DIGITAL GENERATION OF CONTOUR MAPS
FOR RASTER SCAN DISPLAY

ARMY ELECTRONICS COMMAND, FORT MONMOUTH
NEW JERSEY

DECEMBER 1976
DIGITAL GENERATION OF CONTOUR MAPS FOR RASTER SCAN DISPLAY

Victor Vajo
Avionics Laboratory

December 1976

DISTRIBUTION STATEMENT
Approved for public release; distribution unlimited.
Digital Generation of Contour Maps for Raster Scan Display

 Victor Vajo

Control Theory Team
Advanced Avionics Systems Tech Area
Avionics Laboratory, Fort Monmouth, NJ 07703

12. REPORT DATE
DECEMBER 1976

15. SECURITY CLASS. (of this report)
UNCLASSIFIED

Approved for public release; distribution unlimited

19. KEYWORDS (Continue on reverse side if necessary and identify by block number)

Computer Generated Map
Computers
Computer Mapping Techniques

Television
Map Display
Digital Map Generation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
The study proved the feasibility of generating digital contour maps for display on standard TV monitors. Computer programs were written in assembly language for the Singer SKC-2000 Airborne Computer which generate two color (black and white) contour maps for display on a standard 525 line television system.
NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The citation of trade names and names of manufacturers in this report is not to be construed as official Government endorsement or approval of commercial products or services referenced herein.

Disposition

Destroy this report when it is no longer needed. Do not return it to the originator.
# TABLE OF CONTENTS

1. INTRODUCTION .............................. 1
2. BASIC CONCEPT ............................ 1
3. DEFINITION OF DIGITALLY GENERATED MAP SYSTEM ........................ 2
4. DIGITALLY GENERATED MAP EVALUATION SYSTEM ............................. 3
5. INPUT DATA GENERATION ...................... 3
6. RASTER DISPLAY ........................... 4
7. DIGITAL-TO-VIDEO INTERFACE .............. 5
8. SOFTWARE PROGRAM .......................... 5
9. SUMMARY AND CONCLUSIONS ................. 8

## APPENDICES

A. SINGER KEARFUTT COMPUTER NO. 2000 INSTRUCTION SET .................. 11
B. APPENDIX B – LISTINGS OF MAP GENERATOR COMPUTER PROGRAMS .......... 13

## FIGURES

1. Paw Paw, West Virginia – 4,000 data prints ......................... 9
2. Paw Paw, West Virginia – 250 data points .......................... 9

![Distribution availability codes]

[Redacted]
1. INTRODUCTION

This report is concerned with the development of a digitally generated contour map to be displayed on standard raster TV for use in Army aircraft. The requirement for a display of this type is generated by the operations of Army aircraft in map-of-the-earth (NOE) flight during both day and night operation. NOE flight in this case refers specifically to pilotage at or below tree top level.

Current operations of this type are carried out with the pilot dedicated solely to the task of piloting the vehicle. During this mode of flight, the pilot has the time neither to navigate nor communicate with anyone other than the copilot. For this reason the performance of all other tasks are left to the copilot. In addition, the copilot must cue the pilot as to the character of the upcoming terrain including general navigational instructions. Required terrain information is obtained by the copilot from hand-held maps. The copilot mentally integrates the map contour information and verbally passes upcoming terrain characteristics to the pilot. This task performed by the copilot would be difficult enough under normal conditions, but in the environment of a vibrating helicopter at night, the task becomes even more difficult.

The first phase of this study is directed toward improving the transfer of information concerning the approaching terrain to the pilot, thereby reducing the hazards of NOE night flight.

2. BASIC CONCEPT

Any study designed to solve a particular problem must begin with an attempt to determine whether existing equipment is available to solve the problem being investigated.

Map display systems for aircraft are not a new concept. Currently there are in existence various analog map displays such as map plotters, projected map displays, and also stroke written or CRT type map displays. Each of the above suffers from at least one drawback, namely the fact that they require valuable instrument panel space. In addition, most of the projected map and map plotters present only a north-up display as opposed to an aircraft heading-up display. Those units which can produce a heading-up display, do so at a significant increase in unit cost. Because of these factors and an indication that future Army aircraft may have an integrated digital display and possibly Forward Looking Infra-Red (FLIR) or Low Light Level Television (LLLTV), another avenue of approach was selected for investigation.

An alternative technique to providing a map display is to digitally generate the map (DGM) and display the information in standard raster scan television format. Provided the DGM could be proven feasible, several advantages were immediately apparent. The information could be displayed on panel mounted display (PMD) associated with the FLIR OR LLLTV or even a Helmet Mounted Display (HMD), either alone or superimposed by simply video mixing. The requirement for a separate panel display would thereby be eliminated. In addition, since data for the map would be in digital form, the technique would provide the capability for a much more versatile display. A heading-up display, aircraft position indication, and map scale changes would present no problem using the DGM System.
With the emergence of the microprocessor and the significant reduction in the size and cost of computer memories required for data storage, the DGM appeared to be a reasonable area for investigation.

3. DEFINITION OF DIGITALLY GENERATED MAP SYSTEM

Typical maps used by Army aviators have scales of 1:25,000, 1:50,000 or 1:100,000. Since these maps contain a considerable amount of information, three problems immediately present themselves. The first is how much of the data from the maps must be stored, second, what format or scheme is best for storing the data and third, from where is the digital data to be obtained.

Currently, digital terrain elevation information in grid line format is available from the Defense Mapping Agency for certain areas of the United States. The grid line format refers to the fact that elevation information is stored corresponding to fixed X-Y grid increments over the entire map. Since the size of the map and grid resolution is known, the number of data points can be determined, thus fixing the size of the data base. However, it was felt that data storage in grid format would not offer the best approach, since it requires computer time to generate contour information from the grid data. Therefore, direct storage of data in contour format seemed appropriate.

Contour format means the DGM data will be piece-wise linear approximation to the contour lines. Each pair of consecutive points defines the beginning and end of a line segment. The points are stored in contiguous order. In other words, a line drawn sequentially connecting the points would trace out a contour closing on itself and then begin to trace the next contour interval.

Storing the data in contour format eliminates the computational time required to convert from grid to contour data format. It also provides another distinct advantage in that it facilitates a variable data density. This allows more detailed information to be encoded where terrain variations are severe and less information in areas of less terrain variation. A prime example of the advantage of variable data density (contour format) over fixed data density (grid format) is shown by the fact that grid data provides the same data density over a large lake as it does over the peak of a mountain, whereas the contour format can provide greater data diversity in the mountainous area where it is needed and none over water areas. In using a variable data density, however, the size of the data base required to digitally encode a given size map is undefined. Data base size now becomes a function of the severity of the terrain in the area mapped.

It is possible to develop a computer program to generate a contour data base from the Defense Mapping Agency grid format data base. However, it was felt that development of the data conversion program was too time consuming a project on which to expend much effort prior to the establishment of the feasibility of the DGM system. For this reason, a contour data base was encoded manually from existing maps for usage as a test model. Therefore, in summary, a contour format data base was employed in this study and because of the nature of the data base, the amount of data required for a given map size was variable. Additionally, for the purposes of this study, the data bases were generated manually.
4. DIGITALLY GENERATED MAP EVALUATION SYSTEM

In the laboratory test facility, an airborne digital computer (a Singer SKC-2000) was used to develop the computer programs to generate data for the raster display. A special digital to video converter (DVC) was fabricated in-house, especially designed to interface with the airborne computer. The SKC-2000 computer was chosen since an identical computer was installed on the laboratory's CH53 Experimental Vehicle for Avionics Research (EVAR); therefore, after laboratory development of the software programs, the system could be transferred to flight test with minimum difficulty. The design of the DVC was such that it was rugged enough for use during the flight testing.

The SKC-2000 is a 32 bit hexadecimal machine with hardware floating point. Both the laboratory and airborne computers contain sixteen thousand words of memory. Each machine has a teletype with cassettes and a standard size airborne magnetic tape unit. The laboratory model has, in addition, a paper tape punch and reader, a card reader, and a printer/plotter. The printer/plotter allowed the development of the computer programs to be undertaken prior to the fabrication of the DVC. Obviously, the laboratory evaluation system contained sufficient flexibility for a study of this type. All programming was done in assembly language and a listing of the computer assembly language instruction set is given in Appendix A. The instruction set contains several bit manipulation instructions which proved to be extremely useful, especially in encoding the data base.

5. INPUT DATA GENERATION

As previously mentioned, the digital map data (i.e., contour lines) are piece-wise linear approximations to the contour lines. A pair of points define the start and end of each line segment and the sequence of storage of the points gives the path of the contour.

The raw data is obtained in decimal format with X coordinate and Y coordinate specified for each point. Some reference point must be chosen as the origin (0,0) of the data; therefore, the lower left corner of the area to be mapped was selected. A data point beginning a new contour interval is specially flagged to indicate a contour connecting line (non-contour line), enabling the elimination of the line during display.

Since the data in decimal format is not directly usable by the computer, a program was written to convert it to fixed point binary data. The program listing is found in Appendix B. Some use was made of the architecture of the machine in converting to the binary format. Since the computer is capable of both full word (32 bits) and half word (16 bit) addressing, a special scheme for storing the binary data was used. Two types of data words were employed. The first type is called a start point, which is the absolute value of the location of the data point referenced to the origin (0,0). The start point data is encoded as two consecutive 16 bit words, one for X-position and other for Y-position. A start point is identified by the most significant bit of the first 16 bit word being set to one. A hidden line indication is incorporated by using the first bit of the second 16 bit word as that indicator. A one in this bit position signifies the data point is a contour connecting line.
rather than a contour line itself and should not be displayed. Therefore, 15 bits remain to represent absolute X-position and 15 bits for absolute Y-position. With a scale factor of one (i.e., the least significant bit of the 15 remaining bits represents one foot in ground distance), the maximum possible range of X or Y is \((2^{15} - 1)\) or 32,767 feet (approximately 6 miles). This scale factor allows encoding of a map 6 miles by 6 miles. Obviously larger areas may be encoded using larger scale factors with corresponding loss in resolution.

The second type of data word is a delta data word. In order to conserve memory, a delta word format is used to indicate incremental X and Y position referenced to a start word rather than the origin. The delta word is 16 bits in length including both \(\Delta x\) and \(\Delta y\). The first 8 bits are \(\Delta x\), the last 8 bits are \(\Delta y\). The most significant bit of \(\Delta x\) must always be zero indicating it is not a start word. The most significant bit of \(\Delta y\) is used as the hidden line indicator, being set to one if it is a connecting line. There remains 7 bits (6 bits + sign) for indicating \(\Delta x\) and 7 bits (6 bits + sign) for specifying \(\Delta y\). In other words, with a scale factor of one, changes in X or Y position of less than or equal to \(\pm (2^6 - 1)\) or \(\pm 63\) feet can be specified using a delta word format.

The delta word format is of significant use in map areas of high information content, since greater detail is encoded by using smaller line segments to describe the contour variations. All of the data bases used in this study were obtained manually and no attempt was made to automate the data base creation.

6. RASTER DISPLAY

Before describing the raster display utilized in this study, a very brief description of standard television systems is given here for comparison.

Standard television uses a 525 line system, meaning that there are 525 horizontal sweep lines or raster lines used in generating a normal 4 by 3 aspect ratio TV picture. The 525 lines are divided into two fields of 262-1/2 raster lines each, called odd and even fields. These two fields are displayed alternately at the rate 1/60 cycle per second. This rate is rapid enough so the eye is not able to perceive any flicker. The camera and associated electronics photographing the scene obviously generates the video in a compatible odd-even field format for transmission. In reference to resolution, the vertical resolution is divided into 525 discrete raster lines, while the horizontal resolution is very nearly continuous.

The raster display used in this study also utilized a 525 line system, but the 4 by 3 aspect ratio picture was not maintained. Rather, it was decided to use a square picture which significantly reduced the complexity of the software. In order to generate the digital equivalent of a TV picture (which is an analog display rather than digital), it was required to form the picture by using a matrix of discrete dots. The study was directed at achieving only a two color level black and white display. No attempt was made to incorporate any shades of gray capabilities. Using only black or white, simplified both computer programming and interface hardware. The dot matrix consisted of an
array of 256 by 256 discrete points comprising the picture on the screen, each
dot capable of having only two possible states either black or white. This
binary scheme was compatible with the computer storage of the picture infor-
mation, thereby requiring a minimum amount of memory. A single bit in the com-
puter contained the information for one pixel on the monitor; if the bit is set
to one, it appears as a white dot and, if it is set to zero, it appears as a
black dot. As stated before, the TV monitor used was a 525 line system and in
order to make the 256 by 256 bit matrix compatible with the 525 line system,
the same 256 by 256 matrix was used for both the odd and even fields. In ad-
cision, the horizontal sweep was adjusted to obtain a square picture main-
taining the same resolution in both the horizontal and vertical direction.
Again, this was done to simplify the hardware and reduce computer memory re-
quirements.

7. DIGITAL-TO-VIDEO INTERFACE

The purpose of the digital-to-video interface or digital-to-video con-
verter (DVC) was to accept the computer generated picture information at com-
puter rates and display the information at video rates. To accomplish this,
the DVC had incorporated within it two separate banks of 256 by 256 bit mem-
ories. As one memory was being written into by the computer, the other was
being displayed. When the second memory had been filled by the computer, that
memory bank became the display memory and the first became the one being writ-
ten into by the computer. This ping-ponging of memories continued at the
fastest rate allowed by the computer. The switching of memory banks was re-
quired, since reading and writing of the same memory simultaneously would cause
distortion in the displayed picture. The DVC serially transferred data from
the computer by means of direct memory access, eight 32 bit words (256 bits)
at a time. This format of transfer was selected since the computer generated
one raster line (256 bits) at a time. In this manner, the display was opera-
ting asynchronously from the computer, repeating the same digital map until
an updated map had been completed. Under all of the operating conditions, the
DVC was able to accept data much faster than the computer could generate it.

8. SOFTWARE PROGRAM

The description of the assembly language program developed to generate
the digital may be divided into five major sections.

- Determine and save in a temporary memory buffer those data points
  which are within the field-of-view.
- Rotate the selected data into the aircraft reference system.
- Determine the intersection of the contour lines with the raster lines.
- Store the computed intersections in the output buffer with the proper
  bit patterns for output to the display.
- Output the computed data.

A more detailed presentation of the program's operation follows.
a. A Determination of Data Points within the Field-of-View. The data base generated in the format described previously is loaded into the computer memory before starting the main program. The next step is to acquire the aircraft parameters required by the program, namely X-position and Y-position relative to the map origin, and aircraft heading. The parameters are generated and varied by means of a control subprogram (listed in Appendix B), which is capable of varying these three parameters as well as the scale factor by fixed increments through the use of control switches on the computer. The entire data base is then scanned to find those data points that are located within a distance $R_{\text{max}}$ of the aircraft position relative to the data base. $R_{\text{max}}$ is the radius of a circle whose magnitude is equal to one half of the diagonal of the display field-of-view (FOV). The radius of a circle is used rather than a square since the data has not been rotated into the aircraft frame of reference. In order to check the data points with respect to $R_{\text{max}}$, they must be uncompressed. Those points found to be within a distance $R_{\text{max}}$ are stored in another area of computer memory in uncompressed format to be used during the intersection process. At this time another function is also performed. During the scanning of the data base some contour lines leave the FOV. At the point where this occurs, a new start word is created. The scan of the data continues until the contour returns to the FOV or the end of the data is reached. If the contour returns to the FOV, a hidden line is created connecting this point and the point at which the contour exited. As stated previously, hidden lines are lines which do not appear on the screen. This operation, in effect, creates a new data base whose boundaries are entirely within the FOV. The creation of the FOV data base reduces the number of points which must be rotated and scanned for possible intersection resulting in a reduction of processing time during succeeding operations.

b. Coordinate Rotation of the FOV Data Base. All of the data points in the FOV data base are rotated into the aircraft reference system using the following Euler Coordinate Transformation Equations.

\[
\begin{align*}
X_{\text{ac}} &= X \cos \psi + Y \sin \psi \\
Y_{\text{ac}} &= Y \cos \psi - X \sin \psi
\end{align*}
\]

where, $\psi$ is the aircraft heading $(X,Y)$ are the original coordinates $(X_{\text{ac}}, Y_{\text{ac}})$ are the coordinates referenced to the aircraft heading.

After the transformation, the FOV data base is in the proper format for the raster line intersection processing.

c. Raster Line Intersection Processing. Raster line processing involves checking all line segments (except hidden lines) defined by the pairs of points in the FOV data base for their possible intersection with a raster line. This operation is performed for each of the 256 horizontal raster lines.

The computation begins with the selection of the raster line located at the uppermost portion of the FOV and continues by successive increments to the raster line located at the bottom of the FOV. Representation of successive raster lines is obtained by incrementing $Y$ scan by a fixed $\delta y$ scan. In order to compute the intersection of these raster lines with the line segments in the FOV data base, computation of the slope of the segment is necessary.
The remaining computation is a simple determination of the intersection of two straight lines. The determination of the intersections results in a series of \( X_i \)'s for each \( Y \) scan \( i \).

A certain amount of computational time may be saved by first checking to determine if the \( Y \) value of a particular raster line lies within the boundaries specified by the \( Y \) values of the line segments terminal points. In other words, \( Y \) scan \( i \) must lie between \( Y \) \( \text{seg}_m \) and \( Y \) \( \text{seg}_{m+1} \) for an intersection to exist. Slopes are computed only after it has been found that an intersection occurs, thereby resulting in computational savings. The \( X_i \)'s resulting from actual intersections are stored in their proper bit positions in the output scan word buffer.

d. Formatting the Output Scan Word Buffer. The intersection previously determined must next be placed in the proper bit position of the output buffer. Within the computer, a raster line is represented by 256 bits or eight 32 bit scan data words. Each scan data word will contain a one in each bit position for which an intersection occurred. All other bit positions will contain zeros. Using the desired resolution, the determined \( X_i \) is divided by the proper scale factor (X scale) to obtain the bit position into which a one must be placed. The scale factor is determined quite simply, namely X scale is the horizontal FOV width divided by 256 bits. The proper bit position is found as follows:

\[
\text{Bit position } X_n = \left\lfloor \frac{X_i}{X \text{ scale}} \right\rfloor
\]

This scheme results in only one bit being placed on a raster line for each intersection with that raster line. Actually, this scheme results in a satisfactory representation of contour line segments that are in the range of \( \pm 45 \) degrees of vertical with respect to the horizontal raster lines. A problem arises as the contour lines begin to approach being parallel to the raster lines. In this case more than one bit must be placed on a raster line in order to make the line look continuous. The obvious extreme case is a contour line exactly parallel to the raster line. If the contour line falls in between two raster lines, a decision is made as to which raster line the contour line should appear on. These various problems are solved by several different bit filling techniques.

The horizontal contour line is the most easily solved problem after determination of the proper raster line has been made. In this case bits are filled on the raster line between the limits determined by the \( X \) range of the line segment, namely between \( X_m \) and \( X_{m+1} \). Contour lines whose angle with respect to the raster are greater than \( 0^\circ \) but less than \( 45^\circ \) present greater difficulty. This difficulty is increased due to the limitation of having only one raster line in the computer memory at a time rather than having the entire 256 by 256 matrix with which to work. A relatively successful approach was taken to solve the problem. The problem simply stated is -- how many bits must be placed on a raster line for intersections with grazing contour lines. The technique used was to place that number of bits on the raster line that correspond to the absolute value of the reciprocal of the slope of the contour line segment. For example, a line segment intersecting the raster line with a slope of 1/2 (relative to the raster) would have two bits
placed on the raster line at the intersection point, a line with a slope of 1/3 would have these bits, those with a slope of 1/4 would have four bits and so forth. Lines with slopes which have fractional parts were rounded to the nearest whole number. This operation significantly increased the computational time.

e. Output to display. The output of the data to the display is facilitated by using the direct memory access (DMA) capability. This requires only that the number of words and the memory location of the first word be specified. The actual transfer is accomplished by one program statement.

9. SUMMARY AND CONCLUSIONS

Results of this study illustrated the feasibility of generating from digital data a dynamic contour map capable of being displayed on a standard raster TV. Figures 1 and 2 show the results as seen on the screen of two different data bases. Figure 1 shows an area one-half mile on each side known as Paw Paw in West Virginia. This data base was obtained manually from a 1:24000 scale map of the area. The contour intervals displayed are at 100 foot intervals as opposed to 20 foot intervals on the source map. There are approximately 4,000 data points in the field-of-view and the time required to generate the picture is about 1 minute and 45 seconds. As can obviously be seen, there is too much information to facilitate easy interpretation. This suggests that this is probably a worst case condition as far as data processing time since any more information would be of no benefit. One point which must be made is that, even though the display is cluttered with information, it does not show all of the information contained on the source map. This suggests that a one-to-one correspondence between paper map and the digital display of a map is not possible within hardware constraints.

Figure 2 shows a simplified data base of the same area as Figure 1. The data base of Figure 2 contains 250 data points and requires only 10 seconds to generate the full picture. Admittedly, this map is rather sketchy and in no way approximates a typical paper map. However, looking a little more closely at the map, the general trend of the terrain is much more obvious in Figure 2 than in Figure 1. The winding river can be seen and the trend of the mountains is more obvious. This is possible with a reduction in the data storage requirement by a factor of 16 and a reduction in computational time by a factor of 10.

Two possible conclusions can be made from the results of Figures 1 and 2. The first is that if a large amount of terrain detail is desired, possibly some format other than standard map contour format should be investigated, since a one-to-one representation of paper maps is not possible. The second conclusion is that if only the trend of the terrain is needed, this technique could be employed directly with satisfactory results.

In reference to the time required to generate the maps, these times may seem to be long; however, no extreme effort was made to achieve rapid execution of the program. It is felt that with minimum effort the program execution times could be reduced significantly. A factor that should be kept in mind is that a helicopter traveling at 20 knots as it does in nap-of-the-earth flight would not necessarily require rapid updates of the map. In fact, if the map could be continuously updated, the continuous movement of the map display would be confusing rather than helpful.
Figure 1. Paw Paw, West Virginia - 4,000 data points

Figure 2. Paw Paw, West Virginia - 250 data points
It is anticipated that further investigations will be made in the area of digital map generation. These investigations will include the generation of a shades of gray or even color map as opposed to the two color black or white display used in this study. Another area of possible investigation will be alternate formats for the map data, as was previously mentioned. Possibilities for alternate formats may include; slope shading, relief shading, and ridge valley lines. Future investigations will also include flight testing of a digital map system and a projected map display.

This study was successful in that it illustrated the capability of developing software and hardware to generate contour maps from digital data for display on raster TV. The study also pointed out some of the shortcomings of the technique and determined possible areas for future investigations.
<table>
<thead>
<tr>
<th>OP-CODE</th>
<th>LENGTH</th>
<th>MINEMONIC</th>
<th>OPERATION DESCRIPTION</th>
<th>OPERATION SUMMARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000 10...</td>
<td>S</td>
<td>SLLD</td>
<td>Shift A, B Left Logically</td>
<td>Shift By EA</td>
</tr>
<tr>
<td>00000 10...</td>
<td>S</td>
<td>SLCD</td>
<td>Shift A, B Left Circularly</td>
<td>Shift By EA</td>
</tr>
<tr>
<td>00000 10...</td>
<td>S</td>
<td>SLL</td>
<td>Shift A Left Logically</td>
<td>Shift By EA</td>
</tr>
<tr>
<td>00000 10...</td>
<td>S</td>
<td>SRLD</td>
<td>Shift A, B Right Logically</td>
<td>Shift By EA</td>
</tr>
<tr>
<td>00000 10...</td>
<td>S</td>
<td>SRAD</td>
<td>Shift A, B Right Algebraically</td>
<td>Shift By EA</td>
</tr>
<tr>
<td>00001 10...</td>
<td>S</td>
<td>SRC</td>
<td>Shift A Right Circularly</td>
<td>Shift By EA</td>
</tr>
<tr>
<td>00001 10...</td>
<td>S</td>
<td>SRA</td>
<td>Shift A Right Circularly</td>
<td>Shift By EA</td>
</tr>
<tr>
<td>00001 11...</td>
<td>S</td>
<td>SLA</td>
<td>Load A Register</td>
<td>(EA) – A</td>
</tr>
<tr>
<td>00001 11...</td>
<td>L</td>
<td>LDA</td>
<td>Store Index Register</td>
<td>(XR) – EA</td>
</tr>
<tr>
<td>00000 10...</td>
<td>S</td>
<td>STX</td>
<td>Test XR and Skip On Not Equal</td>
<td>Skip If (XR) # (EA)</td>
</tr>
<tr>
<td>00000 10...</td>
<td>L</td>
<td>LCN</td>
<td>Test XR and Skip On Less Than</td>
<td>Skip If (XR) &lt; (EA)</td>
</tr>
<tr>
<td>00000 01...</td>
<td>S</td>
<td>LCM</td>
<td>Load A With EA</td>
<td>EA – A</td>
</tr>
<tr>
<td>00000 01...</td>
<td>L</td>
<td>LCE</td>
<td>Store A Register</td>
<td>(A) – EA</td>
</tr>
<tr>
<td>00000 00...</td>
<td>S</td>
<td>STA</td>
<td>No Operation</td>
<td>No Operation</td>
</tr>
<tr>
<td>00000 00...</td>
<td>S</td>
<td>EEM</td>
<td>Enable Memory Interrupts</td>
<td>SR16 is Set To 1</td>
</tr>
<tr>
<td>00000 00...</td>
<td>S</td>
<td>DPE</td>
<td>Disable Program Interrupts</td>
<td>SR15 is Set To 0</td>
</tr>
<tr>
<td>00000 01...</td>
<td>S</td>
<td>DMR</td>
<td>Disable Memory Interrupts</td>
<td>SR16 is Set To 0</td>
</tr>
<tr>
<td>00000 00...</td>
<td>S</td>
<td>EPI</td>
<td>Enable Program Interrupts</td>
<td>SR15 is Set To 1</td>
</tr>
<tr>
<td>00000 01...</td>
<td>S</td>
<td>HLT</td>
<td>Halt</td>
<td>Halt If Test Equipment Signal Is Present</td>
</tr>
<tr>
<td>00000 10...</td>
<td>S</td>
<td>SET</td>
<td>Set Selected Program Flags</td>
<td>Set Indicated Flags To 1</td>
</tr>
<tr>
<td>00000 11...</td>
<td>S</td>
<td>RST</td>
<td>Reset Selected Program Flags</td>
<td>Restarts Indicated Flags To 0</td>
</tr>
<tr>
<td>00000 00...</td>
<td>S</td>
<td>CFX</td>
<td>Convert Floating To Fixed</td>
<td>(A, B) – A, B</td>
</tr>
<tr>
<td>00000 00...</td>
<td>S</td>
<td>CFR</td>
<td>Convert Fixed To Floating</td>
<td>(A, B) – A, B</td>
</tr>
<tr>
<td>00000 10...</td>
<td>S</td>
<td>EAB</td>
<td>Exchange A And B</td>
<td>(A) = B, (B) = A</td>
</tr>
<tr>
<td>00000 11...</td>
<td>S</td>
<td>SHM</td>
<td>Set Halfword Mode</td>
<td>SR Bit Set To 1</td>
</tr>
<tr>
<td>00000 11...</td>
<td>S</td>
<td>RHM</td>
<td>Reset Halfword Mode</td>
<td>SR Bit Reset To 0</td>
</tr>
<tr>
<td>00000 11...</td>
<td>S</td>
<td>LKA</td>
<td>Load Index Register From A</td>
<td>(A) = XR (13 Low Order Bits)</td>
</tr>
<tr>
<td>00000 11...</td>
<td>S</td>
<td>LKA</td>
<td>Load Index Register From A</td>
<td>Jump To EA</td>
</tr>
<tr>
<td>00000 11...</td>
<td>S</td>
<td>LDA</td>
<td>Data Output From A Register</td>
<td>Jump To EA</td>
</tr>
<tr>
<td>00000 11...</td>
<td>L</td>
<td>LDI</td>
<td>Data Input To A Register</td>
<td>Data Input To Memory</td>
</tr>
<tr>
<td>00000 11...</td>
<td>L</td>
<td>DIM</td>
<td>Data Input To Memory</td>
<td>Load B Register</td>
</tr>
<tr>
<td>00000 10...</td>
<td>S</td>
<td>LDB</td>
<td>Load X Register</td>
<td>(EA) – X</td>
</tr>
<tr>
<td>00000 10...</td>
<td>L</td>
<td>LDB</td>
<td>Load X Register</td>
<td>Carry Status Bits</td>
</tr>
<tr>
<td>00000 10...</td>
<td>S</td>
<td>ALJ</td>
<td>Jump Unconditional</td>
<td>Jump To EA</td>
</tr>
<tr>
<td>00000 10...</td>
<td>S</td>
<td>ALU</td>
<td>Jump Unconditional</td>
<td>Jump To EA</td>
</tr>
<tr>
<td>00000 10...</td>
<td>S</td>
<td>JNA</td>
<td>Jump If A # 0</td>
<td>Jump To EA</td>
</tr>
<tr>
<td>00000 10...</td>
<td>S</td>
<td>JN</td>
<td>Jump If A # 0</td>
<td>Jump To EA</td>
</tr>
</tbody>
</table>

Abbreviations: A Register, B Register, Effective Address, Index Register.
<table>
<thead>
<tr>
<th>OP-CODE</th>
<th>LENGTH</th>
<th>MNEMONIC</th>
<th>OPERATION DESCRIPTION</th>
<th>OPERATION SUMMARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>01100010...</td>
<td>S</td>
<td>JG, JRG</td>
<td>Jump If A _ 0</td>
<td>Jump To EA If (A) _ 0</td>
</tr>
<tr>
<td>0110010000...</td>
<td>L</td>
<td>JG, JAG</td>
<td>Jump If A _ 0</td>
<td>Jump To EA If (A) ≥ 0</td>
</tr>
<tr>
<td>0110011...</td>
<td>S</td>
<td>JL, JRL</td>
<td>Jump If A &lt; 0</td>
<td>Jump To EA If (A) &lt; 0</td>
</tr>
<tr>
<td>0110011000...</td>
<td>L</td>
<td>JL, JAL</td>
<td>Jump If A = 0</td>
<td>Jump To EA If (A) ≤ 0</td>
</tr>
<tr>
<td>01101...001...</td>
<td>L</td>
<td>JGW</td>
<td>Jump On Switch</td>
<td>Jump To EA If Switch On</td>
</tr>
<tr>
<td>01101...010...</td>
<td>L</td>
<td>JGS</td>
<td>Jump On Status Bit</td>
<td>Jump To EA If Status Bit On</td>
</tr>
<tr>
<td>01101...001...</td>
<td>L</td>
<td>JGF</td>
<td>Jump On Program Flag</td>
<td>Jump To EA If Any Flag Tested Is On</td>
</tr>
<tr>
<td>01101...000...</td>
<td>L</td>
<td>JS</td>
<td>Jump To Subroutine</td>
<td>(PC)+2–EA Indirectly, Jump To EA+1</td>
</tr>
<tr>
<td>01101...000...</td>
<td>L</td>
<td>IMP</td>
<td>Modify Index Register Positive</td>
<td>(XH)+(EA) – XR</td>
</tr>
<tr>
<td>01101...000...</td>
<td>L</td>
<td>IMP</td>
<td>Modify Index Register Negative</td>
<td>(XL)+(EA) – XR</td>
</tr>
<tr>
<td>01101...1...</td>
<td>S</td>
<td>RTA</td>
<td>Return Address</td>
<td>Jump Indirect Via EA</td>
</tr>
<tr>
<td>01101...1...</td>
<td>L</td>
<td>RTA</td>
<td>Return Address</td>
<td>Jump Indirect Via EA</td>
</tr>
<tr>
<td>01110...0...</td>
<td>S</td>
<td>STB</td>
<td>Store B Register</td>
<td>(B) – EA</td>
</tr>
<tr>
<td>01110...0...</td>
<td>L</td>
<td>STB</td>
<td>Store B Register</td>
<td>(B) – EA</td>
</tr>
<tr>
<td>01111...0...</td>
<td>S</td>
<td>AND</td>
<td>Logical And</td>
<td>(A) AND (EA) – A</td>
</tr>
<tr>
<td>01111...0...</td>
<td>L</td>
<td>AND</td>
<td>Logical And</td>
<td>(A) AND (EA) – A</td>
</tr>
<tr>
<td>01100...0...</td>
<td>S</td>
<td>SAM</td>
<td>Skip On A Register Masked</td>
<td>Skip Unless (A) and (EA) = 0</td>
</tr>
<tr>
<td>01100...1...</td>
<td>S</td>
<td>SAM</td>
<td>Skip On A Register Masked</td>
<td>Skip Unless (A) and (EA) = 0</td>
</tr>
<tr>
<td>01100...0...</td>
<td>S</td>
<td>MLF</td>
<td>Multiply - Floating Point</td>
<td>(A) × (EA) – A, B</td>
</tr>
<tr>
<td>01100...1...</td>
<td>S</td>
<td>MLF</td>
<td>Multiply - Floating Point</td>
<td>(A) × (EA) – A, B</td>
</tr>
<tr>
<td>01110...0...</td>
<td>S</td>
<td>AFD</td>
<td>Add Floating Double Precision</td>
<td>(A, B) = (EA, EA-2) – A, B</td>
</tr>
<tr>
<td>01110...1...</td>
<td>S</td>
<td>AFD</td>
<td>Add Floating Double Precision</td>
<td>(A, B) = (EA, EA-2) – A, B</td>
</tr>
<tr>
<td>10000...0...</td>
<td>S</td>
<td>ADJ</td>
<td>Add Upper - Fixed Point</td>
<td>(A) + (EA) – Carry – A</td>
</tr>
<tr>
<td>10000...1...</td>
<td>S</td>
<td>ADJ</td>
<td>Add Upper - Fixed Point</td>
<td>(A) + (EA) – Carry – A</td>
</tr>
<tr>
<td>10010...0...</td>
<td>S</td>
<td>ADL</td>
<td>Add Lower - Fixed Point</td>
<td>(B) + (EA) – A</td>
</tr>
<tr>
<td>10010...1...</td>
<td>S</td>
<td>ADL</td>
<td>Add Lower - Fixed Point</td>
<td>(B) + (EA) – A</td>
</tr>
<tr>
<td>10100...0...</td>
<td>S</td>
<td>DVF</td>
<td>Divide - Floating Point</td>
<td>(A, B)/(EA) – A, Remainder – B</td>
</tr>
<tr>
<td>10100...1...</td>
<td>S</td>
<td>DVF</td>
<td>Divide - Floating Point</td>
<td>(A, B)/(EA) – A, Remainder – B</td>
</tr>
<tr>
<td>10110...0...</td>
<td>S</td>
<td>STS</td>
<td>Store Status Register</td>
<td>(SR) = (EA)</td>
</tr>
<tr>
<td>10110...1...</td>
<td>S</td>
<td>STS</td>
<td>Store Status Register</td>
<td>(SR) = (EA)</td>
</tr>
<tr>
<td>10110...0...</td>
<td>S</td>
<td>ADF</td>
<td>* Add - Floating Point</td>
<td>(A) + (EA) – A</td>
</tr>
<tr>
<td>10110...1...</td>
<td>S</td>
<td>ADF</td>
<td>* Add - Floating Point</td>
<td>(A) + (EA) – A</td>
</tr>
<tr>
<td>10111...0...</td>
<td>L</td>
<td>LDS</td>
<td>Load Status Register</td>
<td>(EA) = SR</td>
</tr>
<tr>
<td>10111...1...</td>
<td>L</td>
<td>LOR</td>
<td>Logical OR</td>
<td>(A) OR (EA) – A</td>
</tr>
<tr>
<td>10100...0...</td>
<td>L</td>
<td>LOR</td>
<td>Logical OR</td>
<td>(A) OR (EA) – A</td>
</tr>
<tr>
<td>10100...1...</td>
<td>L</td>
<td>EXO</td>
<td>Exclusive OR</td>
<td>(A) XOR (EA) – A</td>
</tr>
<tr>
<td>10101...1...</td>
<td>L</td>
<td>EXO</td>
<td>Exclusive OR</td>
<td>(A) XOR (EA) – A</td>
</tr>
<tr>
<td>10101...0...</td>
<td>L</td>
<td>MUL</td>
<td>Multiply - Fixed Point</td>
<td>(A) × (EA) – A, B</td>
</tr>
<tr>
<td>10101...1...</td>
<td>L</td>
<td>MUL</td>
<td>Multiply - Fixed Point</td>
<td>(A) × (EA) – A, B</td>
</tr>
<tr>
<td>10110...0...</td>
<td>S</td>
<td>SFR</td>
<td>Subtract - Floating Double Precision</td>
<td>(A, B) = (EA, EA-2) – A, B</td>
</tr>
<tr>
<td>10110...1...</td>
<td>S</td>
<td>SFR</td>
<td>Subtract - Floating Double Precision</td>
<td>(A, B) = (EA, EA-2) – A, B</td>
</tr>
<tr>
<td>11100...0...</td>
<td>S</td>
<td>SBU</td>
<td>Subtract Upper - Fixed Point</td>
<td>(A) - (EA) – Carry – A</td>
</tr>
<tr>
<td>11100...1...</td>
<td>S</td>
<td>SBU</td>
<td>Subtract Upper - Fixed Point</td>
<td>(A) - (EA) – Carry – A</td>
</tr>
<tr>
<td>11101...0...</td>
<td>S</td>
<td>SBL</td>
<td>Subtract Lower - Fixed Point</td>
<td>(B) - (EA) – B</td>
</tr>
<tr>
<td>11101...1...</td>
<td>S</td>
<td>SBL</td>
<td>Subtract Lower - Fixed Point</td>
<td>(B) - (EA) – B</td>
</tr>
<tr>
<td>11100...0...</td>
<td>S</td>
<td>DVD</td>
<td>Divide - Fixed Point</td>
<td>(A, B)/(EA) – A, Remainder – B</td>
</tr>
<tr>
<td>11100...1...</td>
<td>S</td>
<td>DVD</td>
<td>Divide - Fixed Point</td>
<td>(A, B)/(EA) – A, Remainder – B</td>
</tr>
<tr>
<td>11110...1...</td>
<td>L</td>
<td>STI</td>
<td>Store Interrupt Mask Register</td>
<td>(MR) = EA</td>
</tr>
<tr>
<td>11110...0...</td>
<td>S</td>
<td>SBF</td>
<td>Subtract - Floating Point</td>
<td>(A) – (EA) – A</td>
</tr>
<tr>
<td>11110...0...</td>
<td>S</td>
<td>SBF</td>
<td>Subtract - Floating Point</td>
<td>(A) – (EA) – A</td>
</tr>
<tr>
<td>11110...1...</td>
<td>L</td>
<td>LDV</td>
<td>Load Interrupt Mask Register</td>
<td>(EA) = MR</td>
</tr>
</tbody>
</table>
APPENDIX B

LISTINGS OF MAP GENERATOR COMPUTER PROGRAMS

1. Control Program for Map Display
2. Program to Read Input Data Tape
3. Determine Points in Field-of-View and Decompress Data
4. Rotate Data and Compute Intersection
5. Varian Printer Plotter Subroutine
6. TV kaster Output Subroutine
7. Paper Tape Read and Teletype Output Subroutine
8. Program to Format and Compress Contour Data
* CONTROL PROGRAM FOR MAP DISPLAY
*
*
*
*
******************************************************************************
INSTRUCTIONS
* SW5 -- X AIRCRAFT POSITION DRIVE
* SW4 -- Y AIRCRAFT POSITION DRIVE
* SW3 -- AIRCRAFT HEADING DRIVE
* SW2--- SCALING FRESOLUT
* SW0 -- SIGN OF INCR, ON = -1, OFF = +1
******************************************************************************
COSFA  SETX 0444 STRT LOC OF SUBR COS
SINFA  SETX 0606 STRT LOC OF SUBR SIN
RDATA  SETX 8120
FOVPTS  SETX 8100
DTRINTR SETX 8320
ST1  SETX 8000
*
ORG  ST1
FRASTER  DECR 256.000
FRESOLUT  DECR -10.
FXACPOS  DECR 1280.
FYACPOS  DECR 1150.
HEADING  DECR 0.0
SINSI  HEX 0
COSSI  HEX 7FFFFFFF
FSINSI  HEX 0
FCOSSI  HEX 0
INWORDS  HEX 0 NUMBER OF WORDS INPUT
FOVWDK  HEX 0 NO. OF WORDS IN FOV
TEMP  BSS 4
SUBROUT  BSS 8
*
RADIAN  DECR 57.3
ONE  DECR 1.0
FTWO  DECR 2.0
FFIVE  DECR 5.0
FTEN  DECR 10.0
FHUND  DECR 100.0
MINUSONE  DECR -1.0
CONVRTN  HEX 0
BIT1  HEX 80000000
BIT32  HEX 00000001
PLUSONE  HEX 7FFFFFFF
*
ST2  SETX 8040
ORG ST2
JS RDDATA GO READ DATA ONE TIME ONLY
BASE S,FRASTER
LDX S,FRASTER,M
LOOP LDX S,FRASTER,M

**

* START JGU SIGN,0. IS DECR SET
LDA ONE NO. INCREMENT

* SW3RTN STA TEMP
SW5 JGU XAC,5
SW4 JGU YAC,4
SW3 JGU HEAD,3
SW2 JGU SCALE,2
HEAD3 JU HEAD2

* RETURN JS FOVPTS GO DETERMINE POINTS IN FOV
JS DTRIMTR GO COMPUTE INTERSECTIONS WITH RASTER LINES
JU LOOP

* SIGN LDA MINUSONE
JU SW3RTN

* XAC LDA FHUNDR
MLF TEMP MULT BY SIGN
ADF FXACPOS
STA FXACPOS
JU SW4

* YAC LDA FHUNDR
MLF TEMP
ADF FYACPOS
STA FYACPOS
JU SW3

* HEAD LDA FFIVE
MLF TEMP
ADF HEADING
STA HEADING
JU SW2

* HEAD2 LDA HEADING
DVF RADIAN
LDX 6,SUBROUT+6,M
JS COSFA
STA FCOSSI
JS CONV

15
STB COSSI

* LDA HEADING
DVF RADIANG
LDX 6, SUBROUT+6, M
JS SINFA
STA FSINSI
JS CONV
STB SINSI
JU RETURN

* CONV PTR CONVRTN
CFX
SAM BIT1
SAM BIT32
JU SHIFT
LDB PLUSONE
RTA CONVRTN

SHIFT SRAD 1
RTA CONVRTN

* SCALE LDA FTUO
MLF TEMP
ADF FRENSULUT
STA FRENSULUT
JU HEAD3

END

END
* PROGRAM READS IN INPUT DATA TAPE
*
*
*
CHARSV  SETX  8630
TAPERD  SETX  8630
TWRITE  SETX  8650
*
INWORDS  SETX  8012  NUMBER OF INPUT WORDS
STORE  SETX  8670  BEGINNING LOCATION OF INPUT DATA BASE
*
BEGIN  SETX  8100
  ORG  BEGIN
DARDRTN  HEX  0
SV1  HEX  0
SV2  HEX  0
LNCKRTN  HEX  0
CKOUNT  HEX  0
LKOUNT  HEX  0
EOT  HEX  00000004  END OF TAPE
LSPACE  HEX  20000000  LEADING SPACE
LCRLF  HEX  0D0A0000  LEADING OR LF
RUBS  HEX  0000007F  RUBOUTS
ZIP  HEX  0  ZERO
*
STRTB  SETX  8120
  ORG  STRTB
RDATA  PTR  DARDRTN
*
* INITIALIZE SECTION
*
    BASE  5.SV1
    LDX  5.SV1,M
*
    LDA  ZIP
    LXA  0  XR0 IS CHAR COUNT
    LXA  2  XR2 IS INPUT WD CNT
    RST 15  RESET ALL FLAGS
    STA  SV1
    STA  SV2
    STA  LKOUNT
    STA  CKOUNT
*
* READ INPUT DATA AND WRITE IT OUT
*
GNC  JS  TAPERD
    LDA  CHARSV
17
STA SV1  
EXO RUBS IS IT RUBOUT  
JN NXT  
JU GNC  

NXT  
LDA SV1  
EXO EOT  
JN C1  
JGU START1  

C1  
JGW C4,1 PRINT IF SW 1 SET  
JU C2 OTHERWISE SKIP PRINT  

C4  
LDB SV1  
SLLD 24  
JS TWRITE  
JS LINECK  

C2  
LDA SV1  
SBU 64,M HEX 40  
JG NXT1  
JU NXT2  

NXT1  
LDA SV1  
SBU 71,M HEX 47 = G  
JG NXT2  
LDA SV1  
ADU 9,M CONV TO HEX FROM ASCII  
STA SV1  

NXT2  
LDA SV1  
AND 15,M  
LOR SV2 ADD TO REST OF UD  
IMP 0,1,M INC XR0 BY 1  
ICL 0,4,M IS XR0 < 4  
JU C3  
SLL 4  
STA SV2  
JU GNC GET NEXT CHAR  

* 16 BITS OF INPUT COMPLETED  

*  
C3  
SLL 16 SHIFT TO LEFT HALF UD  
STH STORE,2 STORE HALF UD  
IMP 2,1,M INC XR2 UD COUNT  
LDX 0,ZIP CLEAR CHAR CNT  
LDA ZIP  
STA SV2  
JU GNC GET NEXT UD  

*  
* LINE CHECKING SECTION  
*  
LINECK PTR LHCRTN  
LDA LKOUNT  
ADU 1,M  

18
STA LKOUNT
SBU 32,M
JG OTPCRLF
LDA CKOUNT CHAR COUNT
ADU 1,M     ADD 1
STA CKOUNT
SBU 8,M
JG OTPSP

LRTN  RTN LNCKRTN
OTPCRLF LDB LCRLF
     LDA ZIP
     STA LKOUNT
     STA CKOUNT
     JS TTWRITE
     JU LRTN

OTPS P LDB LSPACE
     LDA ZIP
     STA CKOUNT
     JS TTWRITE
     JU LRTN

* *

START1 IMN 2,1,M DECREMENT WD COUNT
NOP
STX 2,INWORDS SAVE WORD COUNT
LDB LCRLF
LDA ZIP
JS TTWRITE
RTA DAARDRTN

* *

END
END
DETERMINE POINTS IN FIELD OF VIEW AND DECOMPRESS DATA

CALCULATION OF AIRCRAFT PARAMS AND VARBLS

FLAG ALLOCATIONS
* FLAG 1 --- IMAG FLG
* FLAG 2 --- OUT OF RANGE FLAG
* FLAG 3 --- SAVE OUT OF RANGE DATA

REGISTER DEFINITIONS
* XR2 -- INCOMING DATA UP COUNTER
* XR3 -- STORED DATA COUNTER
* XR5 -- BASE REGISTER
* XR7 -- RTM REG

BEGIN SETX 81A0
ORG BEGIN
FRASTER SETX 8000 NUMBER OF RASTER LINES
FRESOLUT SETX 8002 RASTER RESOLUTION
FXACPOS SETX 8004 X POSITION
FYACPOS SETX 8006 Y POSITION
INWORDS SETX 8012 NUMBER OF INPUT WORDS
FOVWDK SETX 8014 NUMBER OF WORDS IN FOV
XTRACK SETX 3E00
YTRACK SETX 3E02
STORE1 SETX 8670 STARTING LOCATION OF INPUT DATA BASE
STORE2 SETX A200 STARTING LOCATION OF DATA POINTS IN FOV
FOVRTN HEX 0
FTWO DEC 2.0
SORTTWO DEC 1.4142
TEMP BSS 6
XMIN HEX 0
YMIN HEX 0
XMAX HEX 0
YMAX HEX 0
FXMIN HEX 0
FYMIN HEX 0
FXMAX HEX 0
FYMAX HEX 0
SIGN HEX 80000000
BIT9 HEX 00000000
BIT16 HEX 8000

STRTC SETX 81D0
ORG STRTC
FOVPTS PTR FOVRTN
*
* CAL MAX AND MIN X AND Y IN FLT PT
*
BASE 5,FTWO
LDX 5,FTWO,M
*
DETERMINE FOV RADIUS
LDA FRESOLUT
MLF FRASTER
DVF FTWO FRASTER*FRESOLUT/2
MLF SORTTWO
STA TEMP
*
* FIND LFT EDGE FOV
LDA FXACPOS
SBF TEMP
STA FXMIN
*
* FIND RT EDGE FOV
LDA FXACPOS
ADF TEMP
STA FXMAX
*
* FIND BOTTOM OF FOV
LDA FYACPOS
SBF TEMP
STA FYMIN
*
* FIND TOP OF FOV
LDA FYACPOS
ADF TEMP
STA FYMAX
*
* CONVERT FLT PT TO FIX PT HALF WORD
*
LDX 1,0,M RS XRO
LOOP1 LDA FXMIN,1
LDB 0,M
CFX CONV FLT TO FIXED PT
SLL 16 CONV TO HALF UD
STAH XMIN,1
IMP 1,2,M
ICL 1,8,M
JU NX1
JU LOOP1
*
INITIALIZE SECTION
NX1 LDX 2,0,M
LDX 3,0,M
LDX 7,0,M
* SET STATUS REG FOR FAST SCRATCH PAD OPER.
LDS 8192,M SET BIT 6 IN STS REG
LDA 0,M
STA XTRACK
STA YTRACK
RST 15 RESET ALL FLAGS

NX2
LDAH STORE1,2
SAM SIGN SKP IF SIGN SET
JU NCRMENT GO TO INCR FORMAT

* START WORD FORMAT
STRTWORD EXO SIGN RS START WD IND
STAH XTRACK SAVE X VALUE
IMP 2,1,M INC XR2
LDAH STORE1,2 GET Y
SAM SIGN IS IMAG BIT SET
JU NX3 NO
SET 1 YES, SET IMAG FLAG
EXO SIGN RESET IMAG BIT

NX3
STAH YTRACK SAVE Y VALUE
JU COMP1

* COMRESSED WORD FORMAT

* NCRMENT SAM BIT9 IS IMAG BIT SET
JU NX4
SET 1 SET FLG 1 -- IMAG FLG

NX4
SLL 1 SH SIGN X TO SGN A REG
SRA 9 SIGN EXTEND TO 16 BITS
ADUHR XTRACK,? ADD HF WD TO FSP

* WORK ON Y VALUE
SLL 17 SGN Y TO SGN A REG
SRA 9 SIGN EXTEND
ADUHR YTRACK,? ADD H TO FSP
JU COMP1

* CHECK FOR DATA IN FIELD OF VIEW (FOV)

COMP1
LDAH XMAX
SBUH XTRACK
JL 00R JMP IF XTRACK>XMAX
LDAH XTRACK
SBUH XMIN
JL 00R JMP IF XTRACK<XMIN
LDAH YMAX
SBUH YTRACK
JL 00R JMP IF YTRACK>YMAX
LDAH YTRACK
SBUH YMIN
JL 00R JMP IF YTRACK<YMIN
DATA IS IN FOV

TABLE
LDAH XTRACK
LDBH YTRACK
JGF NX6,1

RETN1
SRA 16
SLLD 16
JGF NX8,8 IS SPECIAL FLG SET
STA STORE2,3

RETN2
ICL 2, INWORDS IS IT LAST WORD
JU ENDC YES, END SECT
IMP 2,1,M INCR WD CNT
IMP 3,2,M INCR OUT WD CNT
JU NX2 RECYCLE

NX8
STA TEMP+2
ICL 2, INWORDS
JU ENDC
IMP 2,1,M
JU NX2

DATA HAS RETURNED TO FIELD OF VIEW

NX5
JGF NX5A,8 IS ANYTHING STORED
RST 2
JU TABLE

NX5A
LDA TEMP+2
STA STORE2,3
IMP 3,2,M
RST 8
RST 2 RS OUT OF RANGE FLG
JU TABLE

RESTORE IMAG BIT

NX3
EAB
LOR SIGN SET IMAG BIT
EAB
RST 1 RS IMAG FLG
JU RETN1

OUT OF RANGE SECTION

OOR
JGF NX7,2
SET 2 SET FLG IF NOT SET
RST 8
JU TABLE

NX7
SET 2 SET SAVE OOV DATA
SET 1 SET IMAG BIT
JU TABLE

ENDC
STX 3, FOVWDK SAVE NO. WD
RTA FOVRTN

END

23
* ROTATES DATA AND COMPUTES INTERSECTIONS

BEGIN
SETX 8290
ORG BEGIN

* DEFINITIONS

FRASTER SETX 8000 NUMBER OF RASTER LINES
FRSOLUT SETX 8002 RASTER RESOLUTION
FXPOS POS XY POSITION
FYPOS POS XY POSITION
SINSI SETX 800A
SSISI SETX 800C
FSINSI SETX 800E
FCSSI SETX 8010
FOWDK SETX 8014 NUMBER OF WORDS IN FOV
TOFORM SETX 859A TOP OF FORM ENTRY POINT
LINEFEED SETX 8590
LINEPRT SETX 8598
TVOUT SETX 7320 TV RASER OUTPUT ENTRY POINT
STORE2 SETX A200 STARTING LOCATION OF DATA POINTS IN FOV
SCANW SETX BFE0

ISECTRN HEX 0
SGNXRTN HEX 0
RNDRTN HEX 0
SLPRTN HEX 0
FONE DEC 1.0
TWO DEC 2.0
BIT16 HEX 8000
THIRTEEN DEC 13.0 HALF THE NUMBER OF RASTER LINES MISSING
IRASTN DEC 229 NUMBER OF VERTICAL RASTER LINES
SCWDIMX HEX E THIS CORRESPONDS TO 8 WORDS 256 BITS
HTEN DEC16 10
BLK1 DEC16 0
PC HEX 0
YSCANST HEX 0
YSCAN HEX 0
YSCANP HEX 0
YSCANNG HEX 0
XP HEX 0
XP2 HEX 0
YP HEX 0
YF HEX 0
SLOPE HEX 0
RECIPO HEX 0 RECIPROCAL OF SLOPE
XZERO HEX 0
XLIMIT HEX 0
XACPOS HEX 0
YACPOS HEX 0
TEMP BSS 6
*
RTHALF HEX FFFF0000
ALLONES HEX FFFFFFFF COMPLEMENT MASK
BIT2 HEX 40000000
MASK1 HEX 80000000
MSKIMAG HEX 7FFFFFFF
*
MASK HEX 80000000
HEX 40000000
HEX 20000000
HEX 10000000
HEX 8000000
HEX 40000000
HEX 20000000
HEX 10000000
HEX 8000000
HEX 40000000
HEX 20000000
HEX 10000000
HEX 8000000
HEX 40000000
HEX 20000000
HEX 10000000
HEX 8000000
HEX 40000000
HEX 20000000
HEX 10000000
HEX 8
HEX 4
HEX 2
HEX 1
ZIP HEX 0
STRTD SETX 8320
ORG STRTD
DTRINTR PTR ISECTRTN
BASE 5,FONE
LDX 5,FONE,M

* INITIALIZE
LDA ZIP
LXA 1
LXA 3
LXA 4 WORD COUNTER
RST 4
RST 1

* VERTICAL RESOLUTION
* CONVERT FLOATING RESOLUTION TO INTEGER
LDA FRESOLUT
CFX
SLL 16
STAH HTEN

* TRANSLATION OF DATA POINTS
*
* CONVERT A/C POSITION TO INTEGER
LDA FXACPOS
CFX
SLL 16
STAH XACPOS
LDA FYACPOS
CFX
SLL 16
STAH YACPOS

* COORDINATE ROTATION OF DATA POINTS
NXLP
LDAH STORE2,4 LD X PT
SUBH XACPOS SUBTRACT A/C POS
STAH XP
LDAH STORE2+1,4 LD Y PT
SAM MASK1 IS IMAG BIT SET
JU NXLP1 NO, GET NXT PT
AND MSKIMAG YES, ELIM IMAG BIT
SET 8 SET IMAG FLG

NXLP1
SUBH YACPOS SUBT A/C POS
STAH YP
LDA XP
MUL COSSI * COS
JS RNDUP
STA TEMP XCOS
LDA YP
MUL SINSI
JS RNDUP
ADU TEMP XCOS+YSIN
AND MSKIMAG ELIM SIGN BIT
STAH STORE2,4
LDA XP
MUL SINSI
JS RNDUP
STA TEMP XSIN
LDA YP
MUL COSSI YCOS
JS RNDUP
SBU TEMP YCOS-XSIN
AND MSKIMAG ELIM SIGN.BIT
JGF NXLP3,8 IF FLG 8 SET RESTORE-IMAG BIT
NXLP2
STAH STORE2+1,4
ICL 4,FOVWDK
JU P4NX
IMP 4,2,M
JU NXLP ROTATE NXT PPT
*
RNDUP
PTR RNDRTN
SAM MASK1
SAM BIT16
RTA RNDRTN
ADUH 1,M
AND RTHALF
RTA RNDRTN
*
NXLP3
RST 8
LOR MASK1
JU NXLP2
*
FIELD OF VIEW LOGIC
P4NX
LDX 4,0,M RS UD PTR
* DETERMINE FIRST SCAN LINE
LDB ZIP
LDA FRASTER NO. RASTER LINES
MLF FRESOLUT RES. FT/RAST. LINE
DVF TWO /2
STA TEMP USE IN MAX MIN CAL
LDB ZIP COMPENSATE FOR MISSING 26 LINES ON TV DISPLAY
LDA FRESOLUT
MLF THIRTEEN
STA TEMP+2
LDA TEMP
SBI TEMP+2
STA YSCANST Y VALU 1ST SCN LN
LDA FOME LD 1
STA PC

27
DETERMINATION OF XMAX & XMIN

DETERMINE LEFT SIDE LIMIT
LDA ZIP
LDB ZIP
SBF TEMP
CFX CONV FLT TO FX
SLL 16
STAH XZERO

DETERMINE RIGHT SIDE LIMIT
LDA TEMP
CFX CONV FLT TO FX
SLL 16
STAH XLIMIT

BEGIN SCAN OF DATA
P4ST
LDA PC
SBF FONE
MLF FRESOLUT
STA TEMP
LDA YSCANST GT 1ST SCN LIN
SBF TEMP
CFX CONV FLT TO FIX
SLLD 16
STAH YSCAN SV CURRENT SCN LN

DETERMINE HALF OF DELTA SCAN
LDB ZIP
LDA FRESOLUT
CFX
SLLD 16
STAH TEMP DSC/2

LDAH YSCAN
SBUH TEMP
STAH YSCANNG YSCAN-DSC/2

LDAH YSCAN
ADUH TEMP
STAH YSCANP

CHECK FOR IMAG FLAG

P4NX0 LDAH STORE2+3.4 GT YP2
SAM MASK1
P4NX1
LDAH STORE2, 4
JS SGNX
STAH XP
LDAH STORE2+1, 4
AND MSKIMAG REMOVE IMAG BIT
JS SGNX
STAH VP
LDAH STORE2+2, 4
JS SGNX
STAH XP2
LDAH STORE2+3, 4
JS SGNX
STAH VP2

* DETERMINE INTERSECTON
PNX0
LDAH VP IS VP LT YSCAN +
SBUH YSCANP
JL PNX3 YES

* PNX1
LDAH VP2 IS VP2 LT YSCAN
SBUH YSCAN
JL PNX2 YES
JU P4NX5 NO, RETURN

* PNX2
LDAH VP2 IS VP2 LT YSCAN -
SBUH YSCANMG
JL COMPUTR COMPUTE INTERSECTION

* SET 1
LDAH XP2
JU P4NX4A

* PNX3
LDAH VP IS VP LT YSCAN
SBUH YSCAN
JL PNX6 YES

* PNX4
LDAH VP2 IS VP2 LT YSCAN +
SBUH YSCANP
JL PNX5 YES
JU P4NX5 NO, RETURN

* PNX5
LDAH VP2 IS VP2 LT YSCAN -
SBUH YSCANMG
JL PNX9 GO DO DOUBLE FILL 6
LDAH VP2
SBUH YSCAN
JL FILL
JU P4NX5 NO, RETURN

* FNX6 LDAH VP IS VP LT YSCAN -
       SBUH YSCANNG
       JL PNX10 YES

* PNX7 LDAH VP2 IS VP2 LT YSCAN -
       SBUH YSCANNG
       JL P4NX5 NO, RETURN

* PNX8 LDAH VP2 IS VP2 LT YSCAN +
       SBUH YSCANP
       JL FILL GO FILL

PNX9 SET 1 SET DOUBLE FILL FLAG
LDAH XP X0 = XP
JU P4NX4A

* PNX10 LDAH VP2 IS VP2 LT YSCAN
       SBUH YSCAN
       JL P4NX5 YES, RETURN

* PNX11 LDAH VP2 IS VP2 LT YSCAN +
       SBUH YSCANP
       JL P4NX12
       JU COMPINTR COMPUTE INTERSECTION

P4NX12 SET 1
LDAH XP2

* P4NX4A STA1 X0
JS COMPSLP
JU SLOPECHECK

* * INCREMENT WORD POINTER

* P4NX5 JGF DBLFILL,1 IS DOUBLE FILL FLAG SET
IMP 4.2,M
RST 4
ICL 4,F0VWDK
JU OUTPUT
JU P4NX0

* DBLFILL RST 1 RESET DOUBLEFILL FLAG
JU COMPINTR COMPUTE INTERSECTION

* RESTORE NEGATIVE SIGN
SGNX PTR SGNXRTN

30
SHA H  
RTH SGNXRTN
LOR NASH1
RTH SGNXRTN

* DETERMINATION OF X POSITION IN SCAN LINE
* X VALUE ASSUMED TO BE IN A REG
* XR8 IS WORD COUNT
* XR1 IS BIT NUMBER
* FILL FLAG = FLAG 4

* DETERMINE WORD NO. & BIT NO.

DTRXPO
	STAHL  TEMP
LDX 8.0M
LDX 1.0M
SBIEH XZERO SUB LMT DGE LMT
JL P4NX10A 00FOV
LDB ZIP
SRA 15
DVD HTEN
JS RNDUP

P4NX10
SBIEH 32M
JL P4NX11 J LT 0
IOL 8.SQDMX
JU P4NX5 00FOV
IMP 8.2M = 0
JU P4NX10

* LINE ONLY PARTIALLY OUT OF VIEW
P4NX10A JGF P4NX10B.4 IS FILL FLG SET
JU P4NX5 NO RETURN

P4NX10B SRA 15 SCALE FOR DIVISION
LDB ZIP
DVD HTEN
JS RNDUP
SRA 16 SHIFT TO LOAD XR1

P4NX10C JN P4NX10D JMP IF NEG
LXA 1 LD XR1
JU P4NX13 GO FILL

P4NX10D ADU 1.M INC A REG
IMN 2.1.M DECR FILL COUNT
JU P4NX10C IF NOT ZERO GO HERE
RST 4
JU P4NX5 RETURN

*  

P4NX11 ADUH 32.M
SRA 15 SHFT & MLY X 2
LXA 1

* PLACE BIT IN SCAN WORD *

P4NX13 LDA SCANWD,8 GET SCAN WD
LOR MASK,1 ADD MASK
STA SCANWD,8 RTN WORD

* JGF P4NX14,4 IS FILL FLG SET
JU P4NX5 NO, RETURN
P4NX14 RST 4 RESET FILL FLAG
P4NX14A IMN 2,1,M DECR FILL COUNT
JU P4NX15
JU P4NX5 RETURN
P4NX15 IMP 1,2,M
ICL 1.64,M IS IT 64
JU P4NX16 YES
LOR MASK,1
STA SCANWD,8
JU P4NX14A

* P4NX16 ICL 8,SCUDMX IS IT END OF SCAN WDS
JU P4NX5 YES, RETURN
IMP 8,2,M NO, INCR WD PTR
LDX 1,0,M
LDA SCANWD,8
LOR MASK,1
STA SCANWD,8
JU P4NX14A

* COMPUTE SLOPE *

COMPSLP PTR SLPTRN
LDAH XP2
SBU XP
JN COMPN1
LDAH XP
JU DTRXP0

COMPN1 STA TEMP
LDAH YP2
SBU YP
SRA 15
LDB ZIP
DVD TEMP
STA SLOPE
RTA SLPRTRN
* COMPUTE INTERSECTION
*
COMPINTR JS COMPSLP
LDH  YSCAN
SBU  YP
SRA  15
LDB  ZP
DVD  SLOPE
JS  RNDUP
ADU  XP
STAH  X0
*
* FILL MODE FOR SHALLOW ANGLES REL. TO RASTER
SLOPECHEK LDA 2,M LD ONE
LDB  ZP
DVD  SLOPE COMPUTE
JL  COMPLEMENT
CKSL JS  RNDUP
SRA  16
CKSL0 STA  RECIPRO
LXA  2
ICL  2,2,M
JN  CKSL1 RETURN IF 0
JU  RTNCK NO, RETURN
CKSL1 LDA  TEMP  XP2-XP
JG  CKSL2
LDA  XP
JU  CKSL3
CKSL2 LDA  XP2
CKSL3 SBU  X0
JL  RTNCK RETURN IF NEG
JN  CKSL4 RETURN IF 0
JU  RTNCK
CKSL4 SRA  15
LDB  ZP
DVD  HTEN
SRA  16
STA  TEMP+4
LDA  RECIPRO
LXA  2
SET 4
ICL  2,TEMP+4
LDX  2,TEMP+4
ICL  2,2,M
IMN  2,1,M
JU  RTNCK  NOT 0 OR LESS
RST  4
RTNCK LDA  X0
JU DTRXPO

* REVERSE COMPLEMENT
COMPLEMENT SBU 1,M
EXO ALLONES
JU CKSL

* FILL COMPUTATION
* FILL SET 4 SET FILL FLAG
LDAH XP
SBUH XP2
SRA 15
LDB 2IP
Dvd HTEN
JS RNDUP
JG FINLNX
LDAH XP2
SBUH XP
SRA 15
LDB 2IP
Dvd HTEN
JS RNDUP
SRA 16
LXA 2
LDAH XP
JU DTRXPO
FINLNX SRA 16
LXA 2
LDAH XP2
JU DTRXPO

* OUTPUT SECTION
* OUTPUT JS TVOUT
JGU 0L0,1 IF SWITCH 1 SET OUTPUT TO LINEPRINTER ALSO
* CLEAR OUTPUT BUFFER
CLRBUFF LDA ZIP CLEAR A REG
LXA 4
CLR STA SCANUD,4
IMP 4,2,M
ICL 4,16,M
JGU CONTINUE
JGU CLR
CONTINUE LDA PC
ADF FONE
STA PC
ICL 3,IRASTN
JGU OL2
IMP 3,1,M
LDX 4,0,M
JGU P4ST

* OUTPUT TO LINEPRINTER
OL1 JS LNEPPT
JGU CLRBUFF GO CLEAR OUTPUT BUFFER

* OL2 JGW OL3,1 IF LINEPRINT SW SET DO TOP OF FORM
RTA ISECTRTH

* OL3 JS TOFORM
RTA ISECTRTH

* END
END

>
**VARIAN PRINTER PLOTTER SUBROUTINE**

* *
* *
* *
DATAOUT  SETX  BFDA  RASTER LINE DATA OUTPUT—CONTAINS 3 BLANK UDS
START  SETX  8570
ORG  START
RASTMODE  HEX  0BE0
REMOENAB  HEX  0B20
TOPOFORM  HEX  0BB3
SYNCSTEP  HEX  0B23
*
XR2SV  HEX  0
XR3SV  HEX  0
XR4SV  HEX  0
WDCOUNT  HEX  16
TEMP  HEX  0
*
OUTCMND  HEX  00000A00
*
OPRTN  HEX  0
WAITRTN  HEX  0
ZIP  HEX  0
*
VPPRTN  HEX  0
TOFRTN  HEX  0
LFRTN  HEX  0
*
LF  PTR  LFRTN
JS  ONE
LDA  SYNCSTEP
DOA  22.C,K
RTA  LFRTN
*
*
TFORM  PTR  TOFTTN
LDA  REMOENAB
DOA  22.C,K
JS  ONE
LDA  TOPOFORM
DOA  22.C,K
RTA  TOFRTN
*
*
PTR  VPPRTN
LDA  ZIP
STA  DATAOUT  CLEAR LEFT BORDER OF LINEPRINTER
STA  DATAOUT+2
STA DATAOUT+4
STX 2,XR2SV SAVE XR2
STX 3,XR3SV SAVE REG 3
STX 4,XR4SV SAVE REG 4

* BEGIN
LDA REMOENAB
DOA 22,C,K
JS ONE
LDA RASTMODE
DOA 22,C,K
RST 15

LDX 1,2,M NUMBER OF VERTICAL BITS PER POINT, -1

LP5
LDX 4,2IP WORD COUNTER

LP2
LDX 2,7,M BIT COUNTER

LDA ZIP
LDBH DATAOUT, 4
SRLD 16 SHIFT DATA TO RT HF B REG
SRCB 1 SHIFT BIT B31 TO BIT A0
SRA 2 SIGN EXTEND A REG 1 BIT
IMN 2,1,M
JU LOOP
JGF LP1,1

STA TEMP
LDA ZIP
LDX 2,7,M
SET 1
JU LOOP

* SET UP TO OUTPUT FORMATTED DATA

LP1
EAB
JS OUT
LDB TEMP
JS OUT
RST 1

* INCREMENT WORD POINTER
IMB 4,1,M
ICO 4,WDCOUNT IS LAST WORD
JU LP3 YES
JU LP2 NO

* REPEAT LINE
LP3
JS LF
IMN 1,1,M DECREMENT VERTICAL bit COUNT
JU LP4

LP4
RST 15

* RESTORE INDEX REGISTERS
LDX 2,XR2SV RESTORE XR2
LDX 3,XR3SV RESTORE REG 3
LDX 4,XR4SV RESTORE

*
* WAIT ROUTINE
ONE
TWO

* OUT
LP8

RTA  VPPRTN
*  
PTR  WAIKRTN
DIA  22,K
SAM  14,M
RTA  WAIKRTN
JU   TWO

PTR  OPRTH
LDX  3,2,M  BITE COUNTER
JS   ONE
LDA  ZIP
SLLD 8
LOR  OUTCMND
DOA  22,C,K  OUTPUT DATA
IMN  3,1,M
JU   LP8
RTA  OPRTH

*  
END

>
TV RASTER OUTPUT ROUTINE

THIS PROGRAM OUTPUTS TO A TV SCREEN

START SETX 8570
TVOUT SETX 85A8
DATAOUT SETX BFE0 OUTPUT DATA BUFFER

ORG START
TVDATA HEX FF705FF0 FIRST 10 BITS ARE COMPL OF NO. OF UDS
RASTRTN HEX 0 LAST 16 BITS ARE START LOC OF DATA SHFTD 1 BIT
ZIP HEX 0 TO THE RIGHT (BFE0)
XR4SV HEX 0

ORG TVOUT
PTR RASTBTN
STX 4,XR4SV SAVE INDEX REGISTER
LDA TVDATA
DOA 19.K
EMI
RCK DIA 19.K CHECK WORD COUNT ZERO
SAM 1.M
JU RCK
LDA ZIP CLEAR BUFFER
LXA 4
CLR STA DATAOUT,4
IMP 4.2,M
ICL 4.16,M
JU CONTINUE
JU CLR
CONTINUE LDX 4,XR4SV
RTA RASTBTN

END
END
* PAPER TAPE READ AND TELETYPING OUTPUT ROUTINE
*
*
BEGIN SETX 8630
ORG BEGIN
CHARSV HEX 0
TTURRTN HEX 0
MAKEN HEX 0
ZIP HEX 0
ENHSE HEX 02DA15ED
*
* PAPER TAPE READ ROUTINE
* PROGRAM READS 8 BIT ASCII CODE INTO LOC. CHARSV
* PROGRAM SKIPS BLANKS
TAPERead PTR TRRTN
TAPE0 LDA ENHSR ENABLE READER
DOA 16, C, K OUTPUT ENABLE
TAPE1 DIA 16, C, K READ STATUS
SAM 16384, M IS TAPE READER RDY
JU TAPE2 YES. GO READ CHAR
JU TAPE1 NO. CK AGAIN
TAPE2 DIA 16, K INPUT CHAR
AND 127, M MASK PARITY
STA CHARSV SAVE CHARACTER
JN TAPE3 IS CHAR A BLANK (00)
JU TAPE0 YES. READ AGN
TAPE3 RTA TRRTN RETURN FROM SUBROUTINE
*
* TELETYPING OUTPUT ROUTINE
* DATA ASSUMED TO BE IN B REG. LEFT JUSTIFIED
*
TTYWRITE PTR TTURRTN
TTYO DIA 16, C, K READ STATUS
SAM 32768, M IS TTY BUSY
JU TTYO1 NO. OUTPUT CHAR
JU TTYO YES. CK AGN
TTYO1 LDA ZIP CLEAR A REG.
SLLD 8 SHIFT 8 BITS TO A REG.
JN TTYO3 OUTPUT IF NOT ZERO
RTA TTURRTN GO TO RTN IF ZERO
TTYO3 LOR MASK SET TTY OUTPUT BIT
DOA 16, K OUTPUT CHAR.
JU TTYO GO OUTPUT NXT CHAR
END
END
* PROGRAM TO FORMAT AND COMPRESS CONTOUR DATA
*
* THIS PROGRAM ACCEPTS DATA IN THE FOLLOWING FORM:
** XXXX/YYYY, . . . , XXXX/YYYY, EOF
* THE * INDICATES HIDDEN LINES
* EOT IS THE END OF TAPE CHARACTER -- 04
* EOF IS AN END OF FILE CARD -- MULTIPUNCH 6 &
*
* FAST SCRATCH PAD MEMORY ASSIGNMENTS
SUM   SETX 3E00
SV1   SETX 3E02
*
I14TRP SETX 7FBC INTERRUPT 14 TRAP LOCATION
I14RTN SETX 7FFC INTERRUPT 14 TRAP RETURN
BEGIN SETX 9000
ORG BEGIN

INSTR1  JGU PUNTRP PUNCH TRAP WAIT LOOP
ENABLP  HEX 02BA616D ENABLE PUNCH
DISIO   HEX 02DA616D DISABLE ALL DEVICES
TEMP    BSS 8
INT14   HEX 20000000
RUBOUTS HEX 7F7F7F7F
EOT     HEX 00000004
EOF     HEX 0000003E END OF FILE CHARACTER
ASTER   HEX 0000002A
SPACE   HEX 00000020
COMMA   HEX 0000002C
SLASH   HEX 0000002F
SIGN    HEX 0000000E
LMINUS  HEX 2D000000
LSPACE  HEX 20000000
MASKZ   HEX 7FFFFFFF
MASK30  HEX 00000300
MASK40  HEX 00000400
DRANGE  HEX 000003FF
SCFACT  HEX 00000001
ONE     HEX 00000001
TEN     HEX 0000000A
HUND    HEX 00000064
THOU    HEX 0000003E8
TENTHOU HEX 00002710
*
DATAWD  HEX 0
XVALUE  HEX 0
YVALUE  HEX 0

41
**XBI NARY HEX 0**
**YBINARY HEX 0**
**XTOTAL HEX 0**
**YTOTAL HEX 0**
**XDIFF HEX 0**
**YDIFF HEX 0**
**MAXDEL HEX 0**
**COMPDATA HEX 0**
**ASCRTN HEX 0**
**MESSORTN HEX 0**
**KOUNT HEX 0**
**DUMMY HEX 0**
**RTN1 HEX 0**
**RTN2 HEX 0**
**PUNRTN HEX 0**

* JGW *+4,0 IS SW 0 SET
  JS MESSO NO TYPE MESSAGE

* PUNCH LEADER
* LEADER LDX 1.25,M YES
  LDB RUBOUTS
  JS PUNCHR PUNCH LEADER
  IMN 1.1,M
  ICL 1.1,M
  JGU LEADER+2

* INITIALIZE SECTION
*
INIT LDX 1.0,M
LDX 2.0,M
LDX 3.0,M
LDX 4.0,M
LDX 5.0,M
LDX 6.0,M
LDX 7.0,M
RST 15 RESET ALL FLAGS
SET 1 SET FLAG 1
LDA 0,M CLEAR A REGISTER
STA DWTHWD

* READ CARD
*
RC LDX 8.0,M CLEAR BUFFER POINTER
JS RDPRTN READ A CARD AND LIST ON LINEPRINTER
* CHECK FOR END OF FILE CARD
LDA 0,M
LDBH  CARDBUFF  LOAD FIRST CHARACTER ON CARD
SLLD  8     SHIFT CHARACTER TO A REGISTER
LXA  9     LOAD INDEX REGISTER FOR CHECK
ION 9.EOF IS IT AN END OF FILE
JGU  PNCHEOF  GO PUNCH AN EOF ON PAPER TAPE

* BEGIN CHECK OF CHARACTERS
CK0 LDA 0,M  CLEAR A REGISTER
LDBH  CARDBUFF,B  GET CHARACTER
SLLD 8  SHIFT CHARACTER TO A REGISTER
STA TEMP  SAVE CHARACTER
LXA 9  TRANSFER TO REGISTER FOR CHECKING
ION 9,SPACE  IS IT A SPACE
JGU  INCR NO, GO INCREMENT BUFFER POINTER

* CHECK IF IT IS A NUMBER
ICL 9,58.M  IS IT LESS THAN 58
JGU  CK1 NO, NOT A NUMBER
ICL 9,48.M  YES, IS IT LESS THAN 48
JGU  PACK NO, IT IS A NUMBER - GO PACK

CK1 ION 9,SLASH  IS IT A SLASH
JGU XEND YES, TERMINATE X FIELD
ION 9,COMMA  IS IT A COMMA
JGU YEND YES, TERMINATE Y FIELD
ION 9,ASTER  IS IT AN ASTERISK
JGU HFLAG  YES, GO SET HIDDEN LINE FLAG
JGU ERMES1 NO, THEN IT IS AN INVALID CHARACTER
HFLAG  SET 2  YES, SET HIDDEN LINE FLAG
JGU INCR  GO INCREMENT BUFFER POINTER

* INCREMENT BUFFER POINTER
INCR IMP 8,1,M  INCREMENT BUFFER POINTER
ICL 8,80.M  IS IT END OF CARD
JGU INCR1 YES, GO TERMINATE LINEPRINTER LINE
JGU CK0 NO, GET NEXT CHARACTER
INCR1

LDB LEFTCR  TERMINATE LINEPRINTER LINE
JS  LPMESG OUTPUT CR
JGU RC  GET NEXT CARD

* PNCHEOF
LDB  EOT  LOAD END OF TAPE CHARACTER FOR OUTPUT TO TAPE
SLLD 24  POSITION CHARACTER FOR OUTPUT
JS  PUNCHR  GO PUNCH EOT CHARACTER
HLT
JGU LEADER  GO BEGIN AGAIN

* NUMBER PACK ROUTINE

PACK LDA TEMP
AND 15,M
STA TEMP
LDA DATAUD
SLL 4
LOR TEMP
STA DATAUD
IMP 2.1,M
ICL 2.5,M
JGU ERMESS2 INPUT DATA OUT OF RANGE
JGU INCR GO INCREMENT BUFFER POINTER

* TERMINATE FIELD NO. 1 *
 *
YEND LDA DATAUD
STA XVALUE
LDA 0,M
STA DATAUD
LDX 2.0,M
JGU INCR GO INCREMENT BUFFER POINTER

* TERMINATE FIELD NO. 2 *
 *
YEND LDA DATAUD
STA YVALUE
LDA 0,M
STA DATAUD
LDX 2.0,M
LDA XVALUE
JS DBC
STA XBINARY
LDA YVALUE
JS DBC DECIMAL TO BINARY CONV
STA YBINARY
JGU WDCOMR

* DECIMAL TO BINARY CONVERSION ROUTINE *
 *
DBC PTR RTN1
STA SV1
LDS 8192,M SET SR6
LDA 0,M
STA SUM
LDB 0,M
LDX 5.0,M SET UP PTR
LDA SV1
SHCYL AND 15,M MASK
MUL ONE,5 MUL BY UNITS
SRAD 1
EAB
ADUR SUM,.7 ADD TO FAST SCR PAD
IMP 5,2>M INC BY 2
INC 5,10>M HAS IT BEEN 5 CHAR
JGU SREST YES
LDA SV1 NO
SRA 4
STA SV1
INC 5,8>M HAVE 4 CHAR BEEN PROCESSED
JGU ++4
JGU SHCYL
LXA 0
ICL 0,7>M XR0 < 7
JGU ++4
JGU SHCYL
JGU ERMESS CONV DATA OUT OF RANGE
SREST
LDA SUM
RTA RTN1
*
* WORD COMPRESSION ROUTINE *
*
UDCOMR JGF NX2,1
JGU NX1
NX2 LDA XBINARY
STA XTOTAL
LDA DRANGE
MUL SCFACT
EAB
SRA 1
STA MAXDEL
LDA YBINAY
STA YTOTAL
RST 1 RSET INITIAL FLG
JGU STRTUD
NX1 LDA XBINARY CK MAG X
SBU XTOTAL FIND DELTA X
STA XDIFF SAVE DIF
JAG ++8 JMP IF DELX + OR 0
ADU MAXDEL ADD MAX DELTA
JAL STRTUD JMP IF -
JGU ++8
SBU MAXDEL
JAG STRTUD
LDA YBINAY
SBU YTOTAL
STA YDIFF
JAG ++8
ADU MAXDEL
JAL STRTUD
JGU ++8 USE DELTA FORMAT

45
SBU MAXDEL
JAG STRTWD

* DELTA WORD FORMATTING ROUTINE *

LDA XBINARY
STA XTOTAL
LDA YBINARY
STA YTOTAL
IMP 1.1,M
LDA 0,M
STA COMPDATA
LDA XDIFF
AND 127,M MASK ANY GARBAGE
LOR COMPDATA
SLL 8
STA COMPDATA
LDA YDIFF
AND 127,M
JGF NX3,2 IS IMAG FLG SET
JGU *+4

NX3
LOR 128,M
LOR COMPDATA
SLL 16
EAB
RST 2 RESET IMAG FLG
JS ASCNUNM
LDB LSPACE LOAD SPACE
JS LPHESG OUTPUT TO LINEPRINTER
JGU CONTINUE

* *
*  OUTPUT TO PAPER TAPE AND *
*  OUTPUT TO TTY IN ASCII NUMBERS *
*  ASSUMES DATA IN B REGISTER *
*  16 BIT OR 4 CHAR OUTPUT IN HEX *
*
ASCNUNM PTR ASCRTN
LDX 5,0,M
LDA 0,M
STA TEMP+2 CLR
ASCYL SLLD 4 SHIFT CHAR TO A
STA TEMP SV CHAR
SBU 9,M SUBTRACT 9
JAL THIRTY JMP IF < 9
JAN FORTY JMP IF NOT 0
JGU THIRTY
FORTY LOR MASK40 ASCII LETTER CODE

46
THIRTY

JGU *+6
LDA TEMP RECOVER CHAR
LOR MASK30 ASCII NUMBER CODE
STA TEMP SV CHAR CODE
LDA TEMP+2 LOAD CURRENT UD
SLL 8 MK RM FO NX CHAR
LOR TEMP ADD NX CHAR
STA TEMP+2 SV CURRENT UD
LDA 0,M
IMP 5,1,M
ICL 5,4,M
JGU **+4
JGU ASCYLN
LDB TEMP+2
JS LPMESG OUTPUT TO LINEPRINTER
LDB TEMP+2
JS PUNCHR
RTA ASCTRTN

* START WORD FORMATTING ROUTINE *

STRUD

LDA XBINARY
STA XTOTAL
SLL 16
LOR SIGN SET SIGN BIT
IMP 1,1,M INCR WORD COUNT
EAB
JS ASCNUM OUTPUT TO TTY
LDB LMINUS LOAD DASH
JS LPMESG OUTPUT TO LINEPRINTER
LDA VBINARY
STA VTOTAL
SLL 16
JGF PL6,2 CHK IMAG FLG
AND MASK2
JGU **+4

PL6

LOR SIGN
RST 2
IMP 1,1,M
EAB
JS ASCNUM OUTPUT Y TO TTY
LDB LSPACE LOAD SPACE
JS LPMESG OUTPUT TO LINEPRINTER
JGU CONTINUE

* CONTINUE LDA 0,M CLEAR A REGISTER
STA DATAWD
JGU INCR GO INCREMENT BUFFER POINTER
ERROR MESSAGES

ERMES1
LDA 5,M
STA KOUNT
LDB TEMP
JS LPMESG OUTPUT TO LINEPRINTER
LAE ERRM1
STA DUMMY
JS MESSOUT
JGU INCR GO INCREMENT BUFFER POINTER

ERMES2
LDA 8,M
STA KOUNT
LAE ERRM2
STA DUMMY
JS MESSOUT
JGU INCR GO INCREMENT BUFFER POINTER

ERMES3
LDA 7,M
STA KOUNT
LAE ERRM3
STA DUMMY
JS MESSOUT
JGU CONTINUE

MESSAGE OUTPUT ROUTINE

MESSOUT
PTR MESSORTN
LDX 4,0,M
LDB DUMMY,I
JS LPMESG OUTPUT TO LINEPRINTER
LAE DUMMY,I
ADU 2,M
STA DUMMY
IMP 4,1,M
ICL 4,KOUNT
RTA MESSORTN
JGU MESSOUT+4

MESSO
PTR RTN2
LDA 15,M
STA KOUNT
LAE MESS1
STA DUMMY
JS MESSOUT
RTA RTN2
* PROGRAM TO READ CARDS AND LIST ON LINEPRINTER
* SWITCH 3 INHIBITS LINEPRINTER
* RRRTHN HEX 0
*
RDRNRT PTR RRRTHN
NXTCD JS READCD GO READ A CARD
LDX 1,0,M INITIALIZE BUFFER INDEX
PRNTCD LDBH CARDBUFF,1 GET STORED CHARACTER
JS LPME SG OUTPUT CHARACTER
IMP 1,1,M INCREMENT BUFFER INDEX
ICL 1,80,M TEST FOR END OF CARD
JGU TERMLN END, TERMINATE LINE ON LINEPRINTER WITH CR
JGU PRNTCD NOT END, PRINT NEXT CHARACTER
TERMLN LDB LEFT CR LOAD B REG WITH CODE FOR CARRIAGE RETURN
JS LPME SG OUTPUT CARRIAGE RETURN TO LINEPRINTER
RTA RRRTHN RETURN
*
*
* SUBROUTINE READCD
* READ ONE CARD INTO CARDBUFF
* DATA IS STORED IN HALFWORDS LEFT JUSTIFIED
*
LEFTCR HEX 00000000
CARDRTN HEX 0
CARDBUFF BSS 80
*
READCD PTR CARDRTN
LDX 1,0,M INITIALIZE BUFFER INDEX
CDUT DIA 23 GET CARDREADER STATUS
SAM 256,M CHECK OR READY
JGU CDUT WAIT FOR READY
CDLOOP DIA 23 READ CARDREADER STATUS
SAM 4096,M CHECK FOR CYCLE FINISHED
JGU CDU CYCLE COMPLETE, CONTINUE
JGU CDLOOP NOT FINISHED YET, CHECK AGAIN
*
CDU DOA 23,C PICK A CARD
CD2 DIA 23 READ STATUS
SAM 512,M CHECK IF DATA RDY
JGU CD2 NOT READY, CK AGAIN
AND 127,M RDY, KEEP 7 DATA BITS
SLL 24 LEFT JUSTIFY
STAH CARDBUFF,1 STORE CHARACTER
IMP 1,1,M INCREMENT BUFFER INDEX
ICL 1,80,M TEST FOR END OF CARD
RTA CARDRTN END, EXIT
JGU CD2 NOT END, GET NEXT CHARACTER

*  *
* SUBROUTINE LPMESG CALL JS LPMESG
* DATA TO BE PRINTED IN B REG LEFT JUSTIFIED

LPRTN HEX 0

LPMESG PTR LPRTN

JGU LP3,3 SKIP LINE PRINTER IF SW 3 SET

* SEND REMOTE ENABLE
LDA 2848,M
DOA 22,C,K

* CHECK FOR RDY
DIA 22 READ STATUS
SRC 2
JL *-2 NOT RDY, WAIT

* SEND CHAR MODE SELECT
LDA 2880,M
DOA 22,C,K

* CHECK FOR NOT BSY
DIA 22
SRC 3
JL *-2 BSY, WAIT

* PUT ASCII CHAR OUT TO LP
LP2 LDA 0,M
SLLD 8 BRING NEXT CHAR TO A
JN LP1 NOT ZERO, PROCEED
JU BOF ZERO DATA, CK BOF AND RETURN

LP1 LOR 2048,M SET SINGLE CHAR MODE
DOA 22,C,K OUTPUT CHAR

* CHECK FOR RDY, BSY, PC BSY
DIA 22
SAM 14,M CK 3 BITS
JGU LP2 OK, GET NEXT CHAR
JGU *-5 NOT OK, WAIT

* BOF
DIA 22 CHECK FOR BOTTOM OF FORM
SRC 5
JRG LP3 NO BOF, RETURN

* BOF FOUND - ISSUE TOF
LDA 2995,M
DOA 22,C,K

* WAIT FOR PC NOT BSY
DIA 22
SRC 4
JL *-2 BSY, WAIT

LP3 RTA LPRTN RETURN

50
* PUNCH ROUTINE (ASSUMES DATA IN B REGISTER)
*
PUNCHR PTR PUNRTN
   LDA INSTR1 SET UP TRAP
   STA II4TRP JMP FOR INT14
   LDA ENABLP ENABLE PUNCH
   DOA 16,C,K ISSUE COMMAND
   LDA 0,M
   PCYL SLLD 8
   JN PUN JMP IF NOT 0
   LDA DISIO DISABLE ALL DEVICES
   DOA 16,C,K ISSUE COMMAND
   DPI DISABLE PROGRAM INT
   RTA PUNRTN
*
PUN
   DOA 16,K OUTPUT CHAR
   LDA INT14
   STA TEMP
   LDI INT14
   EPI
*
WAIT FOR FINISH LOOP
WAIT3
   LDA TEMP IS FLG SET
   JN WAIT3
   JU PCYL
*
TRAP LOOP FOR PUNCH
PUNTRP
   LDA 0,M
   STA TEMP
   LDA ENABLP RS PUN INT
   DOA 16,C,K
   RTA II4RTN
*

* ERRM1
   HEX 202D2D2D SP---
   HEX 494E5641 INV A
   HEX 4C949420 LID_
   HEX 43484152 CHAR_
   HEX 0D0A2020 CRLF__
*
* ERRM2
   HEX 0D0A2020 CRLF__
   HEX 494E5055 INPU
   HEX 54204441 T_DA
   HEX 54412020 TA_
   HEX 4F555420 OUT_
   HEX 4F462052 0F_R
   HEX 414E4745 ANGE
   HEX 0D0A2020 CRLF__
ERRM3
HEX 0D0A2020 CRLF__
HEX 434F4E56 CONV
HEX 20444154 _DAT
HEX 41204F55 A_OW
HEX 54204F46 T_OF
HEX 2052414E _RAN
HEX 47450D0A GE_CRLF

* MESS1
HEX 0D0A2020 CRLF__
HEX 44415441 DATA
HEX 20434F4D _COM
HEX 50524553 PRES
HEX 53494F4E SION
HEX 2050524F _PRO
HEX 4752414D GRAM
HEX 0D0A2020 CRLF__
HEX 53574954 SWIT
HEX 43482023 CH_*
HEX 30205354 0_ST
HEX 4F505320 OPS_
HEX 5052494E PRIN
HEX 54204F55 T_OW
HEX 54200D0A T_CRLF

* END
>
END