AD-A091 736 GOODYEAR AEROSPACE CORP AKRON OH F/G 9/2
CONTOUR DIGITIZING AND TAGGING SOFTWARE (CONTAGRID).(U)
APR 80 N J ADAMS, M ANDERSON, G BIECKER DAAK70-77-C-0223
UNCLASSIFIED GER-16752 ETL-0228 NL

1 OF 2
AD A091 736

ETL-0228

CONTOUR TAGGING

AND

GRIDDING SOFTWARE

(CONTAGRID)

FINAL TECHNICAL REPORT

APRIL 1980

Approved for Public Release:

Distribution Unlimited

Prepared For

U. S. Army Engineer Topographic Laboratories
Fort Belvoir, Virginia 22060

Prepared By

Goodyear Aerospace Corporation
1210 Massillon Road
Akron, Ohio 44315

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

The findings in this report are not to be construed as an official Department of 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.

[illegible]

UNCLA: SIFIED

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

(19) REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
(18) 1. REPORT NUMBER ETL-0228	2. GOVT ACCESSION NO. AD-A091736	3. RECIPIENT'S CATALOG NUMBER (9) rept.
(6) 4. TITLE (*and Subtitle*) CONTOUR DIGITIZING AND TAGGING SOFTWARE (CONTAGRID).		5. TYPE OF REPORT & PERIOD COVERED Final Sep 1977-Dec 1979,
		6. PERFORMING ORG. REPORT NUMBER (14) GER-16752
(10) 7. AUTHOR(s) N. J. Adams, M. Anderson, G. Biecker, R. Messner		8. CONTRACT OR GRANT NUMBER(s) (15) DAAK70-77-C-0223
9. PERFORMING ORGANIZATION NAME AND ADDRESS Goodyear Aerospace Corporation 1210 Massillon Road Akron, Ohio 44315		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U. S. Army Engineer Topographic Laboratories Fort Belvoir, Virginia 22060		12. REPORT DATE (11) April 1980
		13. NUMBER OF PAGES (12) 169
14. MONITORING AGENCY NAME & ADDRESS (*if different from Controlling Office*)		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*)

Automated cartography, automatic gridding, automatic tagging

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

The Contour Digitizing and Tagging Software (CONTAGRID) program demonstrates that a parallel/sequential processor combination can effectively perform automated elevation tagging and elevation gridding operations. It demonstrates that such a processor set is capable of automating the entire Digital Terrain Elevation Data generation task from processing the rasterized input map sheet overlay data to outputting the final digital product.

UNCLASSIFIED 156800

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

SUMMARY

The primary objectives of the CONTAGRID program were to determine whether automatic elevation tagging and elevation gridding processes could be developed for and executed on a SIMD* processor. To support these objectives, a pre-production hardware/software system that in an end-to-end manner transforms raster scanned sheet contour data into a Digital Terrain Elevation Data (DTED) form has been developed. It is somewhat awkward to use. It exhibits distinct hardware/software limitations. Yet, it clearly points to the production potential that a SIMD machine has for reducing the hours required for man-intensive carto tasks.

The system utilizes a combined SIMD/sequential processor - ETL's CDC 6400/STARAN computational facility - as its hardware resource. The application software, which was developed using structured design principles, comprises about 50,000 lines of source code. The amount of software required for the SIMD and sequential processor is about the same. For both types of processors, the source software was split about equally between executable and non-executable code. Source code was written in FORTRAN and APPLE.**

The essential objectives of the CONTAGRID program have been achieved. SIMD-oriented algorithms to accomplish automatic elevation tagging and elevation gridding were developed, implemented, and incorporated into the above system. Using real data, the elevation tagging software has executed fast and reliably; its built-in consistency checks have provided the automatic quality control actions needed to capture errors introduced as the result of incorrect automatic edit or manual operations.

The elevation gridding software provides the promise of higher quality elevation matrix data because it 1) is able to utilize high resolution information developed as the result of the raster scanning operation, and 2) it utilizes more independent known elevation points to develop

*SIMD - Single Instruction, Multiple Data Stream processor.

**APPLE - STARAN Associative Processor Programming Language.

the elevation at a grid mesh point. Color raster CRT display techniques evolved to check the efficiency of the algorithm, have the more universal value of being able to quickly expose errors of the output elevation matrix.

Based on the program results, execution times for the automatic processes of elevation tagging and gridding are projected to be under 10 minutes when using a production system based on a SIMD with processing power like that used in the CONTAGRID program and an I/O-oriented sequential processor like the SEL 32/75.

A major lesson, with far ranging implications, was learned during the course of the program. Raster scanners significantly degrade the quality of contour lines. The pre-production software system's pre-tagging software set was designed on the premise that the raster scanned contour data would conform to the DMA specifications for contour line data. The premise proved false. It was not possible to acquire such data for a full sheet (20" by 20"). Based on raster scanning attempts conducted over a period of over 1/2 year using a variety of DMA raster scanners, it was found that all scanners tried caused relatively severe data degradation during the digitization process; even though high quality source material was scanned, out-of-spec data was observed to be the norm rather than the exception. (GAC now believes data degradation should be accepted as "normal.")

The scanner problem limited system tests and evaluations. Of the raster scanner data collected, GAC was able to select only a nominal 4" by 4" region of SHIRAZ contour data that tended to conform to DMA specifications; the pre-production system was tested using this data. Of the three major sets of pre-production software, namely, the pre-tagging, elevation tagging, and elevation gridding software sets, sensitivity to the scanner degradation problem was found to be confined to certain modules of the pre-tagging software set. GAC has determined several means for making the pre-tagging software insensitive to data degradation. Tests conducted during early 1980 have indicated that the GAC-preferred procedure for making the pre-tagging software immune to the scanner problem is effective.

The pre-production system has proved invaluable for defining the necessary ingredients of a high throughput production system that can create DTED from raster digitized contour data. In particular, hardware modifications suggested include: an on-line color raster interactive edit station(s) that can compare different data types (e.g., vector and raster), high bandwidth channel(s) between the SIMD and sequential processor, much larger array memory for supporting the SIMD processing elements, and a smaller I/O oriented 32-bit sequential processor.

Software modifications suggested include: minimal modification of the pre-tagging software set to make it reasonably immune to raster scan problems, the transfer of the majority of sort, merge, and reformatting tasks to the SIMD processor, the addition of cartography oriented edit software (command and display) to support on-time interactive editing, and the addition of warp, mosaic, and panel software modules.

The CONTAGRID program has shown that the host of tasks that need to be performed by a DTED process oriented cartographic system can be accommodated by using a combined SIMD/sequential processor system. The ability of such a system to reduce man-intensive carto activity has been shown. Sufficient technical data is now (or can be made available) to accurately predict the capability and cost of such a system.

PREFACE

This final technical report records efforts and achievements under the Contour Tagging and Gridding Software program. This program was conducted by Goodyear Aerospace Corporation (GAC), Akron, Ohio and submitted by GAC as GER-16752.

The program was conducted for the U. S. Army Engineer Topographic Laboratories, Topographic Developments Laboratory, Mapping Developments Division (ETD-TD-M) under contract DAAK70-77-C-0223. Mr. R. A. Clark served as the Contracting Officer's Representative and provided valuable assistance in reaching the contractual objectives. This effort was implemented by N. J. Adams, (project engineer), G. A. Biecker, M. A. Anderson, R. W. Messner, J. King, and J. Vocar with contributions provided by R. Faiss.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT'D)

TABLE OF CONTENTS (CONT'D)

LIST OF TABLES

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS - APPENDICES

LIST OF ILLUSTRATIONS - APPENDICES (Cont'd)

SECTION 1 - INTRODUCTION

1.1 BACKGROUND

Since 1974, Goodyear Aerospace Corporation (GAC) has been actively involved, with ETL, in the investigation of STARAN* type processors for Automatic Cartography related problems.

By 1976, GAC had demonstrated that STARAN could use raster scanned pencil drawing inputs and produce clean line and areal symbologies (it also was used to spot, scale, and orient point symbologies). To accomplish the above tasks, STARAN raster-to-vector software with auto-edit capabilities was developed; as required for this procurement, the software was written to tolerate lines with variable line width when converting raster scanned sheet data to a vector data base form. To develop the final graphics output product, software was written not only to convert thin line vector data in data base form back to a raster form but also to embellish it with line symbol attributes. Thus, thin line vector coordinate data, along with a symbol type descriptor, were found to be sufficient for automatically producing such line symbologies as double casement roads, intermittent streams, and railroads, as well as areal symbologies (e.g., swamp regions) with or without included boundary lines and with required fill patterns. For line symbologies, junctions that conformed to cartographic esthetic standards were automatically generated.

During 1976 and 1977, software developed earlier to demonstrate STARAN's cartographic processing power was made more flexible.

*Trademark, Goodyear Aerospace Corporation, Akron, Ohio 44315

In particular, the software was altered to allow variable scan resolutions, different sheet sizes, a greater range of line widths, etc. Input/output formats were set up to be compatible with either the DMA RAPS or ETL-IBM raster devices. This software allowed STARAN to perform cartographic tasks in a stand alone mode.

In 1978, the cartographic software cited above was set up to be used at ETL in conjunction with the CDC 6400 computer. This set-up accounted for the file structures, file sizes, and I/O needed for the large-scale conversion of sheet graphics raster data to a vector form and was used to make raster-to-vector format conversion timing tests. The STARAN Raster Processing Software (STRAPS) developed allowed map sheet region windowing, limited scaling, and plotting of vector data (developed from the raster form) on CALCOMP, CALMA/NOVA, and GERBER plotters. The STRAPS execution time performance tests showed the extraordinary processing speed of STARAN; it demonstrated its reliability and effectiveness.

1.2 PURPOSE

The purpose of this contract was to show that the vector data generated from raster scanned contour map sheets could be reliably associated with elevation data automatically, provided that a limited number of contours that intersect sheet boundaries have been assigned elevations manually. The program was further to provide for the generation of gridded elevation data using the elevation data of contours (in either the DMA standard input format or STARAN format), R/S data in DMA standard input format, and topographic point elevation data in DMA standard input format.

1.3 REPORT ORGANIZATION

Section II of this report provides the reader with a CONTAGRID program summary. Detailed summaries and conclusions of the various program elements are found in Section III; recommendations are found in Section IV. Section V of this report, INVESTIGATION, describes the approach, the testing systems, and the general task involved. Section VI, DISCUSSION, contains a description of the pre-tagging, tagging, gridding, and editing procedures. Section VII describes the RESULTS.

Appendix A provides a high-level description, along with top-level structure charts of the entire CONTAGRID program. Appendices B, C, and D give detailed descriptions of the pre-tagging, tagging, and gridding processes. Appendix E provides a brief description of some utility routines. Appendix F provides the file/record layouts of the various data sets developed during the processing by the CONTAGRID software.

SECTION 2 - CONTAGRID PROGRAM SUMMARY AND CONCLUSIONS - OVERVIEW

The CONTAGRID program had manifold purposes. Primary and secondary objectives of the program were the development and implementation of automated, SIMD-oriented elevation tagging and elevation gridding algorithms. A major system objective of the program was the integration of new and previously developed software sets into a pre-production system that could transform map sheet overlay data into the Digital Terrain Elevation Data (DTED) form.

The DTED transformation process involves three phases, namely, pre-tagging, elevation tagging, and elevation gridding. Prior to the CONTAGRID program, ETL/GAC had shown that the SIMD processor was highly effective for performing the major task of pre-tagging, namely, automatic raster-to-vector data conversion. Related R&D has shown that the SIMD can execute a wide spectrum of other tasks required by an automatic cartographic system (e.g., warping, format conversions, etc.) Now the CONTAGRID program has shown that the primary tasks of elevation tagging and elevation gridding are also performed effectively using the SIMD processor. Moreover, it has verified that the various tasks can be performed in a coherent end-to-end manner.

The CONTAGRID pre-production system that has been developed has limitations, It omits mosaicking, paneling, and warping processes in the end-to-end chain of processes required to produce a DTED product from digitized contour and R/S overlay data. As a consequence, CONTAGRID's data output has the format, but not the exact form of that required for a production product. No fast on-line interactive edit capability is provided. More operator action is required to operate the system than could be tolerated in a production environment. An end-to-end production system requires the omitted processes, fast on-line interactive display/edit capability, and a simplified flexible operator command input capability.

Moreover, it would need to provide job management capabilities, such as command logging, checkpointing, and process status reporting.

The transition from an end-to-end pre-production system to an end-to-end production system requires no formidable technical problems to be solved. Because the pre-production system was structured to accommodate transition, the primary transition tasks will be related to configuring the system to account for the myriad of details encountered in a production environment.

The CONTAGRID program has rounded out the R&D activity needed to technically judge whether or not a properly balanced SIMD/sequential processor combination does indeed provide the appropriate basis for an automated end-to-end cartographic processing system that can tolerate poor quality input data.

CONTAGRID testing has shown that raster scanners substantially degrade data during the digitization process. Certain modules of the pre-tagging software set require modification in order to make them resistant to the scanner problem. The required modifications have been identified and can be implemented readily.

The hardware of the pre-production set (the ETL CDC 6400/STARAN processor set) exhibits deficiencies that would be undesirable in a production cartographic system. The absence of an on-line interactive color raster/vector edit station, the low bandwidth between the sequential and SIMD processors, and the very small SIMD memory were most noticed. The hardware inadequacies tended to make software development difficult, resulted in inefficient editing, and bloated the processing time measurements. The CONTAGRID tests point directly to the hardware changes required for an efficient system. Using projections based on the pre-production system test results, a production oriented auto carto system based on a SIMD with beefed-up memory and an I/O oriented sequential processor (e.g., a SEL 32/75) could perform the major automatic full (20" by 20") SHIRAZ sheet DTED transformation tasks in the times shown below:

● Pre-tagging (including Raster-to-Vector Conversion)	- Less than 20 min.
● Tagging	- 2 min.
● Gridding	- 5 min.

Critical lessons were learned during the course of the CONTAGRID program. Necessarily, such lessons would directly affect the design of a system tailored for the end-to-end cartographic process. In particular, they relate to data conditioning, data editing, system configuration, and system tasking. A few major lessons follow:

- Existing raster scanning devices tend to degrade the cartographic information of the map sheet overlays. Both systematic and non-systematic degradation occurs; furthermore, the degree of degradation is likely to change over the face of the map sheet.

- Software must be designed to tolerate a substantial range of degraded data, or manual editing requirements will rise at an exponential rate.

- An interactive on-line editing capability is essential. The time lags and reformat operations that accompany the passage of data between an off-line editing station and the the main cartographic processor cannot be tolerated. Time lags reduce the efficiency of equipment utilization and the efficiency of a human operator's activity.

- An edit station must have the ability to display fundamentally different data types concurrently (e.g., Vector domain data must be able to be displayed and directly compared to raster domain data in a common coordinate system). Certain tasks are performed best in the raster domain; others are performed best in the vector domain.

- It is more crucial that auto editing operations treat the large number of microscopic faults (e.g., small line breaks, clutter, pips) than the small number of macroscopic features (e.g., numerics). It is primarily the count of independent edit events that affects the human edit load.

- Functional operations must be modularized to support the smooth flow of data between automatic and interactive tasks in an operator-prescribed order. Software should be based on a structured design philosophy.

- The form of file structures at any state of the data transformation process must be designed to support rapid access to randomly selected small regions of the map sheet (i.e., data must be "regionalized" throughout all the process tasks).

- Certain standard format conversions performed to get hard copy proof-plots take exorbitant amounts of sequential processor CPU time. These tasks should be performed only for final quality control assessments and should be performed in the SIMD processor.

- In general, the tasks of data format conversion, sort, and merge should be performed by the SIMD processor so that the sequential processor can be primarily reserved for interactive command processing, systems management, and I/O control functions.

- The simple act of specifying large CPU, memory, and channel hardware resources for a system does not by itself assure that it will be a cost effective cartographic system. It is absolutely essential to balance the hardware resources of the system against the kinds of tasks required by the end-to-end transformation process.

SECTION 3 - RECOMMENDATIONS

- The CONTAGRID R&D study and prior carto-oriented studies have provided sufficient technical information to project that a joint SIMD/sequential processor set can provide an effective basis for an end-to-end cartographic system that would substantially reduce the man-hours required to produce elevation matrices from map overlay data. Therefore, GAC strongly recommends that DMA HTC expedite the introduction of a DTED-oriented turnkey cartographic system (based on such a processor set) into the DMA HTC production environment. Further R&D effort to prove the essential worth of the SIMD processor is not required. DMA ought now decide whether or not to capitalize on the laboratory proven features of SIMD processors in a production environment. If DMA concurs with GAC's assessment that such a system would be desirable, the actions needed to realize such a system should commence without delay.

- The production cartographic system delivered to HTC to transform raster digitized contour data to the DTED from should be end-to end in nature so that all sources of product cost can be attacked in unison. Too often, systems that attack only subsets of the production problem cause additional cost elsewhere; an end-to-end system can make substantial cost savings by combining operations as well as accounting for them. An end-to-end system has greater resources and flexibility and so can accommodate changes aimed at saving man-hours. More over, such a system becomes relatively immune to changes in production emphases.

  A total system's approach is a must when configuring the end-to-end system. Both long term and short term production demands should be considered.

- Assuming that DMA pursues the acquisition of a system, the major goal of future DTED-oriented R&D activity should be to drive the man-intensive activities of the DTED process down to the 40 hour/sheet level by 1985; such activity should begin immediately. The first major emphasis of such an R&D program should be aimed at slashing the time now required for performing R/S compilations. A three-pronged attack should be employed. First, system software should be added that simplifies the R/S compilation task of the operator. This work should develop techniques that allow coarse R/S drawings to be refined on a raster editing station using such aids as toned overlays. Secondly, an investigation of the utility of R/S lines generated automatically from contours ought to be aimed at determining the man-hours required for reshaping such lines. Thirdly, the applicability of more sophisticated splining or convolution techniques to the gridding problem should be investigated in the light of the severe undersampling of elevation data on contour sheets.

  R&D to develop a comprehensive software set for automating the process of creating the DLMS feature data from existing cartographic materials should begin after the DTED R&D plan is established. In particular, it appears that the polygon encoding capability of the DTED system described would be exceptionally powerful. Even the automatic association of a feature number to a polygon appears feasible. The cartographic feature filtering capability of the SIMD has already been demonstrated. Whatever R&D is conducted, it ought to fit within the priority assignments of the master R&D plan developed for achieving the desired software set. The DTED-cartographic system discussed earlier should be designed to accommodate such additional work.

SECTION 4 - SUMMARY AND CONCLUSIONS BY PROGRAM ELEMENTS

4.1 PURPOSE

The present DMA process of converting map sheet overlay data into digital terrain elevation data form - the Digital Terrain Elevation Data (DTED) generation process - relies heavily on the performance of man-intensive tasks. The cost and turnaround time for the process reflects the heavy reliance on manual operations.

The CONTAGRID program was designed as part of a continuing effort to verify predictions that a Single Instruction, Multiple Data Stream (SIMD) processor - STARAN - can be used effectively as the kernel of an automatic cartographic system that executes the process quickly, reliably and with substantial cost reduction. Its primary objectives were to prove the processor's ability to elevation tag and to elevation grid. Solutions to technical problems encountered during the course of the program have been found; the essential program goals have been achieved.

4.2 MAJOR ACCOMPLISHMENTS

The accomplishments of the program were many and varied. They include the following:

1) the first development of an end-to-end cartographic software system based on a combined sequential processor/ SIMD processor hardware system,
2) the development of an automated tagging procedure that includes highly effective self-check quality control features,
3) the development of a SIMD processor-oriented gridding procedure that has the latent capability for substantially reducing R/S requirements,
4) the development of vector domain auto-edit software that complements GAC's existing raster domain auto edit software,

5) the development of edit display and command software to aid manual editing,

6) the evolvement of region oriented file structures to support a "fast" interactive edit display capability,

7) the implicit testing of the scanning characteristics of a variety of DMA raster scanners,

8) the establishment of timing information for the various process tasks and the significance of this information,

9) the successful use of structured software design and development procedures for the total CONTAGRID system software (including that required for STARAN), and

10) the development of unique software debug product test aids.

As a result of the program, measurable software and hardware performance characteristics of the CONTAGRID program can be used to accurately extrapolate the character of a production system with a given throughput capability.

4.3 CONTAGRID PROGRAM METHODOLOGY AND SCOPE

To verify the ability of STARAN to be the basis of an automatic cartographic system, the program required the development of a pre-production end-to-end cartographic software system that would perform the DTED generation process with minimal operator intervention. The ETL CDC-6400/STARAN facility at Fort Belvoir, Virginia was used for simulating the cartographic system hardware. The pre-production system, which utilizes on the order of 50,000 lines of source code ($\approx$25K executable source), now exists.

The value of the pre-production system lies in its ability to yield measurable software and hardware performance characteristics needed to extrapolate the character of a production system with a given throughput capacity. Because software modules of the system are used in a chained manner, the real accumulated impact on data far along the processing chain caused by a seemingly innocent action of a module early in the chain becomes immediately apparent - the system across-module problems (as opposed to intra-module problems) are exposed. By requiring a pre-production system, the problems associated with realistically transmitting large data sets, from one software task module set to the next, had to be addressed.

The pre-production software has in fact, provided the type of information that allows the extrapolation of:

1) algorithm requirements,
2) software function and performance requirements,
3) hardware requirements, and
4) gross human factors requirements
   for a production CONTAGRID system.

The major categories of man-intensive activities involved in the DTED generation process that influenced the design of the pre-production software are:

1) compilation,
2) analog (sheet form) to digital data conversions,
3) parameter association activity,
4) editing, and
5) quality control.

Because the system software was structured with the progressive automation of man-intensive activities in mind, it can incrementally accommodate the incorporation of modules that eliminate or modify man-intensive tasks.

To develop the pre-production system, it was necessary to address not only the structure of application modules, but also, the data sources/destinations, the character of data blocks, and the format of data sent to and received from the modules - the logistics of module servicing were established. Mechanisms for saving/deleting input/output data files, as well as the file structures themselves, have been developed so as to allow the flexibility of execution of modules after input data modifications.

An integrated set of mechanisms for displaying and editing data have been established. The software handles the hidden problems of editing, namely, how edited data are merged into the file structures of non-edited data, how data are deleted, how data types are identified, and how data format differences are handled internally or on input and output when the data sets are very large (10+ megabyte range).

All operations of the pre-production system are logged and reported; thus, the basis for operations management and control is provided.

4.4 DTED GENERATION SEQUENCE AND PROGRAM FOCAL POINT

The pre-production software system accomplished the DTED process in three phases, namely:

1) data preparation (pre-tagging),
2) data tagging, and
3) data gridding.

Prior to the CONTAGRID program, GAC had shown that STARAN could perform the primary task needed to accomplish the major activity of phase 1), namely, raster-to-vector conversion, at a rate in excess of 10 times faster than any other approach, even without optimized software. Other tasks required by phase 1 (line separation by weight, raster-domain auto-editing, etc.) had also been demonstrated so the task of producing phase 1 software at program onset was considered a low cost, low risk task.

The STARAN software for performing phase 2 tagging (the operation that associates elevations to contour vectors) or phase 3 gridding did not exist. Yet, DMA-designed UNIVAC gridding software did exist, and so the development of a gridding algorithm and subsequent software for STARAN was considered a low risk item; the development of gridding procedures became a secondary goal of the CONTAGRID program.

Automatic tagging procedures did not exist prior to the CONTAGRID contract (although initial GAC studies indicated that tagging could be accomplished with STARAN). As a result, the complete development and implementation of an automatic tagging algorithm became the focus of attention and the primary goal of the CONTAGRID program.

Goodyear Aerospace has successfully accomplished both the primary and secondary goals of the CONTAGRID program, namely, the development and implementation of automatic STARAN-oriented TAGGING and GRIDDING procedures.

Both procedures provide vital features not included in the original set of program requirements. In particular, the STARAN tagging procedure embeds a powerful automatic tagging self-verification strategy that pin-points problems automatically - it provides automatic tagging quality control. The STARAN gridding algorithm incorporates more independent points of known elevation in a larger sample space in order to establish the elevation at the matrix mesh points; thus, better elevation matrix quality than can be achieved with DMA's planar interpolation algorithm is expected.* More importantly, the structure of the data in STARAN allows the algorithm to be modified in such a way that the need for R/S data (and associated manual compilation effort) can be expected to be reduced substantially without degrading quality. Finally, unique display procedures for rapidly exposing elevation matrix deficiences have been defined and implemented; samples of the displays are found in the report.

The most severe technical problem encountered during the program showed up where it was least expected - in phase 1 (pre-tagging)

*DMA and ETL are presently evaluating GAC's gridding algorithm performance.

processing. Ultimately, the root cause of the problem, the inability of DMA AC or TC RAPS, Hamilton/Standard, or SCI-TEX raster scanners to maintain line weight tolerance specifications for index and non-index contours, was identified. Goodyear Aerospace now believes the problem to be fundamental to bi-level raster scanners. Thus, GAC believes the increase in the variance of line weights over that found in the sheet material should be considered normal and treated accordingly.

Tests, to date, imply that the pre-tagging software would perform its functions as required if it were "fed" with input raster data that contained contours whose line weights met DMA specifications. However, as described above, it appears unrealistic to expect that such data will ever be available. Since late in 1979, attempts to get data that conforms to specifications have failed. With out-of-spec input data, pre-tagging's present line separation module breaks up contours while sorting them into index and non-index files. While the module could be modified to accommodate out-of-spec data, GAC believes a modification of the pre-tagging procedure that eliminates the need for the line separation function is preferred. (The modification procedure is discussed later in this section.)

Tests to determine the effectiveness of GAC's vector domain feature detection software have become a casualty of the raster scanner problem. The effectiveness of this software necessarily is tied to the level of degradation of the data base; when the pre-tagging modifications are made, the tests can be made.

One benefit of the raster scanner problem is that it quickly exposed the inadequacy of using a plotter as an editing display device. The time lags associated with generating the whole series of displays needed for effective editing is intolerably great. Despite the theoretical availability of a whole series of plot (display) options provided by CONTAGRID's edit aids software, the exorbitant amount of time needed to generate the plot options ensured that few options could be used. It is strongly felt by GAC that an on-line, low lag-time, interactive display that can overlay a rich variety of data types is necessary for conducting efficient editing operations. A color, raster CRT display meets the majority of these requirements.

While it took considerable time to verify the root cause of the phase 1 problem, its ultimate impact on CONTAGRID cartographic processing software will be small. A number of alternative procedures exist that either accommodate the problem or by-pass it.

4.5 Software Development Priorities

The design of the CONTAGRID cartographic system was governed by a simple rule: conserve resources for the primary task, tagging, and (to a lesser extent) for gridding. In practice, the rule simply meant that existing software, CDC or STARAN, was used whenever possible. Tasks were performed on the CDC (and so FORTRAN was used) whenever such tasks did not directly relate to the execution of processing expensive pre-tagging, tagging or gridding tasks. The STARAN code generation was generally restricted to key tasks. The efficiency of a software module was generally ignored except when it related to the task of tagging and to a lesser extent, gridding. It will be seen, when discussing the execution times, that a number of tasks (e.g., sort/merge and format conversion oriented tasks) performed on the CDC-6400 should be transferred to the STARAN in order to decrease program running time.

4.6 Software Design, Development, and Test

A formalized structured design effort was employed to design the CONTAGRID software system. The top-down design was accomplished using a team approach. Members of the team were about equally divided between those familar with STARAN and those familiar with the CDC-6400. The top-level structure chart for the software system is shown in Figures A-1 and A-3. Note that in Figure A-1, from left to right, the 1st (7) structures refer to pre-tagging and the next (5) refer to tagging. All nine structures of Figure A-3 refer to gridding. Table 4-I summarizes the magnitude and distribution of the software effort. The fact that nearly half of all software generated is FORTRAN reflects GAC's effort to conserve man-hour resources for tagging, gridding activity. (See Table 4-II).

Goodyear Aerospace's overall assessment of the team centered structured design approach is that it is effective. Yet, it should be cautioned that in following the top-down approach, one must consider the operations of the lowest level modules and account for the hardware constraints of a system. After performing a top-level design, it is necessary to assess the impositions of the design on all time critical modules.

The use of GAC's STARAN and the CDC CYBERNET aided the software debug operations at the module level. But, no complete simulator of the joint ETL CDC-6400/STARAN facility existed at GAC so most software integration and system testing had to be performed on-site at ETL without the benefit of a preview at GAC. The need to do system integration and testing on-site degraded the efficiency of performing this activity. The basis for the decreased efficiency relates to schedule clashes that occurred when the ETL facility was tied up for performing high priority tasks, to reduced access to the facility for security reasons, and to the awkwardness of using a batch oriented CDC software system for system debug. Goodyear Aerospace substantially underestimated the difficulty of working in this type of environment.

TABLE 4-I. BREAKDOWN OF DESIGN, CODE, TEST MAN-HOURS

FUNCTION	TOTAL (HOURS)	PERCENTAGE OF TOTAL
SYSTEM ANALYSIS & DESIGN	2,271	15
SOFTWARE DESIGN	1,475	9.75
SOFTWARE CODE & DEBUG	6,588	43.5
TEST	1,520	10
INTEGRATION	1,680	11.1
DOCUMENTATION	842	5.6
FACILITY CHARGES, KEYPUNCH, ETC.	172	1.1
LEASE	8	0.05
TRAVEL	584	3.85
	15,140	99.95

TABLE 4-II. INSTRUCTION COUNTS - CONTAGRID

MODULE	STARAN SOURCE		CDC SOURCE	
PRE-TAG (Executable)				
TAPEIN	-		345	
GETVECS	925		400	
GETMVECS	-		1,020	
GETEDS	2,610		370	
PLOTGEN	-		690	
VLDVEC	205		1,600	
VPLOTGEN	-		620	
SQUEEZE	-		1,050	
		3,740		6,095
TAGGING (Executable)				
OPNTAG	840		440	
EDITAGS	-		195	
CLOTAG	2725		420	
TAGPLOT	-		380	
TAPEOUT	-		240	
		3,565		1,680
GRIDDING (Executable)				
PARAM	-		25	
SPLITJ	-		25	
RESAMP1	1,542		400	
RESAMP2	-		900	
SEGDGR	-		364	
NEAT	-		300	
PARTSR	-		594	
FINDINT	585		350	
BILDSRP	-		575	
MERGBAIC	-		355	
GRID	1,320		225	
		3,447		4,113
MISCELLANEOUS (Exec.)				
I/O (Gridding) Sim.	395		395	
Gridding display	600		600	
Utility routines	500		500	
		1,495		
COMMENTS(All modules)		≈12,500		≈12,500
SUBTOTALS		24,747		24,388
TOTAL		49,135		

Quite late in the CONTAGRID program, to overcome the difficulties described, GAC introduced a partial simulator of the ETL system into its Akron facility. This simulator significantly simplified the installation of Gridding software at ETL.

4.7 Software Features/Characteristics (Pre-Tagging)

The function of the pre-tagging software is to convert raster domain contour descriptions to the appropriate vector domain form, to "clean" the data, and to assure the integrity of the contour vector data base that is used for tagging. The software features:

1) automatic separation of index and non-index contours,
2) raster-to-vector conversion,
3) raster domain auto-edit operations,
4) automatic vector domain contour break detection, localization and labeling,
5) automatic vector domain feature classification,
6) automatic vector domain auto-editing,
7) edit display options, and
8) edit command options.

Because of the raster scanner problem, there is little likelihood that the line separation software that yields the first feature listed above will be used for separating index from non-index contours in a production system. While it could be used in modified form for such a purpose, it will probably be reserved for future feature classification algorithms.

The pre-tagging raster-to-vector domain conversion software retains the same raster domain auto edit features as are found in earlier STARAN software. In particular, auto-edit features to remove clutter, to close small breaks, to thin a variable width line to a centerline, and to remove short stubs emanating from a line,exist as part of the present software set. The raster-to-vector software still develops vector data on a regional basis (nominally, in rectangular patches of 160 x 184 pixels), still converts data from raster-to-vector quickly (although not optimally), and stores data in the condensed starburst form. New vector edit software to support pre-tagging and tagging has been developed for the CONTAGRID software system. Auto-edit and manual edit components of the software are complementary.

The auto-edit software that is of most theoretical interest is that which automatically classifies numeric, depression, cut, and fill symbologies. After classification, the software is designed to perform the appropriate auto-edit action.

Because of the raster scanner data problems, it has not been possible to evaluate the effectiveness of this classification and fix software with unfragmented data. Most tests to date have been performed using fragmented data that includes elevation numbers. Even with dirty data, numerics have been detected, localized, labelled, classified, and deleted. Yet, efficiency of the numeric classification software suffers from the dirty data environment. Of numeric vectors existing in data regions tested (36), a shade over 60% were correctly categorized. The remaining numeric vectors that were not categorized were all short and thus failed minimum length numeric vector consistancy checks. The short vectors resulted directly from the fragmentation of the numeric symbols. If GAC's solution to data fragmentation were implemented (see Section 4-11), the probability is that virtually all the short numeric vectors would vanish; the proper categorization of true numeric vectors would then approach 90 to 95 percent.

A more severe problem occurs when non-numeric vectors are falsely categorized as numeric vectors. Relative to the total number of true numeric vectors of the test data, about 15% of non-numeric vectors were categorized as numeric vectors. The falsely classified vectors were derived from tightly curved, snake-like contours that were chopped into adjacent segments by the map region boundary.

The segments formed a vector set that met the conditions of an elevation number and thus were improperly categorized as numeric vectors.

Since the false categorization problem has been caused by the cutting action of the data region boundaries, the problem solution could be as simple as to exclude vectors within a guard strip of region boundaries from the numeric classification test. While this would result in non-categorizations of true numerics in the guard region, the percentage of numerics in the guard region compared to those in the total region would be small. Solutions that apply more rigious tests to distinguish numeric quality certainly could be applied if they are in order.

Vector domain automatic join software that complements similar raster domain software is needed to reduce the need for manual intervention to heal contour line breaks. Using the available test data, the vector domain automatic line "fixing" software correctly developed about 80% of all (14) line join vectors required. The remaining 20% of short joins were not created because end point criteria were not met. Only 60% of all (5) long line joins were accomplished correctly. The remaining 40% of long line joins were built between contour ends and numeric vector fragments that weren't automatically deleted. Also, relative to the total number of correct line joins required, about 10% of unneeded line joins occurred between numeric fragments and other numeric fragments. By eliminating number fragments prior to the line join operation, the long line join operation would become much more efficient.

It should be noted that for a full sheet, about 350 short line breaks would be expected for a 20"x20" region of the type data tested. Using the present line break "fix" software, about 70 would require manual joins. It would appear that manual interactive edit requirements would not be excessive. Nevertheless, GAC believes that the line join algorithm should be modified to raise its effectiveness for at least short line breaks to better than 95%.

Minor modifications to the line join algorithm would be required to attain this level of effectiveness for short line breaks.

For long line breaks, more extensive modifications are required; slope data near line break end points would need to be used; it can be directly derived from the vector data. If an interactive, on-line editing station is employed in a production system, the ability of such software to reduce manual activity is much diminished. Whether the long line break "fix" software modifications should be incorporated into a production software set would largely depend on the nature of available hardware interactive edit station resources of the system.

Tests to determine how well the classification and fix software performs when treating other than numeric symbologies have been even more limited. (Yet, recent feature detection tests performed by GAC apart from the CONTAGRID program have shown that when the preferred GAC "fix" for the line scanner problem (see Section 4-11) is utilized, unfragmented data results and the vectors of even more complex symbologies than those of the CONTAGRID program (e.g., railroads) have been identified at a level of effectiveness in excess of 90%.) The potential value of the software remains high, but algorithm effectiveness tests with clean data are yet required.

The display capability provided to support interactive manual editing includes the capabilities listed next. (Significant restrictions are bracketed.)

1) Ability to extract and plot (display) a variety of vector types including:
   a) original index line vectors,
   b) sections of index line vectors generated by the editor [To get the plot specified without modifying the present software, it is necessary to manually intervene while making a proof plot of index vectors after the edit operation. In the list of index vectors, edit (or join) vectors appear first and so will be plotted first. After they have been plotted, the plotting pen returns

to the origin prior to plotting the original non-edited vectors. At this time, the plotter must be turned off manually. The simple conversion to an automatic shutdown procedure is implicit in the order of the data.],

c) original non-index line vectors,

d) sections of non-index line vectors generated by the editor
[1b) type restriction applies. If "non-index" replaces "index" in the 1b) restriction, the correct 1d) restriction results.],

e) numeric features, and

f) depression, "cut," and "fill features.
[Presently, the various symbologies are lumped into a single plot. Yet vectors of the various symbologies are ordered by symbology and so could be separated during plotting in the 1b) manner described above. In the same manner, the ability to automatically generate the vectors of a given symbology can be implemented readily.]

2) Ability to plot combinations of vector types.
[Combinations of vector types are obtained by over-plotting the data of different files.]

3) Vector identity labels may be plotted in association with the data or the labels can be suppressed.
[The ability to suppress labels on magnified plots was not supplied simply because it was felt that the need for labels was implied by the call for a magnified plot. Modifications to allow label suppression could be provided readily.]

4) Color may be used to distinguish data types plotted in combination.
[The ability to use color to distinguish data types is accomplished by changing color pens at the conclusion of each data file.]

5) Magnification of plot is allowed.
[Magnification was deliberately set to 3:1 for the plotter used at ETL; the degree of magnification seemed the best compromise between the detail required for editing and the time required to generate the plot. Variable magnification could be provided.]

6) Sub-regions of the original sheet can be selected for plotting.
[Data in any one of 9 pre-defined sub-regions of a map region can be plotted. The ability to plot arbitrarily sized and centered regions would be desirable for a production system; software modifications to allow such an arbitrary display (in other than array load increments) would be straightforward but would require substantial new software.]

7) After tagging commences, contours that represent ragged index contours can be displayed using a specified color while each of the non-index tagged contours 1,2,3, ... steps higher than the tagged index contours can be assigned different colors. Non-tagged contours can be plotted with yet another color.
[The ability to use color to distinguish elevation bands is accomplished by changing color pens at the conclusion of the plot of each data file.]

Man·· edit capabilities inherent in the file structures were not implemented simply because it became apparent that such capabilities could not be usefully employed using a plotter. Plot generation time was a deterrent to plotting data in other than standard formats. The deterrent can be eliminated with an on-line interactive display system, and so access to the capabilities would then be meaningful and could be tapped.

To aid in the interpretation of the displays, various types of lists can be or are provided. They include:

1) a list of user parameters*,
2) feature vector lists that consist of master vector ID's with index/non-index designators, and
3) lists of vectors that require editing; plot vector ID's with associated end-points (x,y)'s compose the list.

Edit command capability includes that for:

1) creating vectors between end points,
2) deleting vectors,
3) re-ordering the vectors of a "cut" or "fill" list, and
4) swapping (reclassifying) the vectors of the features.

Punched cards provide the mechanism by which an operator enters commands to the CDC. An implicit feature of the system is that the file structures for display/command activity can readily be used to support a "fast" display/command hardware set.

*A complete list of user and system parameters can be generated at various points along the processing path by using CDC utility subroutines.

4.8 Software Features/Characteristics (Tagging)

After the integrity of the contour vectors is established, tagging commences.

The GAC tagging procedure provides the following features:

- hi-speed tagging,
- convenient manual priming and editing
- potential for substantially reducing manual intervention requirements, and
- rigorous self-verification procedures for automatic quality control.

Goodyear Aerospace's tagging algorithm is based on a simple principle. If a line - a "cutting" line - is drawn through a set of closed contours*, all contours that intersect the line can be tagged uniquely provided that at least one contour that passes through the line is already tagged.

The tagging algorithm accomplishes tagging in two phases. In the first phase, contours that intersect the map sheet neat lines - boundary contours - are tagged *semi-automatically*. Manual support is required in the first phase in order to "prime" the operation (provide certain tagged boundary contours for references) and to tag contours that are inherently untaggable (if adjacent panel information is not used). At the conclusion of phase 1, all boundary contours are tagged and all such contours are artifically closed outside of the periphery of the neat lines. In the second phase of tagging, contours internal to the sheet - closed contours - are tagged fully automatically by using a regularly spaced set of lines that force intersections with the internal contours. At the conclusion of the second phase of tagging, both boundary and closed contours have been tagged. The present pre-production

*Boundary contours of a region are artifically but automatically closed prior to the closed contour phase of the tagging process. At this time all contours are endowed with the equivalent of a correctly marked depression flag.

software does not tag very small contours that are missed by the cutting lines. Such contours can be tagged using spur paths off the main cutting line that extend to the starting point of the vectors. Software development to implement such paths was not completed during the course of the program.

The complexity of the software to tag the very small contours is greatly increased due to the multi-file set-up currently available. The direction of future effort in this area depends largely on whether or not ETL and DMA concur with GAC's solution to the raster scanner problem.

By splitting the tagging operation into two phases, GAC's algorithm can capitalize on both an operator's familiarity with present DMA neat line editing procedures and the data gathered prior to the start of the tagging process. In the first phase of tagging, the "priming" phase, man-intensive activity is directed to what goes on along the North, South, East, and West neat lines - the sheet boundary line. Since an operator's eyes can re-train most readily to items along distinct straight edges (after diverting them to see information associated with neighboring locations), the map boundary line is an optimal line along which elevation priming information can be acquired manually. In the present job flow, boundary lines are already required to be treated substantially (to support mosaicking/paneling tasks, etc.) and so, when adding tagging to the job flow, "priming" and mosaicking tasks could be co-serviced*. Finally, at the moment the tagging of contours along the boundary is complete and correct, all subsequent tagging operations associated with tagging contours internal to the sheet become entirely deterministic, regardless of how convoluted and/or meandering a contour line may be. Thus, by providing CONTAGRID software with edit capability that can be utilized before the start of closed contour tagging, the problem of tagging ambiguity is confined to a boundary line type problem.

*In a production environment, these processes could be serviced automatically along neat lines that are adjacent to or overlap sheet regions that have been processed previously.

The automatic tagging software develcped by GAC provides the particularly attractive feature of automatic self-verification. The verification mechanism employs a 2-step strategy that: 1) checks to determine that elevations along the neat line borders of the map are mutually consistent, and 2) afterward checks to determine that elevation assignments of contours internal to the boundaries are consistent.

By applying the automatic consistency checks in two steps, one can first assure that all contours that cross the neat lines are properly tagged before proceeding to perform the CPU expensive second phase tagging operations; therefore, CPU time required for re-work is minimized.

The self-verification tests have proven to be exceptionally powerful at catching errors that are inadvertently introduced either at the time that the boundary elevation list is assembled or at the time that vector editing operations are performed. (This was dramatically illustrated during GAC's initial effort to tag the SHIRAZ test region. Four successive trials were required before the verification software allowed the elevation list for boundary contours to be used for tagging contour intervals to the sheet boundary. After each trial, the detected error was manually corrected and the elevation list numbers were manually reviewed and considered correct. Three times the checker demonstrated the shortcomings of manual quality control).

Note that when an error is caught, it is identified. To fix the **problem,** the operator back-tracks to the previous edit phase and corrects the problem more quickly because of the clues provided by the consistency checks. In general, capability to back-track*

*The fact that back-tracking is cumbersome when working with the present pre-production software should not mask the essential fact that the file structures support such action. In a production system, back-tracking would be under the control of the command interpretation software that supports the on-line interactive edit station. Commands would necessarily have to be simple and concise in character so that editor efforts would be directed to cartographic operations rather than computer manipulation.

*See Footnote on page 4-33.

small increments in the job stream to make fixes significantly simplified GAC's debug efforts; incremental edit capability would similarly conserve CPU and human resources in a production environment provided that on-line interactive edit hardware were used.

The tagging software was initially exercised with artifical data that would correspond to that of full-sized medium-density sheets in the late spring of 1979. Since then, real data, namely the SHIRAZ Test Region* data, has been used to exercise the tagging software. To "prime" the process, an ordered elevation list for index boundary contours was developed along the neat lines. All non-index boundary contours that were not "cups" were automatically tagged. The boundary contour cups were manually tagged. Subsequently, the closed contour tagging operation located all but the very small closed contours that missed the cutting lines. These very small contours were manually tagged. (In a production system, the very small contours would be tagged automatically.)

It should be noted that the nominal 4"x4" SHIRAZ test region exhibits a distorted proportion of boundary contours to closed contours. In particular, the distribution of contours follows:

Index boundary contours	15
Non-index cup boundary contours	26
Non-index, non-cup boundary contours	33
Closed large contours	15
Closed very small contours	6
Total	96

*See Footnote on page 4-33.

The atypical distribution of contour types shown for the small test region was somewhat unexpected. In retrospect, at least the high ratio of boundary-to-closed contours could have been predicted. When the boundary of a region becomes small relative to the mean spacing between contours, and/or the mean spacing between spot elevations, the ratio of boundary to closed contours necessarily gets large.

No flaws in the tagging algorithm or software have been discovered. Goodyear Aerospace does not expect any software problems with tagging modules because the tagging strategy depends on simple logical rules. Nevertheless, to get performance statistics for a variety of sheet types, it would be desirable to exercise the tagging software with a large and diverse set of input data.

The tagging software not only eliminates the primary requirement for highly man-intensive manual tagging, it also simplifies the demanding secondary problem of manually verifying the correctness of the tagging process.

4.9 Software Characteristics/Features (Gridding)

The gridding algorithm developed by GAC is an off-shoot of DMA's Planar Interpolation technique. It can be efficiently implemented using a parallel processor.

At present, with the implemented algorithm, gains in elevation accuracy should be achieved because more independent points of known elevation in the proximity of a matrix point are used to compute the elevation of the point. (More importantly, the GAC routine can be modified to account for all points in all directions in the proximity of a matrix point. Thus, because estimates of first and higher derivatives of elevation gradients could be made, the potential exists for substantially reducing R/S compilation requirements.)

Goodyear Aerospace's treatment of gridding input data derived from high resolution (1 mil) raster data is particularly note worthy. In reducing the resolution to that required for the output elevation matrix grid, GAC retains high resolution information that indicates where a contour line crosses the edge of each coarse elevation matrix element (10 mil). As a result, where slopes are steep, the ability to accurately define elevation at a mesh point is still maintained. This ability is simply not available when coarse resolution contour measuring devices are used (e.g., the DGR). (See Figure 4-1.)

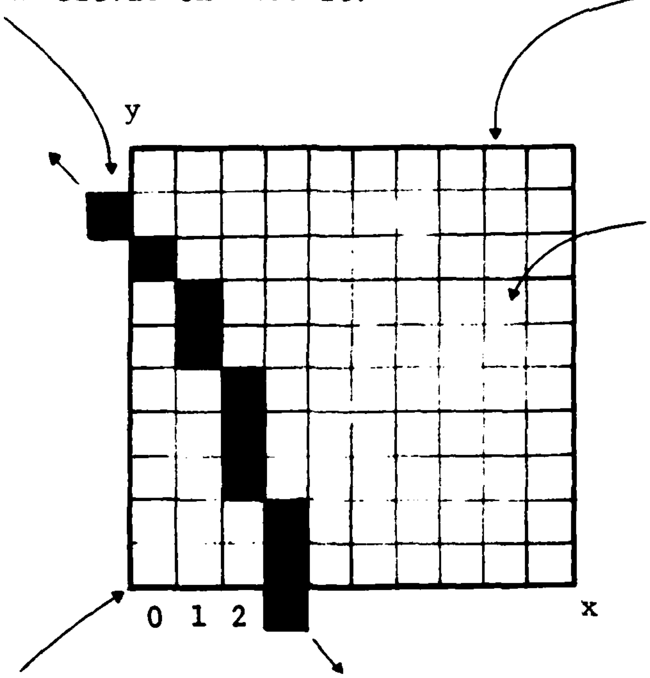
The GAC gridding algorithm has successfully generated the elevation matrix for the 3.78" by 4.35" SHIRAZ Test Region. The outlined section of Figure 4-2 displays a 1:1 copy of the contour sheet data for this region. Figure 4-3 shows a COMTAL display of the output elevation matrix data for the region that was developed using data from two different A/D* converters. The DMA HTC RAPS provided 25 micron resolution raster scanned sheet contour data. A DMA HTC DGR provided 10 mil resolution R/S, hash, spot elevation, and neat line spline data. Both the RAPS and DGR input data are merged and shown as a red line overlay in Figure 4-3. The successful merging of the unequally scaled data from the 2 distinct equipment types, the DGR and RAPS, is demonstrated by the apparently correct overlay of contours and R/S lines. Also, R/S lines, on the average, can be seen to bisect contour loops and intersect contours orthogonally. The fact that color shades or color boundaries generally lie along the contours verifies the general correctness of the gridding procedure.

Color or color/shade assignments to elevations were deliberately chosen to lie at multiples of the contour step. (In the figure, 2 contiguous color shades = 1 contour interval = 200 ft.). The intermediate color shade boundaries, which represent interpolated contours, do generally conform to the locations where they would be expected.

*A/D = analog-to-digital.

fine resolution
rasterized contour
with elevation=2400 ft.

10 mil coarse gridding
resolution element
(Heavy border)



25 micron fine raster
scanning resolution element
(Light border)

I=307
J=408

NOTE:

The elevation/location data retained for coarse grid element I=307, J=408 is

1) the elevation (2400 ft.)
2) the location of the contour point along x or y nearest the lower left (I,J) corner. Two parameters (X=3, XFLAG=1) are required to locate the point. The flag distinguishes between an x or y axis measurement.

Figure 4-1. Coarse Grid Re-Sampling

THIS PAGE LEFT
INTENTIONALLY BLANK

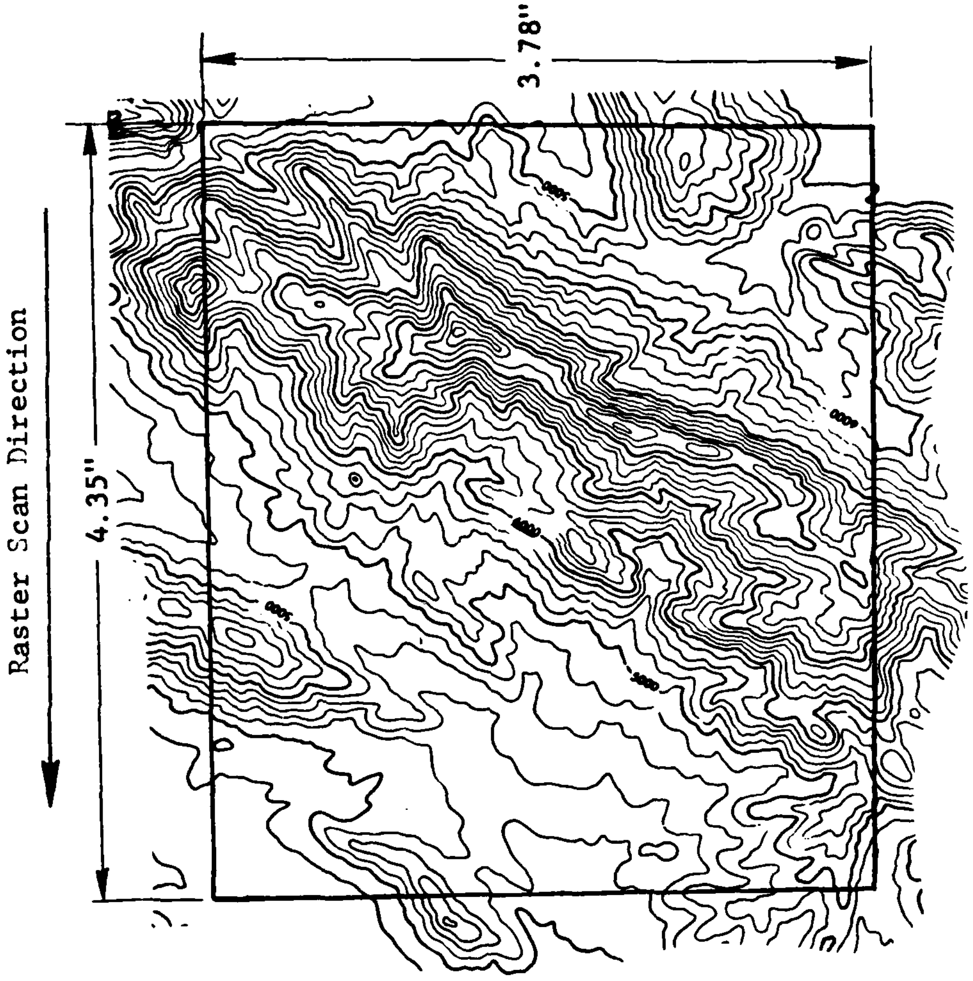
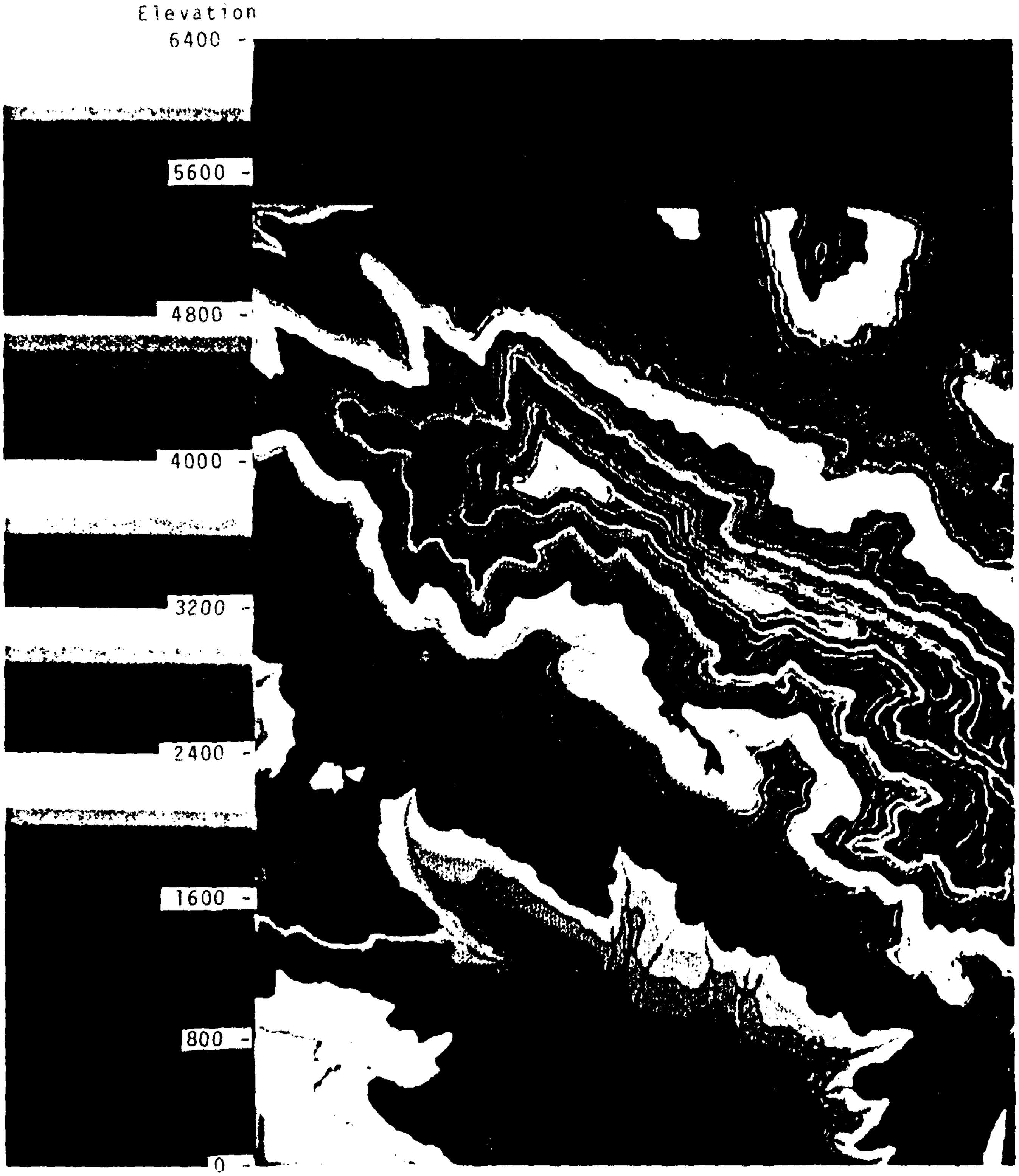


Figure 4-2. Contours of SHIRAZ Test Region



Fiqure 4-3. Elevation Matrix Data

In the matrix regions where color shade changes do not correspond to contours, more detailed examination of the data is required to determine whether or not a data or algorithm problem exists. To date, digital dumps of data in such regions have shown that very small deviations from the contour step sampling interval (less than 2 percent) or outright edit errors have caused the shade demarcation line drifts away from the contour lines. When the drift is caused by a small change that is cartographically significant, it indicates the need for supplemental contour, R/S data, or other data. Note that the color display provides highly effective cues for 1) assessing the overall correctness of the gridding operation, and for 2) spotting local regions that require more detailed inspections.

Goodyear Aerospace is currently awaiting results of ETL/DMA evaluations of the gridding results. Because the GAC gridding procedure is structured, modifications required to optimize or alter the procedure can be introduced and tested quickly.*.

4.10 TECHNICAL PROBLEM (DEGRADATION MECHANISM-IMMEDIATE EFFECT)

The impact of out-of-spec line weight variance on pre-tagging tasks is reviewed below. The pre-tagging activity is subdivided into three major operations, namely, line separation, raster-to-vector conversion, and data "cleaning." The first operation, line separation, performs an automatic raster data filtering operation that is designed to differentiate non-index from index contour lines. The end purpose of the present filter software is to make it simple to manually "prime" the automatic contour elevation tagging operation without the need for any map sheet edge coordinate data and

*After ETL/DMA evaluations are completed, modifications should be made to correct deficiencies. Two deficiencies have already been noticed. First, neatline/ridge stream line intersections aren't treated like contour/ridge stream line intersections. Secondly, one of two elevation gradient estimates made to find the elevation at a grid point is occasionally made using points that straddle a contour line. As a result, a slightly lower (or higher) grid point elevation assignment than is possible for a region is made. Fixes for both problems exist and are straightforward.

without the aid of any support equipment other than a plotter. An operator is only required to compile a list of elevations corresponding to and in the order of the intersections of index contour lines with the map sheet borders (neat lines). After compilation, the list is input into the digital processing stream via punched cards.

The present algorithm differentiates index from non-index contours by contour line weight. It assumes that line weights conform to DMA specifications. Any line section that is 8 mils in width or less is classified as "non-index"; any section greater than 8 mils in width is classified as "index". Files for "index" and "non-index" lines are created. Later, after tagging's elevation priming operation, the (x,y)'s of index contour lines that terminate on the neat lines are automatically ordered and associated with the manually developed elevation lists for index lines.

The success of the separation operation rests on the assumption that the 2.4 mil difference between minimum width index lines (10 mils, ±10 percent) and maximum width non-index lines (6 mils, ±10 percent) does not vanish. The assumption was tested and verified for small test regions using the ETL IBM scanner in a laboratory environment. Yet, in testing the CONTAGRID software, it was discovered that various DMA raster scanners produced line weights that modulate well out of the allowed range for line weights. Thus, for innumerable regions of a map sheet, the (non-index) index lines exhibit line weights with the specification for (index) non-index lines. The line separation filter software, acting on the basis of line weight data, chops up a particular type contour of a given type accordingly. The various pieces are filed into their appropriate files. That, in turn, creates innumerable vectors of a given type that terminate internal to the sheet. Whether or not the automatic editing features of the raster domain software relink the intermittent contour segments of a given file, enormous edit loads are created because edit identity numbers on output plots (used for manual editing or auto-edit verifications) are not only prolific, but tend to overlay each other.

4.11 TECHNICAL PROBLEM (SOLUTION OPTIONS)

It is not known whether the ETL IBM scanner would create the kinds of problems experienced with DMA scanners in a production environment. Because the IBM scanner was out of commission during the course of the CONTAGRID project, no tests were able to be conducted to clarify this issue. In any case, it is necessary to deal with the realities of existing DMA scanners. Goodyear Aerospace considered the following options:

1) by-pass the line separation process and enter the tagging task with no distinction between index and non-index contour vectors,
2) alter the line separation algorithm so that the index/non-index label is attached to a contour on a statistical basis rather than on an exact measure basis, or
3) require new DMA standards for non-index and index contour line weights that make line separation possible.

Option 3 is clearly unrealistic. Re-compilation of the contour data of existing contour sheets would require an exorbitant amount of manpower; less dollar and time costly alternatives exist.

Option 2 appears attractive except that it was noted that at least for the RAPS, average line weights for index lines varied as much as 2 or 3 mils over the sheet. The software needed to support both the adaptive measure of average line weight and that measure required to make the statistical decision of index or non-index is moderately complex and tends to be processing expensive even for STARAN. Moreover, since manuscripts do often contain ink or pencil drawn corrections of contour breaks (e.g., when map data have been mosaicked), it is quite likely that contours would often be put into the "I-don't-know" category, a category that would still require manual editing.

At the present time, GAC considers option 1 the most desirable. At first it appears that it is less attractive because it requires the existence of a digitizing table or an interactive raster edit station within the cartographic system. But, an interactive display station is already needed for a variety of editing and quality control tasks; the priming operation would simply make use of a hardware resource that already existed in the system. With such hardware, cues not available with the present plotter procedure could be employed to accelerate the manual priming process; the operator can be given virtually instant access to specific information needed to support each step of the priming process.

The operational benefits of an option 1 procedure follow:

1. Because index line data are not separated from non-index line date, the raster-to-vector conversion process needs to be carried out only once for a contour map sheet. The time to convert from raster-to-vector is essentially halved.

2. When no data separation is performed, no line break errors are introduced into the data. Thus, auto-edit procedures that are only 80% effective can still reduce the requirement for manual editing by a factor approaching 5:1 . (In contrast, if line separation procedures introduced 20 fold as many errors as existed on the original sheet and auto-editing were only 80% effective, the net editing load imposed on an operator would be about four times that faced on the original sheet. In such a case, automation would increase the editing burden rather than decrease it!).

3. Alternate methods of priming the elevation tagging operation can be accommodated. As an example, as an alternative to GAC's elevation "priming" list, L. Beabout of DMA HTC has suggested that the elevation priming list should consist of an ordered list of the locations of relative high and low points along the neat lines. A procedure using such a list has been examined to determine how well it can be fitted into GAC's tagging software set; it will mesh cleanly into the present software structure.

4. Vector data from the AGDS could be accommodated by the procedure.

While the CONTAGRID program itself could ill afford the time and manpower required to track down the source of the pre-tagging software's problems, GAC's preliminary findings that "healthy" raster scanners introduce substantial line weight variances should benefit future programs. A disciplined study to characterize the data degradation that is caused by raster scanners under different operating conditions is necessary. Scanner characteristics can have a dramatic impact on software character and cost.

4.12 TECHNICAL PROBLEM (RIPPLE EFFECT)

With existing pre-tagging software, after the line separation task is completed, a STARAN raster-to-vector conversion process converts 1) the raster data that is classified as index contour line data into a file of index vectors and 2) the raster data that is classified as non-index contour line data into a file of non-index vectors. The conversion process works effectively but has not been optimized. The raster-to-vector conversion software is highly tolerant of line weight variations, and so, further data degradation is curtailed by the module. Nevertheless, the module relays the degraded data and thereby sets the stage for severe problems in the pre-tagging "cleaning" operations that follow.

The CONTAGRID pre-tagging "cleaning" software has one overriding purpose, namely, to massage the vector data base until the fundamental rule for tagging - THOU SHALT HAVE NO CONTOUR BREAK ANYWHERE INTERNAL TO THE NEAT LINE BOUNDARIES - holds true. Two categories of "cleaning" software exist:

1) software that is used to establish information needed to characterize the cartographic elements (vectors), and
2) software that supports or causes the automatic or manual modifications of the vector data base.

The first category of software detects, locates, labels, and classifies the vector data that require edit action. The second category of software deletes or adds vectors to the data base, it is involved in the "fixing" (or editing) process.

In all cases, the first category of software correctly performed the detection, location, and labeling function for <u>all</u> vectors of the SHIRAZ Test Region* that required editing. [It is crucial to note that only these functions need be automated for the efficient utilization of an interactive edit station. When they are performed correctly, the human editor can be sequentially and systematically confronted with the edit tasks, the local region of the edit tasks can be automatically displayed, and the edit actions can be associated with vectors by label (or, implicitly, by location). (Of course, the software system could be set up to supply a running sum of tasks yet to be completed so the editor knows when editing is complete; operations management aids could also be supplied).]

The last type of first category "cleaning" software, feature classification software, classifies vectors and provides the basis for further simplifying the editing task. As was noted earlier, this software proved vulnerable to the degrading effects of line fragmentation. This software looks for features, i.e., for a particular set of across-vector associations. It measures the associations of a vector with its neighbors and determines whether the set of associations corresponds to those that define a particular feature. When vectors are derived from fragmented (or merged) input data, the relational associations of the vectors (as well as the basic vectors) that correspond to a feature are distorted according to

*SHIRAZ Test Region

Initially, the data of the nominal 20" by 20" SHIRAZ contour sheet was to be used as the test data set for evaluating the CONTAGRID algorithms and software system. When it was discovered that the badly degraded output of the pre-tagging software's line separation module was caused by fundamental limitations of the DMA raster scanner (and no digitized data that met DMA specifications could be made available), it was decided to proceed with the evaluation of the CONTAGRID software with a subset of the digitized SHIRAZ sheet data that came close to meeting the specifications. It was found that the nominal 4" by 4" region of the top left corner of the SHIRAZ sheet met the relaxed requirements; it could be treated with the existing pre-tagging software without severe contour fragmentation problems. As a result, this region, the SHIRAZ test region, has become the primary data source for the end-to-end evaluation of the CONTAGRID algorithms and software system.

the extent of data degradation. If the software sets rigid conditions for classification (in order to avoid wrongly classifying features), it is likely to be ineffective when treating degraded data. Goodyear Aerospace's present software applies weakened association conditions to establish features and thus can categorize some of the features in a "dirty" data environment. Yet, it does so at the expense of making some improper classifications.

The CONTAGRID "cleaning" software that performs classification marks vectors according to whether they belong to one of the following features: numerics, depression contours, cuts, or fills, and then the second category of "cleaning" software modifies the vector data base in a particular sequence: vectors associated with features are treated first and then all other vectors that are open are treated. Specifically:

1) tic mark vectors of depressions, cuts, and fills are deleted,
2) residual depression vectors are marked,
3) cut, fill vectors are assigned a hierarchical structure,
4) vectors identified as numerics are deleted, and
5) vectors needed to join broken contours are added to the data base.

Quite clearly, the effectiveness of the feature classification operation will directly affect how well the auto-edit modules of the "cleaning" software performs. Because only "dirty" vector data (fragmented vector data) has been used to measure the effectiveness of the feature detection/correction software to date, the degree of effectiveness improvement with "clean" vector data (that is expected with modified pre-tagging software) is not yet known, but is expected to be substantial. The basis for this projection relates to the mechanisms for mushrooming degradation which have been observed to hold for the present software. In particular,

1) the feature filtering algorithms presume relatively "clean" features; the vectors of a distorted feature have exhibited a lower probability of being correctly classified when the data base is "dirty,",

2) the auto-edit action demanded for a vector relies on its classification; improper classifications have been observed to cause improper edit actions, and

3) the total number of candidates for classification (and editing) and the number of false features, artifacts, have been observed to increase many fold in a "dirty" data base; in such cases the opportunities for improper vector classification have increased dramatically.

Consider how the auto-editing portions of the "cleaning" modules treated the SHIRAZ test region. The region contains 5 contour elevation numbers. After executing CONTAGRID's line separation and raster-to-vector conversion routines, the digits of the numbers are described as sets of many short vectors that are much different in character from those that would be obtained from clean, unfragmented raster data. This "dirty" data was subjected to CONTAGRID's automatic data "cleaning" modules.

Only about 60 percent of the vectors associated with elevation numbers were classified as having a numeric attribute; these were automatically edited (i.e., deleted). The remaining 40 percent of digit fragment vectors were improperly classified and were not automatically deleted. As a result, the automatic contour line join routine, which connects contour ends where numerics are removed, encountered conditions that precluded line join vectors from being generated in 2 out of the 5 cases where numbers were to be deleted. In general, automatic line join failure resulted either because of a failure to delete number fragments or a situation existed that could induce a false join and so an automatic join was not attempted. If automatic line joining had been executed after an editor had concluded removing unwanted number feature fragments on an interactive display, a higher effectiveness level in a relatively "dirty" data environment would have been likely even without the benefit of any algorithm enhancements.

Based on the results obtained with "dirty" data, in a clean data environment, automatic feature classification/auto-edit software can be expected to simplify an editor's role.

4.13 MANUAL EDITING AIDS

The pre-tagging "cleaning" software of CONTAGRID provides a substantial set of editing aids for "interactive" manual editing. Data file structures are set up to support such editing. Unfortunately, for the CONTAGRID program, the only display resource for editing with assured availability was a CALCOMP plotter.

As a result, for all editing operations, a plot is required. The editor defines edit actions based on the plot and the labels drawn on the plot. The display resource is entirely inadequate as the primary display for production oriented interactive editing for the following reasons.

1. It is not on-line; time lags required to produce hard-copy graphs needed for editing are unacceptably large.
2. The display can't be changed rapidly; until a plot is comnlete, it is not known what additional supplementary data or combinations of data types should be displayed (or not displayed) on the plots.
3. No quick selective display (suppression) of annotation was possible; edit labels at locations where edit demands are greatest tend to overlay and become unreadable. Alternative plotter procedures that reduced the overlay problem removed labels from the locale where editing was required and so forced much eye movement. The high demand for eye movement resulted in editing inefficiency.
4. A common coordinate system for instantly registering two versions of the global data was not available. No mechanism exists for comparing raw sheet data to the vector data in a common coordinate system and/or on a microscopic basis. Quality control judgements cannot be made effectively.

Goodyear Aerospace believes that for a production CONTAGRID system, an on-line interactive CRT edit station is essential. With such a station, the editor has the ability to freely intersperse many manual and automatic tasks without having to continually put up with the time lags related to transferring data to/from off-line display devices. For a variety of tasks, editing is facilitated by displaying raster data, vector data, combined raster/vector data, or other data in combination. It appears that only a raster CRT color display system that is supported by a multiplane memory has the versatility to support the variety of capabilities demanded by a CONTAGRID editing display; GAC believes that an on-line interactive edit station should contain a high resolution color raster display unit.

4.14 TIMING SUMMARY AND PROJECTIONS

This section discusses the significance of timing measurements made while exercising the CONTAGRID pre-production hardware/software system with real input data. It begins with a summary of the performance of the pre-production system and a projection of the performance of a production system. The development of the pre-production system's execution time summary table (see Table 4-III) is then discussed. How the pre-production system's task timing results are used to project the execution time performance of a production-oriented SIMD/sequential processor-based automated cartographic system that eliminates the hardware/software inadequacies exposed by the pre-production system is then discussed. The section concludes with a brief discussion of the impact of the program on reducing man-intensive DTED tasks.

The primary tasks of the CONTAGRID process, automatic elevation tagging and elevation gridding, are shown to be able to be executed in about 1 hour using the relatively inefficient components of the pre-production system. The inefficiency is most evident in the

TABLE 4-III - EXECUTION TIMES FOR PRE-PRODUCTION MODULES COMPRISING END-TO-END CARTOGRAPHIC SYSTEM

(SHIRAZ SHEET)

	OPERATION				TIMING (20" BY 20" SIZE)			
			MAJOR MODULES		CDC (SECS)		STARAN (SECS)	
ITEM	PHASE	ACTION	CDC	STARAN	CPU	I/O	CPU	I/O
1	Pre-Tagging	Copy RLC tape(s) to disc	TAPCOPY	-	165	1,535	-	-
2		Merge odd/even files; input ancillary data; get statistics of RLC records	TAPEIN	-	130	100	-	-
3	" "	Separate Index from non-Index lines; auto-edit; thin; convert raster-to-vector, etc	GETVECS	SEPVEC	155	8,600	2,010	6,400
4	" "	Build master vectors using array load correlation lists.	GETMVECS	-	140	225	-	-
5	" "	Detect, locate, classify, faulty vectors or features and auto edit them.	GETEDS	EDTMGR	25	175	100	60
6	" "	Enter card corrections; create edit vectors, format entered vectors and file.	VALDVEC	VLDJOIN	60	100	40	10
7	" "	Generate proof plot of cleaned-up vector data.	PLOTGEN	-	3,575	6,125	-	-
8	" "	Build contour vectors from original master vectors and edit vectors.	SQUEEZE	-	125	625	-	-
				TOTAL	4,375	17,485	2,150	6,470
9	Tagging	Correlate "Prime" values to boundary contour ends and tag boundary contours	STATOPN	OPNTAG	8	100	10	50
10	"	Enter manual tags into files. (Also, after process below)	EDITAGS	-	2	5	-	-

Pre-Tagging TIME ≈ 5 Hrs.

TABLE 4-III - (Continued)

OPERATION					TIMING (20" BY 20" SIZE)			
			MAJOR MODULES		CDC (SECS)		STARAN (SECS)	
ITEM	PHASE	ACTION	CDC	STARAN	CPU	I/O	CPU	I/O
11	Tagging	Auto verify boundary contour tagging;tag internal contours and verify internal tags.	STATCLO	CLOTAG	12	120	60	55
12	"	Output DGR-like tape of contour vectors.	TAPEOUT	-	800	400	-	-
				TOTAL	822	660	70	1(
13	Gridding	Resample contour data and change coordinate systems.	RESAMPLE	RESAMP	70	160	45	80
14	"	Segregate DGR data into 4 file types	SEGDGR	-	85	95	-	-
15	"	Compute elevations along neat line.	NEAT	-	12	7	-	-
16	"	"Regionalize" DGR R/S data	PARTSR	-	115	285	-	-
17	"	Find R/S intersections with points of known elevation.	FINDINT	FINDIN	90	315	20	180
18	"	Compute elevations along R/S line.	BILDSRP	-	310	355	-	-
19	"	Merge files of Known elevation; reformat data.	MERGBAK	-	400	250	-	-
20	"	Populate elevation matrix	GRID	DAPGAC	260	155	150	80
				TOTAL	1,342	1,622	215	340

Tagging Time
≈1/2 Hr.

Gridding Time
≈1/2 Hr.

timing results of the pre-tagging tasks; on the order of 5 hours is required for their execution. The excessive execution time for this phase can be primarily attributed to the sequential processor's poor reformat capability (see Item 7, Table 4-III) and, in combination, to inadequate SIMD processor storage capacity and to a low SIMD to sequential processor channel data bandwidth (see Item 3, Table 4-III). With a correction of the hardware/software limitations of the pre-production system and the use of an on-line interactive display rather than plots to support editing, GAC extrapolates an almost 20 fold improvement in the execution time required for executing the automatic pre-tagging phase tasks. (This projection is supported by experimental results acquired using STRAPS software that was developed outside of the CONTAGRID program.)

The correction of pre-production system deficiencies extrapolates to a smaller 4-fold performance gain for the elevation tagging and elevation gridding tasks of the CONTAGRID process. Gains would largely result from assigning sort/merge and reformat tasks that are now performed by the sequential processor to the SIMD processor; by eliminating the data channel bottleneck between processors, task re-assignment gains would not be erased by data moves.

In total, based on the measures of execution time for the various CONTAGRID processes, GAC projects that an automated end-to-end production cartographic processing system based on a joint SIMD/ sequential processor set could perform the automatic segments of the pre-tagging, tagging, and gridding tasks in about 1/2 hour. The production-oriented system would require a SIMD with significantly more memory space (e.g., a GAC STARAN-E), a smaller but more I/O efficient sequential processor (e.g., a SEL 32/75 mini), and on-line, interactive multidata form edit/quality control hardware set.

Execution time measurements were made using the SHIRAZ map sheet data and sub-regions of such data. Most often, the SHIRAZ Test Region was used. (Tables 4-IV and 4-V list subregion timing results.)

TABLE 4-IV. PRE-TAGGING AND TAGGING EXECUTION TIMES

MODULE	DESCRIPTION	MACHINES UTILIZED		TIMINGS FROM CDC-DAYFILES		COMMENTS
		CDC 6400	STARAN S-1000	CP EXECU. TIME(SECS)	I/O (SECS)	
TAPEIN	MERGED DMA RLC FILES ON DISK	X		26.1	20.30	FOR 20" DEEP X 4" WIDE SHEET
GETVECS	CONVERTS RASTER-VECTOR	X	X	18.31	512.00	INCLUDES CDC/STARAN I/O FOR(20x4") SHEET
GETMVECS	BUILDS MASTER VECTORS	X		5.51	17.80	
GETEDS	EDITS VECTORS AUTOM'Y & DEFINES SPECIAL SYMBOLS	X	X	1.00	14.00	
VALDVEC	PERFORMS MANUAL EDITING CLEANUP	X	X	2.26	21.93	
PLOTGEN	GENERATES PLOTS OF EDITED DATA	X		143.1	245.00	TIMES WILL VARY WITH PLOTS REQUESTED
SQUEEZE	GENERATES CONTOUR VECTORS FROM MASTER VECTORS	X		4.95	25.00	
STATOPN	TAGS OPEN (BOUNDARY) CONTOUR VECTORS	X	X	0.54	6.25	
EDITAGS	EDITS OPEN VECTORS	X		0.87	1.60	
STATCLO	TAGS CLOSED CONTOUR VECTORS	X	X	4.53	9.66	
EDITAGS	EDITS CLOSED VECTORS	X		0.86	1.60	
TAPEOUT	GENERATES DGR FORMATTED TAPE	X		31.7	15.76	
PLOTAGS	PLOTS TAGGED C. VECTORS BY ELEVATION	X		70.9	113.70	

TABLE 4-V. CONTOUR GRIDDING EXECUTION TIMES

MODULE	DESCRIPTION	MACHINES UTILIZED		TIMINGS FROM CDC-DAYFILES		COMMENTS
		CDC 6400	STARAN S-1000	CP EXECU. TIME(SECS)	I/O (SECS)	
RESAMPLE	REDUCES VECTOR DATA TO 10-MIL GRID	X	X	7.19	15.32	ABOUT A 10:1 INCREASE IN I/O FOR FULL SHEET STARAN TIME=3.6 SECS*
SEGDGR	SEGREGATES DGR DATA INTO S/R POINTS, SPOT ELEV'S, S/R JUNCTIONS, AND NEAT LINE POINTS	X		3.35	3.78	
NEAT	BUILDS NEAT LINES FROM NEAT LINE POINTS USING BI-LINEAR INTERPOLATION	X		2.30	1.37	
PARTSR	PARTITIONS S/R LINES INTO SUB-GROUPS FOR STARAN PROCESSING	X		4.50	11.28	
FINDINT	FINDS INTERSECTIONS OF S/R LINES WITH POINTS OF KNOWN ELEVATION	X	X	3.42	12.57	STARAN TIME 0.65 SECS *
BILDSRP	BUILDS STARAN RIDGE POINTS USING LINEAR INT'N	X		12.36	14.15	
MERGBAK	MERGES FILES AND BACK-SCANS FOR INPUT TO GRIDDING	X		25.29	9.68	MERGE ACCOUNTS FOR ABOUT 85% OF TIME
GRID	PERFORMS GRIDDING	X	X	10.25	30.40	APPROX. 5:1 INCREASE IN I/O FOR A FULL SHEET - STARAN TIME 10 SECS *
TOTAL				69.66	98.55	

*STARAN TIME INCLUDED IN CDC I/O + EXECUTION TIMES

To establish full sheet timing results, the results for sub-regions were linearly extrapolated to the full sheet size (20" by 20"). The linear extrapolations tend to be conservative since the SHIRAZ test region contained more line information/square inch than the sheet's average line information/sq. in. All the principal path CONTAGRID processes are included in the timing measurements.

The raster data was sampled at a 25 micron (≈.001") interval; this resolution was retained up to and through the tagging operation. At the conclusion of tagging (including final editing), resampling was performed at a 10 mil resolution level in order to produce a contour vector tape like that produced by the DGR. Also, within the gridding process and just prior to merging DGR data with that from the raster scanner, the tagged contour vector data was resampled at the resolution of the DGR (but as was noted earlier, fine resolution information about the fractional point of line intersection with the coarse matrix resolution element is retained during resampling). All automatic gridding operations after resampling are performed using data having 10 mil resolution.

A production system design should address the impact of resolution on processing time. When the DGR is used to digitize the SHIRAZ test region, it comprises 164,440 resolvable pixels. When digitizing the same region with the RAPS, it comprises ≈100 times as many resolvable pixels (16,957,440). Processing execution time is nearly proportional to the number of pixels, and so processing time increases as the square of the linear resolution. By decreasing the resolution as soon as possible along the processing chain that leads to the elevation matrix, the processing time can be reduced. (GAC's 1/2 hr projection for executing the automatic pre-tagging, tagging, and gridding tasks assumes no such time savings are made).

To time the various processes, the CDC task monitor was used. It supplies the CDC 6400 CPU and CDC 6400 I/O times for each job submitted to the CDC in printed form on each batch job summary. The corresponding STARAN tasks are timed using STARAN's 100 nanosec resolution timer. Timing results are obtained in the form of keyboard printouts.

Most often, for the CONTAGRID modules, the CDC 6400 I/O and CPU times overlap. The CDC I/O time includes not only the time that data transfers take place between the CDC peripherals (primarily discs and tape) and CDC memory but, also, the time that the I/O channel between CDC memory and STARAN is enabled. (A number of CDC system task times are also bundled into the I/O time.) As a result, STARAN CPU time and STARAN I/O time (to the CDC 6400 or to the STARAN-associated RK05 disc) falls within the CDC 6400 I/O time period.

As programmed, STARAN I/O time to the RK05 disc is not concurrent with STARAN CPU time. On the other hand, when I/O to the CDC takes place, some overlap of STARAN CPU and I/O time can occur.

Table (4-III) summarizes the timing results for the CONTAGRID tasks when digitized SHIRAZ sheet data is used as input. The CONTAGRID tasks are listed in the order that they are executed. A cursory examination of the Table (4-III) suggests that the CDC 6400 operations, both CPU and I/O, dominate the approximately 6 hours of ETL CDC 6400/STARAN system use needed to transform the SHIRAZ digitized map sheet overlay data to the digital terrain elevation matrix data form. Moreover, the pre-tagging phase of the transformation procedure demands the bulk of system usage time (≈5 hours). GAC considers the 5 hours of system use time for pre-tagging unacceptably large for a production system. The large system use time results because of (1) an imbalance of system hardware resources in the ETL system and because of (2) the heavy reliance on the CDC processor for performing the non-critical CONTAGRID tasks. How the various items of the CONTAGRID pre-tagging operation would be handled in a production system is considered next.

Items 1 and 2 -

A CDC 6400 system software package is used to move RAPS RLC raster data from tape to disc. No data format conversions are required and so virtually no CPU time should be required and yet 165 seconds is required. Why? Approximately 20K records (i.e., one per scan line) of about 500 bytes/record are moved. The CDC is apparently charging about 4 millisecond to set up each "read" and "write" of a record. It may be noted that this time is in the ballpark of the time a PDP-11 mini takes to set up a read or write "TRAN" (1.4 millisecond).

Since every record read from CDC tape is written to CDC disc, a total of about 20 megabytes of data are moved between CDC memory and the tape or disc peripherals. This implies an average data move rate of only about 12.5K bytes/second. Since the inherent move rate of the disc and tape is much greater than 12.5K bytes/second (approximately 800K bytes/second and 100K bytes/second, respectively),it is apparent that disc and tape latency is badly degrading the inherent move capabilities of the tape and disc peripherals. It should be noted that the cure to the long I/O and CPU times is to include more records/scan into any given second. (A faster peripheral is not the cure.) The ETL IBM and SCI-TEX raster scanners tend to do this; the AGDS and RAPS records tend to be short. It should be noted that the SCI-TEX and AGDS require about twice as many bytes/scan as the RAPS when sheets having over 3000" of line/(20" by 20") sheet are scanned. It would be desirable to use SCI-TEX record sizes with RAPS type RLC when scanning black/white graphic source material. If this were done, using any number of available minicomputers, Item (1) CPU and I/O times of 8 and 80 seconds, respectively, could be expected. The above times would hold whether the source data comes from tape or directly from the raster scanner. In the latter case, wall clock time would increase.

In a production system, the bulk of the Item 2 task (that of merging files) need not be performed apart from the Item 1 task. Only about 2 seconds of I/O time need be allocated for it. For CONTAGRID, the number of bytes/scan was determined for each scan in order to establish maximum CDC to STARAN block transfer sizes. Such a measure is not required for a production system, but a statistics gathering capability would likely be provided for other reasons. For the same type of CPU activity as was required for CONTAGRID, a production system should allot 5 seconds. Thus, the combined Item 1 and Item 2 production system usage time would be:

HOST CPU time	= 13 seconds
HOST I/O time	= 82 seconds
STARAN CPU time	= 0
STARAN I/O time	= 0 .

Item 3 -

Item 3 processing accounts for the largest component of the time needed to produce the elevation matrix data. The bulk of the Item 3 time can be directly traced to I/O that takes place between STARAN bulk core memory and the RK05 disc that is associated with STARAN's embedded PDP-11 (6400 seconds). Most of the remaining time can be traced to STARAN processing time. As will be shown below, virtually all the I/O time could be eliminated in the production system; the STARAN processing time could be reduced to about 1/4 of that now shown.

The Item 3) operation involves the following processes:

a) Conversion of RLC raster to binary raster
b) "Regionalization" of the binary raster data
c) Separation of index and non-index line raster data
d) Thinning of lines to a one cell center line
e) Removal of line stubs and other auto-edit activity
f) Conversion of centerline data from raster-to-vector
g) Support processing

The process a) above accounts for nearly 1/2 of the Item 3) STARAN processing time, i.e., it requires about 1000 seconds. The process converts about 10 megabytes of RAPS RLC raster into about 400 million one bit pixels. A very simple algorithm was used to make the RLC to binary raster conversion; it did not exploit STARAN's parallel processing capability. Newer algorithms that do exploit the parallel capabilities of the STARAN demonstrate that a production system could accomplish the process in less than 10 seconds. (The GAC routine that converts AGDS formatted RLC to RAPS form RLC uses parallel processing techniques like those required by the production system). About 110 seconds of time are required for the STARAN to CDC I/O. As will be shown below, this I/O can be reduced to well below 10 seconds.

The initial form of the binary data is in the form of scan lines consisting of about 20K pixels. Process b) "regionalizes" this data. It develops patches of data by collecting individual scans until a 160 scan wide strip of data is developed; then the strips are equivalently cut into rectangular chips that are 184 pixels long. The patches of data that result, called "array loads," are nearly square (i.e., they are 160 scan lines wide by 184 pixels deep). Unfortunately, the older STARAN B arrays are not able to contain enough data to allow the "regionalizing" operation to take place internal to the arrays; thus, time expensive move operations are required. (The problem of small array storage space was recognized before the start of the CONTAGRID program and was one of the major reasons for the development of the STARAN E machine. Each array of the newer machine contains 36 times the array memory of the older ETL STARAN). The RK05 disc associated with the STARAN processor was used as the array memory extender for the Item 3 processes. Of course, when the disc is used as array memory, the effective memory to STARAN array Processing Unit bandwidth is only about 15 kilo bytes/second compared to the actual array memory-to-STARAN array processing unit bandwidth of about 750 Megabytes/second. Clearly, operations occur orders of magnitude more slowly. Rather than including the time during which the RK05 disc is used as an array extender as part of the STARAN CPU time, it was categorized as STARAN I/O time. It accounts for all but about 150 seconds of STARAN-to-CDC I/O time.

With the newer STARAN, the 6250 seconds of I/O time used in emulating arrays with the RK05 disc would vanish; sufficient array space exists. Thus, the I/O time required for Item 3 would drop to about 150 seconds. In this remaining 150 seconds, only on the order of 18 Megabytes of data is transferred between STARAN and the CDC 6400. This implies an average transfer rate between the processors of about 120K bytes/second. Since the CDC-to-STARAN channel can sustain data transfers at a maximum rate of 250K bytes/second, it is clear that the drop in transfer efficiency relates to the time needed to set up data transfers between CDC and STARAN. By transmitting 32 scans at a time to the STARAN rather than just 1 at a time, the average transfer rate would begin to approach the 250K bytes/second limit of the channel. The total I/O time for Item 3 would then drop to about 75 seconds. In a production system, a much wider bandwidth between the sequential processor and SIMD processor would be employed. Using a SEL mini and the newer STARAN E, a 26 megabyte/second channel could be implemented using standard hardware. In such case, host-to-SIMD I/O time would be determined by transfer set-up time. If blocks of data were 20K bytes large (as were employed by STARAN for the Large Area Crop Inventory Experiment (LACIE)) and SEL mini standard I/O driver 800 microsecond set up times are used, the total host-to-SIMD I/O time for Item (3) would drop to less than 10 seconds.

A similar amount of set up time would be required by the production system to move data between memory and disc mass storage. Actual I/O time between disc and host memory, assuming a 1.2 Megabyte/second disc drive with average disc head latency of 40 milliseconds, would be about 60 seconds.

Item (3) processes c) through g) demand about 1,040 seconds. In particular, process c) presently demands about 165 seconds. Presuming that no line separation operation is carried out by the production system, this processing time vanishes.

Process d), center line thinning, is performed independently for both index raster data and for non-index raster data and accounts for about 395 seconds of the present STARAN CPU time. By eliminating the requirement for a separate file structure for index and non-index data, the process needs to be executed only once. The processing time would not quite be halved because more steps are required to thin thicker lines than thinner lines. Nevertheless, the processing time would be reduced to about 250 seconds.

Process e), line stub removal or clipping, requires about 185 seconds. The elimination of the two file data structure will not directly affect this operation, because the time of execution is directly related to the total number of open-ended lines found after the center line thinning operation. Indirectly, the elimination of two file data structures will reduce the time required for this operation. By omitting the line separation process, fragmentation of the data will be reduced substantially so far fewer open-ended lines will exist in the data base. Conservatively, assume the reduction to be about 25%. Then, the time required for this process in a production system would hover near 145 seconds.

Process f), the raster-to-vector process, requires about 115 seconds. Once again, the elimination of line separation will have only an indirect influence on this time. Because fewer line segments would be created, fewer vector headers would need to be created. As a result, a production system could expect this process to be performed in about 100 seconds.

Process g), support processing, combines many of the processes used in conjunction with those listed above. It includes such processes as line junction encoding, across patch ("array load") vector correlation list generation, auto-editing operations, etc. In total, the operations account for about 180 seconds of the processing time. The removal of the line separation operation will reduce the time for support processing only in an indirect manner; a production system could expect the processing time for this operation to be reduced to about 165 seconds. In summary, for a production system, Item (3) operation time would be:

STARAN CPU time	=	670 seconds
STARAN I/O time	=	10 seconds
HOST CPU time	=	10 seconds
HOST I/O time	=	60 seconds .

Item 4 -

The Item 4) process chains the ends of vectors of the different patches, i.e., "array" (load) vectors, together until an open-end is found or the chain closes on itself. All array vectors of a given chain are assigned a master vector ID. On the order of 100,000 "array" vectors could be expected for a map like the SHIRAZ sheet; the average "array" vector size is about 80 mils. The chaining operation generally produces less than 1,000 master vectors (unless severe data fragmentation occurs!); the average master vector length would be on the order of 5-10 inches long. array vector headers consist of about 24 bytes so less than 3 megabytes of essential data is involved in this process.

This operation would be performed within STARAN as part of the Item 3 operation in a production system that used the large array memory of the new STARAN E. It would be able to be accomplished with less than 5 seconds of STARAN CPU time and with no I/O penalty. Quite simply, the master vector ID would simply be assigned to its slot in the array vector headers, as the headers are generated by the Item 3 process.

Since the ETL STARAN B has so little array memory space, for the pre-production CONTAGRID system, it was necessary to move data to the CDC prior to performing this operation. Once there, the data was stored on CDC disc (Item 3 output I/O). At design time, it was felt that since the process did not involve much data, it should be performed by the CDC. The relatively large CDC CPU time experienced was not expected. Earlier discussion indicated that considerable CDC CPU time is associated with setting up I/O transfers. About 1/2 of the 140 seconds of CDC CPU time can be attributed to set up; nevertheless, it is estimated that up to 70 seconds of CDC CPU time was used to perform the process. In summary, for a production system, the Item (4) system usage would be:

STARAN CPU time	=	5 seconds
STARAN I/O time	=	(consolidated with Item 3 I/O)
HOST CPU time	=	(consolidated with Item 3 time)
HOST I/O time	=	(consolidated with Item 3 time) .

Item 5 -

The Item (5) CDC I/O is accounted for by program loading, file purging, cataloging tasks as well as the CDC-to-STARAN attachment time. True host-to-STARAN I/O in a production system would require less than 2 seconds; disc-to-host I/O would require about 40 seconds. In a production system, host CPU time would be unchanged since the host acts primarily as an I/O handler for this operation. The STARAN CPU time would drop to about 1/5 of that shown because it would be executed out of high speed page memory rather than bulk core. On the other hand, GAC intends to boost the efficiency of its vector auto-edit software. About 3 times as much processing power is believed required. Thus, for a production system, the Item (5) system usage would be:

STARAN CPU time	=	60 seconds
STARAN I/O time	=	2 seconds
HOST CPU time	=	25 seconds
HOST I/O time	=	40 seconds

Item 6 -

The Item (6) CDC I/O, unlike Item (5) CDC I/O is dominated by CDC system processes, namely, by program loading, program overlaying, file purging, and file cataloging. STARAN-to-CDC attachment time represents only a small fraction of the CDC I/O time. With a production system, host I/O time would be reduced to about 20 seconds. STARAN I/O time would be less than 2 seconds.

The STARAN processing is presently being performed out of STARAN slow memory (1.1 sec) rather than the fast memory (.125 second); and, again, a real gain of 5:1 in processing speed would be obtained in a production system. It should be noted that a production system would not require so much editing (because the line separation procedure would be omitted), and so, 6 second processing could be realized even without changing the algorithm for creating the edit vector increments. Host usage in a production system would be about 2/3 of that of the CDC. Summarizing, Item (6) operations with a production system would be:

STARAN CPU time	=	6 seconds
STARAN I/O time	=	2 seconds
HOST CPU time	=	60 seconds
HOST I/O time	=	20 seconds .

Item 7 -

Item (7) processing causes an enormously large system load. No STARAN activity is involved at present. The processing simply involves preparing vector data for the CALCOMP plotter. Reprogramming could reduce both the CPU time by a factor of 4:1 and I/O time by a factor of about 6:1; to substantially reduce the formatting time, STARAN would have to be used. This would imply ignoring the CALCOMP subroutines used for generating the vector plots. The CALCOMP code would be developed directly in STARAN. If a production system were set up to use STARAN for generating CALCOMP formatted data, the resource usage would be:

STARAN CPU	=	80 seconds
STARAN I/O	=	2 seconds
HOST CPU time	=	20 seconds
HOST I/O time	=	60 seconds .

Item 8 -

Item (8) requires the same kinds of operations as are required by Item (4). If the processing were performed in the STARAN in a production system, the usage times would be:

STARAN CPU time	=	5 seconds
STARAN I/O time	=	2 seconds
HOST CPU time	=	10 seconds
HOST I/O time	=	60 seconds .

With a production system, the total system usage time for all pre-tagging processes would be:

STARAN CPU time	=	830 seconds
STARAN I/O time	=	10 seconds
HOST CPU time	=	140 seconds
HOST I/O time	=	275 seconds .

The tagging phase data requires little explanation with one exception. The TAPEOUT routine requires substantial CDC CPU and I/O time. In a production environment the TAPEOUT routine would not be used except to off-load the production processor. Thus, there is little point in developing STARAN software that would reduce the task time down to the 20 second CPU time/60 second I/O time range. If for some reason it were necessary to develop a fast TAPEOUT module, it could be installed at a later date.

When doing tagging operations, a production system usage would be:

STARAN CPU time	=	70 seconds
STARAN I/O time	=	30 seconds
HOST CPU time	=	20 seconds
HOST I/O time	=	120 seconds .

The processing expensive operations of gridding are already being performed in STARAN. By moving sorting/merging operations into STARAN (Items (14),(16),(19)), system CPU time for these processes can be reduced to about 30 seconds. Such operations were not moved into STARAN for the CONTAGRID program because they are I/O intensive; the slow I/O path between CDC and STARAN would have negated much of the CPU time gain of using STARAN.

Item (18) processing which involves the interpolation of elevations along R/S lines, could be performed in STARAN in about 30 seconds. A gain in processing time of about 40 to 1 over the CDC processing time could be considered an average gain. Because R/S line segments between points of known elevation are not uniform in length a simple

*Excludes the TAPEOUT operation

layout of the R/S interpolation problem in STARAN results in a reduction of STARAN efficiency. The 30 second projection for Item (8) presumes a simple STARAN problem layout and so projects only a 10:1 gain over the CDC CPU time.

The Item (15) task is not CPU expensive and so could remain a host task. If it were moved to STARAN, the neat line linear interpolation operation would require less than one second of STARAN time.

All I/O associated with gridding involves the movement of about 60 Megabytes of data between disc and memory. By using large records, a production system could reduce the number of setups for data transfer to about 8,000. Assuming a SEL set up time of 800μseconds, total I/O set up time for disc-to-memory transfers would require about 6.4 seconds. Actual I/O would require about 50 seconds and disc latency would demand about 320 seconds assuming random file access. When files are contiguous as can be assured, latency time can be halved. Thus, disc-to-host memory I/O should not exceed about 250 seconds.

The set up time to move data to STARAN from host memory would be about the same as that required for set up in moving data from host-to-disc. Therefore, production system usage would be:

STARAN CPU time	=	275 seconds
STARAN I/O time	=	8 seconds
HOST CPU time	=	30 seconds
HOST I/O time	=	250 seconds .

In summary, for all phases of processing, a production system would use:

STARAN CPU time	=	1175 seconds
STARAN I/O time	=	50 seconds
HOST CPU time	=	190 seconds
HOST I/O time	=	650 seconds .

Of course, a production system would be required to perform other than the CONTAGRID tasks to accommodate the DTED process. In particular, it would be required to perform mosaicking, warping, and paneling processes as well as editing support processes. None of the above processes puts much of a CPU load on STARAN. The I/O load on the system would increase as the result of the mosaicking and paneling tasks, but the I/O load would be much less than that imposed by gridding. Of course, the editing load could be determined by the number of interactive edit stations associated with such a system. It is necessary to balance the number of hardware I/O controllers to the response time requirements of the edit stations. By conserving the host machine resources, it becomes possible to perform the myraid short sequential tasks required in an interactive editing environment with minimal time lag.

The man-intensive tasks associated with the DTED process lie in:

1) Pre-compilation,
2) Tagging,
3) Post gridding editing.

By performing mosaicking and paneling in the electronic digital domain, substantial man-hours/sheet should be saved in performing these operations. By using adjacent sheet digital information in most neat line data generation operations, the neat line data development task should also be reduced. Manual R/S data generation would still be required, but the amount of detail required could likely be reduced. Secondary R/S data could be entered directly at edit stations. Based on the CONTAGRID program results, GAC believes the time required to do Tagging can be accomplished using less than 8 man-hours. (Part of the Tagging priming operation would have been accomplished during automatic mosaicking). Furthermore, because of the automatic Q/C associated with the operation, rework will be able to be reduced.

Finally, in the post gridding editing operation, GAC's approach to displaying elevation matrix data should substantially reduce the time to perform final quality control and edit operations.

By being able to view gridded data as an intensity display, direct corrections of the data can be made possible. Where more detail needs to be seen, fast arbitrary magnification can be provided.

Apparently, a production-oriented, CONTAGRID-like cartographic system could reduce the man hours required for all three phases of the DTED process.

SECTION 5 - INVESTIGATION

5.1 APPROACH

The approach is divided into three parts:

1) The testing system,
2) Tasks involved, and
3) Test approach.

5.1.1 Testing System

The three main systems used for this work were:

a) the DMA/ETL Digital Image Analysis Laboratory (DIAL),
b) the GAC STARAN Evaluation & Training Facility (SETF), Akron, Ohio, and
c) the CDC Cybernet Service (terminal link to Cleveland).

The CONTAGRID software was installed on the DIAL facility. The main parts of the above equipment used were:

1) The CDC 6415-8 processor including the 98K core storage, seven and nine-track magnetic tapes, large capacity disks along with the command channel interface link to the STARAN (4-arrays) and the PDP-11/20 supported by two RK05 disks.
2) An off-line CALCOMP plotter with tape drives used for verification of the results of the editing, and tagging software.
3) The GAC SETF, at Akron, Ohio, which was utilized for initial checkout of the STARAN processing. Later, all the gridding software was initially checked out on this equipment. A number of the I/O commands available on the DIAL system were simulated at GAC to allow more rigorous testing to be performed.

4) The COMTAL (512x512) Image and Graphics Display Equipment, which is part of the SETF equipment was also an aid (along with additional software) in allowing the display of the gridded results of the SHIRAZ Test Region.

5) The CDC Cybernet Service, which was valuable for initial checkout of many of the CDC Fortran routines which comprise part of the CONTAGRID software package.

5.1.2 Tasks

In order to accomplish those tasks necessary to tag contours and subsequently generate a grid of elevations, Goodyear Aerospace performed the following assignments.

- Generated CDC I/O software to read and merge Run-Length-Coded (RLC) data produced by the DMA Raster Scanner/Plotters and store the data file subsequently on a CDC disk.
- Developed CDC-Fortran software to send the RLC data to STARAN and receive and store the resulting array vector data.
- Modified the existing STARAN software to include extra vector header information for editing and tagging and separate the contour data into index and non-index files.
- Generated new STARAN software to perform junction coding to allow the detection and classification of numerics and special symbols.
- Generated CDC-Fortran routines to build the master vector information from the array vectors.
- Generated CDC-Fortran software to support the generation by STARAN of numeric and special symbol vector identification lists and new 'join vectors' subsequently produced by the automatic editing routines.
- Generated the corresponding STARAN software.
- Generated CALCOMP plotting software to display proof plots of the edited data along with separate plots of the numeric and special symbol vectors.

. Generated CDC software to manually edit the various existing vector data sets from card input.

. Generated STARAN software to build new master vectors and array vectors from lists sent from the CDC.

. Generated CDC-Fortran routines to build contour vectors from the previously edited vector data sets.

. Generated CDC and STARAN software to automatically tag sheet boundary intersecting contours with elevations from a subset of manually assigned index elevation tagged contours.

. Generated software to edit the boundary contour elevation data sets from cards. This software included the deletion of incorrectly tagged contours and the insertion of corrected elevations for these same contours along with the insertion of elevations for untagged contours.

. Generation of CDC software to manage the I/O of data to STARAN for the tagging of closed contours, i.e., those not intersecting the sheet boundaries.

. Generation of STARAN software to tag the closed contours.

. Generation of CDC software to plot by elevation increment the tagged contour vectors, and the subsequent generation of a DGR formatted data tape suitable for gridding using existing DMA software.

. Generated CDC and STARAN software to reduce/resample 25 micron vector data to the required 10-mil grid.

. Developed software to accept DGR formatted vector data as input.

. Generated CDC software to segment DGR input data into separate files; i.e., ridge/stream (R/S) points, spot elevations, neatline points, etc.

. Included software to build neatlines from neatline points.

. Generated CDC programs to partition R/S line data for STARAN processing.

. Developed CDC/STARAN software to find the intersections of R/S lines with points of known elevation.

. Developed CDC software to build R/S points using linear interpolation.

. Built file merging and backscanning software for developing the input to gridding.

. Generated CDC/STARAN software to perform gridding using interpolation.

A number of other software items were also generated during this contract to run at GAC, Akron, namely:

. STARAN software to display a gridded subsection of a test sheet on the COMTAL display using different color-coded elevations (See SUMMARY and CONCLUSIONS - Figure 4-3).

. Fortran software to run on the SETF equipment to display a hard copy output of sections of raster scanned source material. This software has the ability to 'window' a specific section of a sheet and magnify that section (if required) and allow a user to determine the 'quality' of the raster data (see Figures 6-4, 6-5, 6-6, 6-7).

5.1.3 Test approach

The approach to testing the CONTAGRID software was to use digitized line data generated from a transparency of the SHIRAZ region of Iran and scanned at 1-mil resolution by the DMA Raster Scanner Plotter (RAPS) at the DMA Hydrographic/Topographic Center.

In particular: GAC:

. Used a sub (4"x4") section of the SHIRAZ sheet to checkout the majority of the CONTAGRID software (see Figure 5-1)

. Examined other scanned data produced by:

a) The Hamilton-Standard scanner, and

b) The DMA AC RAPS.

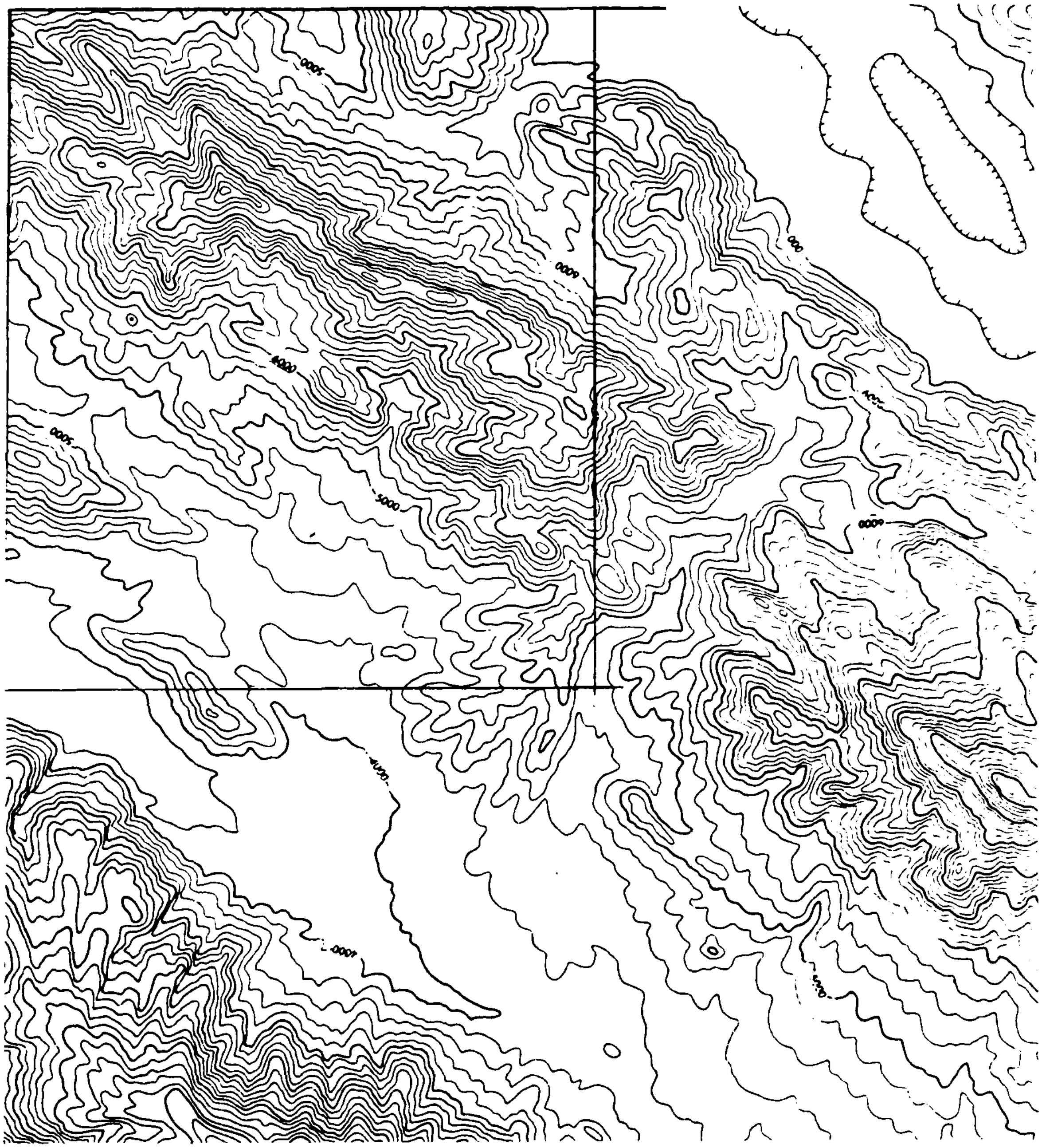


Figure 5-1. Photocopy of Source Shiraz Contour Data 1:1

- Utilized neatline, point elevations and Ridge/Stream data generated by DMA in the gridding process.

Goodyear Aerospace generated the following items during the contract:

- CALCOMP plots of the contour vector information in vector form and separated into index and non-index files after both automatic and manual editing.
- Plots of the numeric and special symbols.
- Plots of the tagged contours by elevation increment.
- A DGR formatted data tape of the tagged contour data (subsequently plotted at DMA).
- A tape of gridded data in the DMA Standard form for Digital Terrain Elevation Data.
- A color COMTAL display (and photographs) of the gridded data set including Ridge/Stream data, etc. The information was color coded by predefined elevation bands.

SECTION 6 - DISCUSSION

6.1 GENERAL

A discussion of the CONTAGRID software is given in this section. The discussion is divided into four sections: pre-tagging, tagging, gridding, and editing. Each section gives the basic techniques involved, together with related information that may help to clarify to the reader the fundamental concepts behind each process. More detailed information on the software developed for these procedures is documented in the Appendices A through D.

6.2 PRE-TAGGING

Pre-tagging involves the processing of the raster-scanned data to (and including) the generation of the untagged contour vectors as shown in Figure 6-1.

The raster-to-vector conversion is performed in a specific manner in relationship to the raster-scanned sheet. It is important to understand this as the I/O contributes significantly to the overall processing time.

Figure 6-2 helps to define, with respect to the image area, a scan line, array load, iteration, and data block. An array load refers to the amount of data processed by one STARAN array and is 224 cells wide and 248 cells long, as shown in Figure 6-2(B).

The RLC data is sent to STARAN until 224 lines have been collected. These are then converted to a 'binary image' of the data block and segmented into separate iterations and stored on the second RK05 disk (Drive 1). This data is then recalled one iteration at a time and processed in the following manner.

- Load the arrays.
- Separate index lines from non-index lines (by width).

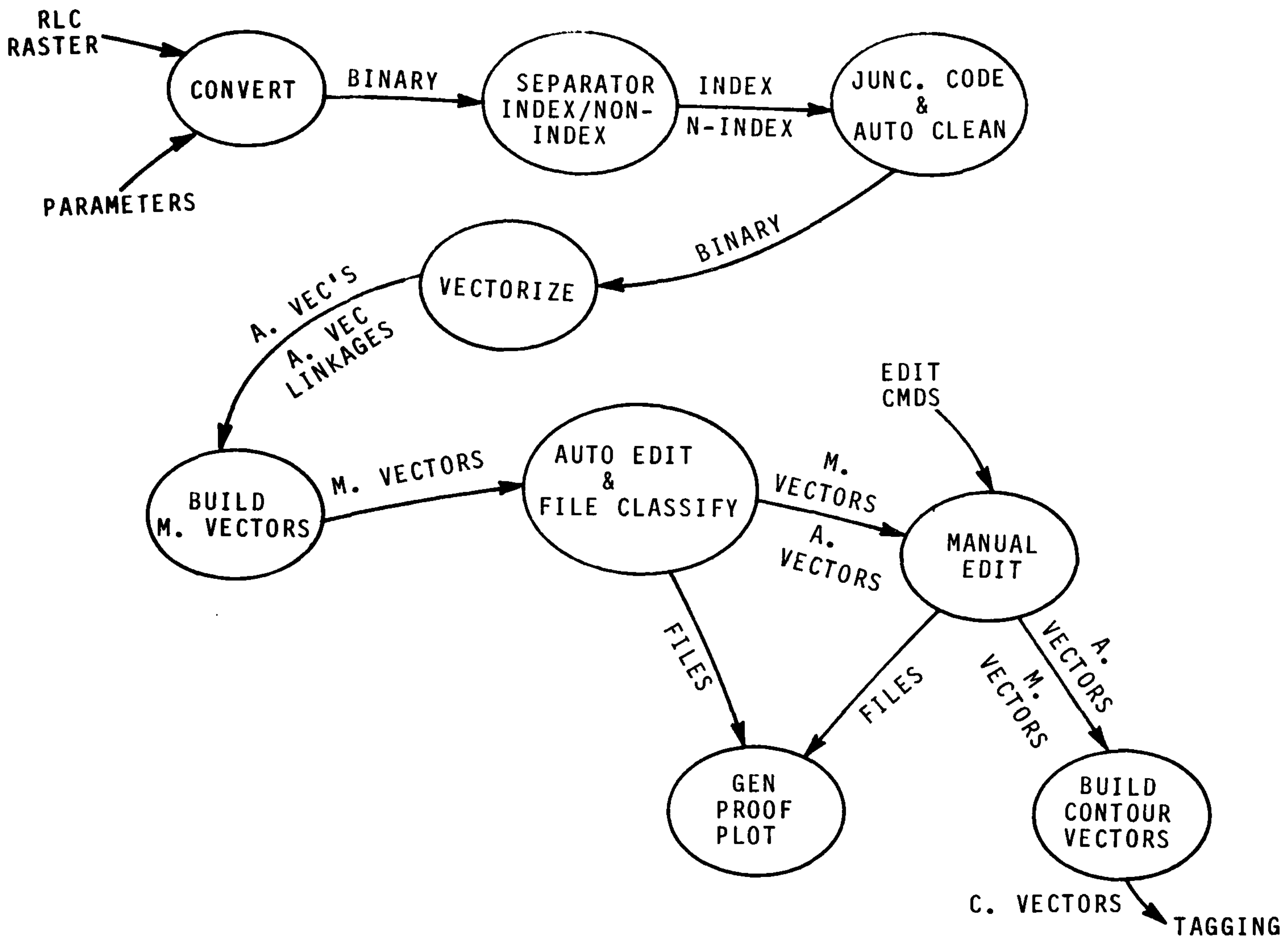


Figure 6-1. Pre-Tagging Data Flow Diagram

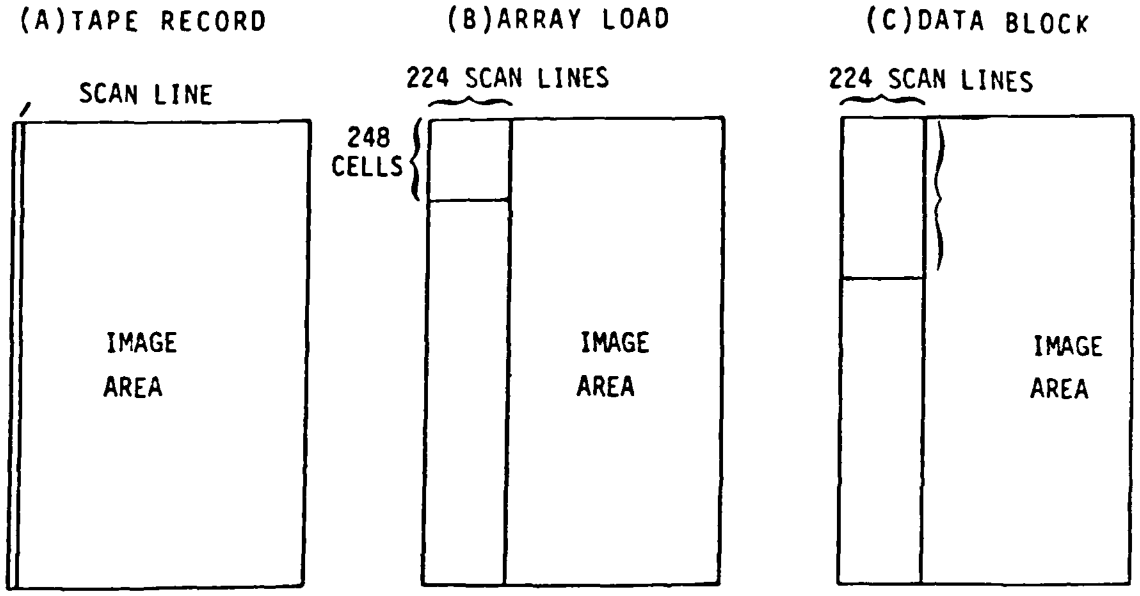


Figure 6-2. Relationship of Image Area to Tape Record, Array Load, and Data Block

- Clean-up data.
- Thin data.
- Code all junction points of 3 or more lines.
- Convert data to array vector form (along with array vector linkage information).

This procedure is performed for all iterations and the vector data is sent to the CDC which later builds the master vectors. The data is processed twice per data block, once to isolate index lines, and once to isolate the non-index lines.

The above steps are repeated for as many data blocks as are required to produce vectors for the complete sheet.

Master vector files are built from the array vectors and the corresponding linkage information. A master vector is a connected set of array vectors whose ends begin/terminate at sheet boundaries, junctions or at open end points.

The next procedure attempts to automatically join up any 'line breaks' existing within the sheet boundaries. Numeric symbols are determined and marked for subsequent deletion. Files of vectors are created which comprise special symbols such as depressions, fills, and cuts.

Plots of all of these files are generated on the CALCOMP for verification and manual editing when required. Each master vector is assigned a unique identification number called a 'correlation number' (Plot ID) which can be printed on the proof plots at one end of the vector. The index and non-index master vector numbers are not unique and use of their master vector numbers (instead of Plot ID's) on plots which composed both these files could lead to misinterpretation. All editing is performed by deleting changing, and adding to these correlation lists. Several cycles through manual editing may be required to completely edit the sheet.-- Plots may be generated after each cycle.

Finally, the edited master vectors are connected to the master vectors using the previously edited joins. This process is called the 'SQUEEZE' routine. The results are contour vectors which are then listed along with any incomplete contour vectors; i.e., those whose ends do not start or end on a sheet boundary or are not closed on themselves within these same confines. These 'dangling' contours were missed during manual editing and are corrected by iterating through the manual edit and the 'SQUEEZE' procedure until a clean contour vector data set is established.

6.3 TAGGING

Tagging involves the association of elevations to the contour vectors and the subsequent generation of data in a form suitable as input to the gridding process. Figure 6-3 shows the data flow through tagging. The tagging process is divided into two sections. First, all sheet boundary intersecting contours are tagged and edited, then the internal (closed) contours are tagged and edited.

The open tagging process is performed by STARAN and utilizes an ordered elevation list of the manually tagged index boundary contours. The index-contours were originally selected because:

1) They are thicker than the non-index contours, and are most easily identified after line separation as a separate plot,
2) They are fewer in number than the non-indexed contours, thus reducing the amount of data requiring manual tagging, and
3) Their elevations are equally spaced, enabling the calculation of the non-index contours elevations to be simplified.

The STARAN performs the following tasks during open tagging.

- Receives the ordered index elevation list (by boundary left, bottom, right, top).
- Generates the ordered sheet boundary list of contours for the specific boundary.

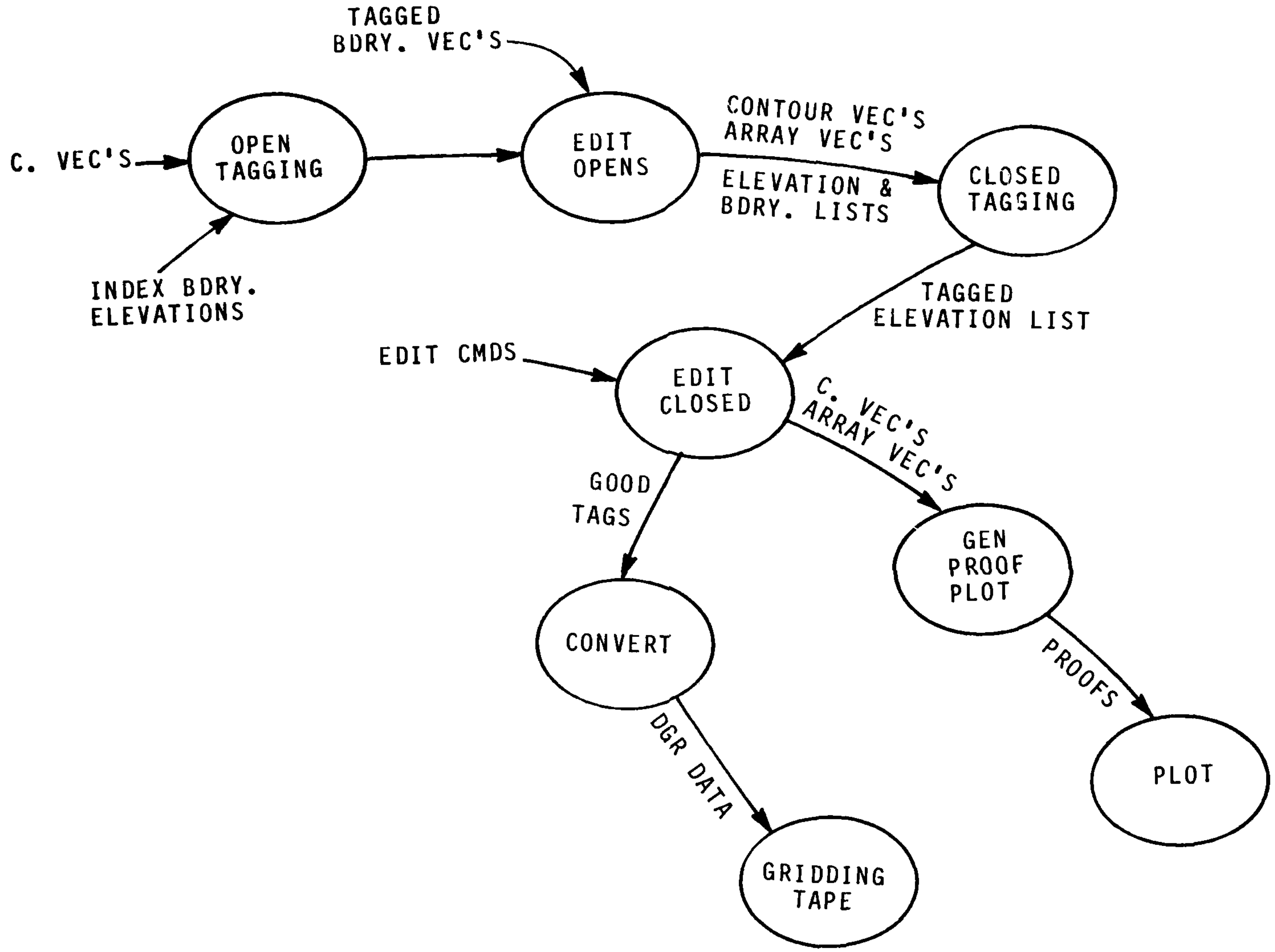


Figure 6-3. Tagging Data Flow Diagram

- Correlates the two lists and identifies any mismatch in the number of index boundary matchings.
- Excludes any 'untaggable' contours due to insufficient data (these are later tagged during editing).
- Tags the remaining boundary contours and sends the elevation list to CDC.

The elevation list is printed and any untagged boundary contours are tagged manually. The elevation list, along with the contour vector and array vector information is used by STARAN for the closed contour tagging process.

The closed contour tagging procedure is performed by STARAN and incorporates the following steps:

1) Receives elevation list and vector information from CDC,
2) generates a counter-clockwise sheet boundary flag file to identify if the slope is rising or falling as each boundary contour intersection is found,
3) Generates ordered North-to-South array boundary lists of contour crossings corresponding to the edge of each Data block,
4) Tags all closed contours crossing the N-S line,
5) Cross-checks the tagged contours against previous N-S lines and flags any inconsistency, repeats steps 3), 4), and 5) for all the data blocks on the sheet, and
6) Sends the updated elevation list to CDC.

Final editing is performed (if required) to complete the tagging.

The tagged contours are plotted by elevation increment; i.e., one plot for the index contours and one plot for each of the non-index elevation increments between two different index contours. Also, a DGR formatted data tape is generated for use by either the new parallel gridding process developed on STARAN or by the existing DMA gridding scheme.

6.4 GRIDDING

This item discusses the basic gridding procedure. The individual steps are identified in Figure A-4 of Appendix A.

The first step is the generation of the contour point of known elevation. These points can be obtained from one of two sources, either:

a) from the DGR input tape (if it contains vector data), or

b) from the output of the tagging portion of CONTAGRID.

In case b) this requires the use of a 'resampling process' to convert/reduce the 1, 2, or 4 "mil" resolution data to the 10-mil grid spacing. This process is performed in parallel by STARAN.

The next step reads the DGR tape which contains the supplementary image data, and segregates the data into four types: stream/ridge (R/S) lines, spot elevation, R/S junctions, and neat line points.

The next task uses the neat line information and contour vector (check boundary) information to build the four neat lines. The process used is a linear interpolation. The next process is to use the R/S points, the contour known-elevation points, and the R/S junction points to produce the R/S intersections. This process again utilizes the STARAN processor. The process then continues by interpolating along the R/S lines to produce R/S points of known elevations.

Following this all points of known elevation; i.e., R/S points, spot elevations, and neat line elevations are merged and converted into a 'backscan ordered' file. This file is used as input to the gridding process along with the left edge neat line elevation points. At this time, STARAN's primary purpose is to perform the 'Dual-axis Parallel Gridding Algorithm' (DAPGAC). Now, it becomes necessary to describe the justification for the

*"mil" = 25 microns.

implementation of a 'new' algorithm. In March 1979, it became apparent that the primary potential benefit of implementing DMA's Planar Interpolation Gridding algorithm on STARAN - the ready validation of STARAN gridding results by the direct comparison to comparable results achieved with DMA's UNIVAC - could not be realized. Prior to this conclusion, GAC had already reported that structuring the inherently sequential algorithm in such a way as to keep the processing elements of a parallel processor busy, caused output grid data to be developed along 63° diagonals. To "un-skew" the output developed using the STARAN version of DMA's Planar Interpolation gridding, heavy I/O demands would be placed on the CDC6400/STARAN computing facility.

Also, at the same time, a number of DMA personnel expressed the opinion that in addition to the fact that the Planar Interpolation Gridding algorithm didn't match the parallel processor architecture, the algorithm had deficiencies that needed attention. Later, it was mutually agreed by ETL, DMA, and GAC representatives that effort to implement the Planar Interpolation Gridding algorithm would be halted and that GAC should suggest a gridding approach suitable for parallel processing. Subsequently, at DMA HTC, GAC described the Dual-Axis Parallel Gridding Algorithm for CONTAGRID (DAPGAC) and recommended that it be implemented under the CONTAGRID program. It was agreed that GAC should proceed with the DAPGAC. The nature of the algorithm and its comparison to the Planar Interpolation Gridding algorithm is summarized in Appendix D.

The output of the gridding process is formatted in accordance with the DMA terrain elevation data file requirements.

AD-A091 736 GOODYEAR AEROSPACE CORP AKRON OH F/G 9/2
CONTOUR DIGITIZING AND TAGGING SOFTWARE (CONTAGRID).(U)
APR 80 N J ADAMS, M ANDERSON, G BIECKER DAAK70-77-C-0223
UNCLASSIFIED GER-16752 ETL-0228 NL

1 OF 2
AD
A091 736

END
DATE
FILMED
81-1
DTIC

6.5 EDITING

This item discusses many of the 'edit' functions performed at different stages of CONTAGRID processing.

The editing processes can be broken down into the following categories:

- Editing to clean up the raster source data,
- Editing to reassign misclassified numeric, depression and special symbol lists, and
- Editing to correct or assign contour vector elevation tags.

It is the first two of these categories that will be discussed here.

6.5.1 Editing Source Data

The raster data that is scanned and used as the source input to CONTAGRID may be error ridden for several reasons:

- The original film may contain dirt specks that get rasterized,
- The line data may have been poorly drawn, i.e., outside the tolerances of parts of the software,
- Although the original film may be within these tolerances, the scanning process itself could generate errors such as:
  - The operator using an incorrect 'threshold' setting, or
  - 'Line blooming' across or down the sheet due to expansion of the scanner drum itself during operation, etc.

Figures 6-4 and 6-5 show plots made of two different machines raster scanned version of the same source material at 1 micron resolution. Each asterisk (*) indicates a single resolution

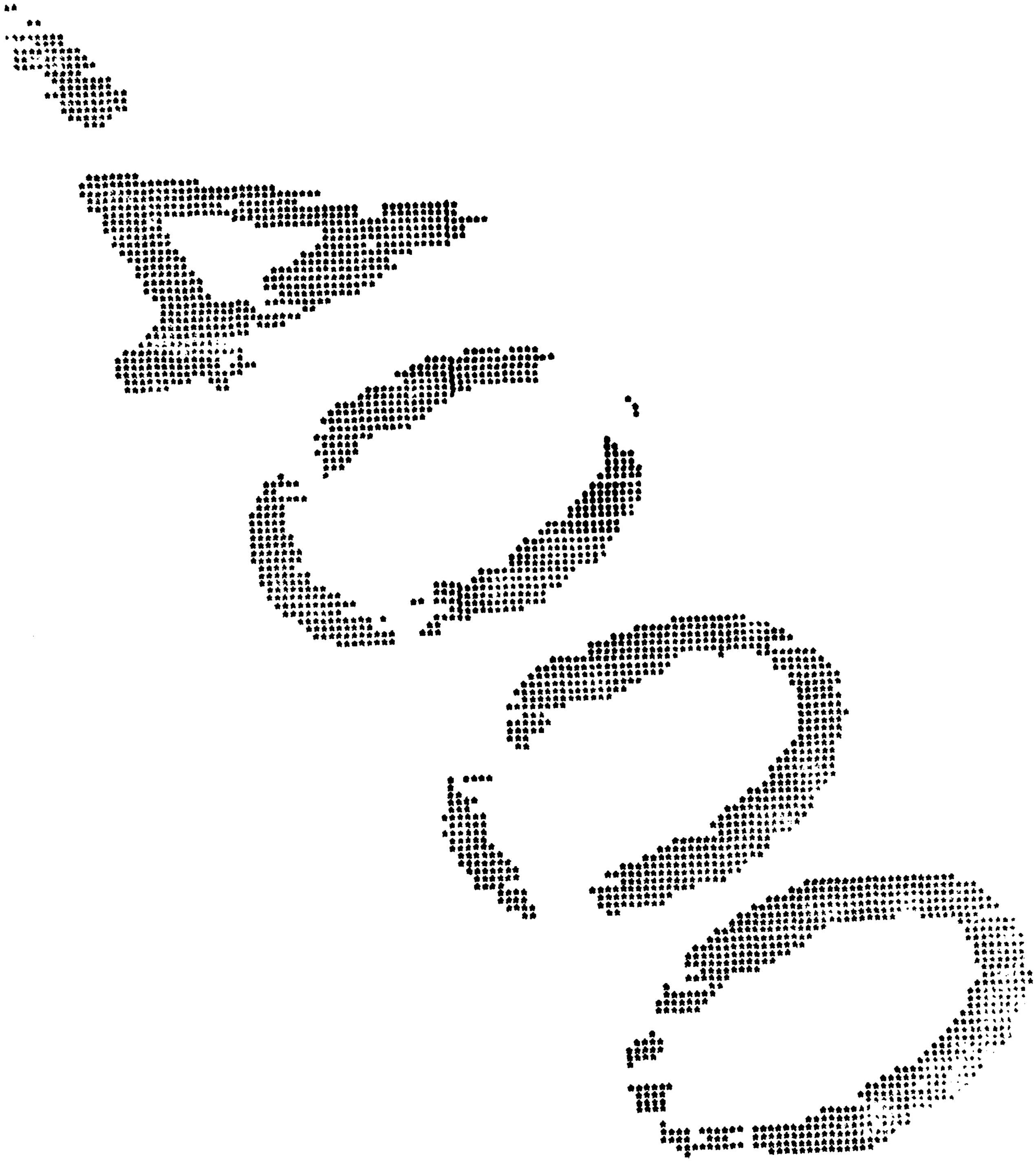


Figure 6-4. Elevations D.M.A. T.C. RAPS

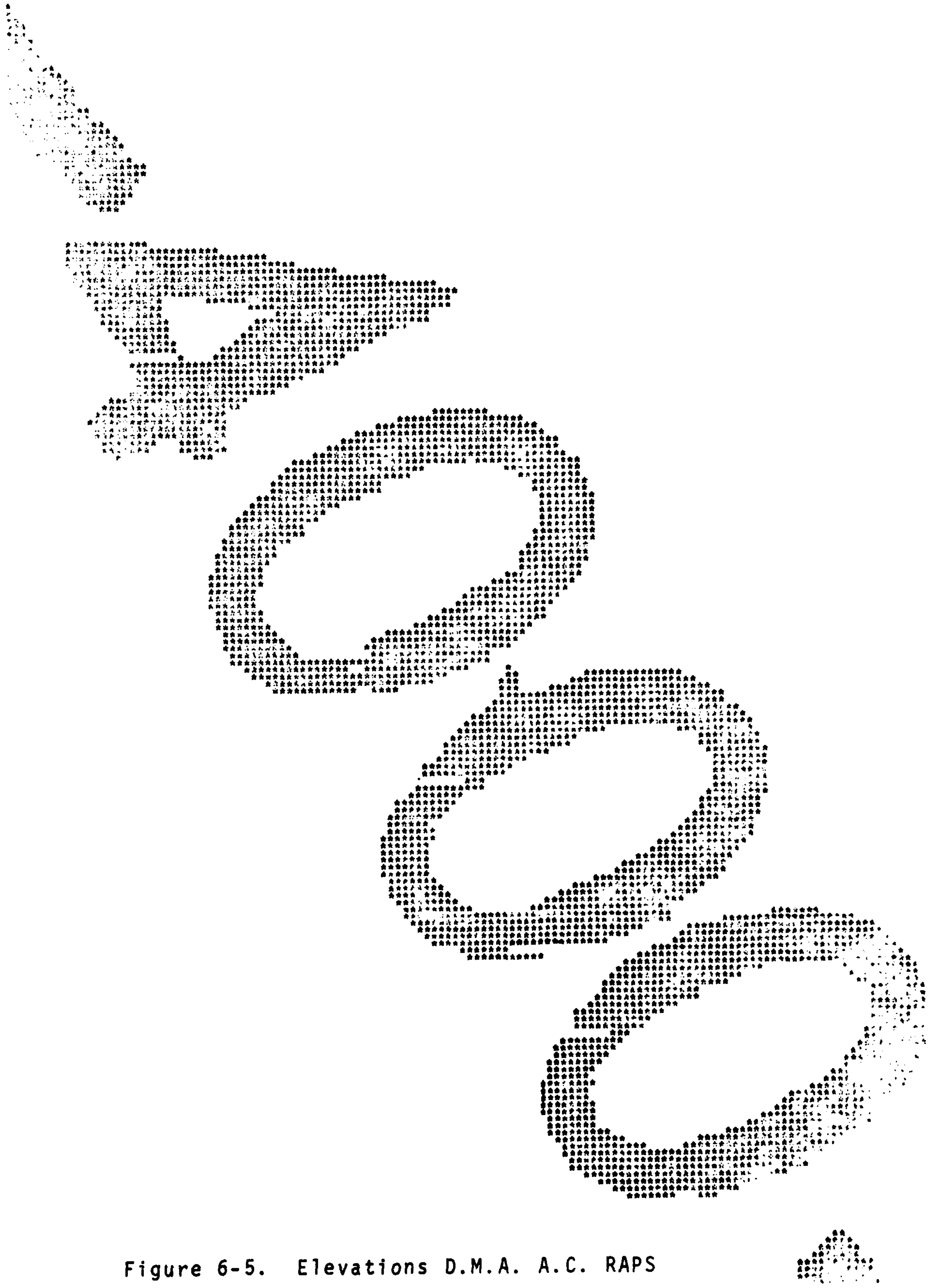


Figure 6-5. Elevations D.M.A. A.C. RAPS

unit. The heavy lines approximate center lines produced after line thinning and the automatic clean-up routines used within the STARAN processing at the beginning of CONTAGRID.

The ability of the automatic editing software to identify these numerics is greatly reduced due to the more complex form of this data (see Appendix B for more information on numeric detection, etc.).

Figure 6-6 shows another situation, this time the source film had 'dirty' spots which caused many of these problems. A large number of the small regions are 'cleaned up' using the CLIP/ Cluster Elimination routines developed for prior raster processing software.

Figure 6-7 indicates the case of 'line coalescing'. Notice how the merging of the two lines has occurred in the scan direction. Again, the solid line conforms closely to the resulting data after line thinning. Each of the short vectors joining the two lines has to be deleted from the resulting vector files by manual editing (see Appendix B). Also, because each vector contains a large amount of header information, it means that these redundant vectors are increasing the intermediate vector file size which consequently increases any I/O or search times on this file.

Two other situations that produce a disproportionate amount of editing are:

- The use of input data whose line weights do not stay within the specified tolerances due to poor compilation, and
- The use of input data whose line weights do not stay within the specified tolerances due to 'blooming' of the scanned data.

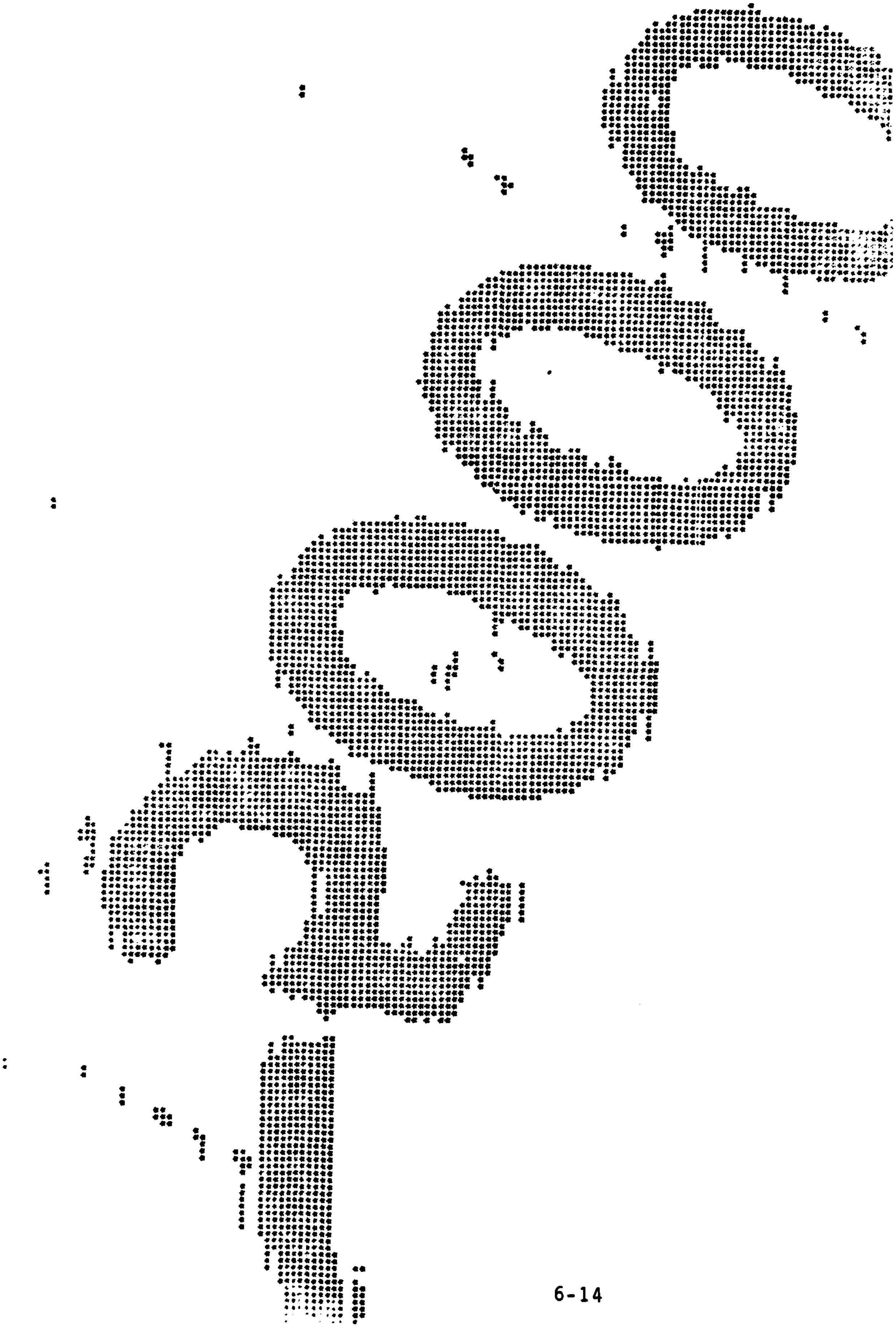


Figure 6-6. Elevations and Clutter

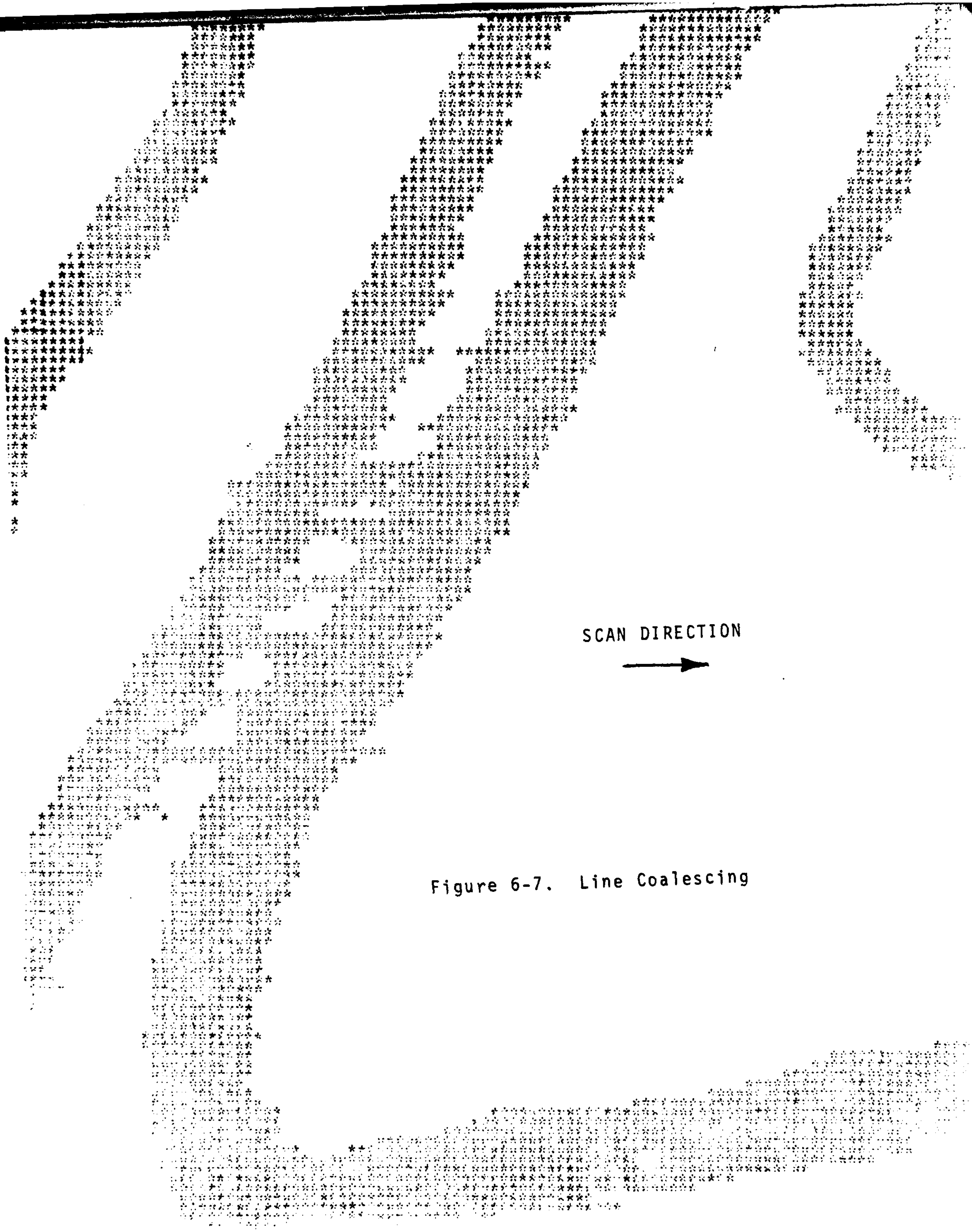


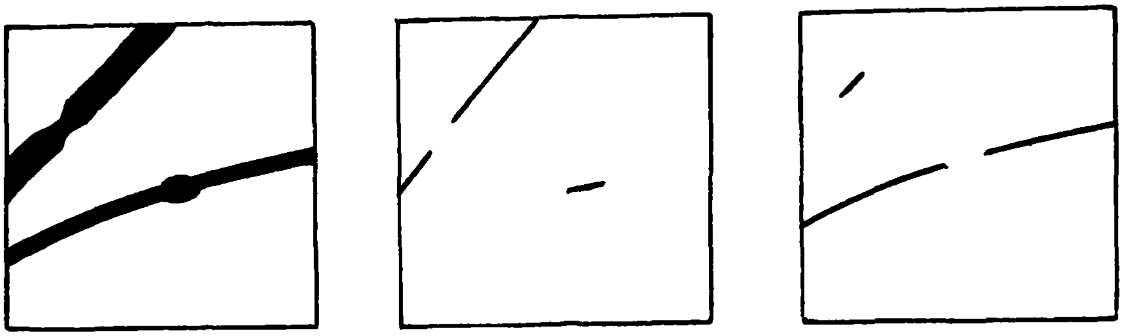
Figure 6-7. Line Coalescing

Figures 6-8 and 6-9 show what happens to these data sets after line separation and thinning. The misclassification causes immense problems for editing.

6.5.2 Editing to repair Line Breaks, etc.

Figures 6-10 and 6-11 show the top-left corner of the SHIRAZ data sheet (shown in Figure 5-1), after line separation, vectorization, automatic editing, and plotting. Each vector has been assigned an identification number which allows it to be edited. Figure 6-12 shows the automatically generated join vectors for the index vectors.

Figures 6-14 and 6-15 show the same two vector sets after manual editing of the problems indicated on the first two plots.

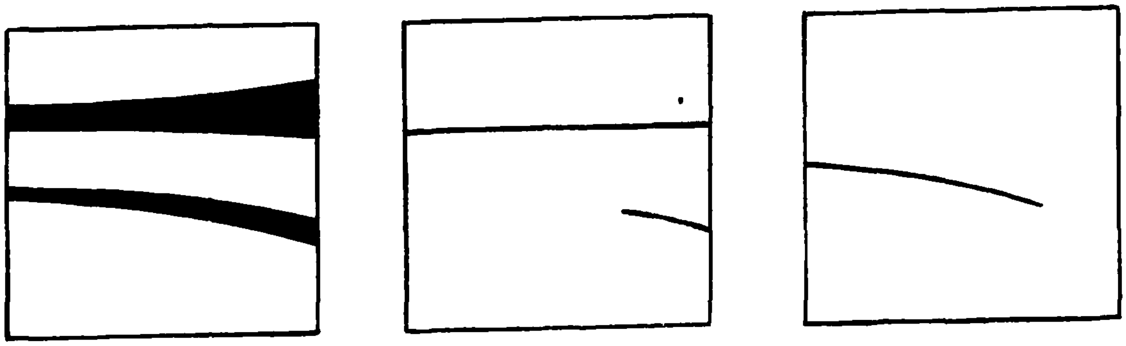


A. INPUT

B. INDEX SEPARATED/ THINNED

C. NON-INDEX SEPARATED/ THINNED

Figure 6-8. Misclassification due to Inconsistencies in Line Weights



A. INPUT

B. INDEX SEPARATED/ THINNED

C. NON-INDEX SEPARATED/ THINNED

Figure 6-9. Misclassification due to 'Blooming'

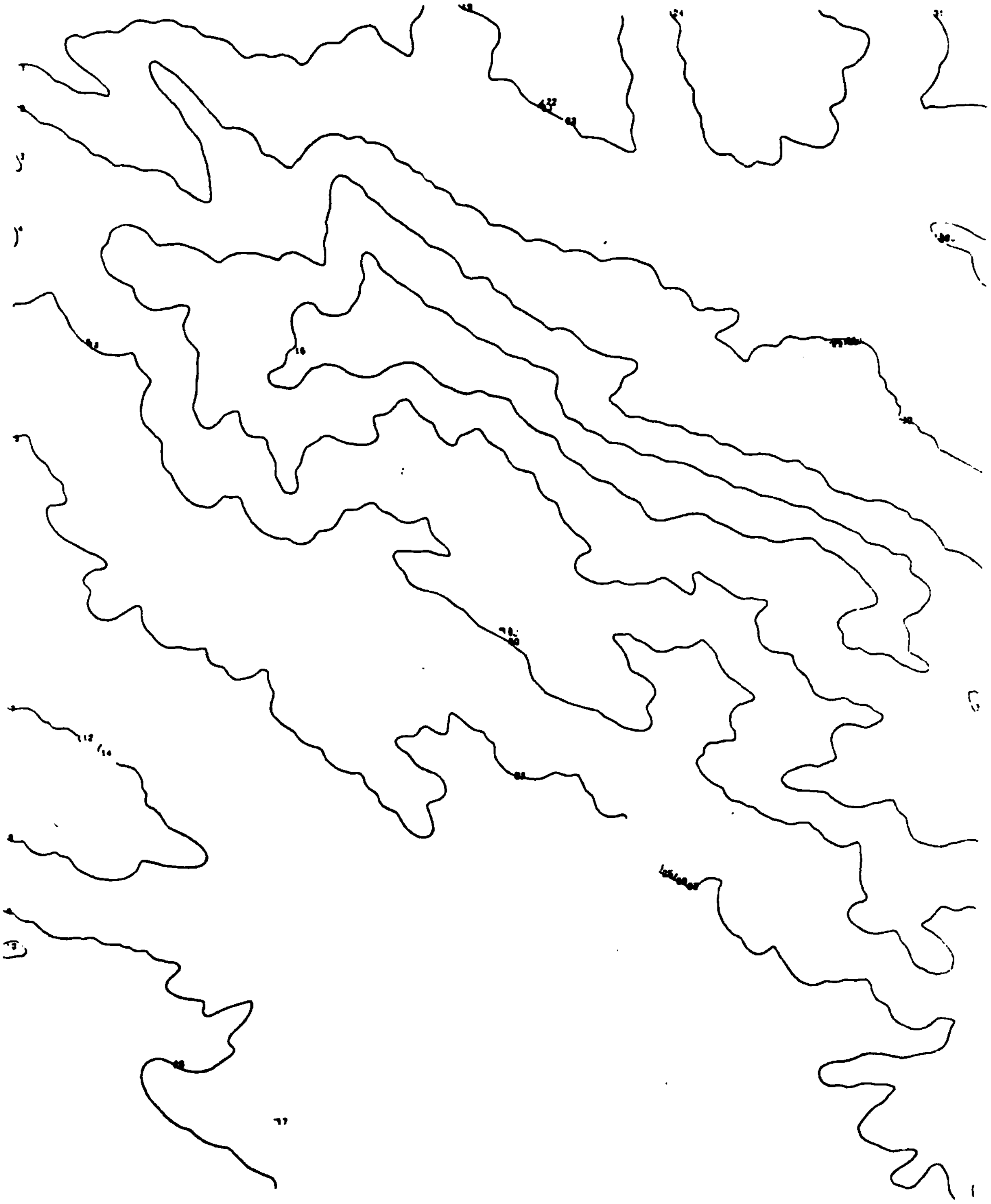


Figure 6-10. Index Contours at 2:1 after Automatic Editing

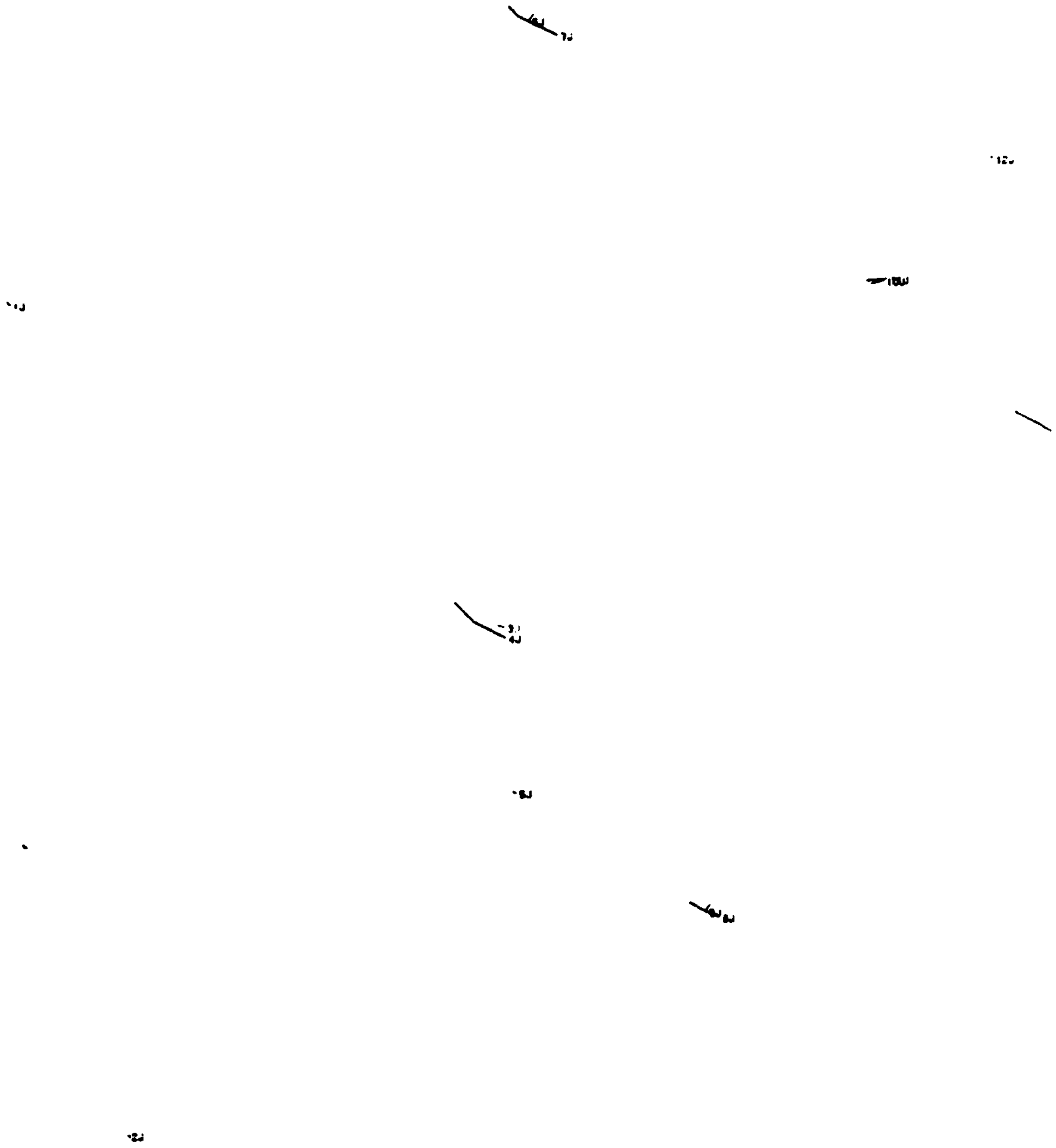


Figure 6-11. Index Contour Join Vectors after Automatic Editing

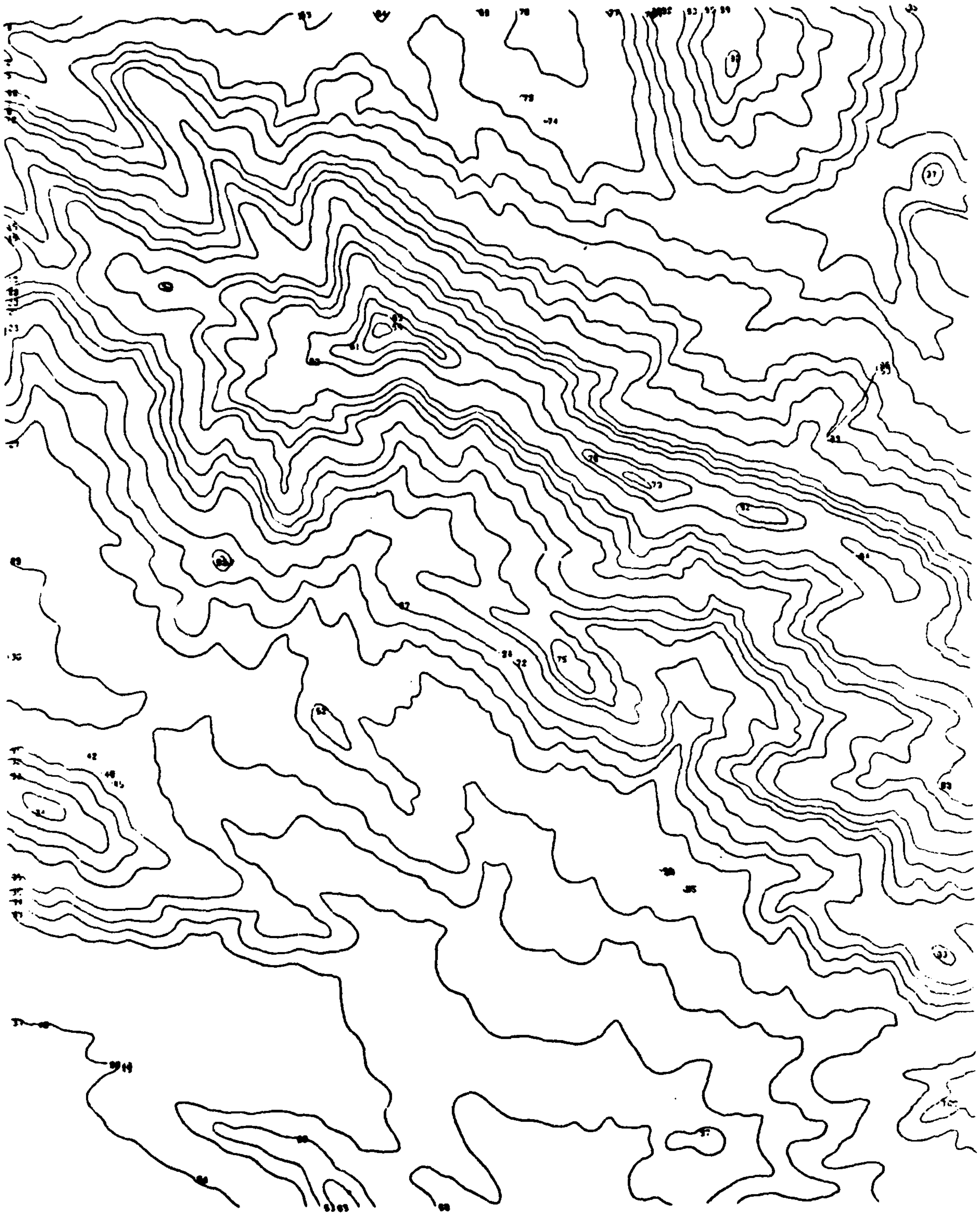


Figure 6-12. Non-Index Contours at 2:1 after Automatic Editing

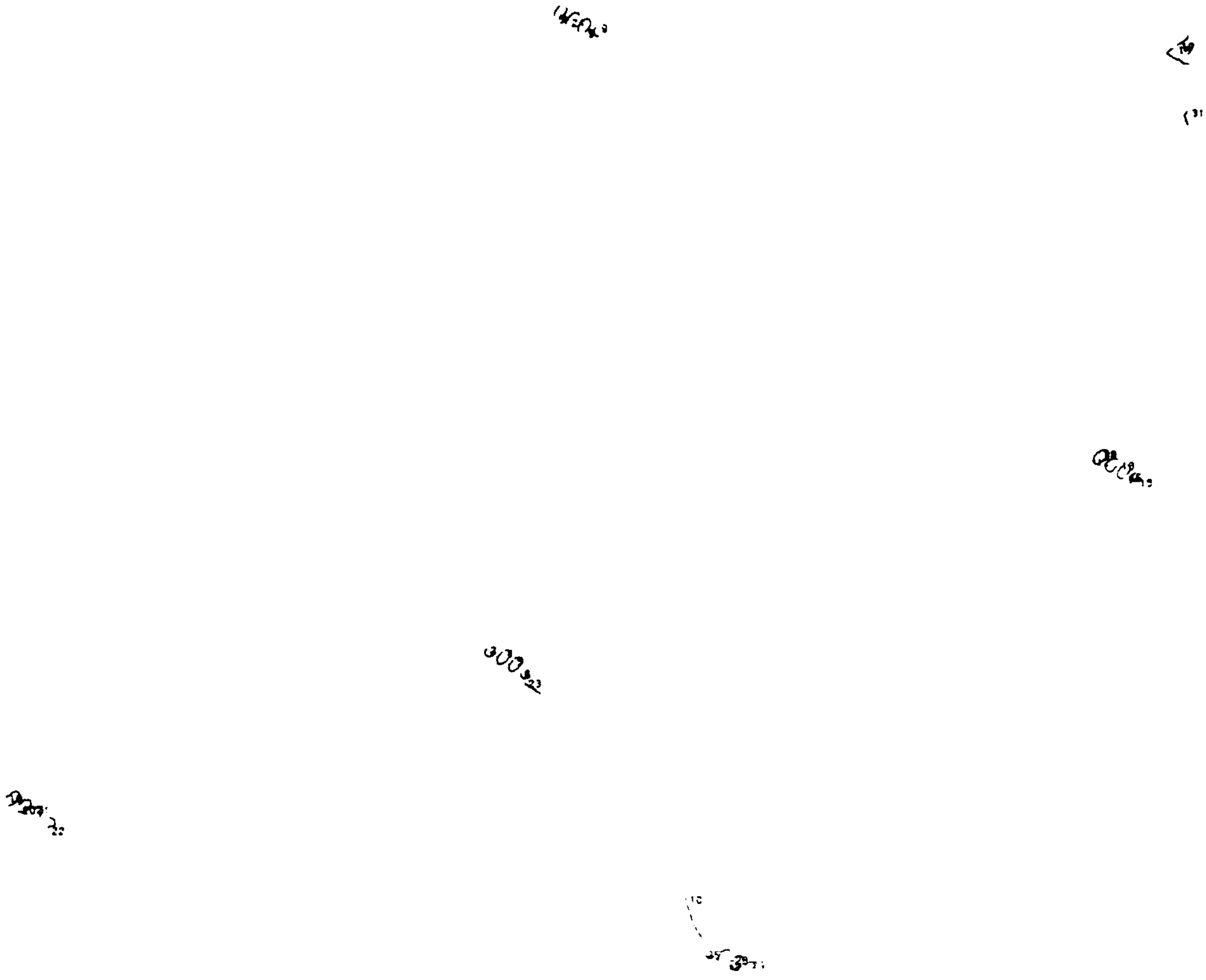


Figure 6-13. Numerics at 2:1 after Automatic Editing

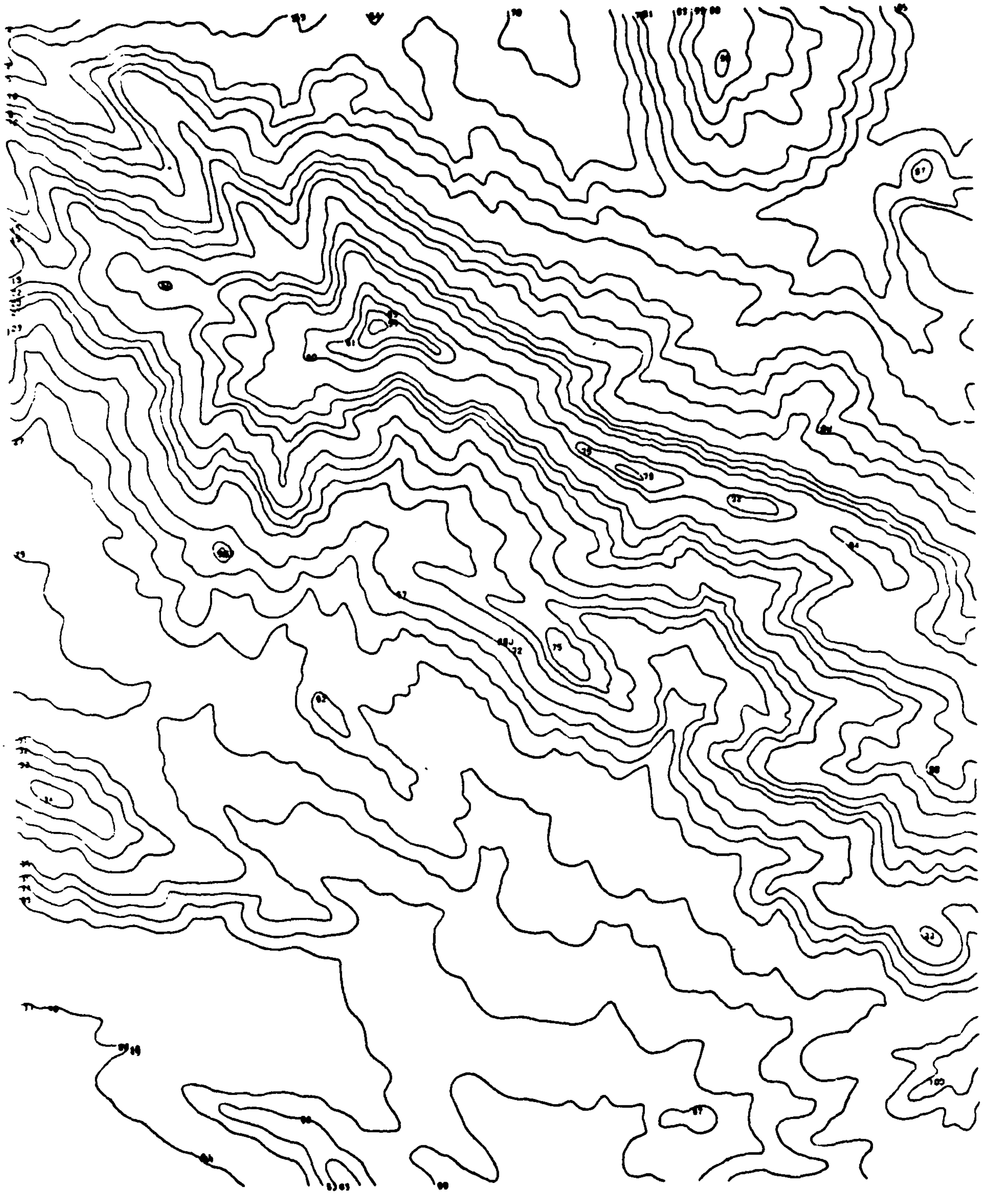


Figure 6-14. Non-Index Contours at 2:1 after Manual Editing

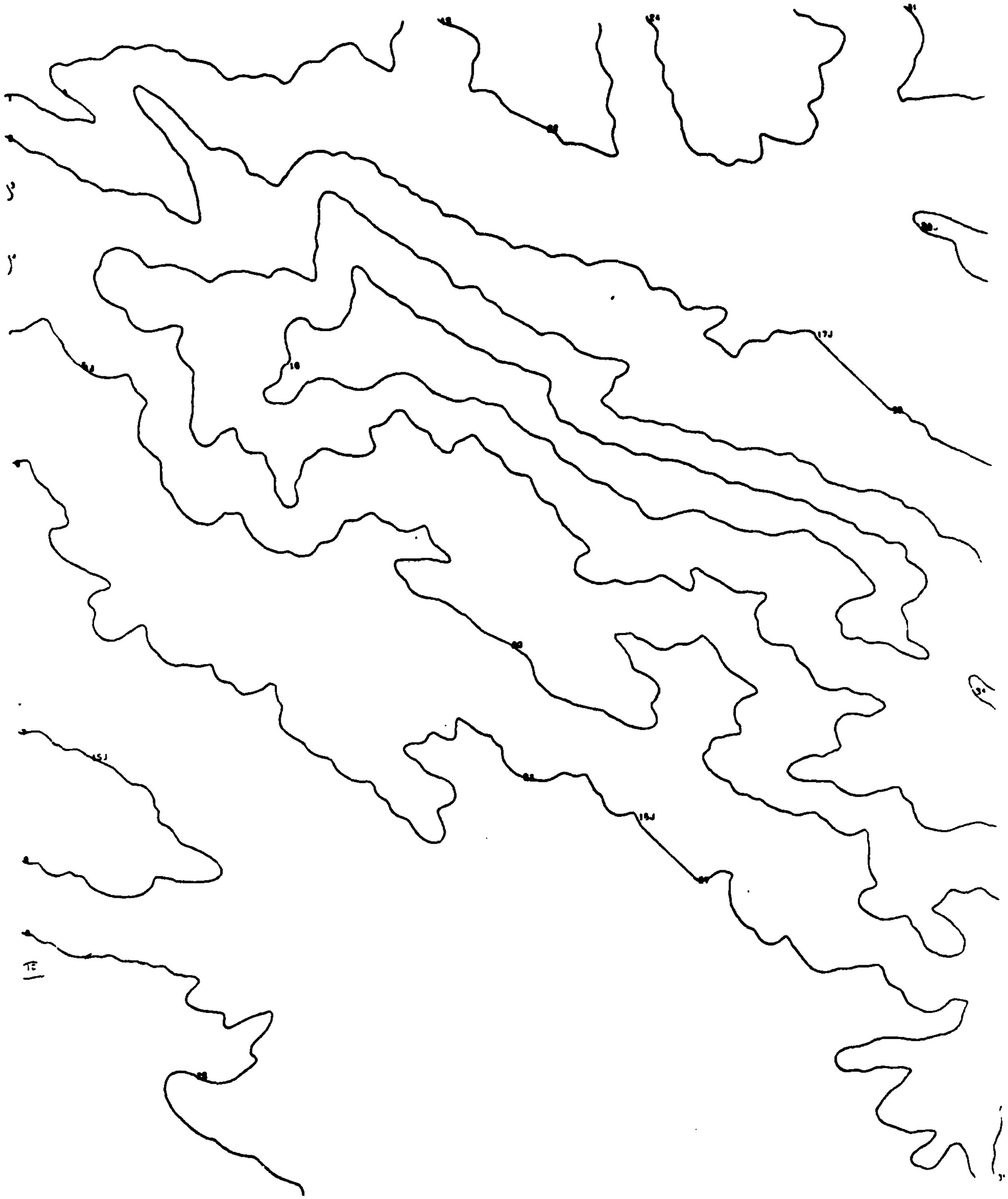


Figure 6-15. Index Contours at 2:1 after Manual Editing

SECTION VII - RESULTS

In general terms, the results of the contractual work are described below.

There exists a CONTAGRID software package which shows the validity of an automated approach to the tagging and gridding of raster scanned contour data.

The above software was structured to allow processing of all usual data resolutions and densities, and accomodate sheets up to 19"x22" data areas. The software was built using a 'structured' approach which allows the user to investigate the performance of individual sections of the process. Specifically, this allows the user to improve individual performance in certain areas and change specific functions from presently manually-oriented approaches to more automatic or interactive methods as new hardware or software techniques become available.

Hard copy outputs have been generated at all the major steps in the process as shown in Figures 7-1, 7-2, and 7-3.

Magnetic tape data was also generated after both tagging and gridding to allow the user to test the data against other 'tried' software.

At various points in the processing self-checking procedures have been incorporated to allow early detection of problems. This allows the user immediate insight into the problem and the capability to overcome the error by repeating the prior process

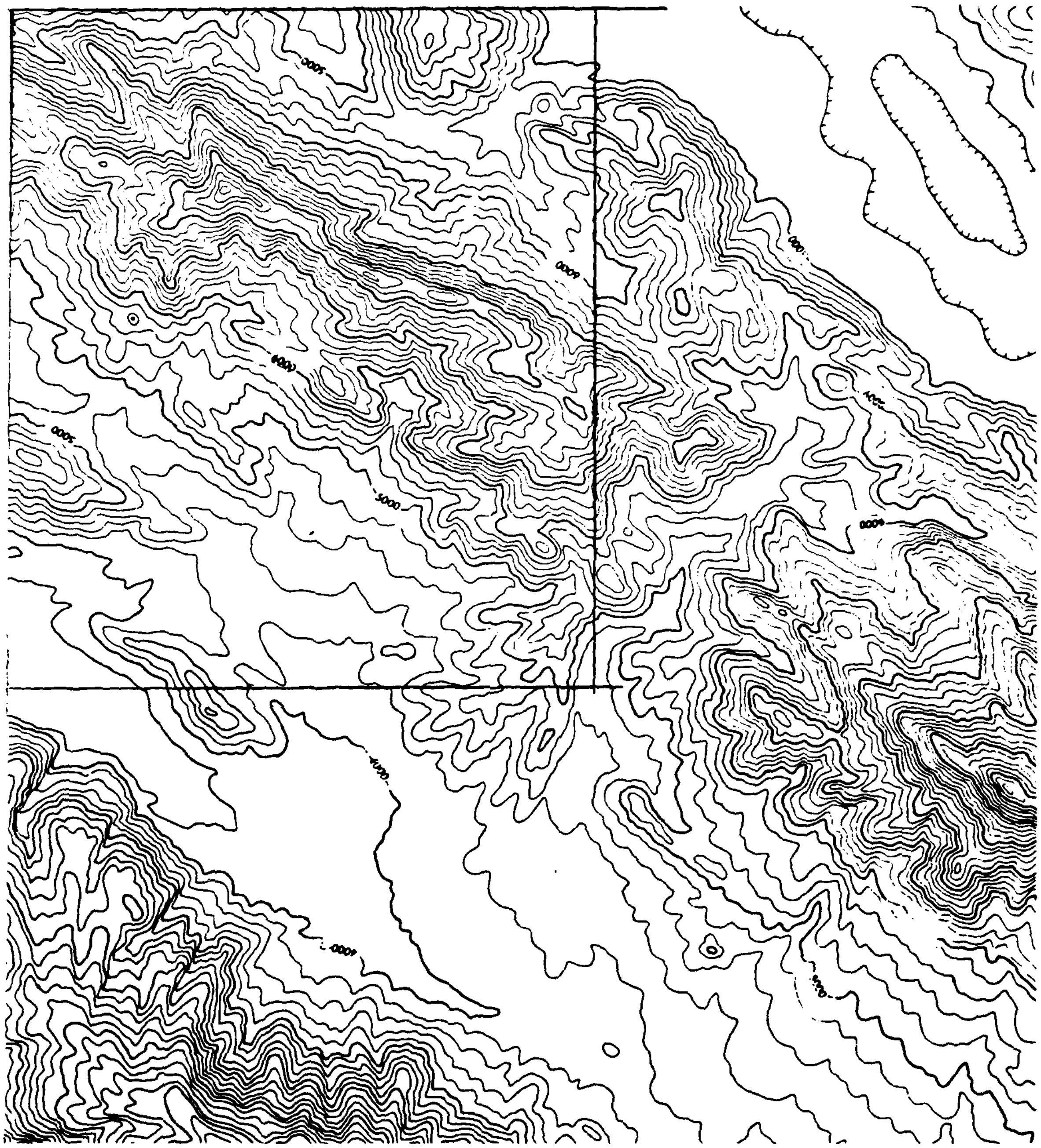
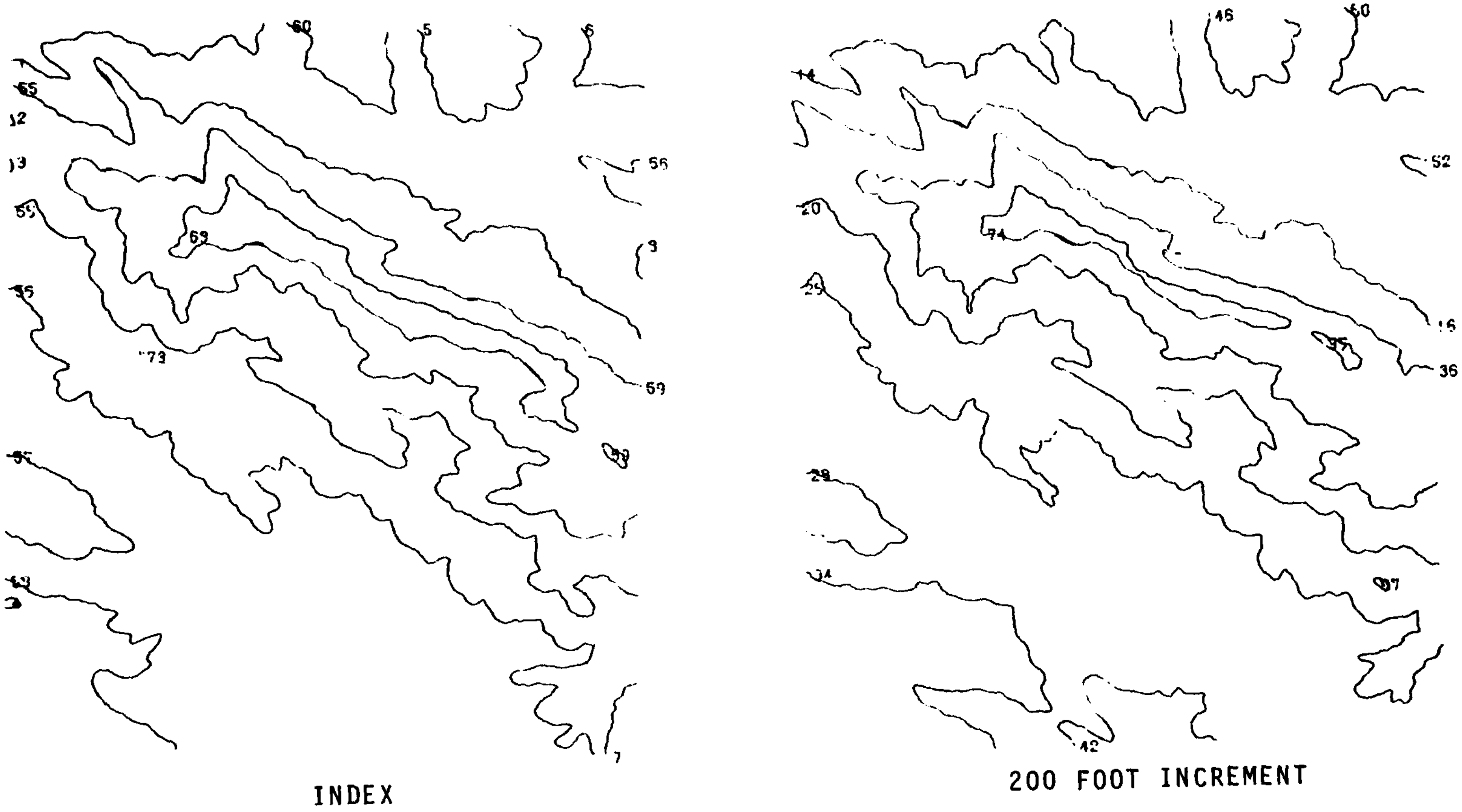


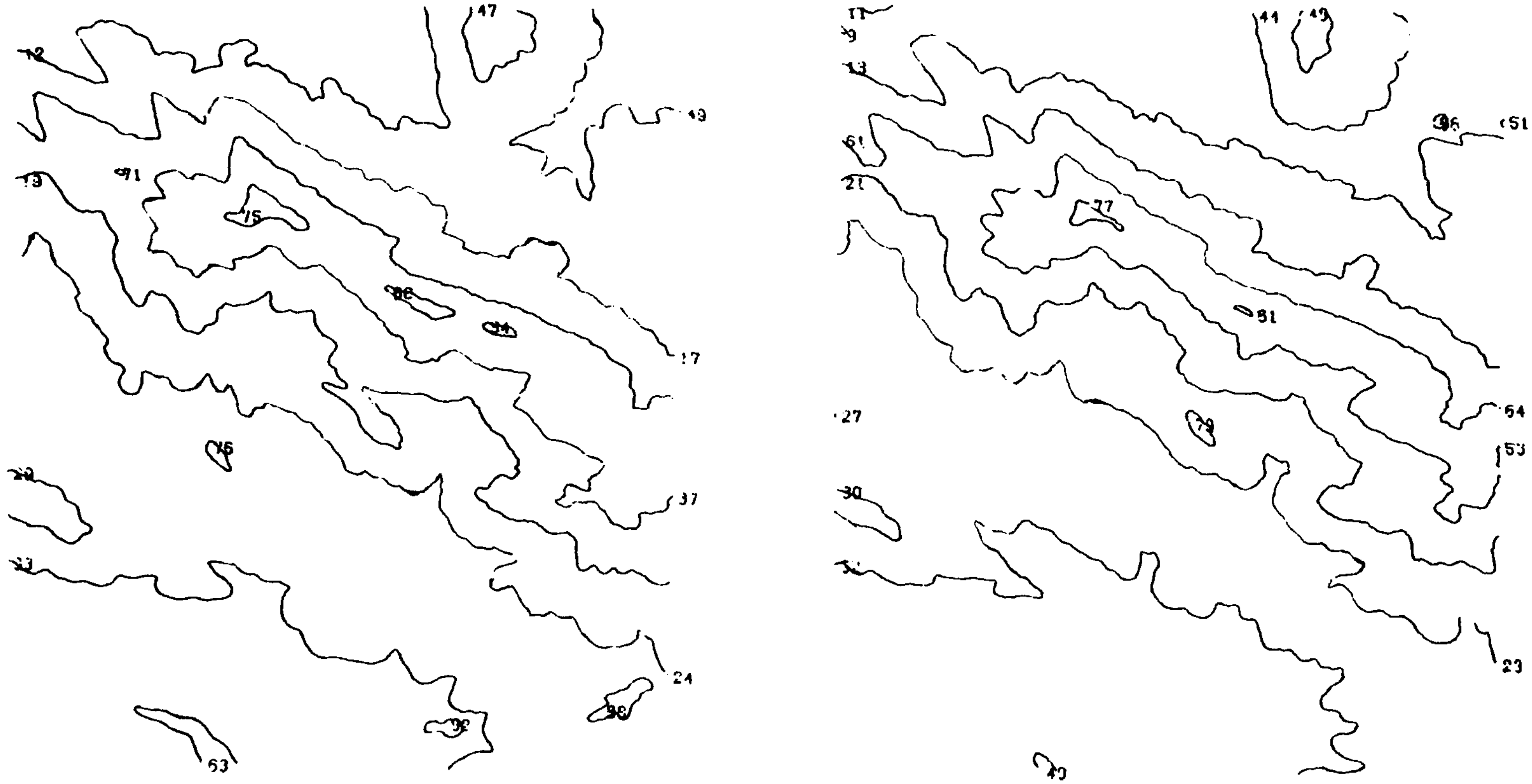
Figure 7-1. Photocopy of Source Shiraz Contour Data 1:1



INDEX

200 FOOT INCREMENT

Figure 7-2. Contour Plots Separated by Elevation Increment
Sheet 1 of 3

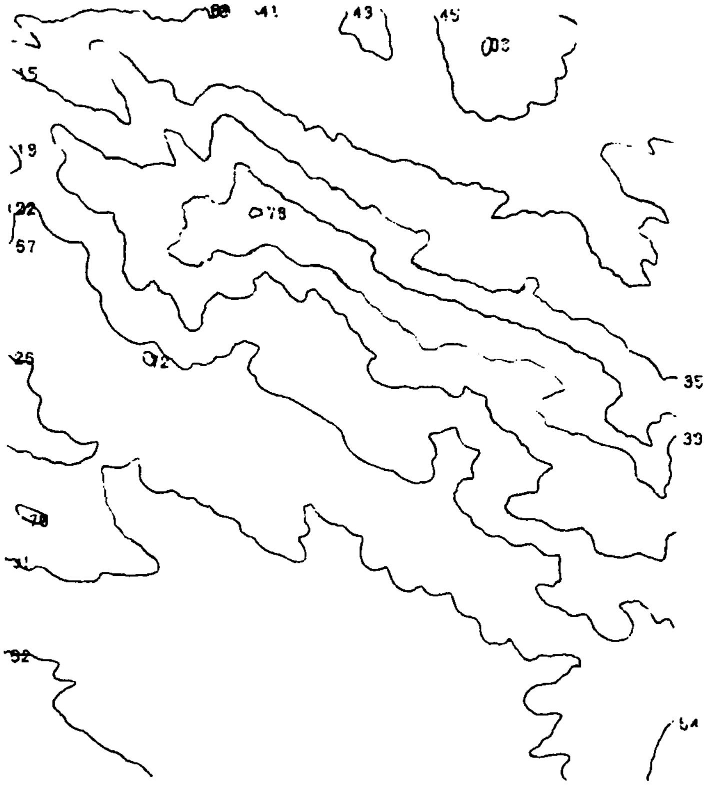


400 FOOT INCREMENT

600 FOOT INCREMENT

Figure 7-2

Sheet 2 of 3



800 FOOT INCREMENT

Figure 7-2 Sheet 3 of 3

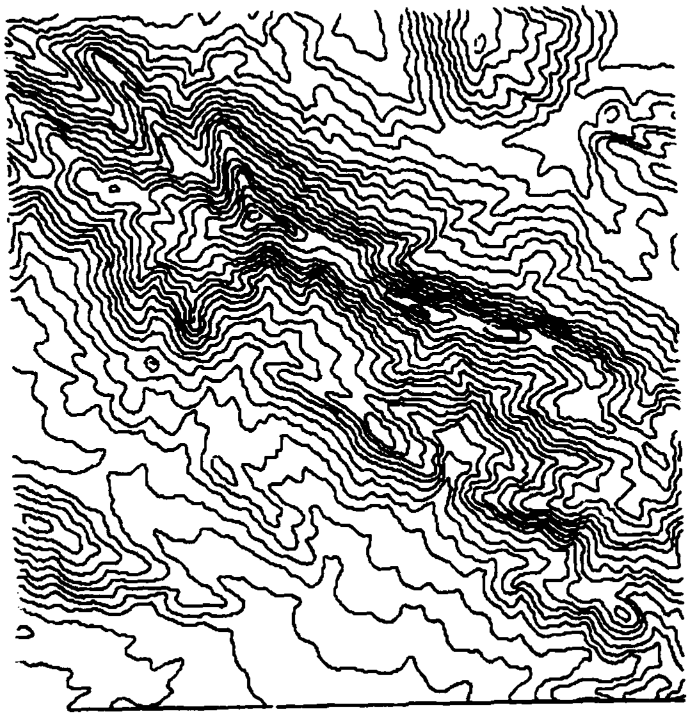


Figure 7-3. SHIRAZ Pseudo D.G.R. from Tapeout

(with the required fix) without restarting the whole package. Although this procedure is not always simple it does reduce the need to restart the process at the beginning.

To aid the user, documentation has been generated to identify all of the steps of the CONTAGRID procedure. Specifically, the job-control cards, the input/output of each process, along with valuable error identification messages have been provided (see User's Guides Parts I and II).

The COMTAL image display software has been developed at Goodyear to allow display and inspection of the results of gridding. This software accepts data in the "DMA standard for Digital Terrain Elevation Data" format. Figure 4-3 shows a color photograph of the 4"x4" test region.

Several support/utility routines were also generated which allow evaluation of a number of different raster-scanned data sets (see Appendix E).

Besides the above results, a number of other important issues have been investigated. Specifically, the need for on-line raster editing, and the possibility of excluding the need for line separation altogether.

APPENDIX A - CONTAGRID TOP-LEVEL STRUCTURE CHARTS & DATA FLOW DIAGRAMS

This appendix contains the top-level structure charts/data flow diagrams for the CONTAGRID software. These charts are comprised of two sections:

- . Contour pre-tagging & tagging (Figures A-1 and A-2)and,
- . Contour gridding (Figures A-3, & A-4)

Appendices B, C, & D describe the pre-tagging, tagging and gridding programs in more detail. Appendix E describes the utility routines developed during this contract.
Appendix F contains file/record layouts of the different data sets generated throughout the CONTAGRID software.

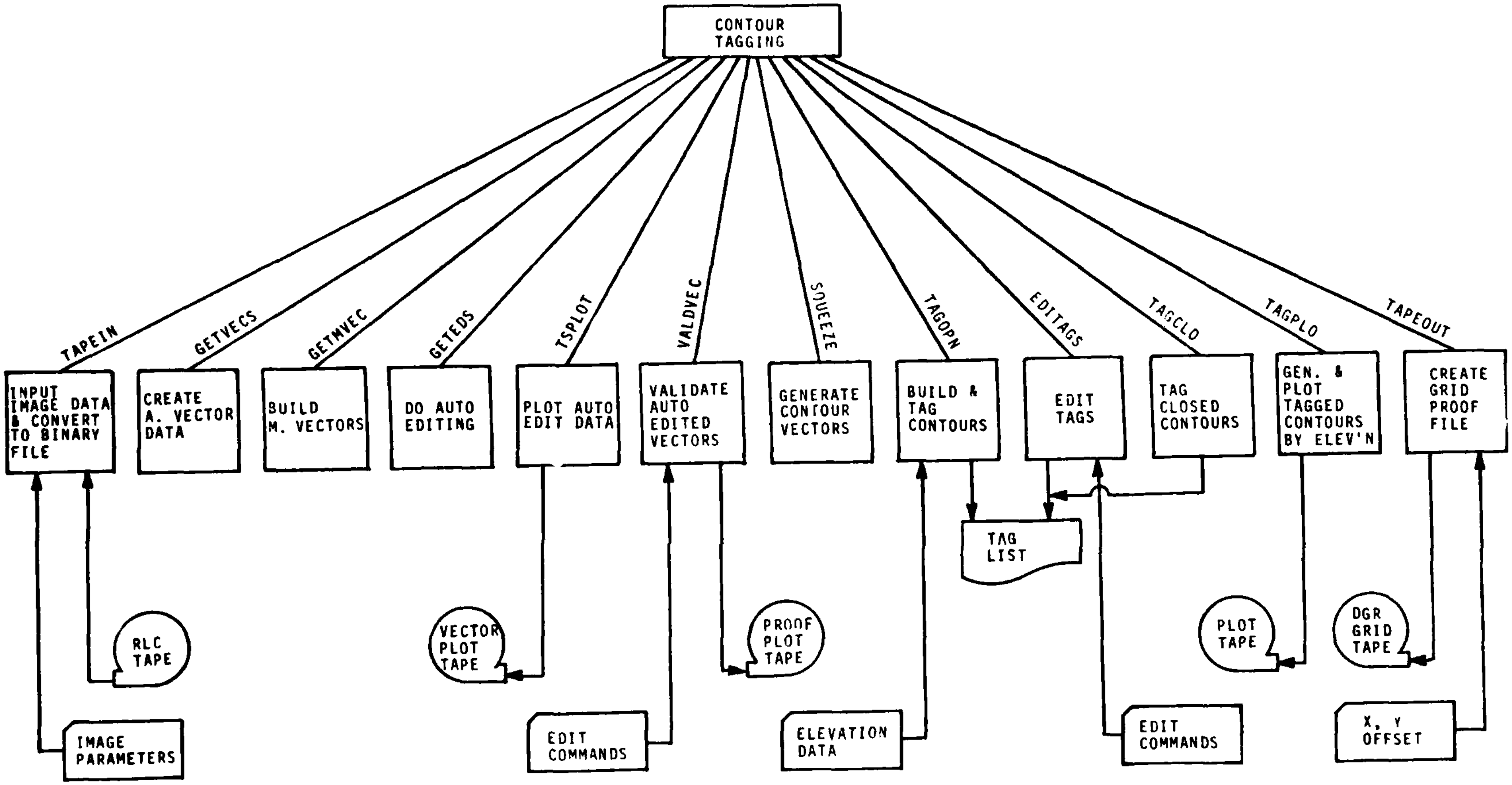


Figure A-1. Contour Tagging Top-Level Structure Chart

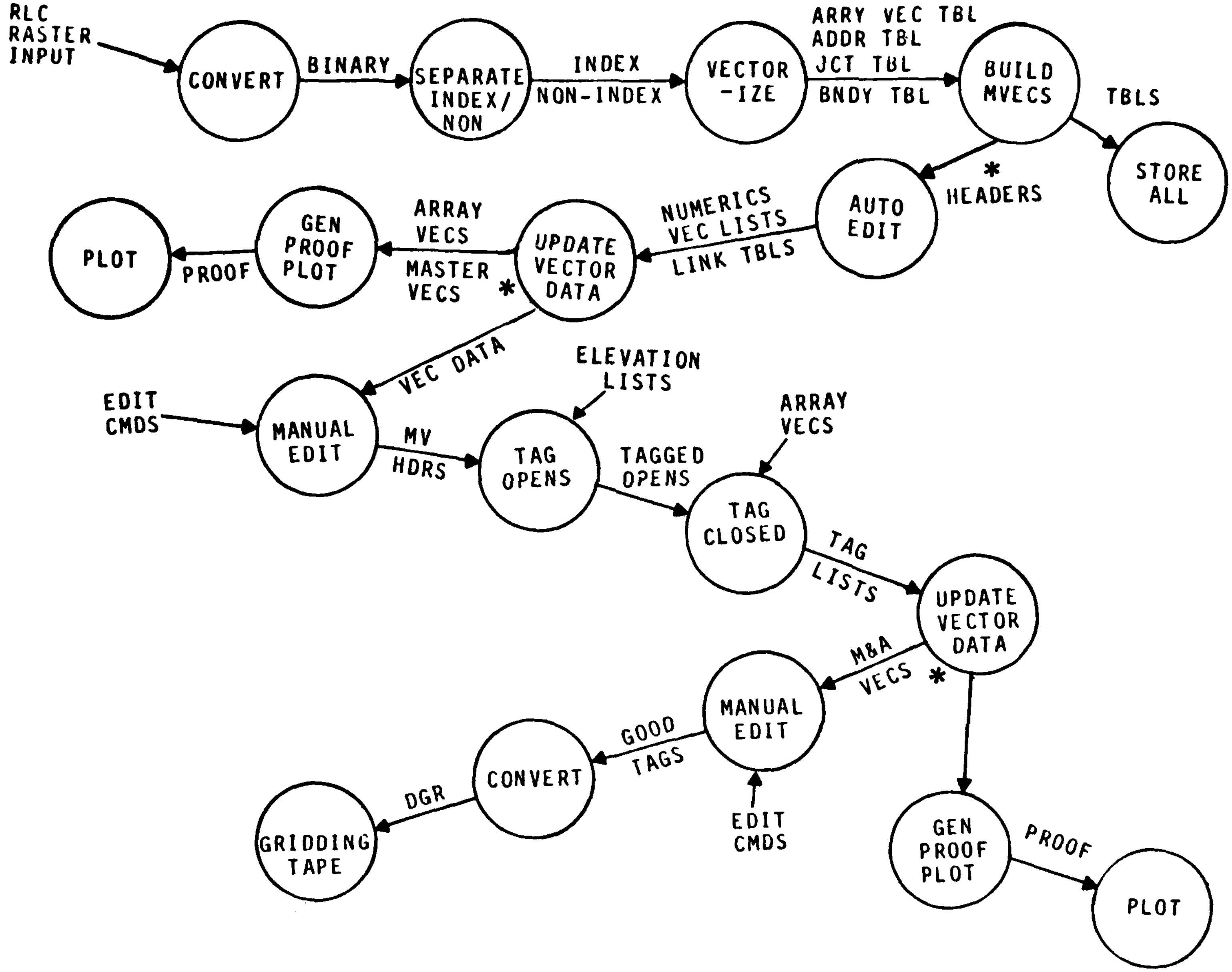


Figure A-2. Pre-Tagging and Tagging Data Flow Diagram

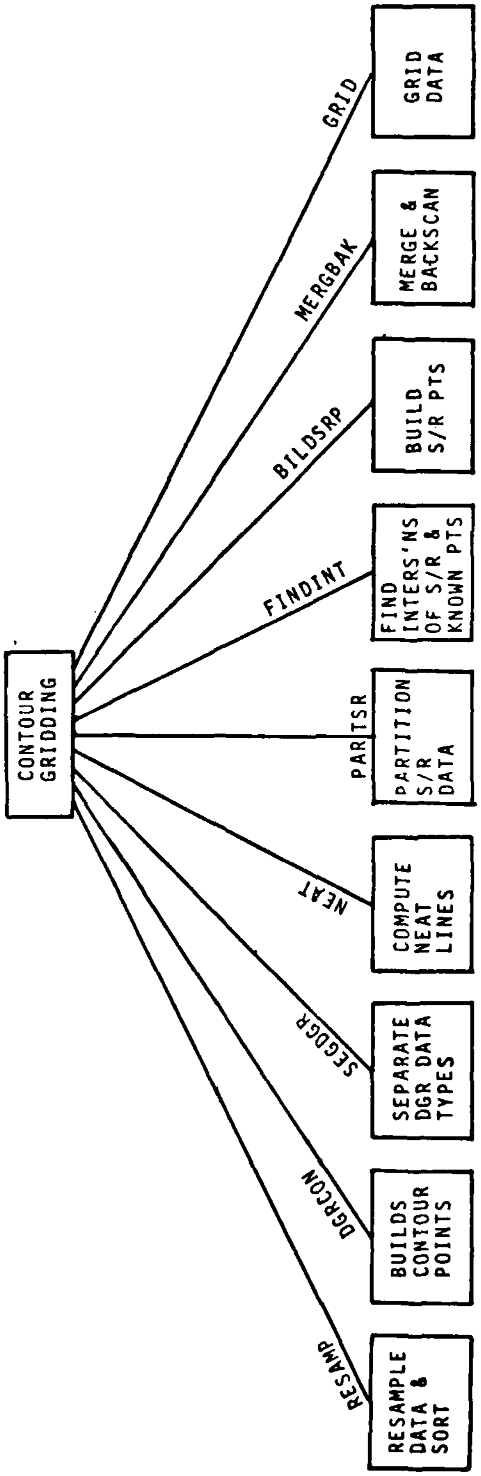


Figure A-3. Contour Gridding Top-Level Structure Chart

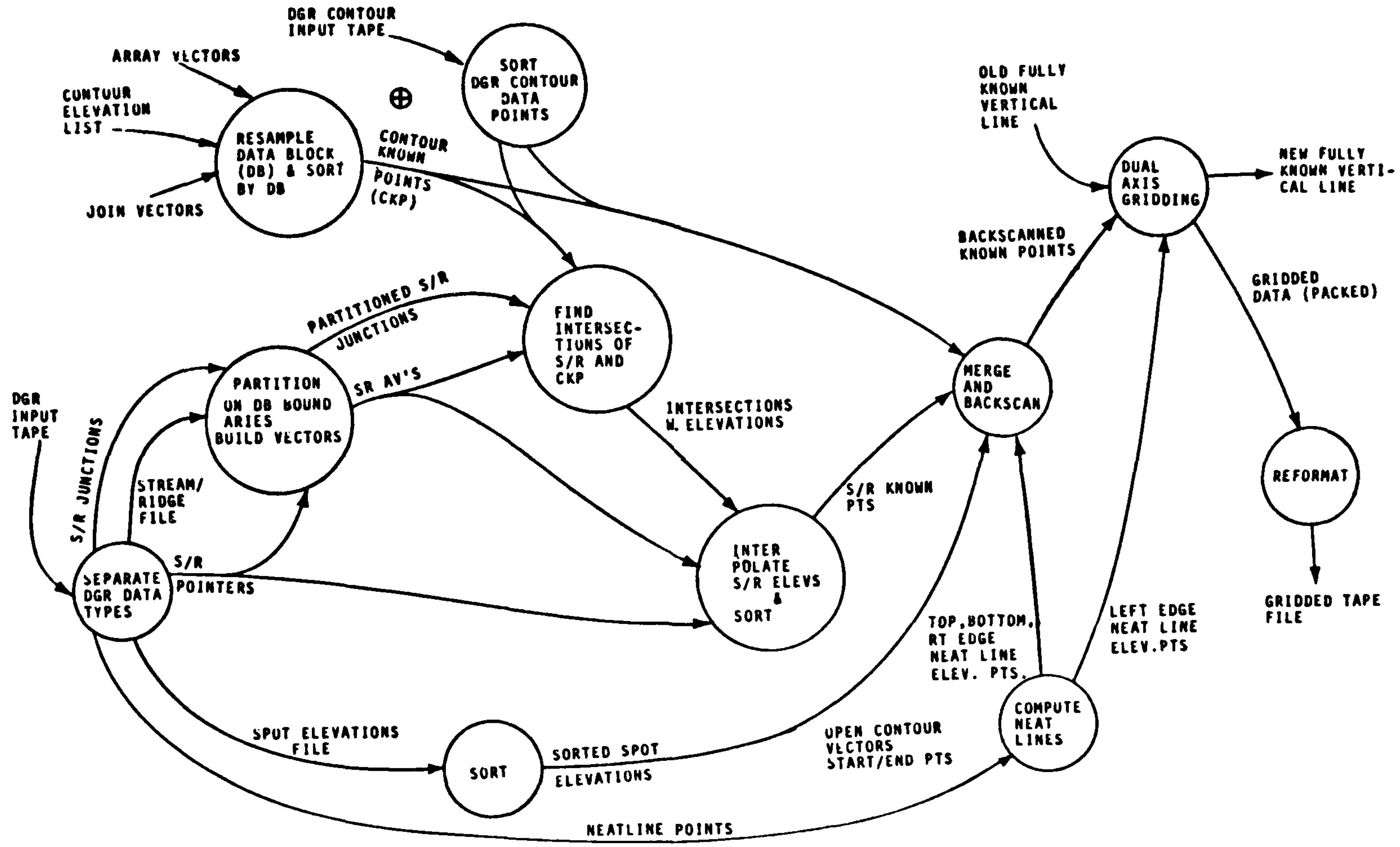


Figure A-4. Contour Gridding Data Flow Diagram

APPENDIX B - PRE-TAGGING

This appendix describes the most important parts of the pretagging software.

1. TAPEIN

This program consists of two programs.

- <u>IMAGE TAPE-DISK</u>: This routine copies the tape to disk (in the case of RAPS format, even & odd records are copied separately).
- <u>TAPEIN</u>: This routine merges the odd-even RLC files (if RAPS data) and generates a parameter file of approximately 120 words from the card input parameters shown in Figure B-1. Many of these parameters define RK05 disk sector addresses and STARAN buffer addresses which are used by the routines which convert the RLC data to it's binary equivalent (see GETVECS).

2. GETVECS

This routine uses STARAN to convert the raster data to vector data. Prior to this conversion, the RLC data is converted to it's binary equivalent one data block at a time. This data is segmented into iterations (called 'chunks' by the STARAN raster processing software) and off-loaded to the second RK05 disk.

Each iteration is loaded into the arrays and the conversion process takes place. Figure B-2 identifies the relationship between a tape record, data block, array load, and iteration.

The following processes are performed on each iteration:

- The binary data is loaded into the arrays.
- Line separation is performed to extract either the index, or non-index lines.
- The data is 'thinned'.

- INPUT DATA TYPE 1=ETL, 2=DMA
- RESOLUTION 1, 2, OR 4
- TOTAL # RECORDS
- SHEET LENGTH (INCHES)
- SHEET WIDTH (INCHES)
- NUMBER OF ARRAYS (STARAN)
- INDEX LINE WEIGHT (MILS)
- NON-INDEX LINE WEIGHT (MILS)
- INDEX INTERVAL (FEET/METERS)
- NON-INDEX INTERVAL (FEET/METERS)
- MAX ELEVATION (FEET/METERS)
- MIN ELEVATION (FEET/METERS)
- MAX NO. OF CHARACTERS IN ELEVATION NUMBER
- MIN NO. OF CHARACTERS IN ELEVATION NUMBER
- NO. OF RECORDS TO SKIP (WINDOWING)

Figure B-1. Tapein - Image Parameters

2. GETVECS - continued

- The data is then cleaned up, i.e., small specks are removed and line pips are deleted.
- Junction coding is performed.
- The data is vectorized and sent to the CDC.

All of the above steps (except for junction coding) are described in detail in Section 3 of ETL-0132 'ASSOCIATIVE ARRAY PROCESSING OF RASTER SCANNED DATA FOR AUTOMATED CARTOGRAPHY II. The array vector format has been updated to include the junction number (if one exists) and the type of the vector ends (see APPENDIX F, Figure F-1). The array vector linkage table is shown in Figure F-2. Junction coding locates junctions of three or more array vectors and assigns a unique junction number to each of the individual headers. If any junctions are found to exist on array boundaries the junction identification number is assigned to the first array vector found which is associated with this junction. If the other array vectors whose ends are common exist in a neighboring array or data block, the junction number is saved and then retrieved at the appropriate time.

3. GETMVECS

This program builds the master vector headers and the list of array vectors associated with each (see Appendix F Figures F-3, and F-4).

The basic sequence of steps is.

1) Read in index link table segment from disk
2) Create sub-master vectors for all segment entries
3) Repeat steps 1. & 2. until all master vectors have been created (from the sub-master vectors) and output to disk - 'dummy' headers and master vector linkages are stored separately.

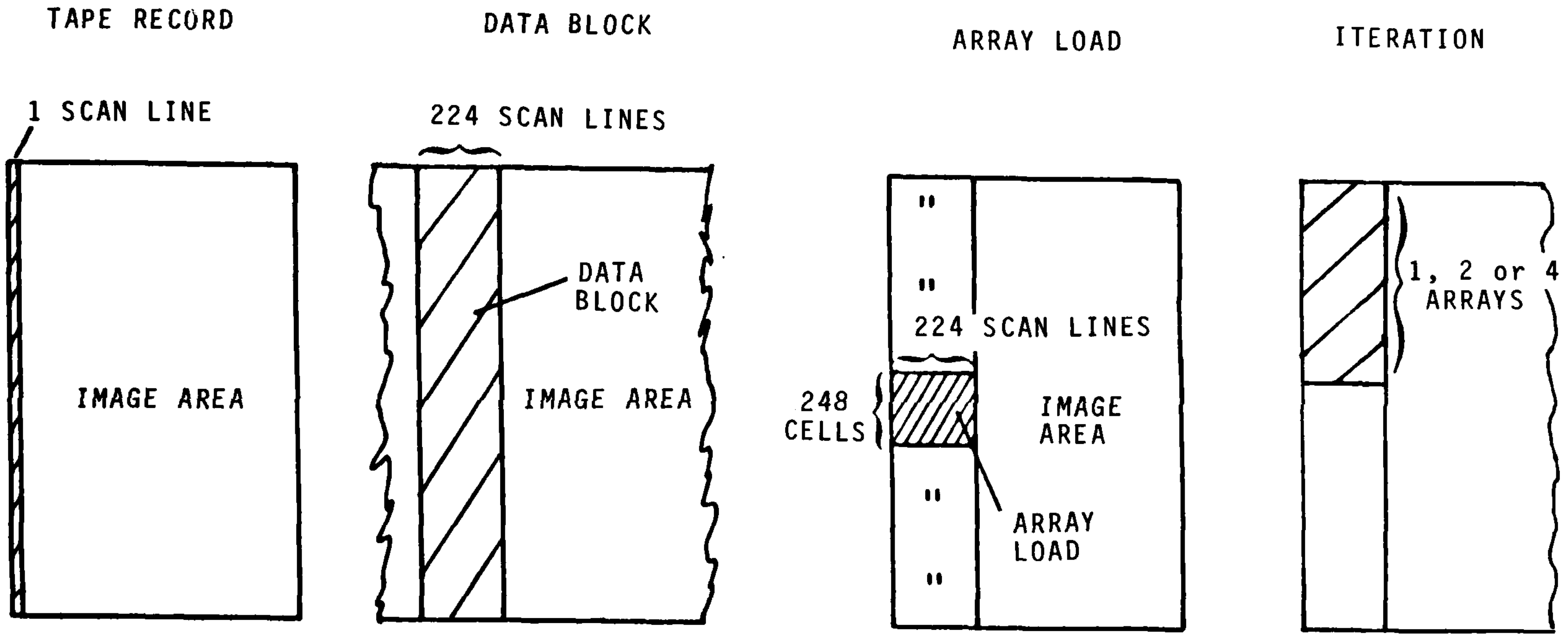


Figure B-2. Relationship of Image Area to Tape Record, Array Load, Data Block, and Iteration

3. GETMVECS - continued

   4) The real master vector headers are created from the dummy headers by filling in the information from the array vector tables.
   5) Steps 1-4 are repeated for the non-index lines.

At the time that the master vector headers are being developed those vectors whose ends are computed to be within a predefined framework around the edges of the sheet have their sheet boundary flags set.

4. GETEDS

This program detects and lists the numeric and special symbol vectors, as shown in Figures F-5 and F-6, attempts automatic editing using STARAN. The basic procedur϶ is as follows:

. Detect symbologies i.e.;

   Locate numerics, list the associated vectors for deletion.

   Locate depressions, remove the tic marks and link the master vectors at the junction (node) points.

   Identify cuts and generate the lists of ordered vectors which define each of the contour sections.

   Identify the fills and remove the tics.

. If no ambiguity arises, join the end points of vectors whose ends are deemed to be part of the same contour vector.

. Establish edit lists for both auto-edited symbologies and non-edited symbologies.

. Develop plots for editing symbologies.

. Utilize symbology I.D.'s to manually complete editing activity.

4. GETEDS - continued

Figure B-3 shows the various symbols. Following is a more specific description of the procedures for identifying these symbols.

Procedure for Identifying Numerics:

- Collect master vectors that have lengths like numeric vectors and are not node/node vectors.
- Use each collected vector as reference and find all other collected vectors in the proximity of the reference.
- Determine whether a reference group has a sufficient number of master vectors to meet the characteristics of an elevation number (group of numerics). If the group qualifies as an elevation number, the master vectors of the group are entered into a numeric vector list.
- When the group has excessive master vectors, a node merge routine is used to combine master vectors into numerics. If the numeric count can be brought within a range for the vector group to be categorized as an elevation number, the master vectors of the group are entered into the numeric vector list (as shown in Figure B-4).
- All master vectors listed as numerics are subject to deletion.

Procedure for Classifying 'Depressions', 'Cuts', and 'Fills'

- Identify tics: collect master vectors that are sufficiently short and that have both an open end and a node end.
- Collect groups of vectors that are adjacent via nodes. (A node/node master vector is used as a group starter).
- Groups with no tics are classified as "cuts".
- Groups with tics and that have node/node vectors of about equal length are classified as depressions.
- Remnant groups are classified as "fills."

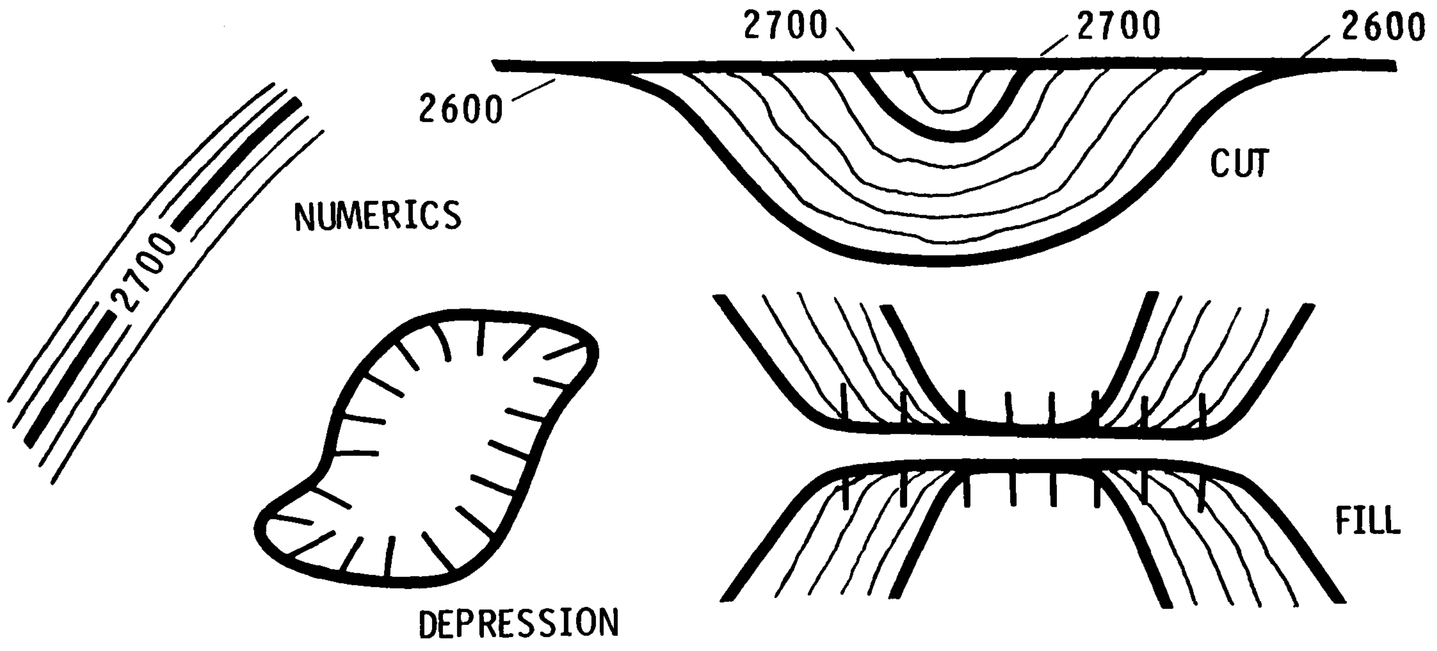
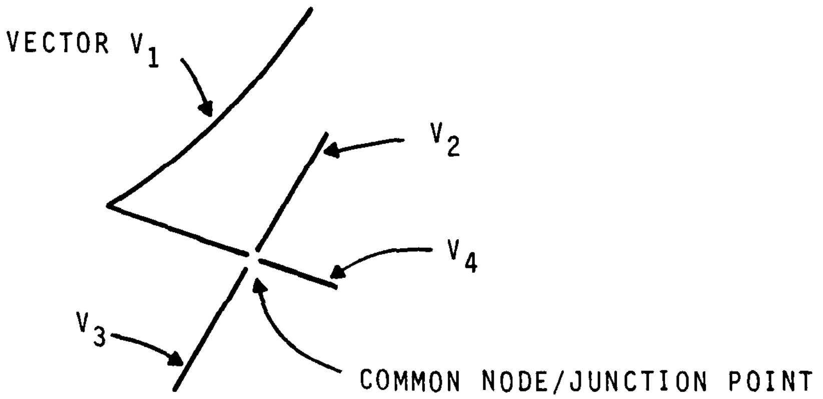


Figure B-3. Numerics & Special Symbols



VECTORS V_1, V_2, V_3, & V_4 REDUCE TO ONE VECTOR V_1

Figure B-4. Vector set Reduction for Numeric Detection

Procedure for Treating "CUTS" (see Figure B-5)

- For a cut group find a node pair A_1, A_{-1} connected by two different arcs, $a_{1, -1}$; $b_{1,-1}$ where the length of $b_{1,-1}$ and the length of $a_{1,-1}$ concatenate $a_{1,-1}$ and $b_{1,-1}$ to form 1st closed contour.

- Find 2 nodes nearest the pair of starting nodes (A_2,A_{-2}) and find the arc common to these 2 nodes ($b_{-a,2}$).

- Concatenate $a_{2,1}$; $a_{1,-1}$; $a_{-1,-2}$; and $b_{2,-2}$ to form the 2nd contour.

- Find 2 nodes nearest the nodes A_2,A_{-2} and not A_1,A_{-1} (Namely, A_3,A_{-3}); Find the arc common to these 2 nodes ($b_{-3,3}$).

- Concatenate $a_{3,2}$; $a_{2,1}$; $a_{1,-1}$; $a_{-1,-2}$; $a_{-2,-3}$; $b_{3,-3}$ etc.

- Continue until search for groups fails.

- Generate edit list and identify elements.

Procedure for Line Join (Index or Non-Index)

- Collect start/end point values of open-ended index and non-index lines.

- Group start/end points within window.

- Pair ends within window.

- Generate array vector, master vectors and linkage information using STARAN.

- Send lists to CDC.

- Update existing lists with new information.

GETEDS also generates a list of unidentifable vectors together with a list of unpaired end parts to be used by the manual edit procedure VALDVEC. (See Tagging User's Guide.)

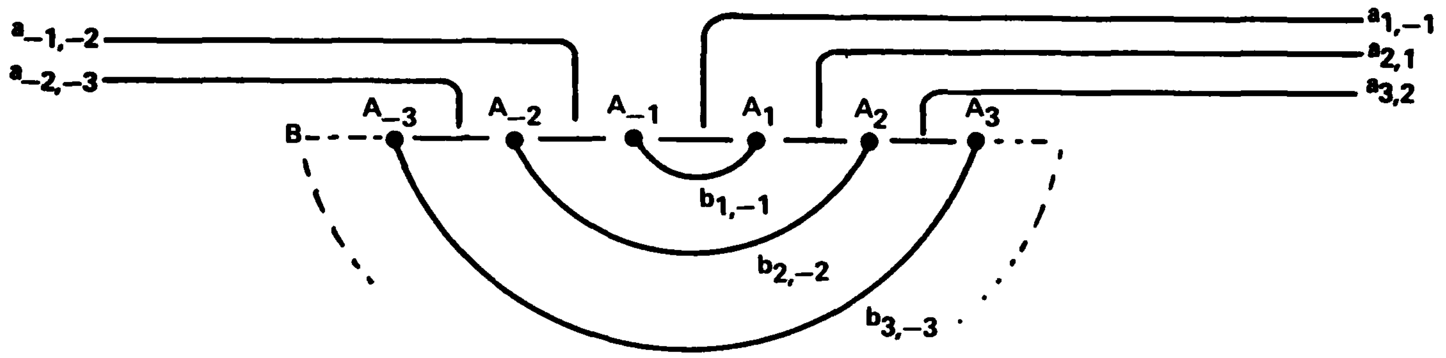


Figure B-5. Cut-Vector Grouping Procedure

5. PLOTGEN

This program builds a plot tape (for CALCOMP) with the following plots from the correlation lists generated by GETEDS.

Vectors of index weight

- Contour vector segments
- Joins

Vectors of non-index weight

- Contour vector segments
- Joins

Numerics

Special Symbols

- Depressions
- Deep cuts
- Fill/cuts

6. VALDVEC

This program uses the manual edit commands, shown in Figure B-6, to 'clean-up' the various vector files. If joining is required STARAN is used in a similar manner to GETEDS to generate the new master vectors and associated array vectors. Plots are generated of the resulting edited files for perusal by the user. Several iterations through VALDVEC may be required before the various data-sets are completely edited.

7. SQUEEZE

This program builds the contour vector headers and associated array vector lists (see Appendix F - Figure F-7, F-8) from the master vectors (join and non-joins). Each contour vector consists of one or more master vectors and exists between two points on the sheet boundary or as a closed loop within these same bounds. If one of these two conditions does not exist then a message is generated for the specific contour vector which identifies it as a'dangling' vector. Such a message

7. SQUEEZE - continued

is generated for the specific contour vector which identifies it as a 'dangling' vector. Such a message requires that the user re-edit the specific contour vector to fix (join) the 'dangling' end to another vector. The squeeze process may then be repeated.

- CHAnge FROM TYPE TO TYPE FROM #
- DELete TYPE #, #, #, #, #,
- JOYn TYPE #, #
- REOrder TYPE #, #, #,

TYPE = INDEX, NON, DEPRESSION, CUT, FILL, # NUMBER

Figure B-6. Manual Edit Commands

APPENDIX C - TAGGING

This appendix documents those programs that perform the actual tagging of the contour vectors, namely:

- . Open tagging (TAGOPN),
- . Editing of the tagged vectors (EDITAGS), and
- . Closed tagging (CLOTAG).

1. TAGOPN

This program attempts the tagging of all contour vectors whose ends are identified as existing at the sheet boundaries and are not manually tagged as input to this process (open index contours).

This program performs the open tagging utilizing the manually identified, ordered set of sheet boundary contour vectors and *input (by cards) in one of the four sheet boundary data sets.* STARAN is used to perform the tagging. Following is the set of steps required to perform the tagging. Figure C-1 shows an example of a portion of a map sheet with border intersections.

First, the open index contours are tagged:

<u>Requirements/Assumptions</u>

- . All index contours exist in a file for such contours.
- . The data set for an index contour includes an ID and start/end point coordinates.

<u>Procedure</u>

- . Proceed in a counterclockwise direction around sheet border and enter an elevation number into their elevation list for each index contour/sheet border intersection encountered (manual).
- . Enter elevation list into the processing system (manual) via cards.

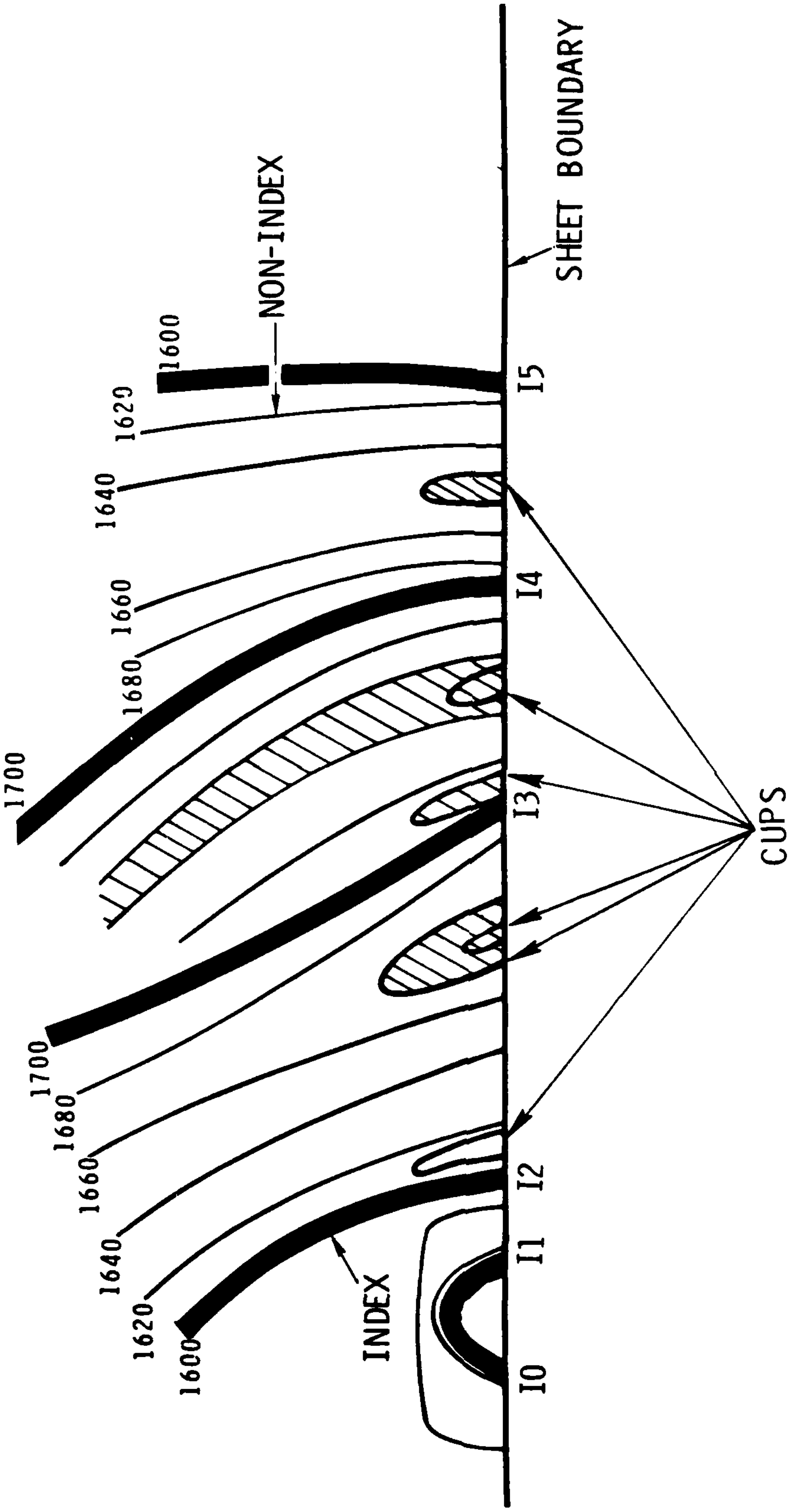


Figure C-1. Contour/Sheet Border Intersections

- Using the file of index contours and the end/start point coordinate data for the contours, determine the index contours that intersect the sheet border (STARAN).
- Sort intersections into west, south, east, and north boundary intersection sets (STARAN).
- Order four contour intersection lists that contain contour ID and intersection coordinate data, as follows (STARAN):

West:	Ascending Y
South:	Ascending X
East:	Descending Y
North:	Descending X

- Concatenate the west, south, east, and north contour intersection lists and associate with the elevation list by order. Report association failure if it occurs (STARAN).

After the steps above have been accomplished satisfactorily, the non-index contour tagging process is begun:

Requirements

- Files for both open indexed and open non-indexed contours exist.
- All open index contours have been elevation tagged.

Definitions

- Between two adjacent index contour/sheet border intersections, the elevation change is up or down one "big step" or zero; between two adjacent non-index contour/sheet border intersections, the elevation change is up or down one "little step" or zero.
- N "little steps" = 1 "big step".
- Between two adjacent index intersections, the index interval, a non-index contour may intersect zero, one, or two times only. (An open non-index contour that intersects the neat line within the index interval twice is defined to be a cup.)

Procedure

- Establish a merged list of index and non-index sheet border intersections (STARAN); (list intersections in the order that they appear in going counterclockwise around the sheet boundary. Make the top west intersection the first list entry). Send list to CDC (required for closed tagging).
- "Remove" all cups from the intersection list (STARAN).
- Identify all index intervals for which all the following conditions are true (STARAN):

  N-1 non-index intersections exist in the interval.

  No intersections are depression flagged.

  One "big step" change occurs across the interval.
- Establish elevations for non-index intersections of all qualifying index intervals (STARAN).
- Put all "cups" back into the intersection list (STARAN).
- Elevation tag non-index intersections adjacent to index intersections according to slope at index intersection (STARAN).
- List elevation file (tagged & untagged) (see Figure F-9).

2. EDITAGS

This program changes/inserts elevations for specific contour vectors from cards. It then updates the corresponding files and prints out the new elevation list. STARAN is not required for this processing.

3. TAGCLO

This program uses STARAN to tag the closed contours.

Requirement

- All boundary contours have been elevation tagged.
- No contours are broken internal to the sheet.
- All closed contours have a depression flag.

Assumptions

- A North-South line drawn across the sheet will intersect (in general) both elevation tagged and non-tagged contours. If the elevation at one intersection is known, the elevations of all other intersections can be found.
- Upon elevation tagging the intersections of a N/S line, fewer contours of a sheet remain non-tagged; by utilizing sufficient N/S lines and tagging ever more contours, all contours will ultimately be elevation tagged.
- A contour tagged along one N/S line is likely to be tagged along other N/S lines. Redundant tagging is desirable for it provides means for automatically checking the validity of the tagging procedure.

Tagging basis

Two flags are required to define the southward pointing elevation slope at the point of intersection of an untagged (closed) contour and the North-to-South cutting lines. The first flag, the depression flag (DF), is associated with the contour. When DF=1 it indicates the contour is a depression contour; when 0, it isn't. At entry to the tagging process, the condition of the flag is known for all closed contours.

The second flag is associated with each intersection of the N/S cutting line and a given contour. It must be computed for each new N/S line. Beginning at the northern end of a cutting line, the intersections of a given contour are counted. The crossing flag (CF) simply indicates whether or not an intersection is even or odd. CF=1 indicates an odd intersection. When both DF and CF are known, it is implicitly known that the southward pointing elevation slope at an untagged intersection is rising or falling. If the slope flag (SF)=1, it indicates that the southward pointing slope is rising; when SF=0 it is falling. Note that SF is the "exclusive or" of DF, CF. By comparing the SF flags of 2 neighning intersections, A and B, where A is north of B, the magnitude of the elevation step from A to B is given by the complement of the "exclusive or" of SF_A, SF_B. Its 2's complement sign is given by the "and" of SF_A, SF_B,. By adding up all elevation steps, the relative elevation of all steps can be found. By adding a bias to the relative elevations so as to cause the correct elevation at a point of known elevation, all intersections are assigned unbiased elevations. The newly found elevations of contours are checked with those found previously in order to pin-point inconsistencies.

Treatment of boundary contours

The previous discussion neglects to indicate how the boundary contours are able to be treated liked closed contours. Since the closed contour tagging process demands that all contours be closed, it requires the closure of all boundary contours. Thus, for boundary contours, the two contour/map boundary intersection points are connected with a line external to the map region. The intersection point of a contour nearest the top west corner of the sheet, as measured in a clockwise sense, is connected to the second intersection point by drawing the external line counter clockwise until it touches the second intersection point. (See Figure C-2.) Because the elevations of boundary contours are known at the neat lines, the "effective" depression flag of each artifically closed boundary contour is able to be established.

The cutting line process presumes that the starting point of the N/S line begins outside of the region of closed contours. In practice, the most noteworthy terminus of the N/S line always lies on the North neat line. Thus, the terminus point may well lie inside some of the artifically closed boundary contours. As a result, the CF flag must account for the fact that the crossing count is low by 1 for such boundary contours.

When treating real data, provision must be made to account for a whole variety of anomalies (e.g., supplementary contours, changed contour increments, vectors used redundantly (as in the case of cuts and fills)). The tagging software structure allows for the inclusion of the vector flags and parameters into the vector headers so that such anomalies can be accommodated.

The exact geometry of the N/S cutting line is arbitrary. At present, the N/S lines follow the paths shown in Figure C-3. By using the lines as shown, only very small contours that fit within the grid can escape being tagged (≈ .160" by ≈ .184). By including spur paths to all small contours, all closed contours can be tagged.

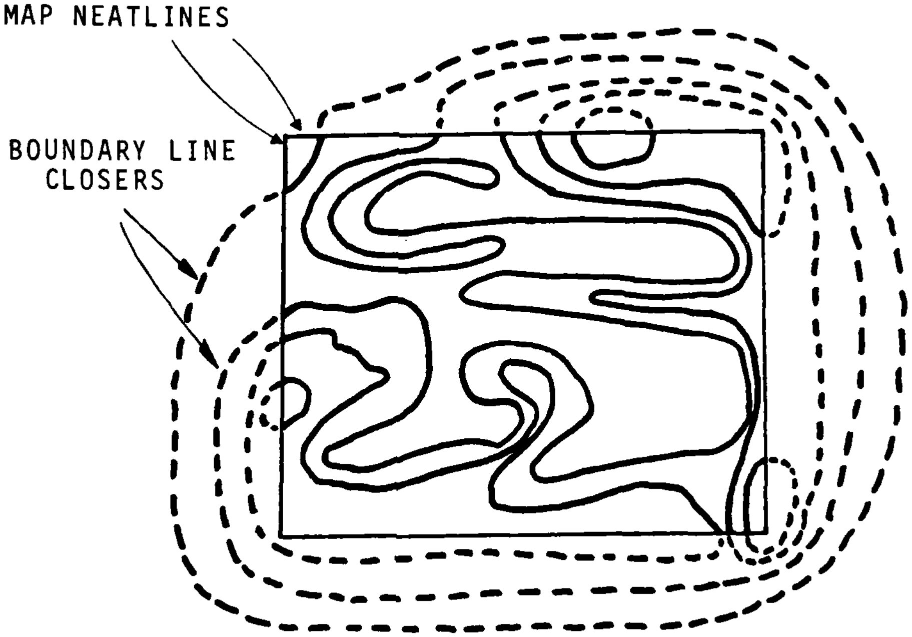


Figure C-2. Boundary Contour Closing

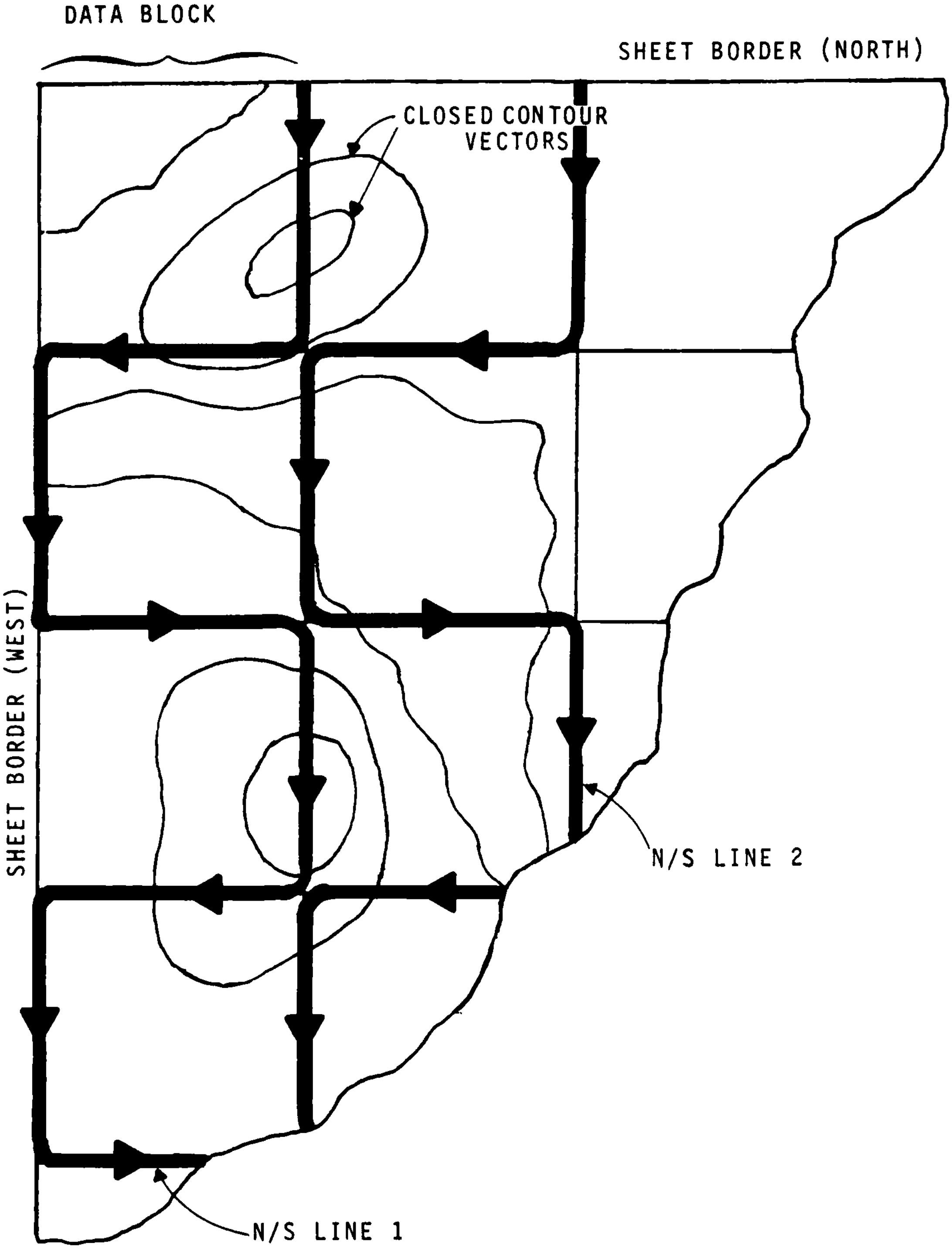


Figure C-3. Top Left Corner of Sheet/Data for Tagging

APPENDIX D - GRIDDING

This appendix describes the gridding processes.

1. RESAMPLE

This program uses STARAN to generate/convert the contour vector data set produced from the contour tagging process into the DGR grid coordinates.

The CDC software sends parameter information to the STARAN programs along with the array vector data on a data block basis. The array vector data is the merged data from the four files, namely: 1) the index, 2) non-index, 3) join-index, and 4) join non-index files.

The STARAN software converts the above data to the required 10-mil grid and sends the data (in buffer loads) to the CDC which builds resampled blocks of this information and saves them on the disk.

2. DGRCON

If the input vector dataset to the gridding process is not the contour tagging output, then this program is used to build the file of contour information from the DGR formatted input tape.

This process consists of two routines: 1) SEGCON and 2) BILDCON. The SEGCON segregates the contour data from the other DGR data and saves this file. The BILDCON sorts this file and separates the information into grid scan line (GSL) format.

3. SEGDGR

This program reads the DGR formatted magnetic tape which contains the supplementary image data and segregates the data into four types:

- Ridge/stream (R/S)lines (Figure F-10)

- Spot elevations (Figure F-11)
- R/Sjunction marks, and (Figure F-11)
- Neatline points (Figure F-12)

The STARAN is not used for this processing.

4. NEAT

This program uses the open boundary contour vector information, and the neatline points from the previous program to generate the four neatlines bounding the image to be gridded (see Figure F-13). STARAN is not used for this processing. The process is performed using a linear interpolation between successive given points for each of the four lines.

Corner point values are computed by a linear interpolation around the corners.

5. GETSRKP

This program makes use of the ridge/stream (R/S) points, the contour known elevation points, and the RS junction points to find the R/S intersections. It then interpolates along R/S lines to get all R/S points of known elevation. It is divided into three routines: 1) PARTSR, 2) FINDINT, and 3) BILDSRP. The PARTSR performs the partitioning of the R/S lines and junction marks along the scan block boundaries (see Figures F-14 & F-15). The FINDINT uses the STARAN to find the R/S intersections points (see Figure F-16). The BILDSRP interpolates between the R/S intersections to produce R/S points of known elevations. The points are then sorted (see Figures F-17 & F-18).

6. MERGBAK

This program merges all the points of known elevation into one file and converts it to a 'backscanned file' (in reverse scan block order) in preparation for gridding (see Figure F-19).

7. GRID

This program uses STARAN to grid and build an output file from the neatline and backscanned data.

The gridding process utilizes a dual-axis algorithm implemented in a parallel fashion on STARAN.

The DAPGAC develops elevation points at the mesh points of a grid in a manner somewhat similar to the manner of the DMA Planar Interpolation Gridding algorithm. Elevations for columns of grid points are developed one column at a time beginning at the left-most column and proceeding to the right-most column. Unlike the Planar Interpolation Gridding algorithm, which needs to develop the elevations of the grid points of a column of such points one point at a time, the new algorithm develops the elevations at all the grid points of a column independently.

Ideally, to determine the elevation at a grid point, the known independent points nearest to the points of interest should be used. Hopefully, the points are well distributed. In practice, much less information is used as will be seen in the discussion tnat follows.

Assuming a cartesian coordinate system with the origin at the grid mesh point to be interpolated, the Planar Interpolation Gridding algorithm uses processed output data to get the influence of input points of quadrants 1,2,3 and uses input data only from quadrant 0 (exclusive of y axis points). The computation involves one input point (the closest point) and two processed points. Over the line defined by the input point and the desired mesh point, a linear interpolation is performed to determine the elevation at the mesh point. (The 2 processed output points are used to determine the elevation (via linear interpolation) at the intersection of the aforementioned line and the line between the two processed points. This elevation data and intersection data allows the linear interpolation to proceed.) The Planar Interpolation Gridding algorithm provides a one axis interpolation procedure.

The DAPGAC uses processed output data to get the influence of points in quadrants 1, 2 (the left half plane minus the y axis points) and uses input points from quadrants 0, 3 (the right half plane including y axis points). (See Figure 1.) Three input points, rather than one, are used to determine the elevation at a grid point, namely:

1) the input point nearest to the North,
2) the input point nearest to the South, and
3) the input closest point (or the point nearest to the East if the closest point is to the North or South).

The first two points lie on a N/S line (the y-axis) through the grid point for which the elevation is required. A linear interpolation develops the 1st *N/S axis estimate for the grid* point elevation, the vertical estimate h_V, according to the rule:

$$h_V = \frac{d_N\ h_S + d_S\ h_N}{d_N + d_S}$$

where

1) d_N, d_S are, respectively, the distances to the North, South points, and
2) h_N, h_S are, respectively, the elevations of the North, South points.

The 3rd point, the "closest" input point in the right half plane, and the processed points of the left half plane are used to make a second axis estimate of the elevation at the grid point. A line running through the 3rd point and the grid point establishes the 2nd axis for interpolation and determines the processed points

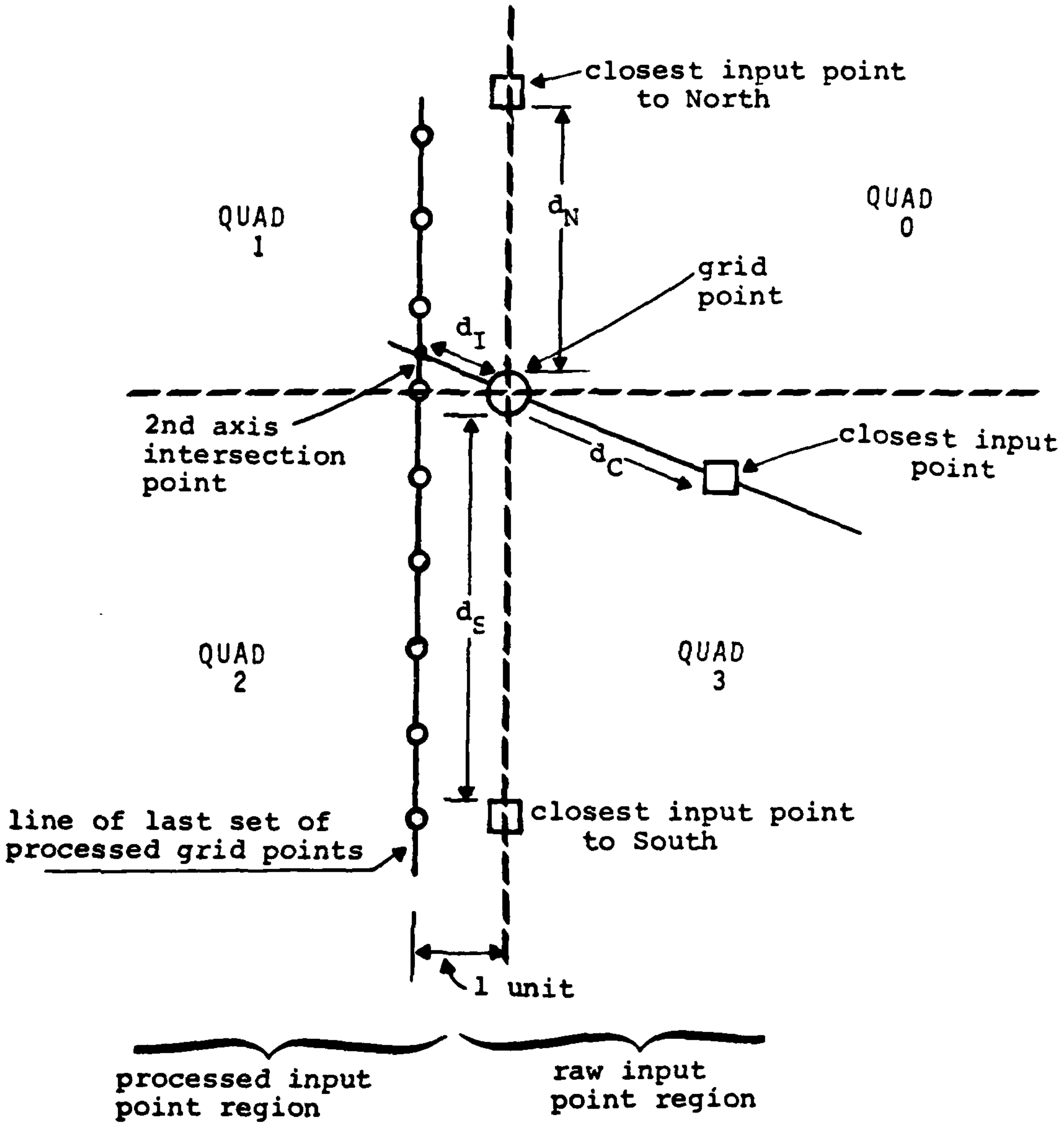

closest input point to North
QUAD 1
QUAD 0
d_N
grid point
d_I
2nd axis intersection point
d_C
closest input point
d_S
QUAD 2
QUAD 3
line of last set of processed grid points
closest input point to South
1 unit
processed input point region
raw input point region

Figure D-1. Processing for One Grid Path needed by the 2nd axis interpolation routine. In particular, the two processed output points required are those that lie along the vertical line at x=-1 (and, thus, belong to the most recently developed column of processes points) and are nearest to the intersection of the 2nd axis and this vertical line. The processed points are used to find the elevation at the intersection point via linear interpolation. When this elevation is established the 2nd axis elevation estimate for the grid point, the closest point estimate, h_C, is computed using the rule:

$$h_C = \frac{d_I \; h_C \; + d_C \; h_I}{d_I + d_C}$$

where

1) d_I,d_C are the distances to the intersection point and "closest" point, respectively, and

2) h_I,h_C are the elevations of the intersection point and "closest" point, respectively.

To get the best elevation estimate for the grid point, h_B, the two elevation estimates, h_V and h_N are combined according to the nearness of points leading to the respective estimates according to the rule:

$$h_B = \frac{\left(\frac{1}{d_V}\right) \; h_C + \left(\frac{1}{d_N}\right) \; h_V}{\left(\frac{1}{d_V}\right) + \left(\frac{1}{d_N}\right)}$$ where

d_V is the smaller of d_N or d_S and

d_N is the smaller of d_I or d_C.

Because the DAPGAC uses more input points to make grid point elevation estimates, it should provide better elevation data than can be provided by the Planar Interpolation Gridding algorithm.

APPENDIX E - UTILITY ROUTINES

Figure E-1 shows an example of the input and output of the utility routine which enables the viewing of a section of a Run Length Coded data tape in DMA RAPS format. This routine enables the user to specify:

1) The characters which define the contour line (in this example,*),
2) The area (n x n pixels) which are defined by each character,
3) The threshold; i.e., the minium number of 'ON' pixels in the n x n area to switch-on the predefined character, and
4) The starting x,y points of the plot relative to the origin.

The routine was used to determine the orientation/origin of the input and the quality of the scanned material.

Figure E-2 shows the corresponding FORTRAN listing for this routine.

```
?PLOT.
 SELECT INPUT FILE 1 OR 2 AND SQUARE SIZE.
?2,5
 SELECT MARK AND SPACE CHARACTERS FROM THESE:
 =+X*0 .-
 12345678   ENTER digit,digit
?4,6
 ENTER TERMINAL WIDTH(CHARS) AND IMAGE FILE SIZE(MAX KEY)
?130,6930
 ENTER START X,Y, AND SWATH WIDTH DELTA X, ALL IN INCHES WITH DECIMALS
?.5,5.,.65
 THAT DELTA X WILL GIVE YOU   1.1 PAGES.
ENTER THRESHOLD IF THIS SET UP IS OK.

?6
 RECORDS   500 THRU  1149, PIXEL 5000 FOR   650PIXELS.
          <--TOP--<<
```

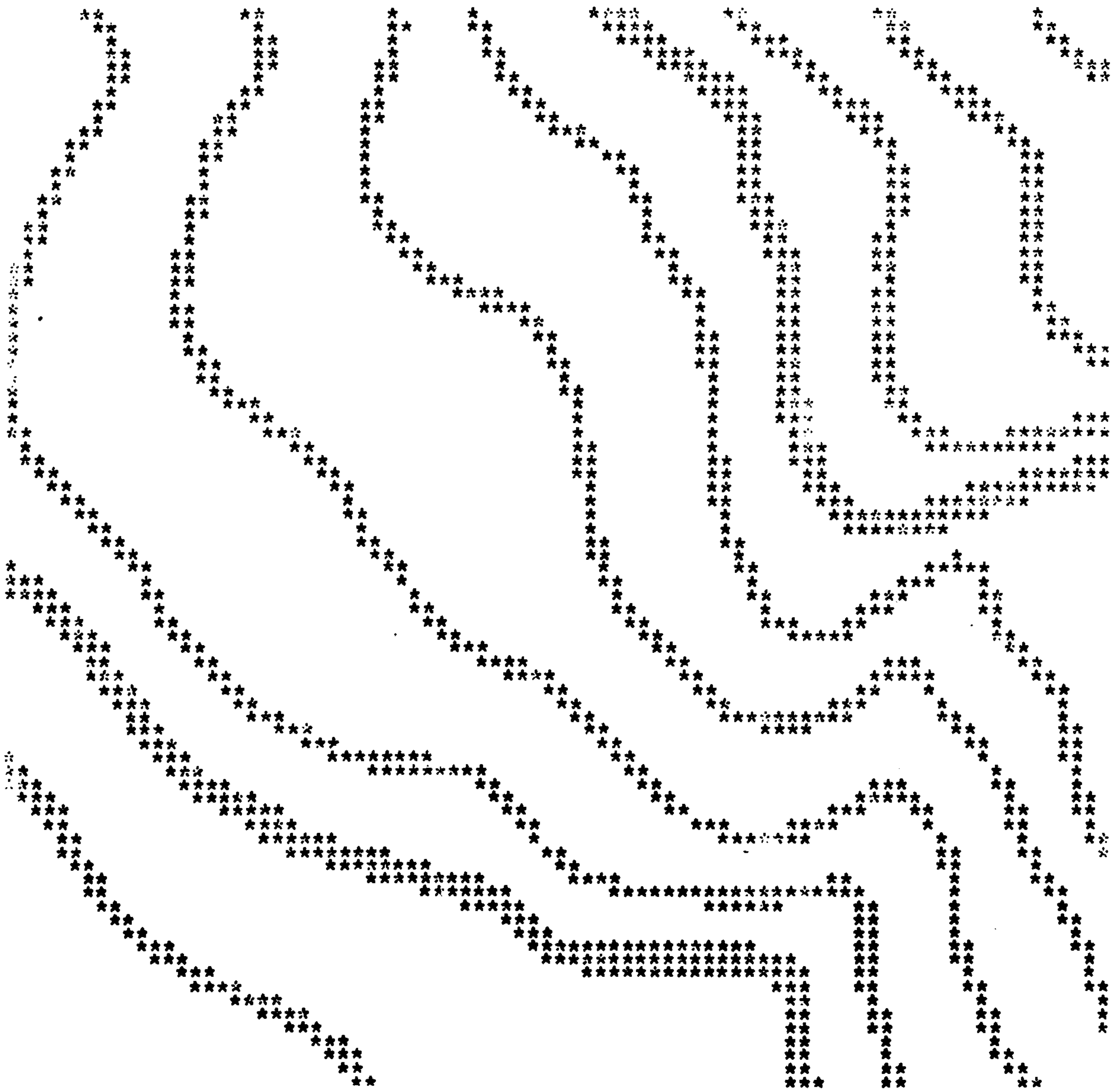


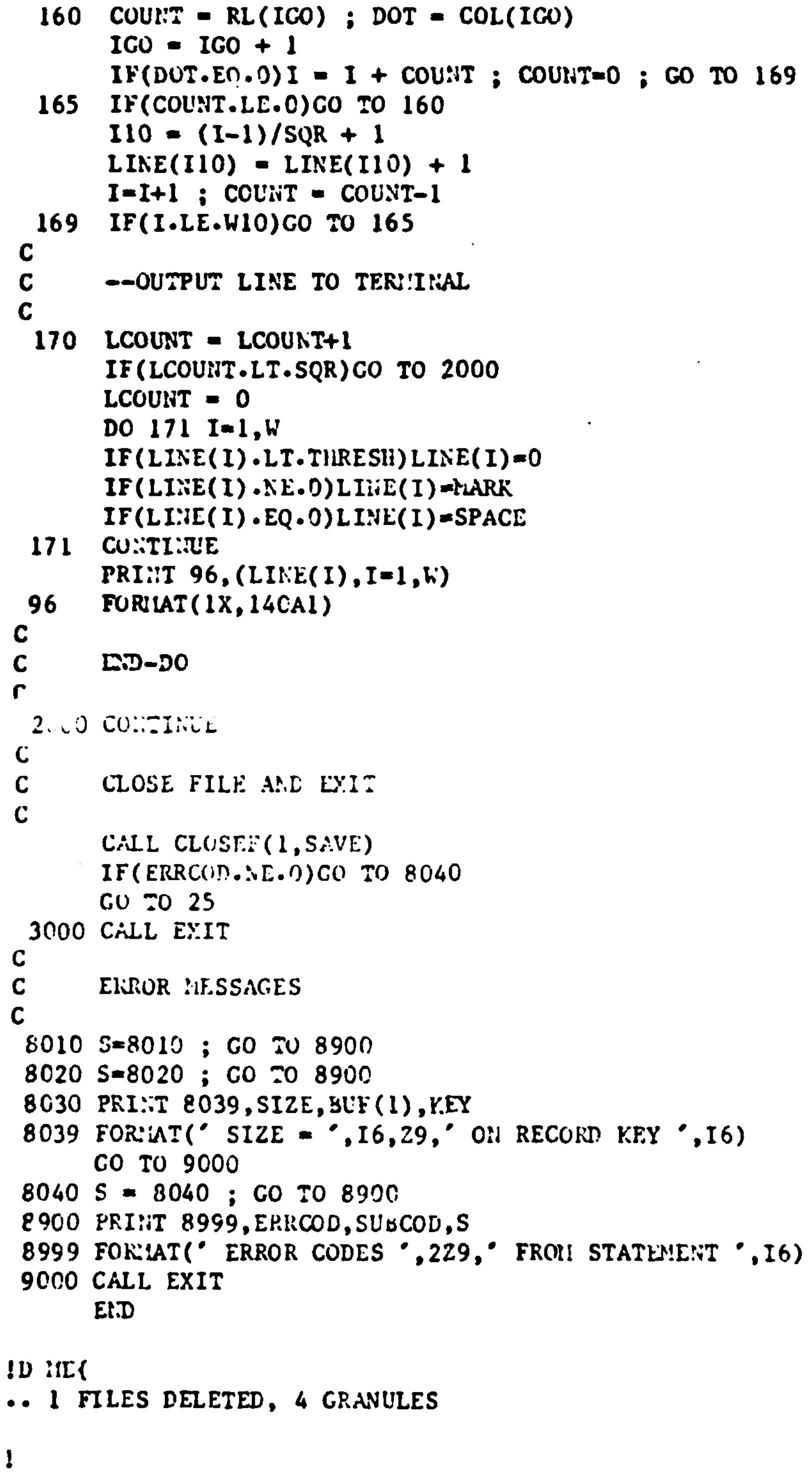
Figure E-1. Raster Display Utility Routine

```
C     QPLOT.
C
      IMPLICIT INTEGER (A-Z)
      INTEGER BUF(100),LINE(141),CHARS(8),RL(400),COL(400)
      INTEGER IN77(2),INNEW(2),NAME(2)
      REAL RX,RY,RDELX,RPAGES
      DATA IN/2/,ACCT/0/,PASS/0/,KEYED/2/,DIRECT/2/,DC/7/,SAVE/2/
      DATA CHARS/'@','+','X','*','O',' ','.','-'/
      DATA IN77/'IN77',' '/,INNEW/'INNE','W   '/
C
      COMMON /ERRORS/ERRCOD,SUBCOD,DCBA
C
C
      CALL ERRSET(ERRCOD,10S,10S,DCBA,SUBCOD)
      GO TO 11
 10   CALL EXITR
 11   SUBCOD=0
      ERRCOD=0
C
C     GET INPUT PARAMETERS
C
 20   CONTINUE
      PRINT 90
 90   FORMAT(' SELECT INPUT FILE 1 OR 2 AND SQUARE SIZE.')
      INPUT I,SQR
      NAME(1)=IN77(1) ; NAME(2)=IN77(2)
      IF(I.EQ.2)NAME(1)=INNEW(1) ; NAME(2)=INNEW(2)
      PRINT 91,CHARS
 91   FORMAT(' SELECT MARK AND SPACE CHARACTERS FROM THESE:',/
     + 1X,8A1,/,' 12345678   ENTER digit,digit')
      INPUT I,J
      MARK = CHARS(I) ; SPACE = CHARS(J)
      PRINT 92
 92   FORMAT(' ENTER TERMINAL WIDTH(CHARS) AND IMAGE FILE ',
     + 'SIZE(MAX KEY)')
      INPUT W,KMAX
      W10 = SQR*W
 25   PRINT 93
 93   FORMAT(' ENTER START X,Y, AND SWATH WIDTH DELTA '
     + 'X, ALL IN INCHES WITH DECIMALS')
      INPUT RX,RY,RDELX
      REC = 1000.*(RX+.0005)
      IF(REC.GT.KMAX)REC = KMAX
      PIX = 1000.*(RY+.0005)
      DELX = 1000.*(RDELX-.0005)
      DELX = (DELX/SQR)*SQR+SQR-1
      LASTRC = REC + DELX
      IF(LASTRC.GT.KMAX)LASTRC = KMAX
      RPAGES = (LASTRC-REC)/(121.*SQR)
      PRINT 94,RPAGES
```

Figure E-2. Raster Display Utility Routine Fortran Listing (sheet 1 of 3)

```
 94   FORMAT(' THAT DELTA X WILL GIVE YOU ',F5.1,' PAGES.',/
     + 'ENTER THRESHOLD IF THIS SET UP IS OK.'/)
      INPUT THRESH
      IF(THRESH.LT.0)GO TO 3000
      IF(THRESH.EQ.0)GO TO 20
      PRINT 95,REC,LASTRC,PIX,W10
 95   FORMAT(' RECORDS ',I5,' THRU ',I5,', PIXEL',I5,' FOR ',
     +  I5,'PIXELS.',//,
     +  10X,'<--TOP--<<',//)
C
C     OPEN INPUT FILE
C
      CALL OPENF(1,NAME,IN,ACCT,PASS,KEYED,DIRECT,DC,0,4,0)
      IF(ERRCOD.NE.0)GO TO 8010
C
C     REPEAT FOR EACH LINE IN RANGE
C     --READ THE RECORD & UNPACK IT
C
      LCOUNT = 0
      DO 2000 KEY=REC,LASTRC
      CALL GETB(1,BUF,400,KEY,4)
      IF(ERRCOD.NE.0)GO TO 8020
      SIZE = IOR(IAND(4Z01FE,ISL(BUF(1),-7)),
     +            IAND(4ZFE00,ISL(BUF(1),9)))
      IF(SIZE.GT.390)GO TO 8030
      LEN = 0
      DO 110 I=1,SIZE
      CALL GETBYTE(BUF,I+3,BYTE)
      RL(I) = IAND(2Z7F,BYTE)
      COL(I) = IAND(2Z80,BYTE)
 110  LEN = LEN + RL(I)
C     (MIGHT CHECK FOR LEN TOO SHORT!)
      COL(SIZE+1)=0
      RL(SIZE+1)=PIX+W-LEN+1
C
C     --IF PIX NOT 0, SKIP OVER FIRST PIX PIXELS
C
 130  IGO = 1 ; COUNT = 0
      IF(PIX.EQ.0)GO TO 150
 131  COUNT = COUNT + RL(IGO)
      IF(COUNT.LE.PIX)IGO = IGO + 1 ; GO TO 131
      RL(IGO) = COUNT - PIX
C
C     --DECODE W10 PIXELS
C
 150  IF(LCOUNT.NE.0)GO TO 152
      DO 151 I=1,W
 151  LINE(I) = 0
 152  I=1
```

Figure E-2. Raster Display Utility Routine Fortran Listing
(sheet 2 of 3)



```
 160  COUNT = RL(IGO) ; DOT = COL(IGO)
      IGO = IGO + 1
      IF(DOT.EQ.0)I = I + COUNT ; COUNT=0 ; GO TO 169
 165  IF(COUNT.LE.0)GO TO 160
      I10 = (I-1)/SQR + 1
      LINE(I10) = LINE(I10) + 1
      I=I+1 ; COUNT = COUNT-1
 169  IF(I.LE.W10)GO TO 165
C
C     --OUTPUT LINE TO TERMINAL
C
 170  LCOUNT = LCOUNT+1
      IF(LCOUNT.LT.SQR)GO TO 2000
      LCOUNT = 0
      DO 171 I=1,W
      IF(LINE(I).LT.THRESH)LINE(I)=0
      IF(LINE(I).NE.0)LINE(I)=MARK
      IF(LINE(I).EQ.0)LINE(I)=SPACE
 171  CONTINUE
      PRINT 96,(LINE(I),I=1,W)
 96   FORMAT(1X,140A1)
C
C     END-DO
C
 2000 CONTINUE
C
C     CLOSE FILE AND EXIT
C
      CALL CLOSEF(1,SAVE)
      IF(ERRCOD.NE.0)GO TO 8040
      GO TO 25
 3000 CALL EXIT
C
C     ERROR MESSAGES
C
 8010 S=8010 ; GO TO 8900
 8020 S=8020 ; GO TO 8900
 8030 PRINT 8039,SIZE,BUF(1),KEY
 8039 FORMAT(' SIZE = ',I6,Z9,' ON RECORD KEY ',I6)
      GO TO 9000
 8040 S = 8040 ; GO TO 8900
 8900 PRINT 8999,ERRCOD,SUBCOD,S
 8999 FORMAT(' ERROR CODES ',2Z9,' FROM STATEMENT ',I6)
 9000 CALL EXIT
      END

!D ME(
.. 1 FILES DELETED, 4 GRANULES

!
```

Figure E-2. Raster Display Utility Routine Fortran Listing
(sheet 3 of 3)

APPENDIX F - FILE/RECORD LAYOUTS

This appendix documents the file/record layouts generated by the CONTAGRID software package.

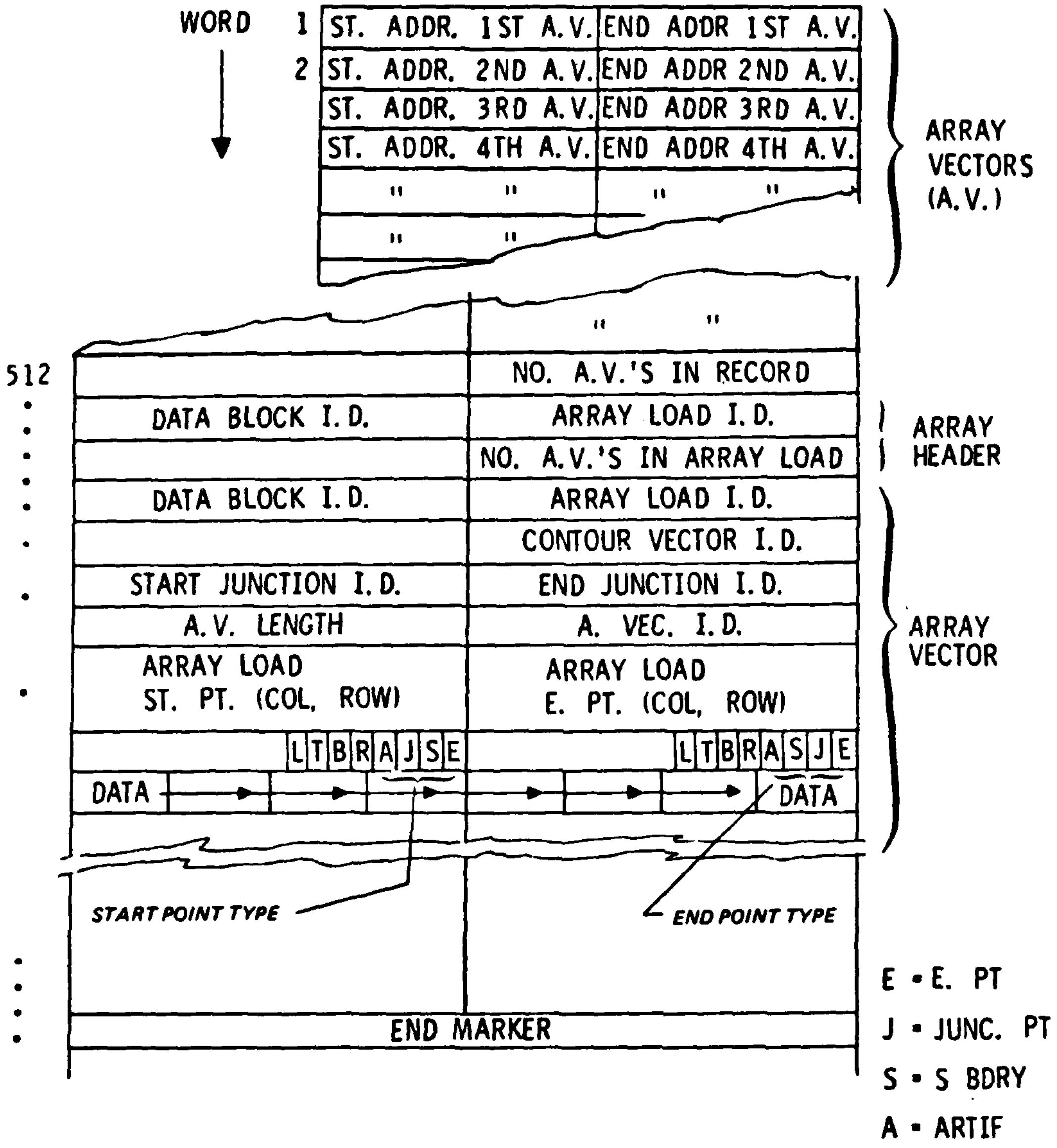


Figure F-1. Array Vector Record Format

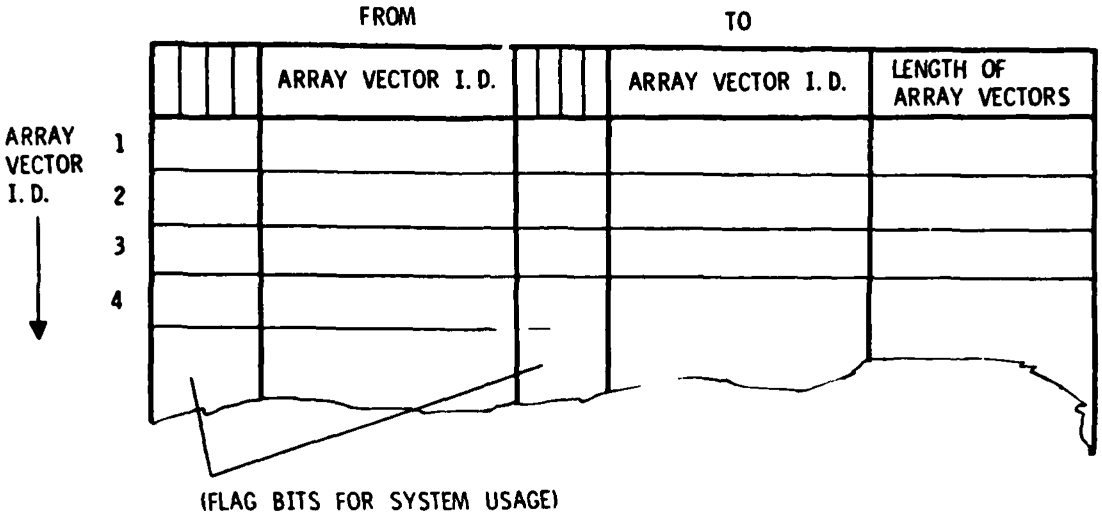


Figure F-2. Array Vector Link Table

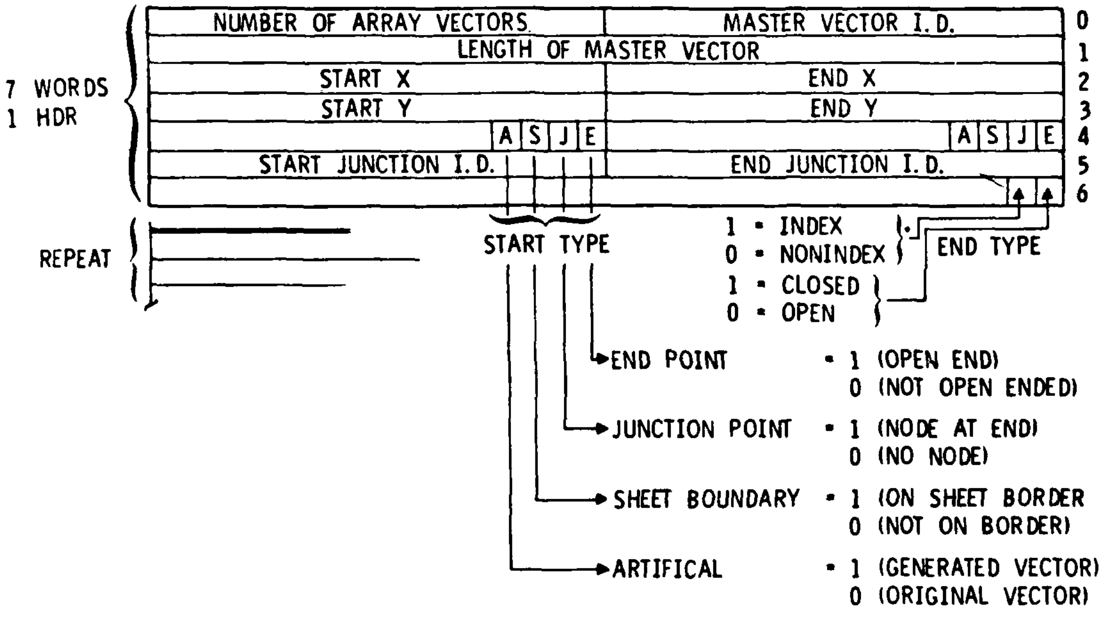


Figure F-3. Master Vector Description

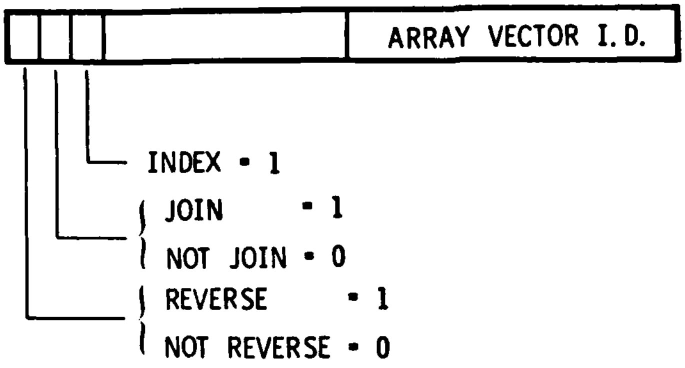


Figure F-4. Master Vector Link List

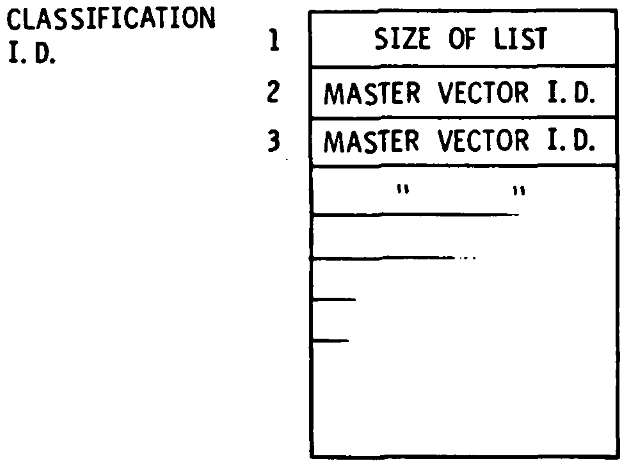


Figure F-5. Index, NON-INDEX, Join and Numeric Classification List Format

SIZE OF LIST	
SYMBOL I.D.	NUMBER OF MASTER VECTOR'S FOR SYMBOL
MASTER VECTOR I.D.	MASTER VECTOR I.D.
" "	" "
SYMBOL I.D.	NUMBER OF MASTER VECTOR'S FOR SYMBOL
MASTER VECTOR I.D.	MASTER VECTOR I.D.
" "	" "
" "	" "

Figure F-6. Depression, CUT, FILL Identification List

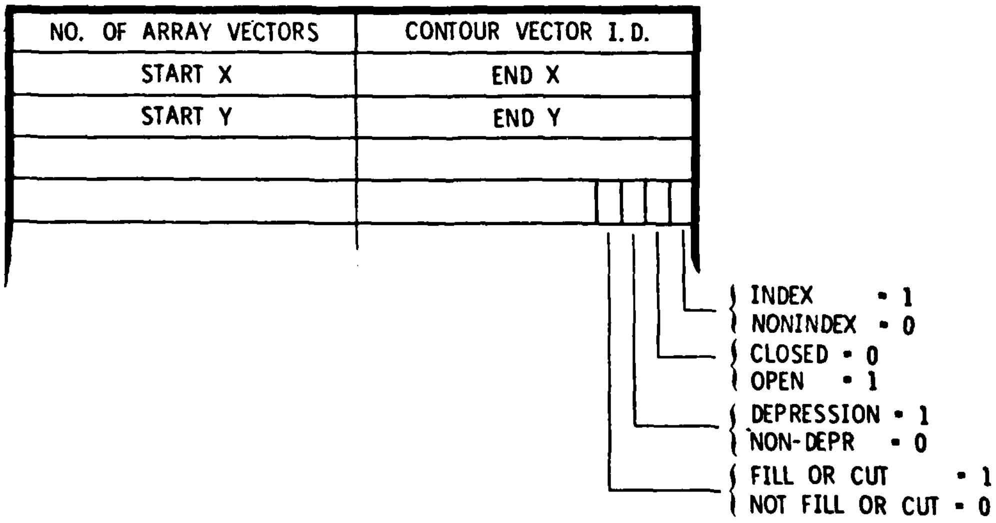


Figure F-7. Contour Vector Header

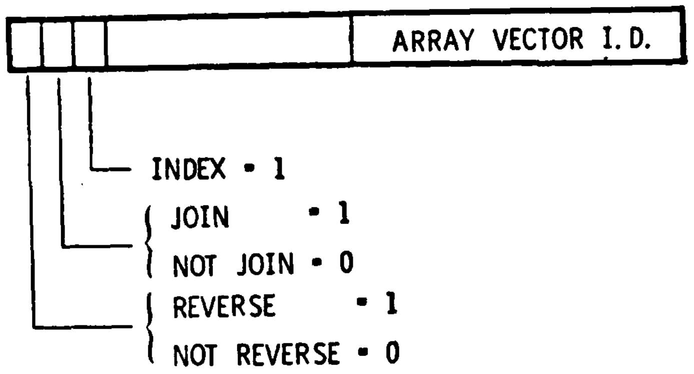


Figure F-8. Associated Array Vector List

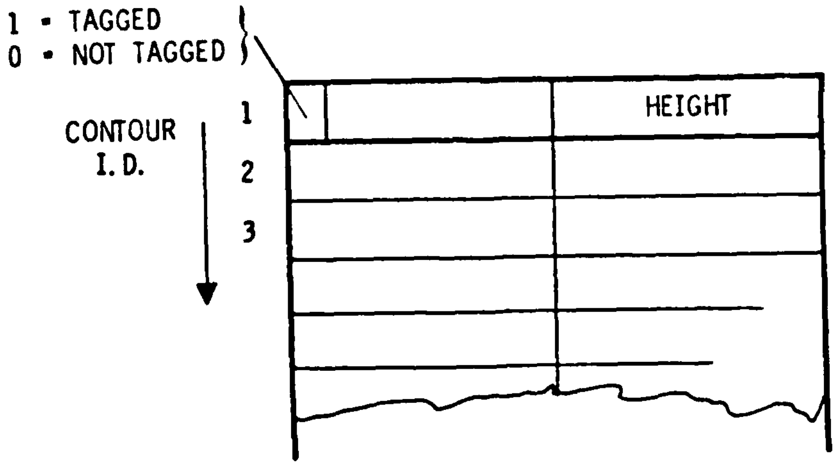
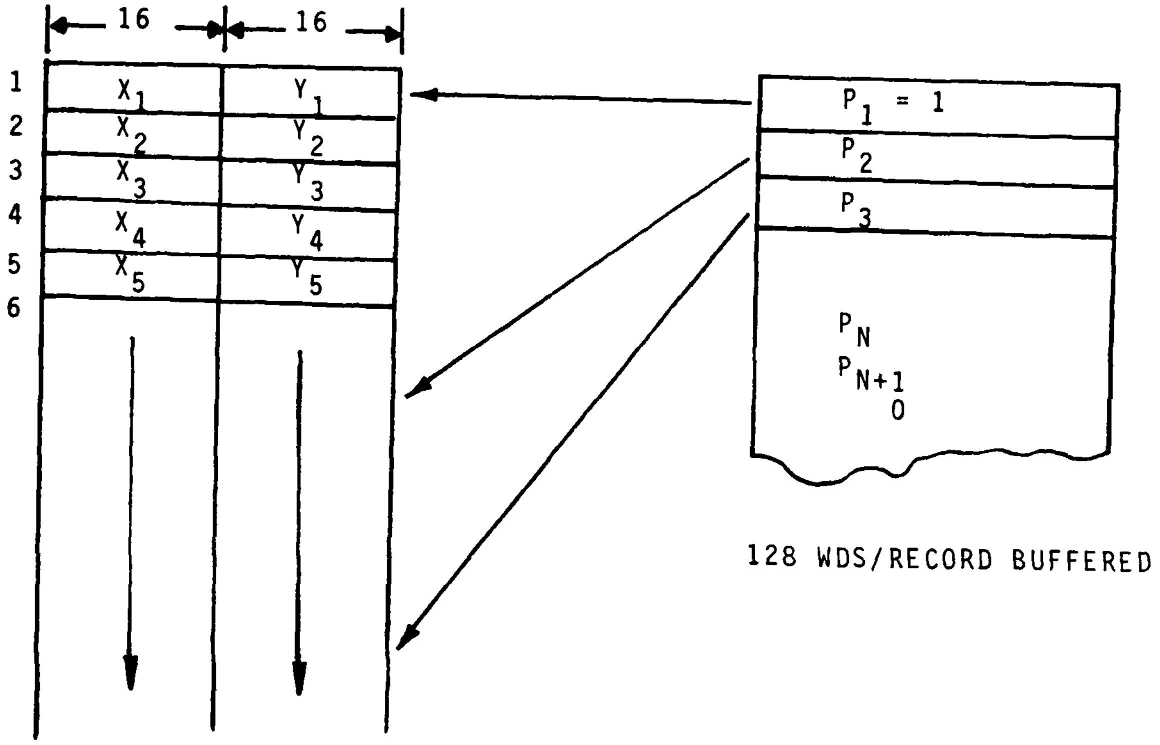


Figure F-9. Elevation List for Contour Vectors

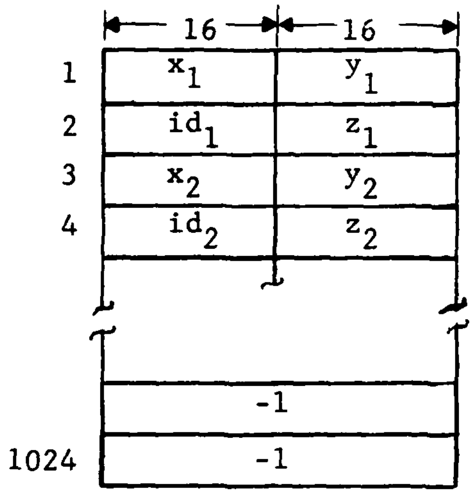


128 WDS/RECORD BUFFERED

1024 WDS/RECORD BUFFERED

R/S X-Y DATA IS ORDERED AS IT CAME OFF THE DGR TAPE. THE LINE POINTERS (ONE POINTER PER R/S LINE) POINT TO THE FIRST POINT IN THAT LINE. THE LAST (N^{th}) POINTER IS FOLLOWED BY THE $N+1^{th}$ POINTER TO DETERMINE THE LAST LINE'S LENGTH. A ZERO ENTRY FOLLOWS TO END THE LIST.

Figure F-10. R/S Data Points and R/S Line Pointers



	16	16
1	x_1	y_1
2	id_1	z_1
3	x_2	y_2
4	id_2	z_2
	-1	
1024	-1	

buffered

1024 words per record

Points are ordered as it comes off the DGR tape.
Last record is filled with -1 values.

Figure F-11. Spot Elevation & R/S Junction Mark Points (Hash Marks)

	16	16
1	x_1	y_1
2		z_1
3	x_2	y_2
4		z_2
⋮	⋮	
	-1	
1024	-1	

buffered

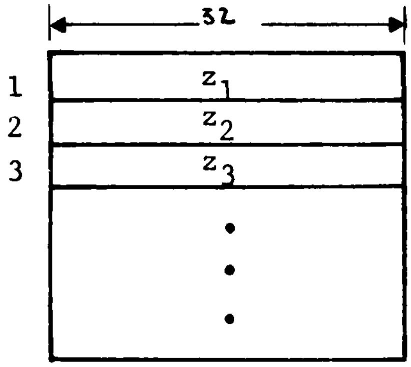
1024 words per record

x,y data is ordered as it comes off the DGR tape.
Last record is filled with -1 values.

First 4 words of first record only:

1	x_1	y_1	corner 1
2	x_2	y_2	corner 2
3	x_3	y_3	corner 3
4	x_4	y_4	corner 4

Figure F-12. Neat Line Points



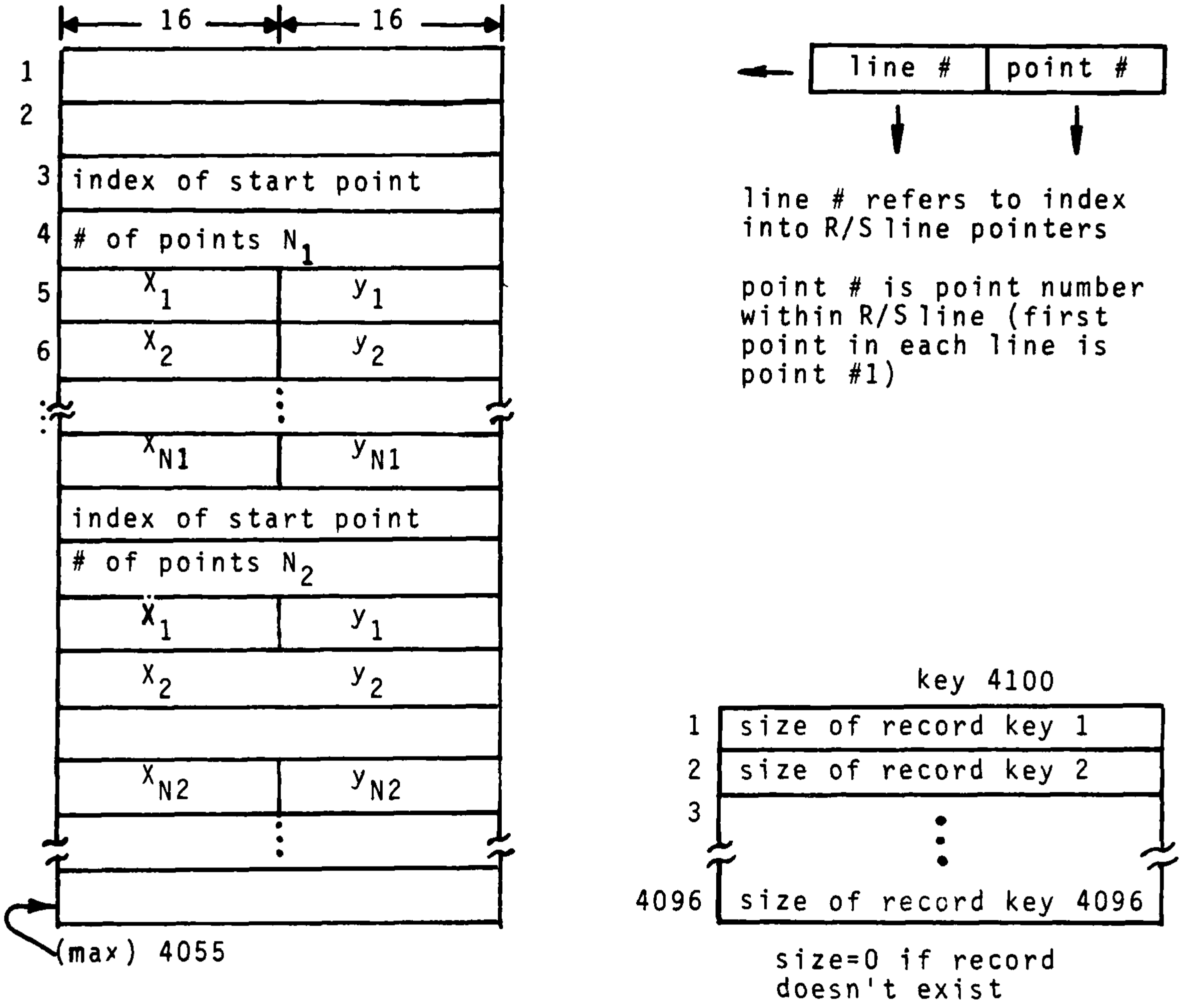
<u>buffered</u>

<u>Four (4) records</u>, one each for LEFT, TOP, BOTTOM, and RIGHT neat lines, in that sequence.

<u>Record sizes</u>: each record has one word for each point along the neat line. The top and bottom lines have a word (with a z-value) for each x-value; the left and right lines have a word for each y-value. The four corners each appear in two different records.

The z-values are in sequence along the neat lines. The top and bottom lines are ordered left to right, and the left and right lines, from top to bottom.

Figure F-13. Neat Lines (Splines)



line # refers to index into R/S line pointers

point # is point number within R/S line (first point in each line is point #1)

size=0 if record doesn't exist

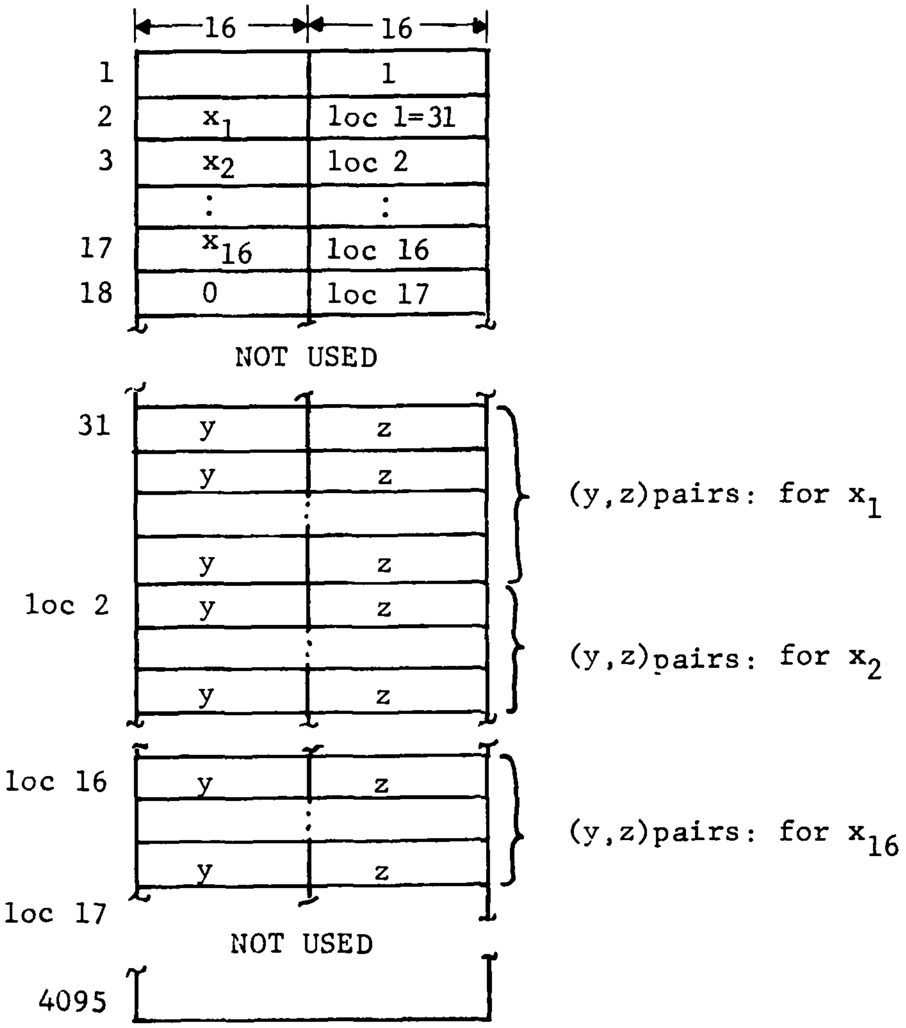
Random access: key = (scan block #-1)*16+(1:16)
(first record if first scan block is 1)

record size record: key=4100, size 4096, initialized to 0 and updated with word count size (index=key)

line segments are all within a scan block (16 GSL, first starts with first GSL right of left neat line.)

only complete segments in record; unused words will be at end of record.

Figure F-14. Partitioned R/S Segments



buffered

4095 words per record

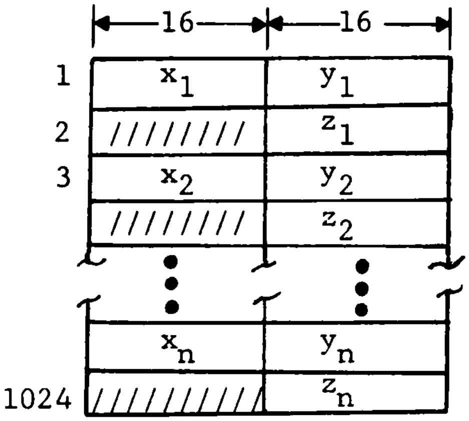
x_1 through x_{16} are x-values of GSL's in this scan block

(y,z) pairs and corresponding id's are sorted by ascending y-value within each x-value group

id's are scan line numbers (from DGR tape) of the points.

y is 0.12.4

Figure F-15. Partitioned R/S Junction (Hash) Mark Points



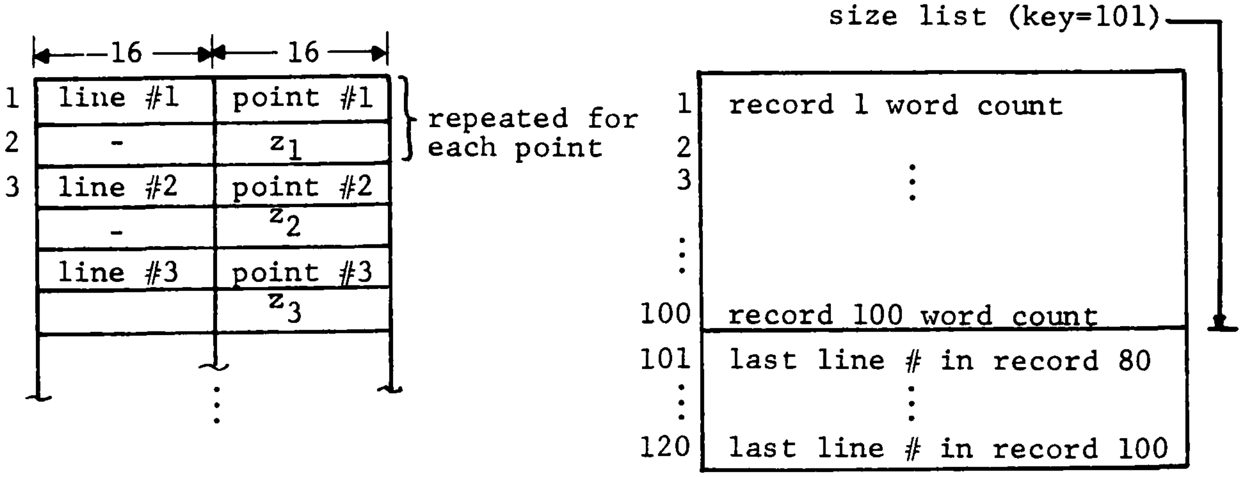
	16	16
1	x_1	y_1
2	////////	z_1
3	x_2	y_2
	////////	z_2
	⋮	⋮
	x_n	y_n
1024	//////////	z_n

buffered

Records 1024 words. Last record may be shorter.
sorted x-major ascending, y-minor descending.

Figure F-16. Sorted Spot Elevation Points

R/S INTERSECTIONS - also used for sorting junction (hash)
marks and R/S known points



Random access

keys 1-80: Up to 1024 words (512 pts) each.
Points are sorted within four-record groups (1-4,5-8, etc.)

keys 81-100: Up to 4096 words (2048 points) each.
All points in records 1-80 are sort-merged together in records 81-100. (No special fill values are expected.)

size list (wds 1-100 in key 101) in common INTSIZE
First 100 words contain word counts of corresponding records. Unused records must have word count = 0!

Words 101-120 each contain line # (in l.s.16 bits) of last valid point in each of records 81-100. Other bits should be 0. Word 101 corresponds to record 81; 102 to 82; etc.

Fiqure F-17. R/S Intersections

1	-
2	-
3	index of start point
4	# of points N_1
5	z_1
6	z_2
⋮	⋮
	z_{N_1}
	index of start point
	# of points N_2
	z_1
	z_2
	⋮
	z_{N_2}
	⋮
(max) 4095	

←

line #	point #

see PARTITIONED R/S
SEGMENTS

buffered

Record size is 4095.

Data parallels PARTITIONED R/S SEGMENTS file, i.e., corresponding words of corresponding records (with PARTITIONED R/S SEGMENTS records taken in key-value sequence) supply z-values for the (x,y) pairs.

z-values of -65536 are to be taken as invalid and the whole point is to be discarded.

There are no "spacer" records to parallel zero-length records in.

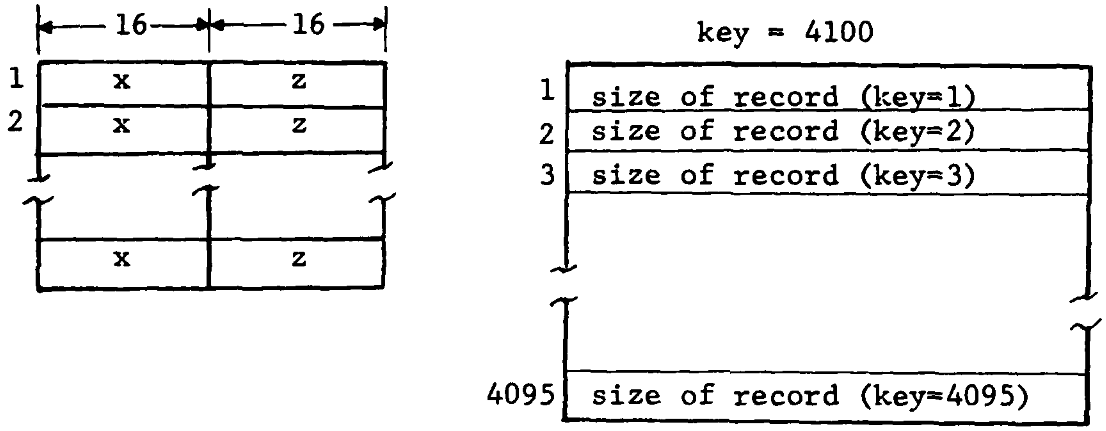
Fiqure F-18. Partitioned R/S Elevations

	16	16
1	x_1	y_1
2	//////	z_1
3	x_2	y_2
	///////	z_2
	x_n	y_n
4096	//////	z_n

buffered

Record 4096 words. Last record may be shorter.
Sorted x-major, ascending; y-minor, descending.
No duplicates in (x,y). Duplicates removed by MERGSRK

Figure F-19. Sorted R/S Known Points



random access

key = GSL-x-value - min-x (left-neat-line-x)
(i.e., GSL-x-value = min-x + key

keys range from 1 (for first backscanned GSL on left-not left neat line) through max-x (right neat line)

record (key=4100) contain sizes of other records.

Figure F-20. Backscanned Known Points