL SUBROUTINE()=RITE2S.(-1.3181..1328.945.90.1
               .16.-1.WORD5.IRE)@
F WORD5        BRENTON R.GROVES@
               CALL SUBROUTINE()=CHSIZS.(7.7)@
               CALL SUBROUTINE()=RITSTS.(33.64.TABL1V.)@
               CALL SUBROUTINE()=RITE2S.(.0..9928.540.90.1.1.-1
               .WORD3.IRE)@
F WORD3        A@
               CALL SUBROUTINE()=RITE2S.(-1.0724..7803.860.90.1
               .31.-1.WORD4.IRE)@
F WORD4        COMPUTER GENERATED     MOVIE BY@
               CALL SUBROUTINE()=STORE.(X.4)@
               DO THROUGH(L102).N=0.1.N.L.119@
               CALL SUBROUTINE()=STORE.(X.3)@
  L102         CONTINUE@
               CALL SUBROUTINE()=STORE.(X.2)@
               CALL SUBROUTINE()=CHSIZS.(5.5)@
               CALL SUBROUTINE()=RITSTS.(25.64.TABL1V.)@
               CALL SUBROUTINE()=RITE2S.(-1.4875.-.2656.600.90.
               1.21.-1.WORD6.IRE)@
F WORD6        HELPFUL ASSISTANCE BY@
               CALL SUBROUTINE()=STORE.(X.4)@
               DO THROUGH(L103).N=0.1.N.L.96@
               CALL SUBROUTINE()=STORE.(X.3)@
  L103         CONTINUE@
               CALL SUBROUTINE()=STORE.(X.1)@
               DO THROUGH(L104).N=1.1.N.L.56@
               DO THROUGH(L104).M=0.1.M.L.4@
               CALL SUBROUTINE()=CHSIZS.(10.10)@
               CALL SUBROUTINE()=RITSTS.(53.128.TABL1V.)@
               CALL SUBROUTINE()=RITE2S.(-1.3181..1328.945.90.1
               .16.-1.WORD5.IRE)@
               CALL SUBROUTINE()=CHSIZS.(7.7)@
```

```
              CALL SUBROUTINE()=RITSTS.(38.64.TABLIV.)@
              CALL SUBROUTINE()=RITE2S.(.0..9928.540.90.1.1.-1
              .WORD3.IRE)@
              CALL SUBROUTINE()=RITE2S.(-1.0724..7803.860.90.1
              .31.-1.WORD4.IRE)@
              CALL SUBROUTINE()=RITE2S.(-1.1422.-.568.880.90.1
              .N.-1.WORD7.IRE)@
F WORD7       JOHN D.COWAN JR.     EDWARD M.KENNAUGH    DAVID
                 M.SHUPE@
              CALL SUBROUTINE()=CHSIZS.(5.5)@
              CALL SUBROUTINE()=RITSTS.(25.64.TABLIV.)@
              CALL SUBROUTINE()=RITE2S.(-1.4875.-.2656.600.90.
              1.21.-1.WORD6.IRE)@
              CALL SUBROUTINE()=FRAMEV.(-1)@
  L104        CONTINUE@
              CALL SUBROUTINE()=RITE2S.(-.083..9031.580.90.1.3
              .-1.WORD8.IRE)@
F WORD8       AND@
              CALL SUBROUTINE()=RITE2S.(-1.1208..5844.870.90.1
              .56.-1.WORD9.IRE)@
F WORD9          NATIONAL COMMITTEE FOR   ELECTRICAL ENGINEERI
                 NG FILMS@
              CALL SUBROUTINE()=RITE2S.(-.9961.-.3719.830.90.1
              .25.-1.WORD10.IRE)@
F WORD10      THE OHIO STATE UNIVERSITY@
              CALL SUBROUTINE()=RITE2S.(-1.2883.-.5844.920.90.
              1.32.-1.WORD11.IRE)@
F WORD11      NUMERICAL COMPUTATION LABORATORY@
              CALL SUBROUTINE()=STORE.(X.4)@
              DO THROUGH(L105).N=0.1.N.L.150@
              CALL SUBROUTINE()=STORE.(X.3)@
  L105        CONTINUE@
              CALL SUBROUTINE()=STORE.(X.1)@
              CALL SUBROUTINE()=RITE2S.(-1.2036..5844.895.90.1
              .83.-1.WORD12.IRE)@
F WORD12      (1) THE STREAMLINES REPRESENT     VALUES OF CONS
                 TANT  H  IN
                  THE RADIATED FIELD.@
              CALL SUBROUTINE()=RITE2S.(-1.2036.-.1594.895.90.
              1.72.-1.WORD13.IRE)@
F WORD13      (2) THE MOVING SEGMENTS IN THE    STREAMLINES RE
                 PRESENT THE
                  D FIELD.@
              CALL SUBROUTINE()=STORE.(X.4)@
              DO THROUGH(L106).N=0.1.N.L.240@
              CALL SUBROUTINE()=STORE.(X.3)@
  L106        CONTINUE@
              CALL SUBROUTINE()=STORE.(X.1)@
              CALL SUBROUTINE()=CHSIZS.(10.10)@
              CALL SUBROUTINE()=RITSTS.(100.100.TABLIV.)@
              CALL SUBROUTINE()=RITE2S.(.0..70875.700.90.4.1.-
               1,W14,IRE)@
               CALL SUBROUTINE()=RITE2S.(.0,-.70875,700,90,4,1,
                1,W14,IRE)@
F W14          *@
               CALL SUBROUTINE()=STORE.(X,4)@
               DO THROUGH(L107),N=0,1,N.L.71@
               CALL SUBROUTINE()=STORE.(X,3)@
  L107         CONTINUE@
               CALL SUBROUTINE()=STORE.(X,1)@
               DIST=.02217@
               DX=.0023@
               X1=.47013@
               X=.47013@
               DIMENSION(Q(500),R(500),S(200),T(200))@
               Q(0)=.47013@
               R(0)=0.@
               DO THROUGH(L1),N=1,1,N.L.48@
  L3           X=X1-DX@
               D1=103.090909091*X*X@
               Y=.8505*SQRT.(1.-D1)@
               D=SQRT.(DX*DX+(Y-Y1)*(Y-Y1))@
               TRANSFER(L2)PROVIDED(.ABS.(D-DIST).L..0001)@
               DX=DX*DIST/D@
               TRANSFER TO (L3)@
  L2           Q(N)=X@
               R(N)=Y@
               X1=X@
  L1           Y1=Y@
               Q(48)=0.@
               R(48)=.8505@
               DO THROUGH (L4), M=1,1,M.LE.N@
               Q(M+N)=-Q(N-M)@
  L4           R(N+M)=R(N-M)@
               N=2*N@
               DO THROUGH (L5),M=1,1,M.LE.N@
               Q(N+M)=Q(N-M)@
  L5           R(N+M)=-R(N-M)@
               DO THROUGH(L110),M=0,1,M.LE.192@
               CALL SUBROUTINE()=RITE2S.(.0,.70875,700,90,4,1,-
               1,W14,IRE)@
               CALL SUBROUTINE()=RITE2S.(.0,-.70875,700,90,4,1,
                1,W14,IRE)@
               CALL SUBROUTINE()=LINE.(.0,.70875,.0,-.70875)@
               CALL SUBROUTINE()=LINE.(.0,.70875,Q(M),R(M))@
               CALL SUBROUTINE()=LINE.(.0,-.70875,Q(M),R(M))@
               DO THROUGH(L109),N=1,1,N.LE.M@
               CALL SUBROUTINE()=LINE.(Q(N),R(N),Q(N-1),R(N-1))
                  @
  L109         CONTINUE@
               CALL SUBROUTINE()=FRAMEV.(-1)@
  L110         CONTINUE@
```

```
          DO THROUGH(L111),N=1,1,N.LE.192@
          CALL SUBROUTINE()=LINE.(Q(N),R(N),Q(N-1),R(N-1))
             @
L111      CONTINUE@
          CALL SUBROUTINE()=RITE2S.(.0,.70875,700,90,4,1,-
          1,W14,IRE)@
          CALL SUBROUTINE()=RITE2S.(.0,-.70875,700,90,4,1,
           1,W14,IRE)@
          CALL SUBROUTINE()=STORE.(X,4)@
          DO THROUGH(L112),N=1,1,N.L.48@
          CALL SUBROUTINE()=STORE.(X,3)@
L112      CONTINUE@
          CALL SUBROUTINE()=STORE.(X,1)@
          DO THROUGH(L115),DV=.0,.174533,DV.L.25.14@
          COSDV=COS.(DV)@
          DO THROUGH(L113),N=0,1,N.LE.192@
          S(N) = COSDV*Q(N)@
L113      CONTINUE@
          DO THROUGH(L114),N=1,1,N.LE.192@
          CALL SUBROUTINE()=LINE.(S(N),R(N),S(N-1),R(N-1))
             @
L114      CONTINUE@
          CALL SUBROUTINE()=FRAMEV.(-1)@
L115      CONTINUE@
          DO THROUGH(L116),DL=1.,-.013465,DL.GE..352734@
          S(0)=Q(0)*DL@
          T(0)=R(0)*DL@
          DO THROUGH(L117),N=1,1,N.LE.192@
          S(N)=Q(N)*DL@
          T(N)=R(N)*DL@
          CALL SUBROUTINE()=LINE.(S(N),T(N),S(N-1),T(N-1))
             @
L117      CONTINUE@
          CALL SUBROUTINE()=FRAMEV.(-1)@
L116      CONTINUE@
          DO THROUGH(L118),M=0,1,M.LE.96@
          N=2*M@
          Q(M)=S(N)@
          R(M)=T(N)@
L118      CONTINUE@
          DO THROUGH(L119),N=1,1,N.LE.96@
          CALL SUBROUTINE()=LINE.(Q(N),R(N),Q(N-1),R(N-1))
             @
L119      CONTINUE@
          CALL SUBROUTINE()=STORE.(X,4)@
          DO THROUGH(L120),M=0,1,M.L.71@
          CALL SUBROUTINE()=STORE.(X,3)@
L120      CONTINUE@
          CALL SUBROUTINE()=STORE.(X,1)@
          N=0@
          DU=.001@
```

```
             DIST=.00664@
             ERROR=.0015@
             U1=0.@
             X1=.1658@
             Y1=0.@
             Q(N)=X1@
             R(N)=Y1@
             N=N+1@
L304         U=U1+DU@
             TRANSFER(L303)PROVIDED(U.G.1.)@
L301         X=.25*SQRT.(.44*(1.-U*U))@
             Y=.3*U@
             D=SQRT.((X-X1)*(X-X1)+(Y-Y1)*(Y-Y1))@
             TRANSFER(L302)PROVIDED(.ABS.(D-DIST).L.ERROR)@
             DU=DU*DIST/D@
             TRANSFER(L304)@
L302         Q(N)=X@
             R(N)=Y@
             N=N+1@
             Y1=Y@
             X1=X@
             U1=U@
             TRANSFER(L304)@
L303         D=SQRT.((X1*X1)+(.3-Y1)*(.3-Y1))@
             TRANSFER (L399)PROVIDED(D.LE.DIST/2.)@
             TRANSFER(L305)PROVIDED(D.G.DIST*1.5)@
L306         Q(N)=.00498@
             R(N)=.3@
             N=N+1@
             TRANSFER(L399)@
L305         X=X1*.5@
             Q(N)=X@
             R(N)=Y@
             N=N+1@
             TRANSFER(L306)@
L399         CONTINUE@
             NR=-N@
             DO THROUGH(L121),DV=.0,.174533,DV.L.25.14@
             COS=COS.(DV)@
             DO THROUGH(L121),M=0,1,M.LE.N@
             CRT=SIN.(1.570796*(1.-3.333333*Y(M)))@
             DX=.0332*COS*CRT@
             CALL SUBROUTINE()=LINE.(X(M),Y(M),X(M)+DX,Y(M))@
             CALL SUBROUTINE()=LINE.(-X(M),Y(M),-X(M)-DX,Y(M)
                )@
             CALL SUBROUTINE()=LINE.(-X(M),-Y(M),-X(M)-DX,-Y(
                M))@
             CALL SUBROUTINE()=LINE.(X(M),-Y(M),X(M)+DX,-Y(M)
                )@
L121         CONTINUE@
             CALL SUBROUTINE()=APLOTV.(NR,Q(0),R(0),1,1,1,4
              2.IRR)@
              IS=30.*SIN.(DV)@
              TRANSFER(L20.L30.L30)PROVIDED(IS)@
L20           IX=2@
              TRANSFER(L40)@
L30           IX=1@
L40           CALL SUBROUTINE()=GENSYM.(IX..ABS.IS)@
              CALL SUBROUTINE()=FRAMEV.(-1)@
L122          CONTINUE@
              CALL SUBROUTINE()=CLOSTP.()@
         END  END PROGRAM(START)@
```

```
 ***    SCATRAN
C             MAIN PROGRAM FOR SPHEROIDAL RADIATOR @
 START          READ INPUT .8.(NA.NB.NC.NSET.KNUMB)@
                READ INPUT .L3.(DELTAU.DIST.ERROR.DV.TSTART.TE
              ND.VSPHER)@
                WRITE OUTPUT .L4.(DELTAU.DIST.ERROR.TSTART.DV.
              TEND.VSPHER.NA.NB.NC.NSET.KNUMB)@
F L4          (//10T.Q*FIRST INCREMENT IN U =*.F10.8.//10T.Q*D
              ISTANCE BETWEEN POINTS =*.F7.6.//10T.Q*MAXIMUN E
              RROR =  *.F15.8.//10T.Q*STARTING TIME =  *.F15.8
              .//10T.Q*TIME INCREMENT = *.F15.8.//10T.Q*ENDING
               TIME =     *.F15.8.//10T.Q*SPHEROID INDEX = *.F1
              5.8.//10T.Q*OUTPUT FORMAT =  *.I15.//10T.Q*CAMER
              AS =          *.I15.//10T.Q*FRAME MARKS =    *.I15
              .//10T.Q*NUMBER OF CYCLES = *.I13.//10T.Q*NUMBER
               OF STREAMLINES = *.I8./)@
                READ INPUT .L3.((S(I).I=0.1.I.L.KNUMB))@
F L3          (4F15.8)     @
                CALL SUBROUTINE ()=CAMRAV.(NB)@
                FLOATING (K)@
                DIMENSION (Q(7800).R(7800).S(15))@
                DO THROUGH (L14).FA=TSTART.DV.FA.L.TEND@
               N=0@
               VSQM1=VSPHER*VSPHER-1@
               PTFV=.25*VSPHER@
               DU=DELTAU@
               U1=0.@
               X1=.25*SQRT.(VSQM1)@
               Y1=0.@
               Q(N)=X1@
               R(N)=Y1@
               N=N+1@
  L304         U=U1+DU@
                TRANSFER TO (L303) PROVIDED (U.G.1.)@
  L301         X=.25*SQRT.((1.-U*U)*VSQM1)@
               Y=PTFV*U@
               D=SQRT.((X-X1)*(X-X1)+(Y-Y1)*(Y-Y1))@
                TRANSFER TO (L302) PROVIDED (.ABS.(D-DIST).L.E
              RROR)@
               DU=DU*DIST/D@
                TRANSFER TO (L304)@
  L302         Q(N)=X@
               R(N)=Y@
               N=N+1@
               Y1=Y@
               X1=X@
               U1=U@
                TRANSFER TO (L304)@
  L303         D=SQRT.((X1*X1)+(PTFV-Y1)*(PTFV-Y1))@
                TRANSFER TO (L399) PROVIDED (D.LE.DIST/2.)@
                TRANSFER TO (L305) PROVIDED (D.G.DIST*1.5)@
```

```
L306         Q(N)=.0049804690
             R(N)=PTFV@
             N=N+1@
              TRANSFER TO (L399)@
L305         X=X1*.5@
             Q(N)=X@
             R(N)=Y@
             N=N+1@
              TRANSFER TO (L306)@
L399         DTME=FA@
             SIN=SIN.(1.570796*(FA-.2))@
              DO THROUGH (L121).M=0.1.M.L.N@
             CRT=SIN.(1.570796*(1.-3.33333*R(M)))@
             DX=.0332*SIN*CRT@
              CALL SUBROUTINE ()=LINE.(Q(M).R(M).Q(M)+DX.R(M
               ))@
              CALL SUBROUTINE ()=LINE.(-Q(M).R(M).-Q(M)-DX.R
            (M))@
              CALL SUBROUTINE ()=LINE.(-Q(M).-R(M).-Q(M)-DX.
             R(M))@
              CALL SUBROUTINE ()=LINE.(Q(M).-R(M).Q(M)+DX.-R
            (M))@
L121          CONTINUE @
             IS=30.*COS.(1.570796*FA)@
              TRANSFER (L40.L41.L41) PROVIDED (IS)@
L40          IX=2@
              TRANSFER (L42)@
L41          IX=1@
L42           CALL SUBROUTINE ()=GENSYM.(IX..ABS.IS)@

              DO THROUGH (L16).KA=0.1.KA.L.NSET@
             I=0@
```

```
***     SCATRAN
C               MAIN PROGRAM FOR SPHEROIDAL RADIATOR @
 START            READ INPUT .8.(NA.NB.NC.NSET.KNUMB)@
                  READ INPUT .L3.(DELTAU.DIST.ERROR.DV.TSTART.TE
                ND.VSPHER)@
                  WRITE OUTPUT .L4.(DELTAU.DIST.ERROR.TSTART.DV.
                TEND.VSPHER.NA.NB.NC.NSET.KNUMB)@
F L4            (//10T.Q*FIRST INCREMENT IN U =*.F10.8.//10T.Q*D
                ISTANCE BETWEEN POINTS =*.F7.6.//10T.Q*MAXIMUN E
                RROR =   *.F15.8.//10T.Q*STARTING TIME =   *.F15.8
                .//10T.Q*TIME INCREMENT = *.F15.8.//10T.Q*ENDING
                 TIME =     *.F15.8.//10T.Q*SPHEROID INDEX = *.F1
                5.8.//10T.Q*OUTPUT FORMAT =   *.I15.//10T.Q*CAMER
                AS =            *.I15.//10T.Q*FRAME MARKS =     *.I15
                .//10T.Q*NUMBER OF CYCLES = *.I13.//10T.Q*NUMBER
                 OF STREAMLINES = *.I8./)@
                  READ INPUT .L3.((S(I).I=0.1.I.L.KNUMB))@
F L3            (4F15.8)     @
                  CALL SUBROUTINE ()=CAMRAV.(NB)@
                  FLOATING (K)@
                  DIMENSION (Q(7800).R(7800).S(15))@
                  DO THROUGH (L14).FA=TSTART.DV.FA.L.TEND@
                 N=0@
                 VSQM1=VSPHER*VSPHER-1@
                 PTFV=.25*VSPHER@
                 DU=DELTAU@
                 U1=0.@
                 X1=.25*SQRT.(VSQM1)@
                 Y1=0.@
                 Q(N)=X1@
                 R(N)=Y1@
                 N=N+1@
  L304           U=U1+DU@
                  TRANSFER TO (L303) PROVIDED (U.G.1.)@
  L301           X=.25*SQRT.((1.-U*U)*VSQM1)@
                 Y=PTFV*U@
                 D=SQRT.((X-X1)*(X-X1)+(Y-Y1)*(Y-Y1))@
                  TRANSFER TO (L302) PROVIDED (.ABS.(D-DIST).L.E
                RROR)@
                 DU=DU*DIST/D@
                  TRANSFER TO (L304)@
  L302           Q(N)=X@
                 R(N)=Y@
                 N=N+1@
                 Y1=Y@
                 X1=X@
                 U1=U@
                  TRANSFER TO (L304)@
  L303           D=SQRT.((X1*X1)+(PTFV-Y1)*(PTFV-Y1))@
                  TRANSFER TO (L399) PROVIDED (D.LE.DIST/2.)@
                  TRANSFER TO (L305) PROVIDED (D.G.DIST*1.5)@
```

```
L306         Q(N)=.0049804690
             R(N)=PTFV@
             N=N+1@
              TRANSFER TO (L399)@
L305         X=X1*.5@
             Q(N)=X@
             R(N)=Y@
             N=N+1@
              TRANSFER TO (L306)@
L399         DTME=FA@
             SIN=SIN.(1.570796*(FA-.2))@
              DO THROUGH (L121),M=0,1,M.L.N@
             CRT=SIN.(1.570796*(1.-3.33333*R(M)))@
             DX=.0332*SIN*CRT@
              CALL SUBROUTINE ()=LINE.(Q(M),R(M),Q(M)+DX,R(M
               ))@
              CALL SUBROUTINE ()=LINE.(-Q(M),R(M),-Q(M)-DX,R
            (M))@
              CALL SUBROUTINE ()=LINE.(-Q(M),-R(M),-Q(M)-DX,
             R(M))@
              CALL SUBROUTINE ()=LINE.(Q(M),-R(M),Q(M)+DX,-R
            (M))@
L121          CONTINUE @
             IS=30.*COS.(1.570796*FA)@
              TRANSFER (L40,L41,L41) PROVIDED (IS)@
L40          IX=2@
              TRANSFER (L42)@
L41          IX=1@
L42           CALL SUBROUTINE ()=GENSYM.(IX,.ABS.IS)@
              DO THROUGH (L16),KA=0,1,KA.L.NSET@
             I=0@
              DO THROUGH (L9),KB=0,1,KB.L.KNUMB@
             K=S(I)@
             V=2.-.63661977*ARCSIN.(K)+DTME@
              TRANSFER TO (L9) PROVIDED (V.L.VSPHER)@
             UMAX=.63661977*ARCCOS.(K)@
             U=0.0@
             U1=0.0@
             Y=0.0@
             Y1=0.0@
             X=.25*SQRT.(V*V-1.)@
             X1=X@
             Q(N)=X@
             R(N)=Y@
             N=N+1@
L8           U=U1+DELTAU@
              TRANSFER TO (L11) PROVIDED (U.G.UMAX)@
             COS=COS.(1.5707963*U)@
             V=2.-.63661977*ARCSIN.(K/COS)+DTME@
             X=.25*SQRT.((1.-U*U)*(V*V-1.))@
             Y=.25*U*V@
```

```
              D=SQRT.((X-X1)*(X-X1)+(Y-Y1)*(Y-Y1))@
               TRANSFER TO (L10) PROVIDED (.ABS.(D-DIST).L.ER
             ROR)@
              DELTAU=DELTAU*DIST/D@
               TRANSFER TO (L8)@
L10            TRANSFER TO (L9) PROVIDED (V.L.VSPHER)@
              Q(N)=X@
              R(N)=Y@
              N=N+1@
              X1=X@
              Y1=Y@
              U1=U@
               TRANSFER TO (L8)@
L11           U=UMAX@
              V=1.+DTME@
              X=.25*SQRT.((1.-U*U)*(V*V-1.))@
              Y=.25*U*V@
              D=SQRT.((X-X1)*(X-X1)+(Y-Y1)*(Y-Y1))@
               TRANSFER TO (L19) PROVIDED (D.L.DIST/2.)@
               TRANSFER TO (L17) PROVIDED (D.G.DIST*1.5)@
               TRANSFER TO (L9) PROVIDED (V.L.VSPHER)@
              Q(N)=X@
              R(N)=Y@
              N=N+1@
              X1=X@
              Y1=Y@
              U1=U@
               TRANSFER TO (L13)@
L19           U=U1@
              COS=COS.(1.5707963*U)@
              V=.63661977*ARCSIN.(K/COS)+DTME@
              X=.25*SQRT.((1.-U*U)*(V*V-1.))@
              Y=.25*U*V@
              D=SQRT.((X-X1)*(X-X1)+(Y-Y1)*(Y-Y1))@
               TRANSFER TO (L13) PROVIDED (D.L.DIST)@
              U1=UMAX@
               TRANSFER TO (L13)@
L17           U=(UMAX+U1)/2.@
              DELTAU=DELTAU/2.@
L20           COS=COS.(1.5707963*U)@
              V=2.-.63661977*ARCSIN.(K/COS)+DTME@
              X=.25*SQRT.((1.-U*U)*(V*V-1.))@
              Y=.25*U*V@
              D=SQRT.((X-X1)*(X-X1)+(Y-Y1)*(Y-Y1))@
               TRANSFER TO (L10) PROVIDED (D.G.(DIST-ERROR))@
              U=(UMAX+U)/2.@
               TRANSFER TO (L20)@
L13           U=U1-DELTAU@
              COS=COS.(1.5707963*U)@
              V=.63661977*ARCSIN.(K/COS)+DTME@
              X=.25*SQRT.((1.-U*U)*(V*V-1.))@
```

```
                Y=.25*U*V@
                D=SQRT.((X-X1)*(X-X1)+(Y-Y1)*(Y-Y1))@
                 TRANSFER TO (L15) PROVIDED (.ABS.(D-DIST).L.ER
               ROR)@
                DELTAU=DELTAU*DIST/D@
                 TRANSFER TO (L13)@
L15              TRANSFER TO (L9) PROVIDED (V.L.VSPHER.OR.U.L.0
                  .0)@
                Q(N)=X@
                R(N)=Y@
                N=N+1@
                X1=X@
                Y1=Y@
                U1=U@
                 TRANSFER TO (L13)@
L9              I=I+1@
                NR=-N@
                 CALL SUBROUTINE()=APLOTV.(NR.Q(0).R(0).1.1.
               1.42.IRR)@
                NE=NE+N@
                N=0@
L16             DTME=DTME+2.@
                ND=ND+1@
                IRR=IRR.RS.18@
                 WRITE OUTPUT .L35.(ND.NE.IRR)@
F L35          (/10T.Q*THE NUMBER OF POINTS PLOTTED IN FRAME*.I
               3.Q* IS*.I7./20T.Q*OFF SCALE POINTS = *.I11.//)@
                NE=0@
                 CALL SUBROUTINE ()=FRAMEV.(NC)@
L14              CONTINUE @
                 CALL SUBROUTINE()=CLOSTP.()@
                 FUNCTION (ALPHA)=ARCCOS.(BETA)@
                ALPHA=ATAN.(SQRT.(1.-BETA*BETA)/BETA)@
                 NORMAL EXIT @
                 END SUBPROGRAM @
                 FUNCTION (ALPHA)=ARCSIN.(BETA)@
                ALPHA=ATAN.(BETA/SQRT.(1.-BETA*BETA))@
                 NORMAL EXIT @
                 END SUBPROGRAM @
        END      END PROGRAM (START)@
```

```
 ***     SCATRAN
C                 THIS PROGRAM WRITES THE END IN BLOCK LETTERS
  START           CALL SUBROUTINE()=CAMRAV.(0)@
                  DO THROUGH(L1).I=0.1.I.L.12@
                  IXS=I+1@
                  IX=4*IXS+3@
                  X=-.039844*IXS-.0332@
                  Y=-.00166*IXS@
                  DO THROUGH(L1).J=0.1.J.L.3@
                  CALL SUBROUTINE()=CHSIZS.(IXS.IXS)@
                  CALL SUBROUTINE()=RITSTS.(IX.100.TABLSV.)@
                  CALL SUBROUTINE()=RITE2S.(X.Y.750.90.1.7.-1.WORD
                     15.IR)@
                  CALL SUBROUTINE()=FRAMEV.(0)@
  L1              CONTINUE@
                  CALL SUBROUTINE()=RITE2S.(X.Y.750.90.1.7.-1.WORD
                     15.IR)@
                  CALL SUBROUTINE()=STORE.(Z.4)@
                  DO THROUGH(L2).N=0.1.N.L.95@
                  CALL SUBROUTINE()=STORE.(Z.3)@
  L2              CONTINUE  @
                  CALL SUBROUTINE()=STORE.(Z.2)@
                  CALL SUBROUTINE()=STORE.(Z.5)@
                  CALL SUBROUTINE()=CLOSTP.()@
F WORD15          0125234@
             END  END PROGRAM(START)@
```

BIBLIOGRAPHY

The following books are useful in supporting the work contained in this dissertation.

For the SC 4020 plotter:

______, Programmers' Reference Manual SC 4020 Computer Recorder (San Diego, Stromberg-Carlson Corporation, August 1965).

B. R. Groves, Ohio State University SC 4020 Subroutine Library (unpublished, contains computer assembly listings at Appendix B, available from Prof. J. Cowan, Jr., E. E. Dept.).

For the FAP programming language:

______, IBM 7094 Principles of Operation File No. 7094-01, Form A22-6703-3 (Poughkeepsie, N. Y., IBM Corporation 1962).

______, IBM 7090/7094 Programming Systems, FORTRAN Assembly Program (FAP) File No. 7090-21, Form C28-6235-5 (Poughkeepsie, N. Y., IBM Corporation, 1963).

______, IBM Reference Card, 7094 FAP Codes Form X22-6691-1 (Poughkeepsie, N. Y., IBM Corporation, 1965).

For the SCATRAN programming language:

Roy F. Reeves, et al., SCATRAN Reference Manual (Columbus, Ohio, Numerical Computation Laboratory, The Ohio State University, 1963).
Section III OS-FAP Assembler
Section IV Programmers' Reference Manual
Section V 7094 System Subroutine Library

For the FORTRAN II programming language:

______, IBM 7090/7094 Programming Systems, FORTRAN II Programming File No. 7090-25, Form C28-6054-5 (Poughkeepsie, N. Y., IBM Corporation, 1963).

______, _IBM 7090/7094 Programming Systems, FORTRAN II Operations_ (Poughkeepsie, N. Y., IBM Corporation, 1963).

For producing magnetic tape:

______, _IBM 729, 7330, and 727 Magnetic Tape Units, Principles of Operation, File No. GENL-05, Form A22-6589-3_ (Poughkeepsie N. Y., IBM Corporation, August 1964).

______, _IBM 729, 7330 Magnetic Tape Record Characteristics, Form X22-6785-4_ (Poughkeepsie, N. Y., IBM Corporation, 1965).