1982

An Improved Flight Simulator Graphics System Using Microcomputer Technology

Wayne D. Parsons
University of Central Florida

AN IMPROVED FLIGHT SIMULATOR GRAPHICS SYSTEM USING MICROCOMPUTER TECHNOLOGY

BY

WAYNE D. PARSONS
B.S.E.E., University of Florida, 1975

RESEARCH REPORT

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

Spring Term
1982

ABSTRACT

The field of computer graphics has continued to provide an efficient means of depicting information about complex phenomena for its users. It has become a widely used tool through which the user may manipulate data to generate a different perspective of the problem at hand and, hence, offers a solution to many varied problems. One area of application for computer graphics is the field of flight simulation.

At the University of Central Florida, research is being conducted in the area of computer graphics simulation to develop a method through which a pictorial representation of the outline of a small airport runway may be modified to appear as viewed by a pilot in a defined airspace.

The purpose of this paper is to provide a means of interfacing a small computer to a flight simulator device as well as a graphics terminal. This new implementation of the software will allow a pictorial display to be continuously modified by the changing positional and attitudinal parameters provided from a flight simulator's input. Another goal of this paper is to generate faster display turnaround times by programming the computer in assembly language. Further, the hardware that accomplishes this task is discussed. Finally, suggestions for continued research in this area conclude the report.

ACKNOWLEDGMENTS

This report represents the culmination of over five years of study during which many kind and dedicated people contributed invaluable assistance.

It has been my pleasure to work with Dr. Christian S. Bauer whose helpful guidance and candid criticism as graduate advisor proved to be a real test of his endurance. I would like to also praise Ms. Lee S. Yi and Ms. Dian E. Brandstetter whose careful editing, knowledge of formatting regulations, and care allowed this report to mature into a professional product.

In my brief professional career I have had the opportunity to have worked for individuals who provided a means of cutting through corporate restrictions and allowed time for study. Foremost among these was the late Mr. Al Stearns who matured a green engineer, befriended and encouraged me to go for the highest level of accomplishment. Al provided me with a singlemindedness of purpose in the worth of my endeavors. Also, Mr. Charlie Hafer recognized the importance of continuing that endeavor within a new environment.

Finally, I would like to thank all of my family for providing encouragement, listening to my frustrations, and taking the time to love me.

Above all, this work is dedicated to Gail, whose unending and tireless devotion to me and our children, Christie and Angela, is cornerstone of all that is love.

TABLE OF CONTENTS

Chapter

I. INTRODUCTION 1

II. THE SYSTEM HARDWARE 4

III. BASIC PRINCIPLES 8

IV. A NEW IMPLEMENTATION 24

V. CONCLUSION AND SUGGESTIONS FOR FURTHER STUDY 53

APPENDICES

A. "FLITSIM" M6800 BASIC 57

B. "FLITSIM" M6800 ASSEMBLY 68

C. INPUT PARAMETER SCALING AND DIRECTION CONVENTIONS 105

D. PRE AMP AND AD-68A BOARD EXPLANATION 108

E. GT-6144 OPERATION 114

LIST OF REFERENCES 121

CHAPTER I

INTRODUCTION

In a paper, O'Callaghan (1972, p. 123) says, "The feature distinguishing a graphic language from other languages for man-machine communication is the use of pictures during the interaction." Work in defining what is expected of a graphic language has resulted in several world-wide conferences where scientists and engineers have tried to find ways of standardizing essential and fundamental concepts. The International Federation for Information Processing (IFIP) Working Conference on Graphic Languages (Nake and Rosenfeld 1972) found that there exists three levels of interest:

a. Formal languages which generate or parse non-string-like objects such as arrays, labeled graphs, etc.

b. Programming languages for interactive graphics.

c. Software packages for both graphics and image processing.

In the 1976 IFIP Workshop on Methodology In Computer Graphics, the committee, after an initial study, decided as a necessary preliminary step, to concentrate on creating a methodology around which a standard could later be built (Guedj and Tucker, 1979).

From this workshop a number of classifications, definitions, and recommendations emerged as an approach for manipulating pictorial representations by a computer. Guedj and Tucker (1979, p. 193) thought that a high-level language methodology would be useful for most computer graphics applications as stated in their notes on the final session of the workshop:

> In application programs using graphics, various transformations are used at different phases of the process, i.e., viewing and modeling situations. All transformations employ the same basic matrix operations. However, their corresponding natural interpretations differ and depend on the context. The practice of using transformations interactively has often led to some confusion of concepts and consequently can strained [constrains, SIC] the portability of the whole application, for instance, to use a transformation usually associated with modeling during the viewing phase or vice versa.

At the University of Central Florida, work in the computer graphics area is being concentrated on its application to small computers. Local interest in a graphics package as it could be applied to a flight simulator problem is perhaps fostered by the University's proximity to the Naval Training Equipment Center, whose task is to provide, upgrade, and modify training simulators of all types for the Armed Services. In a research paper, Larry Holley investigated the need for universal graphics language and reviewed several computer graphics systems. Mr. Holley (1975, p. 5) stated the following:

> The opinion as to whether assembly language or compiler-level language should be used has changed over the brief life-span of the computer business. Originally one would have worked exclusively in assembly language or machine language. The compilers that existed were very inefficient and very slow, and they produced very slow running codes.

Prompted by this research, further work was done by Mr. Jerry Campbell (1979) who took the above recommendations and coupled them with an interest in Flight Simulation. Mr. Campbell's research resulted in a BASIC flight simulation program which graphically outlines the view of an airport runway that is modified with the viewer's changing position in space. However, the two minutes required for the generation of each display is obviously too slow for the dynamically changing flight simulator inputs and hence prohibits a realistic display on a continuous basis. This is due to the computation time required by the high-level interpretive language to perform the required mathematical transformations.

This study is designed to improve the algorithm implementation presented in BASIC by Mr. Campbell. It will allow display updates to occur via six analog signals from a flight simulator. The computer will provide the user with prompts which will allow him to initialize the system, choose a desired airport runway, and tell him when a landed condition has occurred. A transformation from the high-order BASIC code to a corresponding assembly language program allows a more efficient means of manipulating the data and display updates improve to a rate of 2 to 3 times per second. The study uses a subroutine provided by Sublogic's Bruce Artwick (1977) that will perform the mathematical manipulations required. Finally, the system's hardware is discussed. The end product of the report is a system which brings all of these elements together yielding an operational flight simulator.

CHAPTER II

THE SYSTEM HARDWARE

The defined system hardware used in this study is shown in figure 1 and consists of the following component parts:

1. A Southwest Technical Products Corporation Motorola 6800-based Microcomputer with 32K Bytes of memory and a floppy disk operating system. This computer houses an Innovative Technology Model AD-68A Analog to Digital Converter which is used to interface the M6800 with 6 analog signals provided by the ATC-610J Flight Simulator. In addition a Pre-Amp Board which serves to scale the analog inputs (Offset and Gain Adjusts) developed by former University of Central Florida student Art Weeks, from the ATC-610J precedes the AD-68A card. The M6800 peripherals include a Dual Floppy Disk Drive, a Heathkit H14 Line Printer, a ADM Terminal and a Teletype paper type reader.

2. An ATC-610J Flight Simulator developed by Analog Training Computers Incorporated. Detailed information concerning this device may be found in its Service Manual (Analog Training Computers, Inc. 1977).

3. A GT-6144 Digital to Video Graphics Interface developed by Southwest Technical Product Corporation provides the link between the M6800 computer and the television monitor.

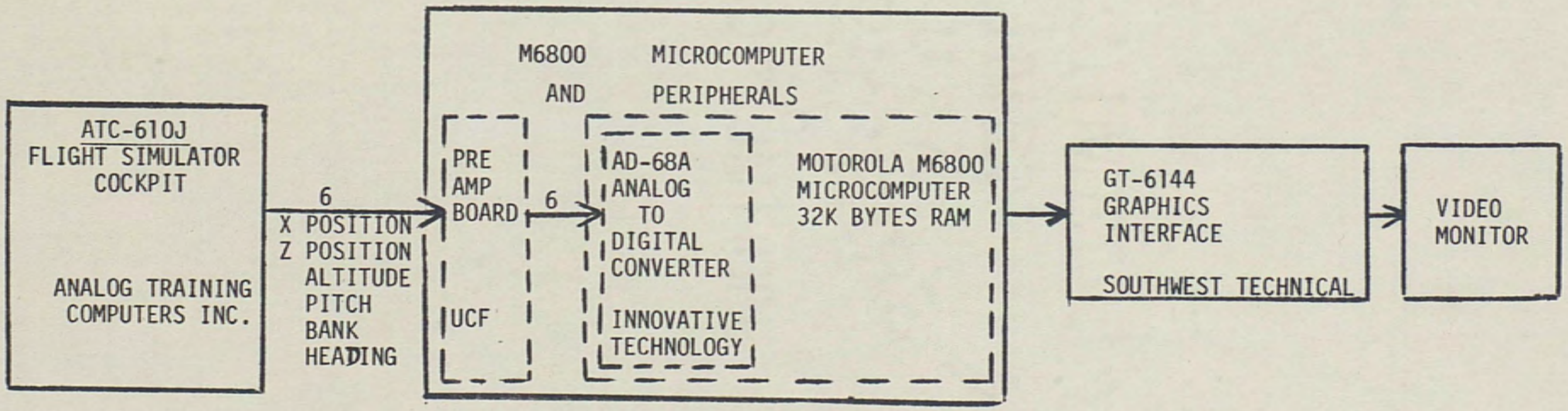


Figure 1. System hardware block diagram

4. A Video Output Monitor is provided for the purpose of displaying the new views of the landing strip.

The assembled program can be loaded into the computer's random access memory via the floppy disk drives and the teletype's paper tape reader.

The six analog signals which will serve to update the configuration of the output are X and Z position, Altitude, Pitch, Bank, and Heading. The initial position parameters are loaded from the sequencing program and the simulator is initialized through instructive computer driven prompts. Otherwise the ongoing simulation will require the analog signals to be allowed to be input from the ATC-610J Flight Simulator upon demand from the computer. As the student pilot maneuvers the simulator towards a landing, the positional signals will change. These signals are provided to the Pre-Amp board which scales all of the signals for a minimum of 0.00 volts and a maximum of 2.50 volts. The minimum and maximum voltage values for each parameter are summarized here:

	0.00 Volts	2.50 Volts
X Position	-28,500 feet	+28,500 feet
Z Position	-28,500 feet	+28,500 feet
Altitude	5,000 feet	0 feet
Pitch	90°	-90°
Bank	90°	-90°
Heading	90°	-180°

When the computer requests new parameter informatin the appropriate analog signal polled will be sampled and compared via the AD-68A card resident within the M6800 (see Appendix D for details of the AD-68A operation). As each signal is polled, a computer subroutine will save the 8 bit hexadecimal digital representation of the analog voltage which it has sampled. The program must then convert this voltage representation of the parameter into actual units of the parameter and further in the case of the angular components of Pitch, Bank, and Heading into the appropriate "sine" or "cosine" of the angle. A summary of the transformations required for each scaled 8 bit parameter input, which is a voltage representation of the value, to the correct 16 bit two's complement parameter units value is shown in Appendix C. It also contains a diagram that defines the direction of movement conventions required by the system software driver. Upon completion of the mathemathical algorithm which serves to clip, translate, and project the 3 dimensional geometry into a two-dimensional picture the information of the new display is provided by the computer in the form of a From (X,Y) and To (X,Y) coordinate within a dimensioned screen. The GT-6144 graphics Interface has the job of receiving the information from the computer and providing the proper video signals so that the new display may be shown. Details of the GT-6144 Graphics Interface Box are provided in Appendix E along with an explanation of its operation.

CHAPTER III

BASIC PRINCIPLES

Before proceeding on with the detailed explanation of the new implementation, it is necessary to briefly outline the principles upon which it is based. This chapter will define the coordinate systems and their interrelationships, review the scaling problem, define the variables used, and identify differences between the BASIC algorithm and the Assembly algorithm.

The variables, X3, Y3, and Z3, are denoted either a viewpoint or eye coordinate (Xe, Ye, Ze), and are respectively the viewer's West/East, Altitude, and South/North coordinate locations within the defined field of play. The angular components which also must be input concerning the viewers attitude are: Pitch, P, the angle of inclination from which the observer views the scene (i.e., nose up or nose down); Bank, B, the angle of tilt of the wings from the true horizon, and; Heading, H, the direction the viewer is facing in relations to the XZ axis plan. In addition, two constants that are required are V, the tangents of the field of view, and W [(screen width /2) - 1] which is used to scale the final output points to the screen. Figure 2 shows how these parameters are defined.

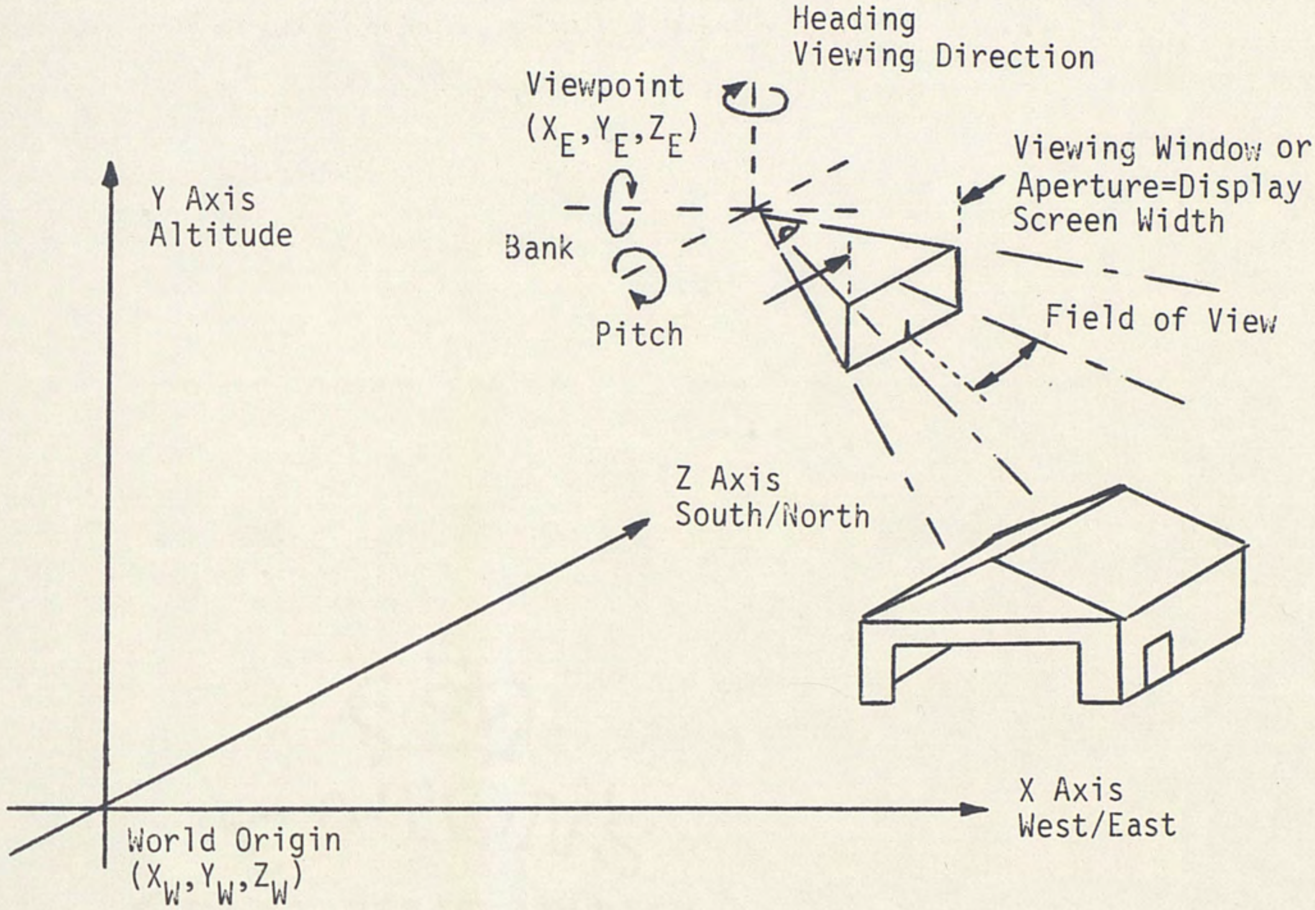


Figure 2. Definition of Parameters

The gaming area of "World" size for the "BASIC" routine implementation is not tightly constrained by the software because it uses a floating point mathematical package. The package will accommodate numbers that exceed the pixel resolution of the GT-6144 display driver. Since the "BASIC" routines only input source is through a keyboard via the operator, there are no hardware constraints on the system field of play.

The Assembly algorithm is constrained in the "World" size because of two factors. First, the addition of the ATC-610J simulator and the Pre Amp scaling hardware limit the field of play to:

±75 nautical miles in the North/South or Z direction,

±75 nautical miles in the East/West or X direction,

and 5000 feet in Altitude or Y direction.

In addition the new implementation's display driver software constrains the gaming area by requiring the 16 bit two's complement mathematical package format. The constraint confines the maximum "area of regard" for the system to be

+32768 to -32767 feet in X, Z, and Y directions.

The new software uses a method through which the 8 bit digitized voltage input (variables X3, Y3, Z3) is used to select the correct unit of measure value (variables XV, YV, ZV) of the parameter. Since the AD-68A allows only 250 possible 8 bit inputs (instead of 256) to select the required 16 bit two's complement value the field of regard or displayable area is confined again. The X and Z component parameters are actually limited to a range of ±28,500 feet and each incremental change in the 8 bit input is equivalent to 228 feet.

Notice the X and Z directions are limited by the scaling for the software driver while the Altitude (5000 feet) is limited by the ATC-610J simulator hardware. This effect allows the aircraft to fly in a "Simulated World" equivalent to the dimensions allowed by the ATC-610J hardware. However, the area of the world which is allowed to be viewed by the software driver is limited to the scaled two's complement (±28,500 feet) value in the X and Z directions and 5,000 feet in Altitude. The problem is further complicated due to the fact that the runway is not located at the simulator world origin. This results in two constant translation offsets in the X and Z directions that must be accounted for in the Assembly version which were not required by the Basic routine. Figures 3 and 4 help to clarify these relationships. Before leaving the discussion of the coordinate scaling of parameters, it should be noted that the altitude parameter has a resolution of approximately:

$$\frac{\text{5000 Feet}}{\text{250 possibilities}} \cong \text{20 feet/incremental 8 bit input change.}$$

This parameter as input from the ATC-610J simulator, however, is not a linear function and consequently each 1000 foot interval required independent scaling.

The angular components of the pilot's attitude are also constrained by the ATC-610J hardware and the Assembly version

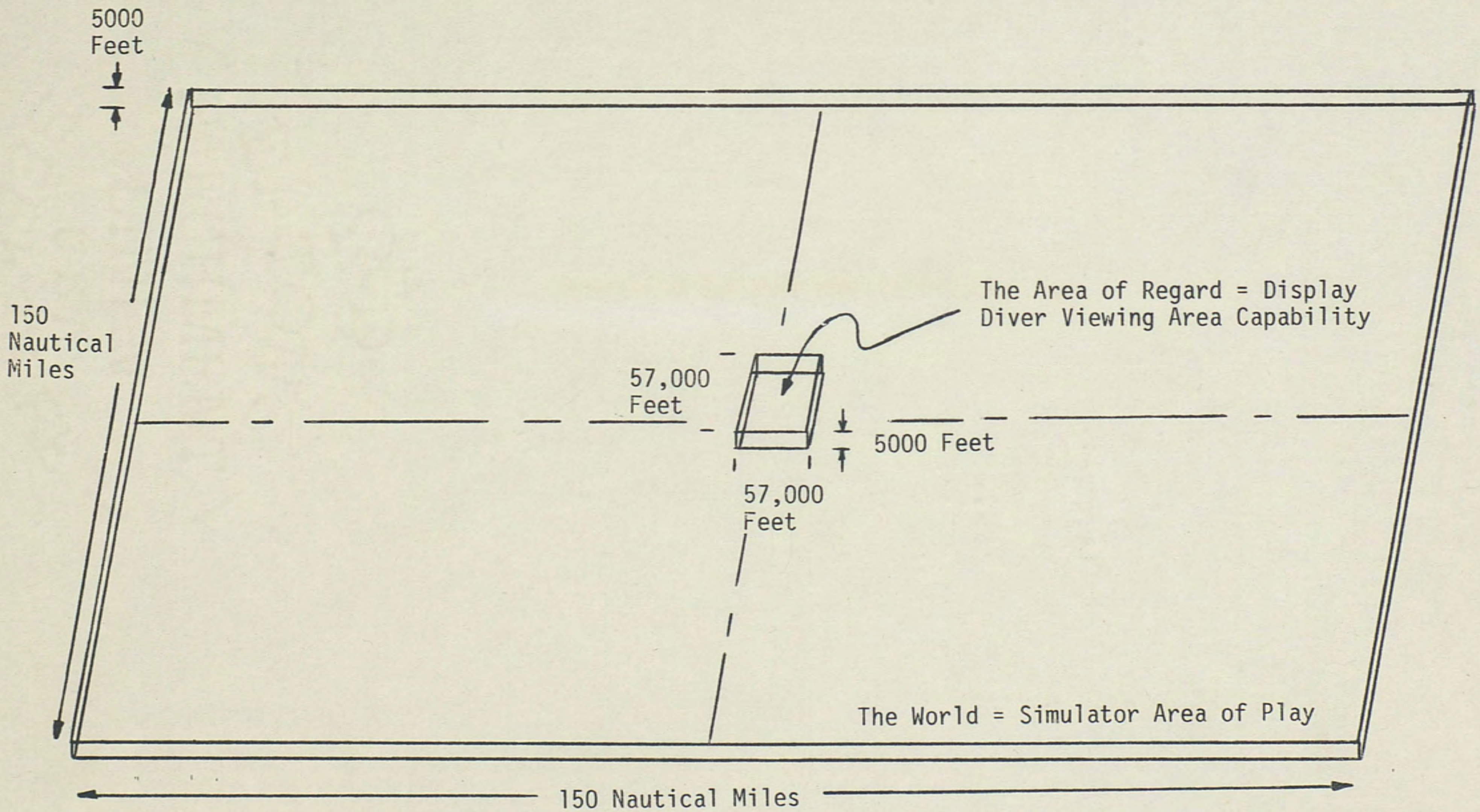


Figure 3. The Scaling Problem

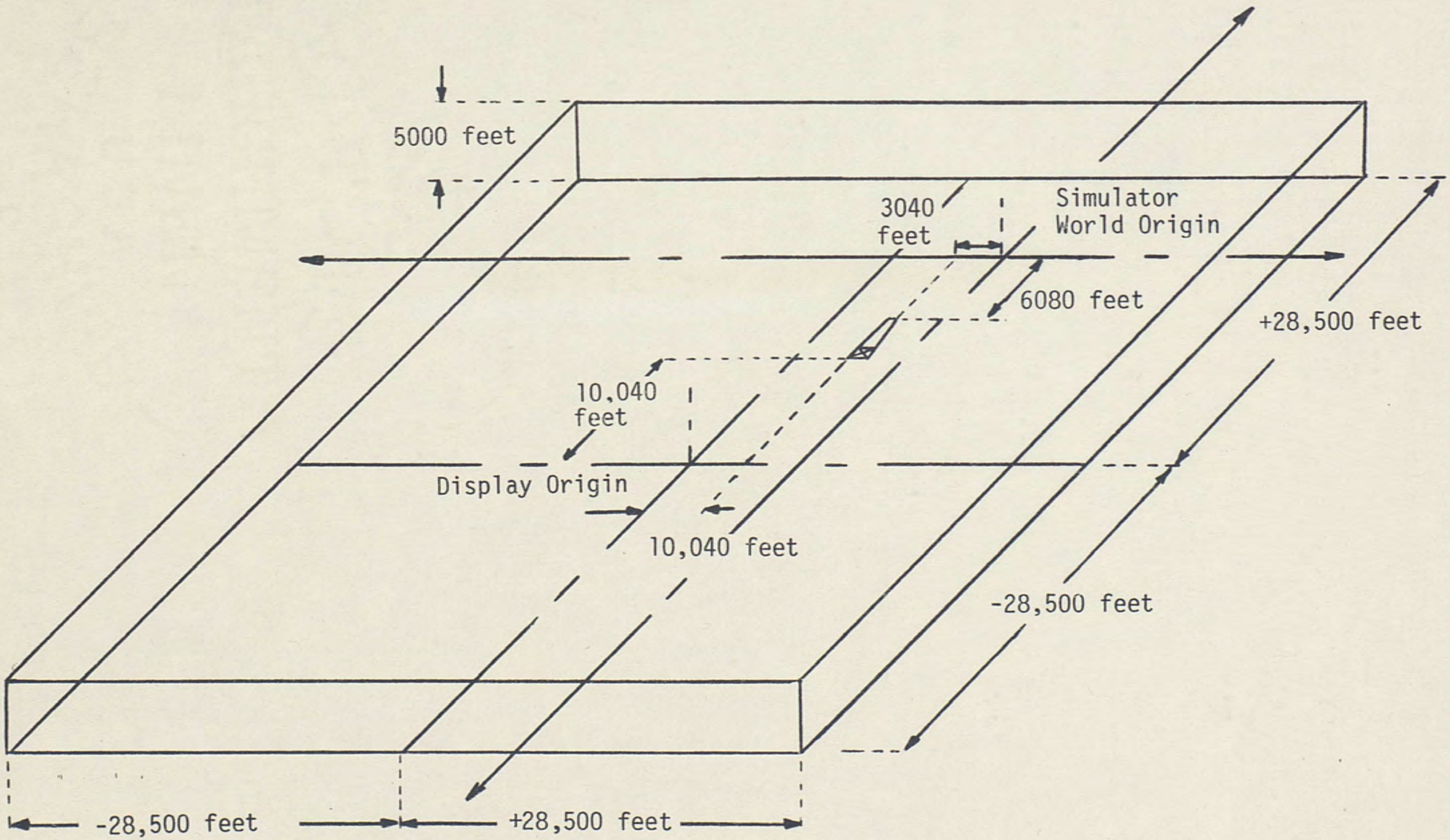


Figure 4. Relationship of Simulator World Origin, Airport and Graphics Display Software Driver Screen Origin.

software. The display driver constrains the Pitch, Bank, and Heading parameters all in the same way. The 16 bit two's complement variable may vary between +178.59 degrees and -180 degrees. The most significant byte of the 16 bits is always 0. Only the least significant 8 bits are used which results in a resolution of 1.4065 degrees.

The real limiting factor for the angular components is the simulator's hardware. The pitch and bank components analog input varies between ±90 degrees. For the pitch parameter the maximum nose up position has a voltage output of 2.5 volts from the Pre-Amp board. The even flight condition occurs at 1.25 volts and max down, -90 degrees, occurs at 0.00 volts. The Bank parameters analog scaling defines the left wing down as, -90 degrees, which is a 2.50 volt input. The right wing down condition is +90 degrees and a 0.00 volt input. The wing's level condition is the 0 degree position and has a 1.25 volt input.

The heading component varies from -180 degrees represented by 2.5 volts, to +90 degrees represented by 0.00 volts. Note that 270 degrees are represented by the heading input from the ATC-610J. The northly heading 0° or 360° is represented by a voltage of 0.860.

The search for the correct 16 bit two's complement value for the angular parameters is done in the same manner as the coordinate parameters. The 8 bit scaled and digitized value input for the parameter (variables PITCH, BANK, HEAD) are used to find the correct 16 bit two's complement value (variables PV, BV, HV) out of 250 possibilities.

In both the "BASIC" and "ASSEMBLY" program implementations, the three dimensional World coordinates of the airport runway are stored in database arrays in RAM. The World X values are stored in array A1 and the World Z coordinates are in A2. There is no World Y (Altitude direction) database array because the runway is assumed to be on the ground. Consequently, all of these values are zeros. It is important to realize the "World Y" defines the Altitude position in space whereas "Screen Y" is depth into the screen and is dependent primarily on the North/South position, the heading direction and the altitude, although the other parameters also affect it. The diagram in Figure 5 helps to show these relationships.

The BASIC program stores the database in two 192x1 arrays and requires two For-Next routines to allow the data to be input to the appropriate dimensional array. The data is stored in units of 100 feet. Conversely, the Assembly program must store the two's complement values of the endpoints of a runway with regard to the Screen origin. The program allows selection of a database between two airport database configurations, both located at the same world coordinates. Each runway's Northeast corner is located 1 nautical mile South and 0.5 nautical miles West of the Simulator or World Origin. This relates to an arbitrarily chosen

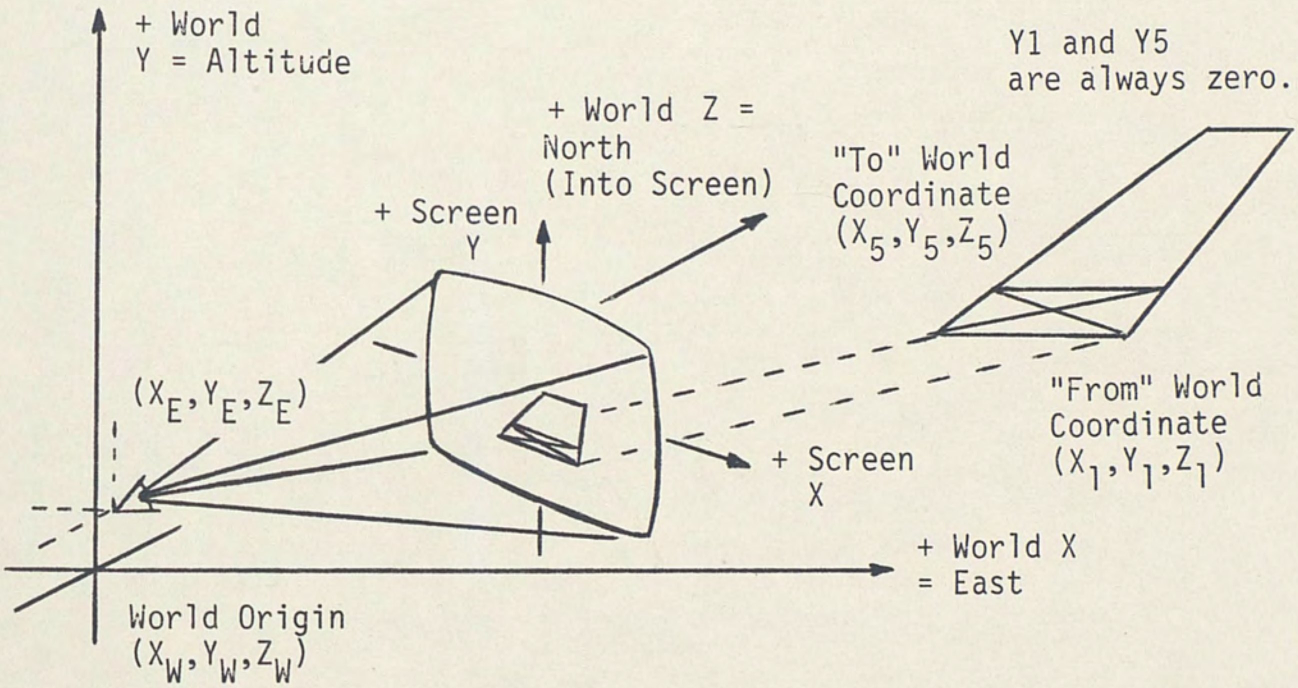


Figure 5. Screen Coordinates Versus World Coordinates

Screen origin which is 10,040 feet South and West of the Southwest corner of the runway (Figure 4).

A significant problem that is faced with a Flight Simulation program is the mathematical array manipulation. It is required of the programmer to generate the appropriate three-dimensional coordinate transformation equation for each data base array element in order to convert the flight perspective into a two dimensional graphics display.

These transformations can be classified under the generic name geometric transformation. Geometric transformations are objective mappings from the coordinate space (Giloi, 1978).

The three types of manipulations which must occur to accomplish the transformation are as follows:

1) Rotation,
2) Translation,
3) Clipping.

It should be remembered that for each manipulation of the data base the transformation must occur without changing the geometrical relationship of the data points in the original data base, that is, the scene should appear to move around the viewer. These requirements are aptly outlined by Paul, Falk, and Feldman (1969).

The Basic routine configures the well-known 3x3 Transformation Matrix through a subroutine which uses the cosine series expansion to find the sine and cosine of the angular parameters. The sines are generated by initially subtracting 90 degrees. Once the sine and cosine of the pitch, bank, and heading are obtained, the Matrix Generator routine forms the 9 element, T1-T9 Transformation Matrix. This study is not intended to explain well documented theory on the required mathematical manipulations, however, a detailed explanation of the Transformation Matrix as it applies to rotation about an axis can be found in Barry, Ellis, Graesser, and Marshall (1969).

The new implementation incorporates a subroutine package developed by Bruce Artwick of Sublogic (1977) to perform the mathematical manipulations. The sine and cosine of the parameter are obtained through look-up tables in order to speed up the processing.

When the T1-T9 elements of the Transformation Matrix have been generated from updated input variables a new display must be generated. At this point both the BASIC program and the Assembly program call the Graphics Display Driver routine to erase the screen. The BASIC routine uses the POKE command to allow it to choose one of five possible directives for the GT-6144 hardware. The POKE command which takes the 1st element of the field pair in decimal and stores it at the decimal location defined by the second element of the pair can choose one of five tasks:

User 1 = PIA Initialization Routine Call

User 2 = Joystick Initialization Routine Call

User 3 = Erase Routine Call

User 4 = PIXEL Routine Call

or User 5 = SHOW2 Routine Call [Draw Line (X1, Y1) to (X2, Y2)].

The second element of the pair is the decimal equivalent of the storage location in memory for the variable. It should be noted that the POKE command is required by the BASIC program so that it may communicate with the assembly complied Graphics Display Driver routine.

The Assembly version program simply stores the number of the desired routine call in the accumulator and branches to the beginning of the Display Driver routine where its value is decoded. When the desired routine has been decoded, another branch is initiated which allows the function to be implemented.

Upon completion of the Transformation Matrix generation which satisfies the rotational mathematical manipulation and erasure of the display screen, a translation is required. The BASIC routine translates each database From coordinate and each database To coordinate by adding the three element vector $\begin{matrix} X3 \\ Y3 \\ Z3 \end{matrix}$ of the Viewer's Eye Coordinate in space to it. The routine temporarily stores these values in scratch-pad memory locations $\begin{matrix} G1 \\ S1 \\ K1 \end{matrix}$ for the From coordinate and $\begin{matrix} G5 \\ S5 \\ K5 \end{matrix}$ for the To coordinate. The translated coordinate is then multiplied by the Transformation Matrix. As each From and To element pair is transformed, the resulting vector between the From and To points is compared to the boundary equations defining the intersection of the Viewing Pyramid with the screen in the Clipping routine.

A good explanation of the requirements of the Clipping Algorithm is given by Blinn and Newell (1978). Figure 6 demonstrates a line that is in need of pushing to the boundary of the pyramid. Flags in the BASIC program C1-C3 are used to code the required direction of push (Left, Right, Up and Down respectively) for the From coordinate and C4-C8 in the same manner for the To coordinate.

Finally variable P2 the projection code is set to 1 if the line is on the screen and made 0 if it is not. Upon completion of the Clipping mathematics the Projection Subroutine is called. This routine first checks to see if either Z1 or Z5 is zero and if they are prevents the eventual divide by zero by setting them to 0.001. The 3 Dimensional start and end point along with information about the screen width ($W = \frac{\text{Screen Width}}{2} - 1 = 32$) are input and the following mathematics will perform the projection:

$$X2 = (X1/Z1) * W$$

$$X2 = (Y1/Z1) * W$$

$$X4 = (X5/Z5) * W$$

$$Y4 = (Y5/Z5) * W$$

In this way, dividing each start and end point of a line that falls within the viewing pyramid by the depth and then multiplying by the half width of the monitor screen, a true perspective image will be obtained.

The order of the mathematical manipulations of Rotation, Translation, and Clipping is important in that different orders of the routines will result in different displays of a point on the screen. It is, therefore, imperative that the updated position

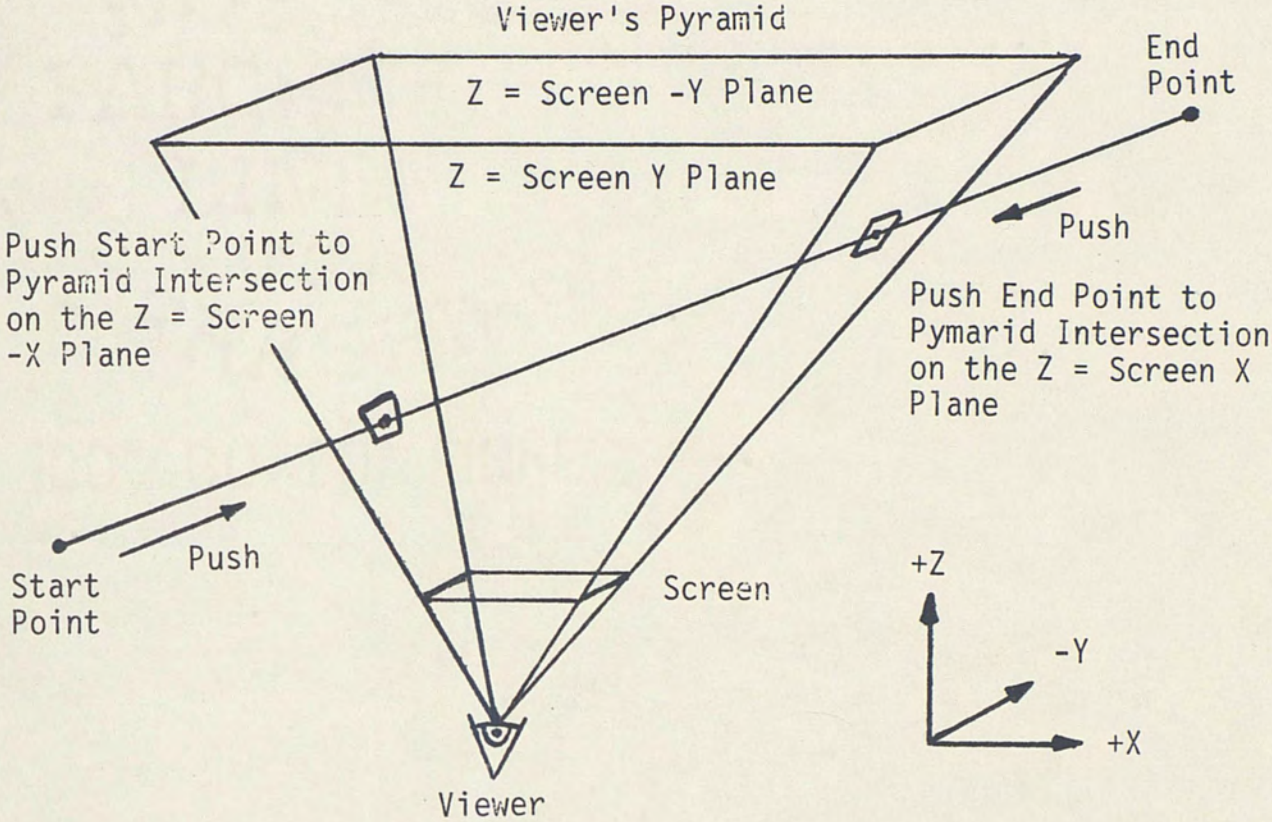


Figure 6. A Line in Need of Clipping

variable information be allowed to be input at this same point in the algorithm as the initial variables were so that the same program flow and hence order is followed. If this were not so the same scene could produce a different display each time the routine was used and displayed. A good summary of what is required of a graphics algorithm can be found in Williams (1972) and specifics on the needs of a graphics manipulating language is available in Takasawa, Moriguchi, and Sakamaki (1972).

When the 2D coordinates are acquired for each of the 192 elements of the database array the new display is drawn. After all the elements have been individually and appropriately Transformed Clipped, and Projected the program is directed to decide again if the 6 input variables have changed. If they have not changed a manual input is requested. If they have changed the screen will be erased and a new Transformation Matrix will be generated.

The new software package encodes the 8 clipping flags C0-C7 instead of C1-C8. The field of view parameter is identified by "AV" rather than "V" and "SCRN" is the parameter used for screen width rather than "W". Also, the new algorithm does the required manipulations in the same order as the Basic algorithm; however, the routine uses the 2's complement math processor containing a fast multiplier subroutine. The multiplier subroutine does not check for resulting overflow conditions which result in the occasional display of points on the wrong side of the scene. The mirrored image display distortion from not implementing the function of the variable P2 is tolerated so that increased processing speed may be obtained.

The final difference in the assembly language routine is that it will return to the point in the routine where the updated parameters from the simulator are obtained automatically. The new input values are compared to the previously stored values. This feature allows the flight simulation to proceed until a landed condition that is, generated simulation altitude = 0 feet, has occurred.

CHAPTER IV

A NEW IMPLEMENTATION

This chapter will explain the details of the Flight Simulator Assembly Program. Although the basic structure of the program flow is similar to the BASIC algorithm, there are some significant departures from the previous implementation. This is due to the M6800 machine architecture which only allows instructions based on an 8-bit word. The BASIC program operating system allows floating point mathematics, and ease in defining, dimensioning, and manipulating arrays within a single command line. It is rather obvious therefore, that a single command line from the BASIC program may take many lines of ASSEMBLY code to perform the equivalent task.

The ASSEMBLY code is contained in three object files OJFLITA, OJFLITB, and OJFLITC on a floppy disk. (The source files are each on a separate disk and are respectively FLITA, FLITB, and FLITC). In addition, the Sublogic 3D to 2D program and airport data bases are presently stored on paper tape. The tape can be loaded after the disk object files are loaded by accessing the SWTBUG monitor, and reading it into memory from the teletype paper tape reader.

From the flow chart provided in Figure 7, it can be seen that five reserve memory arrays must be filled. The format of the Control Array, Input Buffer Array and Output Buffer Array are shown in Figure 8. The Control Array

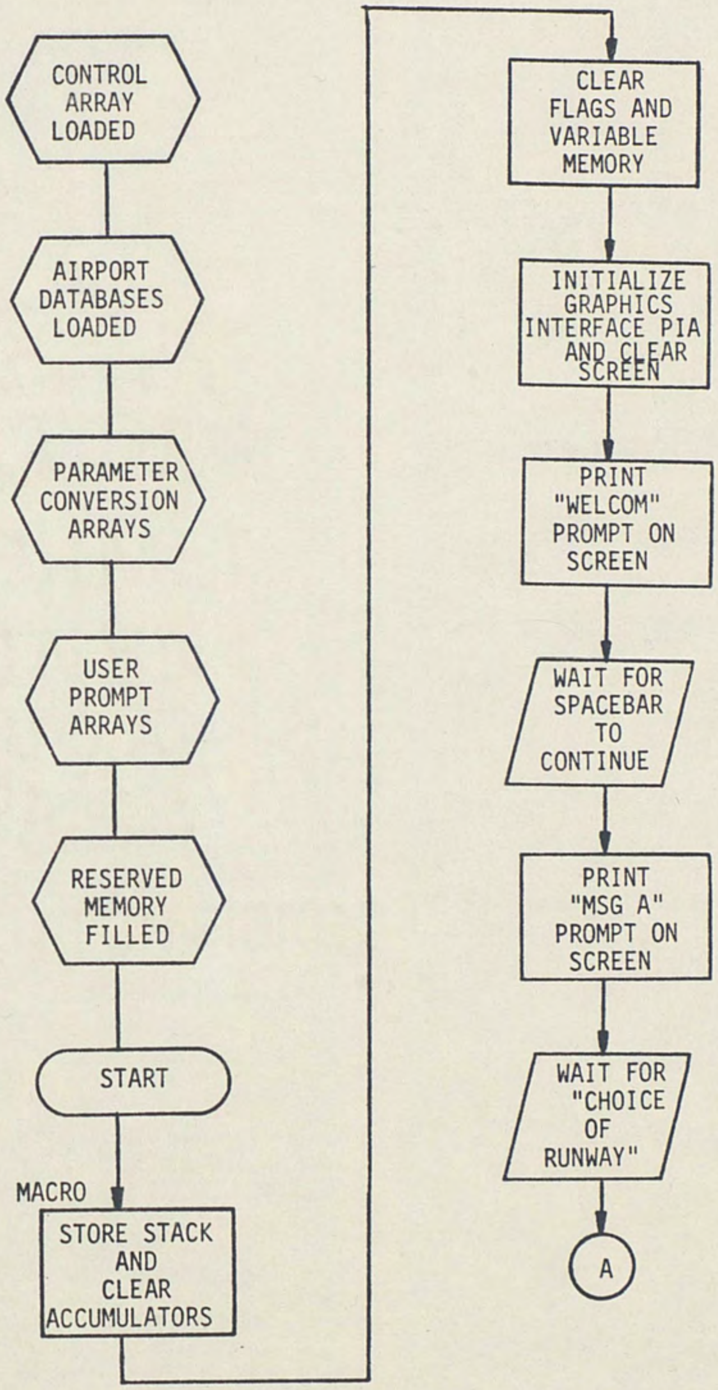


Figure 7. Flow Chart of FLITSIM Assembly Program

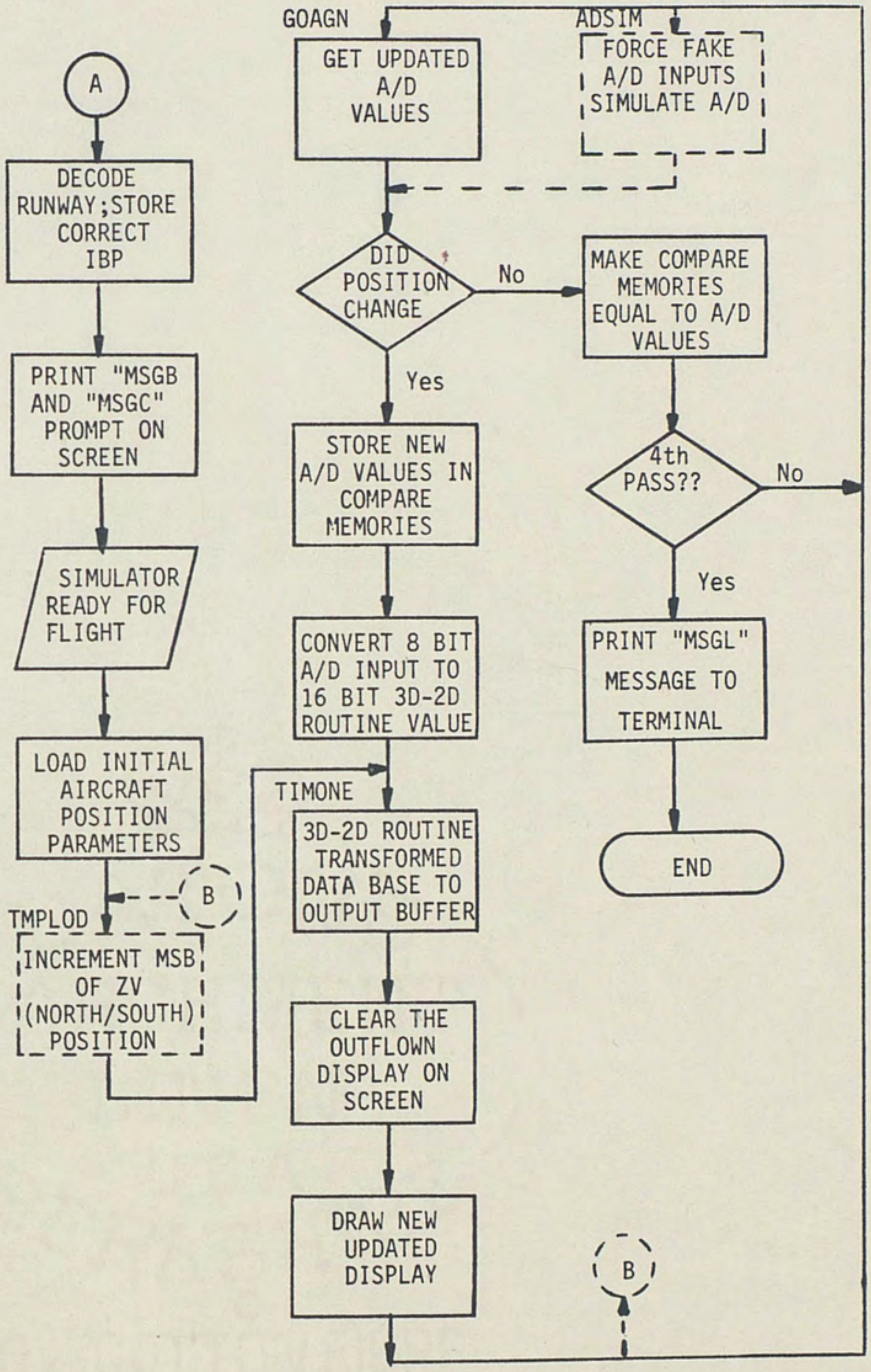


Figure 7. Continued

THE CONTROL ARRAY

Address		Parameter
0100,0101	XV)	Viewer's Location
0102,0103	YV)	In Space
0104,0105	ZV)	
0106,0107	PV	Viewer's Pitch
0108,0109	BV	Viewer's Bank
010A,010B	HV	Viewer's Heading
010C,010D	AV	Field of View
010E	SCRN	Screen's Width
010F,0110	IBP	Input Buffer Pointer
0111,0112	OBP	Output Buffer Pointer

The Input Array

Address		Value
1300		Code
1301	1302	X
1303	1304	Y = 0000 Always
1305	1306	Z
1307		Code
1308	1309	X
130A	130B	Y = 0000 Always
130C	130D	Z
130E		Code

If the Code =

25 Hex A Start Point is defined.
26 Hex A Continue to or End Point is defined.
Any other value = End of Array

The Output Array

Address	Value
1800	Code
1801	From X
1802	From Y
1803	To X
1804	To Y
1805	Code
1806	From X
1807	From Y
1808	To X
1809	To Y
180A	Code

If Code is 55 Hex Line is to be projected on screen. If it is any other value, it signifies the end of the array.

Figure 8. The Format of the Arrays

consists of the memory locations reserved for the six 16 bit positional parameters, the 16 bit Field of View parameter, the 8 bit Screen Width parameter, and two pointers. These pointers identify the start addresses of the airport database array chosen and the scratch pad memory which will be filled by the sequencing program upon transformation of each line to be displayed.

The Control Array resides in memory locations 0100 to 0112 and have the following restrictions:

XV, YV, ZV - represent the pilots X (West/East) position, Altitude, and Z (South/North) position respectively. They are 16 bit two's complement values (± 32767 decimal units) located in memory at addresses 0100 to 0105 with the lower address containing the most significant byte. The maximum "world" size capability is therefore 1252 cubic nautical miles assuming 1 foot = 1 bit scaling. West and South values will fall in the 0000 - 7FFF hex range, while East and North will range from hex FFFF - 8000.

PV, BV, HV - represent the viewers attitude parameters of Pitch, Bank, and Heading. Although they are also 16 bit values the most significant byte is always zero. The allowable range of +127 to -128 units is scaled for an angular range of +178.59 degrees to -180 degrees. This is equivalent to 1.40625 degrees per bit. These parameters are located at memory addresses 0106 to 010B and also have the most significant byte at the lower address.

AV - is the Field of View parameter and is a double precision, 2 byte, two's complement value ranging from 0 to 32767 decimal. The maximum value 32767 represents a 45 degrees half field of view or 90 degree full field of view. The present program will always use this maximum value, however, by performing the calculation:

AV (decimal) = 32767 * tangent of ½ Full Field of View angle desired

either wide angle or telescope views can be generated. Negative fields of view will cause mirror images of the field behind the viewer to be projected and hence are not to be used. AV is located in memory at addresses 010C and 010D.

SCRN - is the single unsigned value of screen width of the display device and can range from 0 to 255. The present program assigns decimal 64 to this parameter because a square screen is assumed by the 3D to 2D converter program, so that the minimum screen pixel dimension must be used. The GT-6144 Graphics Interface has a 64 x 96 pixel video output resolution.

IBP, OBP - are both address pointers and hence unsigned 16 bit values. They are located respectively at addresses 010F, 0110 and 0111, 0112. The Input Buffer Pointer, "IBP" will point to the starting address of the chosen airport database, while "OBP", the Output Buffer Pointer is the pointer to the scratch pad memory which contains the transformed 2 dimensional display data.

The two Airport databases are loaded into memory locations 1300 to 1362 and 1100 to 1296. They represent two configured runways at the same "World" location. The user is able to choose the smaller database which allows faster display times to occur on the screen, or the larger database which contains a land grid and a building in addition to the runway. The 3D to 2D converter routine requires that the screen origin be placed in the center of the screen, however, the GT-6144 Graphics Interface requires it to be at the bottom left. For this reason a coordinate translation is required. The airport runway is located at hex 3D coordinates X = 2738, Y = 0000 and Z = 2738.

The Input Buffer database arrays are formatted as follows:

Control Word: This is an 8 bit code which is either Hex 25 designating a "Start" coordinate point follows, or Hex 26 designating a "Continue To" coordinate follows. Any other hex value in this array position will signify the end of the array.

X, Y, Z Position: These are each 16 bit double precision two's compliment values of the airport's boundary lines and must follow the control word in order. The Y value, Altitude, is always zero and the Z value represents a direction into the screen, in our case South (before the screen)/North (further into the screen). The X position is, of course, our West/East coordinate.

It should be remembered that the most significant byte of the pair is located in the lower address location.

The Parameter Conversion Arrays are designed to produce the proper 3D to 2D routine value for each of the 6 positional parameters of the viewer. The X, Y, and Z coordinates of the pilot's location within the world and his altitude parameters of Pitch, Bank and Heading are each input to the computer via 6 analog signals. These signals are prescaled by the Pre-Amp Board and then applied to the Innovative Technology AD-68A Analog to Digital Converter Board within the computer. The AD-68A card is polled by the computer to allow one of the analog signals to be converted to an 8 bit value. (Maximum value of FA and a minimum value of 00). This means 250 different values for each parameter are allowed and must be scaled to both the "Simulator's World" and the "3D to 2D routines World."

The X (West/East) and Z (South/North) inputs are scaled in the same way. Recall that the Airport Database's front left corner is located at:

X = +10,040 feet (2738 Hex)

Z = +10,040 feet (2738 Hex)

from the Display Terminals origin. Each bit increment is 1 foot and a range of +32,767 (7FFF) to -32,768 (8000) feet is allowed. The Viewer's coordinate system origin is located in the same place as the Display Terminals' origin within the "Graphics Display World," however the Viewers position is defined as follows:

(8000) = -32,768 feet = Max North or East Position

(FFFF) = -1 foot = Min North or East Position

(0000) = 0 feet = Origin

(0001) = +1 foot = Min South or West Position

(7FFF) = +32,767 feet = Max South or West Position

Imposing the 3D Viewers coordiante system over the Display Terminals coordinate system (i.e. Overlay the 3D Viewer coordinate system on top of the origin and axes already defined by the Graphics Display Terminal) the pilot would have to be at his 3D coordinate position:

X = -10,040 feet (D8F0) East of Origin

Z = -10,040 feet (D8F0) North of Origin

to be at the front left corner of the runway. The illustration in Figure 9 shows this relationship. The A/D values input from the simulator are 8 bit bytes and range from 00 to FA Hex. (250 values allowed). This range is defined as follows:

FA = +28,500 feet = Max feet South or West

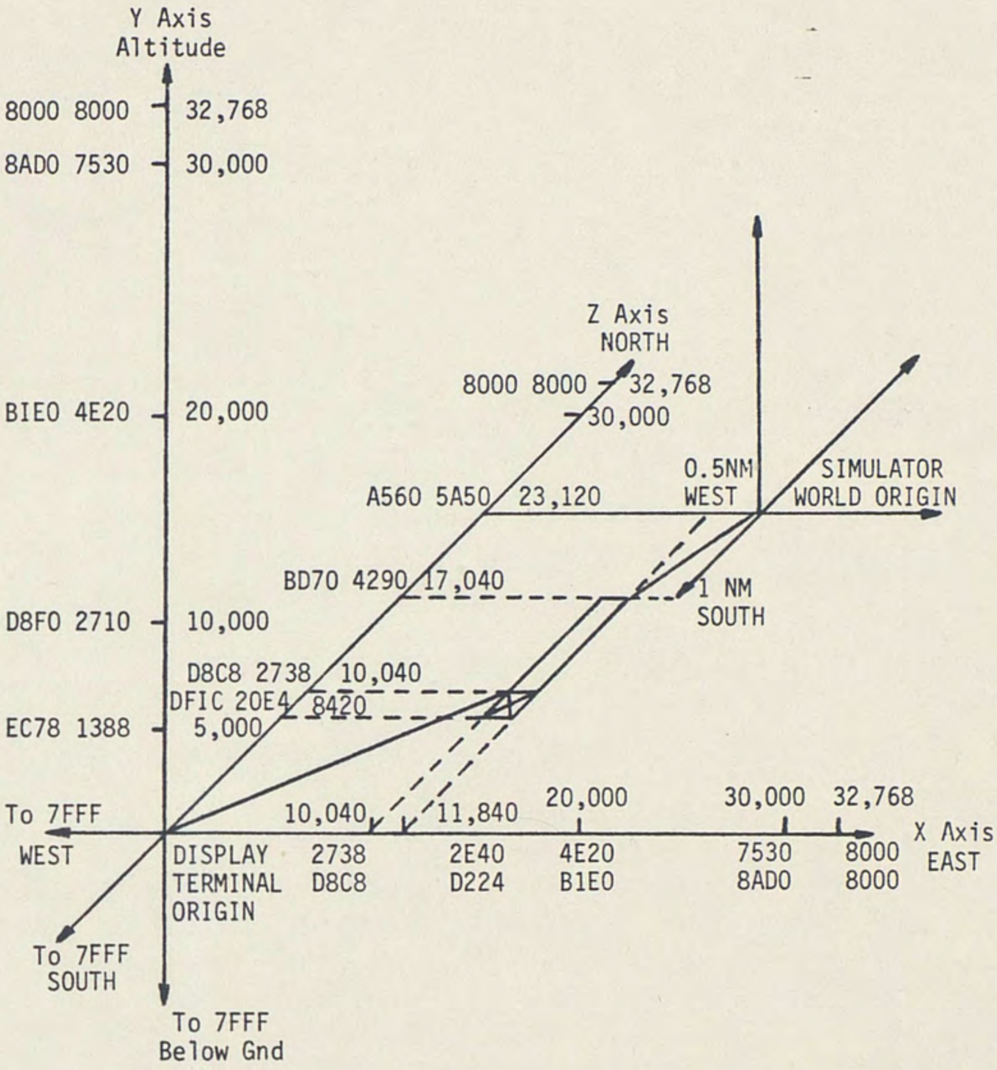
7D = 0 feet = Simulator World Origin

00 = -28,500 feet = Max feet North or East

Each bit increment input from the A/D is equal to 228 feet. The A/D origin is located 1.0 nautical mile North and 0.5 nautical miles East of the back right corner of the airport's runway. Knowing that, 1 nautical mile = 6080 feet,

$$\frac{6080 \text{ feet/nautical mile}}{228 \text{ feet/bit}} = 26.67 \text{ bits/nautical mile}$$

$$\frac{3040 \text{ feet/nautical } \tfrac{1}{2} \text{ mile}}{228 \text{ feet/bit}} = 13.33 \text{ bits/nautical } \tfrac{1}{2} \text{ mile}$$



Dimensions:

Top or Right of Axis = Decimal Distance from Graphics Display Terminal Origin

1st Left or Below Axis = Hex Distance from Graphics Display Terminal Origin

2nd Left or Below Axis = 3D/2D Converter Routine Viewer Coordinate System

Simulator World Size is ± 28,500 feet in X and Z, +5,000 feet Altitude Y.

3D/2D World Size is +32,767 feet, -32,768 feet in X, Z, and Altitude Y.

Scale: 1/8" = 1,000 feet

Figure 9. Relationship of Coordinate Systems

are the numbers which must be subtracted from the A/D Z and A/D X values input to perform the final translation for the airports' location within the "Simulator's World." A portion of the table which converts from the A/D input value to the 3D conversion routine values is shown:

Analog Volts	Already Subtracted A/D Input	Simulator X,Z Distance (Feet)	3D Hex Value Required	3D Decimal X,Z Coordinate (Feet)
2.24	E1	+22,800	3200	+12,800
1.69	A9	+10,032	0020	+32
1.26	7E	+228	D9D4	-9772
1.25	7D	0	D8F0	-10,000
1.24	7C	-228	D80C	-10,228
0.81	51	-10,032	B1C0	-20,032
0.25	19	-22,800	8000	-32,768
0.00	00	-28,500	8000	-32,768

Notice that the A/D Input 7D is on the simulator's axis but is -10,000 feet from the "3D to 2D routine World Axis." This in turn limits the maximum positive 3D coordinate location (South or West) to +12,800 feet from the 3D origin and the maximum negative 3D location (North or East) to -32,768 feet from the origin.

The Y value is the Altitude parameter. The Simulator allows a 5000 foot ceiling while the 3D to 2D converter routine again ranges from +32,767 to -32,768. The 3D value is distance from the viewer's position so that a positive value (0000-7FFF) is distance above the pilot and negative values (FFFF-8000) are distances below the viewers position. The maximum distance below the pilot the airport can be is 5000 feet or EC4A hex. Since this parameter's analog voltage is nonlinear, it is not possible to assign a constant relating the number of feet per .01 volts. Instead, the nonlinearity requires that each 1000 foot interval is scaled independently. Representative values follow:

Analog Volts	A/D Input	Simulator Altitude	3D Hex Value Required	3D Viewer To Ground
2.50	FA	0 Feet	0000	0 Feet
1.70	AA	1000 Feet	FC18	-1000 Feet
1.30	82	2000 Feet	F830	-2000 Feet
0.90	5A	3000 Feet	F448	-3000 Feet
0.50	32	4000 Feet	F060	-4000 Feet
0.00	00	5000 Feet	EC78	-5000 Feet

The attitude parameters are also input by the A/D card and the 8 bit digital word received is a voltage representation of an angle. The 3D to 2D converter routine is set up for a 16 bit double precision two's complement value to be received. The most significant byte (lower address) is always a zero and the least significant byte has a range as follows:

Max = 7F = 178.59 degrees

Mid = 00 0 degrees

Min = 00 = - 180 degrees

This means each bit increment is equivalent to 1.4065 degrees.

The Pitch parameter is the aircrafts position with respect to a transverse axis (nose up or down condition). The simulator limits the angular range from +90 degrees (nose straight up) to -90 degrees (straight dive or down). Each 0.01 volt input or bit increase corresponds to 0.720 degrees. The array is set up as follows:

Analog Volts	A/D Input	Simulator Pitch (Degrees)	3D Hex Value Required	3D Pitch (Degrees)
----	--	---	007F	178.59
0.00	00	-90	0040	90
0.62	3E	-45	0020	45
1.25	7D	0	0000	0
1.89	BD	45	00E0	-45
2.50	FA	90	00C0	-90
----	--	---	0080	-180

Notice that a positive A/D Input (nose up) corresponds to a negative 3D value (nose up, scene moves down).

The Bank parameter is the aircraft's lateral inclination from the horizon, sometimes called Roll. It also allows for a -90 degree (left wing straight down) to +90 degree (right wing straight down) simulator value range, and therefore is also scaled for 0.720 degrees per 0.01 volts. The array is defined as follows:

Analog Volts	A/D Input	Simulator Bank (Degrees)	3D Hex Value Required	3D Bank (Degrees)
----	--	---	007F	178.59
0.00	00	90	0040	90
0.380	26	60	002A	59.06
0.755	4C	30	0015	29.53
0.907	5B	20	000E	19.68
1.08	6C	10	0007	9.84
1.25	7D	0	0000	0.00
1.40	8C	-10	00F9	-9.84
1.55	9B	-20	00F2	-19.68
1.68	A8	-30	00EA	-30.93
2.13	D5	-60	00D5	-60.46
2.50	FA	-90	00C0	-90
----	--	---	0080	-180

The final parameter input is the Heading. Heading is the viewers looking direction within the world with respect to the XY axis Z plane (i.e. due North would be a 0 degree or 360 degree heading). The simulator will allow a range from +90 degrees (looking due East) to -180 (looking due South while turning from the West). Hence, 270 degrees of the 360 degree full circle are accounted for, which is divided by 250 possible values. This makes each bit equivalent to 1.08 degrees. The array is defined as follows:

Analog Volts	A/D Input	Simulator Heading (Degrees)	3D Hex Value Required	3D Heading (Degrees)
----	--	----	007F	178.59
0.00	00	90	0040	90
0.38	26	45	0020	45
0.86	56	0	0000	0
1.28	80	-45	00E0	-45
1.68	A8	-90	00C0	-90
2.00	C8	-135	00A0	-135
2.50	FA	-180	0080	-180

The next group of arrays which must be formatted are the User Prompt Arrays. These five arrays consist of the ASCII values of the alphanumerics used in the messages displayed on the User's Terminal. The messages are designed to first welcome him to the system, allow him to choose a desired Airport Database, instruct him in the initialization of the system and sign off upon completion of the mission. Finally the memory locations which are to be used as system variables are reserved.

The program is actually initiated when the user has completed the "LOAD" procedure (floppy disk and paper tape files), initiated the program counter at memory addresses A048 and A049 to 6000, and then issuing the "G" (go to user subroutine) command from the keyboard while in the SWTBUG monitor.

At hex address 6000 resides the program "MACRO." It is the MACRO's responsibility to call each subroutine as needed to sequence repeatedly through all of the program's requirements. The method of calling routines from a MACRO is useful not only in organizing the tasks to be performed into smaller subtasks, but also in selecting desired branches while debugging.

After storing the stack pointer in reserve memory and clearing both of the accumulators the Macro issues the first subroutine call. Subroutine "CLRMEM" is designed to clear the variable reserve memory locations and flags along with the scratch pad Output Buffer memory. The routine uses two counters "CNT" and "COUNT" which decide where the routine will branch. There are two loops within this routine controlled by the counters. The variable "COUNT" controls the outside loop which will load the starting address of the memory array to be cleared and the B accumulator with the number of address locations to be cleared. The inside loop is controlled by the variable "CNT" which allows the second page of the Output Buffer array to be cleared after completion of the clear of the lower page. When both loops have been completed the variable "COUNT" will equal 3 and the program is directed back to the MACRO for the next instructions.

The Graphics Interface GT-6144 device is controlled by the computer through a (P.I.A.) Peripheral Interface Adaptor printed circuit board. By correctly storing 8-bit bytes to this card over the systems bus lines, the computer is able to format the PIA chip for input and/or output lines and then input or output information to an external device. Subroutine "PIAINT" is designed to initially format the PIA printed circuit board for all outputs and then output the required data to the GT-6144 Graphics Interface terminal to "Erase" the Video Monitor's screen. Both operations are accomplished by loading the A accumulator with a value (range 1 to 6) which selects the desired function. The actual task is then performed by calling subroutine "SCREEN." The six tasks which subroutine "SCREEN" can do are as follows:

1 = PIA Initialization for GT-6144 (output)

2 = PIA Initialization for Joystick Box (input)

3 = Erase Screen

4 = Pixel Output

5 = Show 2

6 = Start Input from Joystick

Subroutine "Screen" resides in memory at location hex 7000. The GT-6144 PIA is initialized when a 1 is loaded into the A accumulator. When "SCREEN" is called the value in the A accumulator is compared and a branch is allowed to direct the program to the code which will perform the desired task. The GT-6144 PIA is located in the third I/O slot of the computer which assigns the "Direction Register" to address 8009 and the "Peripheral Register" to address 8008. The "Direction Register" is formatted for all outputs and then the "Peripheral Register" is readied for output data. The "Screen" routine is then exited and the program returns to the "PIAINT" routine. The A Accumulator is then loaded with a 3 and "SCREEN" is again called. This time the "SCREEN" routine performs the

"Erase" task. The "Erase" routine will set the horizontal position to the far right and then enter a loop which will clear all pixels in a vertical bar moving from right to left. The first time through this loop the vertical bar extends the full pixel height of the screen equal to 96, however, subsequent passes will clear only a pixel height of 64 because the screen is assumed to be square for the 3D to 2D routine. Upon completion of the "ERASE" task the program is directed first back to the "PIAINT" subroutine and then back to the MACRO.

When the PIA has been initialized the user should be informed via message prompts of the details to the setup of the ATC-610J Simulator. Subroutine "PROMPT" will display on the user's terminal the five messages he requires. The first message, "WELCOM," welcomes him to the system and tells him briefly what the system will do. When the user has finished reading the "WELCOM" paragraph he depresses the spacebar. This allows the routine to print "MSGA" which enables the user to choose which airport database he wants to be displayed. After he selects the database "MSGB" is printed which asks him to continue to the simulator initialization prompt "MSGC." This prompt informs the user to set the communication radio frequency to 119.2 MHz and the navigation radio frequency to 110.3 MHz. These are the frequencies developed from the TETERBORO, NEW JERSEY airport. It further instructs the user to set the magnetic compass and the gyro compass to read 360 degrees and the altitude to 1000 feet. Finally, it is desired to set the aircraft's attitude for the "Wings Even" and "Level Flight" condition. The last message "MSGL" will call the "DSPCHR" portion of the routine from subroutine "RELPOS." The routine will first load the index register with the starting address of the next user prompt to be displayed. The user prompts consist of the ASCII values of the alphanumeric characters stored in memory. Subroutine "DSPCHR" is then called which loads the A accumulator with the ASCII value addressed, compares it to the end of message value hex 4, and if it is not equal put the character out to the terminal. It then increments the Index Register and continues in the loop until the hex 4 value is detected. When this end of message value is detected the routine falls out of the loop and waits for a user response from the keyboard. For most messages the response will be the detection of the ASCII value for the spacebar, "A0" after which the routine will load the index register with the starting address of the next prompt array or return to the MACRO. In the case of "MSGA," however, the user must make a choice between which of the two runways he desires to use. In this case the routine is required to detect an input from the user of a

1 = ASCII 31 for the smaller airport database

2 = ASCII 32 for the airport database with the ground grid and the building.

If ASCII 31 is detected, then the program will load the index register with the staring address of the smaller airport database (1300 Hex) and store it in memory location, IBP the Input Buffer Pointer. If the larger database is desired, then 1100 Hex is stored in the IBP. The routine will default to loading the IBP with 1100 if any other ASCII value is received.

Now that the user has initialized the simulator, the "INITLOD" subroutine can be called which will store in memory the correct values for the aircraft's initial position and altitude. It will also load the OBP, Output Buffer Pointer with the correct address (1800) and set the screen width variable AV to hex 40 which is, of course, 64 decimal pixels. The aircraft attitude is initialized for "Level Flight" and "Wings Even" so that the PV and BV variables are both set to hex 0000. The heading of the aircraft is due North because the gyro and magnetic compasses are at 360 degrees. This condition requires a 0000 hex value to be stored in the variable HV. The altitude was set for 1000 feet so the value of -1000 (airport is 1000 feet below viewer hex FC18) is loaded into the variable YV. In order for the aircraft to be in line with the airport runway, the X (West/East) coordinate must be about -11,109 feet from the Display Terminal Origin. This would allow the aircraft to approach the center of the runway. That would mean that the variable XV should be loaded with hex D500. Finally the Z (North/South) coordinate can be loaded with just about any 16 bit two's complement number desired. This is because the aircraft is flying into the airport parallel to the Z axis. The program loads a hex C000 which is -16,384 feet (16,384 feet into screen or North). When the variable memory has been loaded the program returns to the MACRO.

The next MACRO command is identified by the label TEMPLOD and will decrement the most significant byte of the variable named. Any of the 6 positional parameters XV, YV, ZV, BV, PV, or HV may be used here, but ZV is identified in the program. This command coupled with the branch to label TIMONE and then the eventual return branch to TMPLOD allow the program to self loop on itself and display a new scene for each change in the selected variable. This has the effect of keeping all of the other five variables constant and observing the change in a particular variable of interest.

The branch to "TIMONE" allows the call "GO3D2D," for the 3D to 2D Converter routine to take place. This routine developed by Sublogic's Bruce Artwick will generate 2 dimensional start and end points which represent lines to be plotted on the display terminal monitor. Point translation (addition of constants to X, Y and Z) and rotation (multiplication of the 3X3 array by the 3 element vector described in the previous chapter) about the viewers origin are computed. The rotation matrix must be created once per each viewing direction because it will apply to all the points for that view. In the interest of speed, lookup tables are used to compute the sines and cosines of the angular parameters. The line clipping is performed by assigning a 4 bit code to the 2D start point (CO-C3) and a 4 bit code to the 2D end point (C4-C8). The code is formatted as follows:

Start Point	End Point	Point Location
C0=1	C4=1	Left of -X=Z plane
C1=1	C5=1	Right of X=Z plane
C2=1	C6=1	Above the Y=Z plane
C3=1	C7=1	Below the -Y=Z plane

and indicates which side of the viewing pyramid the point lies. The mathematics are performed, if necessary, for a left or right push and the equations are the same as previously described in Chapter 3. Finally the 3D to 2D routine will project the line to the size of screen by performing the division of each space coordinate X and Y by Z (the point's depth). This will allow a true perspective image to be generated on the display device. Before leaving the 3D to 2D explanation it is important to note that in order to increase processing speed, no overflow checking is performed in additions and multiplications. This will cause points which overflow to end up on the wrong side of the screen resulting in display distortions. Detailed information concerning this software package can be found in the Sublogic Manual by Artwick (1977).

When the 3D to 2D routine has completed filling the 2 dimensional (start and end points) Output Buffer array along with the control code the MACRO is able to again call "CLSCRN" which will erase the screen to prepare it for a new display. This is performed by calling subroutine "SCREEN" after first loading the A accumulator with a 3. Subroutine "DBDRAW" is then called by the

MACRO which will draw the transformed 2 dimensional database located in the Output Buffer array as it will look to the viewer. The Output Buffer consists of an ordered series of five 8 bit words which are defined and ordered as follows:

Control Word = 55 as long as there is another line to be drawn on the screen. The detection of any other input signifies the end of the file. It is 8 bits long.

X1 = X coordinate of the start point and is a single precision 8 bit word

Y1 = Y coordinate of the start point and is a single precision 8 bit word

X2 = X coordinate of the end point and is 8 bits long

Y2 = Y coordinate of the end point which is also 8 bits long

The routine loads the index register with the starting address of the array and detects a hex 55 which allows it to continue. It then increments the index register and loads the values contained in locations X1, Y1, X2, and Y2 in preparation for the eventual output draw. The accumulator A is then loaded with a 5 and the routine will call the "SCREEN" subroutine which is the "SHOW2" option. The eventual branch to "SHOW2" is followed immediately by a branch to "SHOW" which will draw the line from (X1, Y1) to (X2, Y2). Upon completion of that lines draw, the routine will return to the label "FEED" which will again compare the A accumulator to hex 55. When the routine does not find a hex 55 in this array location it will return back to the MACRO. The "MACRO" then allows a choice of three possible branches for the continuation of the ongoing simulation. The first branch to label "TMPLOD" will allow the program to change the value of one 3D to 2D routine 16 bit two's complement variable (XV, YV, ZV, PV, BV, or HV) of interest. This enables the program to loop on itself indefinitely so that the user may observe that variables effect on the output display.

The second choice for a branch, "ADSIM," allows the program to be directed to a subroutine named "STAPAR." This routine will allow the user to manually fill each of the six A/D positional inputs for the purposes of providing a static display of one position in space. The A/D variables which have been converted to 3D to 2D routine values are then easily tested to allow the verification of the correlation between the simulator position and the 3D to 2D routine display.

Finally, the last branch to "GOAGN" is the normal operating loop for the program. This loop allows the A/D card to be polled, stores the new data in memory and determines if the relative position of the viewer has changed. If so, then the routine will convert the A/D values into 3D to 2D routine expected values for each of the six parameters. The routine will then rejoin the program at label "TIMONE" and proceed as before until the aircraft has landed or the routine is stopped.

In order for the different paths of branching to be allowed the programmer must manually modify memory by entering the code for a NOP (hex 01) from the monitor's Memory Examine feature for each of the branch statements that are not desired.

Subroutine "UPDATE" is designed to poll the six input channels from the AD-68A card and save an 8 bit voltage value representing the analog voltage detected. The routine will store in temporary variable "ENABLE" the code which specifies the desired channel to be converted. The choices are as follows:

Decimal Code Required	Bit True or Channel	Variable Input from Channel Selected
16	4	X3 (West/East Coordinate)
32	5	Z3 (South/North Coordinate)
2	1	Y3 (Altitude Coordinate)
1	0	PITCH
8	3	BANK
4	2	HEAD

After storing the decimal code required to enable the channel of interest, the program will call subroutine "GETNEW." This routine will enable the A/D-68A card by loading hex 80 into memory address hex 800C. This action will initiate a ramp loop starting from zero which successively increments the value of the B accumulator. This value is applied as an increasing voltage to a voltage comparator whose other input is the enabled analog channel voltage. When the comparator changes state or the maximum value for the B accumulator (FA) is obtained the value of the B accumulator is saved in memory location "NEWDAT." A reset is then performed by addressing the card (hex $800C) and storing a hex 40 to it's control register. The routine then returns to "UPDATE" where "NEWDAT" is stored in the appropriate variable memory location.

When all six channels have been polled , the routine returns to the MACRO where a branch is performed to label "GONORM." Label "GONORM" within the MACRO will call subroutine "RELPOS." This routine determines if the relative position of the aircraft has changed while the program was off calculating the transformations for the last displayed database. The newly received A/D values are stored by routine "UPDATE" in memory locations X3, Y3, Z3, PITCH, BANK and HEAD. Each of these newly updated parameters are compared to its counterpart from the previous run through. These memory locations are, respectively, M1, N1, N2, N3, N4 and N5. If any one of the new values differs from its previously stored counterpart, the routine will return to the MACRO where it will be allowed to produce a new display for the viewer. If all of the values received equal the values from the last pass , then the aircraft has not changed its position. In this event, memory location "FLGDUN" is incremented and then compared to the number 4. If "FLGDUN" equals 4 then it signifies that the routine has tried to obtain a change in the viewers position four times unsuccessfully, and the program will print the signoff message "MSGL" to the screen. Upon completion of the "Landed Message" printout to the user's terminal a software interrupt is accomplished which terminates the user program activity. Finally, if the flag, "FLGDUN" does not equal 4, then the routine will jump back to the MACRO's label "GOAGN" which will again try to "UPDATE" the aircraft's position by polling the A/D for the most recent aircraft position.

With a successful "UPDATE" and change in "RELPOS" the MACRO will be allowed to call subroutine "CNVRT" whose responsibility is to convert the 8 bit newly received A/D values into 16 bit scaled units required by the 3D to 2D routine. Another important requirement of this routine is to refresh the "last run" positional parameters (M1, N1-N5) with the newly received A/D values obtained. Finally, the routine is responsible for the X and Z "A/D World" translation and it will accomplish this by subtracting the constants decimal 13 from the X position

13 x 228 ft/bit = 2,964 feet which is approximately
0.5 NM West

and decimal 27 from the Z position.

27 x 228 ft/bit = 6,156 feet which is approximately
1.0 NM South

since the XV, ZV, and YV, 3D to 2D parameters are all two's complement 16 bit values, the routine will branch to label "XYZ" for conversion of each of these parameters. The updated value (X3, Z3, or Y3) is stored in memory location "CNT" and the index register is loaded with the starting address of the appropriate conversion array (X and Z are scaled the same and hence both will use conversion array XZ3XZV. Y the altitude uses array Y3YV). The call to XYZ will then increment the index register twice for every increment of memory location "CNT" until the value of "CNT" equals the value of the A/D parameter being converted and stored in the A accumulator. When they are equal, the index register will be pointing to the lower address - most significant 8 bit byte value required by the 3D to 2D routine. The A accumulator is then filled from this memory location and the B accumulator is filled with the higher address - least significant 8 bit byte of the 3D to 2D value. The routine will then return to the place within the "CNVRT" routine from which it was called and store the 3D to 2D values stored in the A and B accumulator to the respective 3D to 2D required parameter memory location. The 8 bit attitude parameters PITCH, BANK, and HEAD are all also 16 bit two's complement values when converted to PV, BV, and HV as required by the 3D to 2D routine. However, the most significant byte of these 3D to 2D values must always be filled with a hex 00. For this reason the routine "PBH" when called will accomplish the same function as the routine "XYZ" but will only have to increment the index register once for each time "CNT" is incremented. Since the PITCH, BANK, and HEAD A/D inputs are all scaled differently a different conversion array was required for each parameter. The starting address for the pitch conversion array is "PTCHPV" while the address for the bank conversion array is "BANKBV" and the address for the heading conversion array is "HEADHV."

The only remaining subroutine to be discussed is "STAPAR." This routine simply loads a constant for each 8 bit A/D input parameter into its memory location for the purpose of displaying a stationary view of the display. This routine is the equivalent of a hovering helicopter viewing the runway, where all 6 positional parameters remain frozen and is useful for correlating the A/D input values to the converted 3D to 2D required value.

The routine listing and a memory map may be found in Appendix B along with the 3D to 2D routines' memory dump and the airport database's memory dump. Appendix C contains a chart showing the relationship between A/D input voltages and the required 3D to 2D converter routine format for an input to be recognizable.

CHAPTER V

CONCLUSION AND SUGGESTIONS FOR FURTHER STUDY

The new implementation presented in this report is very close in form to the BASIC algorithm from which it was derived. The new system allows the available resources to be used in the most efficient mode under the control of "machine language." Improvement becomes evident in the areas of speed and the user interface. The BASIC algorithm although easily programmed required approximately 2 to 3 minutes for a new display. The ASSEMBLY version allows a new display every 2 to 3 seconds or faster depending on the size of the airport database being transformed. The OPERATOR INTERFACE in the ASSEMBLY version is more friendly because it allows instructional prompts to be displayed that guide the user through the setup of the system. It also allows the user to choose between two airport databases located at the same "World" location. This ASSEMBLY version, however, is quite long and cumbersome. For that reason it can be difficult to follow and debug. Finally, the resolution of ASSEMBLY package is not as good as the BASIC algorithms floating point package; however the reduced resolution has a negligible effect on the operation of the system.

Several suggestions come to mind for further research in this area. These suggestions fall into the obvious categories of "Hardware Improvements" and "Software Enhancements."

Hardware Improvements could be made in several ways. The selection of a computer to use inherently defines the accuracy and speed. The tradeoff very obviously is controlled by the cost of the system. New technology has provided machines which split the gap between the 8 bit micro and the 16 bit mini computer in cost and features. These new controllers feature floating-point binary Arithmetic Logic Units, 16 bit or more I/O's and other features which gear the processor towards a specific application. As technology advances the selection becomes more numerous and will provide more complexities at a cheaper cost.

The existing system presently requires the program to be loaded via the teletype paper tape reader at 110 baud. This necessary evil requires 20 to 30 minutes of the user's time before he is able to start the program. A dedicated firmware ROM which contains the program would eliminate this source of inconvenience.

The system's resolution could be improved if a 12 or 16 bit Analog to Digital Converter could be used. In addition, if the digital output provided was in double precision two's complement form, the requirement for the conversion arrays for each of the 6 positional parameters and the "CNVRT" subroutine would be eliminated allowing those memory locations to be used for more enhancements.

The most obvious problem with the software package of this new implementation is the "overflow distortion." This is caused by the 3D to 2D sublogic package "FSTMLT" routine located in memory location hex 0200. Since this routine does not check for overflows and is used for both addition and multiplication, overflows that do occur will end up on the wrong side of the display scene causing distortions. At the expense of display turn around times an enhancement which checks for this condition could be implemented.

Another enhancement might be the implementation of a routine which will drive a VOTRAX-type voice synthesizer machine to allow messages to the user to be spoken rather than read from his user's terminal. The VOTRAX unit could also be programmed to simulate the normal TOWER communications that a pilor might receive. A random number generator could be used in calling these messages to provide differing instructions for each mission's approach to the airport.

The present system allows a choice between two display databases located at the same "World" location. If databases could be configured for multiple locations it would be possible to fly a complete mission. That is, it would be possible to takeoff from one airport and land at another. This implementation would require a conversion array for each parameter, and a database array for the airport located within the "area of play" of the system. Unless the requirement of the parameter conversion arrays was eliminated by the use of a 16 bit two's complement Analog to Digital converter, this enhancement might not be achievable due to the amount of memory locations required.

Other software enhancements might allow the "Student Pilot" to receive updated parameter information in the foreground of his display monitor. In addition an algorithm could be generated which would allow the computer to predetermine a best approach for a landing and flag the student if he develops an incorrect attitude (relationship of plane to landing strip - not personality of pilot) while approaching, implementing an autopilot capability. Finally, more complex algorithms might be allowed which would simulate drag factors, wind direction, and malfunctioning instruments. The possibility even exists for a "Trainer Pilot" to drill a "Student Pilot" on emergency procedures by allowing an input to occur from a polled keyboard while the simulation is proceeding. The possibilities for improvement are only limited by time, cost, memory size and imagination.

The author's specific contributions included integrating the following elements into an operational flight simulator system driven by the Motorola M6800 microcomputer:

1) the ATC-610J Flight Simulator,
2) a custom designed Pre-Amp printed circuit board,
3) Model AD-68A Analog-to-Digital printed circuit card,
4) the GT-6144 Graphics Interface,
5) the 3DG68.V3.1 Graphics Driver routine package.

This integrated system provides the University of Central Florida for the first time with a Flight Simulator system which provides visual cues on an aircraft's position in space.

APPENDIX A

"FLITSIM" M6800 BASIC

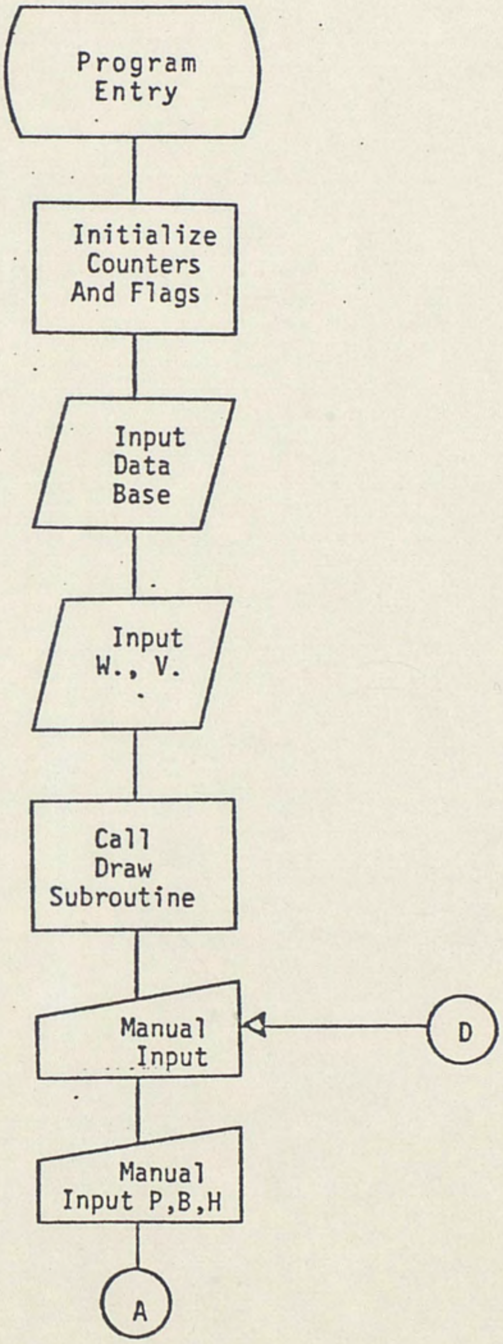


Figure 10. Flow Chart for the BASIC Flight Simulation Program
(Taken from Campbell, 1979)

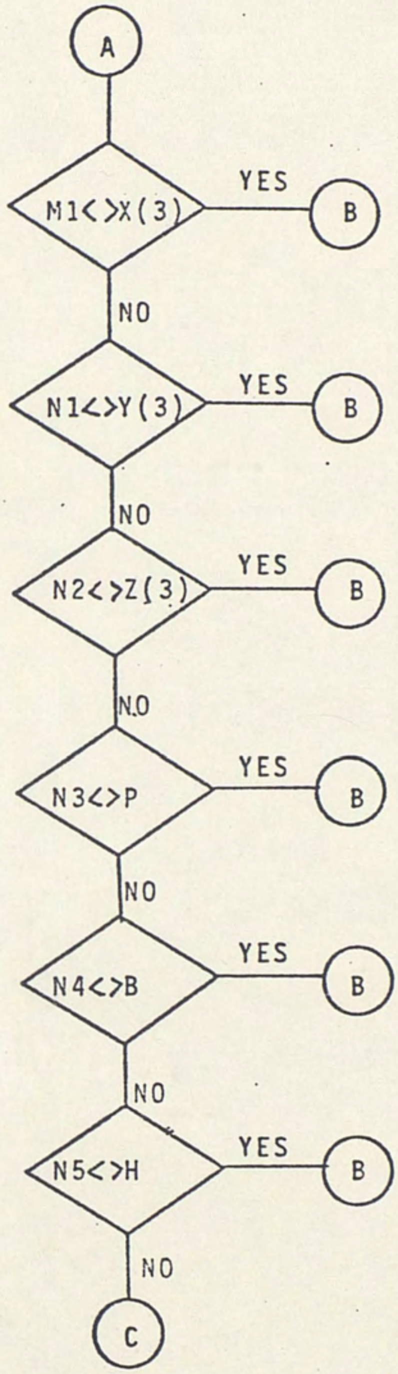


Figure 10. Continued

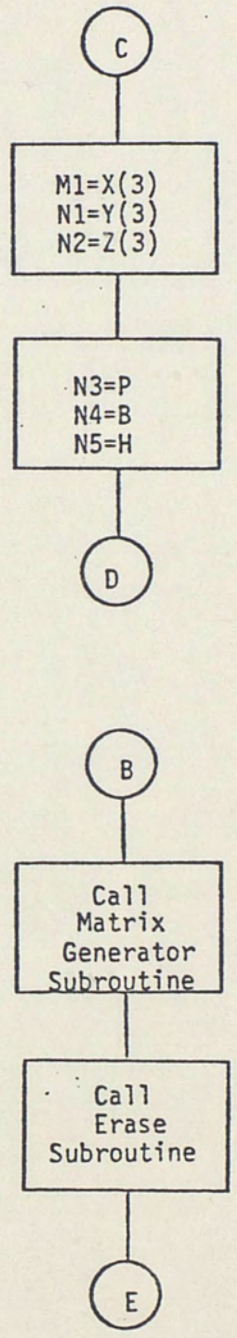


Figure 10. Continued

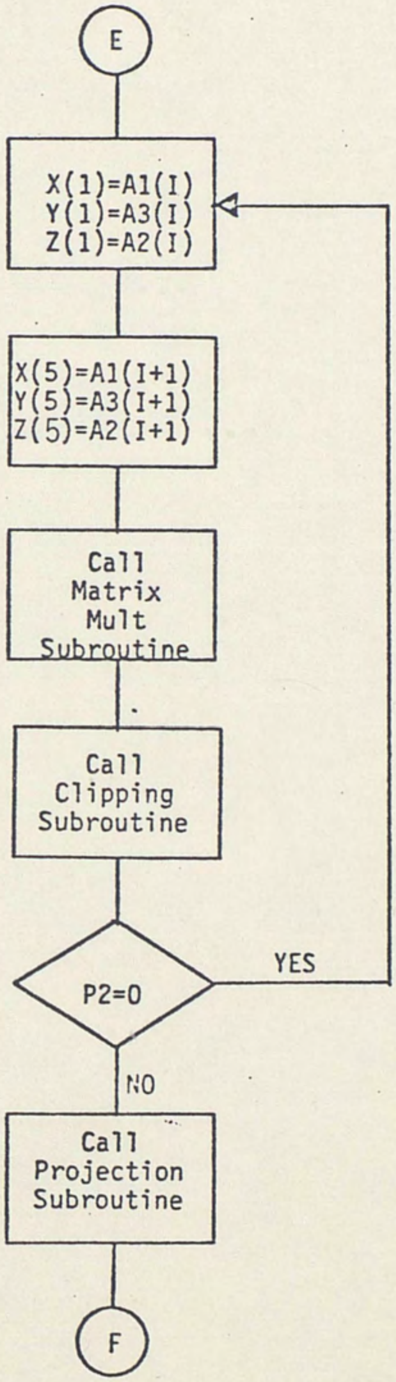


Figure 10. Continued

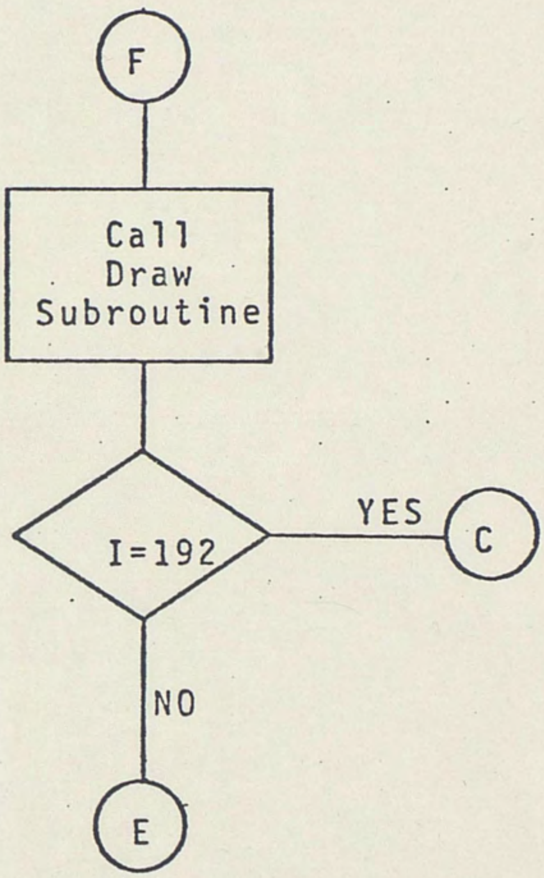


Figure 10. Continued

```
0003 POKE (103, 112)
0004 POKE (104, 00)
0005 LET A=USER (1)
0006 LET A=USER (3)
0007 LET A=USER (2)
0009 DIM A1(192),A2(192),A4(192),A5(192)
0010 DIM A7(192),A8(192),C(8)
0011 M1=0
0012 N1=0
0013 N2=0
0014 N3=0
0015 N4=0
0016 N5=0
0020 FOR I=1 TO 192
0030 READ A1(I)
0040 NEXT I
0050 FOR I=1 TO 192
0060 READ A2(I)
0070 NEXT I
0080 DATA -5,2,2,2,2,3,3,3,3,4,4,3,3,3,3,0,0,-4,-4,-8,-8,
     -8,-8
0081 DATA -5,23,29,29,31,31,31,31,28,28,22,22,20,20,20,20,
     23
0082 DATA 19,10,10,10,10,28,28,28,28,25,25,10,10,10,10,8,
     8,-10
0083 DATA -10,-12,-12,-25,-25,-25,-25,-16,-16,-16,-16,-27,
     -27
0084 DATA -27,-27,-19,-19,-14,-14,-4,-4,-3,-3,11,11,19,4,
     4,4,3
0085 DATA 3,-6,-6,-6,-6,-4,-4,4,-8,-10,-10,-11,-11,-19,-19,
     -19
0086 DATA -19,-17,-17,-8,4,4,4,1,1,1,1,4,-15,-14,-14,-17,
     25,24
0087 DATA 24,25,25,26,26,27,27,26,22,23,23,24,24,23,23,24,
     24,25
0088 DATA -13,-11,-9,-7,-4,-3,0,2,5,7,11,13,17,19,21,23,
     25,27
0090 DATA -23,-24,-24,-22,-22,-21,-23,-22,9,10,10,12,12,
     11,11
0091 DATA 10,11,13,-19,-17,-15,-13,-11,-9,-4,-3,-2,0,2,4,
     8,10
0092 DATA 13,15
0100 DATA -31,-31,-31,-29,-29,-28,-28,-27,-27,-26,-26,-25,
     -25
0101 DATA -24,-24,-21,-21,-21,-21,-25,-25,-28,-29,-31,22,
     22,22
0102 DATA 24,24,27,27,30,30,30,30,28,28,25,25,22,-31,-22,
     22
0103 DATA -3,-3,15,15,21,21,21,21,7,7,18,18,18,18,0,0,0,0,
     13
```

Figure 11. Basic FLITSIM Program Listing
(Taken from Campbell, 1979)

```
0104 DATA 13,4,4,-4,-4,-6,-6,-17,-17,-19,-19,-27,-27,-27,
     -27,-17
0105 DATA -17,-17,-17,-31,-31,-31,0,4,4,4,4,-5,-5,-6,-6,
     -8,-8
0106 DATA 0,-12,-10,-10,-10,-10,-18,-18,-19,-19,-21,-21,
     -12,-16
0107 DATA -9,-9,-12,-12,-13,-13,-16,-23,-24,-24,-27,13,14,
     14
0108 DATA 15,15,14,14,15,15,16,16,15,15,16,16,17,17,18,18,
     17
0109 DATA -22,-20,-18,-16,-13,-12,-9,-7,-4,-2,2,4,8,10,12,
     14
0110 DATA 16,18,7,6,6,4,4,5,5,6,-28,-27,-27,-29,-29,-30,
     -28
0111 DATA -29,-26,-28,3,1,-1,-3,-5,-7,-12,-13,-14,-16,-18
0112 DATA -20,-24,-26,-29,-31
0200 FOR I=1 TO 191 STEP 2
0210 POKE (24304,A1(I)+32)
0211 POKE (24305,A1(I+I)+32)
0212 POKE (24306,A2(I)+32)
0213 POKE (24307,A2(I+I)+32)
0214 LET A=USER (5)
0220 NEXT I
0260 W=31
0270 V=1
0271 INPUT X(3), Y(3), Z(3)
0272 X(3)=-(X(3)/100)
0273 Y(3)=-(Y(3)/100)
0274 Z(3)=-(Z(3)/100)
0275 INPUT P,B,H,
0276 P=-P
0277 B=-B
0278 IF H>90 GOTO500
0279 IF H<0 GOTO540
0282 IF M1<>X(3) GO TO 300
0283 IF N1<>Y(3) GO TO 300
0284 IF N2<>Z(3) GO TO 300
0285 IF N3<>P GO TO 300
0286 IF N4<>B GO TO 300
0287 IF N5<>H GO TO 300
0288 LET M1=X(3)
0289 LET N1=Y(3)
0290 LET N2=Z(3)
0291 LET N3=P
0292 LET N4=B
0293 LET N5=H
0294 GOTO 271
0300 GOSUB 8200
0301 LET A=USER (3)
0310 FOR I=1 TO 191 STEP 2
```

Figure 11. Continued

```
0320 X(1)=A1(I)
0330 Y(1)=0
0340 Z(1)=A2(I)
0350 X(5)=A1(I+I)
0360 Y(5)=0
0370 Z(5)=A2(I+I)
0380 GOTO 8500
0390 A7(I)=INT(X(2))
0400 A8(I)=INT(Y(2))
0410 A7(I+I)=INT(X(4))
0420 A8(I+I)=INT(Y(4))
0430 POKE (24304,A7(I)+32)
0431 POKE (24305,A7(I+I)+32)
0432 POKE (24306,A8(I)+32)
0433 POKE (24307,A8(I+I)+32)
0434 LET A=USER(5)
0440 NEXT I
0450 GOTO 288
0500 IF H<180 GOTO 520
0510 GOTO 282
0520 H=H- 180
0530 GOTO 282
0540 IF H>-90 GOTO 560
0550 GOTO 282
0560 H=180+H
0570 GOTO 282
8200 F=P
8203 GOSUB 8300
8204 R1=N
8206 F=P
8209 GOSUB 8310
8212 R2=N
8215 F=B
8218 GOSUB 8300
8221 R3=N
8222 F=B
8224 GOSUB 8310
8227 R4=N
8230 F=H
8233 GOSUB 8300
8236 R5=N
8237 F=H
8239 GOSUB 8310
8245 T1=N*R4+R5*R1*R3
8248 T2=N*R3+R5*R1*R4
8251 T3=R5*R2*V
8253 T4=R2*R3
8256 T5=R2*R4
8259 T6=-R1*V
8262 T7=-R5*R4+N*R1*R3
```

Figure 11. Continued

```
8265 T8=R5*R3+N*R1*R4
8269 T9=R2*N*V
8272 RETURN
8300 F=F-90
8305 IF F<=-180 THEN F=F+360
8310 IF F<0 THEN F=-F
8315 S=F
8320 IF S>=90 THEN F=180-F
8330 N=F*.01745329
8340 A=N*N
8350 M=A*A
8360 N=1-A/2+M/24-A*M/720+M*M/40320
8370 IF S>= 90 THEN N=-N
8380 RETURN
8500 FOR A=1 TO 5 STEP 4
8510 G=X(A)+X(3)
8520 S=Y(A)+Y(3)
8530 K=Z(A)+Z(3)
8540 X(A)=G*T1+S*T4+K*T7
8550 Y(A)=G*T2+S*T5+K*T8
8560 Z(A)=G*T3+S*T6+K*T9
8570 NEXT A
8600 FOR A=1 TO 5 STEP 4
8610 C(A)=0
8612 C(A+1)=0
8614 C(A+2)=0
8616 C(A+3)=0
8618 IF X(A)<-Z(A) THEN C(A)=1
8620 IF X(A)>Z(A) THEN C(A+1)=1
8622 IF Y(A)<-Z(A) THEN C(A+2)=1
8624 IF Y(A)>Z(A) THEN C(A+3)=1
8626 NEXT A
8632 FOR A=1 TO 4 STEP 1
8634 IF C(A)=0 THEN GO TO 8638
8636 IF C(A)=C(A+4) THEN GO TO 8668
8638 NEXT A
8644 FOR A=1 TO 4 STEP 1
8646 IF C(A)=1 THEN GO TO 8676
8648 NEXT A
8654 FOR A=5 TO 8 STEP 1
8656 IF C(A)=1 GOTO 8686
8658 NEXT A
8662 P2=1
8664 GOTO 8800
8668 P2=0
8670 GOTO 440
8676 A=1
8678 S=5
8680 GOTO 8694
8686 A=5
```

Figure 11. Continued

```
8688 S=1
8694 IF C(A)=1 THEN GO TO 8728
8696 IF C(A+1)=1 THEN GO TO 8714
8698 IF C(A=2)=1 THEN GO TO 8742
8700 IF C(A+3)=1 THEN GO TO 8756
8706 GOTO 8662
8714 K=(Z(A)-X(A))/(X(S)-X(A)-Z(S)+Z(A))
8716 X(A)=K*(Z(S)-Z(A))+Z(A)
8718 Y(A)=K*(Y(S)-Y(A))+Y(A)
8720 Z(A)=X(A)
8722 GOTO 8600
8728 K=(Z(A)+X(A))/(X(A)-X(S)-Z(S)+Z(A))
8730 X(A)=K*(Z(A)-Z(S))-Z(A)
8732 Y(A)=K*(Y(S)-Y(A))+Y(A)
8734 Z(A)=-X(A)
8736 GOTO 8600
8742 K=(Z(A)+Y(A))/(Y(A)-Y(S)-Z(S)+Z(A))
8744 X(A)=K*(X(S)-X(A))+X(A)
8746 Y(A)=K*(Z(A)-Z(S))-Z(A)
8748 Z(A)=-Y(A)
8750 GOTO 8600
8756 K=(Z(A)-Y(A))/(Y(S)-Y(A)-Z(S)+Z(A))
8758 X(A)=K*(X(S)-X(A))+X(A)
8760 Y(A)=K*(Z(S)-Z(A))+Z(A)
8762 Z(A)=Y(A)
8764 GOTO 8600
8800 IF Z(1)=- THEN Z(1)=.001
8845 IF Z(5)=0 THEN Z(5)=.001
8855 X(2)=X(1)/Z(1)*W
8860 Y(2)=Y(1)/Z(1)*W
8865 X(4)=X(5)/Z(5)*W
8870 Y(4)=Y(5)/Z(5)*W
8871 IF X(2)=Y(2) GOTO 8876
8875 GOTO 390
8876 IF Y(2)=X(4) GOTO 8878
8877 RETURN
8878 IF X(4)=Y(4) GOTO 440
8879 RETURN
```

Figure 11. Continued

APPENDIX B

"FLITSIM" M6800 ASSEMBLY

MEMORY MAP

Paper Tape Files

0000-004F	3D to 2D Routine Direct Storage
0050-005D	Screen Routine Reserved Variable Memory
005E-00FF	Free Area
0100-0112	3D to 2D Routine Control Array
0113-0FFF	3D to 2D Converter Routine (Enter Routine at Memory Location 0900)
1000-10C6	Interface Routine (Not Used)
1100-1296	Artwick's Database (Input Buffer Array)
1300-1362	Parson's Field (Input Buffer Array)
1800-18FF	Output Buffer Array

Flit A File

6000-6031	System Macro
6033-6077	Clear Memory Routine
6078-6085	PIA Initialization and Clear Screen Routine
6086-60CD	Prompt Routine
60CE-60F7	Initial Load Routine
60F8-60FB	Go to 3D to 2D Converter Routine
60FC-612D	Data Base Draw Routine
612F-61B9	Update and Get New Data Routine

Flit B File

6200-626A	Relative Position Routine
626B-6311	Conversion A/D to 3D/2D Values Routine
6312-6330	Stationary Parameters Routine
6400-65F4	XZ3XZV Array
6600-67F4	Y3YV Array
6800-68FA	PTCHPV Array
6900-69FA	BANKBV Array
6A00-6AFA	HEADHV Array

Flit C File

7000-711E	Screen Drive Routine
7200-73DF	Welcome Prompt
7400-7683	MSGC Prompt
7700-77FA	MSGL Prompt
7800-7829	Reserved Memory Locations-Clearable
78F0-78F3	Reserved Memory Locations-Non-Clearable
7900-79F1	MSGA Prompt
7A00-7A4D	MSGB Prompt

Figure 12. Memory Map for FLITSIM

```
0010  NAM FLITA
0020  ORG $6000
0030  OPT O
0040 *THE FOLLOWING PROGRAM WAS WRITTEN BY WAYNE PARSONS IN
0050 *CONJUNCTION WITH THE INDUSTRIAL ENGINEERING DEPARTMENT
0060 *AND DR. BAUER AS A PARTIAL REQUIREMENT FOR COMPLETION
0070 *OF A MS DEGREE AT THE UNIVERSITY OF CENTRAL FLORIDA IN THE
0080 *FALL SEMESTER OF 1981. IT IS DEDICATED TO MY WIFE GAIL AND
0090 *OUR TWO CHILDREN CHRISTIE AND ANGELA.
0100 *
0110 ************************************************************************
0120 *
0130 *THE FOLLOWING SECTION OF THE PROGRAM IS THE SYSTEM MACRO
0140 *IT IS DESIGNED TO SEQUENTIALLY CALL THE PROPER ROUTINES IN
0150 *ORDER TO EFFICIENTLY EXECUTE THE PROBLEM ALGORITHM.
0160 MACRO STS STKSAV
0170        CLRA
0180        CLRB
0190        JSR CLRMEM   ;CLEAR MEMORY LOCATIONS
0200        JSR PIAINT  ;INITIALIZE PIA AND CLEAR SCREEN
0210        JSR PROMPT  ;PRINT USER PROMPT MESSAGES
0220        JSR INITLOD ;LOAD INITIAL AIRCRAFT POSITIONAL PARAMETERS
0225 TMPLOD DEC ZV      ;USE THIS STATEMENT FOR DEBUG PURPOSES ONLY !!
0230          BRA TIMONE  ;GO TRANSFORM INITIAL VIEW 1ST TIME THRU
0240 GOAGN  JSR UPDATE  ;GET UPDATED A/D VOLTS
0241          BRA GONORM
0245 ADSIM  JSR STAPAR  ;LOAD STATIONARY PARAMETERS TO SIMULATE A/D
0250 GONORM JSR RELPOS  ;DETERMINE IF RELATIVE POSITION HAS CHANGED
0260          JSR CNVRT   ;A/D VOLTS => PARAMETER
```

Figure 13. Assembly FLITSIM Program Listing

```
0270 TIMONE JSR GO3D2D  ;GO TRANSFORM 3D TO 2D COORDINATES
0280        JSR CLSCRN  ;CLEAR SCREEN FOR NEW DRAW
0290        JSR DBDRAW  ;DRAW TRANSFORMED DATA BASE
0297        BRA TMPLOD  ;USE IN DEBUG MODE ONLY
0298        BRA ADSIM   ;USE IN DEBUG MODE ONLY
0300        BRA GOAGN   ;NORMAL RETURN, SYSTEM OPERATION
0310 *SUBROUTINE "CLRMEM" ATTEMPTS TO CLEAR MEMORY STORAGE LOCATIONS
0320 *AND DATA ARRAYS RESERVED BUT NOT YET CONTAINING ANY MEANINGFUL
0330 *DATA.
0340 CLRMEM CLRB
0350        STAB CNT
0360        STAB COUNT
0370 GETNXT LDAB CNT
0380        BNE MOREOB
0390        LDAB COUNT
0400        INCB
0410        STAB COUNT
0420        CMPB #1
0430        BEQ CLRRML
0440        CMPB #2
0450        BEQ CLROB
0460        CMPB #3
0470        BEQ SUBDUN
0480 CLEAR1 LDAB #$FF     ;TOTAL NUMBER OF LOCATIONS TO CLEAR
0490 CLEAR2 CLRA
0500 CLRMOR STAA $00,X
0510        DEC B
0520        BEQ GETNXT
0530        INX
0540        BRA CLRMOR
0550 CLRRML LDX #$7800    ;CLEAR RESERVE MEMORY LOCATIONS
0555        LDA B #$EF
```

Figure 13. <u>Continued</u>

```
0560        BRA CLEAR2
0570 CLROB LDAB #1         ;CLEAR OUTPUT BUFFER
0580        STAB CNT
0590        LDX #$1800    ;OUTPUT BUFFER STARTING ADDRESS = 1800
0600        BRA CLEAR1
0610 MOREOB DEC B
0620        STAB CNT
0630        LDX #$1900
0640        BRA CLEAR1
0650 SUBDUN RTS            ;ALL DONE ; GO BACK TO MACRO
0660 *SUBROUTINES "PIAINT" AND "CLSCRN" USE THE MEMORY LOCATION
0670 * "GDOPT" TO CHOOSE THE GRAPHICS DISPLAY OPTION DESIRED BY
0680 *THE ROUTINE "SCREEN" AND THEN CALLS IT.
0690 PIAINT LDAA #1
0700        STAA GDOPT  ;GRAPHICS DISPLAY OPTION = PIA INITIALIZATION
0710        JSR SCREEN
0720 CLSCRN LDAA #3
0730        JSR SCREEN
0740        RTS
0750 *SUBROUTINE "PROMPT" WILL LOAD THE INDEX REGISTER WITH THE
0760 *ADDRESS OF THE NEXT USER PROMPT TO BE DISPLAYED AND THEN
0770 *WILL PRINT THE MESSAGE ON THE USER'S TERMINAL. IT WILL
0780 *THEN WAIT UNTIL THE USER HAS READ AND UNDERSTOOD THE
0790 *PROMPT BEFORE PRINTING THE NEXT MESSAGE.
0800 PROMPT LDX #WELCOM
0810        JSR DSPCHR
0815        LDX #MSGA
0816        JSR DSPCHR
0817        LDX #MSGB
```

Figure 13. Continued

```
0818        JSR DSPCHR
0820        LDX #MSGC
0830        JSR DSPCHR
0840        RTS            ;RETURN TO MACRO
0850 DSPCHR LDAA $0,X
0860        CMPA #04
0870        BNE PRTCHR
0880 NOTGOT CLRA          ;PREPARE ACCM FOR RESPONSE
0890        JSR $E1AC     ;ANY RESPONSE ?
0892        CMPA #$31
0894        BEQ LOADIR
0895        CMPA #$32
0897        BEQ LOADIR
0900        CMPA #$A0     ;ASCII SPACEBAR = A0
0910        BEQ NOTGOT
0920        RTS           ;RETURN FROM JSR DSPCHR CALL
0930 PRTCHR JSR $E1D1     ;PUT CHARACTER OUT TO TERMINAL
0940        INX
0950        BRA DSPCHR
0952 LOADIR CMPA #$31
0953        BEQ LOAD1
0954        LDX #$1100     ;RESPONSE TO MSGB PROMPT = 2
0955        STX IBP
0956        RTS            ;RETURN FROM JSR DSPCHR CALL
0957 LOAD1  LDX #$1300     ;RESPONSE TO MSGB PROMPT = 1
0958        STX IBP
0959        RTS            ;RETURN FROM JSR DSPCHR CALL
0960 *SUBROUTINE "INITLOD" LOADS THE SIX POSITIONAL PARAMETERS
0970 *OF THE SIMULATOR'S INITIAL POSITION WHICH WAS PREVIOUSLY
0980 *PROMPTED TO THE USER.
```

Figure 13. Continued

```
0990 INTLOD LDX #$D500    ;-11,190 FEET FROM DISPLAY ORIGIN
1000        STX XV        ;POS = WEST (0000-7FFF) NEG = EAST (FFFF-8000)
1010        LDX #$FC18    ;ALT = 1000 FT => 3D VALUE FROM VIEWER,
1020        STX YV        ;     SO REQUIRE NEGATIVE 3D VALUE (FFFF-8000)
1030        LDX #$F000    ;-4,096 FEET FROM DISPLAY ORIGIN
1040        STX ZV        ;POS=SOUTH (0000-7FFF) NEG=NORTH (FFFF-8000)
1050        LDX #$0000    ;PITCH = BANK = HEADING = 0.0 DEGREES
1060        STX PV
1070        STX BV
1080        STX HV
1110        LDX #$1800
1120        STX OBP
1130        LDAA #$40
1140        STAA SCRN
1250        RTS
1260 *SUBROUTINE "GO3D2D" CALLS THE PROGRAM WRITTEN BY SUBLOGIC'S
1270 *BRUCE ARTWICK WHICH WILL PERFORM THE NECCESSARY MATHEMATICAL
1280 *ALGORITHMS TO SUCCESSFULLY TRANSFORM THE DATA BASE GIVEN
1290 *THE POSITIONAL PARAMETERS STORED IN XV,YV,ZV,PV,BV, AND HV.
1300 *UPON COMPLETION THE OUTPUT ARRAY IS STORED IN MEMORY IN
1310 *THE OUTPUT BUFFER.
1320 GO3D2D JSR $0900     ;3D TO 2D CONVERTER ROUTINE
1330        RTS
1340 *SUBROUTINE "DBDRAW" WILL LOAD THE NEXT (X1,Y1) AND (X2,Y2)
1350 *FROM THE OUTPUT BUFFER IF THE CONTROL BYTE IS HEX 55,
1360 *OTHERWISE IT WILL RETURN TO THE MACRO.
1370 DBDRAW LDX OBP
1380 FEED   LDAA 0,X
1390        CMPA #$55
```

Figure 13. Continued

```
1400        BEQ NXTLIN
1410        RTS
1420 NXTLIN INX
1430        LDAA 0,X
1440        STAA X1
1450        INX
1460        LDAA 0,X
1470        STAA Y1
1480        INX
1490        LDAA 0,X
1500        STAA X2
1510        INX
1520        LDAA 0,X
1530        STAA Y2
1540        INX
1550        STX IDXREG
1560        LDAA #5
1570        STAA GDOPT    ;GRAPHIC'S DISPLAY OPTION = DRAW (SHOW2)
1580        JSR SCREEN
1590        LDX IDXREG
1600        BRA FEED
1610 *SUBROUTINE "UPDATE" ENABLES THE M6800/ATC-610J INTERFACE
1620 *AD-68A CARD THEREBY RETRIEVING THE SELECTED UPDATED
1630 *PARAMETER AND PLACES IT IN MEMORY AT LOCATION "NEWDAT".
1640 UPDATE CLRA
1650        STAA NEWDAT
1660        LDAA #16     ;BIT 4 = 1, X3 ENABLED (+X WEST/-X EAST)
1670        STAA ENABLE
1680        JSR GETNEW
```

Figure 13. Continued

```
1690        LDAA NEWDAT
1700        STAA X3       ;X3 = A/D VOLTS
1710        CLRA
1720        STAA NEWDAT
1730        LDAA #32      ;BIT 5 = 1, Z3 ENABLED (+Z SOUTH/-Z NORTH)
1740        STAA ENABLE
1750        JSR GETNEW
1760        LDAA NEWDAT
1770        STAA Z3       ;Z3 = A/D VOLTS
1780        CLRA
1790        STAA NEWDAT
1800        LDAA #2       ;BIT 1 = 1, Y3 ENABLED (ALTITUDE)
1810        STAA ENABLE
1820        JSR GETNEW
1830        LDAA NEWDAT
1840        STAA Y3       ;Y3 = A/D VOLTS
1850        CLRA
1860        STAA NEWDAT
1870        LDAA #1       ;BIT 0 = 1, P ENABLED
1880        STAA ENABLE
1890        JSR GETNEW
1900        LDAA NEWDAT
1910        STAA PITCH    ;PITCH = A/D VOLTS
1920        CLRA
1930        STAA NEWDAT
1940        LDAA #8       ;BIT 3 = 1, B ENABLED
1950        STAA ENABLE
1960        JSR GETNEW
```

Figure 13. Continued

```
1970        LDAA NEWDAT
1980        STAA BANK      ;BANK = A/D VOLTS
1990        CLRA
2000        STAA NEWDAT
2010        LDAA #4        ;BIT 2 = 1, H ENABLED
2020        STAA ENABLE
2030        JSR GETNEW
2040        LDAA NEWDAT
2050        STAA HEAD       ;HEAD = A/D VOLTS
2055        RTS
2060 * SUBROUTINE "GETNEW" IS CALLED FROM AND RETURNS TO "UPDATE".
2070 * IT GETS INSTRUCTIONS ON WHICH AD-68A CHANNEL TO ENABLE FROM
2080 * MEMORY LOCATION "ENABLE" AND THEN RETRIEVES THE INPUT AND
2090 * LEAVES IT IN MEMORY LOCATION "NEWDAT".
2100 GETNEW CLRB
2110        LDAA ##$80
2120        STAA $800C
2130 ENTER  LDAA $800C
2140        CMPB ##FA
2150        BHI EXIT
2160        BITA ENABLE
2170        BNE EXIT
2180        INCB
2190        BRA ENTER
2200 EXIT   STAB NEWDAT
2210        LDAA ##$40
2220        STAA $800C
2230        RTS
```

Figure 13. Continued

```
2240 STKSAV EQU $78F1
2250 CNVRT EQU $626B
2260 RELPOS EQU $6200
2270 CNT EQU $78F3
2280 COUNT EQU $78F0
2290 GDOPT EQU $7817
2300 SCREEN EQU $7000
2310 WELCOM EQU $7200
2315 MSGA EQU $7900
2316 MSGB EQU $7A00
2320 MSGC EQU $7400
2330 XV EQU $0100
2340 YV EQU $0102
2350 ZV EQU $0104
2360 PV EQU $0106
2370 BV EQU $0108
2380 HV EQU $010A
2390 IBP EQU $010F
2400 OBP EQU $0111
2460 SCRN EQU $010E
2470 X1 EQU $0050
2480 Y1 EQU $0051
2490 X2 EQU $0052
2500 Y2 EQU $0053
2510 IDXREG EQU $781A
2520 NEWDAT EQU $7818
2530 ENABLE EQU $7819
2540 X3 EQU $7800
2550 Y3 EQU $7802
```

Figure 13. Continued

```
2560 Z3 EQU $7804
2570 PITCH EQU $7806
2580 BANK EQU $7808
2590 HEAD EQU $780A
2595 STAPAR EQU $6312
2600  END
```

Figure 13. Continued

```
0010  NAM FLITB
0020  OPT O
0030  ORG $6200
0040 * SUBROUTINE "RELPOS" DETERMINES IF THE RELATIVE POSITION
0050 * OF THE AIRCRAFT HAS CHANGED WHILE THE PROGRAM HAS BEEN
0060 * CALCULATING THE LAST TRANSFORMATIONS. IT DOES THIS BY
0070 * COMPARING THE NEWLY UPDATED VALUES RECIEVED BY THE A/D
0080 * NOW STORED IN LOCATIONS X3, Y3, Z3, PITCH, BANK, AND
0090 * ROLL TO THOSE VALUES WHICH WERE PREVIOUSLY STORED IN
0100 * LOCATIONS M1, N1, N2, N3, N4, AND N5. IF ANY ONE OF THE
0110 * COMPARSIONS ARE NOT EQUAL THEN THE PLANE HAS CHANGED
0120 * IT'S RELATIVE POSITION AND THE PROGRAM WILL JUMP TO
0130 * THE TRANSFORMATION ROUTINES AGAIN. IF THE COMPARSION
0140 * PROVES TO BE THE SAME THEN THE PROGRAM JUMPS BACK TO
0150 * THE ROUTINE "UPDATE" TO TRY FOR A CHANGE IN THE PARAMETERS.
0160 * IF THE ROUTINE DOES NOT GFT A CHANGE IN THE PARAMETER
0170 * AFTER 5 TRIES IT WILL JUMP TO A ROUTINE WHICH WILL END
0180 * THE PROGRAM AFTER A "LANDED" MESSAGE IS PRINTED OUT TO
0190 * THE USER'S TERMINAL.
0200 RELPOS CLRA
0220 EVAL   LDAA X3
0230        CMPA M1
0240        BNE GOTRAN
0250        LDAA Y3
0260        CMPA N1
0270        BNE GOTRAN
0280        LDAA Z3
0290        CMPA N2
0300        BNE GOTRAN
```

Figure 13. Continued

```
0310        LDAA PITCH
0320        CMPA N3
0330        BNE GOTRAN
0340        LDAA BANK
0350        CMPA N4
0360        BNE GOTRAN
0370        LDAA HEAD
0380        CMPA N5
0390        BNE GOTRAN
0400 MAKEQ  LDAA X3
0410        STAA M1
0420        LDAA Y3
0430        STAA N1
0440        LDAA Z3
0450        STAA N2
0460        LDAA PITCH
0470        STAA N3
0480        LDAA BANK
0490        STAA N4
0500        LDAA HEAD
0510        STAA N5
0520        LDAA FLGDUN
0530        INCA
0540        CMPA #4        ;FLGDUN = 4 MAKES 4 TRIES THRU LOOP
0550        BEQ LANDED     ;IF 5TH TRY PRINT LANDED MESSAGE AND QUIT
0560        JMP GOAGN      ;NOT 5TH TRY GO TO MACRO AND TRY FOR UPDATE
0570 LANDED JSR PRINTL     ;LANDED SO GO PRINT MESSAGE
0580        SWI            ;AND QUIT
0590 PRINTL LDX #MSGL      ;INDEX REGISTER <= START OF MESSAGE
0600        JMP DSPCHR     ;ROUTINE "DSPCHR" HAS RTS FOR JSR PRINTL
```

Figure 13. Continued

```
GOTRAN RTS              ;DATA IS DIFFERENT, GO CNVRT AND TRANSFORM
* SUBROUTINE "CNVRT" WILL TAKE THE STORED A/D VOLTS VALUE OF
* EACH OF THE SIX PARAMETERS; (X3, Y3, Z3, PITCH, BANK,
* AND HEAD), CONVERT IT INTO SCALED UNITS EXPECTED BY THE
* 3D TO 2D CONVERTER ROUTINE (XV, YV, ZV, PV, BV, AND HV)
* AND THEN TRANSFER THE VALUES TO THE 3D TO 2D CONVERTER
* MEMORY LOCATIONS FOR USE IN THAT ROUTINE.
CNVRT CLR CNT
      LDAA X3
      STAA M1
      SUBA #13          ;TETERBORO = 0.5 NM WEST OF SIM WORLD CENTER
      LDX #XZ3XZV
      JSR XYZ
      STAA XV
      STAB XV+1
DOZ   CLR CNT
      LDAA Z3
      STAA N2
      SUBA #27          ;TETERBORO = 1.0 NM SOUTH CENTER SIM WORLD
      LDX #XZ3XZV
      JSR XYZ
      STAA ZV
      STAB ZV+1
DOY   CLR CNT
      LDAA Y3
      STAA N1
      LDX #Y3YV
      JSR XYZ
      STAA YV
      STAB YV+1
```

Figure 13. Continued

```
0910        BRA DOPTCH
0920 XYZ    CMPA CNT        ;DOES VALUE = CNT ???
0930        BEQ GETXYZ      ;YES, GO GET 3D-2D ROUTINE VALUE
0940        INX             ;NO, INCREMENT INDEX REG TWICE
0950        INX
0960        INC CNT
0970        BRA XYZ         ;TRY AGAIN UNTIL MATCH OCCURS
0980 GETXYZ LDAA $0,X       ;LOAD A ACCM WITH VALUE MSB
0990        LDAB $1,X       ;LOAD B ACCM WITH VALUE LSB
1000        RTS
1010 DOPTCH CLR CNT
1020        LDAA PITCH
1030        STAA N3
1040        LDX #PTCHPV
1050        JSR PBH
1060        CLRA
1070        STAA PV
1080        STAB PV+1
1100 DOBANK CLR CNT
1110        LDAA BANK
1120        STAA N4
1130        LDX #BANKBV
1140        JSR PBH
1150        CLRA
1160        STAA BV
1170        STAB BV+1
1180 DOHEAD CLR CNT
1190        LDAA HEAD
1200        STAA N5
1210        LDX #HEADHV
```

Figure 13. Continued

```
1220        JSR PBH
1230        CLRA
1240        STAA HV
1250        STAB HV+1
1260        RTS
1270 PBH    CMPA CNT
1280        BEQ GETPBH
1290        INX
1300        INC CNT
1310        BRA PBH
1320 GETPBH LDAB $0,X
1330        RTS
1340 * SUBROUTINE "STAPAR" WILL LOAD THE A/D PARAMETERS FROM VALUES
1350 * PICKED BY THE USER AND FORCE THEM INTO THE ROUTINE AT THE
1360 * PARAMETERS MEMORY LOCATION SO THAT THE "CNVRT" ROUTINE MAY
1370 * BE VERIFIED. IT IS USED ONLY FOR TESTING PURPOSES.
1380 STAPAR LDAA #$7D
1390        STAA X3
1400        LDAA #$AC
1410        STAA Z3
1420        LDAA #$AA
1430        STAA Y3
1440        LDAA #$7D
1450        STAA PITCH
1460        LDAA #$7D
1470        STAA BANK
1480        LDAA #$55
1490        STAA HEAD
1500        RTS
1510  ORG $6400
```

Figure 13. Continued

```
1520 XZ3XZV FDB $8000,$8000,$8000,$8000,$8000,$8000,$8000,$8000,$8000
1530        FDB $8000,$8000,$8000,$8000,$8000,$8000,$8000,$8000,$8000
1540        FDB $8000,$8000,$8000,$8000,$8000,$8000,$8000,$8000,$80C4
1550        FDB $81A8,$828C,$8370,$8454,$8538,$861C,$8700,$87E4,$88C8
1560        FDB $89AC,$8A90,$8B74,$8C58,$8D3C,$8E20,$8F04,$8FE8,$90CC
1570        FDB $91B0,$9294,$9378,$945C,$9540,$9624,$9708,$97EC,$98D0
1580        FDB $99B4,$9A98,$9B7C,$9C60,$9D44,$9E28,$9F0C,$9FF0,$A0D4
1590        FDB $A1B8,$A29C,$A380,$A464,$A548,$A62C,$A710,$A7F4,$A8D8
1600        FDB $A9BC,$AAA0,$AB84,$AC68,$AD4C,$AE30,$AF14,$AFF8,$B0DC
1610        FDB $B1C0,$B2A4,$B388,$B46C,$B550,$B634,$B718,$B7FC,$B8E0
1620        FDB $B9C4,$BAA8,$BB8C,$BC70,$BD54,$BE38,$BF1C,$C000,$C0E4
1630        FDB $C1C8,$C2AC,$C390,$C474,$C558,$C63C,$C720,$C804,$C8EB
1640        FDB $C9CC,$CAB0,$CB94,$CC78,$CD5C,$CE40,$CF24,$D008,$D0EC
1650        FDB $D1D0,$D2B4,$D398,$D47C,$D560,$D644,$D728,$D80C,$D8F0
1660        FDB $D9D4,$DAB8,$DB9C,$DC80,$DD64,$DE48,$DF2C,$E010,$E0F4
1670        FDB $E1D8,$E2BC,$E3A0,$E484,$E568,$E64C,$E730,$E814,$E8F8
1680        FDB $E9DC,$EAC0,$EBA4,$EC88,$ED6C,$EE50,$EF34,$F018,$F0FC
1690        FDB $F1E0,$F2C4,$F3A8,$F48C,$F570,$F654,$F738,$F81C,$F900
1700        FDB $F9E4,$FAC8,$FBAC,$FC90,$FD74,$FE58,$FF3C,$0020,$0104
1710        FDB $01E8,$02CC,$03B0,$0494,$0578,$065C,$0740,$0824,$0908
1720        FDB $09EC,$0AD0,$0BB4,$0C98,$0D7C,$0E60,$0F44,$1028,$110C
1730        FDB $11F0,$12D4,$13B8,$149C,$1580,$1664,$1748,$182C,$1910
1740        FDB $19F4,$1AD8,$1BBC,$1CA0,$1D84,$1E68,$1F4C,$2030,$2114
1750        FDB $21F8,$22DC,$23C0,$24A4,$2588,$266C,$2750,$2834,$2918
1760        FDB $29FC,$2AE0,$2BC4,$2CA8,$2D8C,$2E70,$2F54,$3038,$311C
1770        FDB $3200,$32E4,$33C8,$34AC,$3590,$3674,$3758,$383C,$3920
1780        FDB $3A04,$3AE8,$3BCC,$3CB0,$3D94,$3E78,$3F5C,$4040,$4124
1790        FDB $4208,$42EC,$43D0,$44B4,$4598,$467C,$4760,$4844
1800  ORG $6600
1810 Y3YV FDB $EC78,$EC8C,$ECA0,$ECB4,$ECC8,$ECDC,$ECF0,$ED04,$ED18
```

Figure 13. Continued

```
1820      FDB $ED2C,$ED40,$ED54,$ED68,$ED7C,$ED90,$EDA4,$EDB8,$EDCC
1830      FDB $EDE0,$EDF4,$EE08,$EE1C,$EE30,$EE44,$EE58,$EE6C,$EE80
1840      FDB $EE94,$EEA8,$EEBC,$EED0,$EEE4,$EEF8,$EF0C,$EF20,$EF34
1850      FDB $EF48,$EF5C,$EF70,$EF84,$EF98,$EFAC,$EFC0,$EFD4,$EFE8
1860      FDB $EFFC,$F010,$F024,$F038,$F04C,$F060,$F079,$F092,$F0AB
1870      FDB $F0C4,$F0DD,$F0F6,$F10F,$F128,$F141,$F15A,$F173,$F18C
1880      FDB $F1A5,$F1BE,$F1D7,$F1F0,$F209,$F222,$F23B,$F254,$F26D
1890      FDB $F286,$F29F,$F2B8,$F2D1,$F2EA,$F303,$F31C,$F335,$F34E
1900      FDB $F367,$F380,$F399,$F3A2,$F3CB,$F3E4,$F3FD,$F416,$F42F
1910      FDB $F448,$F461,$F47A,$F493,$F4AC,$F4C5,$F4DE,$F4F7,$F510
1920      FDB $F529,$F542,$F55B,$F574,$F58D,$F5A6,$F5BF,$F5D8,$F5F1
1930      FDB $F60A,$F623,$F63C,$F655,$F66E,$F687,$F6A0,$F6B9,$F6D2
1940      FDB $F6EB,$F704,$F71D,$F736,$F74F,$F768,$F781,$F79A,$F7B3
1950      FDB $F7CC,$F7E5,$F7FE,$F817,$F830,$F849,$F862,$F87B,$F894
1960      FDB $F8AD,$F8C6,$F8DF,$F8F8,$F911,$F92A,$F943,$F95C,$F975
1970      FDB $F98E,$F9A7,$F9C0,$F9D9,$F9F2,$FA0B,$FA24,$FA3D,$FA56
1980      FDB $FA6F,$FA88,$FAA1,$FABA,$FAD3,$FAEC,$FB05,$FB1E,$FB37
1990      FDB $FB50,$FB69,$FB82,$FB9B,$FBB4,$FBCD,$FBE6,$FBFF,$FC18
2000      FDB $FC24,$FC31,$FC3D,$FC4A,$FC56,$FC63,$FC6F,$FC7C,$FC88
2010      FDB $FC95,$FCA1,$FCAE,$FCBA,$FCC7,$FCD3,$FCE0,$FCEC,$FCF9
2020      FDB $FD05,$FD12,$FD1E,$FD2B,$FD37,$FD44,$FD50,$FD5D,$FD69
2030      FDB $FD76,$FD82,$FD8F,$FD9B,$FDA8,$FDB4,$FDC1,$FDCD,$FDDA
2040      FDB $FDE6,$FDF3,$FDFF,$FE0C,$FE18,$FE25,$FE31,$FE3E,$FE4A
2050      FDB $FE57,$FE63,$FE70,$FE7C,$FE89,$FE95,$FEA2,$FEAE,$FEBB
2060      FDB $FEC7,$FED4,$FEE0,$FEED,$FEF9,$FF06,$FF12,$FF1F,$FF2B
2070      FDB $FF38,$FF44,$FF51,$FF5D,$FF6A,$FF76,$FF83,$FF8F,$FF9C
2080      FDB $FFAB,$FFB5,$FFC1,$FFCE,$FFDA,$FFE6,$FFF3,$0000
2090  ORG $6800
2100 PTCHPV FCB $40,$3F,$3E,$3D,$3D,$3C,$3C,$3B,$3B,$3A,$3A,$39,$39,$38
2110        FCB $38,$37,$37,$36,$36,$35,$35,$34,$34,$33,$33,$32,$32,$31
```

Figure 13. Continued

```
2120        FCB $31,$30,$30,$2F,$2F,$2E,$2E,$2D,$2D,$2C,$2C,$2B,$2B,$2A
2130        FCB $2A,$29,$29,$28,$28,$27,$27,$26,$26,$25,$25,$24,$24,$23
2140        FCB $23,$22,$22,$21,$21,$20,$20,$1F,$1F,$1E,$1E,$1D,$1D,$1C
2150        FCB $1C,$1B,$1B,$1A,$1A,$19,$19,$18,$18,$17,$17,$16,$16,$15
2160        FCB $15,$14,$14,$13,$13,$12,$12,$11,$11,$10,$10,$0F,$0F,$0E
2170        FCB $0E,$0D,$0D,$0C,$0C,$0B,$0B,$0A,$0A,$09,$09,$08,$08,$07
2180        FCB $07,$06,$06,$05,$05,$04,$04,$03,$03,$02,$02,$01,$01,$00
2190        FCB $00,$FF,$FF,$FE,$FE,$FD,$FD,$FC,$FC,$FB,$FB,$FA,$FA,$F9
2200        FCB $F9,$F8,$F8,$F7,$F7,$F6,$F6,$F5,$F5,$F4,$F4,$F3,$F3,$F2
2210        FCB $F2,$F1,$F1,$F0,$F0,$EF,$EF,$EE,$EE,$ED,$ED,$EC,$EC,$EB
2220        FCB $EB,$EA,$EA,$E9,$E9,$E8,$E8,$E7,$E7,$E6,$E6,$E5,$E5,$E4
2230        FCB $E4,$E3,$E3,$E2,$E2,$E1,$E1,$E0,$E0,$DF,$DF,$DE,$DE,$DD
2240        FCB $DD,$DC,$DC,$DB,$DB,$DA,$DA,$D9,$D9,$D8,$D8,$D7,$D7,$D6
2250        FCB $D6,$D5,$D5,$D4,$D4,$D3,$D3,$D2,$D2,$D1,$D1,$D0,$D0,$CF
2260        FCB $CF,$CE,$CE,$CD,$CD,$CC,$CC,$CB,$CB,$CA,$CA,$C9,$C9,$C8
2270        FCB $C8,$C7,$C7,$C6,$C6,$C5,$C5,$C4,$C4,$C3,$C2,$C1,$C0
2280  ORG $6900
2290 BANKBV FCB $40,$3F,$3E,$3D,$3C,$3B,$3A,$39,$39,$38,$38,$37,$37,$36
2300        FCB $36,$35,$35,$34,$34,$33,$33,$32,$32,$31,$31,$30,$30,$2F
2310        FCB $2F,$2E,$2E,$2D,$2D,$2C,$2C,$2B,$2B,$2A,$2A,$29,$29,$28
2320        FCB $28,$27,$27,$26,$26,$25,$25,$24,$24,$23,$23,$22,$22,$21
2330        FCB $21,$20,$20,$1F,$1F,$1E,$1E,$1D,$1D,$1C,$1C,$1B,$1B,$1A
2340        FCB $1A,$19,$19,$18,$17,$16,$15,$15,$14,$14,$13,$13,$12,$12
2350        FCB $11,$11,$10,$10,$0F,$0F,$0E,$0E,$0E,$0D,$0D,$0D,$0C,$0C
2360        FCB $0C,$0B,$0B,$0A,$0A,$09,$09,$08,$08,$07,$07,$06,$06,$06
2370        FCB $05,$05,$05,$04,$04,$04,$03,$03,$03,$02,$02,$01,$01,$00
2380        FCB $00,$FF,$FF,$FE,$FE,$FD,$FD,$FC,$FC,$FB,$FB,$FA,$FA,$F9
2390        FCB $F9,$F8,$F8,$F7,$F7,$F6,$F6,$F6,$F5,$F5,$F4,$F4,$F3,$F3
2400        FCB $F2,$F2,$F1,$F1,$F0,$F0,$EF,$EF,$EE,$EE,$ED,$ED,$EC,$EB
2410        FCB $EA,$EA,$E9,$E9,$E8,$E8,$E7,$E7,$E6,$E6,$E5,$E5,$E4,$E4
```

Figure 13. Continued

```
2420        FCB $E3,$E3,$E2,$E2,$E1,$E1,$E0,$E0,$DF,$DF,$DE,$DE,$DD,$DD
2430        FCB $DC,$DC,$DB,$DB,$DA,$DA,$D9,$D9,$D8,$D8,$D7,$D7,$D7,$D6
2440        FCB $D6,$D6,$D5,$D5,$D5,$D4,$D4,$D4,$D3,$D3,$D3,$D2,$D2,$D2
2450        FCB $D1,$D1,$D0,$D0,$CF,$CF,$CE,$CE,$CD,$CD,$CC,$CC,$CB,$CB
2460        FCB $CA,$CA,$C9,$C9,$C8,$C7,$C6,$C5,$C4,$C3,$C2,$C1,$C0
2470  ORG $6A00
2480 HEADHV FCB $40,$40,$3F,$3F,$3E,$3E,$3D,$3D,$3C,$3C,$3B,$3B,$3A,$39
2490        FCB $38,$37,$36,$35,$34,$33,$32,$31,$30,$2F,$2E,$2D,$2C,$2B
2500        FCB $2A,$29,$28,$27,$26,$25,$24,$23,$22,$21,$20,$1F,$1E,$1D
2510        FCB $1C,$1B,$1A,$19,$18,$17,$16,$15,$14,$13,$12,$11,$10,$0F
2520        FCB $0F,$0E,$0E,$0D,$0D,$0C,$0C,$0B,$0B,$0A,$0A,$09,$09,$08
2530        FCB $08,$07,$07,$06,$06,$05,$05,$04,$04,$03,$03,$02,$02,$01
2540        FCB $01,$00,$00,$FF,$FF,$FE,$FE,$FD,$FD,$FC,$FC,$FB,$FB,$FA
2550        FCB $FA,$F9,$F9,$F8,$F8,$F7,$F7,$F6,$F6,$F5,$F4,$F3,$F2,$F1
2560        FCB $F0,$EF,$EE,$ED,$EC,$EB,$EA,$E9,$E8,$E7,$E6,$E5,$E4,$E3
2570        FCB $E2,$E1,$E0,$E0,$DF,$DF,$DE,$DE,$DD,$DD,$DC,$DC,$DB,$DB
2580        FCB $DA,$D9,$D8,$D7,$D6,$D5
2590        FCB $D4,$D3,$D2,$D1,$D0,$CF,$CE,$CD,$CC,$CB,$CA,$C9,$C8,$C7
2600        FCB $C6,$C5,$C4,$C3,$C2,$C1,$C1,$C0,$C0,$BF,$BE,$BD,$BC,$BB
2610        FCB $BA,$B9,$B8,$B7,$B6,$B5,$B4,$B3,$B2,$B1,$B0,$AF,$AE,$AD
2620        FCB $AC,$AB,$AA,$A9,$A8,$A7,$A6,$A4,$A3,$A2,$A1,$A0,$9F,$9E
2630        FCB $9D,$9C,$9B,$9A,$99,$98,$97,$96,$95,$94,$93,$92,$91,$90
2640        FCB $8F,$8E,$8D,$8C,$8B,$8A,$8A,$8A,$89,$89,$89,$88,$88,$88
2650        FCB $87,$87,$87,$86,$86,$86,$85,$85,$85,$84,$84,$84,$83,$83
2660        FCB $83,$82,$82,$82,$81,$81,$81,$80,$80,$80
2670 CNT EQU $78F3
2680 XV EQU $0100
2690 YV EQU $0102
2700 ZV EQU $0104
2710 PV EQU $0106
```

Figure 13. <u>Continued</u>

```
2720 BV EQU $0108
2730 HV EQU $010A
2740 X3 EQU $7800
2750 Y3 EQU $7802
2760 Z3 EQU $7804
2770 PITCH EQU $7806
2780 BANK EQU $7808
2790 HEAD EQU $780A
2800 M1 EQU $780C
2810 N1 EQU $780E
2820 N2 EQU $7810
2830 N3 EQU $7812
2840 N4 EQU $7814
2850 N5 EQU $7816
2860 FLGDUN EQU $781C
2870 MSGL EQU $7700
2880 GOAGN EQU $6016
2890 DSPCHR EQU $609F
2900  END
```

Figure 13. Continued

```
0010  NAM FLITC
0020  OPT O
0030  ORG $7000
0040 SCREEN CMPA #01
0050        BEQ FIRST
0060        CMPA #02
0070        BEQ JINIT
0080        CMPA #03
0090        BEQ ERASE
0100        CMPA #04
0110        BEQ PIXEL
0120        CMPA #05
0130        BEQ SHOW2
0140        CMPA #06
0150        BEQ START
0160        RTS
0170 * PIA INITIALIZATION SECTION
0180 FIRST CLRA
0190       STAA $8009
0200       LDAA #$FF
0210       STAA $8008
0220       LDAA #$3F
0230       STAA $8009
0240       RTS
0250 * ERASE SCREEN SECTION
0260 ERASE LDAA #63
0270       STAA HPOS
0280 HSET  LDAA HPOS
0290       BSR SENDA
```

Figure 13. Continued

```
0300 VBAR  LDAA #128
0310 CLR   BSR SENDA
0320       INCA
0330 MOD   CMPA #224
0340       BNE CLR
0350       DEC HPOS
0360       BGE HSET
0370       LDAA #192
0380       STAA MOD+1
0390       RTS
0400 * SCREEN DRIVE SUBROUTINES
0410 SENDA TAB
0420 SENDB STAB $8008
0430       LDAB #$37
0440       STAB $8009
0450       LDAB 0,X
0460       LDAB #$3F
0470       STAB $8009
0480       RTS
0490 * PIXEL OUTPUT
0500 PIXEL LDAB X1
0510       ADDB #96
0520       BSR SENDB
0530       LDAB #160
0540       SUBB Y1
0550       BSR SENDB
0560       RTS
0570 SHOW2 BRA SHOW
0590 * INITIALIZE PIA INTERFACE TO JOYSTICK
0600 JINIT LDAA #$3D
```

Figure 13. Continued

```
0610       LDX #$8008
0620       LDAB #$00
0630       STAB 2,X
0640       STAA 3,X
0650 * SAVE MACHINE REGISTERS ON INPUT
0660 START STAA ASAVE
0670       STAB BSAVE
0680       STX ISAVE
0690 * GET HORIZONTAL STICK POSITION
0700       LDX #$8008
0710 LOOP1 LDAA 3,X
0720       ANDA #$80
0730       BEQ LOOP1
0740       LDAA 2,X
0750       TAB
0760       ANDA #$80
0770       BNE LOOP1
0780       ANDB #$3F
0790       SUBB #$20
0800       BGE RTN1
0810       NEGB
0820       ADDB #64
0830 RTN1  STAB JHPOS
0840 LOOP2 LDAA 3,X
0850       ANDA #$80
0860       BEQ LOOP2
0870       LDAA 2,X
0880       TAB
0890       ANDA #$80
0900       BNE GOBACK
```

Figure 13. Continued

```
0910        BRA LOOP1
0920 GOBACK ANDB #$3F
0930        SUBB #$20
0940        BGE RTN2
0950        NEG B
0960        ADDB #64
0970 RTN2   STAB JVPOS
0980        LDAA ASAVE
0990        LDAB BSAVE
1000        LDX ISAVE
1010        RTS
1020 * ROUTINE TO DRAW LINE FROM (X1,Y1) TO (X2,Y2)
1030 SHOW   CLRA
1040        LDAB #1
1050        STAB M
1060        STAB N
1070        LDAB X2
1080        SUBB X1
1090        STAB D
1100        BGE BP1
1110        NEG M
1120        NEG D
1130 BP1    BNE BP2
1140        LDAA #$FF
1150 BP2    LDAB Y2
1160        SUBB Y1
1170        STAB E
1180        BGE B8963
1190        NEG N
1200        NEG E
```

Figure 13. Continued

```
1210 B8963  JSR PIXEL
1220        LDAB X1
1230        CMPB X2
1240        BEQ B8990
1250 B8969  TST A
1260        BLT B8981
1270        LDAB X1
1280        ADDB M
1290        STAB X1
1300        SUBA E
1310        BRA B8963
1320 B8981  LDAB Y1
1330        ADDB N
1340        STAB Y1
1350        ADD A D
1360        BRA B8963
1370 B8990  LDAB Y1
1380        CMPB Y2
1390        BNE B8969
1400        RTS
1410 X1 EQU $0050
1420 X2 EQU $0052
1430 Y1 EQU $0051
1440 Y2 EQU $0053
1450 M EQU $0054
1460 N EQU $0055
1470 D EQU $0056
1480 E EQU $0057
1490 HPOS EQU $0058
1500 JHPOS EQU $0059
```

Figure 13. Continued

```
1510 JVPOS EQU $005A
1520 ASAVE EQU $005B
1530 BSAVE EQU $005C
1540 ISAVE EQU $005D
1550  ORG $7200
1560 WELCOM FCC *       WELCOME TO THE M6800 FLIGHT SIMULATOR PROGRA
1570          FCC * THIS PROGRAM IS DESIGNED    TO ALLOW THE USER TO*
1580          FCC * SET UP THE AIRCRAFT'S INITIAL POSITION COORDINATE
1590          FCC * FROM   . USER PROMPTS, ENGAGE THE FLIGHT SIMULATOR
1600          FCC * AND PRESENT AN UPDATED GRAPHICS        DISPLAY AS
1610          FCC * STUDENT PILOT ATTEMPTS TO MANUEVER THE PLANE FOR
1620          FCC * LANDING ON    AN AIRSTRIP. TO USE THE PROGRAM YOU
1630          FCC * FIRST MUST CHOOSE AN AIRPORT DATABASE.
1640          FCC * PLEASE DEPRESS THE SPACEBAR FOR MORE INSTRUCTIONS
1650          FCC *                     *
1660          FCB $04
1670  ORG $7400
1680 MSGC FCC *                         SET THE COMM FREQ TO 119.2 AND *
1690       FCC *THE NAVG FREQ TO 110.3 MHZ     *
1700       FCC *                         SET THE ALTIMETER TO 1000 FEET. *
1710       FCC *                                    *
1720       FCC *                         SET THE MAGNETIC COMPASS TO READ*
1730       FCC * 360 DEGREES.                       *
1740       FCC *                         SET THE GYRO COMPASS TO READ 360*
1750       FCC * DEGREES.                           *
1760       FCC *                         SET THE AIRCRAFT'S ATTITUDE SO TH
1770       FCC *HAT THE WINGS ARE                                   EVE*
1780       FCC *N (NO BANK) AND LEVEL FLIGHT (NO PITCH) IS ATTAINED.
1790       FCC *        *
1800       FCC *           WHEN YOU ARE READY TO BEGIN FLYING PLEASE*
```

Figure 13. Continued

```
1810       FCC * ENABLE THE PROGRAM BY                   DEPRESSING THE*
1820       FCC * SPACEBAR ON YOUR TERMINAL. HAVE A GOOD FLIGHT.!!*
1830       FCC *        *
1840       FCB $04
1850  ORG $7700
1860 MSGL FCC *          THANKYOU FOR FLYING WITH US TODAY ON PARSONS*
1870       FCC * PIPER AIRWAY'S. PLEASE               REMAIN SEATED *
1890       FCC *UNTIL THE AIRCRAFT HAS COME TO A COMPLETE STOP. HA*
1900       FCC *VE                 A GOOD DAY.*
1910       FCB $04
1920  ORG $7800
1930 X3 RMB 2
1940 Y3 RMB 2
1950 Z3 RMB 2
1960 PITCH RMB 2
1970 BANK RMB 2
1980 HEAD RMB 2
1990 M1 RMB 2
2000 N1 RMB 2
2010 N2 RMB 2
2020 N3 RMB 2
2030 N4 RMB 2
2040 N5 RMB 2
2050 GDOPT RMB 1
2060 NEWDAT RMB 1
2070 ENABLE RMB 1
2080 IDXREG RMB 2
2090 FLGDUN RMB 1
2100  ORG $78F0
2110 COUNT RMB 1
```

Figure 13. Continued

```
2120 STKSAV RMB 2
2130 CNT RMB 1
2150  ORG $7900
2160 MSGA FCC *          CHOOSE THE AIRPORT DATABASE AS FOLLOWS: *
2170      FCC *                                  *
2180      FCC *              1 = PARSON'S FIELD AT TETERBORO, NEW*
2190      FCC * JERSEY.                          *
2200      FCC *              2 = ARTWICK'S FIELD AT TETERBORO, NE*
2210      FCC *W JERSEY.                         *
2220      FCB $04
2230  ORG $7A00
2240 MSGC FCC *           NOW TO INITIALIZE THE SIMULATOR, CONTINUE*
2250      FCC * BY PRESSING THE SPACEBAR.      *
2260 * THIS CONCLUDES ALL OF THE PROGRAMMING FOR THE M6800 FLIGHT
2270 * SIMULATOR PROGRAM.
2280  END
```

Figure 13. <u>Continued</u>

```
S1130000E780000080017FFF000000000000007FFF08
S1130010000000000000007FFF002A000112EE2046CD
S11300204001F9B8FFA0006140FFFA80FFA0006121
S11300304000FA80FFA0006112E70000006120D4B4
S11300408659BD005DBD005ABD005ABD005ABD00B1
S11300505ABD006620BE865A20E80EA600B780195E
S1130060B6801827FB39B68018810326F9B68019A3
S11300700839 26E8B68019BDE1D120E0CE8004A677
S11300800002B1EBDE1AC810D260A860ABDE1D1BD62
S113009000AD20E8810327D0BD00DE7E007CB68061
S11300A018810326D7B68019BDE1D120CFB600F957
S11300B0444444448A40B78019BDE1D1B600F9F6FE
S11300C000FA58495849843F8A40B78019BDE1D1A4
S11300D0B600FA843F8A40B78019BDE1D13984075C
S11300E0B700FEB600F9F600FA584958495849FADE
S11300F000FBB700F9F700FA3900000004731204D42
S113010000000000000000000000000007FFFC81095
S11301100018000000000000000007FFFA80018800 42
S113012060C43884071B8A40B78019B6801827FB3F
S1130130FF0144FE050009270DFF0500FE014408EE
S113014020DA03300831FE0142FF05007E0017DE8D
S113015008CD6EDFA8DB78D9E8D58ECFD4D6B88A62F
S113016090EA906A872C9F09B46F812E8506A46F4C
S1130170A57D856EFD0E816AE968B5492D6481759A
S11301800C9CAF1BF9EFD908F99EE8ADE99FF88B6AC
S113019008D3E89FEA9DF8DAEADDEADE6D89EC844A6
S11301A024ECA56AF46E806AB42AA462A132900E8E
S11301B0886CCD6CC86CA9DEA550AD22A46B81758A
S11301C08DACD9DFA06EF9F88DAECDDA9DFE91CA63
S11301D0E5BEA8FACCBEE1BEADFE89EE8D7BC1665C
S11301E0A0CFC1898560F7A88906DD6D9C06F86EED
S11301F0D964ADE6DF669DAAF9CDA944AC46A87ED4
S11302009701D700DF02DF042012DE04399701D7FB
S1130210009604970296059703200139D600D80268
S1130220D74096002A084F7000019200970096026A
S11302302A084F700003920297024F5F7400012452
S11302400049B03D902544674000124049B03D9027D
S11302505446740001240 49B03D9025446740001DE
S113026024049B03D902544674000124049B03D93B
S1130270025446740001240 49B03D902544674 00BA
S11302800124049B03D902544674000124049B03F3
S11302900D90254467400002404 9B03D902544674C2
S11302A0000024049B03D90254467400002404 9BDE
S11302B003D902544674000024049B03D902544613
S11302C07400002404 9B03D902544674000002404DF
S11302D09B03D902544674000024049B03D902549E
S11302E0467D00402A06D7405F40D2403978CCA8EA
S11302F08B2A806A893FCC2EA948256E2142A124ED
```


Figure 14. Memory Dump of 3D to 2D Converter Routine (Taken from Artwick, 1977)

```
S1130300E600A601BB0101F90100F7011BB7011CBE
S1130310E602A603BB0103F90102F7011DB7011EA2
S1130320E604A605BB0105F90104F7011FB7012086
S1130330DF38FE011BDF00DE06DF02BD021CD73AF8
S1130340973BFE011DDF00DE0CDF02BD021CD73C23
S1130350973DFE011FDF00DE12DF02BD021C9B3B46
S1130360D93A9B3DD93CDE18E700A701FE011BDF0B
S113037000DE08DF02BD021CD73A973BFE011DDFF9
S113038000DE0EDF02BD021CD73C973DFE011FDFDD
S113039000DE14DF02BD021C9B3BD93A9B3DD93CD5
S11303A0DE18E702A703FE011BDF00DE0ADF02BD41
S11303B0021CD73A973BFE011DDF00DE10DF02BDB1
S11303C0021CD73C973DFE011FDF00DE16DF02BD95
S11303D0021C9B3BD93A9B3DD93CDE18E704A70598
S11303E0DE383938881AD194D010F8AAC12089F29D
S11303F0890E99F8A8E899FFC96CAD508969BD5579
S1130400ED9FAFDEDDD445DDCD4FAD4EDF9E35D360
S1130410ADDECCD6FDDBE51AADDAFEDFA59E7DB6FD
S11304208D6605C4848505C4C4C68594855CC5777A
S113043085EC841C84E6C507A52484E484468C46A4
S1130440CCDD6CDECDC4ED4E4F5C47DC3F55CDDEDC
S11304508F5EC4D70DDF6EDC8716CEDFDF5F3F4EC5
S11304600744CC46856E1694846C05EC85258C4532
S11304700850C64358505852CC5E44D8404CC848EB7
S1130480D95E4DD6CD565C1CD5CFCDDE8F5CCC5F1A
S1130490CD5E4C4E4C5DCD5D0F5C8DD7EFFEEC4ECA
S11304A0848F4CC6856605A695248DA215C71D9418
S11304B08504842E45C4C5F48D0D855495442D863C
S11304C0CF5EBDD49D4D45DECFDFCDCD8D5EDDD776
S11304D0EDF97DC697DE0C4C157D04DF175EEDDF6C
S11304E004648C06B494045D06C485469C5685E673
S11304F0152585AC252F842C066E3DB5AC070785E4
S1130500005000508001F70500B70501DE3AFF05F1
S1130510028D06FE0504DF0439B605002B0BB60573
S1130520022B0220348D1A20238D0BB605022B02D8
S1130530201A8D0D20235F700501F20500F70500D8
S1130540395F700503F20502F705023 98D0E5F7000
S11305500505F20504F7050439F60500B605017F23
S11305600505CE000FB00503F205022E107005052E
S1130570780505790504485909271720E878050501
S1130580790504485909270ABB0503F905022BED2F
S113059020DFB605042A07FE050409FF0504391308
S11305A013055545021315055546017267000000F1
S11305B0001221177425177405177431177411016B
S11305C060000000000055350021356055355017393
S11305D0270553500213570553550213520553531D
S11305E0053774071353111350001334155354056E
S11305F0135100133715535507776700533107 60BC
```


Figure 14. Continued

```
S11306009623D622DE268D18D73C9625D624DE26C0
S11306108D0EDE1EE702D63CE7018655A700201F9B
S1130620DF3ABD0506F6010E545A4FBD020D39DE00
S11306301E8655A7000909A600A703A601A70496CC
S11306402BD62ADE2E8DD9D73C962DD62CDE2E8D98
S1130650CFDE1EE704D63CE7030808080808DF1EBF
S11306607E092C57043100055560043100071560DC
S11306700515610215550555620215620777660175
S11306801422055562075561017414021561031 59E
S1130690600431000555611555440215621555462F
S11306A002154705551702156307747100343407A2
S11306B0630701143603355505600005556505 5576
S11306C066003452056000055565055566021550EF
S11306D00555170215510555060215640774710076
S11306E03455076307011457033557056000055552
S11306F0700555670034730560000555670555702E
S1130700962884BF270A84DF270984EF275C200604
S11307107E08167E07C29627D626902FD22E973EA8
S1130720D73A9625D624902DD22C903ED23A973E9B
S1130730D73A962FD62E902DD22CBD05069623D6C9
S113074022902BD22ABD020D9B2ED92A972ED72A74
S11307509627D626902FD22EBD020D9E2FD92E97E9
S11307602FD72E972DD72C7E0A429627D626902F48
S1130770D22E973ED73A962DD62C9025D224903E57
S1130780D23A973ED73A962FD62E9B2DD92CBD051E
S113079006962 3D622902BD22ABD020D9B2ED92A52
S11307A0972ED72A9627D626902FD22EBD020D9EA3
S11307B02FD92E972FD72E5F40D22E972DD72C7E50
S11307C00A429627D626902FD22E973ED73A9623C5
S11307D0D622902ED22A903ED23A973ED73A962FE7
S11307E0D62E902ED22ABD05069625D624902DD23E
S11307F02CBD020D9B2DD92C972DD72C9627D626B0
S1130800902FD22EBD020D9E2FD92E972FD72E9726
S11308102BD72A7E0A429627D626902FD22E973E94
S1130820D73A962ED62A9023D222903ED23A973EA2
S1130830D73A962FD62E9B2ED92ABD05069625D6B8
S113084024902DD22CBD020D9B2DD92C972DD72C65
S11308509627D626902FD22EBD020D9E2FD92E97E8
S11308602FD72E5F40D22E972BD72A7E0A42F834F8
S1130870F934E994E9B4E934F924FBB4EB36E9B585
S1130880DF54EB17DB37EB16DB46CF77CB174D7610
S1130890FB57DB17DB27EB17D95FD967DB55FF1E47
S11308A0E95EE930E934F920F933A920E926EB754A
S11308B0E900C934FB24CB24F9B0FB25F934F1B5A4
S11308C0DB54DF37DB17DB77CB06CB37DD36FB6753
S11308D0CB17DB77DF16D717EB564B73EF16FB7787
S11308E0F904FB10FB34EB34D934E914DB34F9702C
S11308F0E934E924E934A9A0ED24E9A5E9A1E9B59D
```

Figure 14. Continued

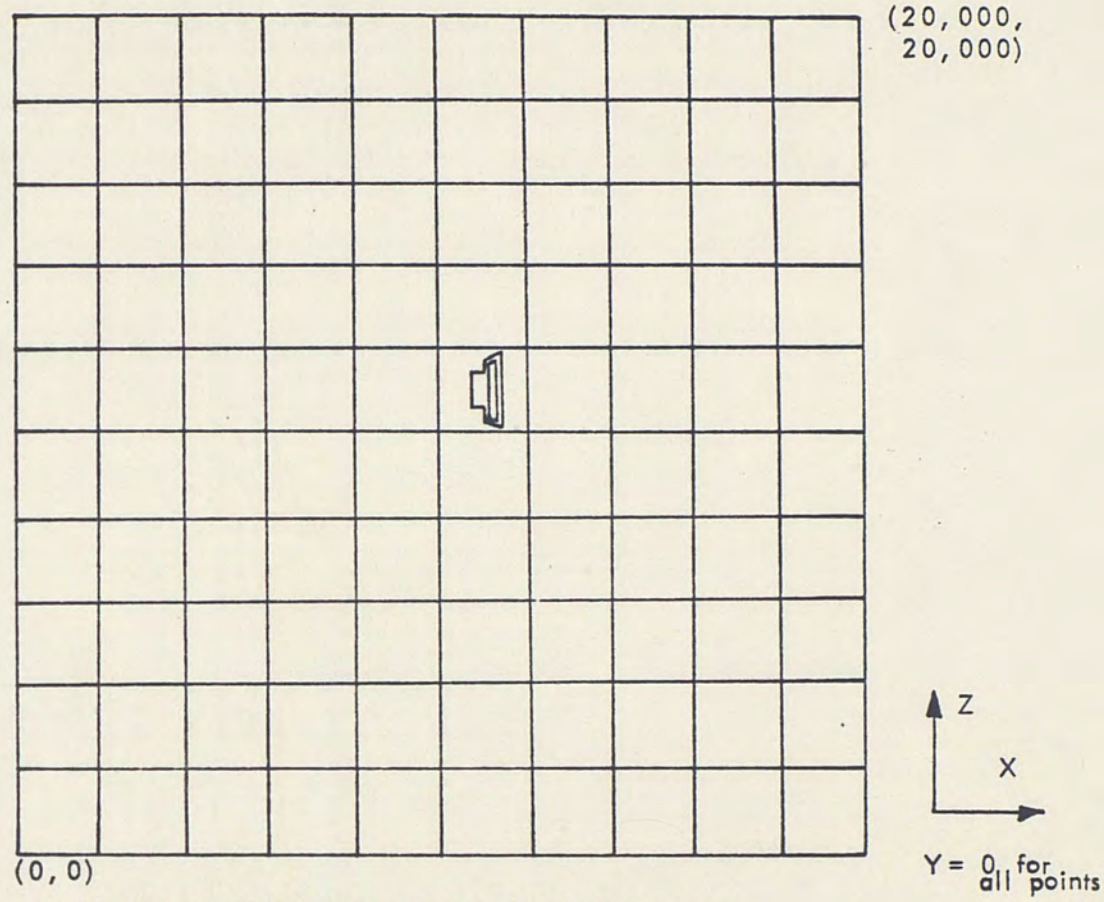
```
S1130900FE010FDF1CFE0111DF1ECE0106DF3ACE11
S11309100113DF3CC606DE3AA60008DF3ADE3CAC3D
S11309200027037E0D8008DF3C5A26EADE1CA60061
S1130930812527108126272DDE1E8604A700DE1CB4
S113094008DF1C3908C60086229719D718BD030092
S1130950BD09ACBD0988BD0A00BD09AC9620942828
S113096026CA7E0AB1DF38DE32DF22DE34DF24DE3F
S113097036DF2696309720DE38BD0988BD09AC964F
S113098020942826A77E0ADD8600C62A9718D71940
S113099008BD0300DF38DE2ADF32DE2CDF34DE2E32
S11309A0DF36DE38BD0A4296289730390808080831
S11309B00808DF1C3935F9B0F934E926E136E9B421
S11309C0DF77FB46DB77D157FF27CF16DB174B7753
S11309D0CD16DD5FFB37CF37DB56FB57FF57DF37CD
S11309E0E927DB24E934EB34E9A4E924E9E0E92745
S11309F0E934C8B6EB36A924F924EB20F910F1B593
S1130A007F0020D62296239B27D9262C0486409744
S1130A1020D62696279023D2222C0696208A209729
S1130A2020D62496259B27D9262C0696208A109713
S1130A3020D62696279025D2242C0696208A08971D
S1130A4020397F0028D62A962B9E2FD92E2C04865A
S1130A50409728D62E962F902BD22A2C0696288A99
S1130A60209728D62C962D9E2FD92E2C0696288A93
S1130A70109728D62E962F902DD22C2C0696288AA5
S1130A8008972839 8D06BD07008D01399F38DE2A65
S1130A909E22DF229F2ADE2C9E24DF249F2CDE2E22
S1130AA09E26DF269F2EDE289E20DF209F289E384C
S1130AB039860A97427D002026107D002827 39BDFB
S1130AC007007A004226F37E092CBD0A847A00428C
S1130AD027F5962027E4942827F07E092C860A9788
S1130AE0427D002026E47D0028270ABD07007A0005
S1130AF04226F320D27E062F7E060034E935EBB57C
S1130B00EF16FB56DF76EF37CB57CB76DA774B17FA
S1130B10CB56FB17EF16DF27DB57CB34DB16DB177F
S1130B20F967F9B5F934FBB6E9B5E9A5F936FBF689
S1130B30F934FB35F934F9B4E834EB36F924E9B582
S1130B40CF56D936EF77FB57DF77DB76EA37DB37DB
S1130B50FB13D937FB769B74FB37CB57EF06DB7758
S1130B60F926D934F935E935EBE4C9B7EFB5E934F9
S1130B70F325FD73E924EAB4F937E9B4DDA4F9B740
S1130B80FB575B76FF37CF57F957E9365D36DB26DF
S1130B90DB76CB374B57DB56C907DB1EDB06DF3F63
S1130BA00C00C0405D2D21C1402D11CB805058F761
S1130BB00BA1FE0BA04F5FA001E2003958F70BA177
S1130BC0FE0BA0E600A60139502ADC20E175CB77A4
S1130BD0DB37D937EF26FD766D67D937CB16DB3790
S1130BE0F9B7F974F9B7FB30F974F934FBD5F93274
S1130BF0E965F9F3FBE4D926F935F935A920E9814A
```


Figure 14. Continued

```
S1130C007FFF7FF57FD77FA67F617F097E9C7E1C57
S1130C107D897CE37C297B5C7A7C79897883776E1A
S1130C20764075037 3B5725470E16F5E6DC96C23C1
S1130C306A6C68A566CE64E762F160EE5ED65CB36D
S1130C405A81584255F4539A51334EBF4C3F49B3DD
S1130C50471C447A41CD3F163C56398C36B933DEB5
S1130C6030FE2E102B1F2826252722231F191C0E8F
S1130C7018F915E112C70FAE0C8C096A0647032457
S1130C80000CB79CBF943FDCBF9CBE95BE96379EC8
S1130C90F64E3F84B79DBE94B784A6DFBE95B6A436
S1130CA0774063C57749684067C066412440634C18
S1130CB06641674060C0664162C06740610C674545
S1130CC0BF8F3D96BE97B784B745B794B79D3FDDB8
S1130CD0F6C6268757D62E96769EB6D73FDAAFD698
S1130CE07649774066 4D674125442F4C75406741EE
S1130CF06748674965407345 2E4862406141674 5CE
S1130D00BE8CBE94BE9C3F8C3F9F3E9CF686AE8DAF
S1130D10B78CB7CCB5C4BEDCAFDCBEDF379CBF4CF1
S1130D2062456551 6D433345 67416BD167446949F9
S1130D3062456F40674066406140278122412C44F0
S1130D40BFD53F9D3F952F8EBE85B7083E8DBEDEA5
S1130D50BE5CB6CDB78DB68EB6D6B61CB75DB67CCD
S1130D607ED96740EF887B44774861416448634596
S1130D70714063C066406644214127C476416045A2
S1130D80F60107F70114BD0EA2D7209721F601093C
S1130D90F70116BD0BA2D7229723F6010BF7011812
S1130DA0BD0EA2D7249725F60107BD0EA4D7269720
S1130DBC27F60109BD0EA4D7289729F6010EBD0E13
S1130DC0A4D72A972BDE28BD0200D72C972DDE222C
S1130DD0D6249625BD0200D72E972FD62A962BBD52
S1130DE0020DD7309731DE24D6289629BD0200D7CC
S1130DF0329733DE20D62E962FBD0200D734973596
S1130E00D6329633BD020DD7369737D6309631BDDC
S1130E10020DD73A973ED62C962DBD020DD73C97A1
S1130E203DD62C962D9B35D934D7069707D63696C2
S1130E30379031D2309709D708D6249625DE26BDBF
S1130E40020DD70A970BD6229623BD020DD70C9722
S1130E500DD6289628BD020DD70E970F4F5F90210F
S1130E60D220D7109711D63A963E9033D232D7126C
S1130E709713D62E962F9B3DD93CD7149715D62A77
S1130E80962EBD020DD7169717FE010CFF0119D63C
S1130E900A960BBD0200D70A970ED6109611BD0215
S1130EA00D9711D710D6169617BD020DD7169717A2
S1130EB07E092C443744AE45664160402340 75C5E5
S1130EC0B69CFE9CB79CBF0CBFD43E8DB68CB7CDF0
S1130ED0BE8EBE8CFF9CBE96BEDDB65CB786A6DF1A
S1130EE06EC163D0674063CC664475C5E74566C48C
S1130EF06F58E5CD6647764D6641654021406645AD

S9 END
```

Figure 14. Continued



S1131100250000000000000264E200000000002500FD
S11311100000007D0264E200000007D02500000064
S1131120000FA0264E2000000FA025000000000178D
S11311307026 4E2000001770250000000001F402676
S1131140 4E2000001F402500000000002710264E20DE
S11311500000271025000000002EE0264E2000008D
S1131160 2EE025000000000036B0264E20000036B0E8
S113117025000000003E80264E2000003E80250011
S113118000000004650264E2000004650250000076
S11311900004E20264E2000004E2025000000000B6
S11311A0002600000004E20250 7D000000000002685
S11311B007D000004E20250FA000000000260FA03D
S11311C000004E20251770000000000261770000054
S11311D04E20251F4000000000261F4000004E2026
S11311E025271000000000262710000004E20252E81
S11311F0E0000000000262EE000004E202536B0005E
S1131200000000002636B000004E20253E800000007D
S11312100026 3E8000004E20254650000000002697
S11312204650000004E20254E2000000000264E208F
S11312300004E202520240000 2E9A262AF80000B7
S11312402DF0262AF800002C8826 29CC00002C88B2
S11312502629CC000002968262AF800002968262AB5
S1131260F8000027F6262C24000 2724262C24002E
S11312700002E9A252C0600002E6B262E1000002D21
S11312 80E1262B1000002805262C060000276026E6
S10A12902C0600002E6E0085 S9

Figure 15. Memory Dump of Artwick's Database
(Taken from Artwick, 1977)

S11313002527380000 20E426273800004290252E--

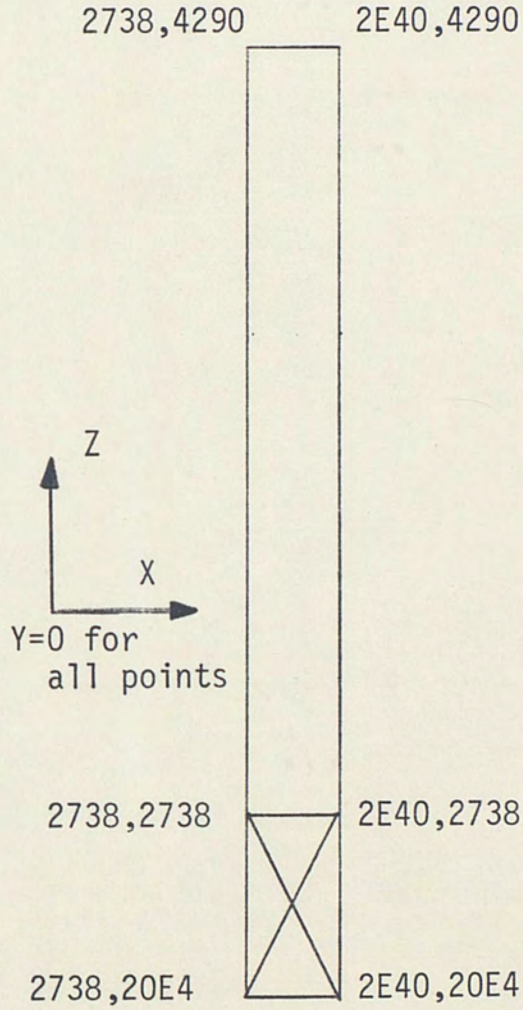


Figure 16. Memory Dump of Parson's Field

APPENDIX C

INPUT PARAMETER SCALING

AND DIRECTION CONVENTIONS

A/D Converter Input As Follows

(+) West/(-) East
X3 (Bit 4 = 1)
+28,500 ft = 2.50 V = FA
0 ft = 1.25 V = 7D
-28,500 ft = 0.00 V = 00

(+) South/(-) North
Z3 (Bit 5 = 1)
+28,500 ft = 2.50 V = FA
0 ft = 1.25 V = 7D
-28,500 ft = 0.00 V = 00

Altitude
Y3 (Bit 1 = 1)
0 ft = 2.49 V = FA
1000 ft = 1.71 V = AB
2000 ft = 1.29 V = 81
3000 ft = .875 V = 58
4000 ft = .499 V = 32
5000 ft = .00 V = 00

Pitch
Pitch (Bit 0 = 1)
Nose Up Max = 2.5 V = FA
Level = 1.25 V = 7D
Nose Down Max = 0 V = 00

Bank
Bank (Bit 3 = 1)
-90 = 2.5 V = FA
-60 = 2.13 V = D5
Left -30 = 1.68 V = A8
-20 = 1.55 V = 9B
-10 = 1.40 V = 8C
Wings Level 0 = 1.25 V = 7D
10 = 1.08 V = 6C
20 = .907 V = 5B
Right 30 = .755 V = 4C
60 = .380 V = 26
90 = .00 V = 00

Heading
Head (Bit 2 = 1)
-180 degrees = 2.5 V = FA
-90 degrees = 1.68 V = A8
0 degrees = 0.86 = 56
+90 degrees = 0.00 = 00

3D to 2D Routine Expects:

(+) West/(-) East
XV = MSB
XV + 1 = LSB
Max = 7FFF = +32,767
Mid = 0000 = 0
Min = 8000 = -32,768

(+) South/(-) North
ZV = MSB
ZV + 1 = LSB
Max = 7FFF = +32,767
Mid = 0000 = 0
Min = 8000 = -32,768

Altitude
YV = MSB
YV + 1 = LSB
Max = 7FFF = +32,767
Mid = 0000 = 0
Min = 8000 = -32,768

Pitch
PV = MSB = 00 Always
PV + 1 = LSB
Max = 7F = 178.59 degrees
Mid = 00 = 0 degrees
Min = 80 = -180 degrees
Each integer unit = 1.4065 degrees

Bank
BV = MSB = 00 Always
BV + 1 = LSB
Max = 7F = 178.59 degrees
Mid = 00 = 0 degrees
Min = 80 = -180 degrees
Each integer unit = 1.4065 degrees

Heading
HV = MSB = 00 Always
HV + 1 = LSB
Max = 7F = 178.59 degrees
Mid = 00 = 0 degrees
Min = 80 = -180 degrees
Each integer unit = 1.4065 degrees

Figure 17. A/D Converter Input and 3D to 2D Routine Formats

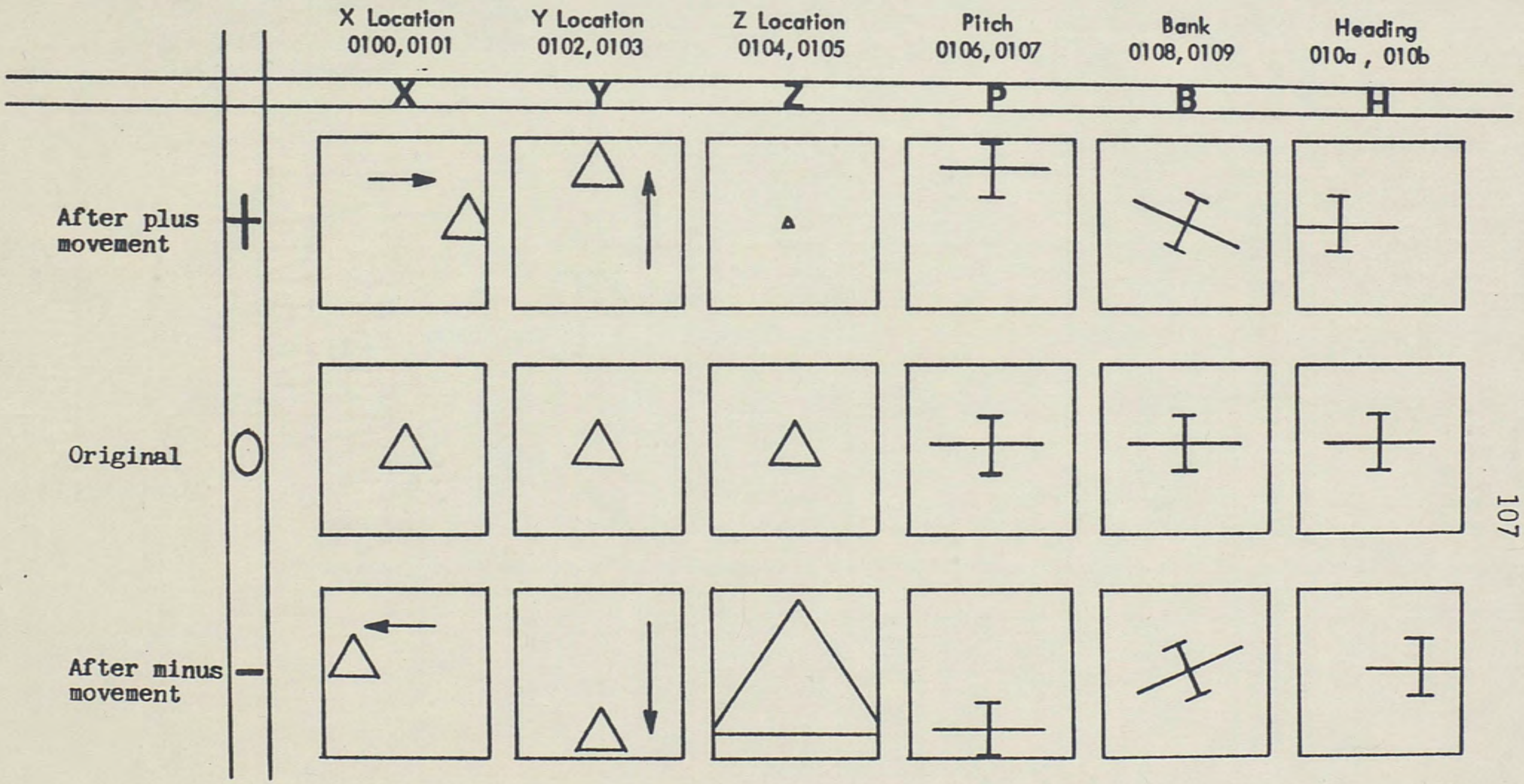


Figure 18. Direction of Movement Conventions
(Taken from Artwick, 1977)

APPENDIX D

PRE-AMP AND AD-68A BOARD EXPLANATION

The AD-68A Analog to Digital Converter card is the actual M6800 link to the ATC-610J Flight Simulator Cockpit. It is preceded by the Pre-Amp board shown in Figure 19.

The Pre-Amp board is to scale the incoming analog signals such that the maximum voltage output will be equivalent to 2.50 volts and the minimum will be 0.00 volts. The Pre-Amp board is powered by the Power Supply shown in Figure 20.

The AD-68A board itself has eight opto-isolated inputs which respond to external stimuli from the Pre-Amp board and eight reed relay outputs suitable for switching low power boards allowing information into the computer. In fact the AD-68A can be thought of as registers of an I/O Port. The relays are controlled by software which will write an eight bit word. A channel or relay is enabled when its corresponding bit is set. Only one channel may be selected at any one time so the software must act much like a polling routine would. A more detailed explanation of the operation of the device can be obtained from the Innovative Technology's Model AD-68A Analog-to-Digital Converter Operation Manuals. Figure 21 is a block diagram of the AD-68A circuit operation and Figure 22 shows a flow chart of the software routine required to operate the board.

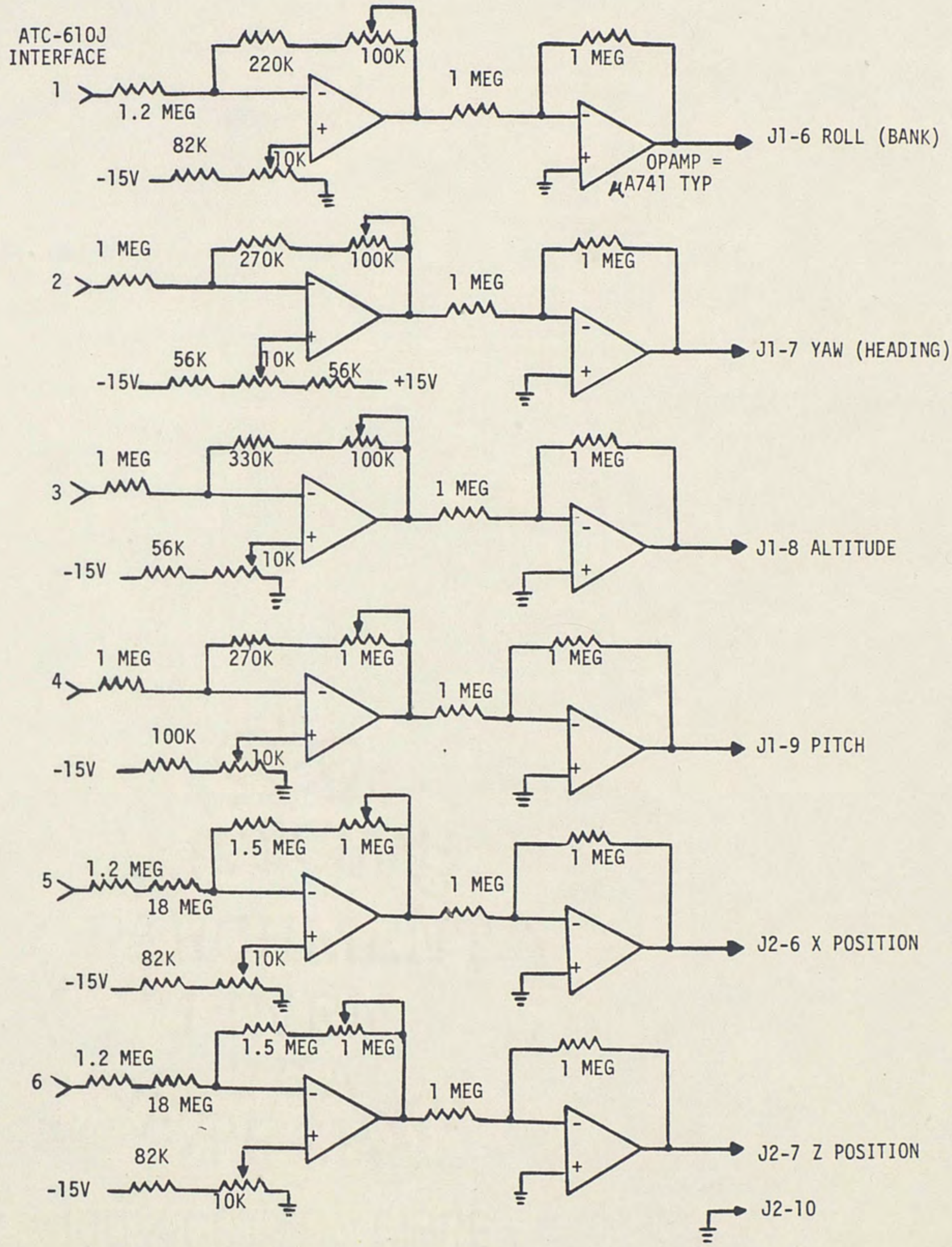


Figure 19. Pre-AMP Board

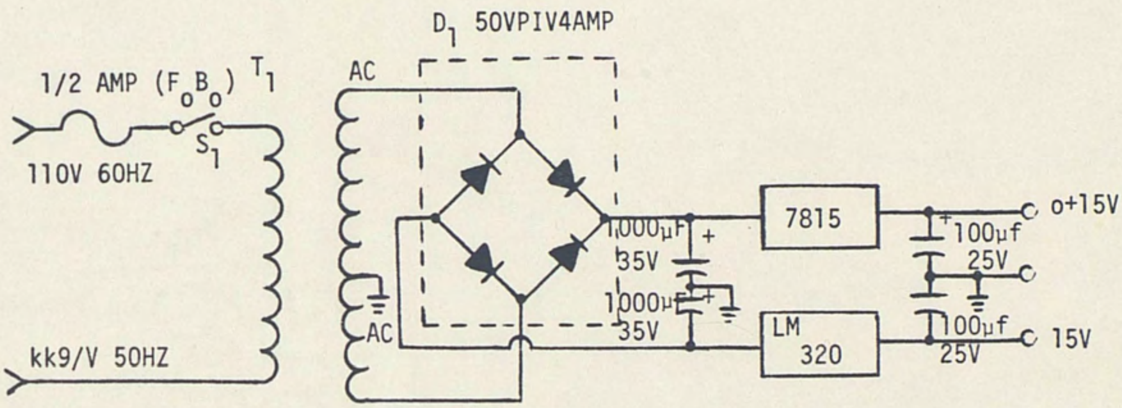


Figure 20. Pre-AMP Power Supply

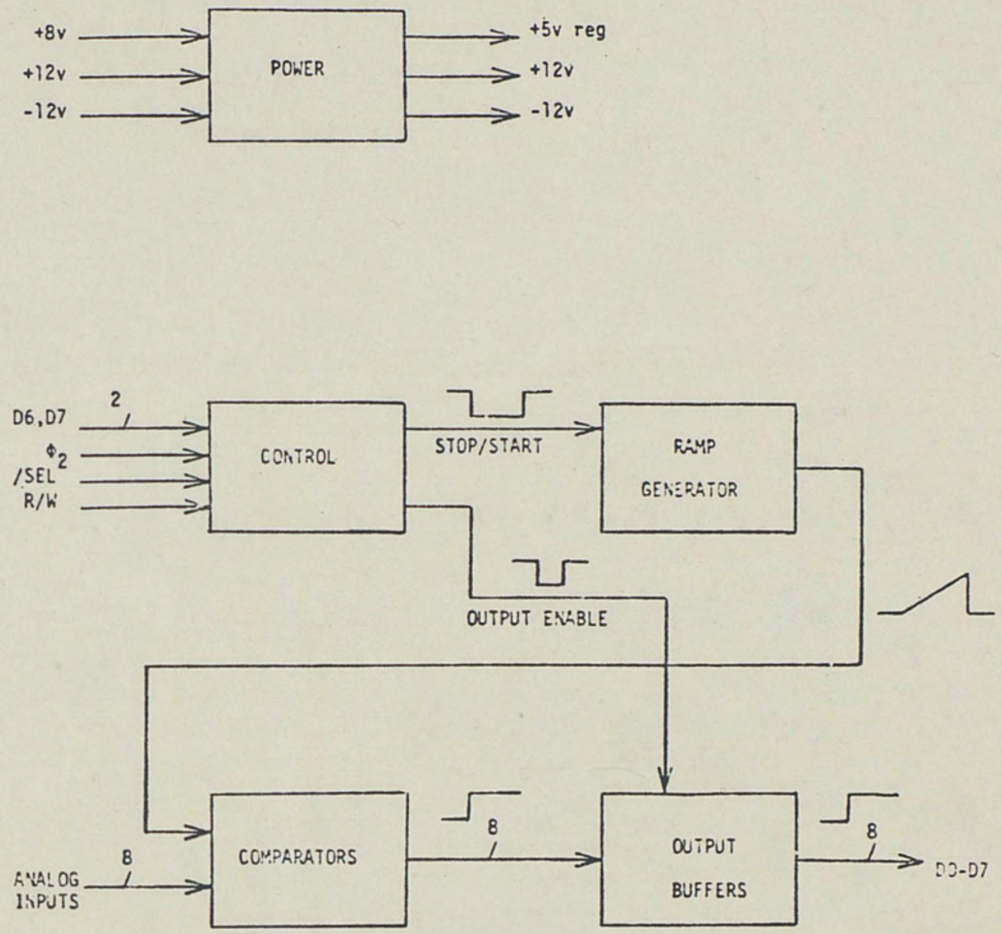


Figure 21. Block Diagram: AD-68A

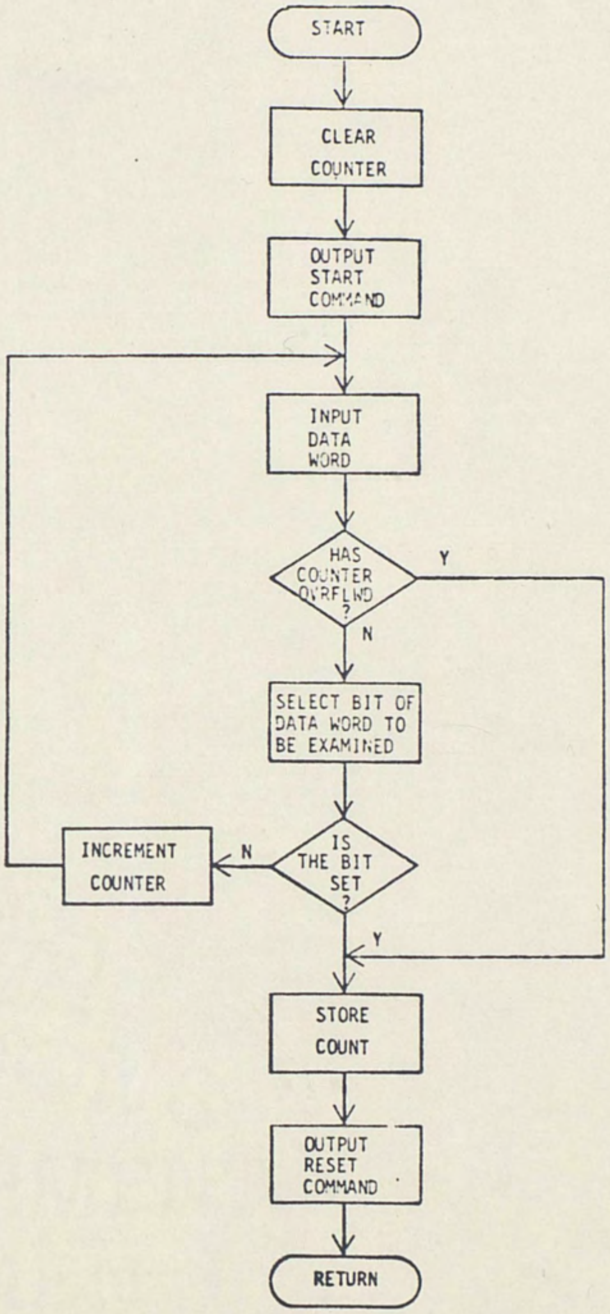
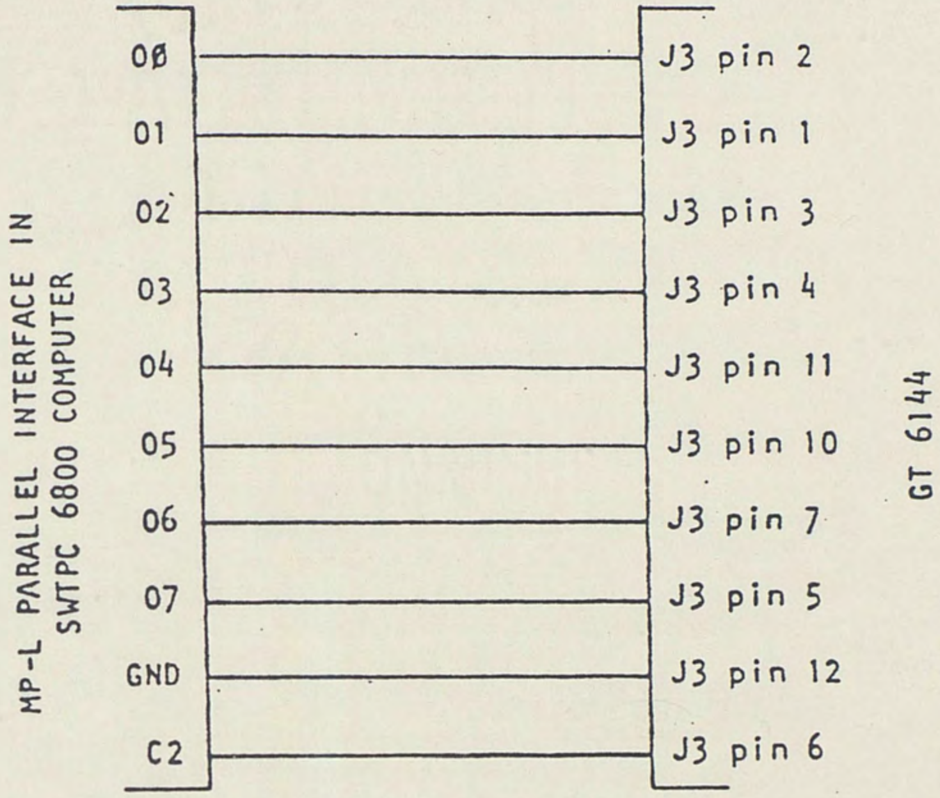


Figure 22. Flow Chart: AD-68A Analog Data Conversion Routine

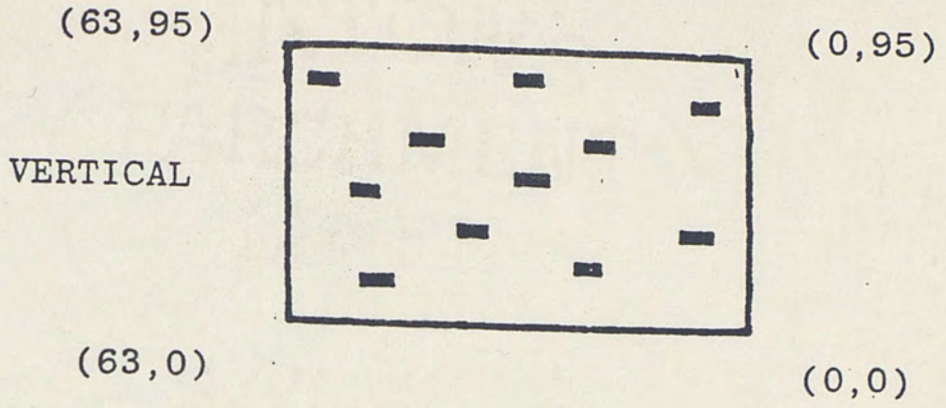
APPENDIX E

GT-6144 OPERATION



Programming

The display of the GT-6144 graphics terminal consists of 6144 small rectangles formatted 64 across and 96 down that can be turned on or off at will under software control. In order for the GT-6144 to do a particular function the data fed to it must be formatted correctly. The coordinate of a particular location is referenced from the bottom right corner of the screen with the first square residing at location (Ø, Ø).



When data is input to the GT-6144 the first byte (8bits) sent to the terminal must be the HORIZONTAL POSITION. The actual position is determined in bits $B_{\emptyset} - B_5$ and is in binary. When bit 6 = 0 a rectangle will be removed at the desired coordinates, bit 6 = 1 a white rectangle will be generated. Bit 7 must always equal Ø for the terminal to know that a HORIZONTAL position is being loaded. A Ø in the bit 7 position causes the data holding flip flops in the terminal to store the present data.

The second byte from the computer contains the VERTICAL coordinate. The location is contained in binary in bit $B_{\emptyset} - B_6$ of this second byte while bit 7 must equal a 1.

FIRST BYTE FROM COMPUTER - HORIZONTAL POSITION

B_7	B_6	B_5	B_4	B_3	B_2	B_1	$B_{\emptyset}$
Ø							

B_5 – $B_{\emptyset}$: Horizontal Position

Ø ——— $63_{1\emptyset}$

B_6: DATA BIT-
Set to 1 to write a rectanble.
Set to Ø to erase a rectangle.

B_7: FF LOAD BIT
this bit should always = Ø for a horizontal position.
When this bit equal Ø bit Ø - 6 are stored in the DATA HOLDING FLIP-FLOPS.

SECOND BYTE FROM COMPUTER - VERTICAL POSITION

B_7	B_6	B_5	B_4	B_3	B_2	B_1	$B_{\emptyset}$
1							

B_6 – $B_{\emptyset}$: Vertical Position

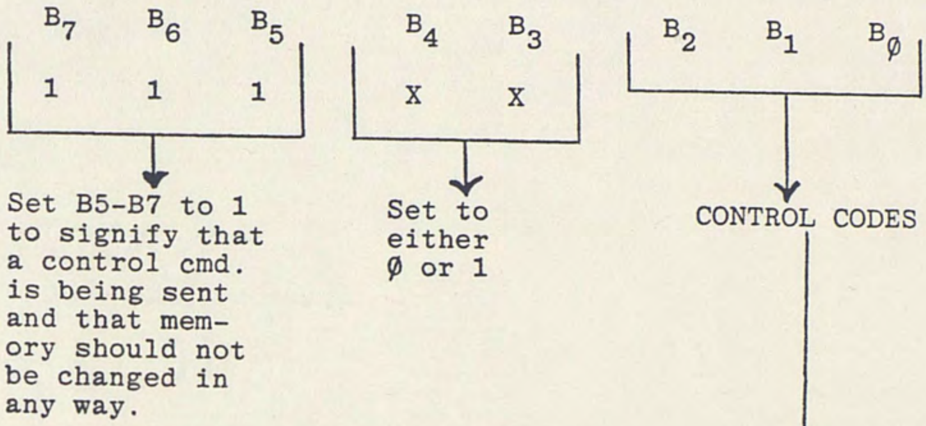
Ø — $95_{1\emptyset}$

B_7: MEMORY STORE BIT
This bit should equal 1 for a vertical position.
A 1 in this position causes the data in the holding flip flops and the data in bits $B_{\emptyset}$ - B_6 to be transferred to the memory.

When programming a design to appear on the screen there are two ways the characters can be loaded - the method used depends on the application. One method is to just send out successive coordinates (H_1V_1); (H_2, V_2) etc. until all H, V locations are specified. With this method two bytes must be sent out for each character. This method is used to write the final transformed From, To vectors to the display screen.

Another method can be used that can result in saving time and memory space. In this method the HORIZONTAL position of a particular column is loaded only once into the terminal. The VERTICAL coordinate of all other characters that have this same HORIZONTAL coordinate can then be loaded by themselves since the HORIZONTAL position is latched in the holding flip flops. This is the method used to erase the display screen.

Since there are 96 characters to be accessed in the vertical direction at least seven address lines must be used. Seven lines gives the possbility of addressing 2^7 (128_{10}) locations giving us 32_{10} extra undefined locations. These extras can be used as control commands for controlling BLANKING ON/OFF, REVERSE SCREEN, etc. The correct format for control commands for the GT-6144 terminal is as follows:



$B_\emptyset$	B_1	B_2	
0	0	0	INVERTED SCREEN
0	0	1	NORMAL SCREEN
0	1	0	DISABLE CT-1024
0	1	1	ENABLE CT-1024
1	-0	0	ENABLE GRAPHICS
1	0	1	BLANKED GRAPHICS
1	1	0	NOT USED.
1	1	1	NOT USED.

NORMAL SCREEN: In the normal screen mode white characters appear on a black background. This applies both to graphics and alphanumeric data.

INVERTED SCREEN: In the inverted screen mode all characters appear as black characters on a white background.

BLANKED GRAPHICS: In this mode no video from the GT-6144 is transferred to the video monitor. This gives an "all rectangles off" condition. This condition does not effect the status of alphanumeric data.

ENABLE GRAPHICS:	In this mode video from the GT-6144 is transferred to the monitor and mixed with alphanumeric data if this data is enabled.
ENABLE CT-1024	In this mode video data from the CT-1024 is mixed with video from the GT-6144. If the 6144 is disabled, only alphanumeric data will appear on the screen.
DISABLE CT-1024	No CT-1024 data is mixed with the GT-6144 video data.

When writing input-output programs care should be taken to optimize them for speed and memory conservation.

LIST OF REFERENCES

Analog Training Computers, Inc. ATC-610 J/K Personnel Flight Simulator Service Manual. West Long Beach, NJ: February, 1977.

Artwick, Bruce. Three Dimensional Microcomputer Graphics M6800 Assembly Language. Culver City, CA: Sublogic Co., 1977.

Barry, C. D.; Ellis, R. A.; Graesser, S. M.; and Marshall, G. R. "Display and Manipulation in Three Dimensions." In Pertinent Concepts in Computer Graphics, Proceedings of the Second University of Illinois Conference on Computer Graphics, pp. 104-153. Edited by M. Faiman and J. Nievergelt. Urbana, IL: University of Illinois Press, 1969.

Blinn, James F., and Newell, Martin E. "Clipping Using Homogeneous Coordination." In Tutorial: Computer Graphics, pp. 197-203. Edited by Kellogg S. Booth. Waterloo, Ontario: University of Waterloo, 1978.

Campbell, Jerry W. "A Microcomputer Implementation of a Flight Simulator Visual Display System." Research report, University of Central Florida, 1979.

Giloi, W. K. Interactive Graphics: Data Structures, Algorithms, Languages. Englewood Cliffs, NJ: Prentice Hall Inc., 1978.

Guedj, Richard A., and Tucker, Hugh A., eds. Methodology In Computer Graphics, IFIP Workshop on Methodology in Computer Graphics, Selliac, France, May 1976. Amsterdam: North-Holland Publishing Company, 1979.

Holley, Larry. "A Search for a Computer Graphics System for the Small-Scale Computer User." Research report, Florida Technological University, 1975.

Nake, F., and Rosenfeld, A. "Preface." In Graphic Languages, Proceedings of the IFIP Working Conference on Graphic Languages, Vancouver, Canada, May 22-26, 1972. Amsterdam: North-Holland Publishing Company, 1972.

O'Callaghan, John F. "Use of a Picture Language to Generate Descriptions of Line Drawings in an Interactive System." In Graphic Languages, Proceedings of the IFIP Working Conference on Graphic Languages, Vancouver, Canada, May 22-26, 1972. pp. 123-143. Edited by F. Nake and A. Rosenfeld. Amsterdam: North-Holland Publishing Company, 1972.

Paul, R.; Falk, G.; and Feldman, J. A. "The Computer Representation of Simply Described Scenes." In Pertinent Concepts in Computer Graphics, Proceedings of the Second University of Illinois Conference on Computer Graphics, pp. 87-103. Edited by M. Faiman and J. Nievergelt. Urbana, Ill.: University of Illinois Press, 1969.

Takasawa, Yoshimitsu; Moriguchi, Shiggichi and Sakamaki, T. "A Graphics Manipulating Language." In Graphics Language, Proceedings of the IFIP Working Conference on Graphics Languages, Vancouver, Canada, May 22-26, 1972, pp. 327-333. Edited by F. Nake and A. Rosenfeld. Amsterdam: North-Holland Publishing Company, 1972.

Williams, Robin. "A General Purpose Graphic Language." In Graphics Languages, Proceedings of the IFIP Working Conference on Graphics Languages, Vancouver, Canada, May 22-26, 1972, pp. 334-349. Edited by F. Nake and A. Rosenfeld. Amsterdam: North-Holland Publishing Company, 1972.