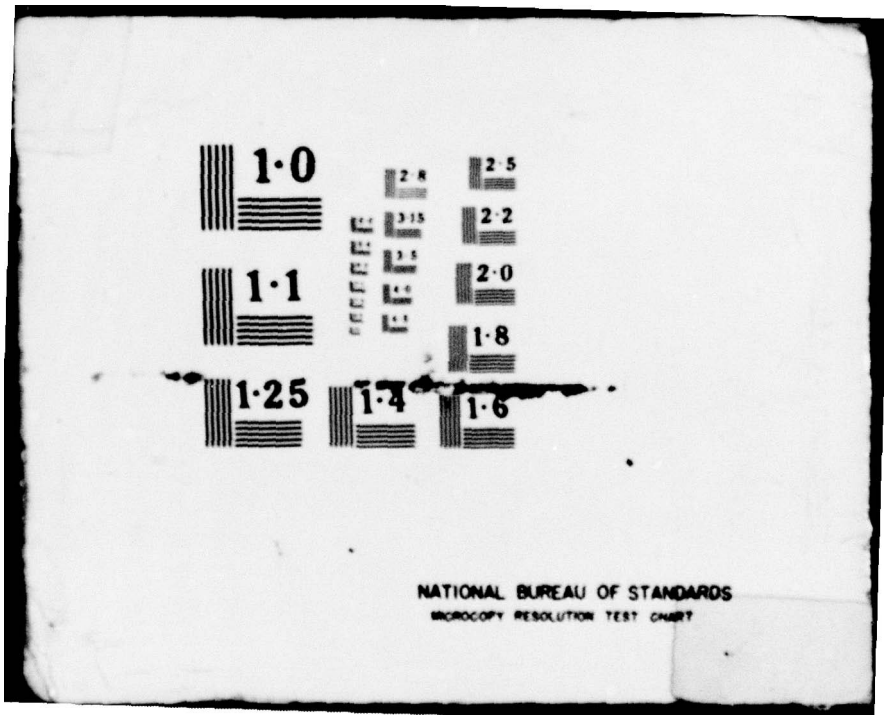
AD-A075 542 AUTOMETRIC INC FALLS CHURCH VA F/G 9/3
FEASIBILITY STUDY FOR FIELD GENERATION OF INPUT FOR RADAR SCENE--ETC(U)
NOV 78 DAAK70-78-C-0208
UNCLASSIFIED FTR-900-0005 ETL-0203 NL

| OF |
AD-
A075642

END
DATE
FILMED
11-79
DDC


1·0
1·1
1·25
1·4
1·6
1·8
2·0
2·2
2·5
NATIONAL BUREAU OF STANDARDS
MICROCOPY RESOLUTION TEST CHART

FTR 900-0006
24 November 1978

ETL-0203 LEVEL II (2)

FEASIBILITY STUDY FOR FIELD GENERATION OF INPUT FOR RADAR SCENE GENERATION FROM DLMS TERRAIN AND ELEVATION DATA

Contract number: DAAK70-78-C-0208
Contract value: $6,887.00

Presented To:

U.S. Army Engineer Topographic Laboratories
Ft. Belvoir, Virginia

Presented By:

Autometric, Inc.
5205 Leesburg Pike
Suite 1308/Skyline 1
Falls Church, Virginia 22041

Destroy this report when no longer needed.
Do not return it to the originator.

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 in this report of trade names of commercially available products does not constitute official endorsement or approval of the use of such products.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: ETL-0203	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): FEASIBILITY STUDY FOR FIELD GENERATION OF INPUT FOR RADAR SCENE GENERATION FROM DLMS TERRAIN AND ELEVATION DATA.	5. TYPE OF REPORT & PERIOD COVERED: Contract Report
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s)	8. CONTRACT OR GRANT NUMBER(s): DAAK70-78-C-0208
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Autometric, Inc. 5205 Leesburg Pike, Suite 1308 Falls Church, Virginia 22041	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 12 25
11. CONTROLLING OFFICE NAME AND ADDRESS: U.S. Army Engineer Topographic Laboratories Fort Belvoir, Virginia 22060	12. REPORT DATE: November 1978
	13. NUMBER OF PAGES: 20
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): FTR-900-0105	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

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

Radar Scene Generation
Algorithm
Disc Storage
Disc Transfer
Planimetry

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the feasibility of performing the conversion of data from a field deployable data base to input data for a radar scene generation algorithm in the field within a less than 10-minute period. Although the method described is not the only method, it is one which will work within the time allotted and fulfills other criteria, such as amount of disc storage required and computer power available.

393 362

SCB

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

FORWARD

The material presented herein represents a low budget attempt to determine the feasibility of generating input to the radar scene generation software in the field. No claims are made as to completeness or accuracy of detail, however, it is believed that the figures as stated herein are accurate enough to insure that the results will be dependable.

The program manager for Autometric was Dr. Clifford W. Greve. The Contracting Officers Technical Representative for ETL was Cpt. Ron Magee.

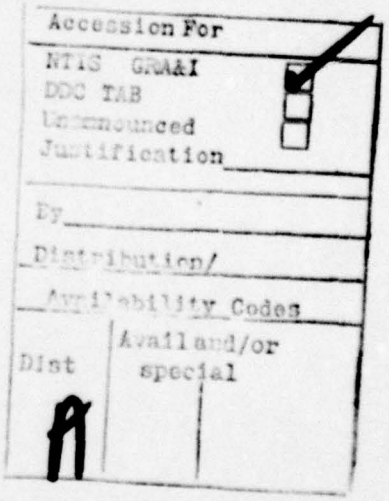


TABLE OF CONTENTS

INTRODUCTION

This report will describe efforts undertaken by Autometric, Inc. to determine the feasibility of performing the conversion of data from a field deployable data base to input data for a radar scene generation algorithm in the field within a less-than-ten-minute time frame. The method described below is not purported to be the only method, or even in any sense an optimal method. It is merely one which would work within the time allotted, and fulfills other criterion such as amount of disc storage required, and computer power available.

Estimates of computer time requirements include, in most cases, the fetch times as well as the actual operation times. It is assumed that the FORTRAN software will be written in such a manner that reasonably efficient code will be generated, and that such things as manual indexing through matrices rather than using the standard FORTRAN double subscripting methods will be used. The areas in which great care must be taken to prevent the generation of inefficient code by the FORTRAN compiler will be indicated where they occur.

In order to address processing speeds, one must make some assumptions as to the equipment which will be used. Therefore, an equipment configuration has been selected for this study. This selection in no way implies selection by the government of this particular configuration for the final system.

Given these conditions, the discussion of the process can begin.

Assumed Hardware Configuration

It is assumed that the hardware configuration available will be as follows. The central computer will consist of a Norden PDP11/70 with enough random access memory to support a thirty-two thousand word user partition in addition to the system and double buffers which will

be outlined below. Basically, this requirement translates to 42K words plus the system requirements. It is assumed that the system will have hardware floating point and direct memory access I/O capability, as well as a memory mapping capability. (All the above with the exception of hardware floating point are standard on the PDP11/70). The disk drives are assumed to be CDC 640 mil spec drives, with 80 megabyte total capacity. For timing purposes, it is assumed that the standard NTDS interface is being used, although we understand that a much faster interface is available for this drive. It is assumed that there will be two drives available, one for the use of the P-II field deployable data base, and one for use in program storage and swapping, and for the storage of intermediate and final generated data. Thus the P-II deployable data base discs will never be written on, thereby minimizing the probability of damaging the data base.

Assumed Processing Procedure

Because much of the information involving the actual scene generation procedure is proprietary to various vendors, some assumptions were required as to the format of the input to the final scene generation software. These assumptions were furnished to Autometric by ETL cognizant personnel, and constitute the best estimates available at this time of the required outputs.

The process basically consists of using a field deployable data base of topographic and planimetric information to generate radial format data which can be used for the generation of four scene to be used at various altitudes. The diameter of these four scenes is respectively 71, 35.5, 19.75 and 8.875 kilometers. Because the requirement for resolution increases as the altitude diminishes, the lower scene must have higher resolution, and therefore all scenes are considered to have the same number of picture elements per radial scan, and the same number of radial scans. It is assumed that there are 500 sweep line with 120 elts each. It has been indicated that the smallest unit of resolution being used currently in the scene generation is on the order of fifty meters. Therefore, we will assume

that the resolution of the field deployable data base will be approximately fifty meters.

The assumption was made that the field deployable data base could be in essentially any format which was required, limited only by the number of disc packs required to contain it, and the fact that only a single drive was available. Considerations of processing to go from the standard DLMS planimetric and topographic data to the P-II deployable data base were not taken into consideration. These processes go on in non-real-time in a base plant environment, and therefore must be considered to be of lower priority, with much higher time allowances, than the real time processing.

Given these considerations, then, the discussion of the individual phases of the problem can proceed.

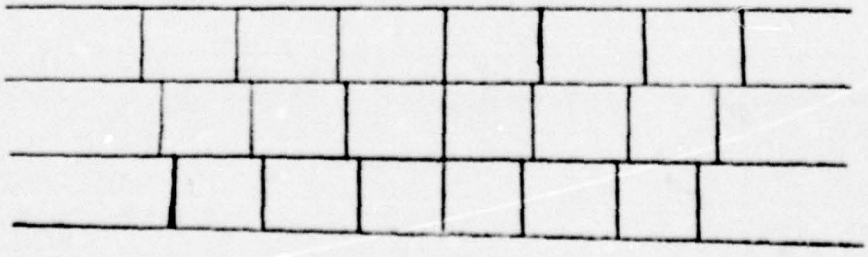
Definition of the P-II Field Deployable Data Base

Before the data base format is defined, a discussion of pertinent requirements would be in order. First of all, the reasons which will be seen later involving buffer sizes and data processing, the size of a unit in the data base should be approximately 71 by 71 kilometers, or possible just slightly larger. This assures that any single target can be covered by at most four portions of the data base, and that the amount of data to be read and processed will include a minimal amount of non-useable data. A smaller size for the unit of storage would admit the possibility that a single target might require nine or more units to cover it, while a larger size unit still could not avoid the possibility of having to use four units, but would require processing considerably more data.

Thus it is seen that the optimum size for a single unit, hereafter called a geounit, would be 71 by 71 kilometers. The next problem is to determine the coordinate system in which these geounits should be stored.

For purposes of insuring continuity of the coordinates throughout the area of interest, it is mandatory that all geounits be placed on one common system. It would be possible to use a single map projection for the entire 4 million square kilometer area, but the distortions at the outside edges would become enormous. The one system which is consistent over all of the European continent, and which is easily convertible to a local tangent system (the system which is used for the final generation) is the geographic (latitude and longitude) system. For purposes of this discussion the selection of the reference ellipsoid for this system is unimportant. The advantages to the geographic system are that every point on the system has a single unique set of coordinates, so that one does not have the problem of crossing grid zones which is so prevalent on systems such as the UTM system. Also, the coordinates are readily convertible to local tangent. There is one problem, however, which must be addressed. This is the problem of the changing scale of the system with latitude, which dictates that the data points, if they are to be evenly spaced in terms of distance on the ground, cannot be evenly spaced in geographic coordinates. Carried one step further, it also indicates that the size of geounits, as defined in units of latitude and longitude, will vary with latitude.

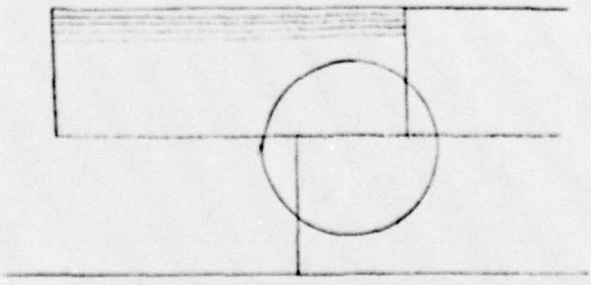
Suppose that the system is defined such that the total data base area is divided into bands in the latitude direction, each band encompassing a distance of 71 kilometers. These bands can actually take equal increments of latitude, since the value of a degree of latitude in terms of ground distance various very little over the latitudes of Europe. Within each band, the size of the geounit in longitude is selected such that the extent of the individual geounits at the maximum latitude of the band is 71 kilometers. A plot of the system in terms of geographic coordinates is shown below.



Of course, the divisions within the various geounits would vary in size in units of latitude and longitude so that the actual spacing for the geounit is 50 meters minimum, but this would be no severe handicap, as the spacing could be recorded at the beginning of the geounit record, and then it simply becomes a multiplier when stepping through the geounit. Indexing of the geounits would be simple because the extent of each geounit would be known given the latitude band, and the longitude of the central meridian would be known. Therefore, given the latitude and longitude of a target, the relevant geounits could be ascertained rapidly, and the disc location of these geounits could then be looked up in the appropriate index to the geounits on the discs to determine which disc needed to be mounted.

It will also be assumed that, for purposes of this study, all data is ordered along lines of constant latitude, rather than along lines of constant longitude. It is realized that the current terrain elevation data base is ordered along lines of longitude, and such an ordering would certainly be acceptable and would change nothing in the results to follow except for the detailed algorithms, but the ordering along lines of constant latitude seemed more natural to the authors, and thus was used.

It will also be assumed that the run length encoded compression scheme for the planimetric data which was suggested by Goodyear Aerospace Corporation will be used for storing the planimetric data in the P-II data base. There may be other possible methods, but this works within the disc space and retrival time requirements, and therefore will be assumed.



ASSUME DATA ORGANIZATION ALONG SCAN LINES

GENERAL TARGET SCENE

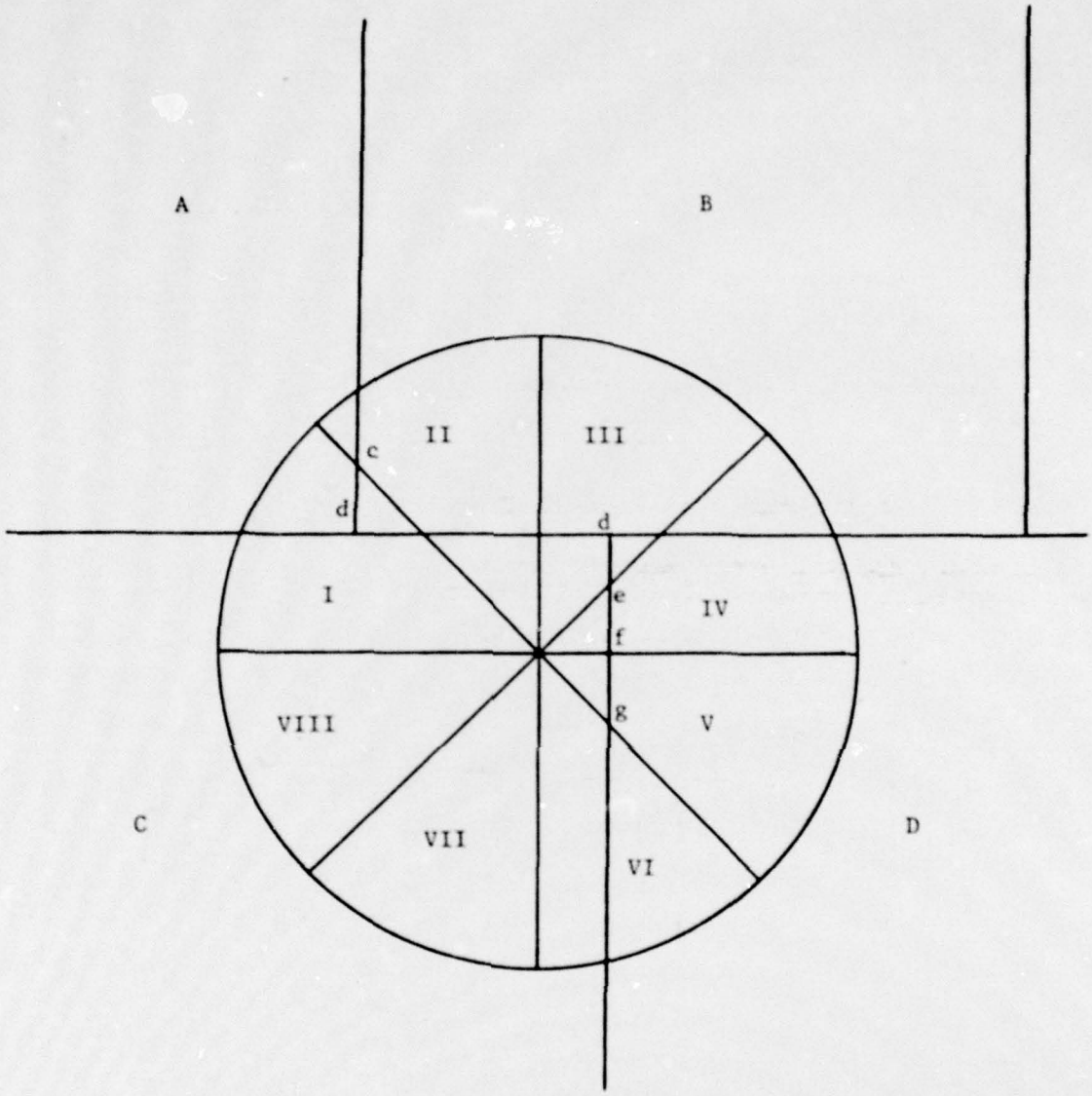


FIGURE 1

CONSIDERATION IN DATA STORAGE

- DLMS PLANIMETRIC DATA BASE HAS INFORMATION AT 25M RESOLUTION. REQUIREMENT FOR PERSHING II IS 50M. THEREFORE STEP 1 IN DATA COMPACTION IS TO PROCESS DLMS DATA TO 50M RESOLUTION.

- STEP II IS TO CONVERT DLMS DATA TO RASTER FORM AND RUN LENGTH ENCODE. STUDIES INDICATE 12 TO 1 COMPRESSION. THAT IS, 1 32 BIT DATA WORD PER 12 50M PIXELS.

- ASSUME USE OF CDC MODEL 640 MOVING HEAD DISK DRIVE WITH 4K DATA BUFFER.

- NTDS FAST CHANNEL HAS A DATA TRANSFER RATE OF 250,000 16 BIT WORDS/SECOND.

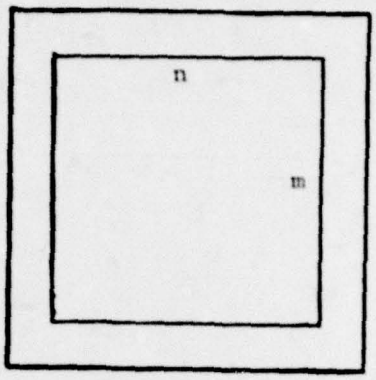
GEO UNIT = 1420 X 1420 50M PIXELS.

12:1 COMPRESSION ACHIEVED ONLY IN DIRECTION OF SCAN. THEREFORE ON THE AVERAGE $\frac{1420}{12}$ X 1420 = 168,033 32 BIT WORDS PER GEO UNIT OR 672,132 BYTES/GEO UNIT.

ASSUME ELEVATIONS IN ABSOLUTE @12 BITS/PIXEL (5M RESOLUTION GIVE 10240M RANGE.) EACH GEO UNIT REQUIRES 711 X 711 X 12 BITS FOR TOPO INFO = 758,282 BYTES/GEO UNIT THEREFORE:

TOTAL/GEO UNIT = 1.5×10^6 BYTES

It can easily be shown that the optimum amount of non-redundant data on a disk, assuming a certain fringe area to insure that any target can be contained on a single disk, is optimum if the number of geounits or non-reduntent information on the disk is an even square, and the geounits represent a square on the ground. This is easily seen.



$(N)\ (M) = K$ K Constant

minimize $(M+1)\ (N+1)$

$$(n+1)\ (m+1) = (n+1)\ \left(\frac{k}{n}+1\right) = k + \frac{k}{n} + n + 1$$

$$\frac{2\ (n+1)\ (m+1)}{2n} = \frac{k}{n^2} + 1 = 0 \quad \text{for min}$$

$$n^2 = k$$

$$n = k$$

so $m = k$ and $m = n$

Although the CDC disc unit has a maximum storage of 80 megabytes, only about 60 megabytes are useable for data. Thus there is a storage capability of about forty geounits per disk pack, assuming that the disc is formatted with five sectors per track.

However, problems occur when squares are required for the data stored on each disc. As was seen, a single disc will hold approximately 40 geounits, including overlap. This would allow only twenty five non-redundant geounits to be stored per disc, assuming that each disc contained a square area of data. The real problem arises because, as will be seen, the run length encoding scheme is not absolute, and therefore there is no way to predict a-priori what the compression effect will be. Thus, it might be possible on several occassions to arrive at an area in which only sixteen non-redundant geounits of information would fit on a disc, which would confuse both the book keeping for the disc library, and also waste considerable area on the disc.

Rather than take that approach, suppose that one were to use a disc to store a tier of three geounit extent in latitude, and as much as could be handled in longitude. For the square case, one would get twenty-five non-redundant geounits per disc on the average. For this case, one would get 40/(3+1) -1 or 9 new columns of geounits, for a total of 27 non-redundant geounits per disc. In addition, should the compression factor drop, one could easily store one less column of three geounits on the disc which would leave a maximum of wasted space of about ten percent on that one disc, and the overall block nature of the data base would remain unchanged. Indexing would be simple, because each disc would contain geounits along a band of latitude three geounits wide.

It could be argued that a band four geounits wide would be slightly more efficient (a single disc would hold 28 new geounits) but the cost of having to stop one column short in the case of low compression would be increased.

Also, one should note that the nature of the storage scheme defined above also allows significantly more than twenty-seven non-redundant geounits to be stored per disc in the case of areas of sparse planimetric detail which admit very large compression factors.

DISC TRANSFER TIMES

PLANIMETRY

. $\frac{1420 \text{ PIXELS/SCAN LINE}}{12 \text{ AVE COMPRESSION FACTOR}} = 120 \frac{\text{EFFECTIVE PIXELS}}{\text{SCAN LINE}}$

. 120 FEATURE WORDS = 240 16 BIT WORDS

. $\frac{1956 \text{ WORDS/SECTOR}}{240 \text{ WORDS/SCAN LINE}} = 8.15 \frac{\text{SCAN LINES}}{\text{SECTOR}}$

. USE A TABLE AT THE BEGINNING OF EACH GEO UNIT TO STORE THE BEGINNING AND ENDING LATITUDE FOR THE DATA WORDS IN EACH 20K BUFFER LOAD.

. FROM TARGET LATITUDE AND LOGITUDE:

1) COMPUTE GEO UNITS NEEDED
2) READ TABLES
3) COMPUTE BUFFER LOADS TO READ
4) READ SECTORS TO FILL COMPUTER CORE BUFFER

. DATA TRANSFER RATE:

ASSUME 5 SECTORS/TRACK
1956 WORDS/SECTOR

READ TIME FROM DISK TO COMPUTER

a) AVERAGE ACCESS TIME = 30MS
b) AVERAGE LATENCY TIME = 8.33MS
c) READ TO BUFFER AT 1.66 μS/WORD = 3.25MS
d) TRANSFER TIME AT 4 μS/WORD = 7.82MS
e) TOTAL TRANSFER TIME PER SECTOR 49.40MS

However, if we assume that one writes an entire core buffer in a somewhat contiguous manner on disc, a conservative estimate might be that there would be a 30ms access time for the first read, and about 5ms on the average for the reads after that. There are about five reads for a twenty buffer of information (twenty K bytes), which would take about 147ms to complete under the above assumptions. Because it will be needed later, the total time to transfer thirty-two K bytes would be approximately 220ms.

ORGANIZATION FOR PROCESSING DATA

ASSUMPTIONS

1. We go right from P-II data base to radial data base - the intermediate raster was used for convenience in polygon to raster conversion. According to ETL personnel it is not really needed.

2. The data in the geounits are ordered along rows of constant latitude. Run length coding obviously works in either lat. or long.

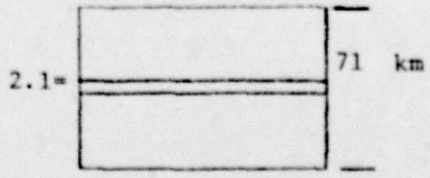
3. There is a 30 byte buffer available to generate 1/8 of an entire radial scene simultaneously (without having to output partial results to disc) 1/8 scene is 30000 bytes. This buffer is not double buffered.

4. There are 2 - 20K buffers for double buffering input data from the P-II data base. By use of the memory mapper, and the data channels in the PPP 11/70, only one of these (the one being used in computations) need to be the user area at one time. There is 64K - 50K or 14K left for user code and scratch storage.

SEQUENCE IN DATA PROCESSING

The basic sequence is to complete 1/8 of a scene at a time. The 1/8 figure is chosen as a balance between buffer size and number of passes through the data base, and is near optimum for the problem at hand.

The process begins by reading in a 20K buffer of planimetric data from geounit A. A segment, assuming a compaction ratio of 12, covers 20000/473 or 42 lines or 2.1 km.



All planimetric information is extracted and placed in the output, for the area covered by I. The area processed first is denoted by the red band in the drawing. The mathematics involved will be discussed later, for only the disc transfers will be studied.

Note that the output scene covers several data base units. Now, all the data base units cover essentially the same area on the ground, and have the same number of pixels. Let us look at the total reads involved. See table on following page.

Now, there are 4 scanes with radii of 35.5, 19.75, 8.875 and 4.4375 km respectively,

Thus, the total vertical extent (records) to be read in the worst case is:

$$8.828\ (35.5 + 17.75 + 8.875 + 4.4375) =$$
$$8.828\ (66.5675) = 587$$

say 600 km.

A 20K buffer of planimetric data covers 2.1 km, so there are 286 reads of planimetric data. A 20K buffer of topo data covers $20000/\frac{71000}{100} + 1$ X 1.5 rows X 100 meter/row = 1.877 km. So there are 319 reads of topo data. Each read takes 147ms or total disc time for input is 147 X (319 + 286) = 89 seconds. Output of the four completed scenes would take 4 X 8 X .220ms = 7.04 seconds.

Total disc time 96 seconds.

Now for the computations. If the disc I/O is not double buffered, the computation time adds to the output time, which is the pessimistic approach.

Look at the case of a single segment, and a single buffer load, as represented on the next page.

VERTICAL EXTENT OF DATA TO BE READ TO COVER EACH OUTPUT AREA

OUTPUT SEGMENT	HEIGHT COVERED "ORDINARY"	WORST CASE HEIGHT COVERED "CAUSED BY OVERLAP"
I	.707 R from A & C	c-d from B
II	R from A & C	R-d from B - (c-d) from
III	R from B & C	d-e from D
IV	.707 R from B & D	e-f from C
V	.707 R from D	f-g from C
VI	R from D	g-(-R) from C
VII	R from C	
VIII	.707 R from C	
	SEE FIGURE 1 FOR CLARIFICATION	2R or less depending on where intersection lie.

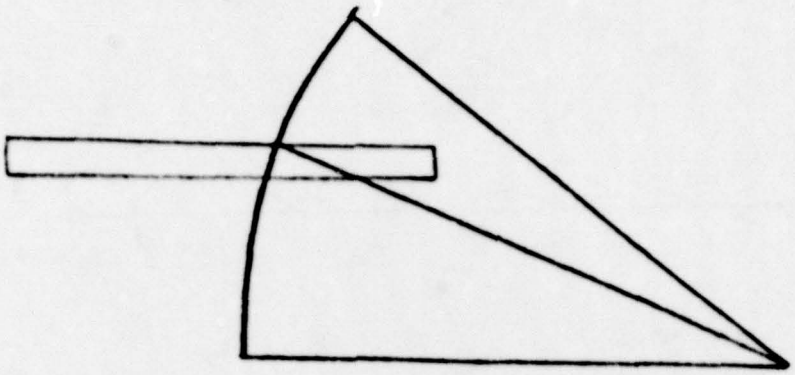


FIGURE 2

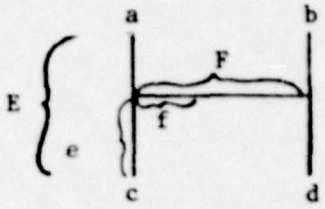
The idea is to fill in as many pixels of output as is possible from this buffer. There are several phases to the computation. First of all, the point in the output must be converted to geographics so that its position in the input can be determined. Secondly, the value at the input must be found, and third, it must be masked into the output.

Running some simple numbers says that to do this by brute force methods takes many minutes. Therefore, we must get smart.

Let's look at the problem in pieces. First of all, check the problem of conversion from local tangent to geographics. Suppose as a preliminary exercise when we began a scene that he were to compute a grid of local tangent to geographic values. Some preliminary computations show that, to better than 3 meter accuracy a grid of 6 X 6 stations would easily cover a quarter of the scene. Thus 36 X 3 or 96 words of storage would be needed for this table. Its computation would entail 2 to 3 seconds 4 times or a total of 12 seconds during the scene generation.

Given this table, one could linearly interpolate for latitude and longitude. A bilinear interpolation takes 2 divides, a multiply and and an add to find the start point and 3 more adds to address the other points. The actual computations take the form.

$$X = \left(\left(\frac{a(e) + c(E-e)}{E} \right) (F-f) + \left(\frac{b(e) + d(E-e)}{E} \right) f \right) / F$$



or for lat, long & ht.

		Time each
INDEXING	2 Div (Integer)	10.55 μs
	1 Mult (Integer)	5.30 μs
	12 Adds	2.30 μs
COEFFICIENTS	4 Subtracted	2.30 μs
	4 Stores	1.00 μs
COMPUTATION	39 Loads	1.00 μs
	18 F Hq. Mults.	2.5 μs
	9 F Hq. Adds	1 μs
	9 F Hq. Divides	3.5 μs
	3 Stores	1 μs

Which is still obviously too much work to do each time. But, we will be going along straight lines in the output. Thus, by computing a point every 50 or so points by the above method, one can reduce the process of moving from point to point to the adding of an increment to three values or 6 loads, 3 adds and 3 stores gets the new point lat, long & ht. offset.

Assume, on the average, only one point in 20 will need rigorous computation, whith the rest using simple increment addition. (For the small scenes, the compression could be even higher, but in large scenes, the bands of 2.1 km extent cover less than 20 points along a scan, so twenty should be a good number on the average).

So the operations to compute 20 latitude & longitude is

2 Int. Div.
1 Int. Mult.
9 F add
71 adds
153 loads
64 stores
18 F mults
9 F divides

or an average of 492/20 or 25 μs per point.

We now have the point in geographics. At this point, the problem splits into whether one is working topo or planimetry. First, however, one should look at the problem of computing the desired local tangent position of each output point. At first glance, it seems simple, because it is simply moving down a radial line. However, if the sines and cosines are recomputed for each 20K buffer of information which is read in during the process of generation an eighth of a scene, the computation time could be considerable. The solution to the problem is to store a table of the sines and cosines of the various radial scan lines for the scene currently being worked on in core, so that these can be looked up readily to determine the steps to take in moving down the scan, and also in computing such things as the first point on each scan which falls in the current buffer area.

Because these sine and cosine computations take a considerable amount of time, it would be adviseable to generate the same octant of all four scenes sequentially, preserving the table from one to the next. Thus, the table will have to be generated only eight times, giving a total of 500 sines and 500 cosines to be computed. This process should take less than two seconds total to accomplish.

There is also the problem of determining where to start each sweep line generation when each data buffer is in core. For every buffer read, there will be 125 lines to determine the start position on. This will require a divide and a conversion to integer. There are 605 buffer reads, so there will be 125 X 605 divides and conversion to integer, which will take approximately 1 second to perform. Thus, using the above scheme, the problem of determining the starting position on each scan line, and the

constants to use in stepping down the scan line will take approxmately 1 second total time for the generation of four scenes.

Now let's look at the topographic problem. For each point in the output, one needs to

1. Compute the latitude and longitude
2. Do a bilinear interpolation for elevation
3. Add the offset
4. Store

taking these steps individually, we see that (1) has already been found to take 25 μs. Because (1) will be done for some points which will not prove to be in the actual buffer area, because of the searching algorithm described above, one could assume a 10% waste, or 27.5 μs per point on the average.

The bilinear interpolation is the same as before, but is only in elevation, and so takes

INDEXING	2 Int div	
	1 Mult	
	4 Adds	
COEFFICIENTS	4 Subtracts (storage in registers)	
COMPUTATIONS	6 F mults	
	12 loads	4 divs
	3 F adds	4 shifts
	3 F divides	4 masks
or a total of	3 F divides	6 I div
	6 mults	1 I mult
	8 I adds	3 F adds
		4 shifts
	12 loads	4 masks

or 142 μs per point.

Adding the offset is one add or 2 μs. Storing consists of a masking operation, which at most could be 2 loads, a mask, a shift and an add, or 5 μs per point.

One thus arrives at an estimate of 149 μs per point for elevation determination, exclusive of latitude & longitude determination.

For planimetry, the problem is somewhat different. The computation of latitude and longitude is the same, but the major problem is not interpolation (the nearest point values must be used) but unpacking the run length encoded data if one were to read through t he buffer from the start to find each point, the problem is intrackable. Suppose, however, that for each row of data one were to keep the 4 variables:

(Min Long, Max Long, Attribute, Location) stored in a core buffer.

Determining the row of the point is simple. By the process described above, the scans tend to move monotonically in Longitude, therefore, if the point does not lie in the current section for that row, one must move in the appropriate direction from the location in the reference stack, and update the reference stack.

The point is that the entire buffer must be traversed once to form the beginnings of rows, and at most once again to handle the monotonic change. The first traverse requires that each of 5000 entries be read, and the 2nd byte compared to 0. This involves 5000 loads, 5000 adds, and 5000 compares. For the average of 42 rows, for the first record one must do a load, 2 masks, 6 shifts, 5 adds, and 4 stores. (Assume cache memory used for this critical phase).

This initial set up the tables

5210	I adds	.3 μs
5042	Loads	1 μs
5000	compares	.3 μs
84	masks	.3 μs
84	shifts	1 μs
164	stores	1 μs

To traverse once takes

5000*	1 I adds		5000	I adds (INT)	2.3
	2 compares		10000	compares	.3
	1 mask	or	5000	masks	.3
	4 loads		20000	loads	1
	8 stores		40000	stores	1

The previous page took care of all changes in the table for a given record, or 84.4 ms/record.

Now, for each point, one has only to select the row, which takes an add and a divide, a multiply and add to get the address of 2 loads, and two compares. Storage takes 3 masks, 3 shifts, 3 adds, and 3 stores. This gives a total per point of

6 I adds
1 I divide
1 I multiply
2 loads
3 masks
3 shifts
3 stores

The above gets done for every point, and takes 40.65 μs per point.

Now one can add up the operations, assuming no parallel processing.

Total disc read time	96 secs.
derivation of 4 Lat. long grids	12 secs.
Computation of point Lats & Longs (twice for each pt. with 10% waste)	(50 μs X 240000) + 10% 13.2 secs.
Computation of starting radii for buffers	1.2 secs.
Elevation interpolation	(149 μs X 240000) 35.76 secs.
Setting up buffers for planimetry (284 buffers X 84378)	23.96 secs.
Moving through on planimetry	9.75 secs.
Setting up size & cosine tables	3 secs.
	194.87 secs.

CONCLUSION

The above discussion has been carried out in considerable detail. It is believed that every major contribution to the total processing time has been taken into account. There is one major area, however, which has not been accounted for, and which cannot be accounted for prior to writing the actual software and running benchmarks. This is the area of program control and logical overhead.

For the above reasons, it would be wise to place roughly a factor of three safety factor on the above numbers. With double bufferring of only the input, this should still get the total computational time to well under ten minutes.

The above estimates assumed that the software was being written by a competent programmer who was striving to write extremely efficient code. This times could easily go up by an order of magnitude if the same type of careful analysis and procedures which were followed in the above analysis are not followed in the actual programming.