ETL-CR-74-1

ASSOCIATIVE ARRAY PROCESSING FOR TOPOGRAPHIC DATA REDUCTION

Interim Technical Report

March 1974

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

Contract DAAK02-73-C-0336

Goodyear Aerospace Corporation
1210 Massillon Road
Akron, Ohio 44315

Approved for Public Release;
Distribution Unlimited

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

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1 REPORT NUMBER: ETL-CR-74-1	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE *(and Subtitle)*: Associative Array Processing for Topographic Data Reduction	5 TYPE OF REPORT & PERIOD COVERED: Interim, June 1973 to Feb. 1974
6. PERFORMING ORG. REPORT NUMBER: GER-16073	
7 AUTHOR(s): R. G. Radosevic, Dr. J. Jun, C. Bruno, N. J. Adams	8. CONTRACT OR GRANT NUMBER(s): DAAK02-73-C-0336
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 Lab., Fort Belvoir, Virginia 22060	12. REPORT DATE: Feb. 1974
	13. NUMBER OF PAGES: 281
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)*

Associative array processing
Coordinate transformation
Interpolation
Automated cartography
Line thinning (Skeletonization)

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

Three general applications were evaluated to determine their suitability for associative array processing implemented with a STARAN* AAP. The three applications are: AS-11BX postprocessing, raster processing-automated cartography, and simultaneous multi-exposure analytical calibration (SMAC) program.

*T. M. Goodyear Aerospace Corporation, Akron, Ohio 44315.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

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

19. KEYWORDS (Continued)

Line symbol generation
Line break detection and correction
Character recognition
Analytical camera calibration
Matrix operation
Search operations
Instantaneous contours
Scene generation
SLAR image processing

20. ABSTRACT (Continued)

AS-11BX postprocessing involved coordinate transformation and interpolation of elevation data produced by the AS-11BX stereomapper. Associative array processing techniques were developed and programmed on the STARAN AAP to validate the timing estimates. These functions conform very well to associative array processing.

The raster processing functions evaluated for this study were: registration mark detection, line separation, character recognition, line thinning, vectorization, skew correction, line break detection, line smoothing, and line symbol generation. A STARAN S-1000 can realize a two order of magnitude improvement over an IBM 360/40 for this application.

The SMAC program effort was directed toward an evaluation of the effectiveness with which the STARAN AAP could perform various SMAC related search and matrix operations. STARAN's matrix arithmetic performance for large data fields will not offer an advantage to the SMAC program but the search operations will provide a great advantage.

SUMMARY

1. GENERAL

The objective of this study was to estimate the adaptability of a STARAN associative array processor (AAP) manufactured by Goodyear Aerospace Corporation (GAC) to the data processing problems of the U.S. Army Engineer Topographic Laboratories (ETL).

The study was implemented in two phases. Phase I incorporated a functional survey and explicit planning involving both GAC and ETL. The following six data processing tasks were examined:

1. AS-11BX postprocessing
2. Raster processing - automated cartography
3. Simultaneous multi-exposure analytical calibration (SMAC) program
4. SLAR image processing
5. Scene generation
6. Instantaneous contours

Phase II consisted of the study, validation, and reporting of the areas of interest selected at the conclusion of Phase I. Items 1, 2, and 3 were selected for the more detailed Phase II study analysis based on ETL priority and interest. The other three items are discussed in Appendix D.

2. AS-11BX POSTPROCESSING

The AS-11BX is a stereo mapper that uses a pair of aerial stereo photographs to compute elevations for the model area that is described by the overlapping portion of the two photographs. Coordinate transformation and interpolation of the resulting elevation data are the primary postprocessing tasks.

The elevation values are transformed from the model coordinate system to a local coordinate system to provide a common reference for the various model areas. The local coordinate system is situated at a point on the earth's surface. The transformed points then are interpolated to produce a set of regularly spaced elevations. The interpolation is performed with a two-dimensional quadratic curve fit. The processed elevations will ultimately be used in a digital topographic data base.

During this study, associative array processing techniques were developed and timing estimates derived for the two postprocessing functions. These techniques were then programmed on the STARAN AAP to validate the timing estimates. A summary of the results is shown in Table I.

TABLE I - SUMMARY OF AS-11BX POSTPROCESSING RESULTS

Function	STARAN S-1000 timing	
	Estimated (sec)	Validated (sec)
Coordinate transformation	4.18	3.66
Interpolation	18.94	23.78
Total	23.12	27.44
No. AS-11BX's serviced	74	65

The results show that the AS-11BX postprocessing functions conform very well to associative array processing. In fact, the number of AS-11BX's that could be serviced are probably greater than the number that would exist in one installation. Therefore a logical conclusion is that STARAN should assume additional postprocessing tasks. Some tasks that should be evaluated for STARAN application are:

1. Generation of the model data (elevations) from parallax data
2. Application of model deformation characteristic corrections to the model data.
3. Transformation of the model data from the local coordinate system to a geographic coordinate system.
4. Consideration of other interpolation techniques.

3. RASTER PROCESSING - AUTOMATED CARTOGRAPHY

Raster processing is that part of automated cartography that digitally processes properly positioned line and character data to produce the appropriate map symbology. A scanner/plotter is used to digitize the positional data and plot the resulting symbology.

The objective of this study effort was to evaluate the suitability of various time-consuming raster processing functions for associative array processing. This evaluation was accomplished by developing or restructuring appropriate algorithms, estimating the STARAN AAP execution times, validating selected functions in the STARAN evaluation and training facility, and comparing the results to those presently experienced with an IBM 360/40.

The raster processing functions evaluated for this study were: registration mark detection, line separation, character recognition, line thinning, vectorization, skew correction, line break detection, line smoothing, and line symbol generation.

Except for character data, the primary difference between the approach presently implemented in ETL's IBM 360/40 program and that proposed for the STARAN AAP is in the method utilized to represent the map image. The IBM 360/40 represents lines as a series of vectors and requires the creation of vector lists. The AAP however, stores the map image (one area at a time) directly (intact) in the associative arrays and processes the data in this format using simple logic operations. This technique completely eliminates the time consuming raster-to-vector-to-raster conversions required by the sequential processor. Also, performance with the AAP approach is independent of map density.

Study results show that a STARAN S-1000 can realize a two order of magnitude improvement over the IBM 360/40 for this application. More specifically, the raster processing functions evaluated for STARAN application required approximately 9 1/2 hr of 360/40 processing time and were estimated to require about 52 sec of STARAN time. The total 360/40 processing time for all raster processing functions is about 13 1/2 hrs.

The impressive study results show an extremely good fit between the raster processing requirements and the associative array features of a STARAN. Additional effort should be directed toward verification of basic AAP concepts with real map data, evaluation of additional raster processing functions such as

color merging, and the development of a complete operational system concept with appropriate man-machine interfaces to edit data and control the problem.

4. SIMULTANEOUS MULTI-EXPOSURE ANALYTICAL CALIBRATION (SMAC) PROGRAM

The Simultaneous Multi-Exposure Analytical Calibration (SMAC) program is an operational software program implemented on a Univac 1108 sequential computer. The program involves an analytical camera calibration technique that uses known stellar data.

The SMAC program involves: the association of photographed stars with cataloged stars; the matrix solution of simultaneous equations that involve the camera distortions, star positions on the camera exposure, and cataloged star positions; conversion routines to update cataloged star positions; automatic measurement editing; and statistical testing of camera calibration results.

The main emphasis of this study effort was directed toward an evaluation of the effectiveness with which the STARAN associative array processor could perform various SMAC related search and matrix operations. The search operations, an exact match search of the BOSS catalog and a between-limits search of the Smithsonion Astronomical Observatory (SAO) catalog, were evaluated for a STARAN Parallel Head Disc/Drum (PHD) system.

For various size matrices, matrix addition, subtraction, multiplication, and inversion were evaluated using a standard STARAN and a STARAN with two potential options. One potential option involves STARAN's high-speed parallel input/output interface channel with an appropriate random access memory. The other option is a hypothetical hardware floating-point arithmetic unit.

Univac 1108 execution times were provided for the two main SMAC programs. This however did not provide timing visibility at the task level for a direct comparison with the STARAN timing estimate. Therefore, STARAN timing estimates also were derived for many of SMAC's arithmetic operations in an attempt to include, with the

search and matrix operations, enough functions of one major program to allow a timing comparison between the STARAN and Univac 1108.

All STARAN timing estimates for SMAC functions were derived for a STARAN S-1000 (four arrays) using double-precision floating-point arithmetic. The matrix operations, particularly matrix inversion, were very well suited to array processing. However, because of the large data fields involved, STARAN's software implemented arithmetic precludes any STARAN advantage over the Univac 1108 and in fact, for this condition they are comparable. However, with the hypothetical options mentioned above, the four-array STARAN would show at least a one order of magnitude advantage over the Univac 1108 for matrix inversions.

The result obtained by totaling the STARAN timing estimates for the appropriate search, matrix, and miscellaneous arithmetic operations was initially assumed to represent a significant part of SMAC's preliminary program execution time. However, a comparison of the above result with the Univac 1108 execution time provided by ETL reveals an apparent STARAN advantage of about 40:1. Since STARAN's matrix and miscellaneous arithmetic operations cannot be supporting this advantage, the STARAN search operations and/or other SMAC functions not included in the STARAN estimates must account for the difference.

There is little doubt that the STARAN search operations in the SMAC programs are much faster than those for the present Univac 1108. ETL has implied that these operations represent a significant part of the overall execution time and therefore STARAN probably will have a significant impact upon the SMAC program. However, a definite conclusion cannot be drawn until the search times are obtained for the present Univac 1108 implementation.

FOREWORD

This interim technical report records efforts and achievements under the "Associative Processing for Topographic Data Reduction" program conducted by Goodyear Aerospace Corporation (GAC), Akron, Ohio. This report is submitted by GAC as GER-16073.

The program was conducted for the Computer Sciences Laboratory (CSL) of the U.S. Army Engineer Topographic Laboratories (ETL) under Contract DAAK-02-73-C-0336. Mr. W. E. Sanburn served as the Contracting Officer's Representative. Contributors to the effort were Messers. R. G. Radosevic (project engineer), N. J. Adams, C. J. Bruno, and Dr. J. Jun.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Continued)

LIST OF ILLUSTRATIONS (Continued)

LIST OF TABLES

LIST OF TABLES

1. INTRODUCTION

1.1 GENERAL

Goodyear Aerospace Corporation (GAC) conducted a seven-month program for the U.S. Army Engineer Topographic Laboratories (ETL) to analyze the applicability of an associative array processor to ETL data processing. The contract was executed in two phases.

1.2 PHASE I - FUNCTIONAL SURVEY AND DETAILED PLANNING

Phase I of the study called for a brief evaluation of many of ETL's data processing problems to determine their suitability to associative array processing. Phase I began by:

1. Surveying various present and proposed ETL data processing (DP) tasks.
2. Determining significant constraints imposed upon each task as a result of existing and proposed system hardware and overall ETL objectives.
3. Determining ETL's priorities and preferences for the GAC conducted study and validation tasks.

At this point GAC disseminated and utilized the information derived above to generate a specific task list for the study-validation effort (Phase II). Each of ETL's DP tasks was evaluated to determine the degree to which their respective algorithm solutions could be effectively executed with an AAP. Once the tasks were ordered as a function of their AAP adaptability, ETL's priorities and preferences were factored into the process and a preliminary study-validation task list was generated for Phase II.

The six tasks investigated are listed below in the order of ETL's priority:

1. AS-11BX postprocessing
2. Raster processing - automated cartography
3. Simultaneous multi-exposure analytical calibration (SMAC) program.
4. SLAR image processing
5. Scene generation
6. Instantaneous contours

The first three tasks were selected by ETL for detailed study. The first two tasks were also selected for validation.

1.3 PHASE II - STUDY, VALIDATION, AND DOCUMENTATION

Phase II of the proposed program consisted of:

1. Executing the selected study tasks,
2. Performing the selected validation tasks in the STARAN
3. Documenting the overall program effort, results, and recommendations

Execution of the selected study tasks involved system analysis, algorithm structuring, and system timing estimates. It was necessary to analyze and structure the algorithms in detail to permit the most efficient parallel execution on the STARAN AAP. System performance estimates included the timing required for computation, data transfer, and data manipulation.

The STARAN Evaluation and Training Facility was utilized to implement and test the selected validation tasks. The previously structured algorithms were coded, assembled, debugged, and tested. The objective of this validation effort was to verify the STARAN AAP computation and data manipulation timing estimates derived above and to demonstrate the suitability of AAP architecture for these problems.

The documentation effort consisted of status reports, and this report.

1.4 REPORT ORGANIZATION

The analysis and results for each task of Phase II are recorded in the four sections of the main body:

1. AS-11BX postprocessing
2. Raster processing - automated cartography
3. Simultaneous multi-exposure analytical calibration (SMAC) program
4. Conclusions and Recommendations

Each of the first three sections are broken down into an investigation and discussion. The investigation contains an account of the

task being studied, how the analysis (and validation in two cases) were performed and the results. The discussion provides a more detailed account of the study (analysis) and the validation, in terms of associative array processing techniques. A detailed description of the programs that were written to validate the timing estimates performed during the study are given also. The method of recording the validated times are presented.

The conclusions contain all inferences that could be made based upon the result of the investigation with respect to the study, validation, and results. The recommendations are the logical outcome of the conclusions and other information that was obtained either directly or indirectly during the study.

Appendix A contains the test data input and results of the two validated AS-11BX postprocessing functions. Program printouts are also provided.

Appendix B contains computer printouts associated with the raster processing validation effort. Included are binary printouts of the arrays showing the test pattern, line thinned results, and single and double line symbol generation results. Also included are validation program listings.

Appendix C includes algorithms used in the SMAC study effort.

Appendix D provides a brief description of the three study tasks considered during Phase I, but not included in the detailed study. These three functions are SLAR image processing, scene generation, and instantaneous contours.

Appendix E contains STARAN performance data for most Associative Processor Programming LanguagE (APPLE) instructions.

Appendix F contains information on special STARAN hardware configurations. The basic STARAN architecture is referenced and a brief description is provided for a hypothetical hardware arithmetic option

Appendix G describes a "tree summing" technique that applies STARAN features to produce the summation of many values in a small number of additions.

2. AS-11BX POSTPROCESSING

2.1 INVESTIGATION

2.1.1 Background

The AS-11BX is an advanced stereo mapper currently being developed by the Bendix Corporation under contract to Rome Air Development Center. The mapper produces parallax values from a pair of stereo exposures by extracting and correlating density data in the overlapped area (model area) of the two pictures. After determining the parallax for all desired points in the model area, the AS-11BX then will use these data to compute elevation values that will represent the topography of the model area.

GAC understands that the nominal operating time required by the AS-11BX to process one model area of about 150,000 points will be approximately 30 min; including operator intervention.

The elevations produced from a pair of stereo exposures are referenced in space to one of the two camera positions that produced the stereo pair. The referenced camera position defines the location of the model coordinate system.

The elevation data resulted from correlations performed along epipolar lines. The correlated points are irregularly spaced along the epipolar lines, and the epipolar lines are not normally aligned with any coordinate axis. Therefore, the elevation data produced by the AS-11BX are irregularly spaced in each model area.

For this study, AS-11BX postprocessing is defined as processing of AS-11BX output data by the STARAN associative array processor. AS-11BX postprocessing is primarily concerned with two tasks: coordinate transformation and interpolation.

The coordinate transformation task is required for transformation of topographical data from many independent model coordinate systems (where the elevations were generated) into a common (local) coordinate system. The local coordinate system is located at a point on the earth's surface.

The interpolation task uses the transformed topographical data points (which are irregularly spaced) to generate elevation data at regularly spaced locations in the local coordinate system.

2.1.2 Objectives

The primary objective of this investigation was to determine the number of AS-11BX devices that can be supported by a STARAN S-1000 associative array processor (AAP) in a real-time environment. Therefore, the main task was to estimate and validate the time required by the AAP to perform the two post-processing tasks of coordinate transformation and interpolation.

In addition, the system sizing was estimated and an approach to a total system configuration that could support the postprocessing functions was formulated. Consideration also was given to special AAP configuration and options that could enhance the processor's performance.

2.1.3 Study

ETL-CSL provided information on the two postprocessing functions in terms of algorithms, required field sizes, number of sample points per model area, AS-11BX processing time per model area, and general AS-11BX operational characteristics.

The form of the basic coordinate transformation algorithm is shown below:

$$\begin{pmatrix} X_L \\ Y_L \\ Z_L \end{pmatrix} = S \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} X_M - X_0 \\ Y_M - Y_0 \\ Z_M - Z_0 \end{pmatrix} .$$

This algorithm and its implementation is fully explained in the discussion on coordinate transformation (Section 2.2.1).

The form of the interpolation algorithm is shown below:

$$C = \frac{\begin{vmatrix} \Sigma Z_i & \Sigma X_i & \Sigma X_i^2 \\ \Sigma X_i Z_i & \Sigma X_i^2 & \Sigma X_i^3 \\ \Sigma X_i^2 Z_i & \Sigma X_i^3 & \Sigma X_i^4 \end{vmatrix}}{\begin{vmatrix} 5 & \Sigma X_i & \Sigma X_i^2 \\ \Sigma X_i & \Sigma X_i^2 & \Sigma X_i^3 \\ \Sigma X_i^2 & \Sigma X_i^3 & \Sigma X_i^4 \end{vmatrix}}$$

A detailed description of this algorithm and its implementation is given in the discussion of interpolation (Section 2.2.2).

Field lengths within the associative arrays were organized to accommodate the data, preserve accuracy, and maximize the parallel capabilities of the arrays. The AS-11BX uses 18-bit data fields for its output parameters. The rotation matrix in the coordinate transformation algorithm requires a 24-bit field length to retain 18-bit accuracy for the elevation data. All computational results for both the coordinate transformation and interpolation functions were optionally assigned 20-bit field lengths.

The remaining associative array bits (62 to 256 for coordinate transformation and 98 to 256 for interpolation, respectively) were used as a "scratch area" which provided intermediate storage for results, such as products, that exceed the allocated field lengths. The large resultant fields in the scratch area were then truncated to the 20-bit field length allocated for the parameter. A detailed diagram and description of the associative array organization is provided in Sections 2.2.2.3 and 2.2.4.3.

Timing estimates for the AS-11BX postprocessing functions considered array loading and unloading, array computations, and miscellaneous overhead tasks such as data scaling and housekeeping.

The array loading and unloading time is directly related to the amount of data output by the AS-11BX and the number of array transfers required during the computations. This time was estimated for both associative array interfaces; the 32-bit common register interface, and the 256-bit parallel input/output interface. Possible system configuration for both approaches is shown and discussed (in Section 2.2.6).

The computation time estimates for the AS-11BX postprocessing functions were based upon STARAN performance for the APPLE assembly language, a four-array STARAN (STARAN S-1000), and the required field lengths. Performance data for the APPLE assembly language mnemonics are listed in Appendix E.

Timing estimates for the miscellaneous overhead tasks such as data scaling and housekeeping were accounted for by increasing the array loading/unloading and computation times by 25 percent.

As a reference, computation times were also estimated for a hypothetical hardware arithmetic option that could be developed to greatly enhance STARAN performance. This potential option is briefly discussed in Section 2.2.6 and Appendix F.

2.1.4 Validation

The coordinate transformation and interpolation programs were written using the APPLE assembly language and tested in the STARAN Engineering and Training Facility (SETF).

The primary validation objective was to verify the STARAN execution times for the basic postprocessing software routines. No attempt was made to develop an operational program.

During the course of the AS-11BX postprocessing validation effort, the STARAN system in the SETF was being modified and upgraded. As a result, two of the four arrays and the internal performance monitor were unavailable during this period of time.

The detailed timing estimates had been made assuming a standard STARAN S-1000 (four-array system). Therefore, the times obtained during validation were adjusted to reflect the utilization of two more arrays. For this adjustment, the array loading and unloading times were doubled because twice the amount of data was processed. The array computation times were the same because only parallel computations were involved.

Normally, the internal performance monitor is used to measure program execution times. Because the monitor was not available during this validation effort, a simple but effective alternate method was utilized. Instead, three program sections (array loading, array unloading, and array computations) were executed separately many thousands of times. An indicator lamp was automatically set upon completion of each process. The program took several minutes to execute and a stopwatch was used to

measure the elapsed time. The measured time then was divided by the number of program executions to obtain the execution time for the tested routine.

This timing approach has been used many times and even though it is more time consuming, the results are just as accurate as those obtained with the internal performance monitor. For reference purposes, a description is provided in (Section 2.2) of how the internal performance monitor would normally be utilized to measure STARAN program execution times.

Controlled test data were used to verify that the coordinate translation and interpolation programs were producing correct results. The test data were synthetically generated to simplify verification of the results. The test data are described more fully in Section 2.2 and in Appendix A.

2.1.5 Results

The estimated and validated timing for the two main postprocessing functions of coordinate transformation and interpolation are discussed below. Included are timing estimates for support functions and for the incorporation of two hardware options. These results are derived from the estimated and validated times discussed in Section 2.2.

Table 2-I is a timing summary for the two main AS-11BX postprocessing functions. These results indicate that a standard STARAN can simultaneously perform the postprocessing tasks for many AS-11BX devices.

Table 2-I shows estimated times to perform the coordinate transformation and interpolation functions utilizing one-, two-, and four-array STARAN systems. Estimates for the time required by a Univac 1108 are included to provide a rough comparison between the STARAN associative array processor and a high-performance sequential computer. The STARAN S-1000 shows about a 5-to-1 improvement over the Univac 1108. All estimated times in Table 2-I include a 25 percent overhead factor to account for miscellaneous housekeeping tasks and scaling of the data during arithmetic computations.

TABLE 2-I - TIMING SUMMARY FOR MAIN AS-11BX POSTPROCESSING FUNCTIONS

Main functions	STARAN timing (sec)				Univac 1108 (sec)
	S-250	S-500	S-1000		
	Estimated	Estimated	Estimated	Validated	Estimated
Coordinate Transformation	9.07	5.8	4.8	3.66	17.2
Interpolation	59.7	38	18.94	23.78	114
Total	68.77	43.8	23.12	27.44	131.2
No. BX's serviced	25	39	74	65	13

The validated times for the two functions were timed on a STARAN S-500 (two arrays) and projected for a STARAN S-1000 system. The estimated number of AS-11BX's that could be simultaneously serviced was based on an average processing time of 30 min per model area for the stero mapper.

The estimated and validated times were within 21 percent of each other. The validation time for the interpolation program is somewhat higher than the estimated value because the scaling requirements for the fixed point arithmetic computations were greater than anticipated.

Table 2-11 is an extension of Table 2-1 and includes timing estimates for various support functions. The secondary functions of sorting and corner turning are routines required to support the interpolation process. They could be handled in a sequential (host) computer, but the STARAN estimates are included for reference purposes. Estimated input/output times were derived for transferring the model data (150,000 points) from the AS-11BX to STARAN at the rate of 1.1 μsec per word, and transferring the interpolated results (100,000 elevations) from STARAN to magnetic tape at a rate of 288,000 bits per second. The input time is approximately 0.5 sec and the output time is approximately 7.42 sec. Outputting the X and Y values as well as the Z values (elevations) increases the output time to 22.2 sec. The

TABLE 2-II - TIMING SUMMARY FOR AS-11BX POSTPROCESSING ON STARAN S-1000 (WITH ADDED FEATURES)

Main functions	STARAN S-1000 timing (sec)				
	Estimated	Validated	With Parallel I/O	With Hardware Arithmetic	I/O & Hardware Arithmetic
Coordinate Transformation	4.18	3.66	1.17	2.83	0.34
Interpolation	18.94	23.78	15.26	16.98	8.6
Total	23.12	27.44	16.43	19.81	8.94
No. BX's serviced	74	65	109	90	201
Support functions					
Sort and corner turning	8.44	8.44	8.44	8.44	8.44
Input/Output	7.92	7.92	7.92	7.92	7.92
Cumulative Total*	31.56	35.88	24.87	28.25	17.38
No. BX's serviced	54	47	68	60	98

*This total does not include I/O time (see discussion).

input/output times are not included in the accumulated totals shown in Table 2-II because one block of data or results can be input or output while another block is being processed.

Included in Table 2-II are timing estimates for a standard STARAN utilizing parallel input/output (PIO) channels and special hardware floating point arithmetic units.

The PIO allows a 256-bit data transfer, which significantly reduces the I/O times and increases the number of AS-11BX machines that can be simultaneously serviced. A more detailed description of the PIO is given under Section 2.2.6 and in Appendix F.

The hardware arithmetic units have not been developed, but will offer a viable option for the future. Except for fixed point addition and subtraction, the hardware approach would provide a significant performance advantage over the present software implemented for fixed and floating point arithmetic operations. This option is discussed also under Section 2.2.6 and in Appendix F.

The results in Tables 2-I and 2-II indicate that the standard STARAN S-1000 performs the AS-11BX postprocessing functions very effectively. However, should increased STARAN performance be required due to increased complexity of the postprocessing task, the additional associative arrays, the PIO option, and/or the potential hardware arithmetic units could be incorporated as required.

2.2 DISCUSSION

2.2.1 General

The postprocessing functions, timing estimate derivations, and overall system considerations are discussed in detail below. The algorithms for both the coordinate transformation and interpolation functions are given together with the array loading, array computations, and array unloading techniques. Other miscellaneous postprocessing tasks are discussed and details about considerations that relate to the total system concept also are given.

2.2.2 Coordinate Transformation Study

2.2.2.1 General - Figure 2-1 is a diagram of a point produced by the AS-11BX and referenced to both the model coordinate system $(X,Y,Z)_M$ and the local coordinate system $(X,Y,Z)_L$. The object of the coordinate transformation process is to transform all points from the model coordinate system to the local coordinate system.

2.2.2.2 Algorithm - The mathematical relationship between the two coordinate systems is given below:

$$\begin{pmatrix} X_L \\ Y_L \\ Z_L \end{pmatrix} = SA \begin{pmatrix} X_M - X_0 \\ Y_M - Y_0 \\ Z_M - Z_0 \end{pmatrix} , \qquad (1)$$

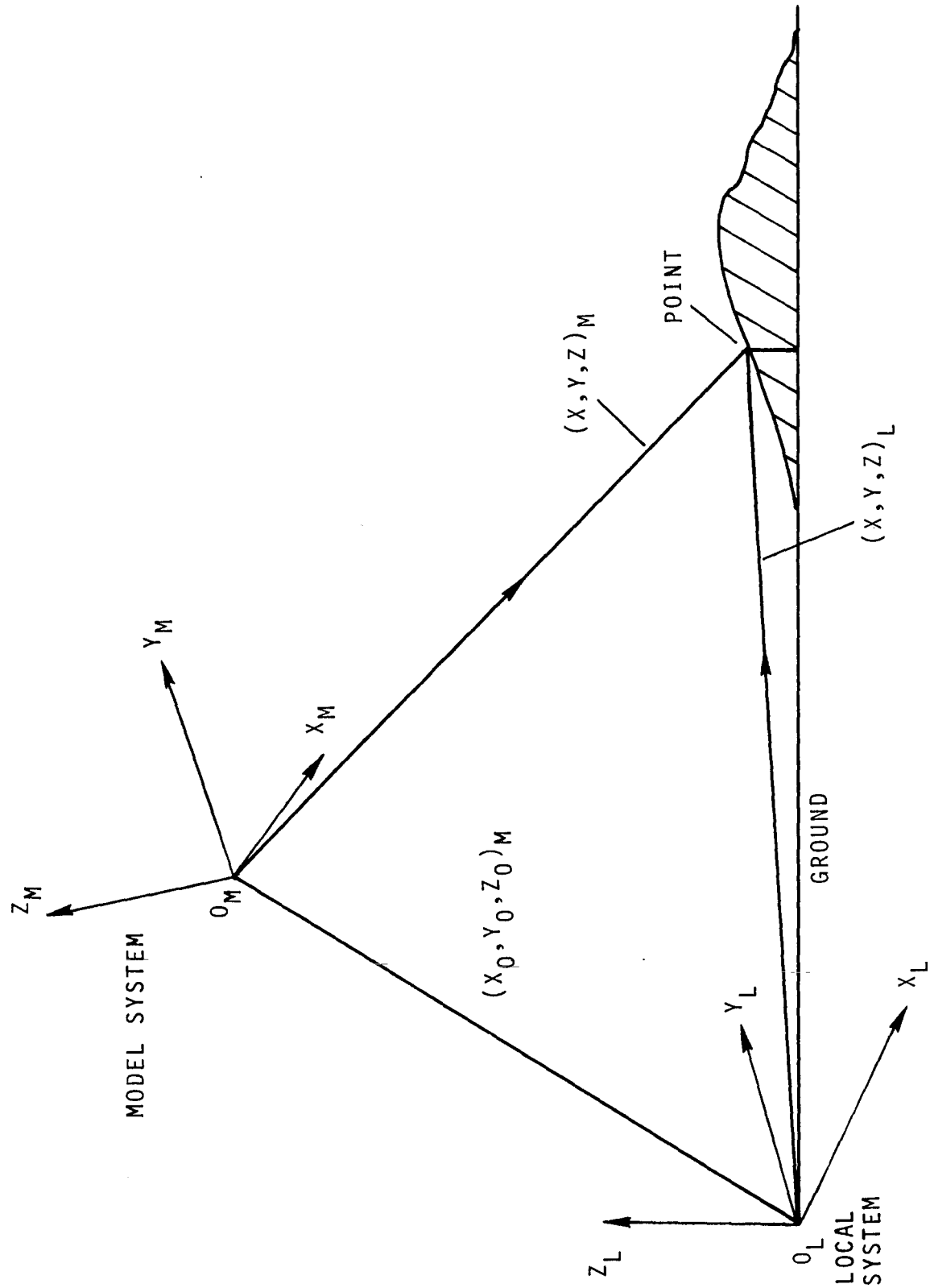


Figure 2-1 - Relationship between Coordinate Systems

where

(X_L, Y_L, Z_L) = point whose (X,Y,Z) values are taken with respect to the local coordinate system,

(X_M, Y_M, Z_M) = point whose (X,Y,Z) values are taken with respect to the model coordinate system,

A = 3x3 rotation matrix whose elements are direction cosines defining the relationship between the axes of the model and local coordinate systems,

S = scale factor, and

(X_0, Y_0, Z_0) = distance in the model coordinate system of the origin O_L of the local coordinate system from the origin O_M of the model coordinate system.

Figure 2-2 is a flow diagram of the coordinate transformation program.

Constant data such as the rotation matrix elements, and the scale factor, etc. are loaded into the STARAN arrays. The variable points (elevationsin the model coordinate system) then are introduced into the arrays and the transformation algorithm is executed to generate the results (elevations in the local coordinate system). At this point the transformed points are unloaded into bulk core memory, and a new block of input values is obtained. This process continues until all the points have been translated.

2.2.2.3 Array Organization - Figure 2-3 shows the array organization for the various inputs.

The variable input data X_i, Y_i, Z_i, i = 1, 2, ... correspond to the model coordinates of each point in the model space area. The X,Y,Z values of each point are stored into Fields F1, F2, and F3 of the first word of each group of three words. Each value then is replicated in the other two words.

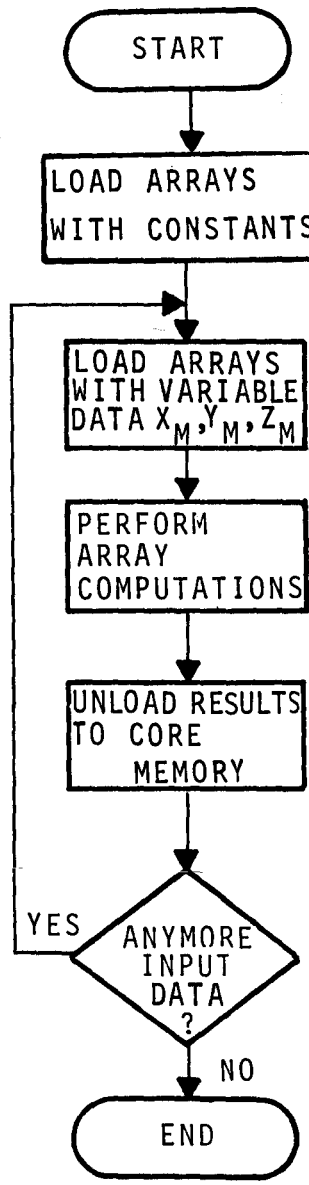


Figure 2-2 - Coordinate Transformation Flow Diagram

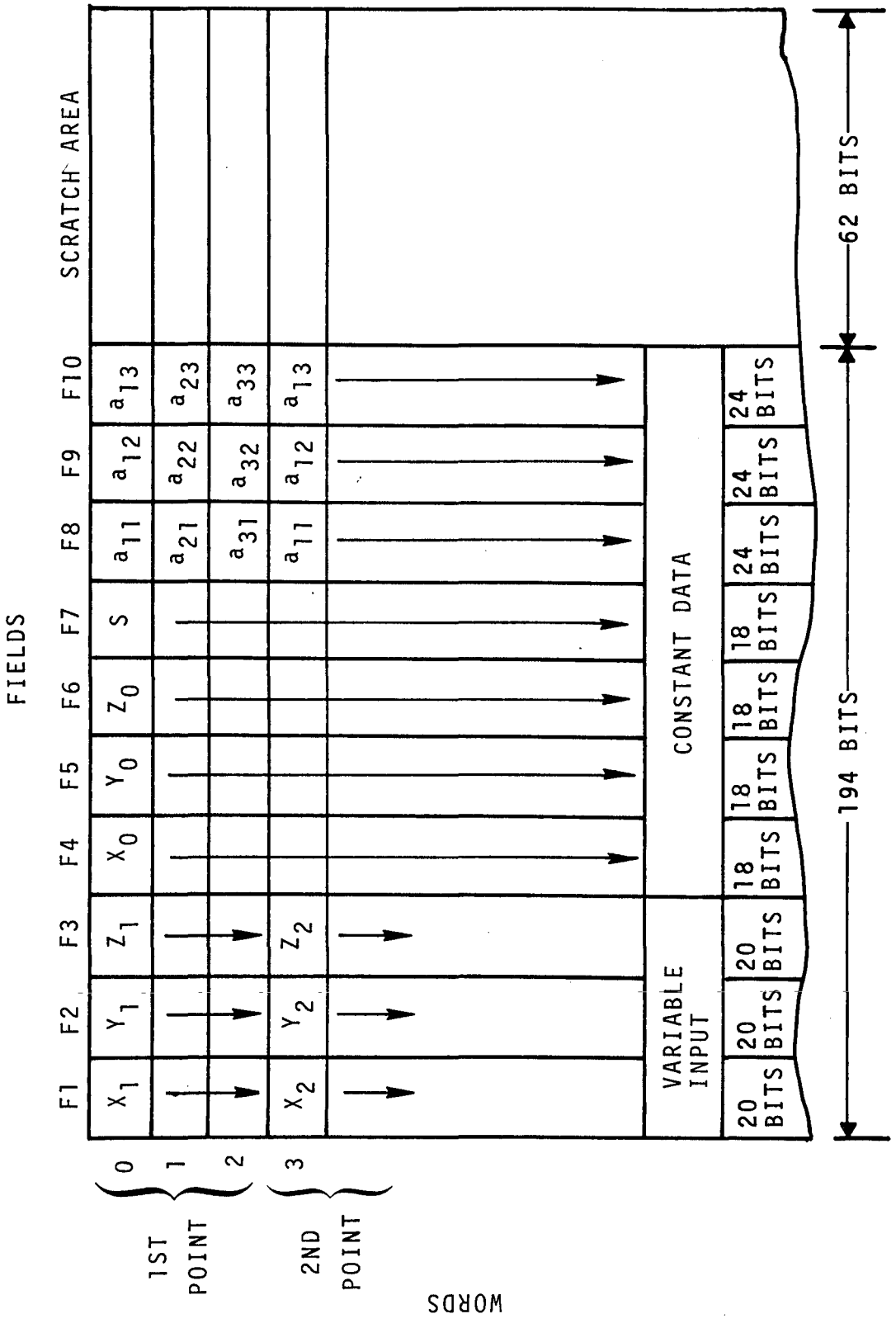


Figure 2-3 - Array Organization for Transformation Algorithm

The constant data X_0, Y_0,Z_0, and S are stored into Fields F4, F5, F6, and F7 of all words.

The other constant data, namely the elements a_{ij} of the rotation matrix, are stored into Fields F8, F9, and F10 of three words as shown in Figure 2-3. This sequence is then repeated for all words.

The scratch area is the remaining bits in each word, and is used for intermediate storage of arithmetic results. Organizing the data in this manner facilitates effective utilization of the parallel techniques available, and will be better understood after reading the discussion on array computations.

2.2.2.4 Array Loading Time - The constant data used throughout the processing are loaded from the high-speed data buffer (HSDB) area to all words of arrays participating in the processing. Since this is a "once only" operation compared to the loading times of the variables, the time is negligible and will not be estimated.

The variable data are loaded to Fields F1, F2, and F3 of every third word of the arrays through the 32-bit common register as shown in Figure 2-4.

The first word of each group of three words is then replicated in the other two words using parallel techniques.

The time required to transfer the variable data from bulk core memory to the associative arrays is estimated below:

(2.5×10^{-6}) (3) (150,000) = 1.125 sec.

2.5 μsec per 32-bit transfer from bulk core to the array.

3 transfers/point

150,000 data points

2.2.2.5 Data Replication Time - Replication is a parallel operation as illustrated in Figure 2-5.

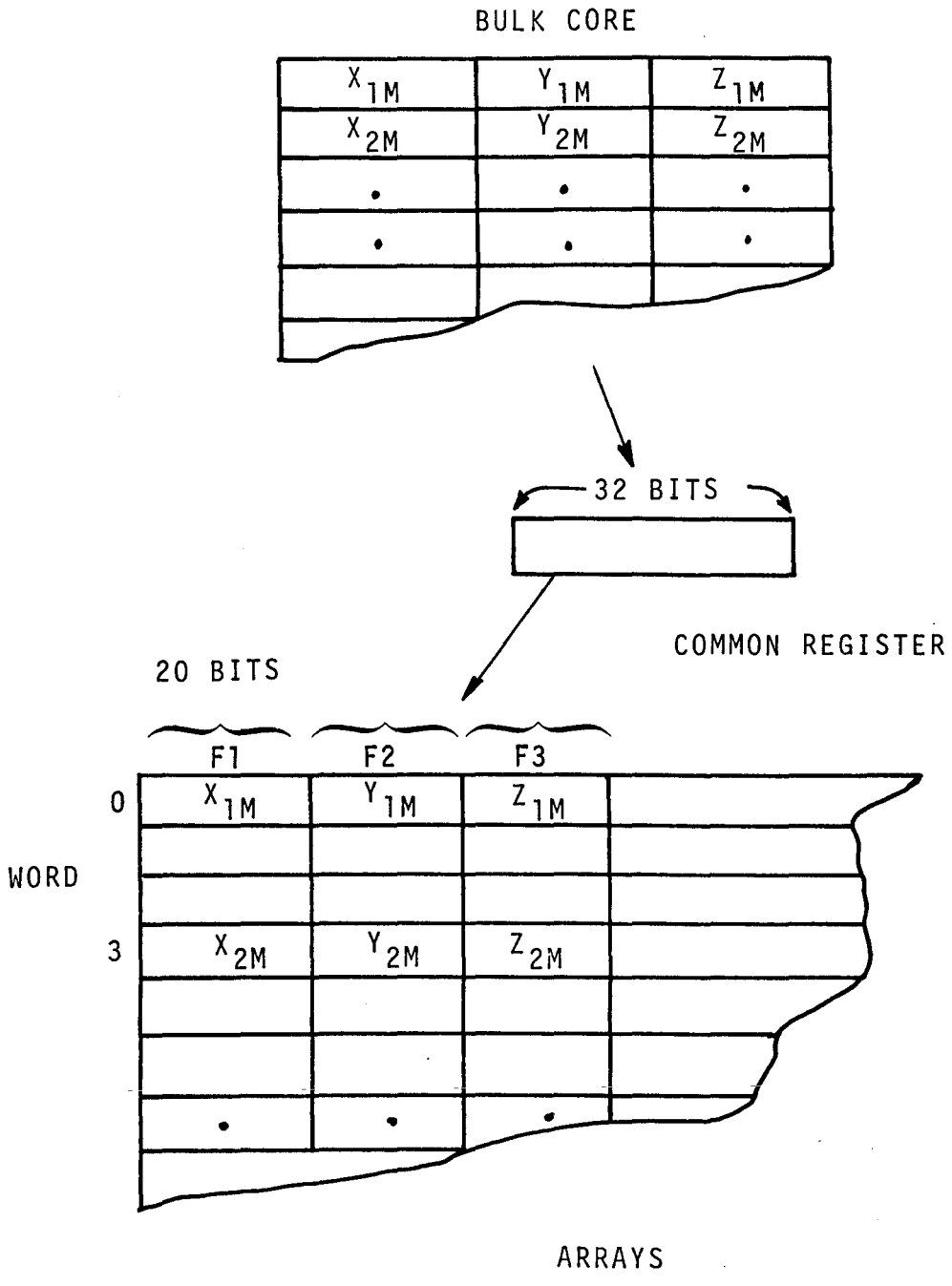


Figure 2-4 - Array Loading (Variable Input)

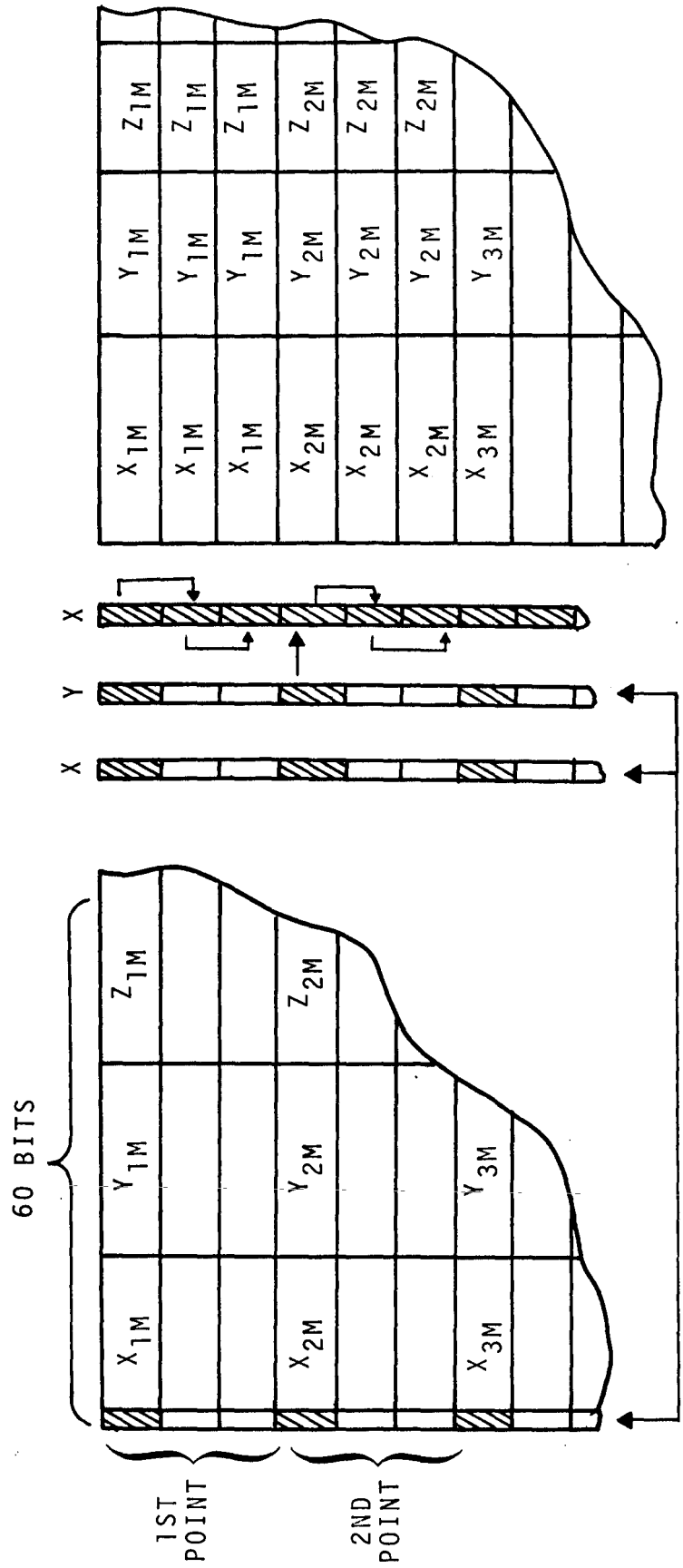


Figure 2-5 - Input Data Replication

The sequence of operations on each of the 60 (256 bit) columns is:

1. Load the X and Y registers (256 bits) with the bit column.
2. Rotate the Y register down one bit.
3. Logically OR the contents of X and Y; store the result in X.
4. Rotate the Y register down another bit.
5. Logically OR the contents of X and Y; store the result in X.
6. Store the X register back into the array bit column.

The time to perform the replication is estimated below:

$$\frac{(0.9 \times 10^{-6})(60)(150{,}000)}{(85)(4)} = 23.8 \text{ msec.}$$

0.9 μsec/bit to replicate into next two words
60 bits/data point (three 20-bit fields for X,Y,Z)
150,000 data points
85 points processed simultaneously/array
4 arrays

2.2.2.6 Array Computation Time - Once the arrays have been loaded, the parallel computations (performed on all words at the same time) are initiated. The computations are those required to perform the relationship shown in Equation 1.

The operations required to compute the results, and the order in which they are performed, are shown below.

1) Compute the following: $X_m - X_o$
$Y_m - Y_o$
$Z_m - Z_o$

The operations are performed by subtracting Fields F4,F5, F6 from F1,F2,F3 respectively and storing the results in F1,F2,F3. This requires three subtractions:

F1 - F4 → F1 (See Note on page 2-17)
F2 - F5 → F2
F3 - F6 → F2

Note: This means that the contents of F4 are subtracted from the contents of F1 and the result is stored in F1.

2) Compute the elements of array multiplications:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} X_m-X_o \\ Y_m-Y_o \\ Z_m-Z_o \end{pmatrix} ,$$

giving:

$$\begin{matrix} a_{11}(X_m-X_o) & a_{12}(Y_m-Y_o) & a_{13}(Z_m-Z_o) \\ a_{21}(X_m-X_o) & a_{22}(Y_m-Y_o) & a_{23}(Z_m-Z_o) \\ a_{31}(X_m-X_o) & a_{32}(Y_m-Y_o) & a_{33}(Z_m-Z_o) \end{matrix}$$

This requires three multiplications, and three move fields:

F1 x F8 → scratch area

Move significant 20 bits to F1.

F2 x F9 → scratch area

Move significant 20 bits to F2.

F3xF10 →scratch area

Move significant 20 bits to F3.

3) Sum the three values in each word:

$$a_{11}(X_m-X_o)+a_{12}(Y_m-Y_o)+a_{13}(Z_m-Z_o)$$
$$a_{21}(X_m-X_o)+a_{22}(Y_m-Y_o)+a_{23}(Z_m-Z_o)$$
$$a_{31}(X_m-X_o)+a_{32}(Y_m-Y_o)+a_{33}(Z_m-Z_o)$$

This requires two additions:

SCR(1)+SCR(2) → SCR(1)

SCR(1)+SCR(3) → SCR(1)

SCR is the scratch area.

4) Multiply the previous values by the scale factor:

$$Sa_{11}(X_m-X_o)+Sa_{12}(Y_m-Y_o)+Sa_{13}(Z_m-Z_o)$$
$$Sa_{21}(X_m-X_o)+Sa_{22}(Y_m-Y_o)+Sa_{23}(Z_m-Z_o)$$
$$Sa_{31}(X_m-X_o)+Sa_{32}(Y_m-Y_o)+Sa_{33}(Z_m-Z_o)$$

The total timing estimates to perform the array computations are:

$$\frac{(2.1 \times 10^{-3})(150,000)}{(85)(4)} = 0.926 \text{ sec.}$$

2.1 msec is required to compute 3 subtractions, 2 additions, and 4 multiplications (based on AAP timings in Appendix E),
150,000 data points
85 points/array
4 arrays

2.2.2.7 Array Unloading Time - Once the 20-bit results have been computed, they are unloaded via the 32 bit common register to an area of bulk core memory (see Figure 2-6). The array unloading timing estimates to transfer the data are shown below:

$(2.6 \times 10^{-6})(3)(150,000) = 1.17$ sec;

2.6 μsec per 32-bit transfer from the array to bulk core
3 transfers/point
150,000 data points

2.2.2.8 Total Estimated Execution Time - The total estimated times to perform the coordinate transformation for a model area were:

Array loading	=	1.170
Array replication	=	0.024
Array computation	=	0.930
Array loading	=	1.125
Total	=	3.349 sec

This total is increased by 25 percent to account for miscellaneous housekeeping items not discussed and also for any scaling during the computations:

Total = 3.349 x 1.25 = 4.18 sec;

therefore, coordinate transformation estimation = 4.18 sec.

2.2.3 Coordinate Transformation Validation

The coordinate transformation program was written and tested in the STARAN Evaluation and Training Facility (SETF) on the S-500 (two-array system). The timings obtained were modified to reflect the utilization of two more arrays. This modification was performed by doubling the array loading and unloading times because four arrays would have been used instead of the two arrays.

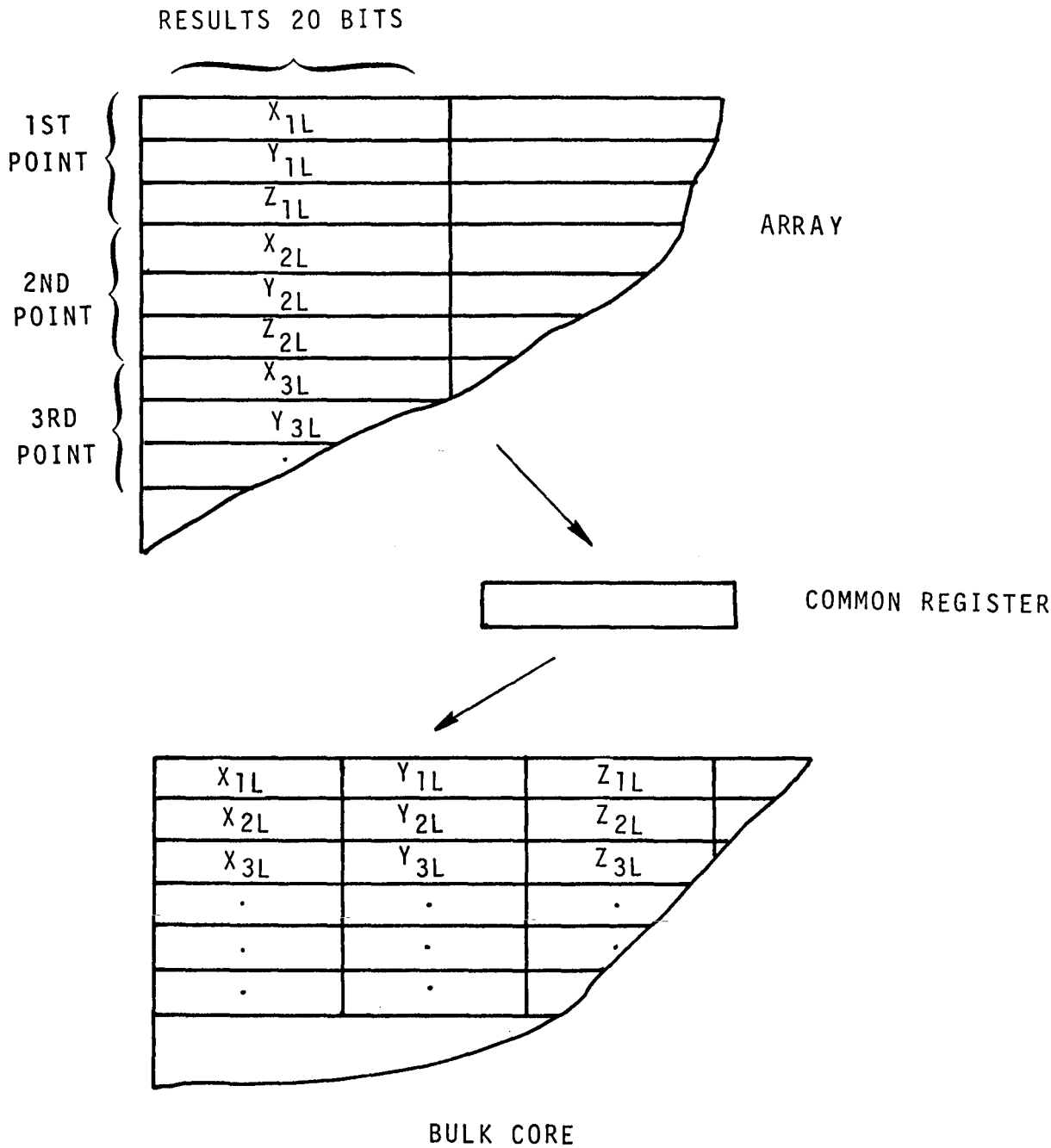


Figure 2-6 - Unloading Results

This modification resulted in the processing of twice as many points during each set of array computations. The array replication and computation times were the same because of the parallel processing capabilities of the STARAN.

Program timing is usually performed by using a performance monitor. A routine is initiated at designated points in the program and the time is automatically computed and printed out on completion of the routines. Since the STARAN S-500 was being upgraded to a STARAN S-1000 at the time of validation, the performance monitor was not available. However, a simple and effective alternate method was used.

The three sections of the program (array loading, computation, and unloading) were executed separately many thousands of times. An indicator (panel control light) was set upon completion of this process. The routines took several minutes to execute, and a stopwatch was used to measure the times.

The times then were divided by the number of routine executions to obtain the execution time of the tested routine for the number of points processed at one execution. This time was modified to reflect the utilization of two more arrays. This method has been used many times, and although the approach requires more time to test, it produces results that are just as accurate as those obtained when using the performance monitor. The results are shown in Table 2-III.

TABLE 2-III - MODIFIED VALIDATION TIMES FOR COORDINATE TRANSFORMATION

Routine	Number of times routine performed, N	Time,T, on S-500 (sec)	T/N= 85x2 pts* (msec)	2T/N 85x4 pts (msec)	Time/model (sec)
Array loading	$2x10^5$	296	1.48	2.96	1.31
Array replication	$2x10^6$	136	0.068	0.068	0.03
Array computations	$2x10^5$	490	2.45	2.45	1.08

*Three words/point; 42 points/array; two arrays.

TABLE 2-III - MODIFIED VALIDATION TIMES FOR COORDINATE TRANSFORMATION (Continued)

Routine	Number of times routine performed, N	Time,T, on S-500 (sec)	T/N= 85x2 pts* (msec)	2T/N 85x4 pts (msec)	Time/model (sec)
Array unloading	$2x10^5$	281	1.405	2.81	1.24
Total					3.66

*Three words/point; 42 points/array; two arrays.

N = number of times the routine was executed,
T = total time recorded for that routine on the S-500
T/N = time for one execution of the routine on the S-500
2T/N = modified time for one execution of the routine on the S-1000
Time/model = the modified time for the model area.

2.2.4 Interpolation Study

2.2.4.1 General - Figure 2-7 shows the overlapped area of two stereo photographs.

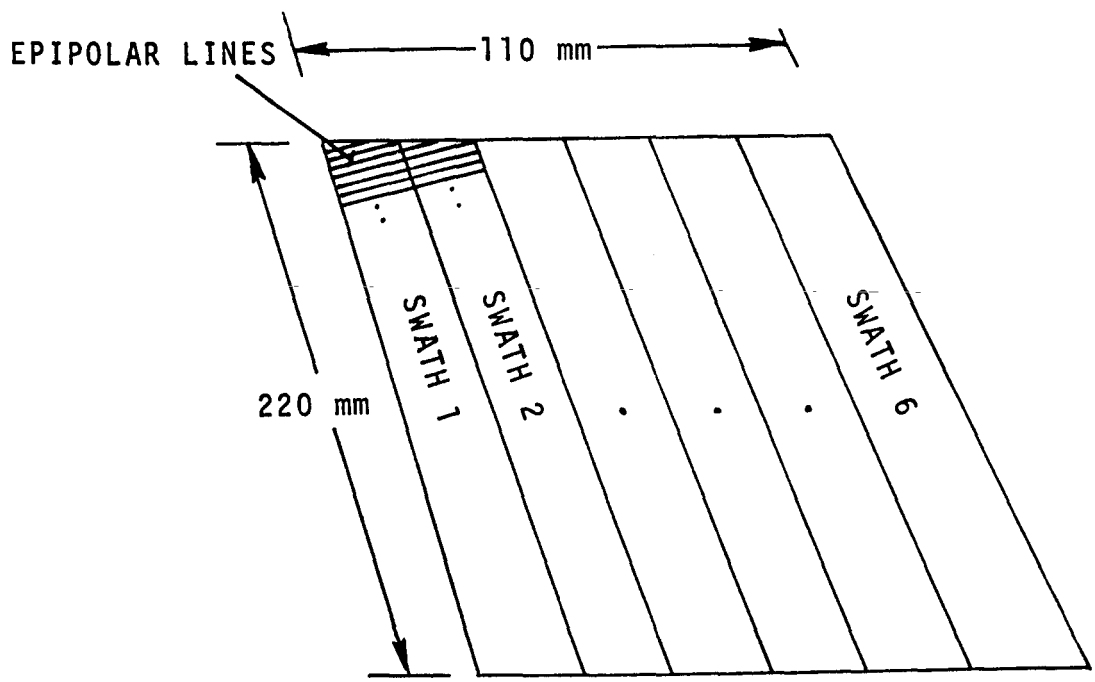


Figure 2-7 - Model Space Area

This overlapped area is called the model space area. The model space is divided into six equal strips termed swaths.

Data are organized on tape (or output directly from the AS-11BX) by swath and by epipolar line within each swath.

The diagrams illustrated in Figure 2-8 show a corner of the model space area. The AS-11BX performs a correlation along the epipolar lines to determine parallax and thus produce elevation values. The resulting elevations are located at irregularly spaced intervals along the epipolar lines as indicated by the X's in Figure 2-8A.

A maximum of 58 points can be generated on each epipolar line. For a model area containing 150,000 data points, this will be an average of 34 points/line.

The objective of the postprocessing interpolation function is to produce a set of regularly spaced (in X and Y) elevation values; these are illustrated by the O's at the intersections of the grid lines of Figure 2-8B. The grid lines are 0.5 mm apart in both vertical and horizontal directions. The first interpolation is performed by taking the five closest Z values (X's in Figure 2-8A) of each intersection point of the epipolar line that they lie on and the vertical grid lines. These sets of five points are used to generate the required values ('•' in Figure 2-8A).

When all the intersection points have been computed, they are used as the inputs to the second interpolation process. Again the five Z values ('•' in Figures 2-8A and 2-8B) closest to a grid intersection point are used to generate a value at that point. This process is performed for all grid intersections.

2.2.4.2 Algorithm - The interpolation method used was a least squares quadratic fit to the five closest points to the grid intersection point as shown in Figure 2-9.

The grid intersection point is the intersection of the X,Z axes.

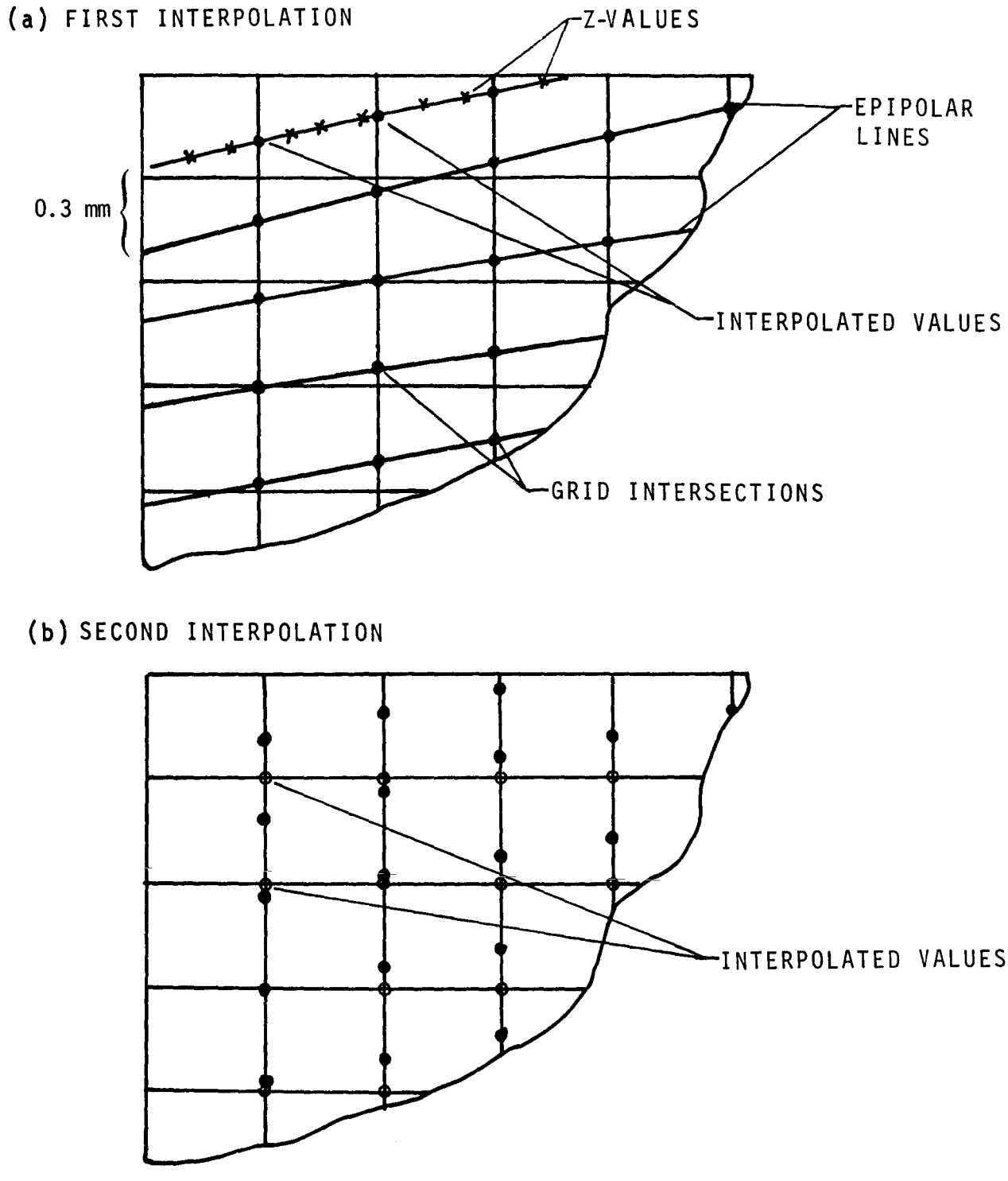


Figure 2-8 - Two-Dimensional Interpolations

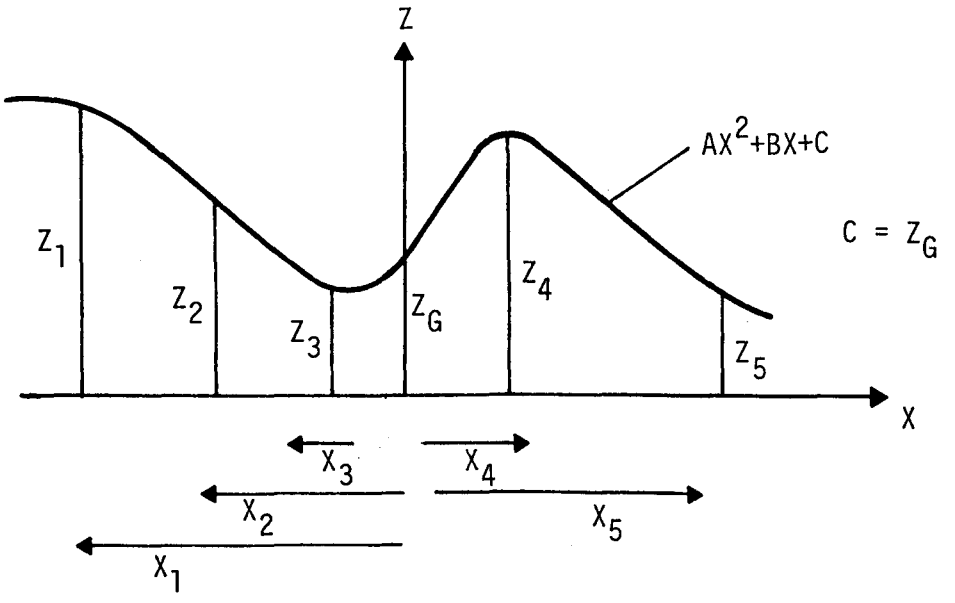


Figure 2-9 - Least Squares (Quadratic) Fit

The resulting least squares quadratic fit equation is shown below:

$$\begin{pmatrix} \Sigma Z_i \\ \Sigma X_i Z_i \\ \Sigma X_i^2 Z_i \end{pmatrix} = \begin{pmatrix} 5 & \Sigma X_i & \Sigma X_i^2 \\ \Sigma X_i & \Sigma X_i^2 & \Sigma X_i^3 \\ \Sigma X_i^2 & \Sigma X_i^3 & \Sigma X_i^4 \end{pmatrix} \begin{pmatrix} C \\ B \\ A \end{pmatrix} \qquad (3)$$

Because C is the elevation at the grid intersection point, this is the value of the quadratic equation that is required.

The X_i are defined as being the distances in the X-direction of each of the Z_i points from the grid intersection point Z_g. These values would be computed when determining the five elevation points (Z_i) closest to the grid intersection point. Performing this computation in the STARAN arrays is a relatively negligible operation and is not shown in this discussion.

Rearranging and solving Equation 3 for C using Cramer's rule gives:

$$C = \frac{\begin{vmatrix} \Sigma Z_i & \Sigma X_i & \Sigma X_i^2 \\ \Sigma X_i Z_i & \Sigma X_i^2 & \Sigma X_i^3 \\ \Sigma X_i^2 Z_i & \Sigma X_i^3 & \Sigma X_i^4 \end{vmatrix}}{\begin{vmatrix} 5 & \Sigma X_i & \Sigma X_i^2 \\ \Sigma X_i & \Sigma X_i^2 & \Sigma X_i^3 \\ \Sigma X_i^2 & \Sigma X_i^3 & \Sigma X_i^4 \end{vmatrix}} = Z_G.$$

Using the following notation for convenience, Equation 4 becomes:

$$R = \frac{\begin{vmatrix} F & B & C \\ G & C & D \\ H & D & E \end{vmatrix}}{\begin{vmatrix} A & B & C \\ B & C & D \\ C & D & E \end{vmatrix}}. \qquad (5)$$

Expanding Equation 5 produces:

$$R = \frac{FCE + GDC + HBD - (HC^2 + FD^2 + GBE)}{ACE + BDC + CBD - (C^3 + AD^2 + B^2E)} \qquad (6)$$

2.2.4.3 Array Organization - Because multiplications are the most time-consuming operations to be performed by the STARAN, the array fields are organized to reduce the number of multiplications while providing adequate scratch area within array words for additional computations.

By allocating the values corresponding to A,B,C ... etc. as shown in Figure 2-10, all elements of the numerator and denominator of Equation 6 can be computed by using only three multiplications. This allocation will produce the values shown in Figure 2-11.

With this array image in mind, the processing procedure will now be described. Figure 2-12 is a functional flow diagram of the interpolation program.

A set of constant masks (256 bit columns) are created and stored in the arrays. The variable X,Z values for each point are loaded from bulk core memory to the arrays. The interpolation algorithm is applied to generate the resulting values, which are then unloaded to bulk core memory.

The following sections describe each block of the functional flow diagram and includes the timing estimates.

2.2.4.4 Mask Generation Time - A set of eight masks were generated and stored in Bit Columns 245 through 252. These masks were 256 bit fields where each bit corresponded to a unique word of the array. These masks were used at different points in the program execution

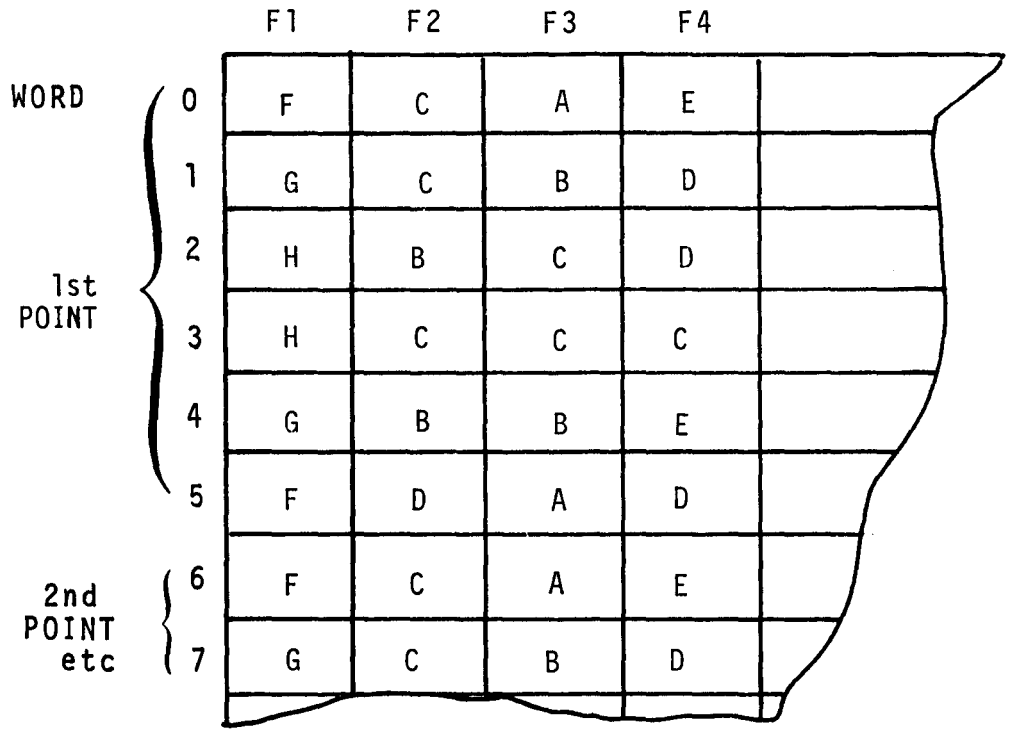


Figure 2-10 - Array Organization

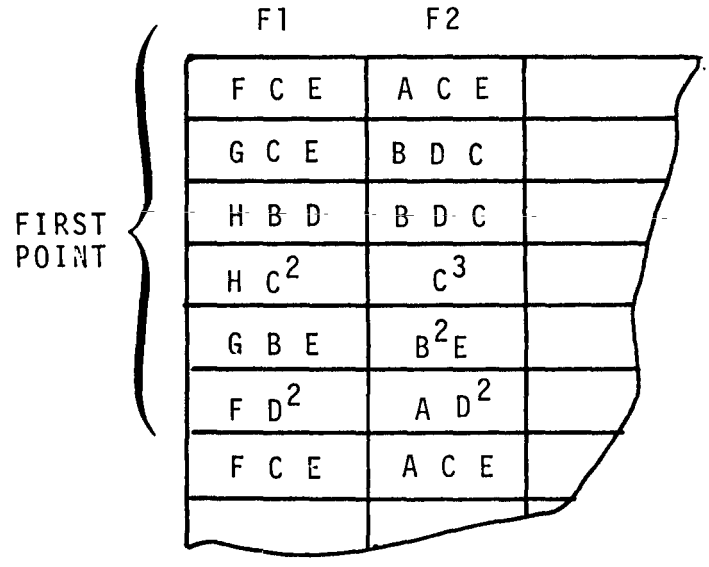


Figure 2-11 - Results of Array Multiplications

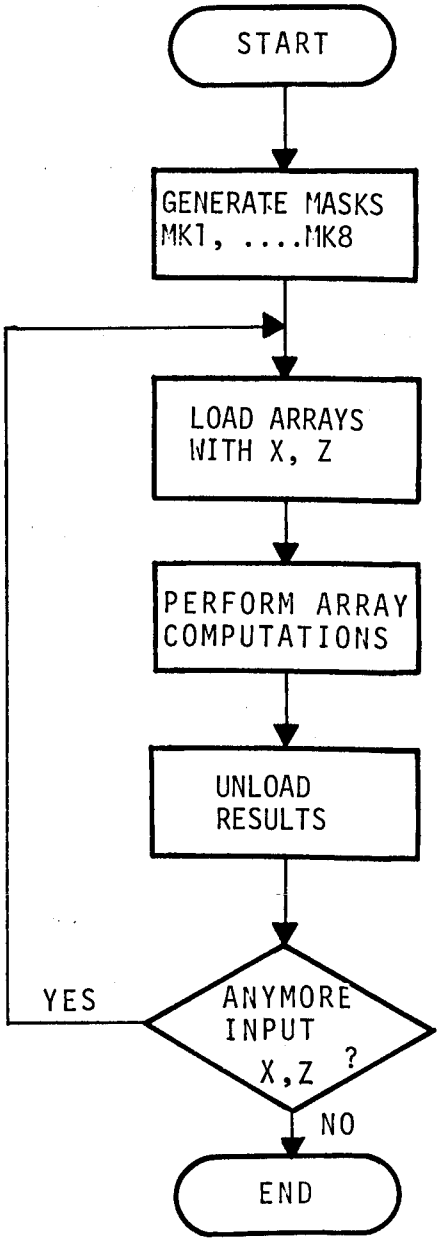


Figure 2-12 - Interpolation Flow Diagram

to select those words of the arrays that would participate in some particular computations (parallel mode operation). Because this is a "once only" operation and the same masks are used for all the interpolation data, the timing is negligible.

2.2.4.5 Array Loading Time - The arrays were loaded in a sequential manner through the 32-bit common register into Fields F1 and F3 as shown in Figure 2-13. The timing estimate to load the data for both interpolations is shown below:

$$(161{,}000 + 97{,}000)(2.65 \times 10^{-6})(10) = 6.84 \text{ sec}$$

2.65 μsec per 32-bit transfer from bulk core to the array.
10 transfers/point
161,000 interpolations (first direction)
97,000 interpolations (second direction)

The number of required interpolations for both directions is based on the following information:

Model space area = 220 mm x 110 mm
Output grid = 1/2 mm

Therefore, the number of output grid point elevations is:

$$(220 \times 1/2)(110 \times 1/2) = 96{,}800.$$

The amount of input data (epipolar lines) every 3/10 mm is

$$\frac{220}{3/10} = 733 \text{ epipolar lines.}$$

Therefore, the number of grid elevations is 733 x 110 x 1/2 or 161,260.

2.2.4.6 Array Computation Time - The operations required to compute the results, and the order in which they are performed, are shown below.

1) Generate the other elements required for the least squares fit :; X_i^2, X_i^3, X_i^4, X_iZ_i, $X_i^2Z_i$.

This step requires five multiplications, namely:

1. F1 x F1 → F2 (meaning the content of F1 was squared and the result was stored in F2)

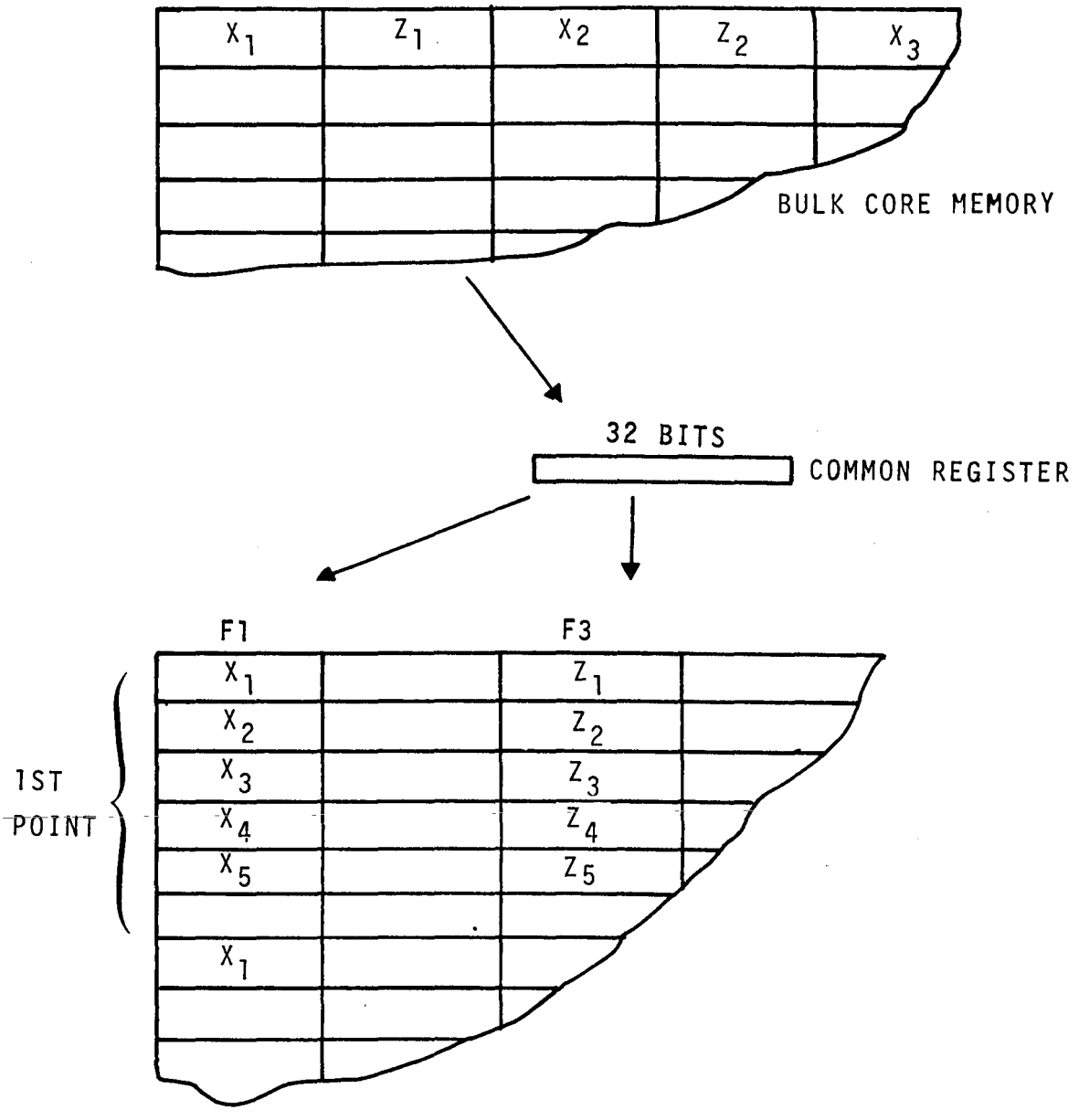


Figure 2-13 - Array Loading

2. F2 x F1 → F4
3. F4 x F1 → F5
4. F1 x F3 → F6
5. F2 x F3 → F7

These multiplications were performed in a parallel mode on all words in the arrays. The result is shown in Figure 2-14B

2) Sum the five elements of each field

The summation is obtained by performing a 'tree sum' (see Section for a discussion on tree summing).

A summation over a range of five words for one field requires:

3 shifts (word-to-word)
3 additions (field-plus field)

Therefore for seven fields, the following is required:

21 shifts
21 additions

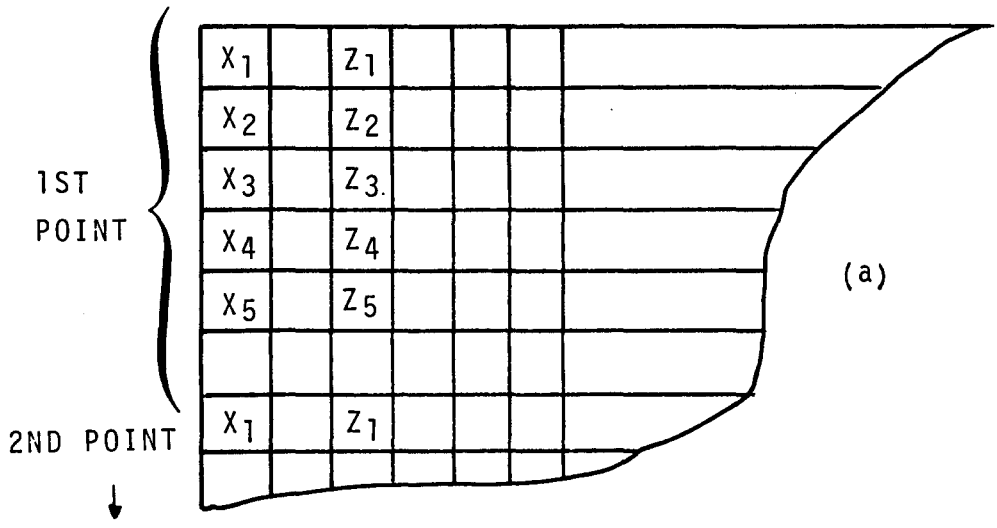
The resulting values are shown in Figure 2-15B.

Figure 2-15B is the same diagram but using the notation discussed earlier.

3) Array Replication

The replication process is similar to that discussed in the coordinate translation program. The sequence of operations on each of the seven fields (7 x 20 bits) is:

1. Load the X and Y registers (256 bits) with the bit column.
2. Rotate the Y register down one bit.
3. Logically OR the contents of X and Y; store the results in X.
4. Rotate the Y register down another bit.
5. Logically OR the contents of X and Y; store the result in X.
6. Rotate the Y register down another three bits.
7. Logically OR the contents of X and Y; store the results in X
8. Store the X register back into the array bit column.



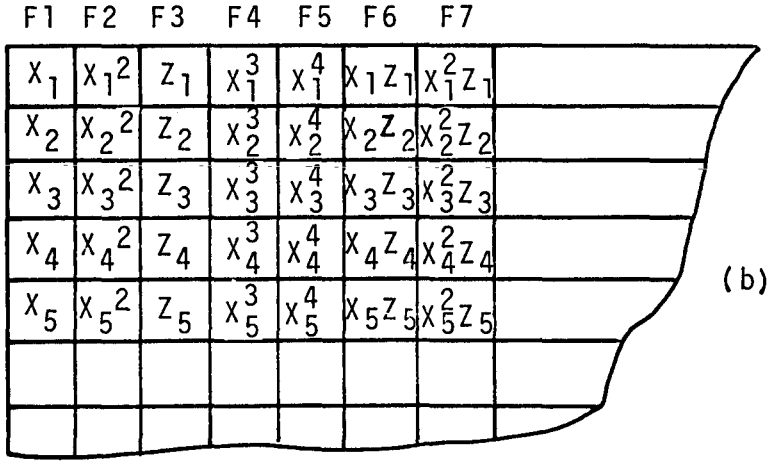


Figure 2-14 - Generation of Elements for Least Squares Fit

X, Y ARE 256-BIT STORAGE REGISTERS
*SHIFT Y DOWN ONE BIT POSITION
**LOGICALLY OR X AND Y AND STORE RESULT IN X.

Figure 2-15 - Replication Process

Figure 2-16 shows the sequence of operations Figure 2-15C shows the result.

The time to perform the replication is estimated below:

$$\frac{(1.35 \times 10^{6})(140)(258{,}000)}{(85)(4)} = 143 \text{ msec.}$$

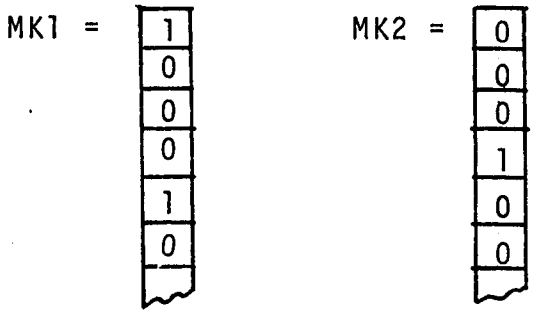
1.35 μsec/bit to replicate into next five words
140 bits/data point (seven 20-bit fields).
258,000 (161,000 + 97,000) data points
42 points processed simultaneously/array
4 arrays

4) Move field-to-field operations

At this point the masks, MK1, MK2, ... MK8, were used in field-to-field masked move operations to generate Figure 2-15D.

The sketch below shows how the new fields were generated.

Example: Generate Field F4 of Figure 2-15D from Figure 2-18C.
Assume:



These six-bit fields are repeated through the first 252 bits of the 256-bit masks.

The operations are as follows:

1. Load M register with MK1
2. Move F5 to F4
3. Load M register with MK2
4. Move F2 to F4

The time to perform the total operations (see Figure 2-15) was approximately:

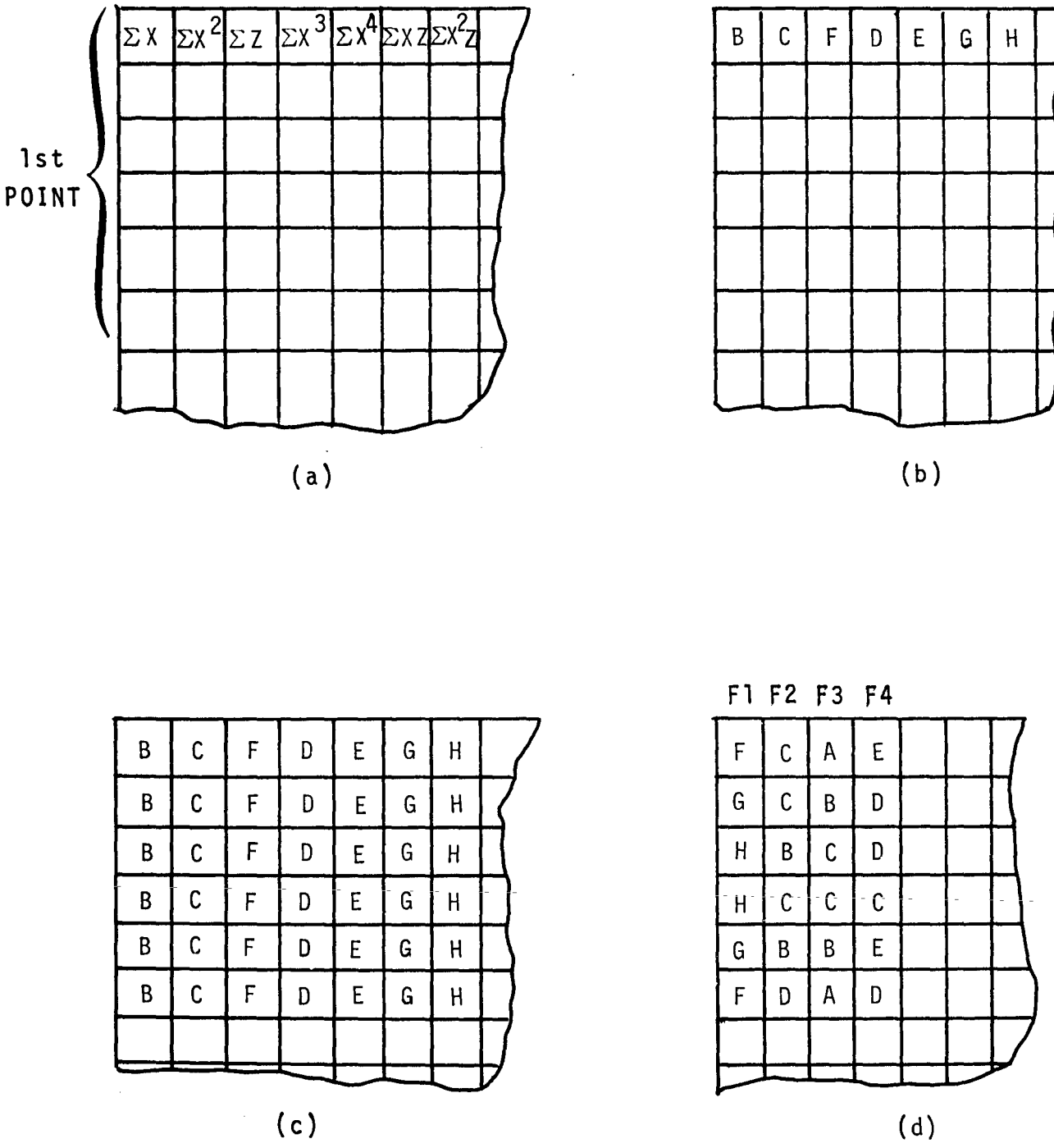


Figure 2-16 - Field-to-Field Manipulations

10 Load M register instructions
10 Move field-to-field instructions

Figure 2-15D corresponds to Figure 2-10 which was discussed under the array organization.

5) Obtain elements of numerator and denominator

By performing three multiplications, all elements of the numerator and denominator of the least squares fit are obtained (see Figure 2-17A). The operations are:

$F2 \cdot F4 \rightarrow SCR(1)$
$F1 \cdot SCR(1) \rightarrow F1$
$F3 \cdot SCR(1) \rightarrow F2$

6) Sum the values (numerator and denominator)

The negative results of the last three values in F1 and F2 are taken and the six words (for each point) are tree summed for both fields. The resulting values are located in Fields F1 and F2 of the first word of each point (see Figure 2-17B). The summation over the six words requires:

6 shifts
6 additions

7) Divide the numerator by the denominator

The result is obtained by dividing the contents of Field F1 by F2:

$F1 \div F2 \rightarrow F1$

The final result is shown in Figure 2-17C.

The total timing estimates to perform the array computations are:

$$\frac{(258{,}000)(4.9 \times 10^{-3})}{42 \times 4} + 0.143 \text{ sec} = 7.53 \text{ sec},$$

where 4.9 msec is required to compute:

8 multiplies
1 divide
13 moves
27 shifts
27 adds
10 load M registers (negligible)

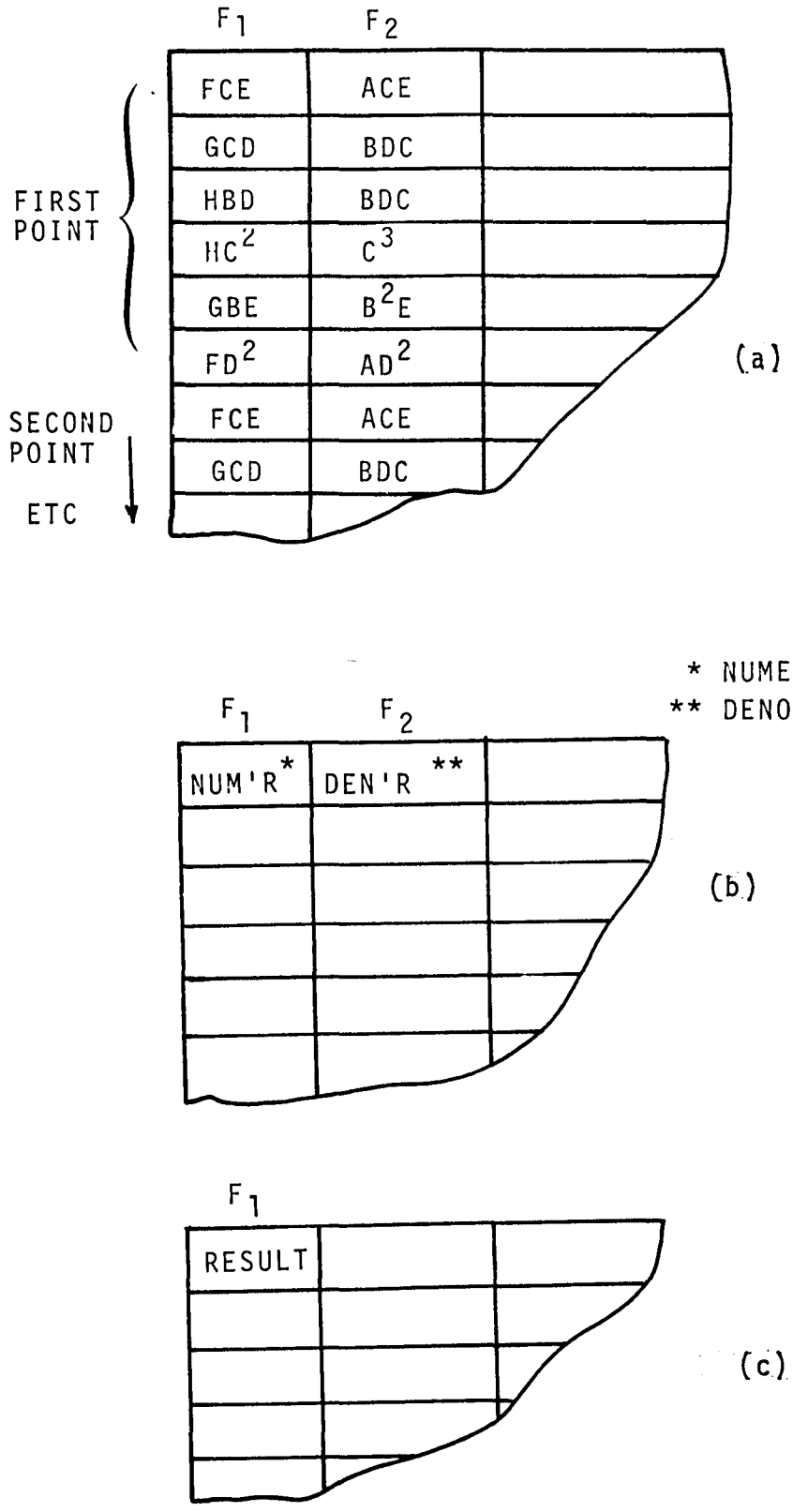


Figure 2-17 - Final Result of Computations

and there are:

258,000 data points
42 points/array
4 arrays
0.143 msec = replication time

2.2.4.7 Array Unloading Time - Once the 20-bit results have been computed, they are unloaded via the 32-bit common register to an area of bulk core memory.

The array unloading timing estimates to transfer the data are shown below.

$(258{,}000)(3\times10^{-6})$ sec = 0.78 sec

3.0 μsec per 32-bit transfer from the array to bulk core
258,000 points

2.2.4.8 Total Estimated Execution Time - The total estimated times to perform the interpolation (two dimensional) for the model area was:

Item	Time (sec)
Array loading	6.84
Array computations	7.53
Array unloading	0.78
Total	15.15 sec

This estimate was increased by 25 percent to account for miscellaneous housekeeping items not discussed and also for any scaling during the computations:

Total = 15.15 x 1.25 = 18.94 sec;

2.2.5 Interpolation Validation

The interpolation program was written and tested in the SETF on a STARAN S-500 (two-array) associative array processor.

The timings obtained were modified to reflect the utilization of two more arrays. This modification was performed by doubling the array loading and unloading times because four arrays would have

been used instead of the two arrays. This modification resulted in the processing of twice as many points during each set of array computations. The array replication and computation times were the same because of the parallel processing capabilities of the STARAN.

The same method of timing was used in this interpolation validation program as that in the coordinate transformation program. A similar set of results are shown in Table 2-IV.

TABLE 2-IV - MODIFIED VALIDATION TIMES FOR INTERPOLATION

Routine	No. of times routine performed, N	Time, T on S-500 (sec)	Two arrays T/N (msecs)	Four arrays, 2T/N (msecs)	Time/model (sec)
Array loading	1×10^5	482	4.82	9.64	7.32
Array computation	1×10^4	205.8	20.58	20.58	15.62
Array unload-	1×10^6	553	0.553	1.106	0.84
				Total	23.78

Legend:

N = number of times routine was performed
T = total time (in seconds) that the routine took on the S-500 (two array machine)
T/N = resulting time for one execution of the routine on the S-500
2T/N = modified time to execute routine using two more arrays
Time model = time to process the whole model area

2.2.6 System Hardware Considerations

2.2.6.1 General - The STARAN system sizing considerations and three general hardware approaches for implementing a STARAN associative array processor to perform the AS-11BX postprocessing tasks

are described below. The three hardware approaches include an on-line system, off-line system, and somewhat specialized high-speed configuration.

2.2.6.2 System Sizing - Table 2-V lists memory capacities for the standard STARAN memory components. Table 2-VI lists various items to be considered in sizing a STARAN system for the basic AS-11BX postprocessing tasks of coordinate transformation and interpolation.

The Page 0 memory size is a function of the number of subroutine instructions, while Pages 1 and 2 memory sizes are a function of the number of program instructions. Tables 2-V and 2-VI show that the standard capacity page memories can adequately contain the subroutine and program instructions. The instructions can be alternately executed (in blocks) from Page 1 and Page 2 memories. Should the program instructions exceed the capacity of the page memories, instructions can be executed from one page memory while the next block of instructions are transferred into the other page memory from bulk core.

TABLE 2-V - STARAN MEMORY CAPACITY

Memory	Size (words)	
	Standard	Maximum
Page 0 (32 bits)	512	1024
Page 1 (32 bits)	512	1024
Page 2 (32 bits)	512	1024
High-speed data buffer (32 bits)	512	1024
Bulk core (32 bits)	16K	32K
Associative arrays (256 bits)	256 (1 array)	8192 (32 arrays)

The high-speed data buffer provides some transfer time advantage compared to the bulk core and therefore is used to store various constants required to implement the algorithms. According to Tables 2-V and 2-VI, the standard capacity high-speed data buffer will adequately contain the problem constants.

TABLE 2-VI - STARAN SYSTEM SIZING CONSIDERATIONS

Item	Quantity
Program instructions	626 words
Subroutine instructions	172 words
Miscellaneous constants	44 words
AS-11BX output data (X,Y,Z)	450,000 per model area
AS-11BX processing rate	30 min per model area
STARAN postprocessing rates	
S-250 (1 array)	69 sec per model area
S-500 (2 array)	44 sec per model area
S-1000 (4 array)	23 sec per model area

In general, the bulk core memory size is a function of the required amount of data buffering while the number of associative arrays is a function of the required STARAN performance.

Since STARAN can service many AS-11BX stereo mappers simultaneously (see Table 2-VI) and the volume of AS-11BX output data is large, a data block size had to be selected that would not create a severe data buffering problem. For example, if the data blocks were equal to a model area of data, a 450,000-word buffer would be required for each AS-11BX. Therefore a data block size of approximately 10,000 words was selected.

This data block size represents an area of 34 by 99 points with each point requiring three 18-bit words (X, Y, and Z). This size was selected because:

1. The AS-11BX generates an average of 34 elevations along each epipolar line (across a swath).
2. By using approximately 99 epipolar lines of data (34 by 99 points), sufficient data are available to efficiently perform the corner-turning required to support the interpolation process.
3. The assumption was made that STARAN could directly access the AS-11BX memory and that 10,000 words could be made available in each stereo mapper for buffer storage.

4. The 10,000-word data block could be readily accommodated in STARAN's 32,000 word (maximum size) bulk core memory.

As a result of the above considerations, assumptions, and approach, the minimum standard STARAN required to implement the basic AS-11BX postprocessing tasks is given in Table 2-VII. This system would be capable of simultaneously servicing approximately 25 AS-11BX stereomappers.

TABLE 2-VII - MINIMUM STANDARD STARAN MEMORY REQUIRED FOR BASIC AS-11BX POSTPROCESSING TASKS

Memory	Size (words)
Page 0	512
Page 1	512
Page 2	512
High-speed data buffer	512
Bulk core	32K (twice basic)
Associative arrays	256 (1 array)

2.2.6.3 System Configurations - The study results (Section 2.1.5) indicate that STARAN is capable of simultaneously servicing multiple AS-11BX stereomappers directly on-line, in real time, or off-line, at a fast rate. This section describes three simplified system configurations that could accomplish the basic AS-11BX post-processing tasks.

The first approach utilizes a standard STARAN interfaced directly to the AS-11BX devices via STARAN's standard 32-bit interface channel. In the second system configuration, STARAN would process the AS-11BX data in an off-line mode. This system would essentially be the same as the one above except that the AS-11BX data would be obtained from magnetic tape. The third hardware configuration also utilizes a standard STARAN, but it is interfaced to the AS-11BX devices via its 256-bit (per array) interface channel. In addition this system incorporates a hypothetical hardware floating point arithmetic option.

The first approach is pictured in Figure 2-18. In this figure, each stereomapper would independently generate elevation data until it accumulated a block of data (approximately 3366 points; 10,098 words). At this point the stereomapper would interrupt the multiplexer and then continue generating elevation data for its next data block.

The multiplexer functions will be to pass all interrupts to the custom interface unit and in response to STARAN commands, data, and control signals between the AS-11BX devices and the custom interface unit.

The custom interface unit will generate a code for each stereomapper interrupt that will serve to identify the particular device that originated the interrupt. This code then will be transferred to STARAN. The custom interface unit also performs under STARAN command; the data transfer of AS-11BX output data to STARAN, and the transfer of processed data from STARAN to the magnetic tape units.

Upon receipt of the interrupt code, STARAN will write this code into a "status table" in chronological order. At the appropriate time, STARAN then will check the table to determine which block of AS-11BX data will be processed next.

As shown in Figure 2-18, STARAN's bulk core memory will be divided into three areas: an input buffer and two output buffers. Since the STARAN was assumed to access the AS-11BX directly, the time required to transfer a data block into STARAN will be a small part of the postprocessing time. Therefore, after issuing a command to the custom interface unit for a new block of data, STARAN will wait for its input buffer to be filled before processing the data.

However, the time required to transfer the processed data to the output tape units will be a significant part of the postprocessing time. Therefore, the two output buffers will be required to permit the transfer of processed data to the tape units while a new block of data is being processed.

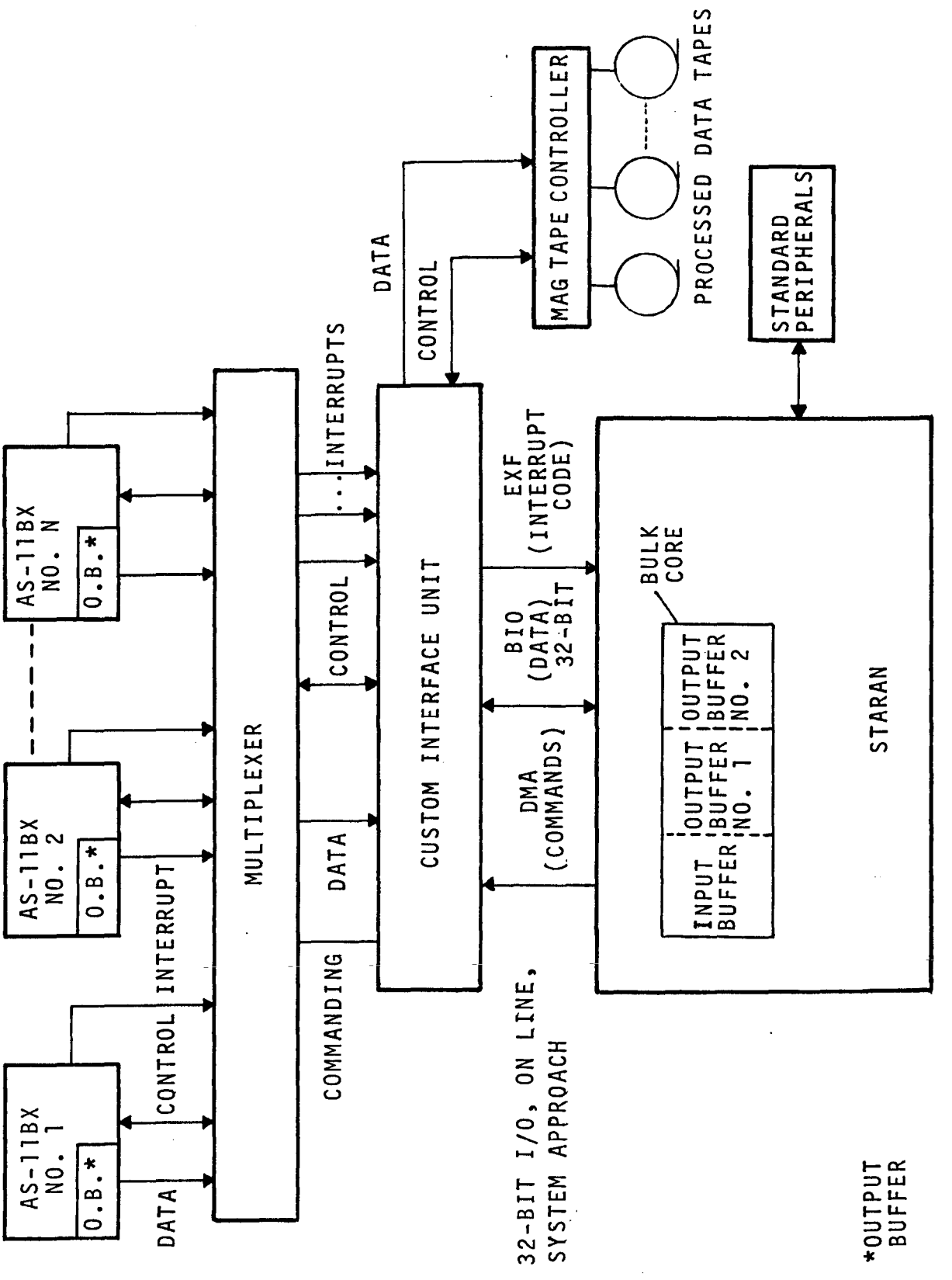


Figure 2-18 - On-line System for AS-11BX Postprocessing Using STARAN's 32-bit Interface

When STARAN completes the postprocessing for a data block, the processed data will be in one of the output buffers of bulk core memory. Then STARAN will:

1. Command the custom interface unit to transfer the processed data to the appropriate tape unit.
2. Check the "status table" to determine which AS-11BX has the "oldest" data block ready for postprocessing.
3. Command the custom interface unit to transfer the "oldest" data block from the appropriate AS-11BX to the input buffer of STARAN's bulk core memory.

STARAN then will process the new data block and store the results in the available output buffer.

The second system configuration, an off-line approach, is shown in Figure 2-19. Since the magnetic tape I/O time in one direction (input or output) does not exceed the postprocessing time (see Section 2.1.5), the AS-11BX data feasibly can be processed off-line from magnetic tape units without significantly affecting the overall postprocessing performance. To accomplish this processing, a smaller data block size must be selected and STARAN's bulk core memory divided into four parts: two input buffers and two output buffers.

While STARAN uses an input and output buffer to process one block of data, the last data block processed will be transferred from the other output buffer to the output tapes and the next data block will be transferred from the input tapes to the other input buffer. By buffering the input and output data in this manner, the tape input/output times can be ignored because they do not exceed the postprocessing time. In this way, the overall system performance will not be appreciably affected.

The above off-line implementation argument will not apply to the next system configuration to be discussed because the postprocessing time will be significantly less than the tape I/O time. Therefore the tape I/O time would not permit full realization of the improved STARAN performance.

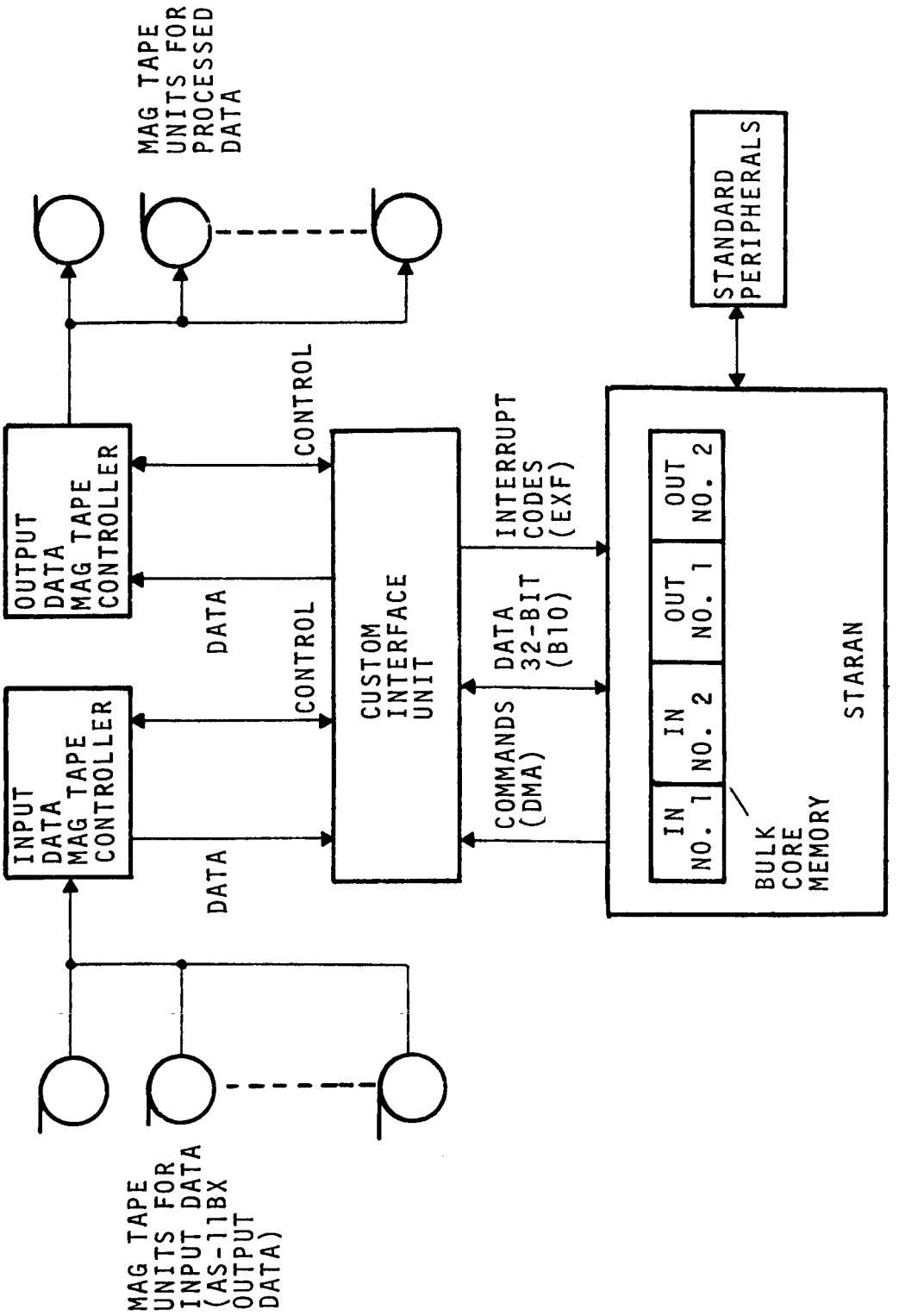


Figure 2-19 - Off-line System for AS-11BX Postprocessing Using STARAN's 32-bit Interface

Figure 2-20 shows STARAN interfaced to the AS-11BX stereomapper through its 256-bit (per array) I/O channel. Also included are hypothetical hardware (floating point) arithmetic units. The objective of this configuration is to reduce the associative array load/unload time and illustrate a method by which the performance could be improved for arithmetic operations.

The overall system operation is about the same as that shown in Figure 2-18 with some additions. The multiplexer/buffer will have the additional function of providing buffer storage for two data blocks of information. One buffer will be for new data from the stereomappers and the other will be for processed data. The basic purpose of the buffers will be to reformat the data for STARAN's 256-bit data channel.

In addition to performing all the functions previously described for the custom interface unit (see Figure 2-18), the parallel I/O unit also controls the 256-bit data channel operation. Any pair of parallel I/O unit ports can be interconnected via software control to permit a 256-bit data channel between STARAN and an external device. Each transfer would take approximately 0.5 μsec.

The hardware arithmetic units shown in Figure 2-20 have not been developed, but were included to give some indication of how the performance of a standard STARAN could be significantly improved should the need arise. All words of all arrays would be simultaneously transferred to their respective hardware arithmetic units in a bit-serial manner. The desired arithmetic operation would be performed and the results then would be transferred back to the arrays in a bit-serial manner. Performance for the hardware arithmetic units was estimated at 3 μsec per bit. Implementation approaches for hardware arithmetic are described in Appendix F.

The three system configurations described above accomplish the basic AS-11BX postprocessing tasks of coordinate transformation and interpolation in an idealized hands-off mode. However, a truly

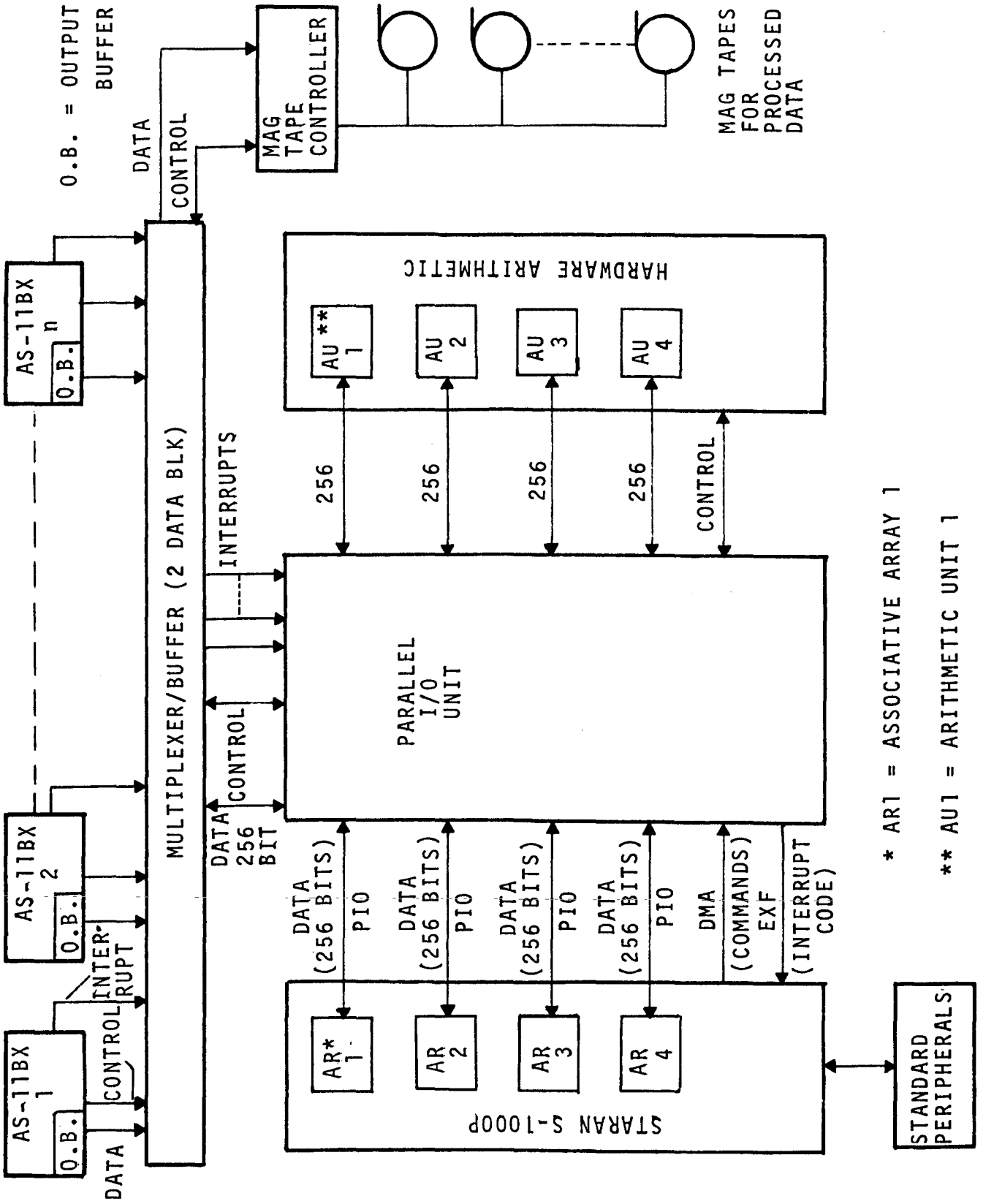


Figure 2-20 - On-line System for AS-11BX Postprocessing Using STARANs' 256--bit Interface Channel and Hardware Arithmetic

operational system would require manual data verification and editing during the postprocessing sequence. Therefore, one or more interactive displays with appropriate control would be required for the man-machine interface. Also, it may be desirable, but not mandatory, that a host sequential computer be utilized for various housekeeping tasks and overall system control. The host computer, interactive displays, and control equipment could be interfaced to the above described systems via the custom interface unit.

2.2.7 Miscellaneous Housekeeping Considerations

In a truly operational system, manual data verification and editing would be an integral part of the process requiring a significant amount of housekeeping. However, only those housekeeping items required to directly support the basic postprocessing functions evaluated in this study are discussed below.

Table 2-VIII shows the minimum functions required to maintain an operational system in a real-time environment.

While the STARAN can handle the miscellaneous support tasks, a host sequential computer could also do just as well, and therefore the available system configuration will determine the best allocation of these tasks.

Each function in Table 2-VIII is described briefly below: A block of data is read into the STARAN memory from the AS-11BX machines via some multiplexing device (discussed under Section 2.2.6). A block of data would consist of approximately 34 x 99 points. A sort program is initiated to identify and group the sets of five points closest to each interpolation point prior to performing that interpolation. The sort is required before both interpolations.

After the first interpolation, the results (R_i) are ordered along the epipolar lines (R_1, R_2, R_3, R_4, ... R_{x-1}, R_x) as shown in Figure 2-21. Prior to performing the second interpolation, it is necessary to corner-turn and use the results ordered in the vertical direction (R_1, R_{x+1}, R_{3x+1} ... etc.). This step is done

TABLE 2-VIII - GENERAL SYSTEM FUNCTIONS

Task	Function	Time (sec)
1	Read a block* of data into bulk core from the AS-11BX	0.11
2+	Transform the data from the model coordinate system to the local coordinate system	0.093
3	Sort the data for interpolation in one direction	0.075
4≠	Interpolate	0.26
5	Perform corner-turning on results (address modification)	§
6	Sort the data for interpolation in the other direction	0.075
7≠	Interpolate	0.16
8	Transfer data from bulk core memory to storage device	0.47
	Total	1.245

*There are 45 blocks per model area.

+Described under coordinate transformation study (Section 2.2.2) and validation (Section 2.2.3).

≠Described under interpolation study (Section 2.2.4) and validation (Section 2.2.5)

§This time is negligible compared with the other operations (part of the array unloading time).

by outputting the results to address-modified locations that perform the corner-turning operations.

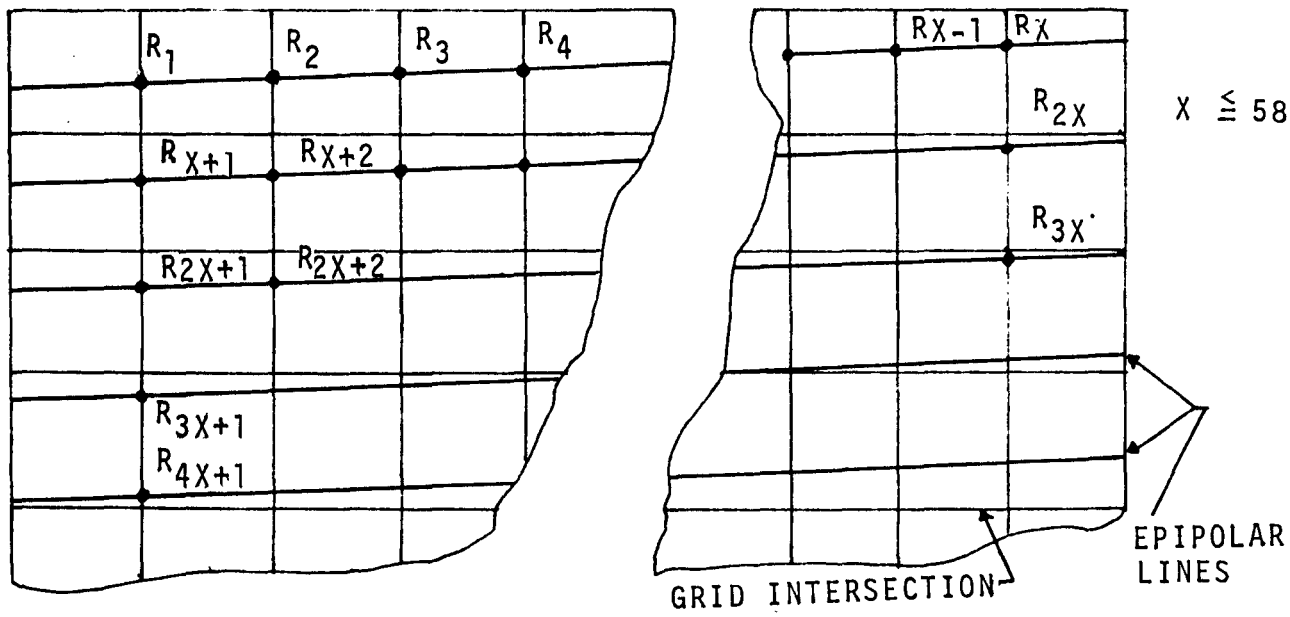


Figure 2-21 - Section of 34 x 99 Point Block of Swath

The results from the second interpolation (located in bulk core memory) are written onto magnetic tape.

2.2.8 Additional Postprocessing Tasks

During the course of the study, discussions with ETL and DMA indicated that additional AS-11BX postprocessing tasks might be applicable to associative array processing. Since the study results show that a STARAN AAP can very effectively execute some basic postprocessing tasks, it would be logical to evaluate STARAN's effectiveness for additional functions.

Some additional postprocessing tasks are listed and discussed below:

1. Computation of model data (elevations) from parallax data.
2. Application of model deformation corrections (MDC) to the model data.

3. Transformation of points from a local coordinate system to a geographic coordinate system.

A minicomputer within the AS-11BX presently computes the elevations from the parallax data. The AAP could significantly reduce the AS-11BX processing time by assuming this task.

Model deformation corrections must be applied to the model coordinates to correct for distortion created by the panoramic stereo exposures.

If X,Y,Z is a point in the model coordinate system, then the equations below define the corrected X_c, Y_c, Z_c point:

$$X_c = a_0 + a_1X + a_2Y + a_3X^2 + a_4XY + a_5Y^2 + a_6X^3 + a_7X^2Y + a_8XY^2 + a_9Y^3$$

$$Y_c = b_0 + b_1X + b_2Y + b_3X^2 + b_4XY + b_5Y^2 + B_6X^3 + b_7X^2Y + b_8XY^2 + b_9Y^3$$

$$Z_c = C_0 + C_1X + C_2Y + C_3X^2 + C_4XY + C_5Y^2 + C_6X^3 + C_7X^2Y + C_8XY^2 + C_9Y3 + C_{10}Z$$

The geographic coordinate system provides a better reference for a large area of the earth's surface and also provides a unique reference for a group of local coordinate systems.

A geographic reference system for the points is obtained by transforming those points referenced to the local coordinate system, otherwise known as a local space rectangular (LSR) system, to a universal space rectangular (USR) reference system at the earth's center. These geocentrically referenced points can be transformed to a geographic system of longitude, latitude, and height.

In addition to the postprocessing tasks discussed above, consideration is being given to an interpolation scheme that utilizes "weighted - distance - averaging". This study used a two-dimensional least squares quadratic fit as the interpolation technique. The "weighted - distance-averaging" technique would compute the elevations at each grid point by applying weighting factors to the elevations around each grid point based upon their

respective distances from the grid point. This method would remove the corner-turning requirement and reduce the sorting task. The computations involved in this process would probably be less complicated and require fewer multiplications as terms of the fourth power (needed in the least squares technique) would not be involved.

3. RASTER PROCESSING-AUTOMATED CARTOGRAPHY

3.1 INVESTIGATION

3.1.1 Background

Raster processing is that part of automated cartography that operates on properly positioned line and character data to generate correct map symbology. In the approach described below, the line and character data are digitized by a scanner/plotter device and processed by a general-purpose digital computer. The same scanner/plotter then is used to produce the final color separation negatives.

A simplified overview of an automated cartography process is shown in Figure 3-1. This process begins with stereo pairs of aerial photos and concludes with a map of the photographed area. The UNAMACE is a stereomapper that uses stereo pair inputs to generate orthophotos and profile data. The planimetric compiler is used to manually trace and generate an overlay for each class of planimetric data (roads, streams, railroads, etc.) on the orthophoto. CONPLOT is one of several software programs that utilize profile data to generate contour lines that are plotted on an overlay.

A maximum of 12 overlays can be accommodated by the system software with each overlay containing the positional information for a particular class of map data. The cartographic scanner/plotter is initially used to scan and digitize each overlay and store the digitized data on magnetic tape; one tape per overlay. At this point the data processor performs the raster processing and outputs up to five magnetic tapes; one for each color separation negative. The cartographic scanner/plotter is employed again to plot the properly symbolized map data on the color separation negatives. The final map is printed with plates produced from the negatives.

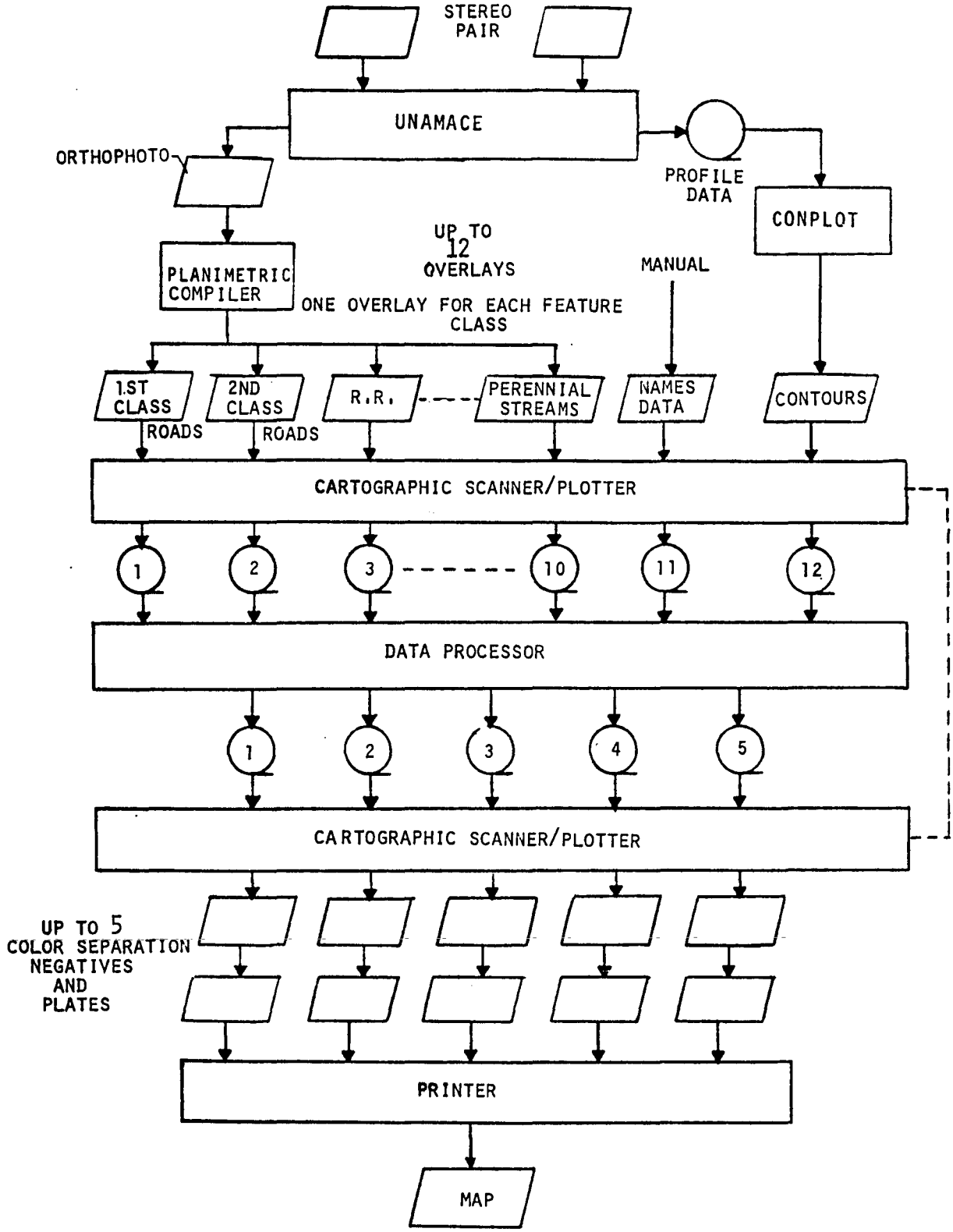


Figure 3-1 - Automated Cartography Overview

In addition to the automated functions within raster processing, there are requirements for manual verification and editing at various intervals of the raster processing sequence. Therefore, to provide the man-machine interface required for a total system, Figure 3-1 would normally include appropriate interactive displays and controls connected to the data processor.

Information and reference data pertinent to the raster processing tasks were derived from the ETL-CR-72-12 cartographic scanner/plotter final technical report. This report was submitted to the U.S. Army Engineer Topographic Laboratories at Fort Belvoir, Virginia, by International Business Machines Corporation, Federal Systems Division of Gaithersburg, Maryland, under Contract DAAK02-71-C-0139.

The approach described in the IBM final report is implemented on an IBM 360/40 sequential computer and the raster processing tasks are divided into four major software programs. These programs and their respective tasks are listed below:

1. MAP 01 - preliminary processing
    a. Registration mark detection* (see note below)
    b. Line separation
    c. Symbol recognition*
    d. Skew correction*
    e. Array formatting
    f. Line thinning*
    g. Color separation
    h. Line vectorization*
2. MAP 02 - image edit
    a. Edits recognized symbols
    b. Character association
    c. Line break detection and correction*
    d. Line smoothing*
    e. Manual editing*
3. MAP 03 - output tape processing
    a. Raster generation*
    b. Line symbol, area pattern, and symbol generation*

4. MAP 04 - output tape formatting
    a. Merging of all MAP 03 output tapes corresponding to a color separation*.

Note: the functions listed above with asterisks were executed and timed for a sample problem involving eight input overlays and three color separation negatives.

The map area (and therefore each overlay) was approximately 192 sq. in. (12 by 16 in.) and each overlay was scanned at four milli-inch centers. The resulting map for this sample problem was produced from the IBM final report and is shown in Figure 3-2. The final report shows the total execution time measured on an IBM 360/40 to be approximately 13 1/2 hr. From ETL data, the most time consuming raster processing function was found to be line thinning and vectorization that require approximately 4-1/4 hr. or about 1/3 of the overall execution time.

3.1.2 Objectives

In general, the study objective was to evaluate the suitability of a STARAN associative array processor when applied to raster processing tasks. More specifically, the objectives were to develop plausible parallel processing techniques for the most time-consuming raster processing functions, estimate their execution times on a STARAN S-1000 (4 arrays), and compare the results to those experienced with the present IBM 360/40 system implementation. An additional objective was to implement several routines in Goodyear's STARAN evaluation and training facility to validate the estimated execution times.

3.1.3 Study

The main thrust of the raster processing study was initially directed toward line thinning, vectorization, and skew correction. As stated previously, the first two items account for approximately 35 percent of the processing time in the presently implemented IBM 360/40 system. However, initial study results were favorable toward the STARAN AAP and therefore, additional raster processing tasks were included in the study to provide a broader base from which to evaluate STARAN effectiveness.

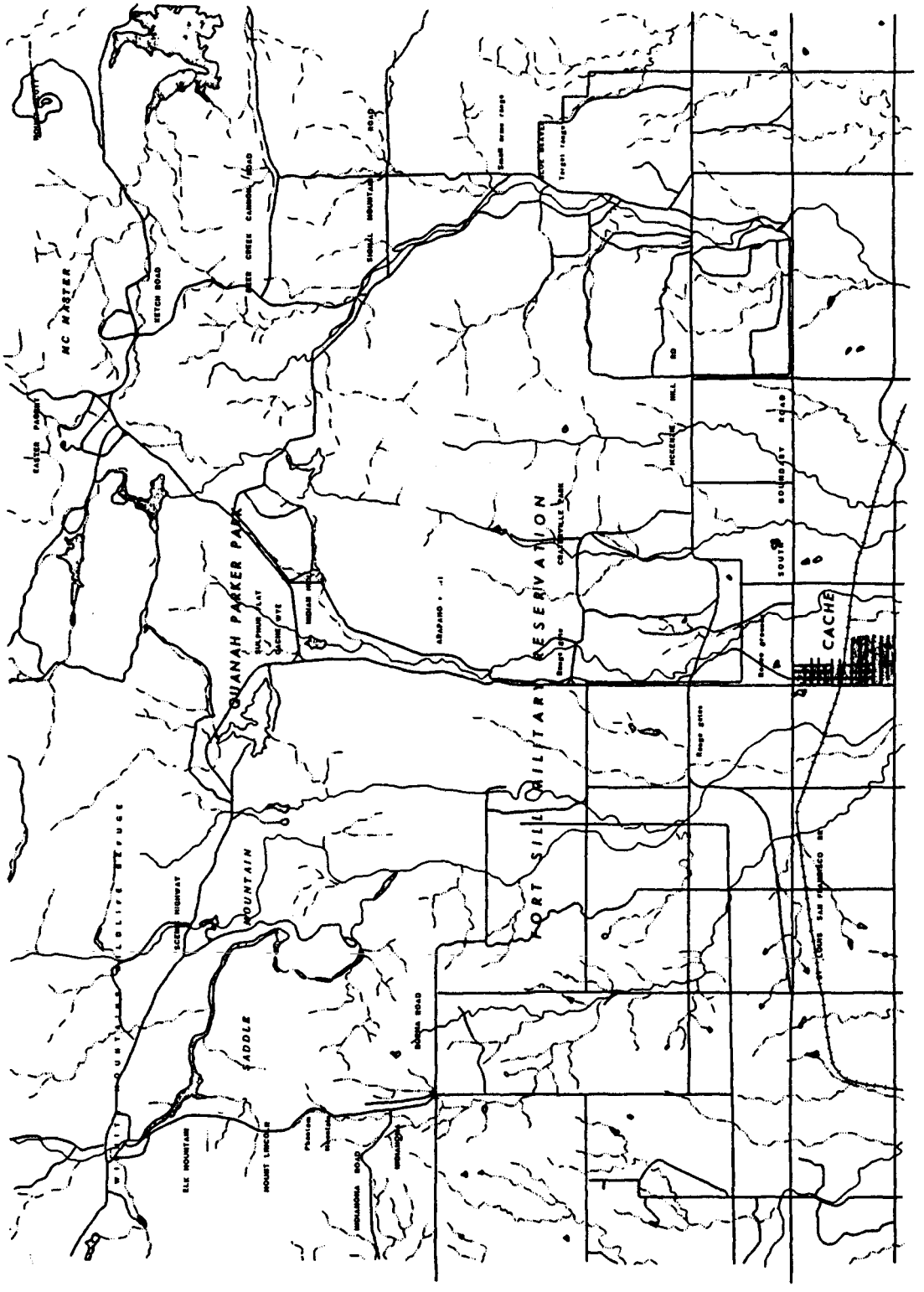


Figure 3-2 - Resulting Map for 8-Overlay Sample Problem

Because of this increased scope, the tasks were evaluated with respect to their major operations that were assumed to represent most of the processing time. The increased scope limited the attention that could be given to the overall system considerations and miscellaneous housekeeping tasks, but this was still considered the most effective way to achieve the desired study objectives.

The raster processing functions considered for STARAN application are listed below:

1. Registration mark detection
2. Line separation
3. Character recognition
4. Line thinning
5. Vectorization
6. Skew correction
7. Line break detection
8. Line smoothing
9. Line symbol generation

The above functions represent most of the MAP 01, MAP 02, and MAP 03 program tasks. The limited time available precluded a complete investigation of all raster processing functions such as color merging, and symbol generation for railroads and intermittent streams. However, basic approaches to these functions have been considered and can be further developed at a later date.

Two basic approaches were considered to implement the required functions. One utilizes existing procedures and algorithms modified only to the extent required to permit execution within the AAP. The second approach reconstructs an area of the pattern within the STARAN arrays and performs all tasks (including symbol generation) before data are transferred to temporary storage for color merging. Most algorithms were developed for the second approach because this approach could most effectively utilize STARAN's array architecture. However, the registration mark detection, line separation, and character recognition functions

were generally implemented in accordance with the first approach as their algorithms could be effectively adapted for AAP processing.

Timing estimates were derived for a standard STARAN S-1000 (4 arrays) using the APPLE assembly language performance times given in Appendix E. Array loading and unloading was assumed to take place via the standard 32-bit interface path. All estimates were increased by 25 percent to account for miscellaneous overhead factors. The sample problem described previously for the IBM 360/40 implementation was used in this study to provide a common reference between the execution times experienced with the IBM 360/40 and those estimated for the STARAN AAP. The STARAN timing was therefore estimated for data that represented a 192 sq. in. map area with 8 overlay inputs and 3 color separation negatives as output. The overlays were scanned at 4 milli-inch centers, which allows each 256-bit associative array to store approximately one square inch of the overlay area.

3.1.4 Validation

The raster processing validation effort dealt with line thinning and line symbol generation (single and double line thickening). The former represents the most time-consuming operations and the two functions together represent the most fundamental raster processing tasks. A test pattern program was written to produce a set of bit configurations (array images) that could be used to validate the line thinning and line thickening functions. All programs were tested in the SETF on the STARAN S-500 (two array) system. These test patterns consisted of horizontal and vertical intersecting lines, a 45-deg line, and a circular line, each line being nine bits wide. The line thinning algorithm then was used to thin the test pattern image. The original algorithm was not totally satisfactory and was modified to produce something closer to the desired results.

Validation of the symbol generation program was restricted to single- and double-line thickening. A series of line-thickening

programs (TH1K3, TH1K5, TH1K9) were written to generate single lines of various widths.

A utility program was written to display the results with a binary (ones and zero's) printout rather than the usual hexadecimal format. This program also simplified checking and evaluating the results. The elapsed processing time was measured using the STARAN performance monitor. This device is an internal counter that measured the time to perform single or multiple machine instructions.

3.1.5 Results

Timing estimates for the raster processing functions considered in this study effort are shown in Table 3-I. This table contains the processing times estimated for a STARAN S-1000 (4 arrays) and measured for an IBM 360/40. The STARAN times included a 25 percent overhead factor. The timing for both systems involves a 192 sq in. map area with 8 overlay inputs (scanned on 4 milli-inch centers), and 3 color separation negatives as the output. The results show that STARAN has a several hundred to one time advantage over the IBM 360/40.

The number of overlays affected by each operation are included in Table 3-I for reference. Question marks were used to indicate functions for which execution times were not obtained. Therefore, the "total" IBM 360/40 execution time does not include all functions listed in the table.

Line thinning and vectorization are combined in Table 3-I because the IBM 360/40 execution times were provided this way. Vectorization, however, has no signifance for STARAN since the AAP approach uses the raster image intact. Line symbol generation includes single-line, double-line, railroad, and intermittent stream symbology. The last two items were not given detailed consideration during this study, but were estimated by assuming that the processing would be 10 times more complex than that required for a single-line symbol. In the AAP approach, line break detection, and line smoothing can be accomplished in one operation and therefore one time is shown for both.

TABLE 3-I - TIMING RESULTS FOR RASTER PROCESSING STUDY*

Operation	No. of overlays	IBM 360/40 (actuals)	STARAN S-1000 (Estimated)
Registration mark detection	8	7 min	0.084 sec
Line separation	1	?	26.25 sec
Character recognition	1	1.0 Hr	1.25 sec
Line thinning and (vectorization)	7	4.2 Hr	12.5 sec
Skew correction	8	1.4 Hr	0.25 sec
Line break detection	7	1.4 Hr	6.25 sec
Line smoothing	7	1.4 Hr	
Line symbol Generation	7	?	5.4 sec
	Totals	9.5 Hr	51.6 sec+

*12 x 16 in. map area with 8 overlays and 4-mil resolution.

+Requires approximately 5.5 min of tape I/O.

Tape I/O time was not included in the STARAN estimates because STARAN execution times are so short that in an actual application, many completely buffered scanner/plotters would be serviced simultaneously. With this completely buffered arrangement, all scanner/plotters (for on-line operation) or many tape units (for off-line operation) would be loading random access memory buffers simultaneosuly and STARAN would then directly access and process a block of data as it was accumulated in the buffers.

The validation results for line thinning and line symbol generation are shown in Table 3-II. Included in the table for reference are the corresponding estimated execution times. The validated results are within 60 percent of the estimated times for the single line symbol generation function and much closer for the others.

Part of the additional validation time was due to additional steps required to eliminate "holes" that resulted because of the finite image resolution. Additional validation time was also encountered

TABLE 3-II - TIMING RESULTS FOR RASTER PROCESSING VALIDATION

Function	STARAN S-1000 calculation	
	Estimated (sec)	Validated (sec)
Line thinning	4.4	4.3
Single line symbol generation	0.21	0.33
Double line symbol generation	0.88	0.97

because the algorithm was programmed in a manner that minimized the effort required to verify the function. With additional time, there are areas within the single line symbol generation program that could be streamlined to significantly reduce the execution time.

3.2 DISCUSSION

3.2.1 General

Details of the raster processing study-validation effort are discussed below. The specific functions addressed during this study are:

1. Registration mark detection
2. Line thinning and "vectorization"
3. Skew correction
4. Line symbol generation
5. Line break detection
6. Line smoothing
7. Line separation
8. Character recognition

Description and illustrations are provided to convey the basic concepts involved and the associative array techniques utilized to implement the above functions with a STARAN AAP. Also included are derivations of the estimated exection times for the above functions.

The discussion validation contains a description of the methods used to execute and time the software programs written for the line thinning and line symbol generation algorithms.

3.2.2 Registration Mark Detection

A detailed description of the operations and timings required to perform the registration mark detection are given below. The general approach is the same as that used in the existing system except the STARAN approach utilizes its array features to simplify the task. The areas of the overlay assumed to contain the registration marks are stored in the four arrays. The object of this task is to verify that these areas do contain the registration marks and then to calculate the X and Y coordinates of their central points. These steps are performed by the following operations:

1. Store the areas assumed to contain the registration marks into the arrays (see Figure 3-3A). Each of the four registration marks would be stored in one of the four arrays of the STARAN S-1000.
2. Sum the 'ON' cells in one direction (vertical or horizontal) for all bit or word slices (see Figure 3-3B). An 8-bit working (scratch) area is reserved in each array; eight bits are required to allow for a maximum count of $256(2^8)$ 'ON' cells. The summation is performed by moving each 256-bit column (or row) to the least significant bit column of the working area shown in Figure 3-4. All the 'ON' cells are summed using a parallel technique called "tree summing" (described in Appendix G), which requires 8 additions. The resulting 8-bit value then is stored in a second scratch area. This process is performed on all columns (or rows) of the image, as shown in Figure 3-4.
3. Integrate the summed "ON" bits as shown in Figure 3-3C. This integration is an accumulation of all values at points up to a desired point of interest. For this problem, the first point is added to the second point and the result is stored in the second point. This value is then added to, and stored in, the third point, etc. Sequentially this method

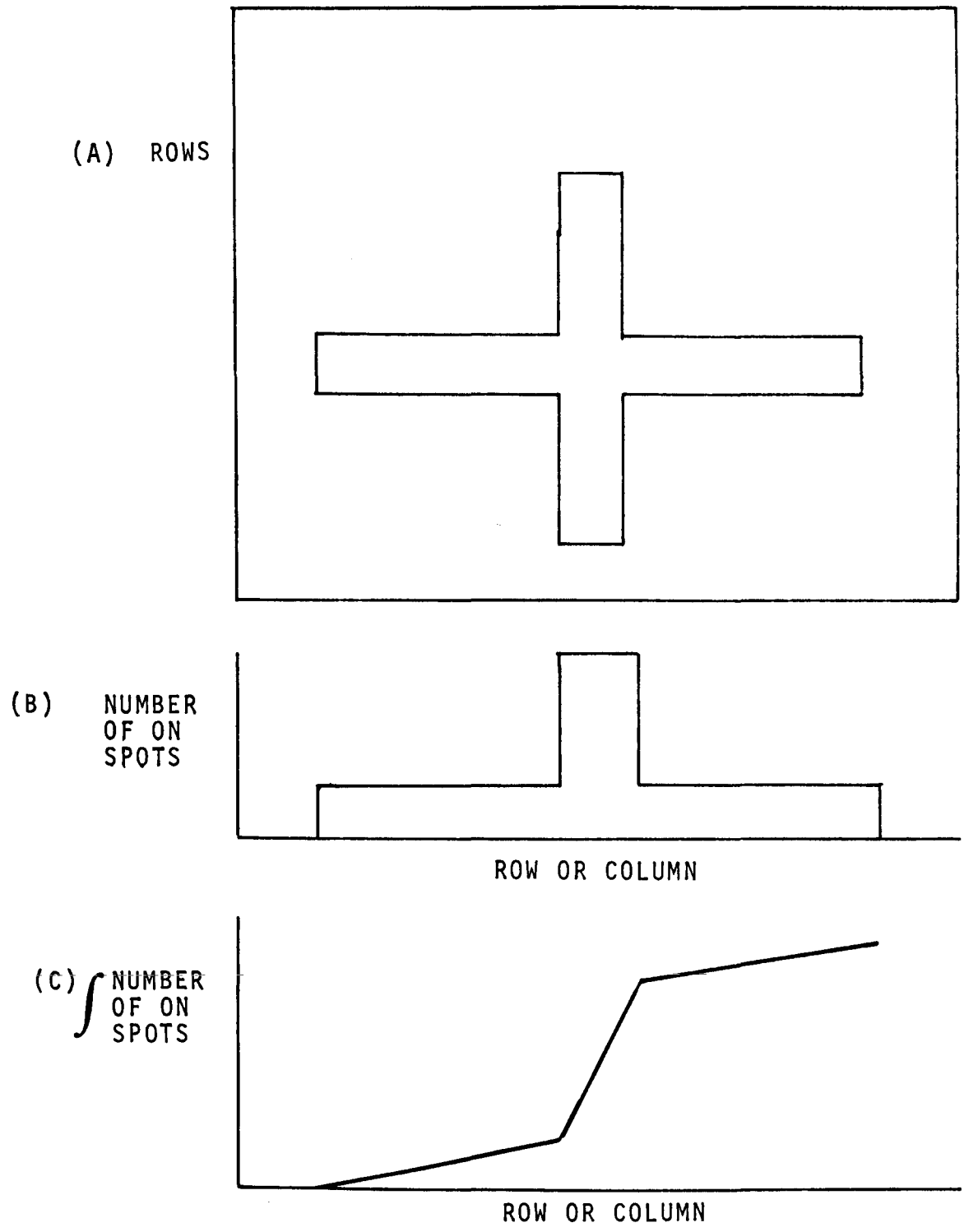


Figure 3-3 - Registration Mark Detection

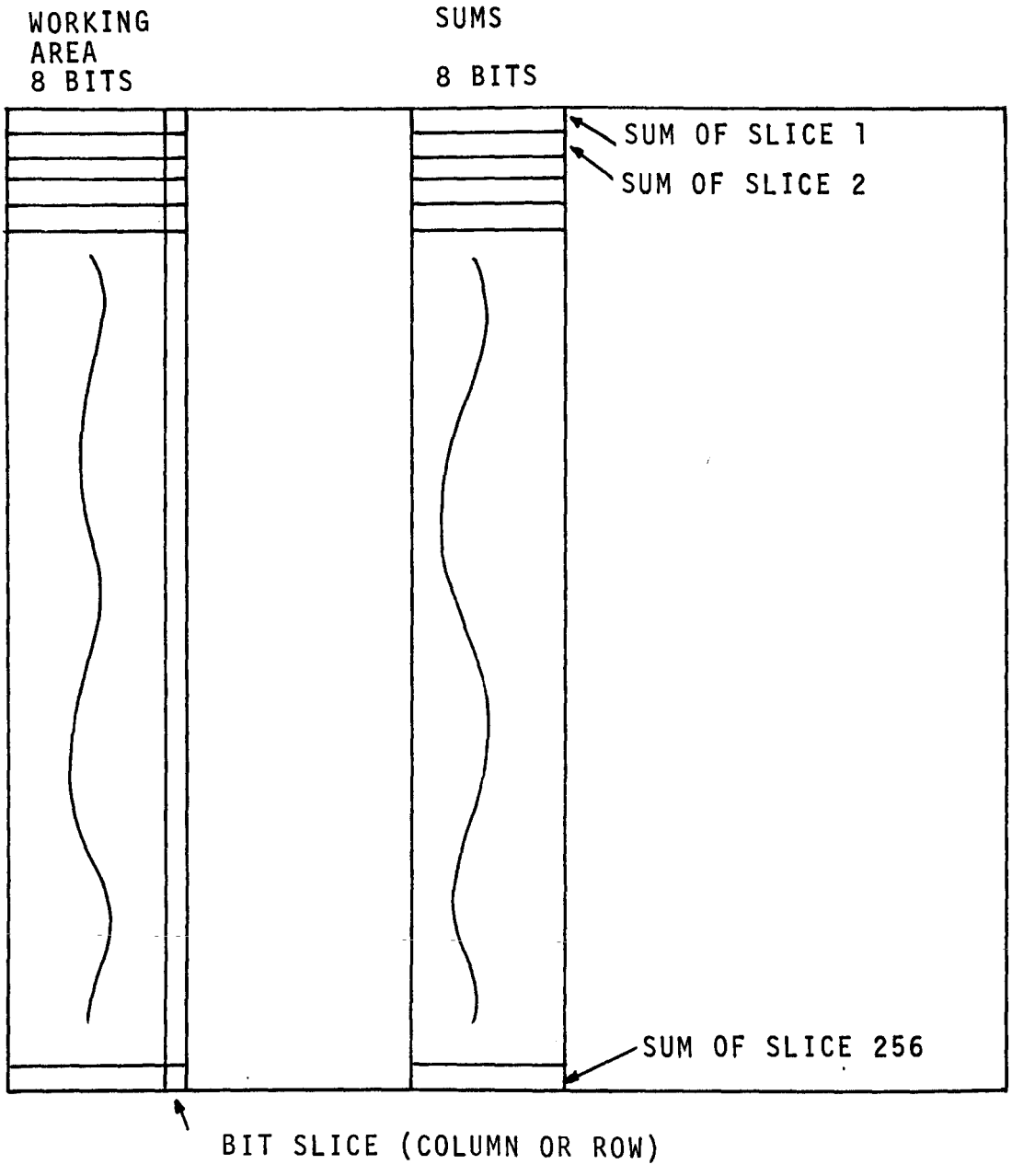


Figure 3-4 - Registration Mark Detection-Number of "ON" Cells

requires 255 additions. However this same result may be obtained with fewer additions using the following array technique.

The "sums" field is divided into 16 groups of 16 values starting with the second word as shown in Figure 3-5. The values in the SUM Field corresponding (by word) to the 16 positions in each of the first 15 groups are 'tree summed' (modulus 16) in the ACCUM Field. This requires four additions. The first word and last 15 words (16^{th} group) are masked out of this operation. The integrals at the points 1, 17, 33, ... 241 are now obtained by performing the following operations:

1. Store the contents of the SUM field Word 1 in ACCUM field Word 1; this gives the integral at the first point.
2. Add this value to the Group 1 summation in the next group and store the result in the last word of Group 1 as shown in Figure 3-5.
3. Repeat this process for the other groups, obtaining one integral value (starting point) for each group.

This process requires 15 additions.

The method used to obtain the other 15 values in each group is illustrated in Figure 3-6. The "starting points" (last word of each Accum Field group) are simultaneously added to their respective next words in the Sums Field with the results stored in the first word position of the next Accum Field group. Then the first word in all Accum Field groups are added to their respective next words in the Sum Field with the results stored in the second position of the Accum Field groups. This process continues until all values of the integral are obtained, which requires 15 adds.

The integrals at each of the 256 points have now been found. These integrals are searched to determine if there exists a sharply rising central portion as shown in Figure 3-3C; this determines if a centerline is present. The time to perform this operation is negligible. When a centerline is located, the X and Y coordinates of the center of the registration mark are determined by:

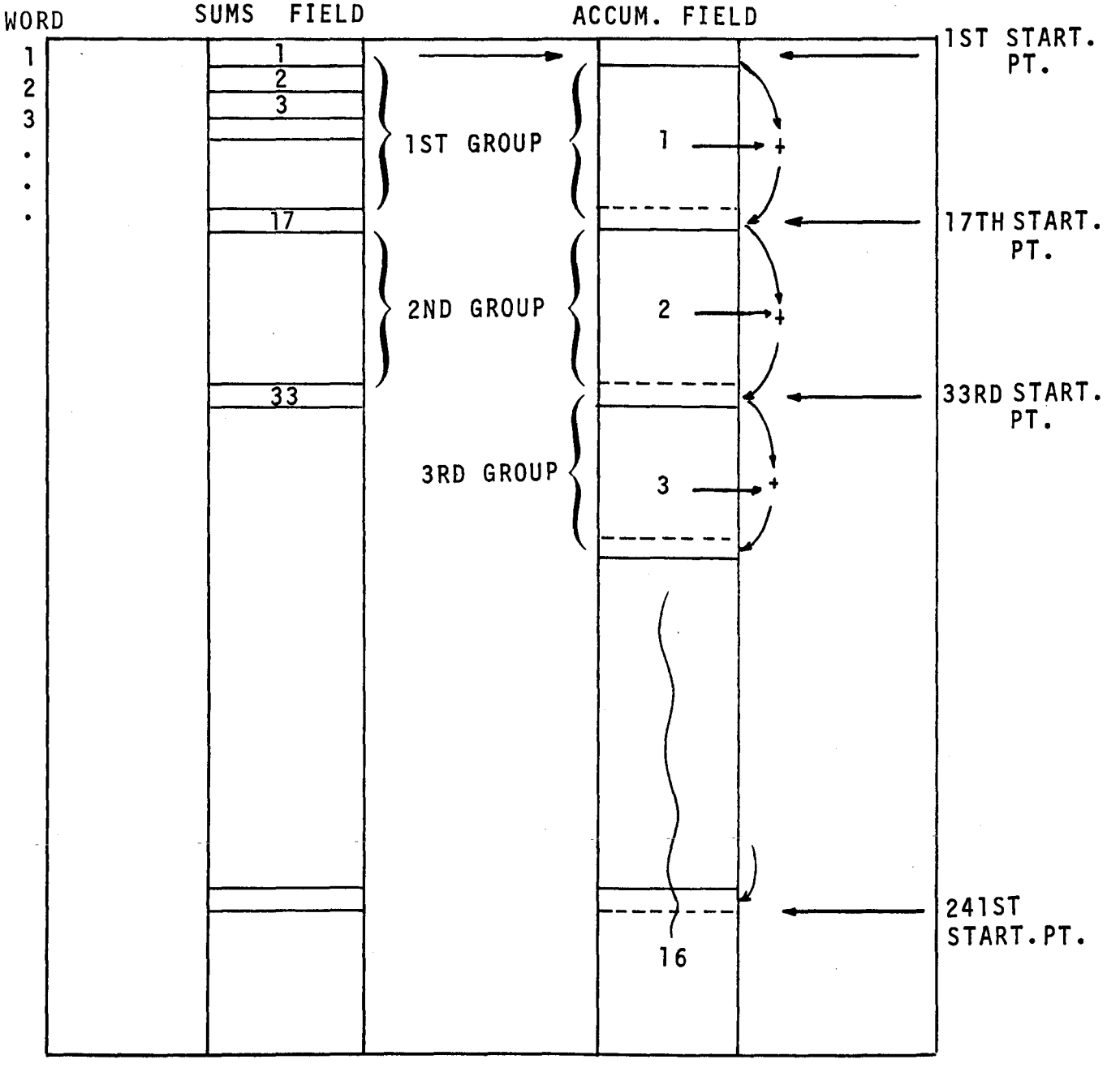


Figure 3-5 - Registration Mark Detection-Integral at Starting Points

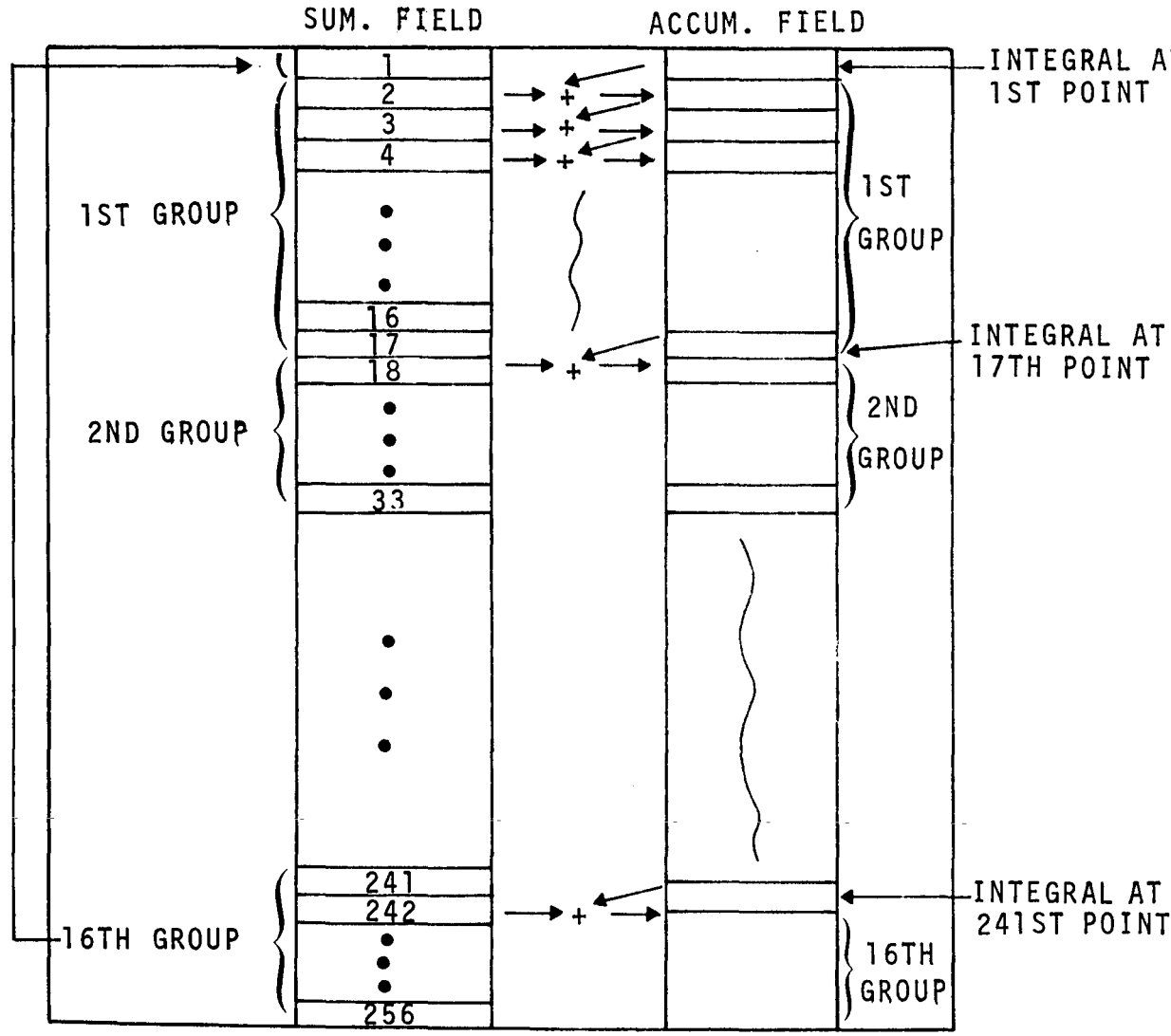


Figure 3-6 - Registration Mark Detection-
Integral at each Successive Point

$$X = \frac{\sum_{\text{all columns}} N(C)C}{\sum_{\text{all columns}} N(C)}; \qquad Y = \frac{\sum_{\text{all rows}} N(R)R}{\sum_{\text{all rows}} N(R)}$$

where,

N(C) represents any of the C columns (bit slices), and
N(R) represents any of the R rows (words).

Each of the above X,Y values may be computed very easily by using the following method.

To compute the denominator, perform a 'tree sum' on the 256 summed values (SUM FIELD); this requires eight additions. To compute the numerator, introduce the values 1, 2, 3, ... 256 to the same field of every word. Multiply the 256 summed values (SUM FIELD) by this field and tree sum the results. The results of the numerator then is divided by the denominator to give the required results.

The timing estimations for the registration mark detection are derived below.

1. Load arrays

$(4.4 \times 10^{-3})(4) = 17.6 \times 10^{-3}$ sec.

where the values are:

4.4 msec to load an array
4 arrays (and registration marks)

2. Sum on cells - both directions

$(12 \times 10^{-6})(8)(250)(2) = 48.0 \times 10^{-3}$ sec.

where the values are:

12 µsec per 8-bit add
8 adds per tree sum
250 tree sums
2 directions (vertical and horizontal)

3. Accumulate sums (integrate)

$(15 \times 10^{-6})(34)(2) = 0.99 \times 10^{-3}$ sec.

where the values are:

15 µsec. per 12-bit add
34 adds per integral
2 directions - (vertical and horizontal)

4. Compute X and Y center points

$$[(15 \times 10^{-6})(8) + 115 \times 10^{-6} + (30 \times 10^{-6})(8) + (115 \times 10^{-6})](2) =$$

$$1.18 \times 10^{-3} \text{ sec.}$$

where the values are:

15 μsec per 12-bit ADD (denominator),
8 ADDS per denominator tree sum,
115 μsec per multiply (numerator),
30 μsec per 20 bit ADD (numerator),
8 ADDS per numerator tree sum,
115 μsec per divide, and
2 directions X and Y.

Note: the load time for constants 1, 2, ... 256 is not included because this operation is done once only and the results are then retained in the array.

5. Compute total time

$$(17.6 + 48.0 + 0.99 + 1.18)(1.25) =$$

169.4 msec ,

where

1.24 is a 25% overhead factor for general housekeeping, etc.

3.2.3 Line Thinning

The object of line thinning (or skeletonization) is to reduce a line to a single-cell thickness along its centerline. The resulting centerline then is used as a reference to produce the appropriate map symbol for the type of data (first class roads, streams, railroads, etc.) represented by the particular overlay.

In general, lines are thinned by removing edge cells until unit thickness is attained. One layer of edge cells is stripped away per pass through the digitized image data. Therefore the number of passes through the data required to perform line thinning is about equal to one-half the cells that make up the line width.

More specifically, the approach used with sequential processors is to test each cell with respect to the status of its eight neighbors. Essentially, a cell will be removed if one of the orthogonal neighbors is zero, and removal of the cell does not destroy continuity of a line. The map image (lines) is usually represented by run-length coded data that gives the location of the "white" to "black" and "black" to "white" transition points along a scan line.

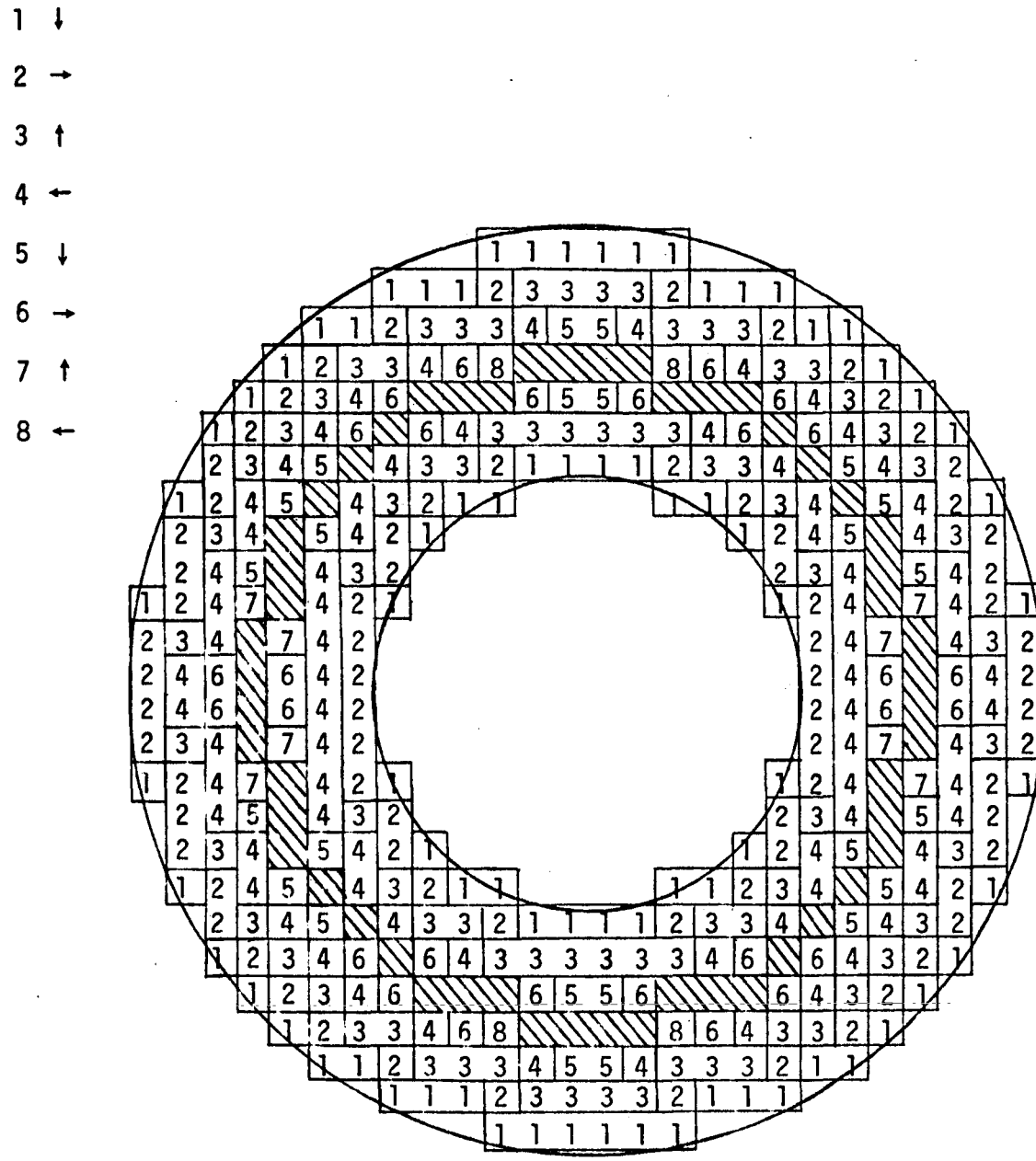
In contrast, the STARAN AAP approach to line thinning utilizes the digitized overlay data stored in the associative arrays as a binary image. For the sample problem referenced in this study, the overlays were scanned on four milli-inch centers. Therefore about one square inch of the overlay can be stored in each 256 bit by 256 word array with four square inches of data accommodated by a STARAN S-1000.

Line thinning within each array is implemented using simple logic operations between each 256-bit slice of data and the data slice on either side of it. An overview of the AAP approach to line thinning is illustrated in Figure 3-7 with a circular line pattern that is about seven cells thick. The cells within the pattern are initially ON or logical 1's. After eight passes of the line-thinning algorithm, the shaded area remains as the thinned line.

The numbers within the cells of the circular pattern represent the array pass that removed them. On the first pass through the array, the algorithm was executed for each 256-bit horizontal data slice (array word) moving through the array from top to bottom. The second pass traversed from left to right, the third from bottom to top, etc., as indicated by the numbers and arrows in the upper left part of Figure 3-7. Moving through the array in all four directions results in faster thinning and helps to preserve line symmetry.

The conditions listed at the bottom of Figure 3-7 represent the basic rules implemented in the line-thinning algorithm. If the

LINE THINNING



REMOVE IF: (ALL REF. OLD DATA)

A. "0" ON ONE SIDE

B. "1" ON OTHER SIDE

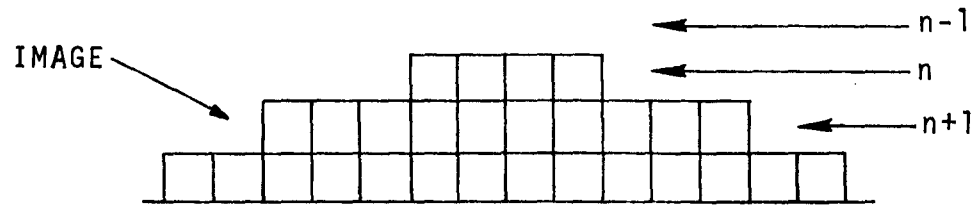
C. >1 UNIT WIDE

Figure 3-7 - Line Thinning Overview using a STARAN AAP

algorithm is being applied to a horizontal slice of array data, then the rules essentially mean that any cell in the data slice will be removed if the cells above and below are logically different and the cell in question has at least one horizontal neighbor. The algorithm operates on data existing in the array at the beginning of the pass.

While these rules do not represent the complete algorithm, they are basic tests and will be used in some examples to more specifically convey the STARAN AAP concepts for line thinning. Figure 3-8 shows part of an image that is similar to the top of the circular pattern given in Figure 3-7. Each enclosed block is a Logical 1 that represents a basic resolution cell of the digitized image. The basic line-thinning algorithm is given also in Figure 3-8 and executed for the 256-bit array Word n. The sequence of operations is as follows:

1. Read Word n and use it as a mask. In other words, only those columns with a Logical 1 in Row n will be involved in the result. As will be shown later, other conditions are factored into the generation of the mask.
2. Logically "exclusive OR" the preceding (n-1) and succeeding (n+1) rows. In general, this operation tests the cells above and below to see if they are different (one "0's" and the other "1's"). If they are different, then the cells in question must be on the edge of the image and can be removed. This procedure is the basic algorithm test.
3. Logically AND the mask with the above result. This step is done to involve the mask, the importance of which will be understood later.
4. Logically "exclusive OR" the word being checked (n) with the above masked test results. This step then produces the new n. For this example, the Logical 1 cells in Word n were on the edge of the image and were removed.



BASIC ALGORITHM

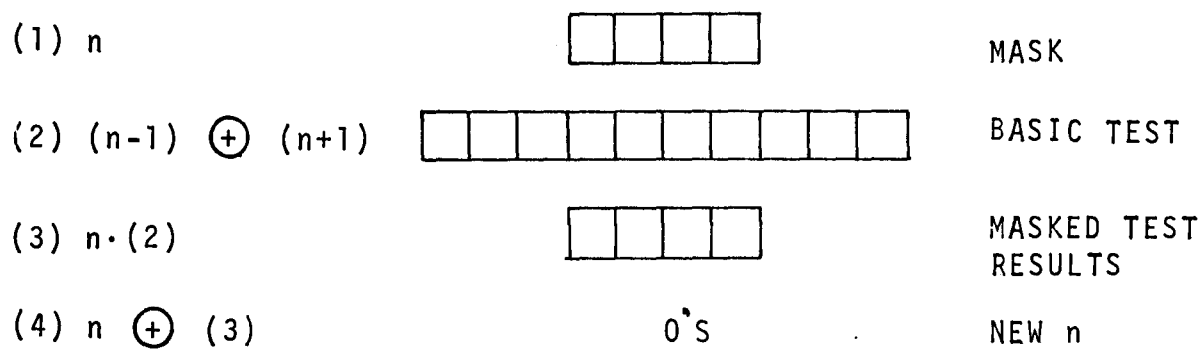


Figure 3-8 - Line Thinning Example 1

As mentioned previously, the algorithm is executed for each 256-bit word (row) of the array. Assuming this pass of the algorithm is progressing from top to bottom of the array, then the next execution of the algorithm is shown in Figure 3-9 where the n of this figure is the n+1 of Figure 3-8. As shown in Figure 3-9, execution of the algorithm in this example results in the retention of the center four cells of Word n since they are not on the edge of the image. In addition, the algorithm is applied to the data that existed in the array at the beginning of the present pass through the array. Since the tests only involve the rows above and below the rows under consideration, the new n is stored in the array at n-3. Therefore, after each pass the image is shifted three

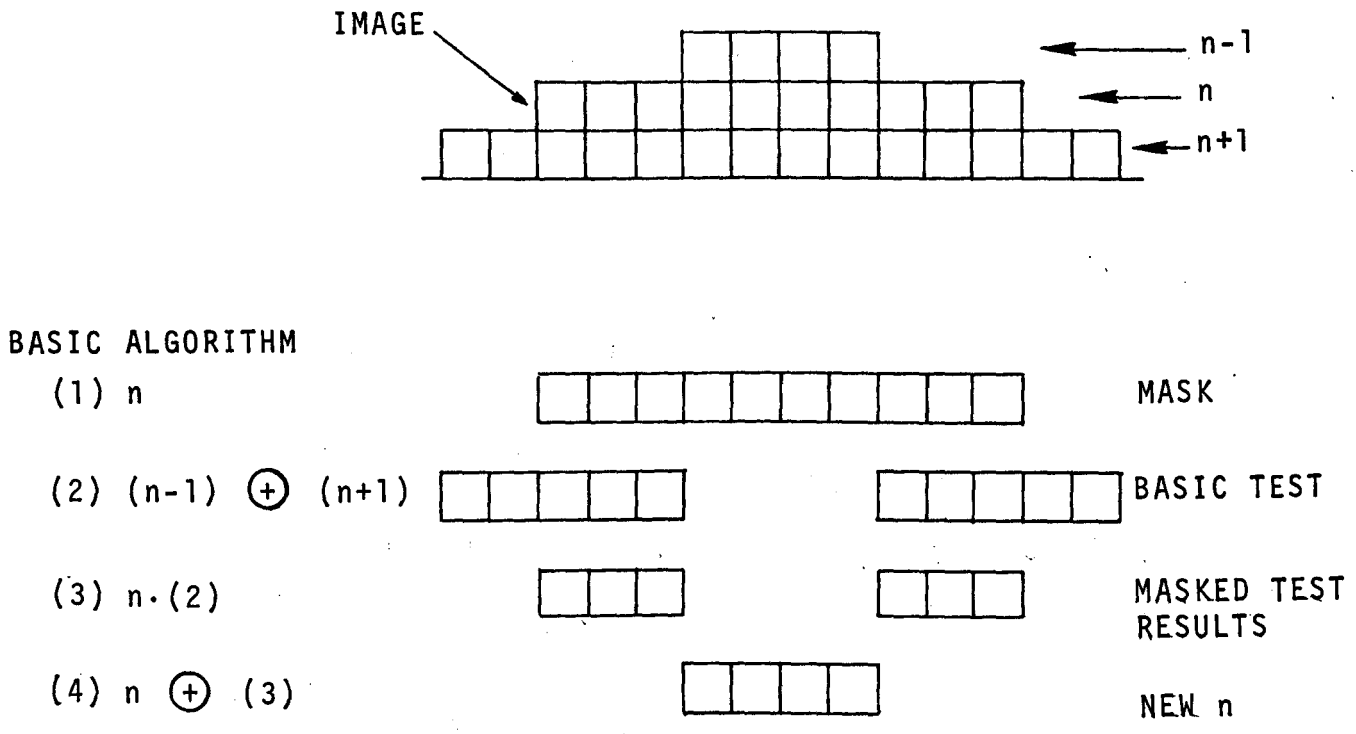
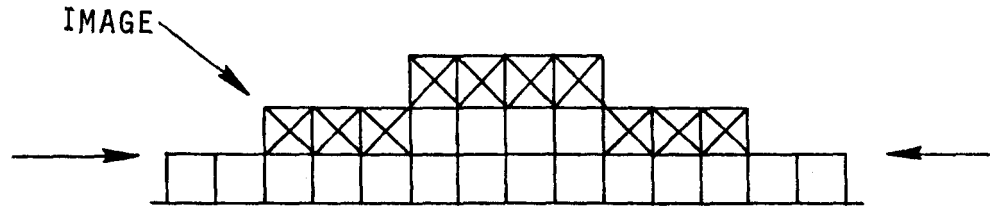


Figure 3-9 - Line Thinning Example 2

places. However, after four passes (one in each direction) the image is in its original position. Figure 3-10 shows the results of performing the basic line-thinning algorithm in one direction through two array words of the image as described in the previous examples.

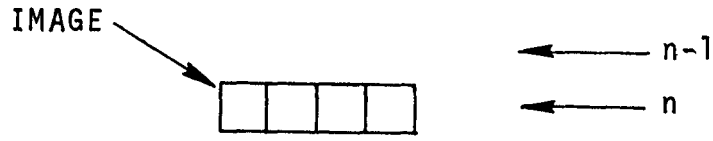
Figure 3-11 shows that the basic algorithm will not eliminate the cells when unit thickness is reached. This condition is required because the lines are thinned in orthogonal directions and parts of the line image may thin faster than other depending on the line direction. Therefore it is necessary to make as many passes of the algorithm through the array as required to thin all of the lines without causing line breaks where the thinning rate is higher.



FIRST TWO ROWS TESTED IN FIRST TOP TO BOTTOM PASS

X - REMOVED CELLS

Figure 3-10 - Line Thinning Results for Examples 1 and 2



BASIC ALGORITHM

(1) n		MASK
(2) (n-1) + (n+1)	0'S	BASIC TEST
(3) n • (2)	0'S	MASKED TEST RESULTS
(4) n ⊕ (3)		NEW n

Figure 3-11 - Unit Thickness Line Thinning Example

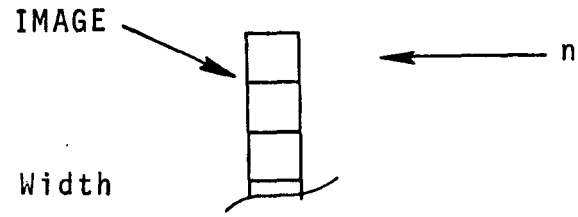
As stated previously, one of the test conditions is that the image be more than one cell wide. This requirement avoids the case where the ends of a unit thickness line would be "nibbled" away. The one cell wide condition is resolved in the mask generation routine illustrated in Figure 3-12. Again assuming that the algorithm is executed on the array words (rows) moving from top to bottom, the sequence of operations is:

1. Read the array word being check (n).
2. Shift the word to the right one bit (cell) position.
3. Logically AND the unshifted and shifted words.
4. Shift the above result left one bit (cell) position.
5. Logically OR the last two results.

For the single cell width image shown in Figure 3-12A, the mask is zero and the image will not be involved in the basic line-thinning algorithm. Therefore the cell being tested will be retained. Figure 3-12B shows that the above routine will, however, permit all cells of an image greater than one cell wide to participate in the line-thinning algorithm operation.

The basic line thinning algorithm discussed in the preceding examples serve to convey, in a relatively uncluttered manner, a general concept of how an associative array processor can be applied to the line-thinning problem. In reality however, additional steps are required to meet all conditions encountered by this task. For example, the basic test described in the example will preserve a line of unit thickness, but not one that is two cells thick. This problem and others are resolved in the latest version of the STARAN AAP line-thinning algorithm, which is shown in Figure 3-13.

Steps 5 through 11 modify the mask generated at step 4 to preserve continuity between two areas touching at their corners. The algorithm will produce thinned lines that are within one cell of the anticipated position when a small radius (less than 0.05 in.) is involved with line thickness greater than four cells. For all other cases, the thinned line seems to be right in the center of the original line. However, until more testing is done, the algorithm at this stage must be considered preliminary.



a. Single Cell Width

MASK ALGORITHM

(1) n 

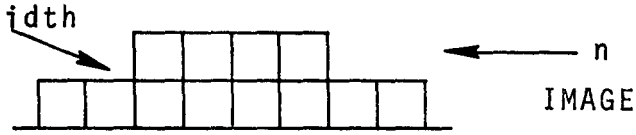
(2) SH n RT 

(3) (1) • (2) 0'S

(4) SH (3) LT 0'S

(5) (3) + (4) 0'S MASK

b. Multiple Cell Width



MASK ALGORITHM

(1) n 

(2) SH n RT 

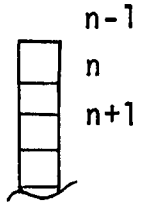
(3) (1) • (2) 

(4) SH (3) LT 

(5) (3) + (4)  MASK

Figure 3-12 - Mask Generation Examples

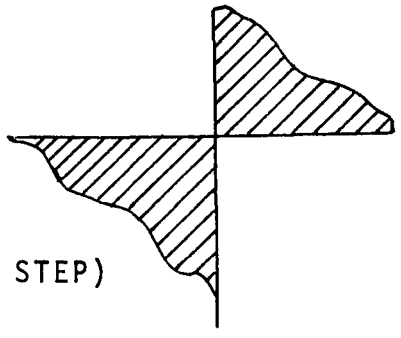
MASK OUT POINTS ALONG n THAT ARE ONE CELL WIDE



(1) SHIFT n RIGHT ONE BIT
(2) n (1)
(3) SHIFT (2) LEFT ONE BIT
(4) (2) + (3)

(USE TO GENERATE MASK IN STEP 12)

PRESERVE CONTINUITY BETWEEN TWO AREAS TOUCHING AT THEIR CORNERS



(5) (n-1) ⊕ (n+1)
(6) (n-1) • (5)
(7) SHIFT (6) RIGHT ONE BIT
(8) (n+1) • (7)
(9) SHIFT (6) LEFT ONE BIT
(10) (n+1) • (9)
(11) (8) ⊕ (10) (USE TO GENERATE MASK IN NEXT STEP)

GENERATE MASK

(12) (3) • (11) (MASK)

TEST

(13) (n-1) ⊕ (n+1) MASKED → A* (SAME AS (5) MASKED)
(14) (13) + B (B CONTAINS STEP 13 RESULTS FROM PRECEEDING n TEST)
(15) (14) • n → n-3
(16) A → B (SAVES STEP 13 RESULTS FOR SUCCEEDING n TEST)

NOTE: A AND B ARE 256 BIT STORAGE LOCATIONS WITHIN THE ARRAY.

Figure 3-13 - Preliminary Version of STARAN AAP Line-Thinning Algorithm

Timing estimates to perform the line-thinning process utilizing a STARAN S-1000 (four array) system are shown in the calculation below. The algorithm execution times were based on the assumption that the array (map) image was 250 x 250 bits. The border surrounding this image was used as a scratch area. The STARAN execution time for the line-thinning algorithm will be independent of overlay density.

The map image is loaded into the arrays for line thinning and will remain in the arrays for skew correction, line break detection, line smoothing, and line symbol generation. At this point the data is transferred to temporary storage before color merging. The array load times therefore are included in the following estimates while the array unload times are included in the line symbol generation timing estimates.

The timing estimate for the line-thinning process is calculated as follows:

1. Data transfer into array

$$(4.4 \times 10^{-3})(12 \times 16)(7) = 5.9 \text{ sec.}$$

where the values are:

4.4 = msec to load an array via common register,
12 x 16 = size of map area in inches (1 sq. in. per array), and
7 = overlays.

2. Algorithm execution time

$$\frac{(20 \times 0.26 \times 10^{-6})(250)(8)(12 \times 16)(7)}{4} = 3.5 \text{ sec.}$$

20 = operations per slice,
0.26 = μsec per operation (average),
250 = slices per array pass,
8 = array passes per thin,
12 x 16 = is size of map area in inches,
7 = overlays, and
4 = associative arrays (1 sq. in. of map area per array).

3. Total line thinning time

$$3.5 + 5.9 = 9.4 \text{ sec.}$$

3.2.4 Skew Correction

The position of the registration marks determined previously are used to compute the amount of skew or misalignment between overlays. The digitized image must be rotated and translated for proper alignment between all overlays. Each overlay is divided into small areas that fit into the STARAN arrays as indicated in Figure 3-14. The overlays were scanned on 4 milli-inch centers and therefore each 256-bit by 256-word array can accommodate approximately one square inch of the overlay image.

Figure 3-14 shows the array image areas divided into the sizes that would exist after skew correction. Actually, the data transferred into the array(s) prior to skew correction would be aligned with the orthogonal axes (original scan direction) and be large enough to contain the skewed areas pictured in Figure 3-14. The excess data then would be ignored when the corrected and processed image is transferred from the array. The array image area initially transferred into the arrays therefore will overlap their respective adjacent area.

In Figure 3-14, the corrected position of the upper left hand corner of each array image area is computed and retained by a housekeeping routine. The rotation part of the skew correction is accomplished in the arrays with simple orthogonal data shifts that are described below. The translation is accomplished by using the previously calculated upper lefthand corner positions of the rotated image to assign appropriate memory locations to the image data as it is transferred from the arrays to temporary storage just prior to color merging.

The most time-consuming part of the AAP approach to skew correction was assumed to be the rotation task. The translation task was considered to be negligible with respect to the array loading and unloading time that is taken into account in the line thinning and line symbol generation functions. Therefore the rotation task and its associated timing estimate is described below.

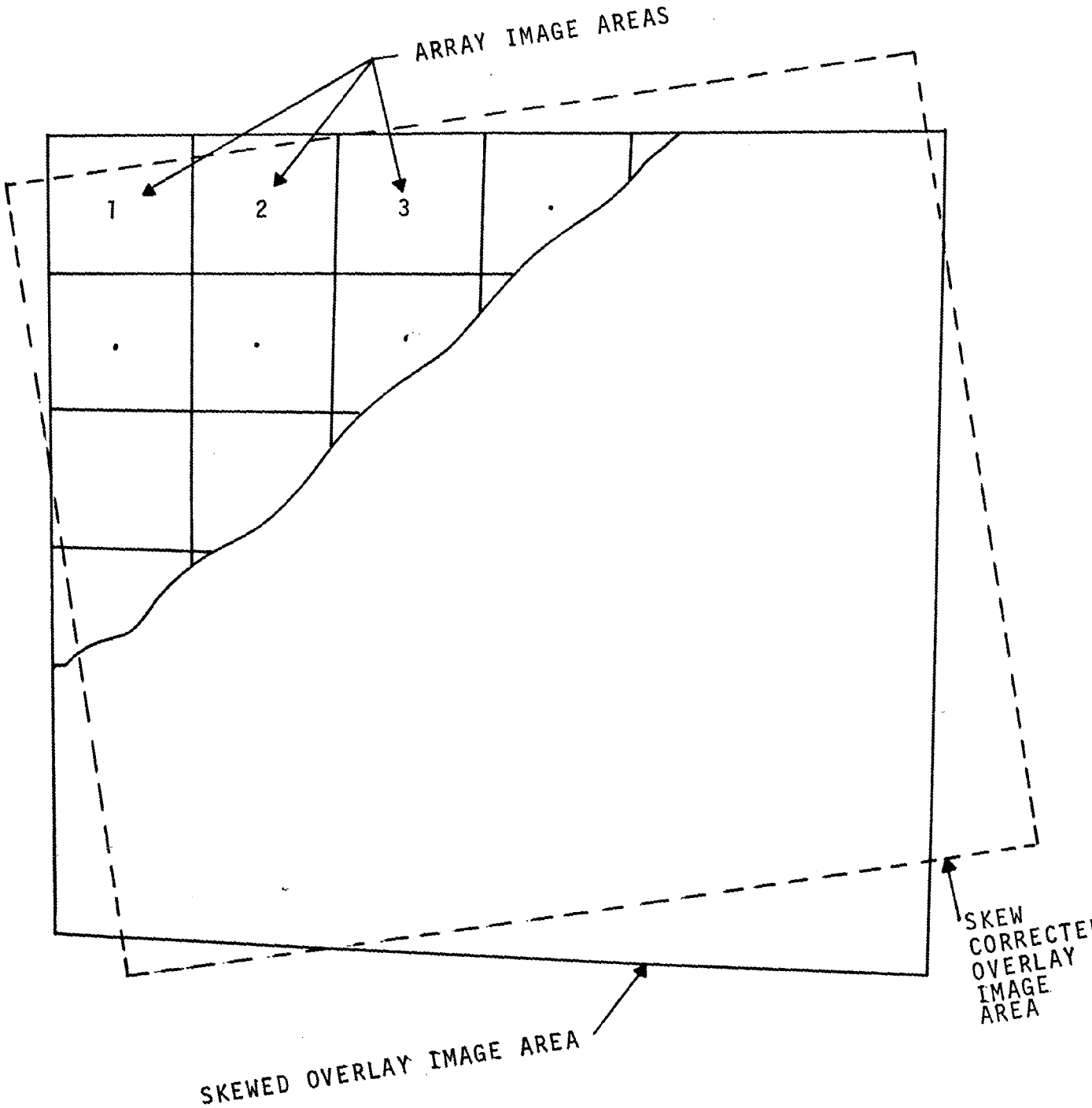


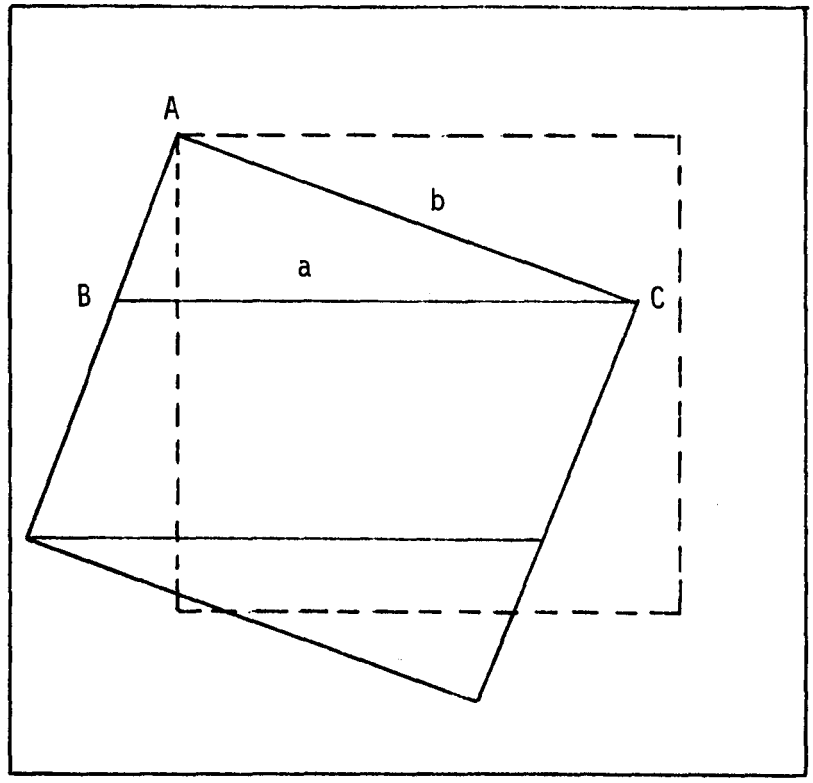
Figure 3-14 - Skewed Overlay

With the upper lefthand corner used as a positional reference, the rotation is performed about that point.

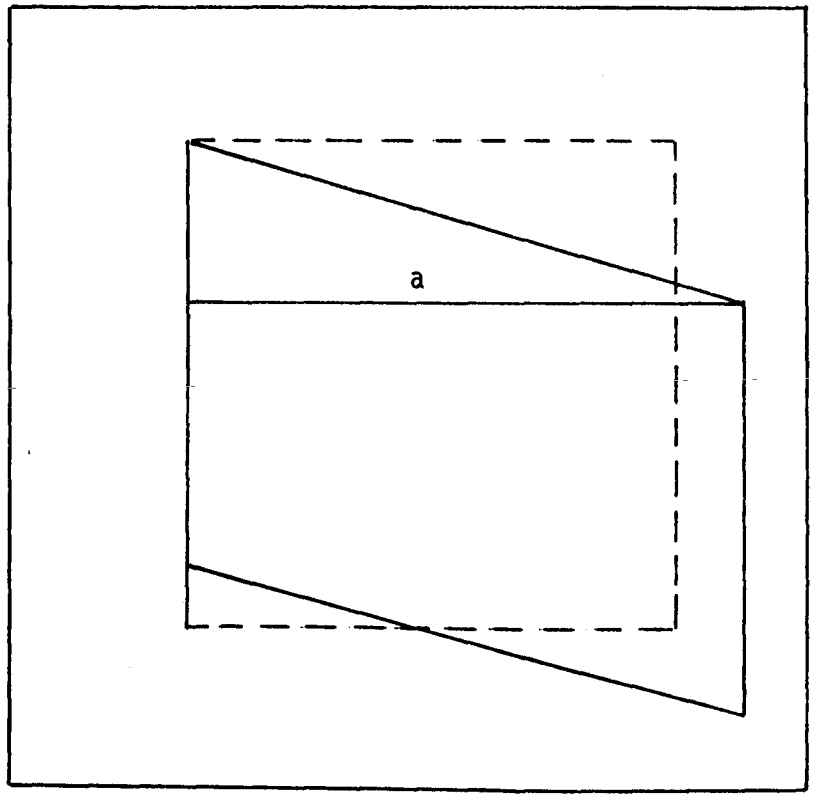
1. Step 1 - Shift the array words horizontally as shown in Figure 3-15.

   Each array word or group of words are shifted by amounts determined by the amount of rotation of the overlay. The amount of shift is the same for all array loads of a particular overlay and is calculated prior to these array operations. Notice that the edge of the shifted image is wider than the corrected image because the hypoteneuse of triangle ABC is larger than AC, which is equal to the top dashed line.
2. Step 2 - Shift bit columns vertically as shown in Figure 3-16. Each bit column or group of columns are up-shifted by amounts predetermined from the amount of rotation of the overlay.
3. Step 3 - Compress the image area. The image area compressed in the horizontal direction (as indicated by Figure 3-17) by removing a bit slice at appropriately spaced intervals and moving all bit columns to the left to close the gaps. The bit slice (column) will be tested to ensure that the slice does not contain a vertical time that would cause an uncorrectable line break. This test could consist of a tree sum of the slice and a "greater than" compare of the result with some specified upper-bound value.
4. Step 4 - Expand the image area. The image area is expanded in the vertical direction (as indicated by Figure 3-17) by moving all words downward to create a single word gap at appropriately spaced intervals. Line breaks will result, but these will be only one cell wide and easily corrected utilizing the line break detection and correction routine.

   The timing estimations for the skew correction routine are derived below:



SKEWED
ARRAY
IMAGE



HORIZONTALLY
SHIFTED
ARRAY IMAGE

Figure 3-15 - Skew Correction - Horizontal Shifting

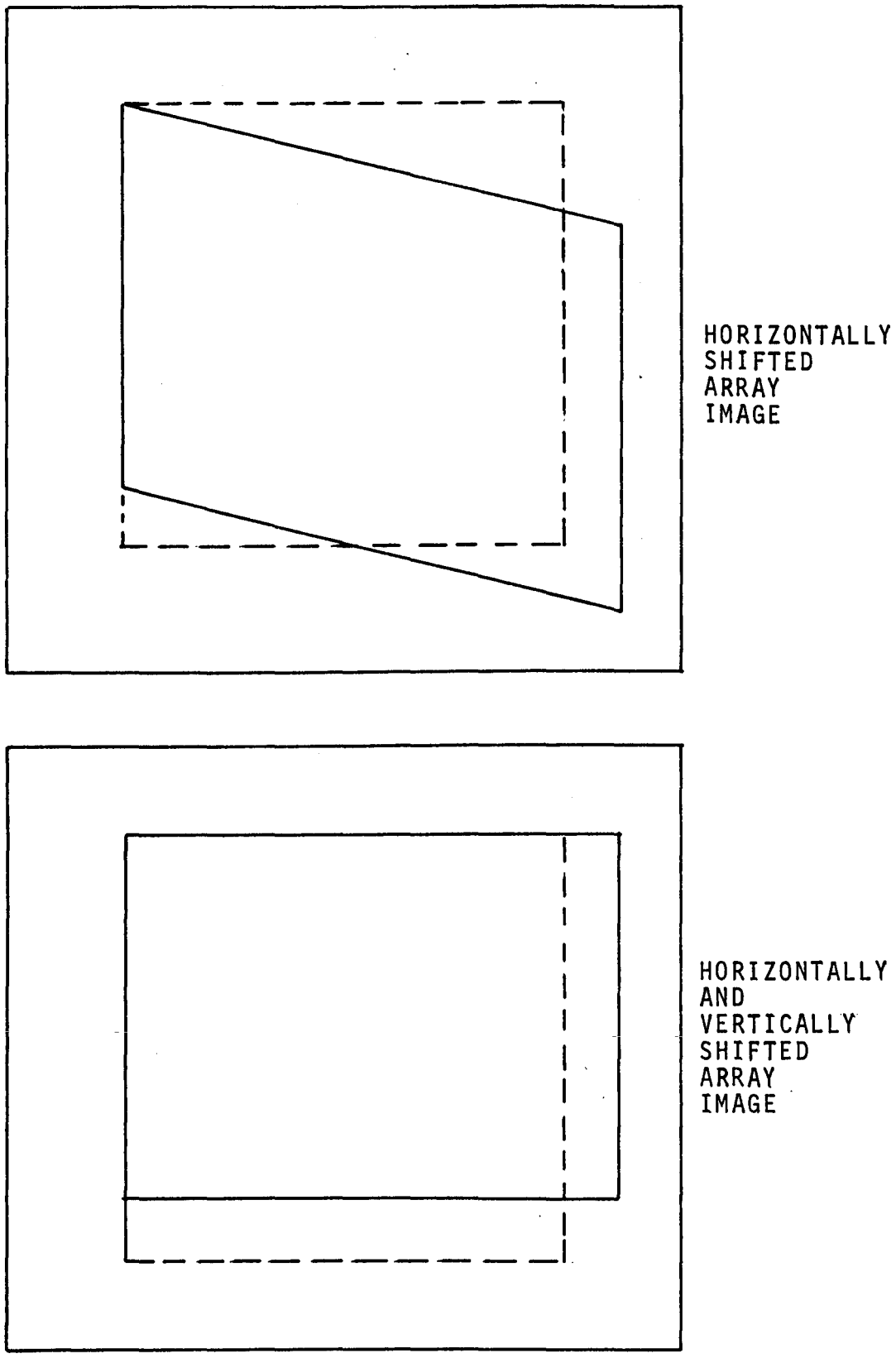
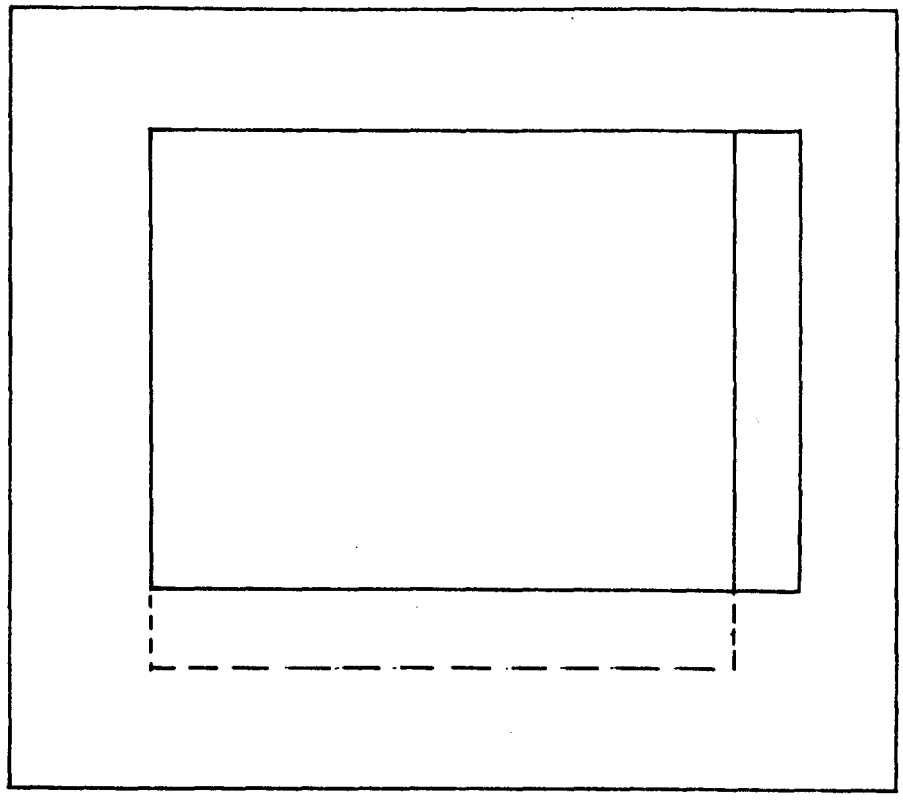


Figure 3-16 - Skew Correction - Vertical Shifting



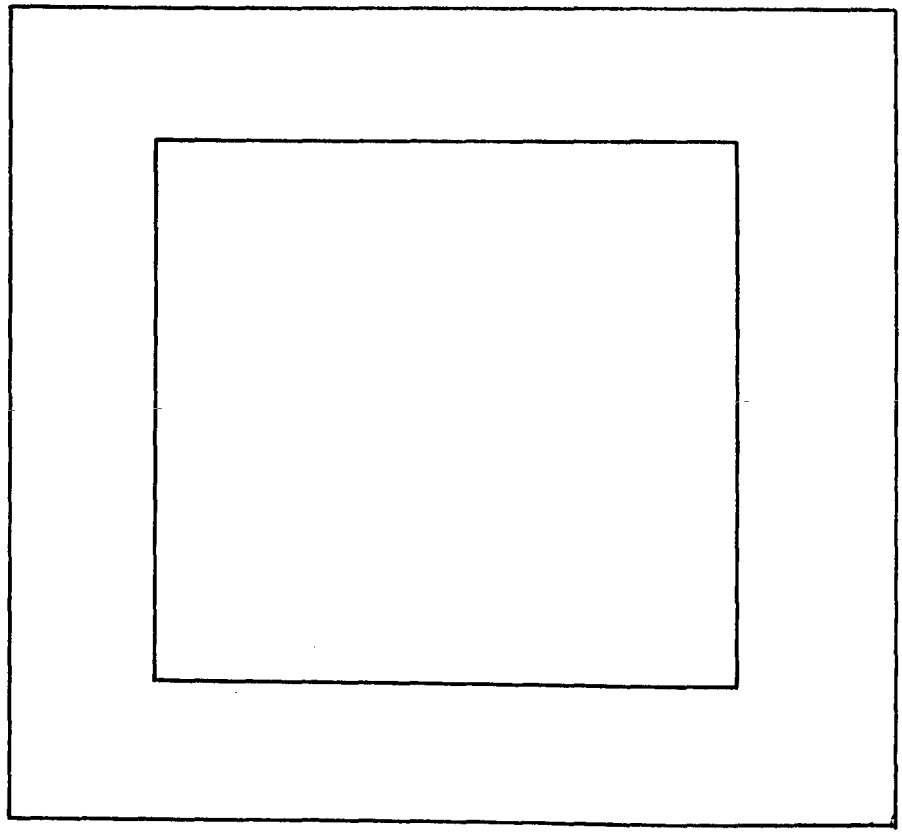


Figure 3-17 - Skew Correction - Compression and Expansion

1. Shifting

$$\frac{(0.69 \times 10^{-6})(250)(2)(12 \times 16)(8)}{4} = 0.13 \text{ sec},$$

where,

0.69 = μsec per shift operation,
250 = shift operations each direction,
2 = directions (horizontal and vertical),
12 x 16 = map area in inches,
8 = overlays, and
4 = associative arrays.

2. Reposition

$$\frac{(0.43 \times 10^{-6})(250)(2)(12 \times 16)(8)}{4} = 0.08 \text{ sec},$$

where,

0.43 = μsec per reposition operation.

3. Total skew correction time

$$(0.13 + 0.08)(1.25) = 0.26 \text{ sec}$$

where,

1.25 = 25-percent increase factor to cover various housekeeping operations.

The diagrams associated with the description of this technique show an exaggerated amount of rotation. In practice there would exist only a small rotation; this implies that there would be very few lines added or deleted within each group of array words that are shifted.

This skew correction routine is a good application of the multi-dimensional access capability of the STARAN associative array processor. This multidimensional access allows the programmer to work with word or bit slices.

3.2.5 Line Symbol Generation

In general, symbol generation is the process required to generate the proper map symbology from line thinned overlay data and decoded overlay characters. A part of symbol generation is line symbol generation that is required for first class roads, second class

roads, railroads, streams, etc. The main thrust of this study with regard to line symbol generation was directed toward single and double line symbology. Intermittent streams and railroads were considered to the point where some basic approaches were derived, but not developed. This discussion presents a detailed description of the STARAN AAP approach to single and double line symbol generation and their respective timing estimates.

The AAP approach to line symbol generation is fundamentally simple. The basic problem encountered with an AAP approach, which can only operate orthogonally on array data, was to produce lines of specified thickness regardless of the line's direction. The STARAN algorithm simply requires each word of the array containing line thinned data to be logically OR'ed back into the array in a circular pattern. Each logical 1 encountered will then generate a circular image. All 1's that make up the thinned line therefore will form a solid line whose thickness is equal to the d meter of the circular pattern as shown in Figure 3-18. With this approach the line thickness will be constant regardless of line direction.

In reality, the thickened image must be built behind the thinned line at a distance at least equal to one-half of the line symbol thickness. This step is required to preserve those parts of the thinned line yet to be processed. This process is described in greater detail by using the thinned line shown in Figure 3-19. Each square represents an ON cell (logical 1) in an array. The single line symbol generation routine will operate on each array word moving through the array from top to bottom. When word n is reached, a "circular" pattern of logical 1's will result as shown in Figure 3-20. For this example a thickness of nine cells was selected. The dots in Figure 3-20 indicate the locations of the thinned line ON cells.

Since it is necessary to contend with finite resolution, the circular pattern is an approximation. Also it was necessary to add some additional cells inside the synthetic circle to avoid holes

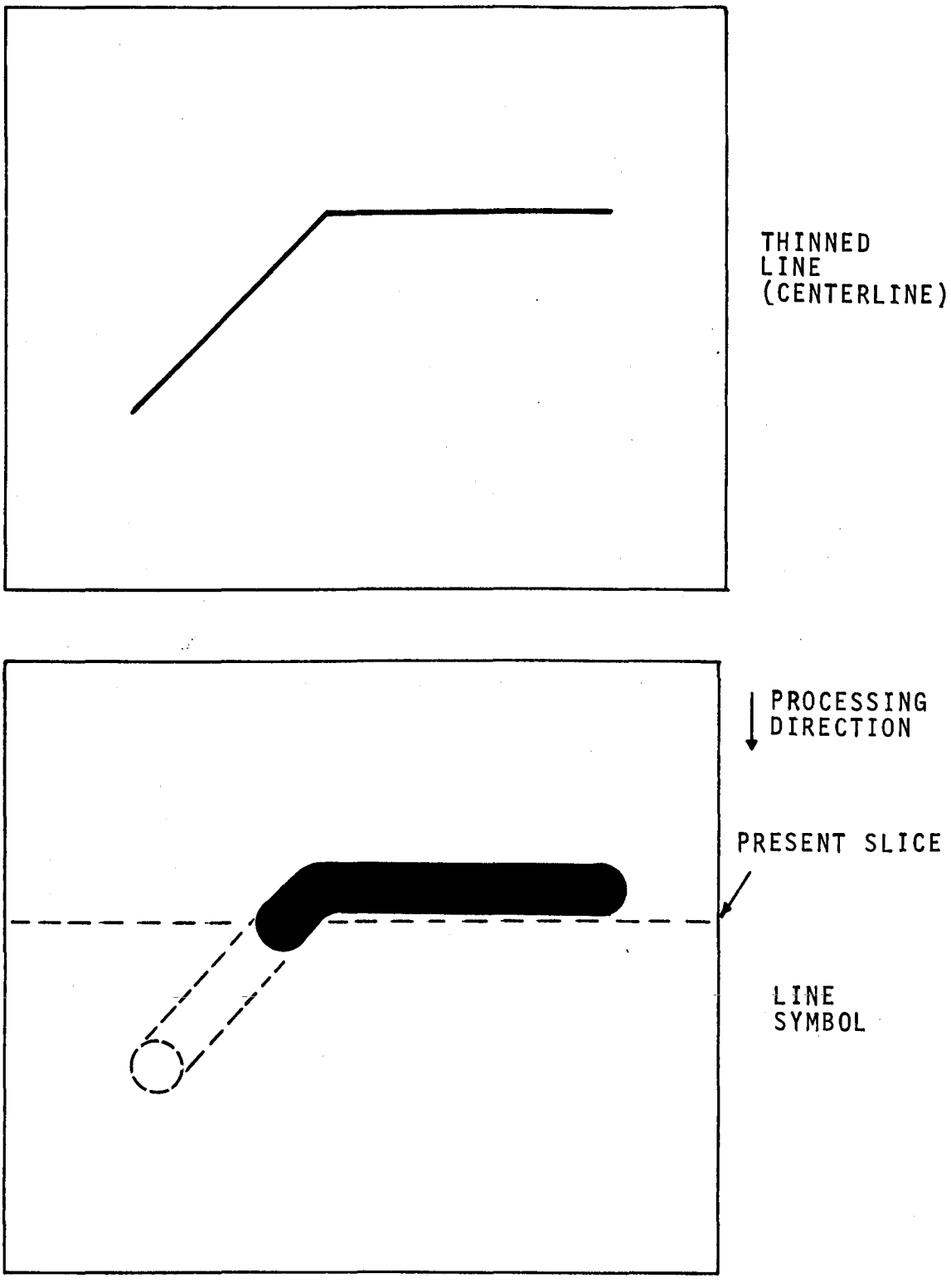


Figure 3-18 - Line Symbol Generation Overview

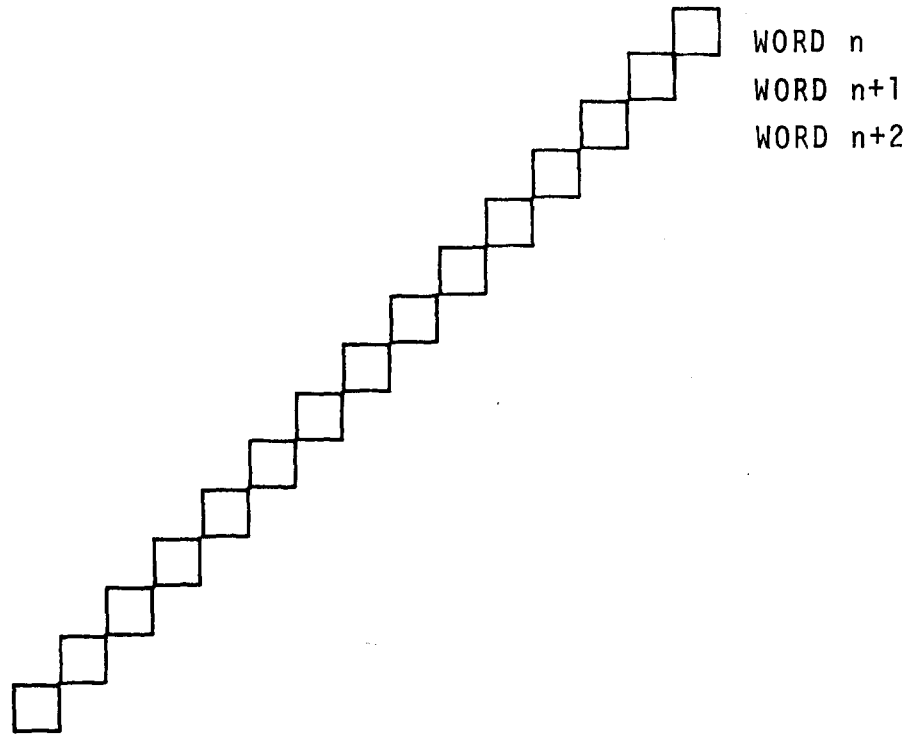


Figure 3-19 - Thinned Line

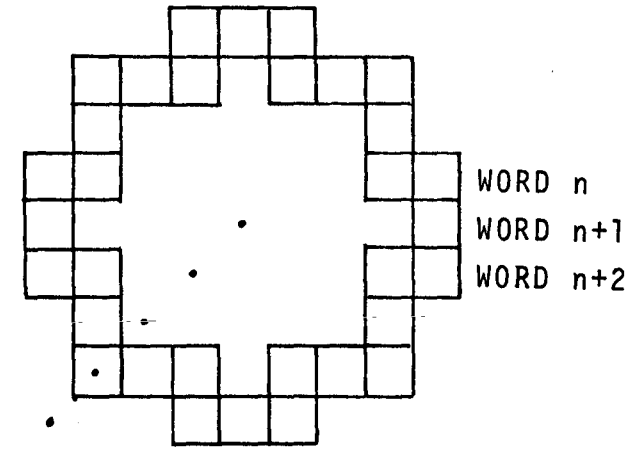


Figure 3-20 - Detailed Pattern of Symbol Generation Algorithm for 9-Cell Thickness

in the line symbol for certain line directions. From this figure it can be seen that each array word is OR'ed back into the array 32 times to generate the circular pattern.

Figure 3-21 and 3-22 show the results after processing words (n+1) and (n+2) respectively. Figure 3-23 shows the line symbol outline that would be produced after the array was processed. The time required to generate the line symbols is independent of the overlay's line density or complexity.

A double line can be generated simply with two single line generation processes. First, the approach described above must be used to produce a line that has a thickness equal to the distance between the outer edges of the double lines. Next, a second single line must be generated that has a thickness equal to the space between the double lines. The second line is generated on top of the first line and instead of ORing the circular pattern in the array for the second line, the pattern is "exclusive ORed". This action causes the logical 1 bits that represent the first line to be zeroed where the second line is overlayed to create the double line. This approach results in constant line thickness and spacing regardless of line direction. Since the thinned line is used twice, it is necessary to use one half of the array to store the thinned line and the other half to build the double-line image.

Symbol generation for railroads and intermittent streams is a more complex problem because both require operations at fixed linear distances along irregular lines (crossbars for railroads and dashed lines for intermittent streams). Two approaches were briefly considered, but sufficient time was not available to develop these ideas. However, timing estimates were derived for these tasks assuming that each would require 10 times the processing complexity of a single line.

Timing estimates for the line symbol generation operations are derived below for a STARAN S-1000. These times include array unloading time. The arrays initially were loaded for line thinning and the array load times are included in those estimates.

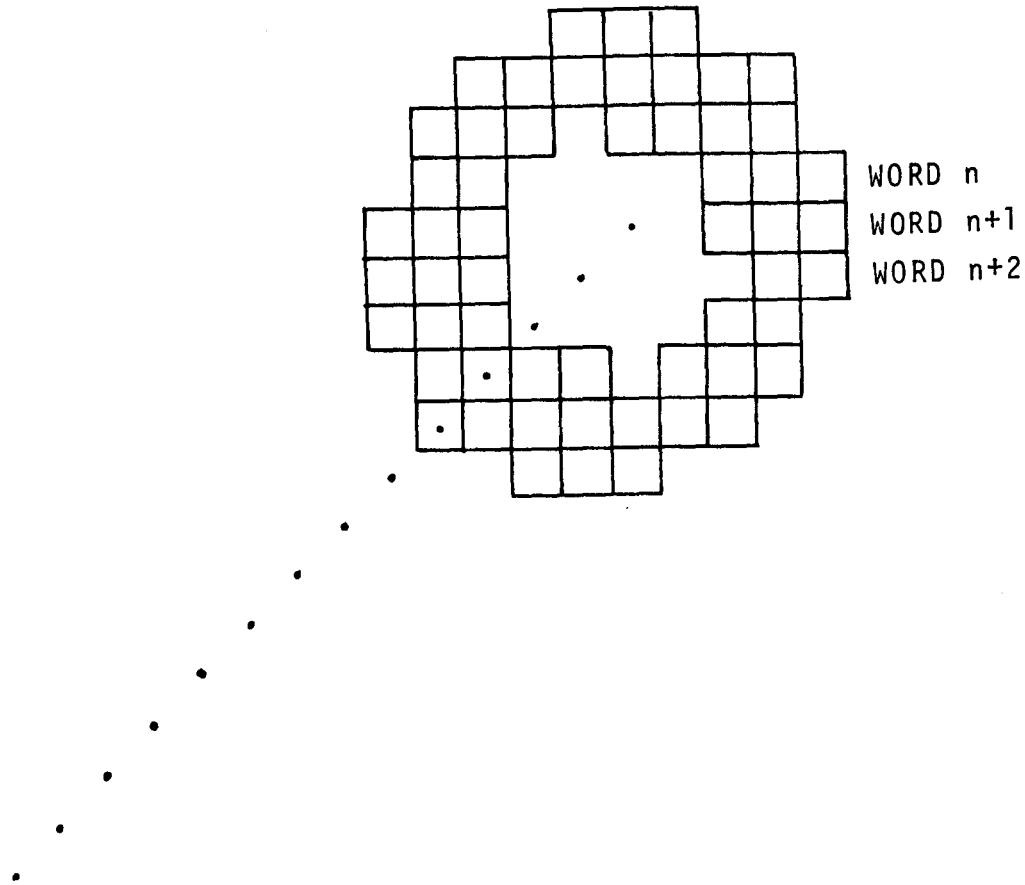


Figure 3-21 - Line Symbol for Two Cells

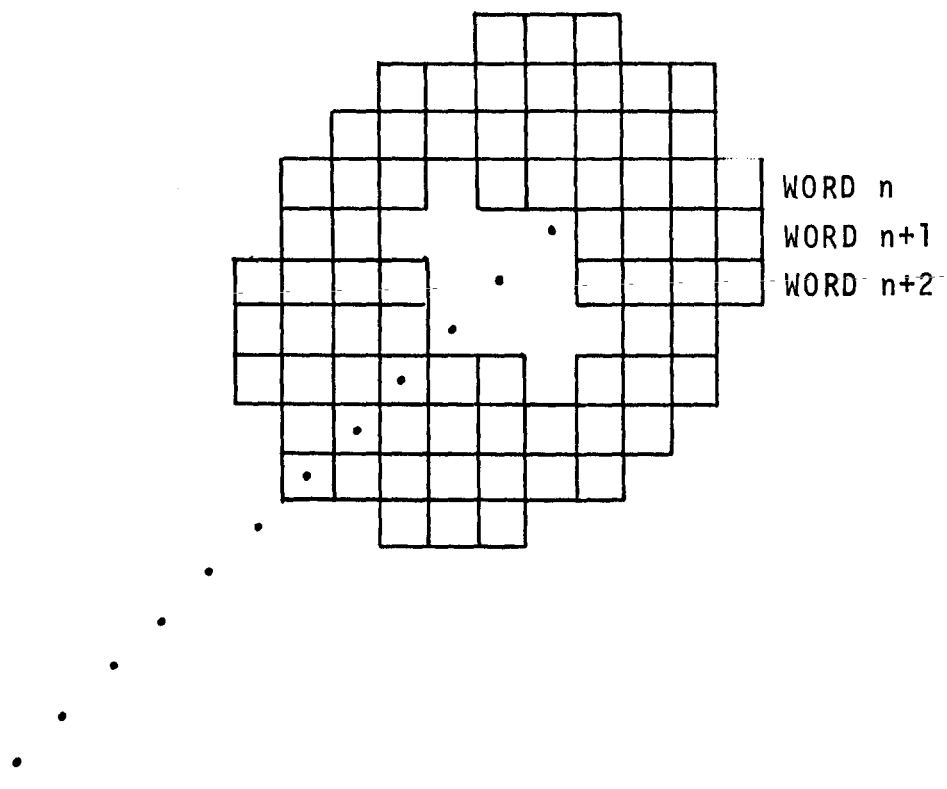


Figure 3-22 - Line Symbol for 3 Cells

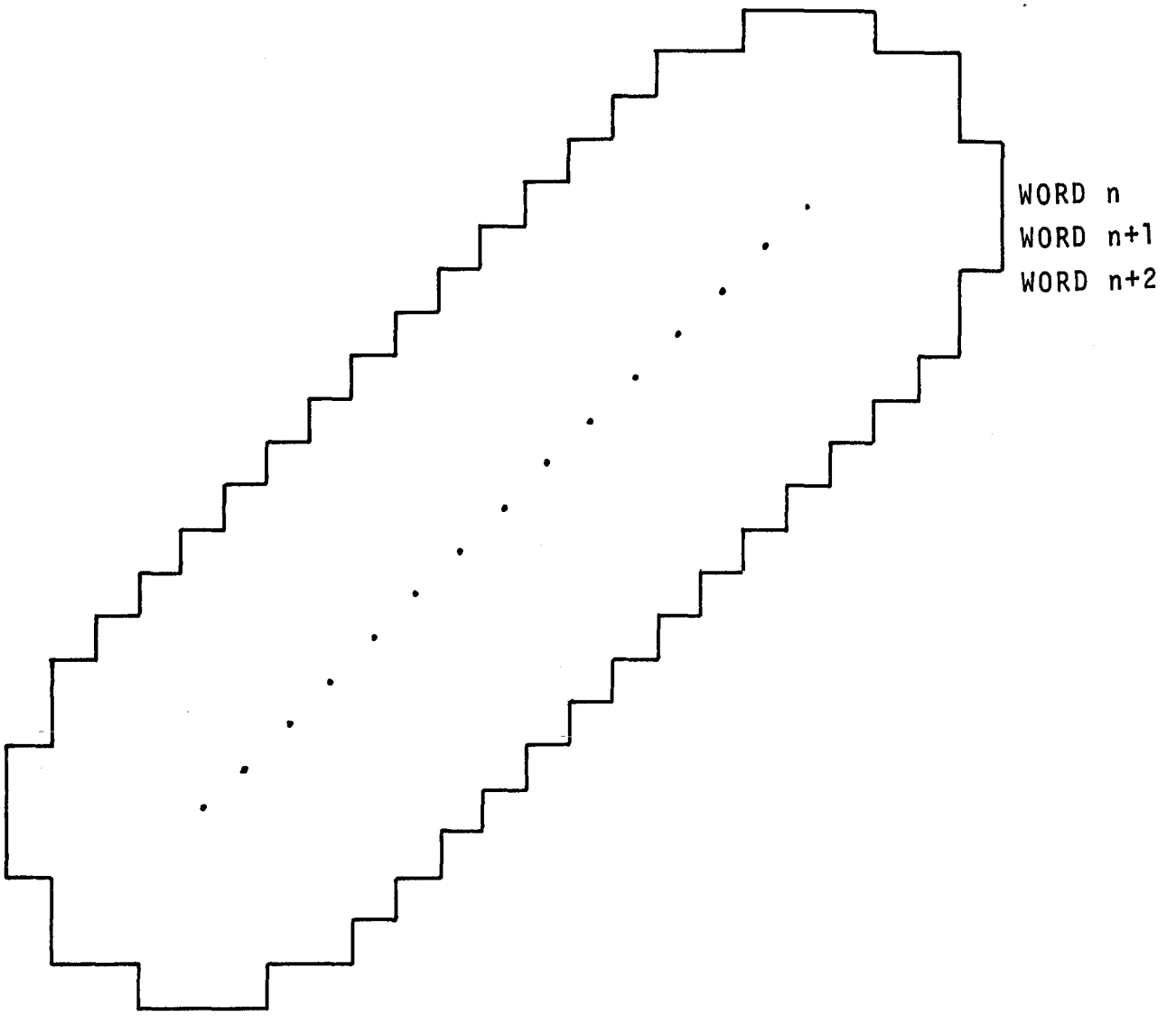


Figure 3-23 - Line Symbol for 15-cell
45-Deg Line

The timing estimate for the line symbol generation is calculated as follows:

1. Single line

$$\frac{(75 \times 0.019 \times 10^{-6})(250)(12 \times 16)(6)}{4} = 1.0 \text{ sec.}$$

where,

75 = operations per slice,
0.19 = μsec per operation (average),
250 = slices per array pass,
12 x 16 = map area in inches,
6 = overlays, and
4 = associative arrays (1 sq in. per array).

2. Double line (Assume 4 x single line)

$$\frac{4 \times 1.0}{6} \times 2 = 1.30 \text{ sec.}$$

1 = overlay
4 = associative arrays (1/2 sq in. per array)

3. Intermittent streams and railroads

Guess: 10 x single line
(10)(0.17)(2) = 3.4 sec.
1 overlay

4. Data Transfer out of array

$(4.4 \times 10^{-3})(12 \times 16)(7) = 5.9$ sec.

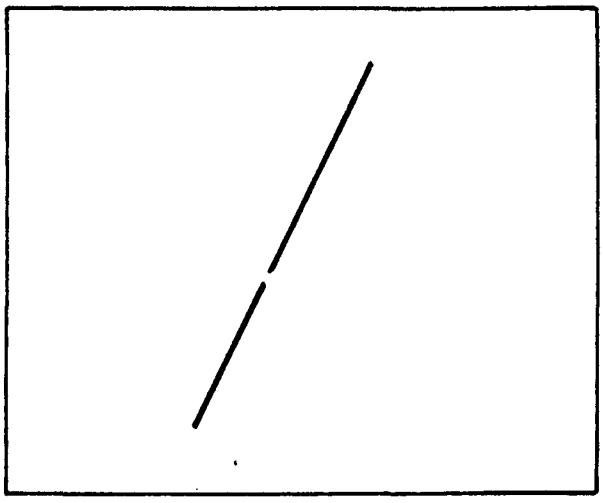
4.4 = msec to unload an array via common register,
12 x 16 = max area in inches (1 sq in. per array), and
7 = overlays.

5. Total Line Generation Time

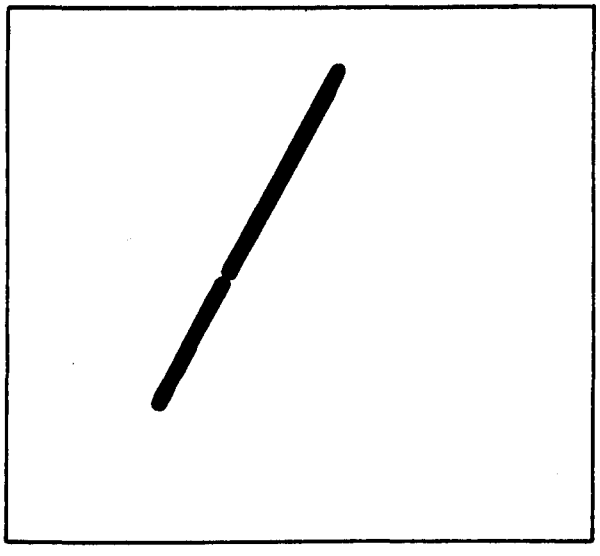
1.0 + 1.3 + 3.4 + 5.9 = 11.6 sec.

3.2.6 Line Break Detection and Correction

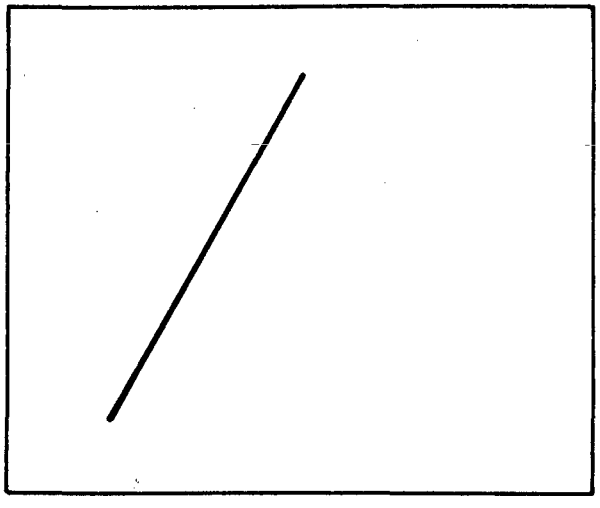
Line breaks may be caused by operations such as the repositioning (expansion or contraction) that takes place during the skew correction process. This occurrence can be rectified by employing the line-thickening process (line symbol generation) to solidify and join the line sections, and then rethinning (line thinning algorithm) that line to a single cell thickness. This approach is illustrated in Figure 3-24.



BROKEN
LINE



SLIGHTLY
THICKENED
LINE



THINNED
LINE

Figure 3-24 - Line Break Detection and Correction

3.2.7 Line Smoothing

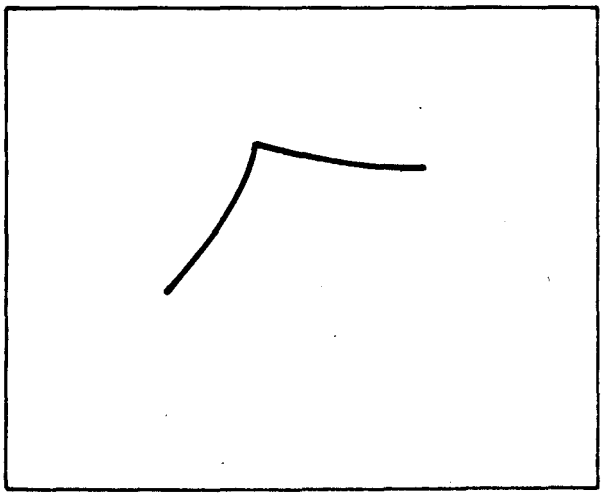
The line smoothing operation can be performed by employing the same thickening-thinning process as discussed above in the line break detection and correction and shown in Figure 3-25. Even though the line thickening required for line smoothing would normally require more thickness than that needed for line break detection, the two raster processing tasks could be executed in one operation. The timing was estimated for this operation by adding together the execution times (less array load/unload times) for the line thinning and single line symbol generation routines.

3.2.8 Line Separation

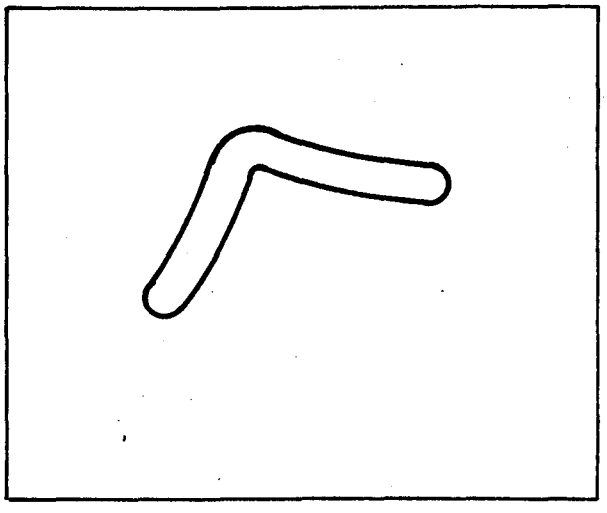
A detailed analysis and timing estimation for the line and character separation technique derived for the STARAN AAP are given below. Each character grouping is preceded by a solid triangle symbol and contained between bracket symbols as shown in Figure 3-26. The size of the solid triangle and all characters are expected to be within the bounds of a 32 x 32 matrix. Compared to any other images, the solid triangle has many more ON elements within the 32 x 32 matrix. The fact that the triangle has more ON elements than other symbols is used in the detection process. The general approach is to search for a solid triangle within every possible 32 x 32 bit window location in the array. This is done by summing the total ON elements within these windows and testing the results to determine if the sums are greater than 450; a number slightly less than the maximum number of ON cells anticipated for a solid triangle.

The detection process can be performed by efficiently using STARAN's powerful between-word communication and AAP capabilities. The sequence of operations required to implement the line separation procedure is given below:

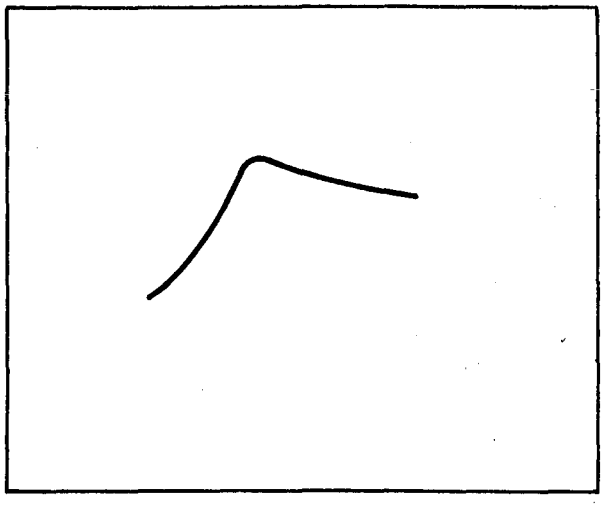
1. Store the overlay image in the arrays.
2. Perform tree sum modulus 32 on the left-most array bit slice giving 8 results.
3. Referring to Figure 3-27, store these 8 results in Field F1 of another array at word locations 1, 32, 64, ... 224.



THINNED
LINE



THICKENED
LINE



SMOOTHED
LINE
(ABOVE THINNED)

Figure 3-25 - Line Smoothing

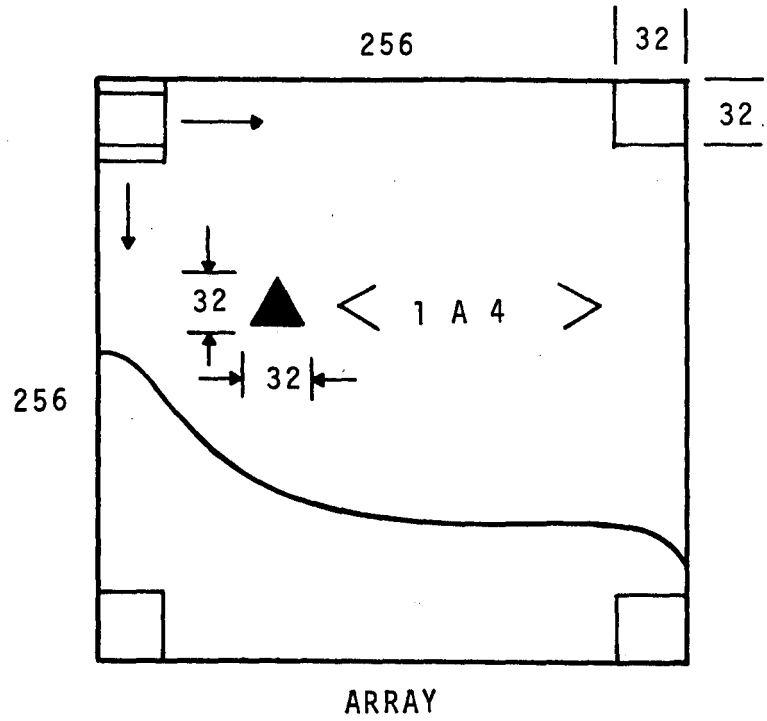


Figure 3-26 - Line Separation

4. Move bit slice up one bit and repeat Steps 2 and 3 and store the results in F1 of array words 2, 33, 65, ..., 225.
5. Repeat Steps 2, 3, and 4 up to 32 times. This results in 32 x 8 = 256 values stored in F1 as shown in Figure 3-27. The first 224 of these 256 results represent a window slit 32 bits deep by one bit wide moving down the bit slice from top to bottom.
6. Repeat Steps 2 through 5 for the other 31 bit slices of the first 32-bit columns. Store the results in Fields F2, F3, ..., F32.
7. Perform 31 field-to-field additions to sum the values in Fields F1, F2, ...F32. Store the results in the ACCUM-FIELD of Figure 3-27. The first 224 results of this accumulation process correspond to the 224 possible 32 x 32 bit windows in the first 32-bit field of the array image.

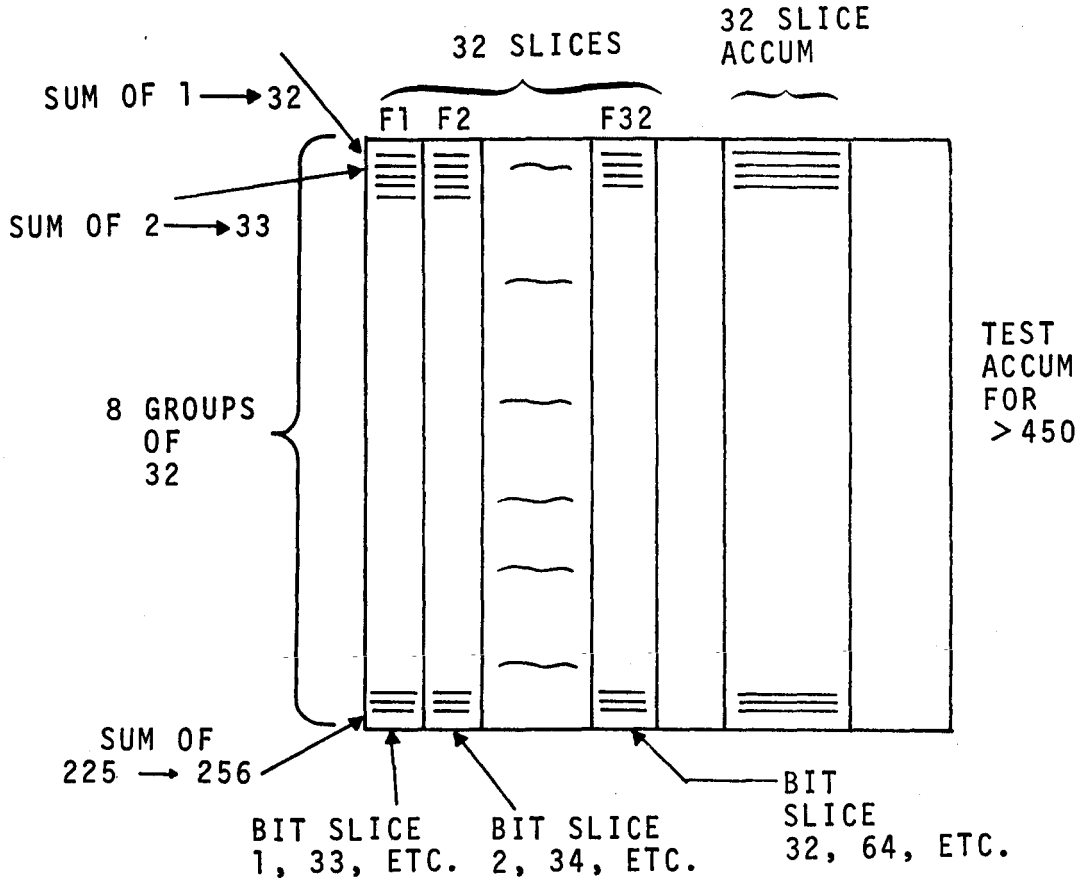


Figure 3-27 - Array Storage for Bit Slices of Window

8. Search the 224 results in the accumulation field for sums greater than 450. These locations reflect the positions in the arrays of the triangles and then are stored in bulk core memory. This step effectively utilizes STARAN's associative search capability.

   Since each triangle will probably result in multiple responders to the "greater than 450" searches, it may be necessary to perform a maximum search on each group to obtain a more accurate location for the solid triangle. If needed, this additional test would require negligible time compared to the overall execution time for the line separation task. The rest of this sequence of operations are required to move the 32 x 32 bit windows toward the right side of the array, one bit column at a time.

9. Subtract the left-most bit field (the first time this is F1) from the ACCUM-FIELD. This removes the left-most bit column that is now not part of the window.
10. Perform Steps 2 and 3 on the next bit slice (the first time this is the 33rd bit column), and store the results in the field used in the subtraction process of Step 6.
11. Add this field to the ACCUM-FIELD.
12. Repeat Step 8.
13. Repeat Steps 9 through 12 until all bit slices, and therefore all 32 x 32 bit windows, have been processed.

Once the whole array image is processed, the locations of the solid triangles are used to locate the succeeding characters. The timing estimates for the line separation routine utilizing a STARAN S-1000 with four arrays are derived below.

1. Load arrays - via common register

$$(4.4 \times 10^{-3})(12 \times 16)(1) = 0.84 \text{ sec.}$$

where,

4.4 = msec/to load one array via common register,
12 x 16 = map area in inches (overlay), and
1 = overlay.
Note: 1 sq in. is processed/array.

2. Tree sums - Modulus 32

$$(25 \times 10^{-6})(32)(256)(12 \times 16)(1) = 9.8 \text{ sec},$$

where

25×10^{-6} = μsec for the tree sum of 1 bit slice (Modulus 32)

32 = tree sums/bit slice (bit slice rotated 31 times)

256 = bit slices per array,

12×16 = map area in inches,

1 = overlay, and

4 = associative arrays (1 sq in. per array).

3. Set up ACCUM FIELD

$$\frac{(15 \times 10^{-6})(32)(12 \times 16)(1)}{4} = 0.02 \text{ sec.}$$

where

15 = μsec/10-bit addition,

32 = additions,

12×16 = sq in. map area,

1 = sq in./array, and

4 = arrays.

4. Accumulations

$$(15 \times 10^{-6})(2)(256)(12 \times 16)(1) = 0.3 \text{ sec},$$

where

15 = μsec per 10-bit add (or subtract),

2 = accumulator operations (1 add and 1 sub.) per bit slice,

256 = 32 bit slices,

5. Total line separation time

$$(0.84 + 9.8 + 0.02 + 0.3)(1.25) = 13.7 \text{ sec},$$

where

1.25 = a 25 percent overhead factor for housekeeping, etc.

3.2.9 Character Recognition

Character recognition is the process required to identify the unknown characters located by the line separation routine. The basic technique described herein is the same as that utilized by the referenced sequential computer, but the algorithm has been restructured for execution on the STARAN AAP. Basically the identification is implemented by deriving classification parameters for each unknown character and comparing these parameters with those contained in a library of known characters. The discussion that follows describes the classification parameters, how they were derived, and how they are associated with the known data to identify the unknown characters.

Each character is contained within a 32 x 32 bit matrix and the classification parameters are simple Fourier Transforms of nine rows and nine columns of the matrix as shown in Figure 3-28. The Fourier Transform (FT) consists of only the DC component (zero harmonic), first harmonic, and second harmonic with each harmonic containing both the sine and cosine terms. With these five FT parameters per line sample and 18 line samples, each unknown character is classified with 90 FT parameters.

Each of the 18 line samples (H1, H2, etc.) is represented by 32 bits. To derive the FT for a character, the 18 line samples are loaded into the first bit slice of the associative arrays as shown in Figure 3-29. Each array can accommodate eight line samples, therefore three arrays are required. The FT "weighting factors" (a_1, a_2, b_1, and b_2) are the same for each line sample. Therefore the four sets of weighting factors" are simultaneously loaded into the three arrays. The 32 word FT weighting factors are the same for each of the 18 line samples.

The DC component of the FT for one line sample is the sum of its ON bits. Therefore, referring to Figure 3-29, the DC components for the 18 line samples are computed by performing a modulus 32 tree sum (see Appendix G) on the line sample data. This computation requires 5 adds. Each component of the FT harmonics is the

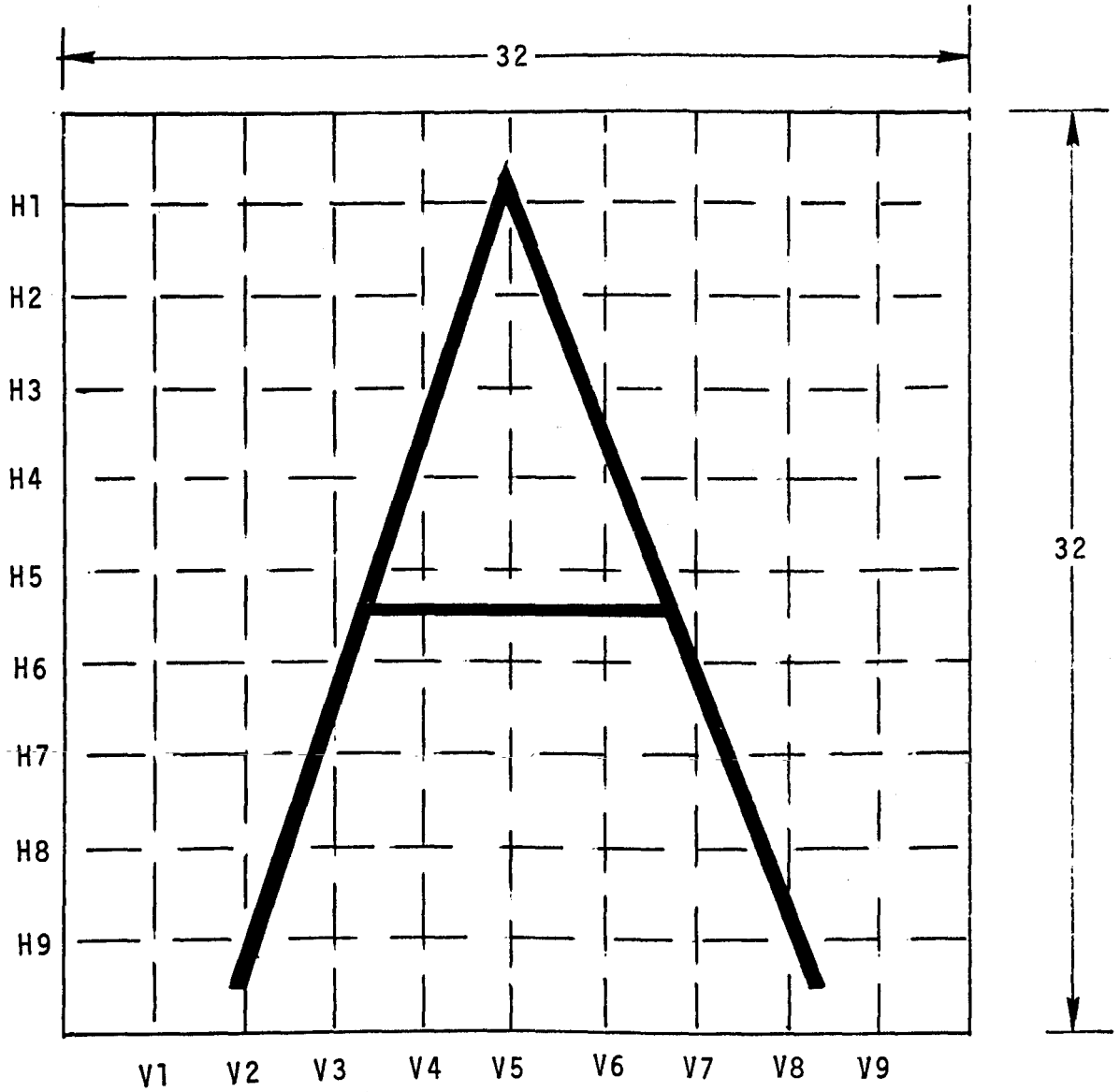


Figure 3-28 - Classification Parameters

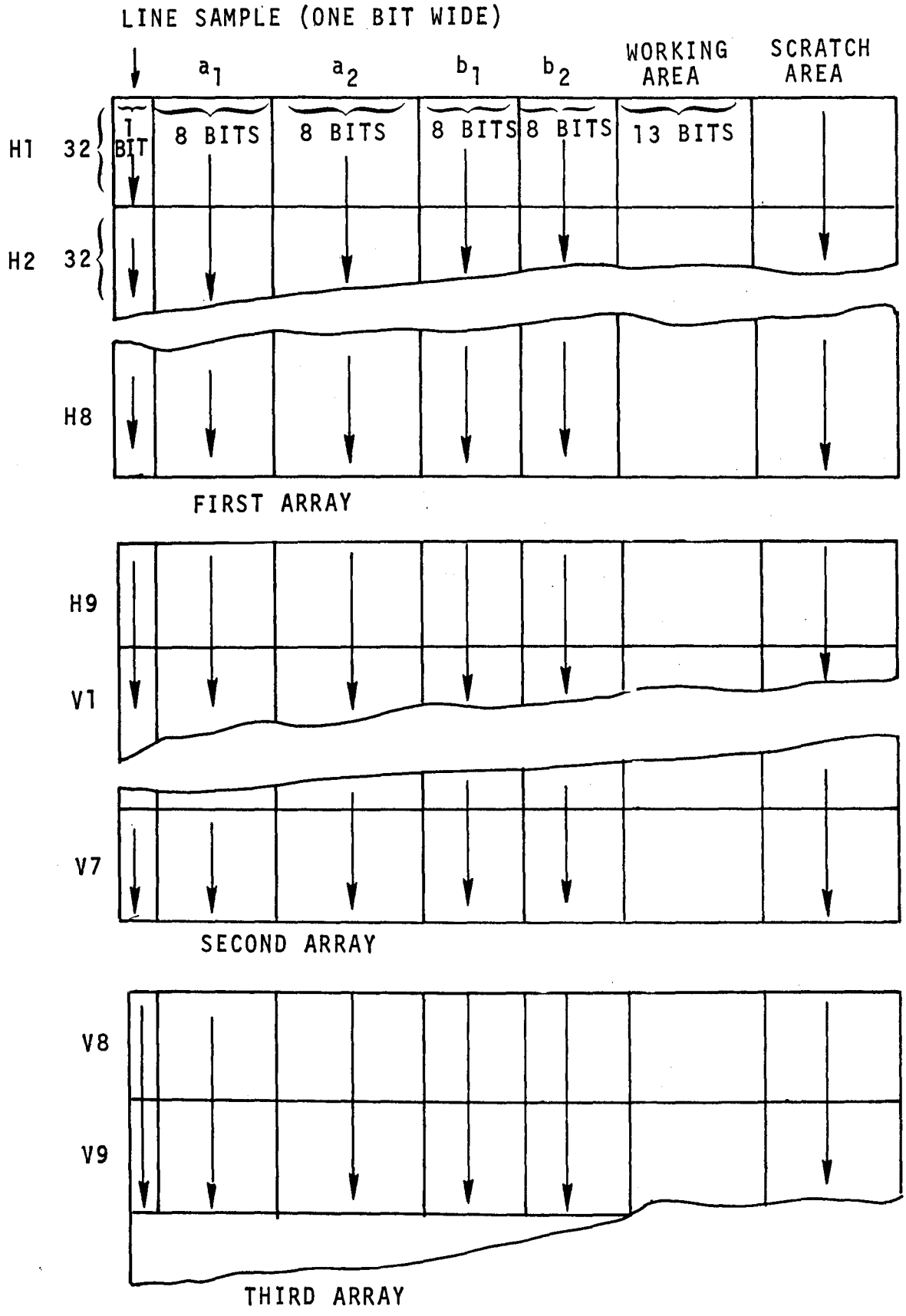


Figure 3-29 - Computation of Fourier Transforms

sum of those words in the weighting factors that correspond to a logical 1 in the line sample data. Therefore each AC component is derived for all 18 line samples by using the line samples as masks and performing modulus 32 tree sums for each of the four AC components. The mask allows selected words to participate in the operation.

The 90 classification parameters just computed for one unknown symbol are transferred to bulk core memory. A new set of line samples for an unknown character are loaded into the arrays and the FT's are computed. This procedure continues until classification parameters have been derived for all unknown characters.

The unknown characters then are identified by comparing them with the known character file. The comparison test is essentially a least squares fit between the classification parameters of the unknown characters and those of each character in the known file. The operations required to perform the identification are as follows:

1. For each of the 39 known characters, load their 90 classification parameters into the arrays as shown in Figure 3-30.
2. Load the 90 classification parameters for one unknown character into the first field of array memory.
3. Using the array working area, compute the differences between the 90 classification parameters of the one unknown character and those of the first field (A through H) of the known characters.
4. Square the differences.
5. Sum the 90 differences associated with each of the eight known character sets using a modulus 128 tree sum.
6. Transfer the eight "sum of the difference squared" results to the $\Sigma DIFF^2$ field shown in Figure 3-30.
7. Repeat Steps 2 through 6 for the remaining four known character fields (I through *).

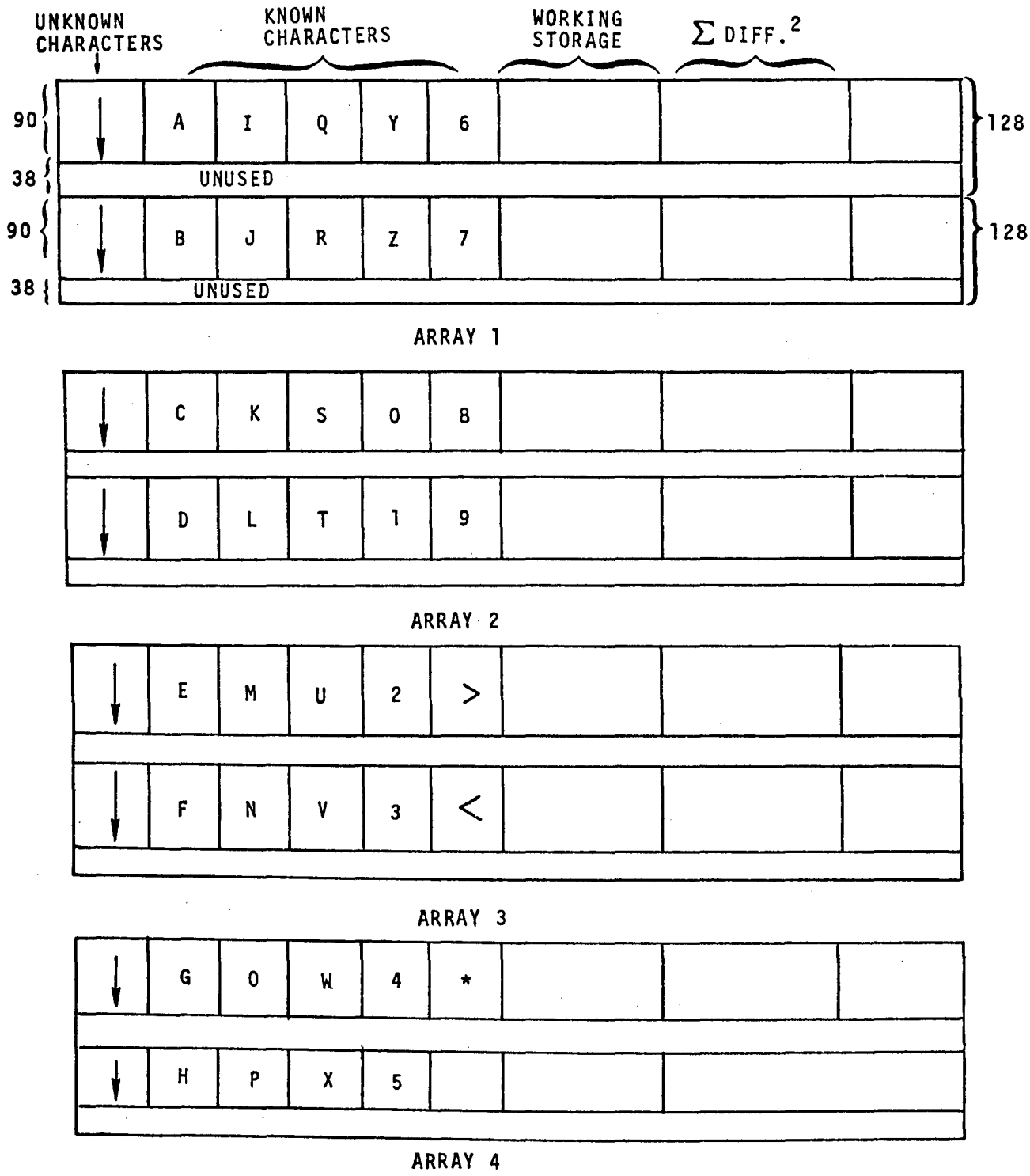


Figure 3-30 - Character Association

8. Perform a "minimum value" search of the $\Sigma DIFF^2$ field to identify the unknown characters. This identification is accomplished in one operation.
9. Repeat Steps 2 through 8 until all unknown characters have been identified.

Timing estimates for the major symbol recognition tasks are derived below.

1. Load arrays with FT weighting factors

$$(3.0 \times 10^{-6})(32)(4) = 384 \text{ µsec},$$

where

3.0 = µsec to load one constant from Bulk core to AM via C register.

2. Load the character sample

$$(2.0 \times 10^{-6})(18)(N) = 36N \text{ sec},$$

where

2.0 = µsec to load one 32-bit value to AM
18 = total number of values to be loaded (H1, ... H9, V1, .. V9), and
N = total number of symbols.

3. Move FT weighting factors to working storage using masked H_i, V_i, i = 1, 2, ... 9)

$$(6 \times 10^{-6})(4)(N) = 24N \text{ µsec},$$

where

6.0 = µsec for a masked move of 8-bit weighting factor to working storage,
4 = number of weighting factors (a_1,a_2,b_1,b_2)
N = total number of symbols.

4. Tree summations

$$[25 \times 10^{-6} + (80' \times 10^{-6})4]N = 345N \text{ µsec},$$

where

25 = µsec to 'tree sum' the unit bit slice modulus 32,
80 = µsec to tree sum the 8 bit fields in working storage (modulus 32),

4 = number of components of the two harmonics (first and second), and

N = total number of symbols.

5. Unload the results of the FT

$(2.0 \times 10^{-6})(5)(18)N = 180N$ μsec,

where

2.0 = μsec to transfer each parameter of each result to bulk core memory,

5 = number of parameters in each result,

18 = number of results, and

N = total number of symbols.

6. Load arrays with parameters of all known characters

$(2 \times 10^{-6})(90)(38) = 6.84$ msecs,

where

2 = μsec to transfer are parameter from Bulk core to AM,

90 = number of parameters associated with each character, and

38 = total number of possible characters.

7. Load arrays with unknown character's parameters

$(2 \times 10^{-6})(90)(N) = 180N$ μsec,

where

2 = μsec to transfer one parameter from bulk core memory to AM,

90 = number of parameters/character,

N = total number of unknown characters.

8. Summation of the square of the parameter differences

$[16 \times 10^{-6} + (192 \times 10\text{-}6) + 240 \times 10^{-6}]5N =$ 2.24N msec,

where

16 = μsec for the 13-bit subtraction,

192 = μsec for the square of 13-bit field,

240 = μsec for the tree sum of the 13-bit field

5 = μsec number of possible fields character is located in, and

N = number of unknown characters.

9. Minimum search of results of previous computation

$(20 \times 10^{-6})N = 20N$ μsec,

where

20 = μsec of search time, and

N = number of unknown characters.

10. Total time for character recognition

$[384 + 36N + 24N + 345N + 180N + 6840 + 180N + 2240N + 20N](1.25) = 9.03 + 12.33N$ msec = 1.87 sec,

where

1.25 = a 25 percent overhead factor for housekeeping, etc., and

N = 151 characters (number of characters on overlay).

3.2.10 Validation

The two raster processing functions of line thinning and line symbol generation were validated in the SETF on the STARAN S-1000 system. To minimize the effort, one array was used to perform the various array manipulations and computations involving the various processes while another was used as a temporary storage area. The execution times obtained were then projected for the four-array system.

A program was written to generate the test data that was used to validate the two raster processing functions. Horizontal, vertical, and diagonal lines were selected as part of the test data because these represented simple multidirectional lines.

A small circle was selected as a test pattern because it essentially represents lines in all directions. Since the line-thinning algorithm, operates on the line images in orthogonal directions, it was assumed that a small circle would present a stringent test of the algorithm.

Line thinning is the most complex of the raster processing functions that was validated. The THIN routine operates on a 245 x 245 bit matrix portion of a map (overlay). The matrix of test data consists

of lines (9 or 10 bits wide) to test the algorithm with both add and even line thickness.

The THIN routine performs a minimum amount of housekeeping to initialize line counters etc; it then performs the line-thinning algorithm eight times. This process reduces the lines from their original width to a unit thickness of one bit. The vertical and horizontal lines thinned to an exact 'one-bit' thick centerline. The diagonal line was also thinned to a centerline, but the ends were bent because the ends of the initial diagonal line were not square to the sides.

These "non-square" line endings could occur when the overlay lines that cross array boundaries are not perpendicular to the boundary. Therefore, to eliminate "pips" at the array boundaries of the final mosaiced map, each array load should contain sufficient overlap with its adjacent neighbors to permit deletion of the distorted boundary area.

The circle pattern produced as a result of the line-thinning process did contain some distortion, but this was within one cell of the expected results.

The validated times to perform the line thinning is shown below.

$$\frac{(13.26 \times 10^{-3})(192)(1)(1)}{(4)} = 0.64 \text{ sec},$$

where

13.26 = msec of validated time/array load,
192 = one overlay, which equals 12 x 16 sq in.,
1 = number of overlays,
1 = number of sq in./array load, and
4 = number of simultaneously processed arrays.

The symbol generation program validates the two functions of single and double line thickening.

The single line thickening produced a one-to nine-bit expansion of the thinned vertical, horizontal, diagonal, and circular test patterns.

The validated time for the single-line thickening process is shown below.

$$\frac{(7.3 \times 10^{-3})(1)(192)(1)}{4} = 0.35 \text{ sec},$$

where

7.3 = msec of time/array load,
4 = number of number of arrays processed simultaneously
1 = number of square inches/array load,
192 = one overlay 12 x 16 square inches, and
1 = number of overlays.

Double-line thickening produces two 9-bit wide lines with a 9-bit gap between them. Essentially a 27-bit-wide line is produced utilizing the single-line thickening process, and a nine-bit-wide line is removed from the center.

The validated time for the double-line thickening process is shown below.

$$\frac{(8.9 \times 10^{-3})(192)(1)(\frac{250}{100})}{4} = 1.1 \text{ sec},$$

where

8.9 = msec of time measure/array load of 100 array words,
192 = square inches/overlay,
1 = overlay,
250 = array words/square inch, and
4 = arrays.

4. SIMULTANEOUS MULTI-EXPOSURE ANALYTICAL CALIBRATION (SMAC) PROGRAM

4.1 INVESTIGATION

4.1.1 Background

The Simultaneous Multi-Exposure Analytical Calibration (SMAC) program is an operational program produced by DBA Systems of Melbourne, Florida, for the U.S. Army Engineer Topographic Laboratories at Fort Belvoir, Virginia, under Contract DAAK02-69-C-0099. The purpose of the SMAC program is to provide an analytical technique for camera calibration using known stellar data. The SMAC program involves:

1. The association of stars in photographic exposures with known cataloged stars.
2. The matrix solution of simultaneous equations involving the known and exposed star positions and camera parameters such as lens distortions and orientation.
3. Many conversion routines to provide a common position and time reference between cataloged and photographed stars.
4. Automatic measurement editing
5. Statistical testing of camera calibration results.

The first two items were considered potential areas for application of a STARAN associative array processor (AAP).

By starting with a minimum of three manually identified stars in each frame, the program automatically computes initial approximations of the absolute orientation angles for each exposure of each camera. The remaining measured stars are automatically identified and the program goes into a single camera SMAC reduction. After processing the data for all cameras of the array on a single camera basis, a similar procedure is used to execute the simultaneous multi-camera array reduction.

The SMAC program is made up of two major sections:

1. Preliminary programs
2. Multiple camera reduction programs

The preliminary programs are further divided into the following five programs:

1. Data preparation program
2. General catalog search program
3. Preliminary orientation computation program
4. Stellar search program
5. Single camera adjustment program

The SMAC programs are presently operational on a Univac 1108 sequential computer. The programs are written in FORTRAN IV and executed using double-precision floating-point arithmetic.

4.1.2 Objectives

The primary objectives for this study were to evaluate the effectiveness with which STARAN could perform various SMAC related search and matrix operations. A secondary objective was to evaluate the suitability of STARAN when applied to miscellaneous time-consuming parts of the SMAC program. Performance of the operational SMAC program presently implemented on the Univac 1108 sequential computer would then provide a reference against which the resulting STARAN timing estimates could be evaluated.

In addition to the matrix inversions required by the SMAC programs, the investigation of STARAN AAP suitability for matrix operations will include matrix multiplication, addition, and subtraction. Each will be evaluated for various size matrixes.

4.1.3 Study

To meet the study objectives, the STARAN timing estimates had to be derived for various SMAC functions and the results compared with those execution times measured for the same functions on the Univac 1108. These steps are facilitated by dividing the SMAC study into three parts:

1. STARAN search operations
2. STARAN matrix operations
3. General SMAC functions

The SMAC related search operation is part of the general catalog search program and stellar search program. The former program

requires an exact match search of star identification numbers in the BOSS star catalog, while the latter program involves a between-limits search of star positions in the Smithsonian Astronomical Observatory (SOA) star catalog.

The matrix operations (Part 2) relate to the inversions required as part of the single camera adjustment program and the multiple camera reduction program. The inversions involve matrixes that are in the neighborhood of 11 by 11 elements and 33 by 33 elements, respectively.

The general SMAC functions (Part 3) will deal with the arithmetic conversion functions that seem to dominate the general catalog search program, the preliminary orientation computation program, and the stellar search program.

The SMAC reference problem presented by the Engineer Topographic Laboratory's Computer Science Laboratory (ETL-CSL) was:

1. Three cameras
2. Twelve exposures (total)
3. One-hundred stars per exposure

The Univac CPU time for the above problem was as follows:

1. Preliminary programs - 13 min
2. Multiple camera reduction programs - 2 min

Initially the assumption was made that Univac 1108 execution times could be obtained for functions within the major programs listed above. However, the effort to obtain the desired timing breakdown was beyond reasonable means. Therefore, the functions evaluated in this three-part study were assumed to represent those parts of the preliminary programs that are responsible for a significant part of the overall execution time. Based on the execution times for the preliminary programs, some rough comparison then could be made between STARAN and the operational system.

All STARAN timing estimates were based upon:

1. One STARAN S-1000 with 4 arrays
2. Floating-point double-precision arithmetic
3. APPLE programming language

The estimates include a 25-percent overhead factor.

For future reference, estimates are also included for two non-standard options that could offer significant improvement over standard STARAN performance. One option would utilize STARAN's 256-bit interface channel that can accommodate a 512 million bit per second transfer rate. This channel is a standard STARAN interface channel that would require a storage device with a similar transfer rate. The other option would be a hypothetical (as yet undeveloped) hardware floating point arithmetic unit. This potential option could interface to the standard STARAN arrays via each array's 256-bit array words simultaneously. This potential device is briefly described in Appendix F.

Timing estimates for the STARAN search operations were derived for a STARAN-parallel head disc/drum (PHD) system. The PHD is a head per track device that permits multiple tracks to be read or written simultaneously. The PHD used for this study has a 256-bit data path, a 512 million bit per second transfer rate (2M bits/sec/track), a latency of 17 msec, and a capacity of 300 million bits. The estimates took into account the PHD characteristics, STARAN performance, record and file formats, file size, and the STARAN configuration.

The performance estimates for the STARAN matrix operations were addressed to matrix addition, subtraction, multiplication, and inversion. The primary thrust was directed toward matrix inversion that was evaluated for two conditions. One condition is for matrix elements that are less than or equal to the total word capacity of the associative arrays. The second case applies to matrixes that exceed the array capacity. For a STARAN S-1000 (1024 array words), the dividing point would be a 32 by 32 element matrix. The basic estimates were derived for the 32-bit array interface.

An alternate matrix inversion method was developed for matrixes greater than 32 by 32 elements, which reduces the computation time at the expense of I/O time. This approach would be useful when utilizing STARAN's 256-bit I/O since the I/O time would then become negligible.

All matrix inversions were implemented using an "exchange" method with complete pivoting. This method is essentially the same as the Gauss-Jordan reduction technique. The timing estimates were made for full matrixes and 10 iterations were assumed for SMAC related inversion estimates.

Rough timing estimates for the general SMAC functions were derived by locating or deriving the appropriate algorithms, determining number and type of arithmetic operations required to solve the algorithms, and applying the STARAN execution times (given in Appendix E) to the arithmetic operation count. Also taken into consideration was the amount of data and the number of available STARAN associative array words. Because of the large number of arithmetic operations involved per data load, the array load/unload times were negligible and not taken into consideration.

The applicable algorithms were obtained or derived from the SMAC programs final report submitted to ETL-CSL by DBA Systems, Inc. and are included in Appendix C.

4.1.4 Results

STARAN timing estimates are included for the general SMAC functions, search operations, matrix addition (or subtraction), matrix multiplication, and matrix inversion. All estimates were based on double-precision floating-point arithmetic and a four-array STARAN.

Timing estimates for the general SMAC functions are given in Table 4-I as a function of the subtasks under consideration. The estimates are for 3 cameras and 12 exposures (total) with approximately 100 stars per exposure. Ten iterations were assumed for functions requiring multiple iterations to obtain a solution.

Timing estimates are also included for a hypothetical (as yet undeveloped) hardware floating-point arithmetic option to illustrate a viable approach that could be developed to extend STARAN performance. This potential option could improve STARAN performance by approximately one order of magnitude for applications requiring double-precision floating-point arithmetic. This option is discussed briefly in Appendix F.

The Univac 1108 CPU times measured for the sample problem are included in Table 4-I as a general reference, but a direct comparison of performance between the two devices is misleading. It was initially assumed that those parts of the SMAC programs evaluated for STARAN represented a significant part of the overall SMAC execution times; at least 50 percent. However, STARAN computation time is proportional to field size and therefore with double-precision floating-point arithmetic, there is no obvious reason for a STARAN S-1000 to significantly outperform the Univac 1108 as indicated by the results in Table 4-I.

TABLE 4-I - STARAN TIMING ESTIMATES FOR GENERAL SMAC FUNCTIONS

Functions	Estimated STARAN S-1000 time (sec)*+		Actual Univac 1108 time (min)
	Standard	Hardware arithmetic≠	
A. Preliminary programs	20.7 sec	2.82 sec	13 min
I Data preparation	-	-	
II General catalog search	0.4	0.09	
III Preliminary orientation computation	2.7	0.24	
IV Stellar search	14.9	2.3	
V Single camera adjustment (STARAN estimate for 11 x 11 matrix inversions only)§	2.7	0.19	
B. Multiple camera reduction Program (STARAN estimates for 32 x 32 matrix inversions only)§	2.6 sec	0.19 sec	2 min

*+Estimated for double-precision floating software arithmetic. Times do not include various editing and testing functions.

≠Hypothetical hardware floating-point arithmetic option (not presently developed).

§10 iterations were assumed for all iterative functions.

The search functions and/or various automatic editing and statistical testing functions in the SMAC programs may account for a large part of the Univac 1108 execution time. If this is the case, then the "apparent" STARAN performance advantage could result from STARAN's superior ability to perform the search functions and/or because the STARAN estimates to not include the editing and testing functions. However, the absence of a more detailed timing breakdown for the Univac 1108 execution times precludes making a direct comparison of the times estimated for the STARAN S-1000 and measured for the Univac 1108.

Results of the STARAN timing estimates for SMAC related search operations are shown in Table 4-II. With one exception, the estimates were derived for a STARAN parallel head disc/drum (PHD) system.

The exception is the fine search part of the SAO catalog search that involves data that were processed prior to this search and will be residing in STARAN's bulk core memory. In Table 4-I the BOSS catalog search is included as part of the General Catalog Search Program estimate and the SAO catalog searches are included as part of the Stellar Search Program estimates.

Timing estimates for matrix addition (or subtraction), matrix multiplication, and matrix inversion are shown in Table 4-III. The results were derived for a standard four-array STARAN where array loading and unloading was produced via the 32-bit interface. Timing estimates are given for two approaches to matrix inversion. The two approaches differ in implementation with respect

TABLE 4-II - STARAN TIMING ESTIMATES FOR SMAC RELATED SEARCH OPERATIONS

Search operation	STARAN S-1000 Estimated search times (sec)
BOSS Catalog*	0.068
SAO Catalog*	1.22
Coarse search	
Fine search	0.08

*Estimated for STARAN Parallel Head Disc/Drum (PHD) system

TABLE 4-III - STARAN S-1000 TIMING ESTIMATES FOR VARIOUS MATRIX OPERATIONS

Matrix size	STARAN S-1000 execution time*,+			
	Addition or subtraction	Multiplication	Inversion	
			Basic	Alternate
11 x 11	2.3 msec	15.6 msec	89 msec	89 msec
32 x 32	13.1 msec	260.6 msec	264 msec	264 msec
128 x 128	209.7 msec	20.41 sec	17.4 sec	26.2 sec
512 x 512	3.36 sec	28.5 min	19.1 min	27.6 min

*Estimated for a standard STARAN using the 32-bit interface for array data transfer.

+Estimated for double-precision floating-point arithmetic via software.

to data arrangement and processing sequence. The basic approach minimizes the array I/O time at the expense of increased calculation time, but derives an overall time advantage over the alternate technique that minimizes calculation time at the expense of increased array I/O time. The difference between the two methods is only realized when the matrix size exceeds the STARAN capacity. The calculation and array I/O times for the two matrix inversion methods are separated and shown in Table 4-IV.

TABLE 4-IV - STARAN S-1000 TIMING BREAKDOWN FOR MATRIX INVERSIONS

Matrix size (elements)	STARAN S-1000 execution times*,+,≠					
	Basic Approach			Alternate Approach		
	Calculations	I/O	Total	Calculations	I/O	Total
11 x 11	88 msec	1 msec	89 msec	88 msec	1 msec	89 msec
32 x 32	256 msec	8 msec	264 msec	256 msec	8 msec	264 msec
128 x 128	15.8 sec	1.6 sec	17.4 sec	9.5 sec	16.7 sec	26.2 sec
512 x 512	16.6 min	2.5 min	19.1 min	9.7 min	17.9 min	27.6 min

*Estimated for a standard STARAN using the 32-bit interface for array data transfer.

+Estimated for double-precision floating-point arithmetic via software.

≠Estimated for full matrices.

The timing estimates for matrix inversion were derived using an "exchange" technique with complete pivoting. This method is essentially the same as the Gauss-Jordan reduction technique. The estimates are for full matrixes; sparse matrixes would require proportionately less execution time.

Included for reference are timing estimates for the various matrix operations utilizing STARAN's 256-bit parallel input/output (PIO) channel with a compatible random access memory and the hypothetical hardware floating-point arithmetic units discussed previously and briefly described in Appendix F. These estimates are given in Tables 4-V and 4-VI for matrix addition (or subtraction) and multiplication, respectively. The results for matrix inversion are given in Tables 4-VII and 4-VIII for the basic and alternate methods respectively.

TABLE 4-V - STARAN S-1000 TIMING ESTIMATES FOR MATRIX ADDITION (OR SUBTRACTION)

Matrix size (elements)	STARAN S-1000 execution times			
	Standard	PIO	Hardware Arithmetic	PIO and Hardware Arithmetic
11 x 11	2.3 msec	0.92 msec	1.7 msec	0.3 msec
32 x 32	13.1 msec	0.92 msec	12.5 msec	0.3 msec
128 x 128	209.7 msec	14.6 msec	199.7 msec	4.6 msec
512 x 512	3.36 sec	0.25 msec	3.19 sec	0.24 sec

*Estimated for double-precision floating-point arithmetic.

+Hypothetical hardware floating-point arithmetic (presently undeveloped).

The PIO option would be most effective for matrix addition (or subtraction) and matrix inversion by the alternate method. The hardware arithmetic would be more useful for matrix multiplication and matrix inversion by the basic method.

TABLE 4-VI - STARAN S-1000 TIMING ESTIMATES FOR MATRIX MULTIPLICATION

Matrix size (elements)	STARAN S-1000 execution times*			
	Standard	PIO	Hardware Arithmetic+	PIO and Hardware Arithmetic
11 x 11	15.6 msec	14 msec	3.74 msec	2.2 msec
32 x 32	260.6 msec	250.4 msec	51.1 msec	40.9 msec
128 x 128	20.41 sec	19.33 sec	4.43 sec	3.34 sec
512 x 512	28.5 min	24.1 min	8.7 min	4.3 min

*Estimated for double-precision floating-point arithmetic.

+Hypothetical hardware floating-point arithmetic (presently undeveloped).

TABLE 4-VII - STARAN S-1000 TIMING ESTIMATES FOR MATRIX INVERSION - BASIC METHOD

Matrix size (elements)	STARAN S-1000 execution times*,+			
	Standard	PIO	Hardware Arithmetic≠	PIO and Hardware Arithmetic
11 x 11	89 msec	88 msec	6.3 msec	6.3 msec
32 x 32	264 msec	257 msec	26.6 msec	18.5 msec
128 x 128	17.4 sec	15.8 sec	2.2 sec	0.60 msec
512 x 512	19.1 min	16.7 min	3 min	0.7 min

*Estimated for double-precision floating-point arithmetic

+Estimated for full matrixes.

≠Hypothetical hardware floating-point arithmetic (presently undeveloped).

TABLE 4-VIII - STARAN S-1000 TIMING ESTIMATES FOR MATRIX INVERSION - ALTERNATE METHOD

Matrix size (elements)	STARAN S-1000 execution times*,+			
	Standard	PIO	Hardware Arithmetic≠	PIO and Hardware Arithmetic
11 x 11	89 msec	88 msec	6.3 msec	6.3 msec.
32 x 32	264 msec	257 msec	26.6 msec	18.5 msec.
128 x 128	26.2 sec	10.0 sec	17.6 sec	0.82 sec.
512 x 512	27.6 min	10.3 min	18.7 min	0.84 min.

*Estimated for double-precision floating-point arithmetic.

+Estimated for full matrixes.

≠Hypothetical hardware floating-point arithmetic (presently undeveloped).

4.2 DISCUSSION

4.2.1 General

The details of the three-part SMAC study are discussed below. The three parts are:

1. STARAN search operations
2. STARAN matrix operations
3. General SMAC functions

The STARAN search operations are described with respect to the STARAN Parallel Head Disc/Drum (PHD) system proposed to implement this function. For the STARAN matrix operations, a detailed description is given of the concepts involved and the associative array techniques utilized to implement the algorithms. The general SMAC functions are discussed by briefly describing the functions, operational sequence, arithmetic operations required by each function volume of data involved, and respective execution times.

4.2.2 STARAN Search Operations

4.2.2.1 General - There are two types of searches involved in the SMAC program:

1. BOSS catalog search
2. SAO catalog search

The BOSS catalog search involves locating stars within the BOSS catalog whose BOSS number corresponds to the reference stars. Figure 4-1 shows the BOSS catalog file and record formats.

The SAO catalog search involves locating all stars in the Smithsonian Astronomical Observatory catalog whose right ascension and declination values fall within certain predetermined values. Figure 4-2 shows the SAO catalog file and record formats.

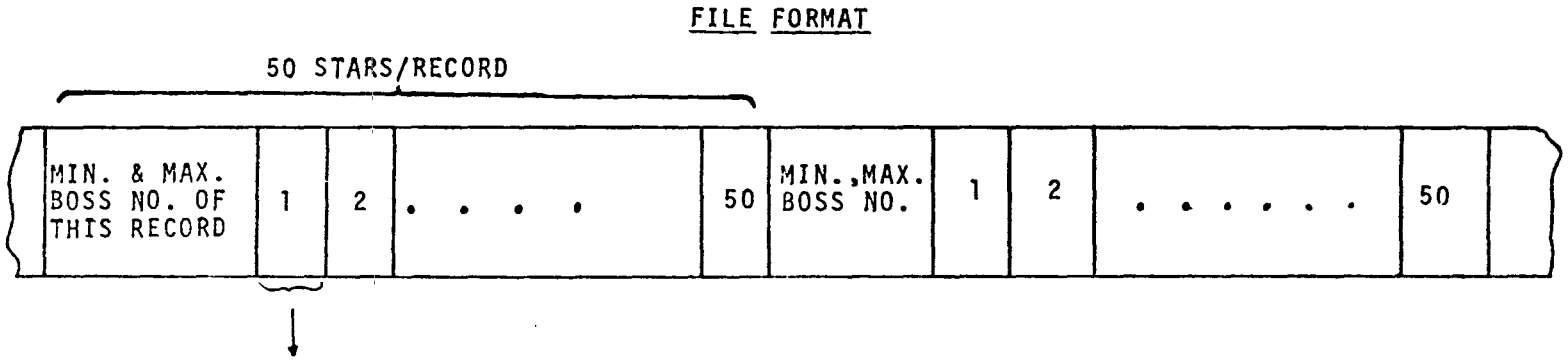
Search operations can be performed very efficiently in a STARAN parallel head disc/drum system. A PHD is essentially a disc/drum with a multiple number of read/write heads that can simultaneously transfer data; in this case there will be a 256-bit wide data channel.

Modifications to both the SAO and BOSS catalog are necessary to obtain maximum searching efficiency between the STARAN and the PHD. The modified record formats for both of these tapes are shown in Figure 4-3.

More detailed descriptions of the two operations are given below.

4.2.2.2 BOSS Catalog Search - The BOSS catalog is a subset of the SAO catalog and contains approximately 33,000 stars. This catalog is modified for searching efficiency and stored on the PHD. Each star has a record length of 276 bits. This length is larger than the maximum number of bits located on one track within one sector on the PHD (256 bits), therefore two tracks within a sector are allocated to each record (star). Each of these records will occupy two STARAN array memory words (see Figure 4-4). Each block on the PHD is 256 tracks by 256 bits long and will contain 128 records.

There are 260 sectors/track and therefore in one revolution approximately 260 blocks of information are available. The whole BOSS catalog requires about 258 blocks, thus the entire catalog may be stored in one revolution.



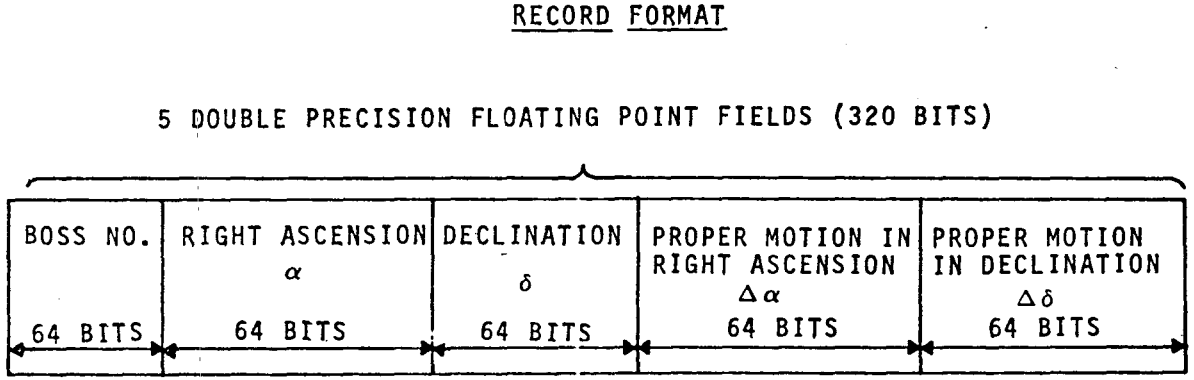


Figure 1 - Boss Catalog File and Record Formats

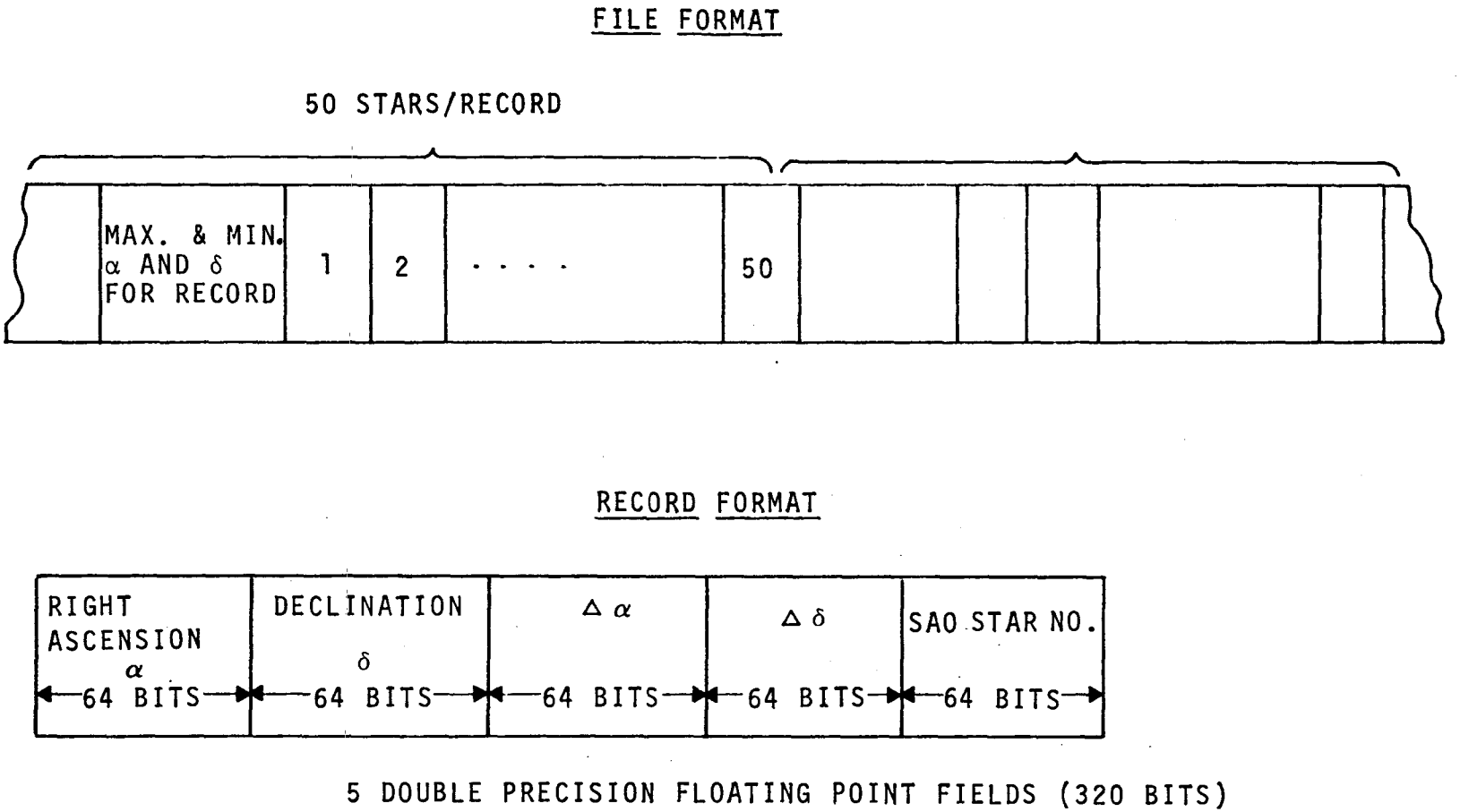


Figure 4-2 - SAO Catalog File and Record Formats

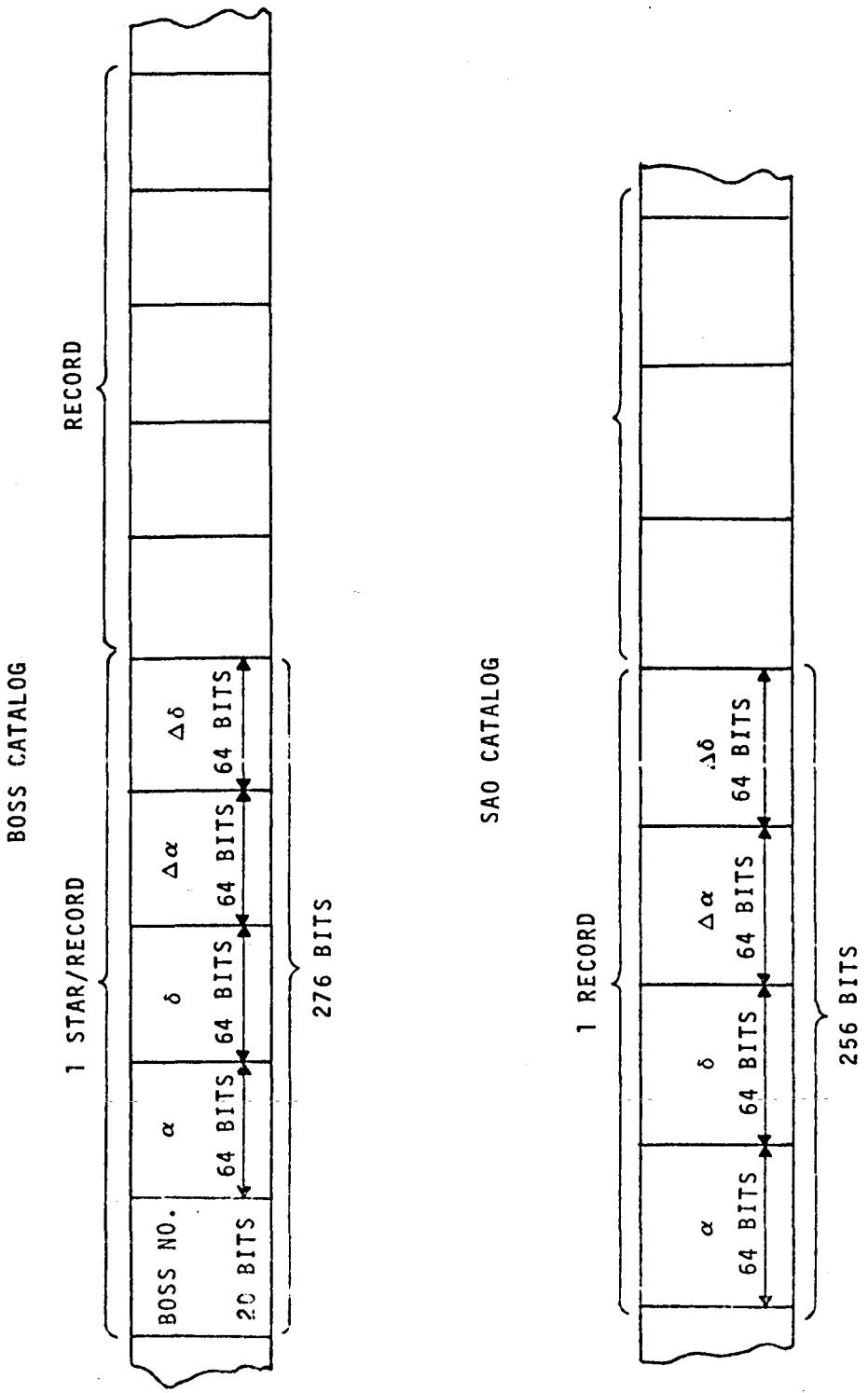


Figure 4-3 - Modified SAO and BOSS Catalogues

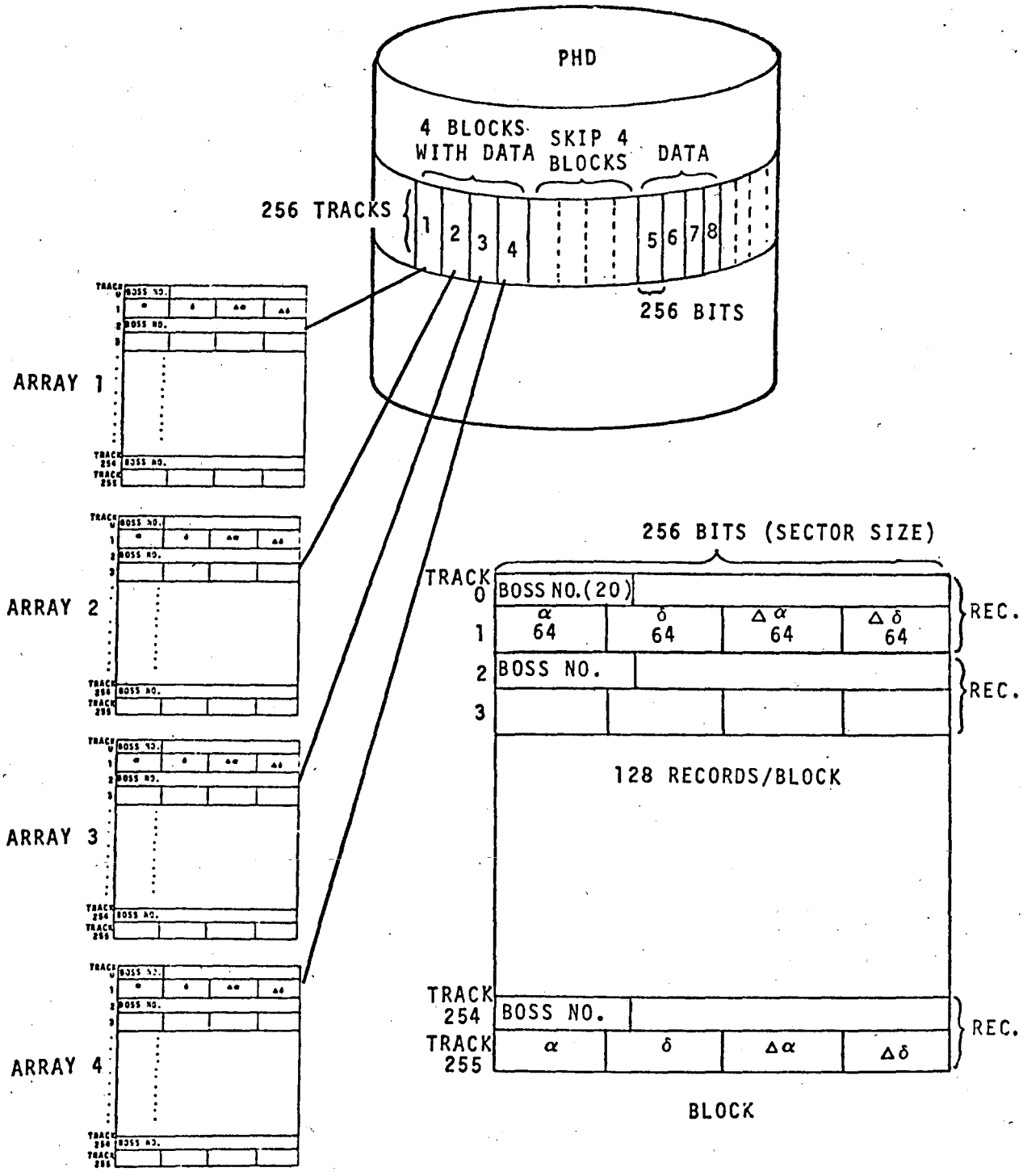


Figure 4-4 - PHD and Array Record Format

As stated earlier in the discussion, this problem is based on the following assumptions:

1. There are 12 exposures.
2. There are 100 stars per exposure.
3. Four stars are manually selected from each exposure and are identified by BOSS number. The 48 stars then are used for the BOSS catalog search.

The searching sequence is as follows:

1. Read four blocks of data into four array memories. The time for this operation is:
   $(0.5)(256)(4) = 512$ μsec,

where

0.5 μsec/bit = transfer rate,
256 = number of bits/array, and
4 = number of arrays.

2. Perform the 48 "exact match" operations on the BOSS catalog star numbers for the 48 reference stars and and move the matched stars to the high-speed data buffer (HSDB). The total time for this operation is:
   $(1.0)(48) + (4.0)(48) + (1.0)(8)(24) = 432$ μsec,

where:

1.0 μsec = time to load reference stars number from HSDB to common register,
48 = total number of reference stars,
4.0 μsec = time to perform match between common register and 20-bit array field,
1.0 μsec = time to move 32-bit array field to HSDB,
8 = number of 32-bit moves per 256 bit array word, and
24 = an estimated "upper band" on the probable number of exact matches.

3. Wait until the next four blocks of data reaches the "read head" on the PHD and repeat operations 1 and 2 until the whole BOSS catalog is searched. Since each four-block search requires 432 μsec, the search is made on every other four blocks of data. This procedure

provides approximately 512 μsec to perform the necessary searching and data moving operations. As a result, the whole BOSS catalog can be searched in two disc revolutions. For a revolution time of 34 msec, the total BOSS catalog search will therefore be 68 msec.

4.2.2.3 SAO Catalog Search

4.2.2.3.1 Types of Searches - The SAO catalog search actually involves two searches. The first is a coarse search that locates stars on the SAO catalog tape that are in the area of interest. The responding star positions are updated to the time of exposure and a fine search is performed.

4.2.2.3.2 Coarse Search - The SAO catalog contains about 160,000 stars. The modified catalog is stored on the PHD using the record format shown in Figure 4-3. The SAO number is excluded in the record definition because it is not required in this program. The new record length of 256 bits fits into one sector track or one array word. Each block on the PHD may store 256 records (stars); the whole catalog requires:

$$\frac{160{,}000}{256} \simeq 640 \text{ blocks.}$$

Unlike the BOSS catalog search, the whole file will not fit into 256 tracks; 3 x 256 tracks of storage are required. The object of the SAO catalog search is to find stars contained in the SAO tape whose right ascension and declination (α,δ) fall within the range of values found on each exposure, that is:

$$\alpha_1 < \alpha < \alpha_2;\ \delta_1 < \delta < \delta_2$$

where

α_1, α_2 and δ_1, δ_2 are the minimum and maximum values for a particular exposure.

Four searches are needed therefore to satisfy these criteria.

The sequence of search operations is as follows (refer to Figure 4-5:

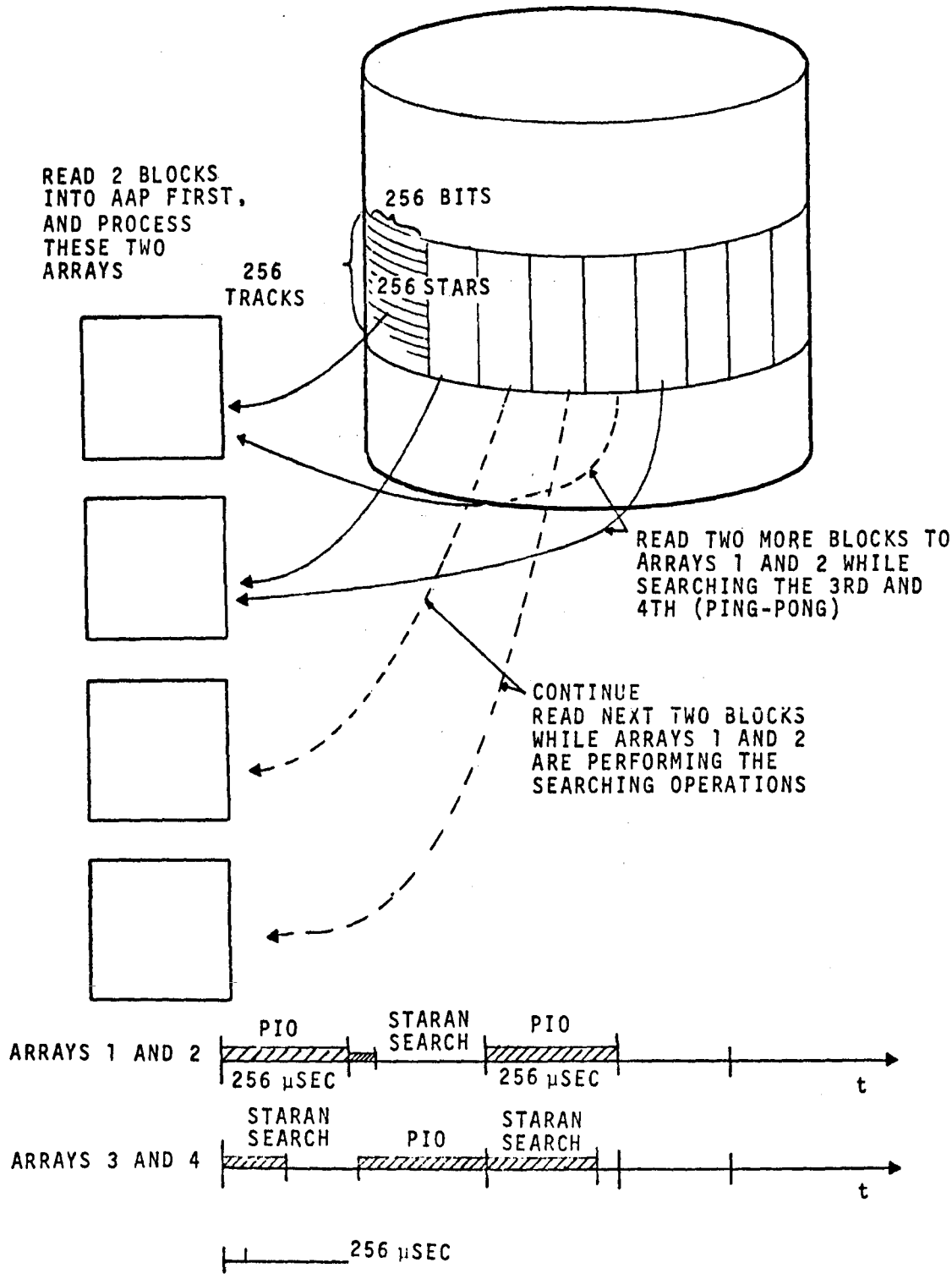


Figure 4-5 - SAO Catalog Search Sequence

1. The PIO controls two array memories (for example, Arrays 1 and 2) and two blocks of data are read into these two arrays. The STARAN controls the other two arrays (Arrays 3 and 4) and performs the four necessary searching operations. The operations are two between-limit searches. Data that satisfy the conditions are moved to bulk core memory by STARAN.
2. The PIO and STARAN switch array control when the PIO completes loading two blocks of data and Step 1 above is repeated. Figure 4-5 shows this "ping-pong" operation. Using this search technique, only three revolutions are required to search the whole data base of 160,000 stars for one exposure. For the 12 exposures, the total searching time is:

$$(3)(34 \times 10^{-3})(12) = 1.22 \text{ sec},$$

where

3 = number of revolutions,
34 msec = time/revolution, and
12 = number of exposures

To maintain an uninterrupted ping-pong search operation, the SAO catalog records are randomly stored. This procedure is explained by the following discussion. The time to load two data blocks into two arrays is:

$$(0.5)(256)(12) = 256 \text{ µsec},$$

where:

0.5 µsec = load time/bit slice of 256 words,
256 = number of bit slices, and
2 = number of arrays.

Within the above time shot, STARAN has to perform four searches.

The star positions are stored in the SAO catalog as 64- floating point numbers (see Figure 4-3). However, the assumption was made

that 32 bits would provide sufficient accuracy for this between-limits search task. For 32-bit fields, each between-limits search will require approximately 8 μsec. There are 224 μsec left to move those responding stars to bulk core memory.

Each record (star) has 256 bits to be moved, which takes the following time:

$$(2)\frac{(256)}{32} = 16 \text{ μsec},$$

where:

2 μsec = time to perform the 32-bit transfer, and
256 = number of bits to be transferred for each response.

This equation shows that $\frac{224}{16}$ or 14 responses may be performed in each load. If the records are ordered on the PHD according to α,δ, or both, then the responses will normally be either none or more than fourteen.

A total of about 400 stars are expected to respond in one exposure. There are 640 blocks of data to be searched per exposure. This amount will require 320 STARAN loads (two blocks/load). If the stars are stored in a completely random fashion, the average number of response stars per load will be:

$$\frac{400}{320} = 1.25$$

This number is much less than the maximum allowed response number of 14. Therefore, by storing the stars randomly, the chances of disturbing this "ping-pong" scheme are very unlikely.

The SAO catalog search was based on an approach that searched the whole catalog for each exposure. From the above arguments, the fact becomes obvious that multiple exposures could be tested against each STARAN data load. As a result, the total search time of 1.22 sec could be reduced proportionally. Therefore:

Total SAO catalog search time = $\frac{1.22}{N_E}$ sec,

where N_E is the number of exposures tested per STARAN load.

4.2.2.3.3 Fine Search - After updating all stars responding to the coarse search described above, a more accurate between-limits search will be performed. The objective of this search will be to associate each star of the 12 exposures (100 stars per exposure) with the updated SAO cataloged stars (400 stars per exposure area) to identify the photographed stars.

This operation is described briefly:

1. Load the 400 updated stars, associated with a particular plate, into the associative arrays from bulk core memory.
2. Bring one star from the corresponding plate to the common register and test against the 400 stars in the arrays. An association is made if the differences in right ascension, α, and declination, δ, are within a specified range. This association can be accomplished by four, 32-bit comparisons (greater than or less than common register). If there is a match, an indicator will be set to reflect the association. This process will be repeated for all 100 stars of each plate.
3. Repeat Steps 1 and 2 for each plate. The total time for this operation is:

$$\{400(4)(2) + 100[2(2) + 4(8)]\}\ 12 = 81.6 \text{ msec},$$

where

- $400(4)(2)$ = stars, each with 128 bits, that move into array memory from bulk core memory (μsec),
- 100 = stars in a plate,
- $2(2)$ = transfer time for α and δ (2 μsec each),
- $4(8)$ = comparisons (8 μsec each), and
- 12 = total number of plates.

4.2.3 STARAN Matrix Computations

4.2.3.1 General - Matrix manipulation accounts for a large portion of the time for both the single camera adjustment and multiple camera reduction programs. Instead of going into a detailed treatment for both programs, a comprehensive description of the approach of a parallel processor to many matrix manipulations is given.

Parallel properties will be shown to be inherent in matrix operations. This fact makes the development and implementation of a parallel algorithm for matrix manipulations extremely easy.

In the following discussion, matrix addition and/or subtraction, matrix multiplication, and matrix inversion techniques are considered under three subsections. Examples are given to show the memory maps within STARAN's associative array. A comparison with conventional sequential computers in terms of major operations is also provided. A STARAN S-1000 (four array) system is assumed.

4.2.3.2 Matrix Addition and Subtraction - The following notation will be used during this discussion:

$$A_{mxn} \pm B_{mxn} = C_{mxn}$$

where

A_{mxn}, B_{mxn}, and C_{mxn} stand for matrixes of m rows and n columns. Elements of these matrixes are a_{ij}, b_{ij} and c_{ij} where i = 1, 2 ... m and j = 1, 2 ... n. Addition or subtraction is the simplest matrix operation. In sequential computers, normally mn addition (or subtraction) operations are required. In an associative array processor (AAP), only one operation is normally required. This statement assumes that the array capacity is large enough to accommodate all of the matrix elements.

The STARAN S-1000 has 1024 words (processing elements); therefore, the total number of operations required utilizing the S-1000 are:

$$\left[\frac{mn}{1024}\right] \text{ (additions or subtractions),}$$

where [ ] represents an integer value, or the next higher integer

if it contains a fractional part. Table 4-IX gives a comparison of the number of major operations required in the two computers.

TABLE 4-IX - MATRIX ADDITION (OR SUBTRACTION) OPERATIONS

Matrix size	Sequential computer	STARAN S-1000
$C_{mxn} = A_{mxn} \pm B_{mxn}$	$mn \oplus$	$[\frac{mn}{1024}] \oplus$
$C_{30x30} = A_{30x30} \pm B_{30x30}$	$900 \oplus$	$1 \oplus$
$C_{64x60} = A_{64x60} \pm B_{64x60}$	$3840 \oplus$	$4 \oplus$

Legend:

$\oplus$ = addition or subtraction operations.

Two operations are covered by one symbol because the time to perform either operation is approximately the same.

In referring to the general formula in Table 4-IX, a 32x32 matrix addition on a STARAN S-1000 using double-precision floating-point arithmetic takes the following time:

$$[\frac{32x32}{1024}] \ (816x10^{-6}) = 816 \ \mu sec,$$

where

$[\frac{32x32}{1024}]$ = the number of addition operations.

816 μsec = the STARAN double-precision floating-point addition time (see Appendix E)

Figure 4-6 shows a section of an array memory and indicates how the elements a_{ij}, b_{ij} of two matrixes A, B may be arranged within a STARAN array to perform matrix addition (or subtraction) in a parallel fashion.

The fact is easy to see that the operation:

ADD (or SUBTRACT) FIELD (a) to FIELD (B) and store the results in FIELD (C) produces the sum (or difference) of all elements of the two matrixes.

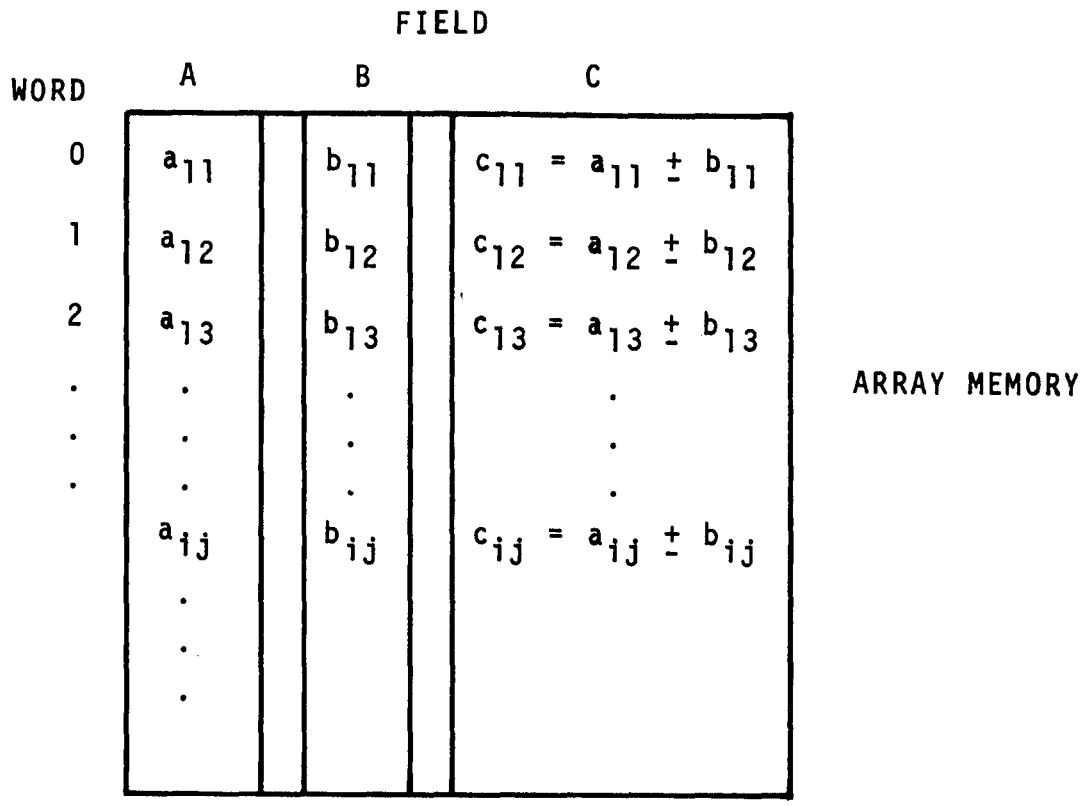


Figure 4-6 - Array Layout for Matrix Addition and Subtraction

4.2.3.3 Matrix Multiplication - The following notation is used below for the multiplication of two matrixes:

$$A_{mxp} \cdot B_{pxn} = C_{mxn}$$

where the elements c_{ij} of the matrix C_{mxn} are defined by:

$$c_{ij} = \sum_{k=1}^{p} a_{ik} \cdot b_{kj},$$

where

$i = 1, 2, \ldots m$, and

$j = 1, 2, \ldots n$.

In a sequential computer, mpn operations are required and each involves one multiply and one addition. In a parallel processor,

these operations may be distributed among the processing elements. In STARAN, the number of multiplications will reduce to:

$$[\frac{mpn}{1024}] \otimes .$$

There are 'p' elements to be summed for each row of matrix A times each column of Matrix B. The 'tree sum' technique is used to accumulate these p values (see Appendix G). Therefore $\log_2 p$ additions are required.

Table 4-X gives a comparison for the number of major operations required in the two types of computers.

TABLE 4-X - MATRIX MULTIPLICATION OPERATIONS

Matrix size	Sequential computer	STARAN S-1000
$C_{mxn} = A_{mxp} \cdot B_{pxn}$	mpn $\otimes$	$[\frac{mpn}{1024}] \otimes$
	mpn $\oplus$	$[\frac{mpn}{1024}] (\log_2 P) \oplus$
$C_{30x30} = A_{20x32} \cdot B_{32 \cdot 30}$	28,800 $\otimes$	$[\frac{30 \cdot 32 \cdot 30}{1024}] \otimes$
	28,800 $\oplus$	$[\frac{30 \cdot 32 \cdot 30}{1024}]$ 5 $\oplus$
$C_{64x64} = A_{64x32} \cdot B32 \cdot 60$	122,880 $\otimes$	120 $\otimes$
	122,880 $\oplus$	720 $\oplus$

By referencing the general expression in Table 4-X, a 32x32 matrix multiplication on a STARAN S-1000, using double-precision floating-point arithmetic, takes the following time:

$$[\frac{32^3}{1024}](3.59 \times 10^{-3}) + [\frac{32^3}{1024}](\log_2 32)(816 \times 10^{-6}) =$$

$$0.2454 \text{ sec},$$

where

$[\frac{32^3}{1024}]$ = number of multiplication operations,

3.59×10^{-3} msec = double-precision floating-point multiplication time (Appendix E),

$[\frac{32^3}{1024}](\log_2 32)$ = number of addition operations, and

816 μsec = double-precision floating-point additon time (Appendix E).

If fixed-point arithmetic is used, multiplication operations take much longer than addition operations. In this situation the Winograd method of matrix multiplication should be used in which the number of multiplications are reduced by almost one half, and the number of addition operations are slightly increased. Some extra housekeeping operations are also required in this method[a].

Figure 4-7 shows a memory map of the data arrangement within the array memory for the STARAN approach to matrix multiplication. For better understanding, the multiplication of Matrix A_{3x4} by Matrix B_{4x2} is discussed.

Field 1 (F1) and Field 2 (F2) in Figure 4-7A contain the matrix elements a_{ik} and b_{kj}, respectively; notice the repetitive arrangement of these elements within the array memory words. All four components of all elements c_{ij} of the Matrix C are obtained in one multiplication; namely:

F1 x F2 → F3.

These components are shown in Field F3 of Figure 4-7A. Each group of four components $a_{ik}\ b_{jk}$ are summed by utilizing the "tree sum" technique discussed in Appendix G; the result is seen in Figure 4-7B.

The total number of word used are:

mpn = 3 x 4 x 2 = 24

Although the elements a_{ik}, b_{kj} are repeated at different points within their respective fields, this fact does not imply that each has to be loaded repeatedly. Between-word communication is very efficient within the arrays and is much faster than 'load' times. This means that once one set of the elements has been loaded, it

---

[a] Knuth, Donald E.; The Art of Computer Programming: Volume 2 - Seminumerical Algorithms. Reading, Mass., Addison-Wesley Publishing Co. 1968. p. 427.

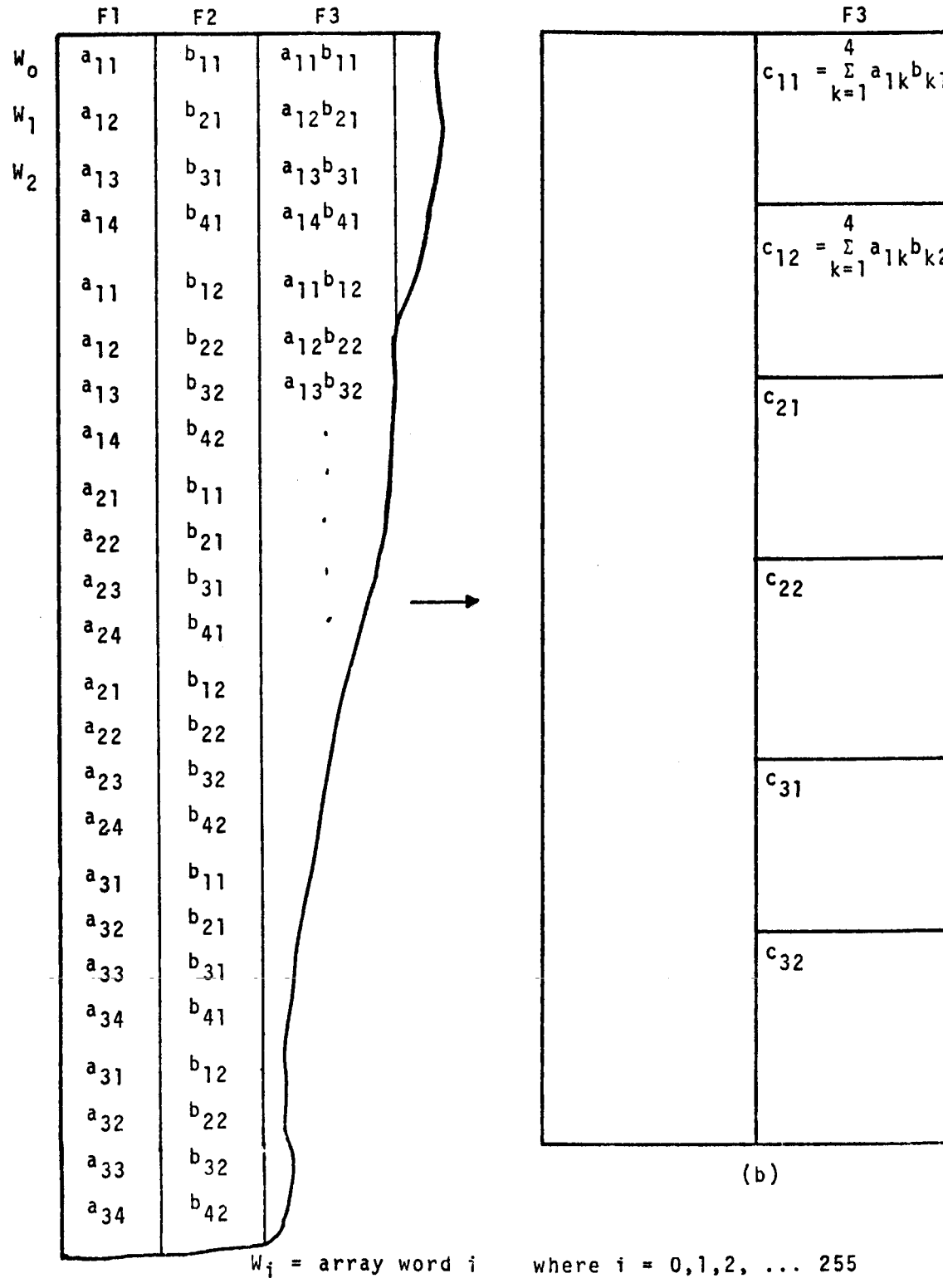


Figure 4-7 - Array Memory for Matrix Multiplication

can be replicated in other required locations. This replication procedure minimizes array loading times. This method utilized in replicating the variable input data within the AS-11BX postprocessing (coordinate translation) is such a method.

4.2.3.4 Matrix Inversion

4.2.3.4.1 General - The techniques involved in matrix inversion are more complicated than the other matrix arithmetic operations, but are still very applicable to associative array processing. There are both direct and indirect approaches to the problem; this discussion shows one particular direct method.

Although there are several techniques available, the two most widely used are:

1. Gaussian method
2. Gauss-Jordan method

The Gaussian method requires a "back substitution" in the matrix inversion process. This method is more suitable for a sequential processor because of the serial type approach. The Gauss-Jordan method requires more steps if implemented on a sequential machine, because the matrix inverse is directly computed. However, this technique is well suited to STARAN because there is essential parallelism within this method and ensures maximum use of the associative array processing capabilities of the STARAN.

Included in this discussion is a reference to "complete pivoting", which reduces errors and increases stability in the matrix inversion process. These errors may be obtained when a matrix element of small magnitude is used as a divisor during the matrix inversion. Pivoting is normally a time-consuming process, but this area is greatly improved by fully utilizing the parallel capabilities of STARAN.

4.2.3.4.2 Matrix Inversion (Basic Approach for $N \leq 32$) - The basic matrix inversion technique used in a STARAN S-1000 for a matrix is not larger than 32x32 elements. This size is the largest matrix that can be inverted using this technique in a four array AAP (1024 words). A set of four simultaneous equations are used here to show how the matrix inversion process works.

$$\left.\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 &= y_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 &= y_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 &= y_3 \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 &= y_4 \end{aligned}\right\} . \quad (1)$$

Writing in matrix form, Equation 1 becomes:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = A \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} . \quad (2)$$

In rewriting Equation 2 and expressing x_i in terms of y_i:

$$A^{-1} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} , \quad (3)$$

where A^{-1} is the inverse of Matrix A.

One way of accomplishing this inversion is to exchange x_i with y_i in Equation 2 step by step and modify Matrix A accordingly.

The following diagram shows the exchange process. After each exchange, a new (modified) matrix is obtained denoted by A', A", etc.

$$A \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \rightarrow A' \begin{bmatrix} \textcircled{y_1} \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \textcircled{x_1} \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \rightarrow A'' \begin{bmatrix} y_1 \\ \textcircled{y_2} \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_1 \\ \textcircled{x_2} \\ y_3 \\ y_4 \end{bmatrix} \quad (4)$$

$$\rightarrow A''' \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ y_4 \end{bmatrix} \rightarrow A'''' \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Compared to Equation 3, A"" is seen as the inverse of A. This "exchanging method" is really the Gauss-Jordan method stated a different way. The exchange process does not have to be performed in the sequence shown in Equation 4. Any x_i(i=1,2,..) may be paired with any y_j(j=1,2,..) and executed in any order, but no x_i or y_i may belong to more than one pair. Each time a pair, (x_i,y_j) is chosen, there is a corresponding matrix element a_{ij} which is used as a divisor to divide the whole column j; a_{ij} is called the pivoting element. Care must be taken in choosing the a_{ij}, because a small (or zero) value acting as the divisor will produce large errors or meaningless results. Therefore, properly chosen pivoting elements will minimize any error within the matrix inversion process.

A quick examination of how the matrixes in Equation 4 are to be modified for each pair of exchanges reveals the inherent parallel property, and also suggests the parallel algorithm for implementation on the STARAN.

By rewriting Equation 1, then:

x_1	x_2	x_3	x_4	
a_{11}	a_{12}	a_{13}	a_{14}	y_1
a_{21}	a_{22}	a_{23}	a_{24}	y_2
a_{31}	a_{32}	a_{33}	a_{34}	y_3
a_{41}	a_{42}	a_{43}	a_{44}	y_4

This form will show the exchange process more clearly. Assume a_{11} is the largest element among the a_{ij}, so a_{11} is chosen as the pivoting point, and the corresponding pair of (x_i,y_j) to be exchanged is (x_1,y_1). The result of this exchange is shown in Equation 6.

	Exchanged			
(y_1)	x_2	x_3	x_4	
$\frac{1}{a_{11}}$	$0 - \frac{1}{a_{11}} a_{12}$	$0 - \frac{1}{a_{11}} a_{13}$	$0 - \frac{1}{a_{11}} a_{14}$	(x_1)
$\frac{a_{21}}{a_{11}}$	$a_{22} - \frac{a_{21}}{a_{11}} a_{12}$	$a_{23} - \frac{a_{21}}{a_{11}} a_{13}$	$a_{24} - \frac{a_{21}}{a_{11}} a_{14}$	y_2
$\frac{a_{31}}{a_{11}}$	$a_{32} - \frac{a_{31}}{a_{11}} a_{12}$	$a_{33} - \frac{a_{31}}{a_{11}} a_{13}$	$a_{34} - \frac{a_{31}}{a_{11}} a_{14}$	y_3
$\frac{a_{41}}{a_{11}}$	$a_{42} - \frac{a_{41}}{a_{11}} a_{12}$	$a_{43} - \frac{a_{41}}{a_{11}} a_{13}$	$a_{44} - \frac{a_{41}}{a_{11}} a_{14}$	y_4

(6)

The modified matrix related to Equation 6 may be formulated by rearranging the first equation in Equation 1; namely:

$$a_{11} x_1 + a_{12} x_2 + a_{13} x_3 + a_{14} x_4 = y_1$$

becomes (7)

$$\frac{1}{a_{11}} y_1 - \frac{a_{12}}{a_{11}} x_2 - \frac{a_{13}}{a_{11}} x_3 - \frac{a_{14}}{a_{11}} x_4 = x_1 .$$

The coefficients in Equation 7 are the first row of the matrix shown in Equation 6. By substituting Equation 7 into the second equation within Equation 1, the result is:

$$a_{21} \{\frac{1}{a_{11}} y_1 - \frac{a_{12}}{a_{11}} x_2 - \frac{a_{13}}{a_{11}} x_3 - \frac{a_{14}}{a_{11}} x_4\} + a_{22}x_2 +$$

$$a_{23}x_3 + a_{24}x_4 = y_2 ;$$

that is,

$$\frac{a_{21}}{a_{11}} y_1 + (a_{22} - \frac{a_{12}}{a_{11}} a_{21}) x_2 + (a_{23} - \frac{a_{13}}{a_{11}} a_{21}) x_3 +$$

$$(a_{24} - \frac{a_{14}}{a_{11}} a_{21}) x_4 = y_2 \qquad (8)$$

Again, the coefficients in Equation 8 correspond to the second row of the matrix in Equation 6. The other rows may be derived in the same way.

The general rules for exchanging one pair (x_i, y_j) may be summed up as follows:

1. Select a pivoting element a_{ij}

2. Replace the location of a_{ij} with 1 and use a_{ij} as a divisor to divide the whole column j, which contains that pivoting element.
3. Replace the location of a_{ik} with 0 and perform
   (column k) = (column k) - (column j)$\cdot a_{ik}$
   for all $k \neq j$.

A total of N pairs of exchanges (x_i, y_j) are needed to complete the inversion process for a matrix of N x N elements. Interchanging the columns and the corresponding rows required pivoting elements along the diagonal. These elements are required to maintain the proper order for the matrix. One column at a time operates in parallel, and with some modifications, a complete parallel operation for all column is possible if enough processing elements within the parallel processor are available.

A description is given below of the operations required in the STARAN approach to matrix inversion for $N \leq 32$ utilizing the 'one pair exchange' technique mentioned earlier.

The sequence of operations required to invert a 4x4 matrix is given in Table 4-XI. Figure 4-8 and 4-9 illustrate these operations:

TABLE 4-XI - SEQUENCE OF OPERATIONS

Sequence operation	Figure
1. Select pivoting element a_{ij} (a_{11})	4-8A
2. Replace a_{ij} (a_{11}) with a 1	4-8B
3. F2 = F1/a_{ij}; Column j only (i=j=1)	4-8B
4. Expand Column j in F2 to all words	4-8B
5. Write F3 by bring row i (i=1)	4-8C
6. Replace Row i with 0; also Column j is set to 0	4-8C
7. F1 = F1-F2$\cdot$F3	4-8D

Field F1 in Figure 4-8D corresponds to the matrix in Equation 6. The operations involved in STARAN for a pair of exchanges are thus:

1. Move data masked from field to field
2. Expand elements within a field

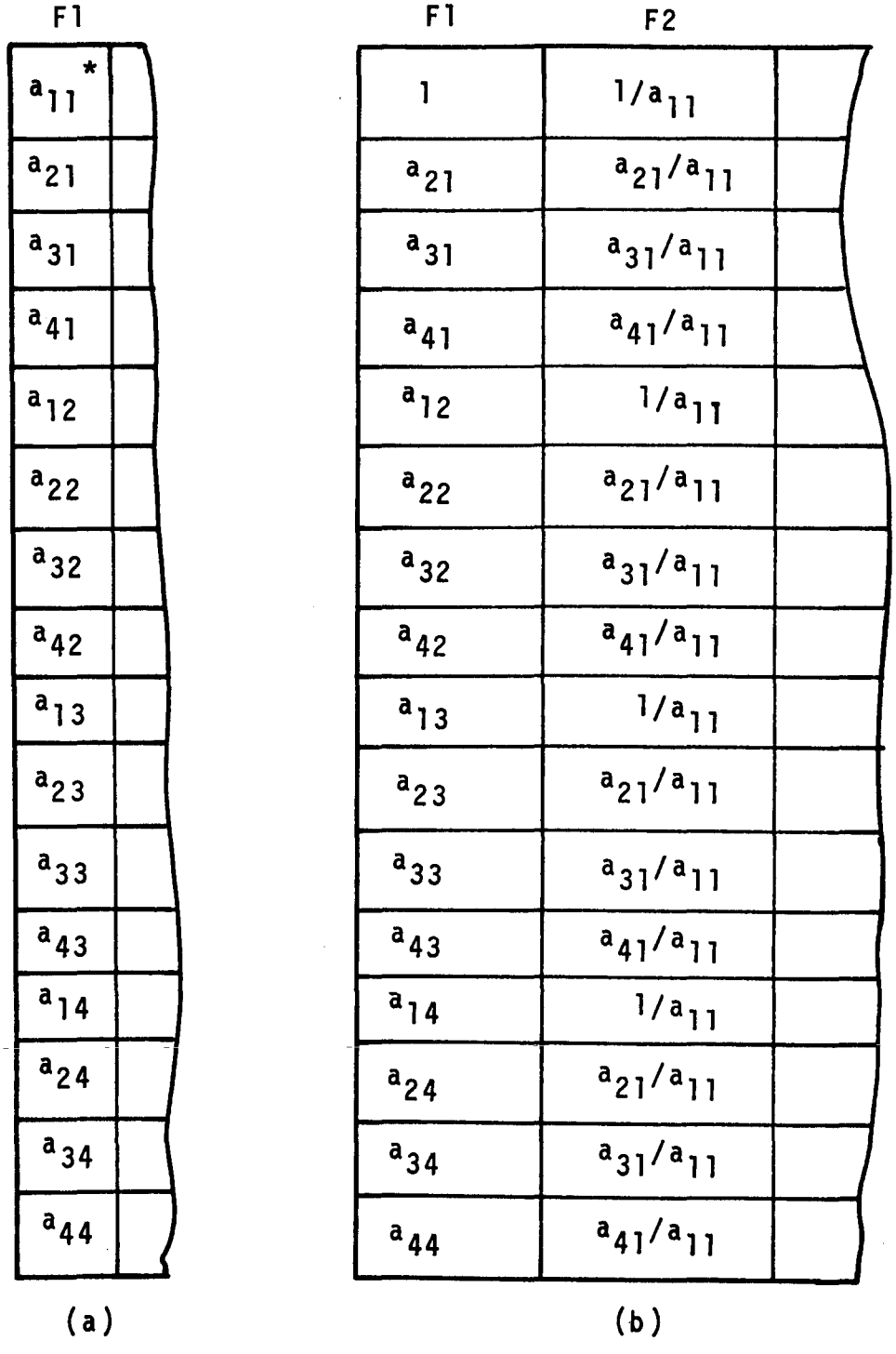


Figure 4-8 - Array Maps for Matrix Inversion (Sheet 1 of 2)

F1	F2	F3
0	$1/a_{11}$	-1
0	a_{21}/a_{11}	-1
0	a_{31}/a_{11}	-1
0	a_{41}/a_{11}	-1
0	$1/a_{11}$	a_{12}
a_{22}	a_{21}/a_{11}	a_{12}
a_{32}	a_{31}/a_{11}	a_{12}
a_{42}	a_{41}/a_{11}	a_{12}
0	$1/a_{11}$	a_{13}
a_{23}	a_{21}/a_{11}	a_{13}
a_{33}	a_{31}/a_{11}	a_{13}
a_{43}	a_{41}/a_{11}	a_{13}
0	$1/a_{11}$	a_{14}
a_{24}	a_{21}/a_{11}	a_{14}
a_{34}	a_{31}/a_{11}	a_{14}
a_{44}	a_{41}/a_{11}	a_{14}

(a)

F1	F2	F3
$1/a_{11}$	$1/a_{11}$	-1
a_{21}/a_{11}	a_{21}/a_{11}	-1
a_{31}/a_{11}	a_{21}/a_{11}	-1
a_{41}/a_{11}	a_{21}/a_{11}	-1
$0 - \frac{1}{a_{11}} a_{12}$	$1/a_{11}$	a_{12}
$a_{22} - \frac{a_{21}}{a_{11}} a_{12}$	a_{21}/a_{11}	a_{12}
$a_{32} - \frac{a_{31}}{a_{11}} a_{12}$	a_{31}/a_{11}	a_{12}
$a_{42} - \frac{a_{41}}{a_{11}} a_{12}$	a_{41}/a_{11}	a_{12}
$0 - \frac{1}{a_{11}} a_{13}$	$1/a_{11}$	a_{13}
$a_{23} - \frac{a_{21}}{a_{11}} a_{13}$	a_{21}/a_{11}	a_{13}
$a_{33} - \frac{a_{31}}{a_{11}} a_{13}$	a_{31}/a_{11}	a_{13}
$a_{43} - \frac{a_{41}}{a_{11}} a_{13}$	a_{41}/a_{11}	a_{13}
$0 - \frac{1}{a_{11}} a_{14}$	$1/a_{11}$	a_{14}
$a_{24} - \frac{a_{21}}{a_{11}} a_{14}$	a_{21}/a_{11}	a_{14}
$a_{34} - \frac{a_{31}}{a_{11}} a_{14}$	a_{31}/a_{11}	a_{14}
$a_{44} - \frac{a_{41}}{a_{11}} a_{14}$	a_{41}/a_{11}	a_{14}

(b)

Figure 4-8 - Array Maps for Matrix Inversion (Sheet 2 of 2)

3. One divide
4. One multiply
5. One subtract

The first two operations may be done effectively in STARAN because an AAP has a very flexible and extensive communication ability. The major operations are the last three arithmetic functions. In this respect, the total major operations to exchange all pairs (x_i, y_j), that is, to invert a matrix of N by N is:

$$N\ (\oplus\!\!\div,\ \otimes,\ \oplus)\ .$$

As mentioned before a properly chosen pivoting element for each pair of exchanges is important for obtaining a meaningful matrix inversion (or solutions of a set of linear equations). In a sequential computer, this choice presents a big problem. In general, it requires searching through matrix elements on the order of N^2 times and interchanging two rows and two columns. In STARAN, this operation is very simple. The operation requires searching the field containing the matrix elements (F1 in Figure 4-8) for the word that has the largest magnitude; those columns and rows that have already been exchanged are excluded from this search. In a floating-point number representation, a search on the exponent field meets this requirement; therefore, this is not classified as a major operation.

An index field attached to each word to keep track of the order that the pivoting elements are chosen will suffice for reordering the inverted matrix if necessary.

The major operations required by the types of computers to perform the matrix inversion ($N \leq 32$) are shown below:

1. STARAN S-1000.

   $$n\ \otimes + n\ \oplus\!\!\div + n\ \ominus$$

2. Sequential computer[a]

   $$\frac{n^3}{3}\ \otimes + \frac{n^2}{2}\ \oplus\!\!\div + \frac{n^3}{3}\ \ominus$$

[a] These are the approximate number of operations required for the Gauss method when n is large.

In referring to Step 1 above, a 32x32 matrix inversion on a STARAN S-100 takes the following time:

$$32(3.59 \times 10^{-3}) + 32(3.11 \times 10^{-3}) + 32(816 \times 10^{-6}) = 240.5 \text{ msec},$$

where:

32	=	number of rows and columns of the matrix,
3.59 msec	=	double-precision floating-point (dpfb) multiplication time,
3.11 msec	=	dpfb division time, and
0.816 msec	=	dpfb subtraction time.

4.2.3.4.3 Matrix Inversion (N>32) - The basic matrix inversion approach used in a STARAN S-1000 for a matrix larger than 32x32 is discussed below.

If a matrix of N by N elements is to be inverted (where N>32), then not all of the N^2 elements can be stored in one STARAN S-1000 array memory. Several methods may be used to solve this problem. The following method is well suited to the STARAN S-1000. The approach is to partition the matrix of N x N (N>32) elements into submatrixes α_{ij}; each is a matrix of 32 x 32 or less. Each submatrix can be accommodated in 1024 STARAN array words. For example, Matrix A_{90x90} may be divided into:

$$A_{90x90} = \begin{bmatrix} [\alpha_{11}]_{32x32} & [\alpha_{12}]_{32x32} & [\alpha_{23}]_{32x26} \\ [\alpha_{21}]_{32x32} & [\alpha_{22}]_{32x32} & [\alpha_{23}]_{32x26} \\ [\alpha_{31}]_{26x32} & [\alpha_{32}]_{26x32} & [\alpha_{33}]_{26x26} \end{bmatrix}.$$

If each submatrix is considered as a unit, then the idea used before may be applied and a matrix equation similar to Equation 2 can be written:

$$A \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix},$$

where x_i, y_i are column vectors of 32 elements or less.

By exchanging x_1 with y_1, x_2 with y_2, and x_3 with y_3, and modifying α_{ij} accordingly, just as Equation 4, the result is A^{-1}.

The following diagram shows the exchange of x_1 with y_1; notice the similarity with Equations 5 and 6.

(x_1)	x_2	x_3	
α_{11}	α_{12}	α_{13}	y_1
α_{21}	α_{22}	α_{23}	y_2
α_{31}	α_{32}	α_{33}	y_3

(y_1)	x_2	x_3	
α_{11}^{-1}	$-\alpha_{11}^{-1}\alpha_{12}$	$-\alpha_{11}^{-1}\alpha_{13}$	(x_1)
$\alpha_{21}\alpha_{11}^{-1}$	$\alpha_{22}-\alpha_{21}\alpha_{11}^{-1}\alpha_{12}$	$\alpha_{23}-\alpha_{21}\alpha_{11}^{-1}\alpha_{13}$	y_2
$\alpha_{31}\alpha_{11}^{-1}$	$\alpha_{32}-\alpha_{31}\alpha_{11}^{-1}\alpha_{12}$	$\alpha_{33}-\alpha_{31}\alpha_{11}^{-1}\alpha_{13}$	y_3

Similar operations apply to $x_2 \leftrightarrow y_2$, $x_3 \leftrightarrow y_3$. In general, a matrix of N x N elements (N>32) may be divided into $[\frac{N}{32}]^2$ submatrixes:

$$\alpha_{ij} \qquad i = 1, 2 \ldots [\tfrac{N}{32}]$$

$$j = 1, 2 \ldots [\tfrac{N}{32}]$$

and $[\frac{N}{32}]$ pairs of exchanging $x_i \leftrightarrow y_i$ are required. In each exchange, the following steps are necessary:

1. Invert a submatrix of 32x32 (or less), this requires (32 ⊗ , 32 ÷, 32 ⊖) (or less)
2. $[\frac{N}{32}]^2 - 1$ pairs of submatrix multiplications
   Each pair requires 32 ⊗ and $32 \cdot \log_2 32 = 160$ ⊕
3. $([\frac{N}{32}] - 1)^2$ pairs of submatrix subtractions.
   One subtract operation for each pair (see matrix addition and subtraction section).
4. $\{[\frac{N}{32}]^2 + 2([\frac{N}{32}] - 1)^2\}$ loads of the submatrixes α_{ij} to the STARAN array memory, each I/O load accommodates 1024 elements.

There are $[\frac{N}{32}]$ pairs of exchanges performed. The total major operations including I/O therefore are:

$$\{32(\div) + 32n^2\,(\times) + (32 + (n-1)^2)\,(-) + 160\,(n^2-1)\,(+) + (3n^2 - 4n + 1)\,(I/O)\}\ n\ ,$$

where

$n = [\frac{N}{32}]$, and

(I/O) = input and output of 1024 elements.

Notice that the total number of multiplications is $32n^3$; that is:

$$32\,[\frac{N}{32}]^3 \sim \frac{1}{1024}\,N^3$$

This result is expected. Normally, the operations required to invert a matrix of N by N elements is proportioned to N^3, and the STARAN S-1000 can perform 1024 operations simultaneously.

If the matrix to be inverted is a sparse matrix, the number of operations can be significantly reduced. This fact is also true in a sequential computer.

The array map layouts and manipulation of the data within the arrays for the partitional submatrixes is the same as previously discussed for inverting a matrix of NxN elements where $N \leq 32$.

Table 4-XII shows the estimated times for the major operations for different size matrix inversions. Double-precision floating-point arithmetic is assumed.

TABLE 4-XII - MATRIX INVERSION TIMES - BASIC APPROACH

Item	Estimated time for major operations	Estimated I/O time	Total Time
A_{11}^{-1} x 11	88 msec	1 msec	89 msec
A_{32}^{-1} x 32	256 msec	8 msec	264 msec
A_{128}^{-1} x 128	15.8 sec	1.6 sec	17.4 sec
A_{512}^{-1} x 512	16.6 min	2.5 min	19.1 min

4.2.3.4.4 Matrix Inversion (Alternate Approach for N>32) - Another approach to invert a matrix (N x N) with N>32 is a straightforward extension of the method used for N≤32.

Instead of partitioning the matrix, the whole matrix is worked on utilizing a virtual memory concept. For each pass, up to 1024 elements are processed until all N^2 elements have been operated on. The major operations, including I/O are:

$$\{[\frac{N^2}{1024}] \cdot \text{(I/O)} \ , \ [\frac{N}{1024}] \ \text{(÷)} \ , \ [\frac{N^2}{1024}] \ .(\text{(×)} \ , \ \text{(+)})\} \ N.$$

This method has approximately the same number of multiplications and divisions, less additions and subtractions, but more I/O. Unless adequate I/O capabilities are available, this method should not be used.

Table 4-XIII lists the estimated time using this approach for purpose of comparison with the previous method (see Table 4-XII).

TABLE 4-XIII - MATRIX INVERSION TIMES - ALTERNATE APPROACH

Matrix size	Major operations	Time for operations	I/O time	Total
$A^{-1}_{128 \times 128}$	128 ÷ 128·16 × 128·16 +	9·5 sec	16.7 sec	26.2 sec
$A^{-1}_{512 \times 512}$	512 ÷ 256·512 × 256·512 +	9.7 min	17.9 min	27.6 min

A comparison of the times required by the two matrix inversion methods for N>32 shows:

1. The basic approach requires less I/O time.
2. The alternate method requires less computation time.
3. In total, the basic approach is faster than the alternate method.

Based upon these observations, the basic approach, is preferable if a 32-bit I/O channel is used in the STARAN. However if a parallel input/output (256-bit) channel is used, then I/O time will be greatly reduced and the alternate method discussed would be faster.

4.2.4 General SMAC Functions

4.2.4.1 Part I - Data Preparation Program

The Data Preparation Program prepares and arranges the working data tape to be used in programs described below. Certain constants to be used in the different programs are prepared and organized in correct locations in the data tape. This initial setup phase is sequential in nature; the parallel processor therefore offers no improvement in this task. In both sequential and parallel processors, this phase does not represent a significant processing time. Therefore this phase was not given detailed consideration.

4.2.4.2 Part II - General Catalog Search Program

At least three well distributed reference stars in each exposure must be manually identified to compute the preliminary orientation of the exposures. These stars are manually identified by their BOSS catalog numbers. The BOSS catalog contains approximately 33,000 of the brightest stars. This program searches the BOSS catalog for the BOSS number and the cataloged position and then updates the cataloged position of the stars to the time the exposure was taken. A coordinate conversion then changes the right ascension α and declination δ (updated versions) to true zenith distance ζ_t and azimuth Z. The major functions that constitute the General Catalog Search Program are illustrated in the functional flow diagram in Figure 4-9. Algorithms[a] for the two functions designated II-3 and II-4 are provided in Appendix C. Table 4-XIV is a summary of the estimated STARAN S-1000 computation times required by each function of the General Catalog Search program for the referenced problem. The referenced problem has 3 cameras,

[a]The algorithms were obtained or derived from the following reference: Gyer, M.S.; Haag, N. N. ; and Llewellyn, S. K.: Documentation for the Multi-camera Stellar SMAC Computer Program. Fort Belvoir, Virginia, U.S. Army Engineer Topographic Laboratories. June 1970.

12 exposures, and 100 stars per exposure with 4 stars per exposure used for the initial orientation. Included in Table 4-XIV are the array memory requirements, the number of arithmetic operations required to implement each algorithm, and the number of algorithm executions needed by each function for the referenced problem. The array memory allocation is the number of associative array words simultaneously required to execute the algorithms. When the array memory requirements exceed that available in a STARAN S-1000 (1024 words), then multiple algorithm executions are required as indicated in Table 4-XIV.

The standard STARAN computation times were calculated using the number of arithmetic operations per algorithm execution, the number of algorithm executions, and the STARAN performance data for double-precision floating-point arithmetic. The STARAN performance data are given in Appendix E. The computation times shown for the system with hardware arithmetic were calculated using the performance given in Appendix F, which briefly describes this hypothetical hardware arithmetic option.

In this program only the reference stars are processed. While the STARAN S-1000 can process up to 128 stars simultaneously, (see Table 4-XIV, References II-3 and II-4), there are only 48 reference stars (4 stars/exposure, 12 exposures) to be processed for the reference problem. Therefore STARAN's potential is not fully utilized in this program for the referenced problem. The BOSS catalog of stars are searched for the 48 reference stars. STARAN with a PHD can perform search operations very effectively. Since the 1108 execution time could only be determined for the sum total of the preliminary programs, there is no visibility into the execution time required for the basic functions. Therefore it is not possible to draw an explicit conclusion about the significance of STARAN's advantage over the Univac 1108 for this particular search task.

The estimated execution time in STARAN is derived from the arithmetic operations per algorithm execution, and the number of times that the algorithm is executed. For example, in Table I, Reference II-3:

$$T = [55 \ \circledast \ + \ 32 \ \oplus \ ] \cdot (1);$$

TABLE 4-XIV - STARAN TIMING SUMMARY FOR GENERAL CATALOG SEARCH PROGRAM

Function*		Array Memory allocation+	Arithmetic operations per algorithm execution	Algorithm executions	STARAN S-1000≠ computation time	
Ref. no.	Description				Standard system	Hardware arithmetic§
II-1	Manually identifying stars	-	-	-	-	-
II-2	Search BOSS catalog for the reference stars $(\alpha,\delta)_{1950}$	66,000 words (33,000 stars, 2 words/star)	48 searches (48 reference stars)	258	68	68
II-3	Update $(\alpha,\delta)_{upD} \leftarrow (\alpha,\delta)_{950}$	384 words (8 words/star, 48 reference stars)	55 ⊗ 32 ⊕	1	224	18
II-4	coordinate conversion $(\zeta_t, z) \leftarrow (\alpha, \delta)_{upD}$	384 words (8 words/star, 48 reference stars)	18 ⊗ 15 ⊕	1	77	6.3
		Total			369	92

*Algorithms are given in Appendix C.

+The referenced problem has: 3 cameras, 12 exposures, 100 stars per exposure, with 4 stars per exposure used for initial orientation.

≠Execution time calculated for double-precision floating-point arithmetic.

§A hypothetical hardware arithmetic option is described in Appendix F.

⊗ = multiplies or divides

⊕ = adds or subtracts

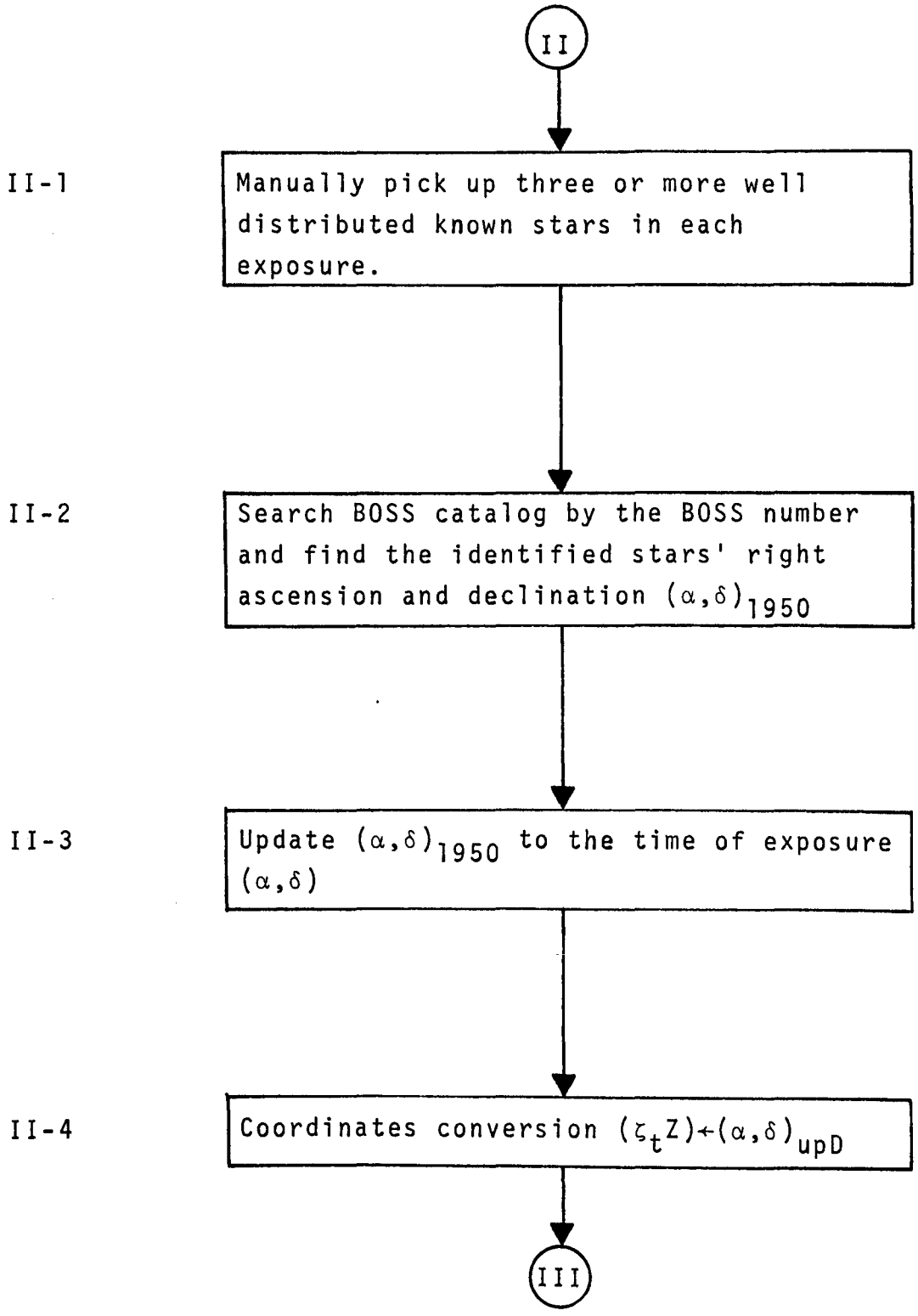


Figure 4-9 - Functional Flow Diagram of General Catalog Search Program

$$T_{standard} = [(55 \times 3600) + (32 \times 820)](1)\ \mu sec = 224\ msec,$$

where:

55 = multiplies,

3600 μsec = time per double-precision floating-point multiply (see Appendix E),

32 = adds,

820 μsec = time per double-precision floating-point add (see Appendix E), and

1 = algorithm execution required

4.2.4.3 Part III - Preliminary orientation Computation Program

In order to correlate the plate measurements with stellar positions, the orientation of the camera must be known with a good degree of accuracy. This preliminary orientation computation program accepts data from the general catalog search program and uses the stellar positions of the manually defined stars and their corresponding measurements from the exposure to compute an approximation of the absolute orientation of the cameras. In doing this preliminary orientation computation, the corrections, other than refraction and principal point, are neglected.

This program is divided into two parts. First, find the approximate preliminary orientations by using the direction cosine of the identified stars in the image space and their corresponding direction cosines computed from the tape in the object space. Second, use the output from the first stage and refine the results further by utilizing a more rigorous least squares adjustment of the photo coordinates of the reference stars. Several adjustments (iterations) are normally required.

The flow diagram shown in Figure 4-10 illustrates the major functions that make up the preliminary orientation computation program. Table 4- XV is a summary of the estimated STARAN S-1000 computation times required by the referenced problem for each function of the preliminary orientation computation program. The significance of the various columns in Table 4-XV are the same as that described for the Table 4-XIV in the preceeding discussion (general catalog search program).

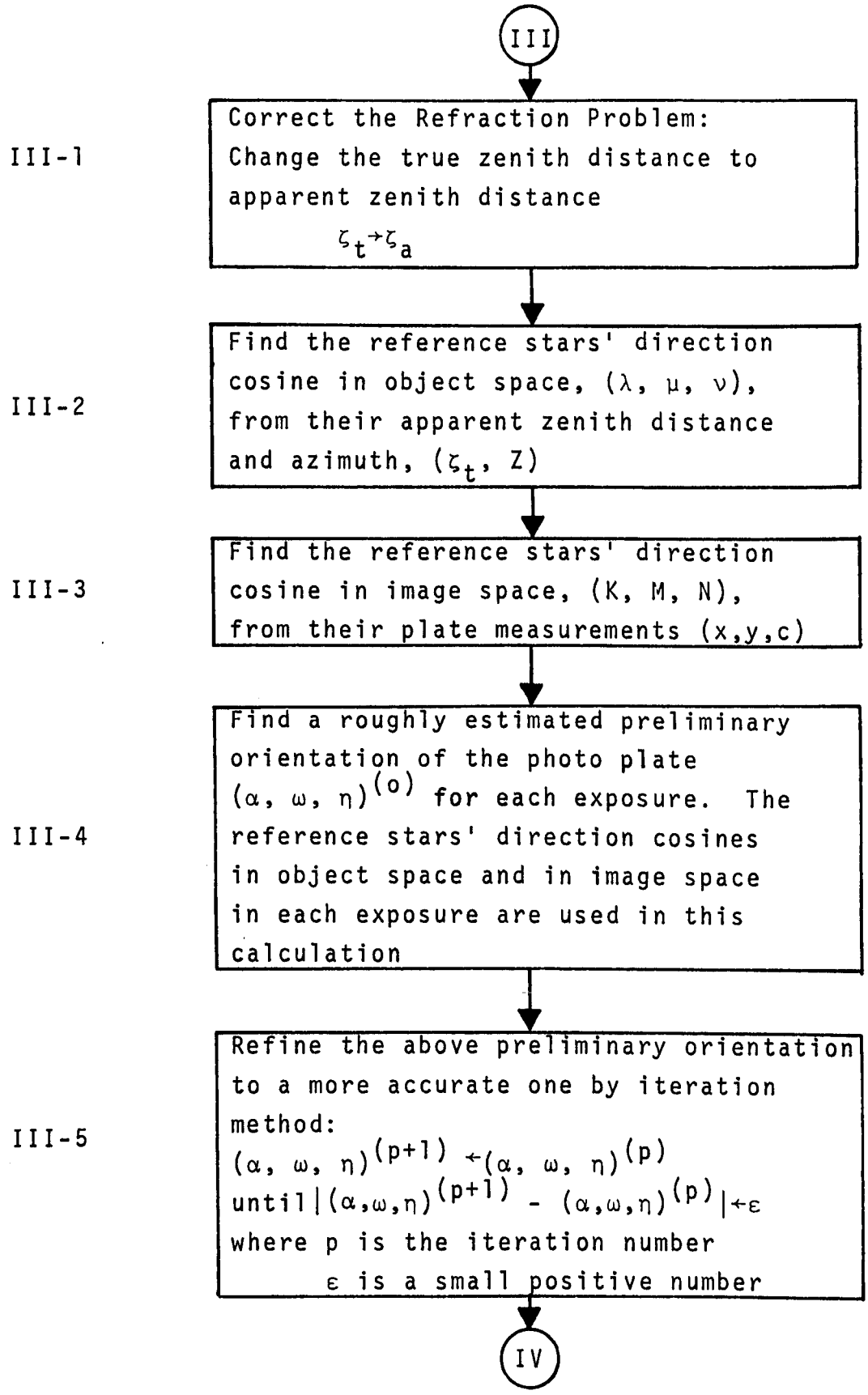


Figure 4-10 - Preliminary Orientation Computation Program - Functional Flow Diagram

TABLE 4-XV - STARAN TIMING SUMMARY FOR PRELIMINARY ORIENTATION COMPUTATION PROGRAM

Function*		Array memory Allocation+	Arithmetic operations per algorithm execution	Algorithm executions	STARAN S-1000≠ computation time	
Ref. no.	Description				Standard system (sec)	Hardware arithmetic§ (msec)
III-1	Correct refraction problem $\zeta_a \leftarrow \zeta_t$	96 words (2 words/star, 48 reference stars)	19 p ⊗ 16 p ⊕	1	0.81	67
III-2	find direction cosine in object space $(\lambda,\mu,\nu) \leftarrow (\zeta_a, Z)$	144 words (3 words/star, 48 reference stars)	10 ⊗ 10 ⊕	1	0.044	4
III-3	find direction cosine in image space $(K,L,M) \leftarrow (x\ y,c)$	144 words (3 words/star, 48 reference stars)	3 ⊗ 3 ⊕	1	0.013	1.1
III-4	$\begin{pmatrix}\alpha\\ \omega\\ \eta\end{pmatrix}^{(o)} \leftarrow \left\{\begin{pmatrix}\lambda\\ \mu\\ \nu\end{pmatrix}, \begin{pmatrix}K\\ L\\ M\end{pmatrix}\right\}$	384 words (32 words/exp., 12 exposures)	34 ⊗ 44 ⊕	1	0.15	14.5
III-5	refining solutions $(\alpha\omega\eta)^{(p+1)} \leftarrow (\alpha\omega\eta)^{(p)}$ until $\lvert(\alpha\omega\eta)^{(p+1)} - (\alpha\omega\eta)^{(p)}\rvert < \varepsilon$	648 words (54 words/exp., 12 exposures)	37 ⊗ 41 ⊕	1	1.67	150
		Total			2.69	0.24

*Algorithms are given in Appendix C.

+The referenced problem has 3 cameras, 12 exposures, and 100 stars per exposure, with 4 stars per exposure used for initial orientation.

≠Execution time calculated for double-precision floating-point arithmetic.

§A hypothetical hardware arithmetic option is described in Appendix F.

⊗ = multiplies or divides ⊕ Add or subtracts p = iterations (10 assumed)

Since this program processes only the reference stars, just part of the 1024 array words are used as may be seen from the "array memory allocation" column in Table 4-XV. Therefore, for this reference problem (12 exposures), STARAN's full potential is not being utilized.

4.2.4.4 Part IV - Stellar Search Program

The stellar search program, using the plate measurements and the initial camera orientations, computes stellar coordinates on the celestial sphere. These coordinates are matched with the location of stars cataloged by the Smithsonion Astrophysical Observatory (SAO) and contained on magnetic tape. Once a successful comparison has been made, the apparent zenith distance and azimuth of the cataloged star is associated with the plate measurement.

A flow diagram showing the major functions that make up the Stellar Search Program is shown in Figure 4-11. For the referenced problem, Table 4-XVI summarizes the STARAN S-1000 computation times estimated for each function of the stellar search program. The items in Table 4-XIV have the same significance as described for those in Table 4-XIV, (general catalog search program).

STARAN's parallel processing ability is particularly attractive here. The same algorithm applies for all stars in: star location updating, coordinate translation, refraction correction to the star's apparent zenith distance, and the star matching process (stars in a plate matched to the updated SAO star location). Since each plate of the referenced problem contains approximately 100 stars, one or more plates may be processed concurrently through the previous functions.

In this program where all stars of the exposures are involved, STARAN would be fully utilized as indicated by the array memory allocation and algorithm execution columns of Table 4-XIV. In addition to full utilization, STARAN's exceptional search capabilities would be utilized in subparts IV-4 and IV-6. The STARAN search operations are described in Section 4.2.2.

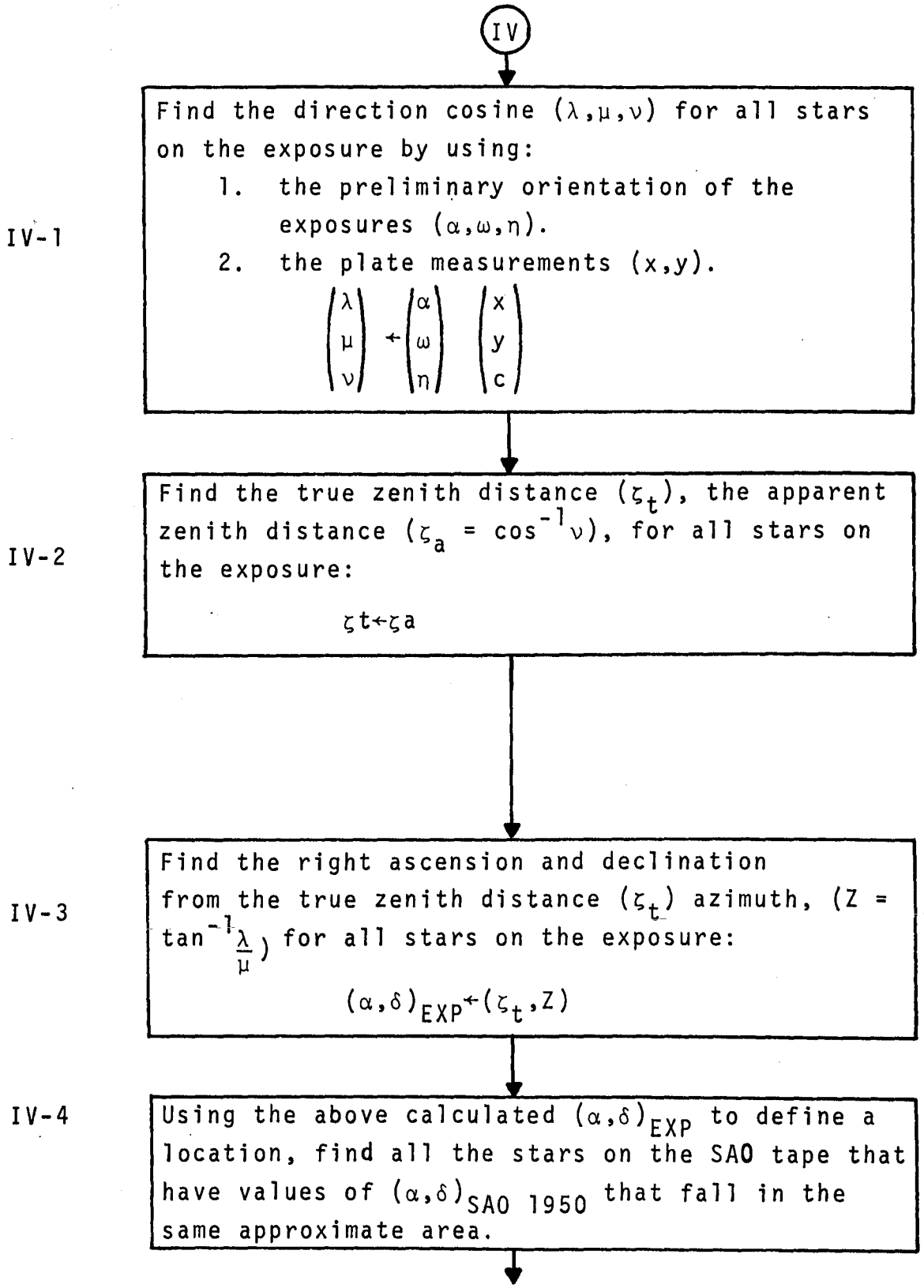


Figure 4-11 - Flow Diagram of Stellar Search Program Functions (Sheet 1 of 2)

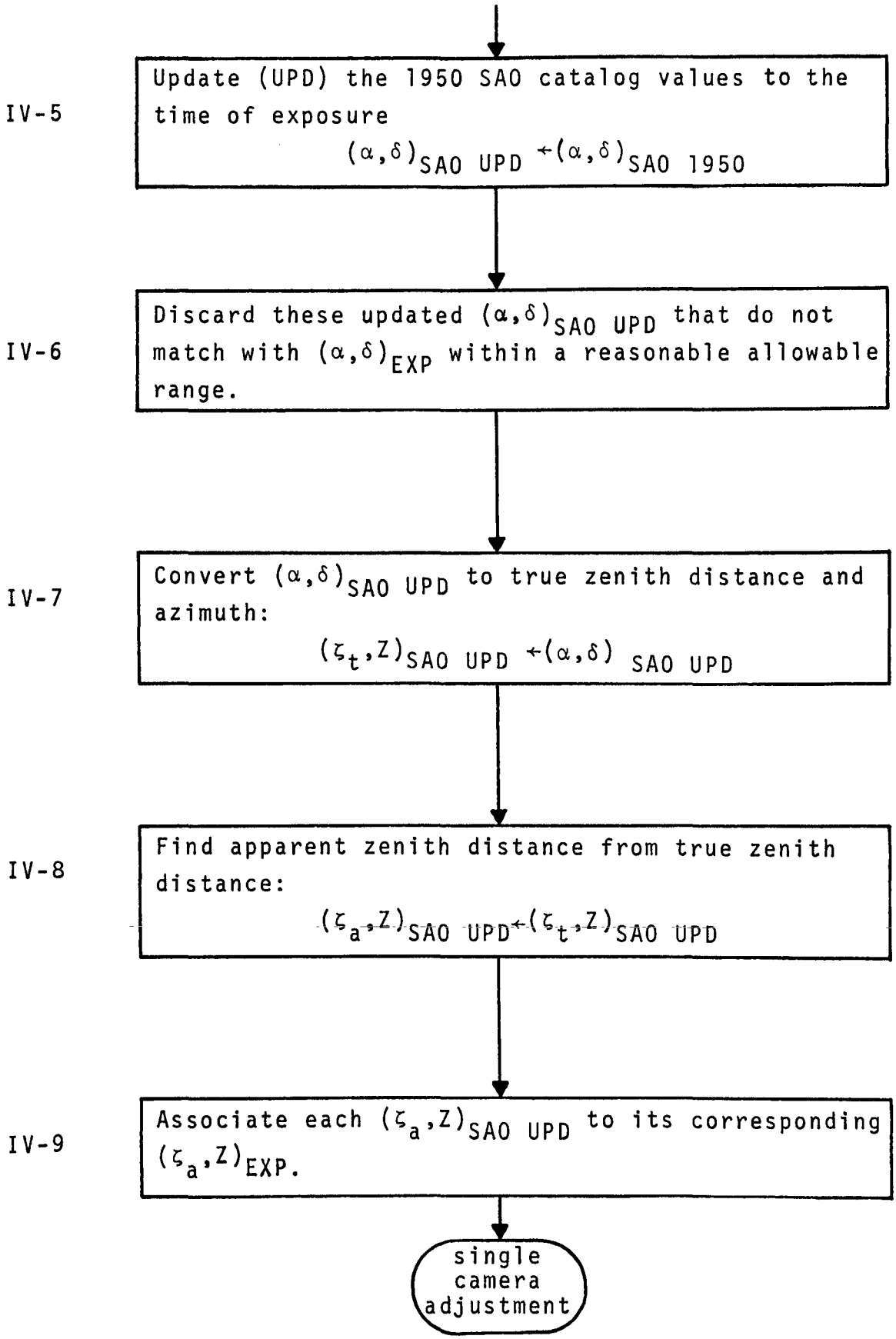


Figure 4-11 - Flow Diagram of Stellar Search Program Functions (Sheet 2 of 2)

TABLE 4-XVI - STARAN TIMING SUMMARY FOR STELLAR SEARCH PROGRAM

Function*		Array memory allocation+	Arithmetic operations per algorithm execution	Algorithm executions	STARAN S-1000 computation time≠	
Ref. no.	Description				Standard system (sec)	Hardware arithmetic§ (sec)
IV-1	Find direction cosines for all stars $\begin{pmatrix}\lambda\\ \mu\\ \nu\end{pmatrix} \leftarrow \left\{\begin{pmatrix}\alpha\\ \omega\\ \eta\end{pmatrix}, \begin{pmatrix}x\\ y\\ c\end{pmatrix}\right\}$	12,000 words (10 words/star, 100 stars/exp., 12 exposures)	4 ⊗ 4 ⊕	12	0.21	0.02
IV-2	Correct refraction $\zeta_a = \cos^{-1}\nu$ $\zeta_t \leftarrow \zeta_a$	2400 words (2 words/star, 100 stars/exp., 12 exposures)	15 ⊗ 15 ⊕	3	0.2	0.015
IV-3	Coordinate conversion $Z = \tan^{-1}\frac{\nu}{\mu}$ $(\alpha,\delta)_{EXP} \leftarrow (\zeta_t, Z)$	4800 words (4 words/star, 100 stars/exp., 12 exposures)	37 ⊗ 30 ⊕	6 (2 exp each time)	0.94	0.084
IV-4	Find stars on SAO whose $(\alpha,\delta)_{SAO\ 1950}$ lie within certain range	160,000 words (all stars in SAO catalog)	2 between limits searches per exposure (total 12 exposures)	3480 320/exp. 12 exp.)	1.22	1.22
IV-5	Star update $(\alpha,\delta)_{SAO\ UPD} \leftarrow (\alpha, \delta)_{SAO\ 1950}$	38,400 words (8 words/star, about 4800 stars updated)	55 ⊗ 32 ⊕	40	8.97	0.67

TABLE 4-XVI - STARAN TIMING SUMMARY FOR STELLAR SEARCH PROGRAM (Continued)

Function*		Array memory allocation+	Arithmetic operations per algorithm execution	Algorithm executions	STARAN S-1000≠ computation time	
Ref. no.	Description				Standard system (sec)	Hardware Arithmetic§ (sec)
IV-6	Discard $\|(\alpha,\delta)_{EXP} \leftarrow (\alpha,\delta)_{SAO\ 1950}\| > \varepsilon$	400 words/plate, 12 plates	100 between limits search	12	0.08	0.08
IV-7	Coordinates conversion $(\zeta_t, Z)_{SAO\ UPD} \leftarrow (\alpha,\delta)_{SAO\ UPD}$	9600 words (8 words/star, 1200 stars)	18 ⊗ 15 ⊕	12 (1 exp. each time)	0.9	0.08
IV-8	$\zeta_a \leftarrow \zeta_t$	2400 words (2 words/star, 1200 stars)	19p ⊗ 16p ⊕ (max p = 10)	3	2.4	0.21
IV-9	Associates the calculated and the measured (ζ_a, Z)	1200 words (1 word/star)	1200 between limits search	2	0.02	0.02
			Total		14.9	2.3

*Algorithms are given in Appendix C.

+The referenced problem has 3 cameras, 12 exposures, and 100 stars per exposure with 4 stars per exposure used for initial orientation.

≠Execution time calculated for double-precision floating-point arithmetic.

§A hypothetical hardware arithmetic option is described in Appendix F.

⊗ = multiplies or divides ⊕ adds or subtracts p = iterations (10 assumed)

4.2.4.5 Part V - Single Camera Adjustment Program - The single camera adjustment program performs three main functions: (1) it determines precise estimates of the camera parameters and absolute orientation angles prior to the multicamera adjustment, (2) it edits the measurement data, and (3) it tests the significance (statistical) of the radial and decentering distortions.

The initial assumption was that the main computations in this program involve matrix inversion that are discussed in Section 4.2.3. Study results indicate, however, that this assumption may be erroneous. Without a detailed breakdown of the Univac 1108 execution times, a specific conclusion cannot be made.

4.2.4.6 Multiple Camera Reduction Program - The multiple camera reduction program simultaneously determines the camera calibration parameters, relative orientation angles, and refraction parameters for multicamera arrays of up to five cameras. Essentially, the computations involved in this program are similar to the single camera adjustment program in that it involves matrix inversions. Only the matrix inversion function was considered and the timing estimate was derived in Section 4.2.2 (STARAN matrix operations).

5. CONCLUSIONS AND RECOMMENDATIONS

5.1 GENERAL

The following conclusions and recommendations were formulated from the results derived during the course of this study. Each of the three study tasks are individually discussed below.

5.2 AS-11BX POSTPROCESSING

Based upon the results of the AS-11BX postprocessing study and validation, the following conclusions were derived:

1. The AS-11BX postprocessing tasks are well suited to STARAN associative array processing.
2. The standard STARAN S-1000 (four arrays) is more than adequate for the basic postprocessing functions under consideration.
3. Additional postprocessing tasks can be accommodated by the STARAN AAP.
4. STARAN can efficiently perform the postprocessing in either an on-line or off-line mode.

The postprocessing tasks are well suited to associative array processing (AAP) because the algorithms can be structured to permit full utilization of the array elements during major operations. The standard STARAN S-1000 can simultaneously service many (over 50) AS-11BX stereomappers. Therefore, STARAN can easily accommodate additional postprocessing tasks. However, with or without additional tasks, a single-array STARAN S-250 would probably suffice.

By providing storage buffers for the data output by multiple AS-11BX's or magnetic tape units, the STARAN can ignore the transfer rate of the data sources and efficiently perform the postprocessing tasks in either an on-line (direct to AS-11BX's) or off-line (tape unit data source) mode. This subject is pursued more fully in Section 2.2.6.

The contractor recommends that:

1. Additional postprocessing tasks be evaluated to determine their suitability to associative array processing.

2. An operational system concept, involving an appropriate man-machine interface, be developed.

Some additional postprocessing tasks suggested by the Defense Mapping Agency Aerospace Center are:

1. Generation of the model data (elevations) from the parallax data. This task is presently slated for the AS-11BX.
2. Application of model deformation characteristic corrections to the model data.
3. Transformation of the model data from the local coordinate system to a geographic coordinate system.

These tasks are discussed more fully in Section 2.2.8.

At this point an operational system concept should be developed to factor in the non-AAP postprocessing functions such as data verification and editing. This would put more realistic bounds on the performance requirements of the various postprocessing tasks.

5.3 RASTER PROCESSING - AUTOMATED CARTOGRAPHY

On the basis of the impressive study results, the contractor has concluded that an extremely good match exists between the STARAN associative array processor (AAP) and the basic raster processing tasks. The timing estimates for a standard STARAN S-1000 (four arrays) show at least a two order of magnitude improvement over the execution times measured on the presently implemented IBM 360/40. The STARAN performance advantage is primarily due to these facts:

1. Most raster processing tasks are implemented in the STARAN AAP by storing the image directly (intact) into the arrays and operating directly on this image using simple logic instructions; one of the things STARAN does best.
2. The above approach eliminates the requirements to vectorize the line data, which is a very time consuming task.

3. Tasks involving arithmetic operations, and character processing used algorithms that were derived or restructured to realize full array utilization with moderate field lengths.
4. Many image points are processed simultaneously.
5. Algorithm performance is generally not a function of map density.
6. The intermediate tape I/O time required by the present sequential processor is eliminated because the processing time per task is in the order of seconds instead of hours.

In an overall system concept, STARAN could be time-shared between many completely buffered scanner/plotters (on-line in real time) or magnetic tape units (off-line operation) to essentially eliminate the effect of the I/O transfer time from the data source.

The contractor recommends that:

1. The basic AAP concepts of line thinning and line symbol generation be verified, using real map data, to provide a direct qualitative comparison with the results of the existing implementation.
2. Additional raster processing tasks be evaluated for the AAP implementation such as color merging and symbol generation for railroads and intermittent streams.
3. A concept be developed for an operational system that would include a STARAN and an appropriate man-machine interface.

5.4 SIMULTANEOUS MULTI-EXPOSURE ANALYTICAL CALIBRATION (SMAC) PROGRAM

The STARAN with a parallel head disc/drum can offer a significant advantage for the SMAC related search operations. However, for matrix operations involving double-precision floating-point arithmetic, a standard four-array STARAN is about equivalent to a Univac 1108.

The effect on SMAC program performance of the time savings realized by the STARAN search operations cannot be positively identified because the measured Univac 1108 execution times do not permit visibility to the level of this task. However, ETL has implied that the search times are significant and therefore a STARAN would probably contribute to an overall reduction of the SMAC program execution time.

The conclusion can be made that the matrix operations for double-precision floating-point arithmetic will provide satisfactory performance for a test bed environment. However, if STARAN were to be applied primarily to applications requiring matrix arithmetic, the random access parallel input/output memory, and hardware floating-point arithmetic options should be developed to realize significant improvement in performance.

The contractor recommends that an attempt be made to obtain a more detailed breakdown of the Univac 1108 execution times for the SMAC program. This breakdown will provide a reference for the estimated STARAN S-1000 executions times from which more specific conclusions can be drawn.

APPENDIX A - AS-11BX POSTPROCESSING VALIDATION DATA

1. INTRODUCTION

This section provides detailed information pertinent to the validated AS-11BX postprocessing function: coordinate transformation and interpolation. A printer format description is provided in Section 2 so the reader can interpret the various printouts of test data and test results. Section 3 describes the input test data, results, and program printout for the coordinate transformation function; Section 4 describes the same items as Section 3 for the interpolation function.

2. PRINTER FORMAT DESCRIPTION

A brief outline of the format of the printed output is provided to aid the reader.

Figure A-1 corresponds to the first lines of the printout of the input test data used for the coordinate transformation program.

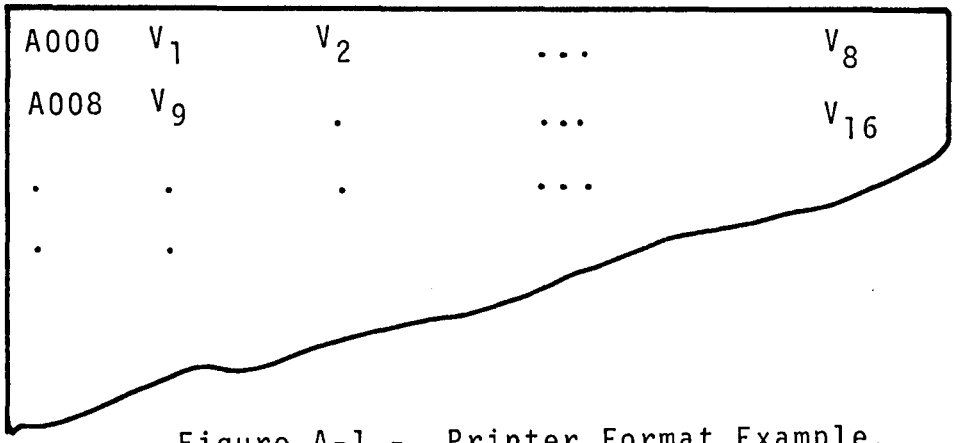


Figure A-1 - Printer Format Example.

Hexadecimal integer formatting is used throughout. The fixed point values (V_1, V_2, etc.) are 32-bit fields, each associated with a different 16-bit bulk core memory address.

V_1 is a 32-bit fixed point value stored at location $A000_{16}$ (Hexadecimal). V_2 is a 32-bit fixed point value stored at location A001; V_9 is stored at A008, etc.

3. COORDINATE TRANSFORMATION

3.1 General

The details about the test data used to validate the coordinate transformation program and the test results are given below. Included are the computer printouts of the input data, test results, and coordinate transformation program.

3.2 Test Data

Equation 1 below shows the relationship between points referenced to both the model (X_M, Y_M, Z_M) and local (X_L, Y_L, Z_L) coordinate systems:

$$\begin{pmatrix} X_L \\ Y_L \\ Z_L \end{pmatrix} = SA \begin{pmatrix} X_M - X_0 \\ Y_M - Y_0 \\ Z_M - Z_0 \end{pmatrix}$$

Equation 1 is described in Section 2.2.2.2. Elements of the 3-by-3 rotation matrix are direction cosines defined as follows:

$$\begin{aligned}
a_{11} &= \cos\alpha \cos\gamma - \sin\alpha \sin\beta \sin\gamma, \\
a_{12} &= \cos\alpha \sin\gamma + \sin\alpha \sin\beta \cos\gamma, \\
a_{13} &= -\sin\alpha \cos\beta, \\
a_{21} &= -\cos\beta \sin\gamma, \\
a_{22} &= \cos\beta \cos\gamma, \\
a_{23} &= \sin\beta, \\
a_{31} &= \sin\alpha \cos\gamma + \cos\gamma \sin\beta \sin\gamma, \\
a_{32} &= \sin\alpha \sin\gamma - \cos\alpha \sin\beta \cos\gamma, \text{ and} \\
a_{33} &= \cos\alpha \cos\beta.
\end{aligned}$$

Angles α,β,γ are defined by assuming two superimposed orthogonal coordinate systems (X_L, Y_L, Z_L and X_M, Y_M, Z_M) as shown in Figure A-2. Then α,β,γ are the angles of system X_L, Y_L, Z_L when it is rotated about the axes Y_M (primary axis), X_M (secondary axis), and Z_M (tertiary axis), respectively.

The relationship of the model and local coordinate systems used to validate the coordinate transformation program is shown in Figure A-3. In this figure; $\alpha = 0^o$ deg; $\beta = 0^o$ deg; and $\gamma = 30^o$ deg.

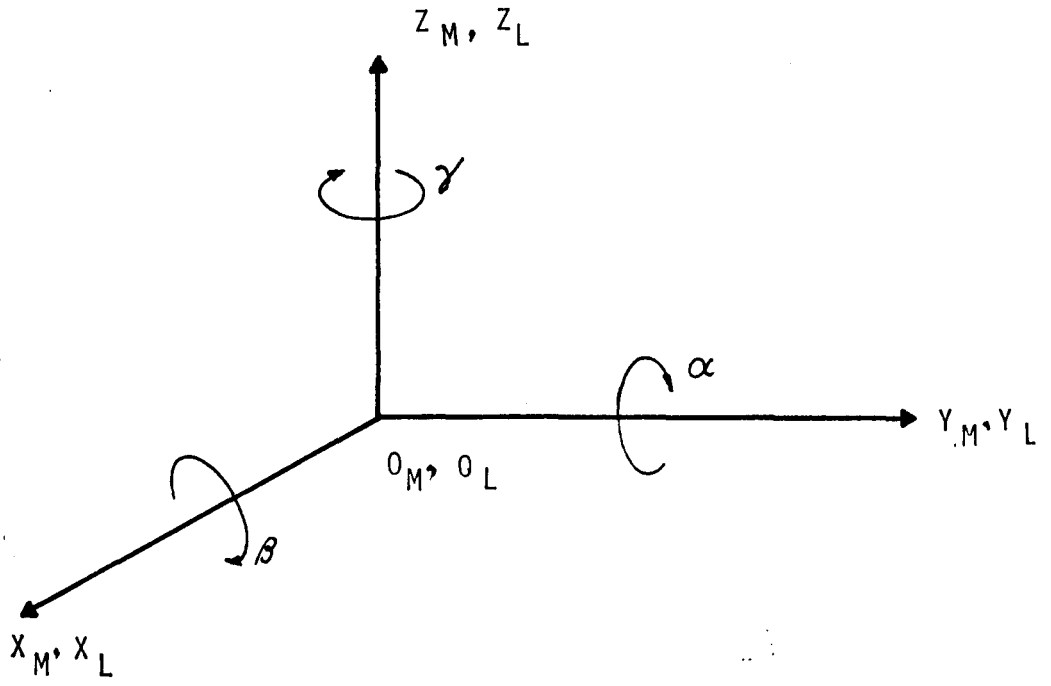


Figure A-2 - Rotational Relationship between Two Orthogonal Coordinate Systems

Also, the origin of the local coordinate system with respect to the model coordinate system is:

$$X_0 = 0;$$
$$Y_0 = 0;$$
$$X_0 = 30°$$

The elements of the rotation matrix are computed from the α, β, γ values above and substituted into Equation 1 along with the values of X_0, Y_0, Z_0 to result in:

$$\begin{pmatrix} X_L \\ Y_L \\ Z_L \end{pmatrix} = \begin{pmatrix} \sqrt{3/2} & 1/2 & 0 \\ -1/2 & \sqrt{3/2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_M \\ Y_M \\ Z_M + 64 \end{pmatrix} \quad (2)$$

where the scale Factor S is 1. The above data will be constant for all points in the model system.

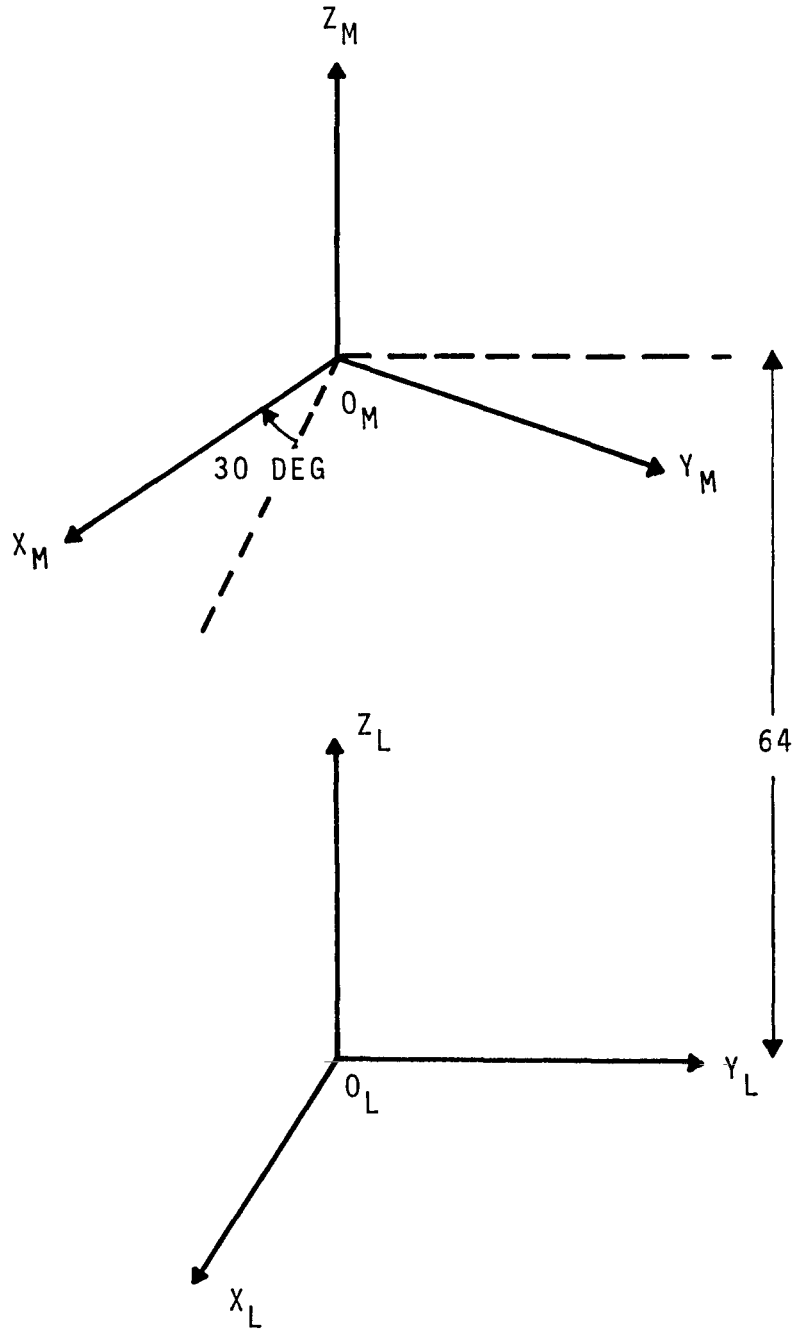


Figure A-3 - Relationship between Model and Local/Coordinate System

The variable input data X_M, Y_M, Z_M was chosen as: $X_M = N\sqrt{3}$; $Y_M = N$; and $Z_M = N$, where N = 0, 1, 2, 3, ... 169.

These test data are essentially points along a straight line eminating from the origin of the model coordinate system. This pattern for the input test data was chosen to simplify verification of the transformed results. The two-array STARAN S-500 used to validate the program required 170 points of test data (85 points per array; (see Section 2.2.2.3) for a full load. With three parameters per point (X_M, Y_M, Z_M), the test data consisted of 510 values.

The decimal and hexadecimal test values for the rotation matrix are given in Table A-1. The elements were implemented in 24-bit fields with a 22-bit fractional part.

The values of X_0, Y_0, Z_0 were implemented in 18-bit fields with a 4-bit fractional part. The scale factor, S, was allocated an 18-bit field with a 2-bit fractional part. The decimal and hexadecimal values for X_0, Y_0, Z_0, and S are shown in Tables A-II.

The values of X_M, Y_M, Z_M ($N\sqrt{3}$, N, N) were implemented as 20-bit fields stored right justified in the 32-bit memory locations of bulk core starting at hexadecimal address $A000_{16}$. These values were given four places for their fractional part. The 510 values (170 points) of input test data are shown hexadecimally in the computer printout shown in Figure A-4.

The constant data (rotation matrix, local grid origin, and scale factor) were first stored in the array. Then the input test data values were transferred from bulk core memory into the two arrays and the array computations were performed. Upon completion of the computations, the test results were stored back into an area of the bulk core memory.

3.3 Test Results

The anticipated test results are derived by substituting the expressions for the input test data values into Equation 2 and expanding as shown below:

TABLE A-I - ROTATION MATRIX VALUES FOR TEST PROBLEM

Elements of the Rotation matrix	Decimal value	Hexadecimal value
a_{11}	$\sqrt{3}/2$	ØØ376DØØ
a_{12}	1/2	ØØ2ØØØØØ
a_{13}	Ø	ØØØØØØØØ
a_{21}	-1/2	ØØEØØØØØ
a_{22}	$\sqrt{3}/2$	ØØ37ØØØØ
a_{23}	Ø	ØØØØØØØØ
a_{31}	Ø	ØØØØØØØØ
a_{32}	Ø	ØØØØØØØØ
a_{33}	1	ØØ4ØØØØØ

TABLE A-II - LOCAL GRID ORIGIN AND SCALE FACTOR VALUES FOR TEST PROBLEM

Parameter	Decimal value	Hexadecimal value
X_0	Ø	ØØØØØØØØ
Y_0	Ø	ØØØØØØØØ
Z_0	-64	ØØØFFCØØ
S	1	ØØØØØØ4

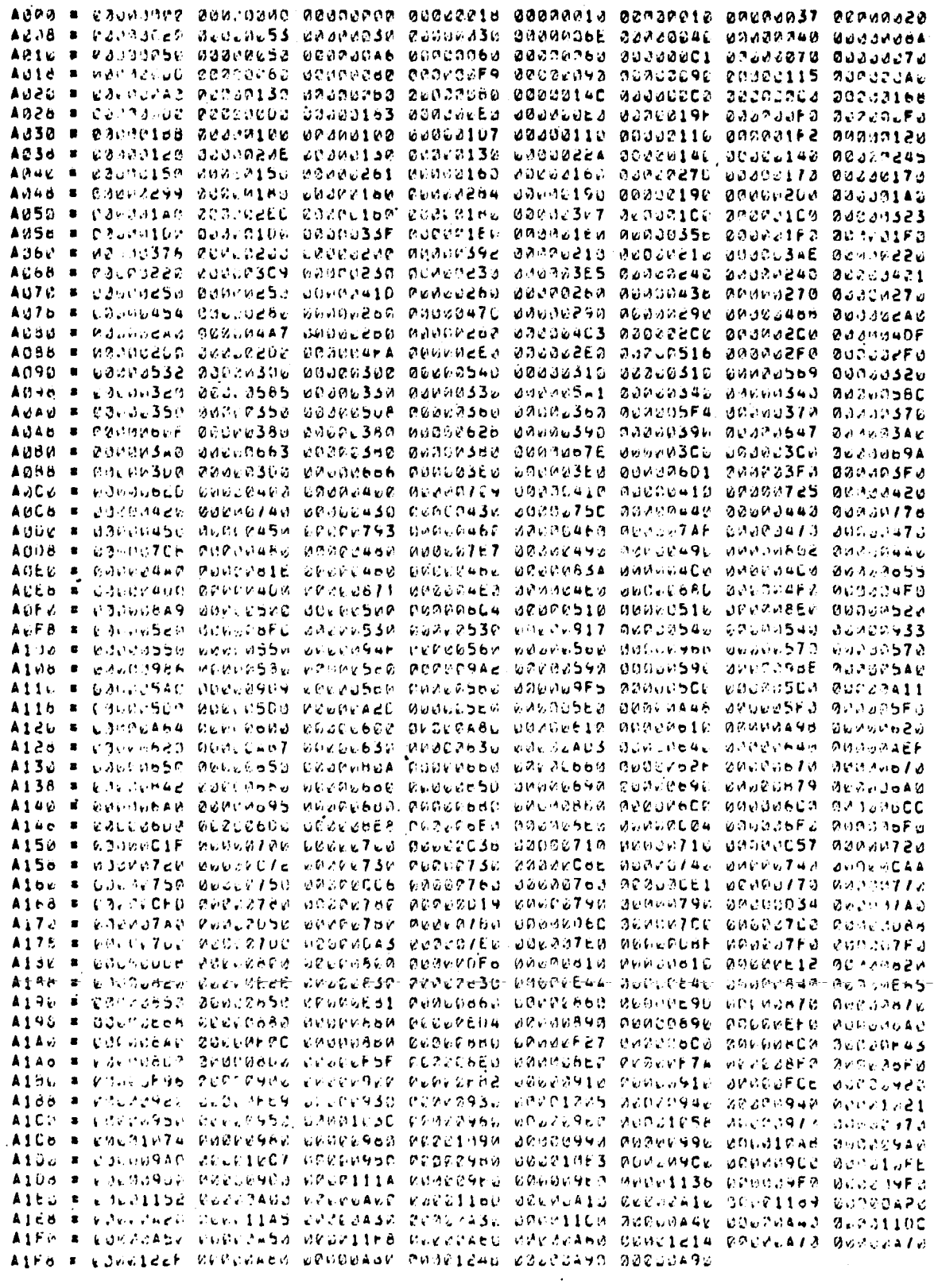


Figure A-4 - Coordinate Translation Test Data Printout
(Sheet 1 of 2)

$$\begin{pmatrix} X_L \\ Y_L \\ Z_L \end{pmatrix} = \begin{pmatrix} \sqrt{3}/2 & 1/2 & 0 \\ -1/2 & \sqrt{3}/2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} N\sqrt{3} \\ N \\ N + 64 \end{pmatrix}, \quad (3)$$

where

$$X_L = \frac{3N}{2} + \frac{N}{2},$$

$$Y_L = -\frac{N\sqrt{3}}{2} + \frac{N\sqrt{3}}{2}, \text{ and}$$

$$Z_L = N + 64.$$

The anticipated test results for the transformed input data are then given as:

$$X_L = 2N$$

$$Y_L = 0, \text{ and}$$

$$Z_L = N + 64,$$

where

$$N = 0, 1, 2 \ldots 169.$$

After performing the array computations, the actual test results were stored in an area of bulk core memory. The results are in 20-bit fields (four bits for fractional values) stored right justified in locations $A000_{16}$-$A1FD_{16}$ as shown in the printout of Figure A-5.

Each group of three words, reading from left-to-right and down the page, correspond to the X_L, Y_L, Z_L values of the 170 test points. The test result values are equal to the predicted results of 2N, 0, N where N = 0, 1, 2, ... 169 for the 170 points.

3.4 Program Printout

Figure A-6 is a side-by-side computer listing of the coordinate transformation program written in the APPLE[1] assembly language. The columns of information in the listing are explained below:

1. Column 1 contains the load location of the instruction.

[1] GER-15637A: APPLE Programming Manual. Akron, Ohio, Goodyear Aerospace Corporation. September 1973.

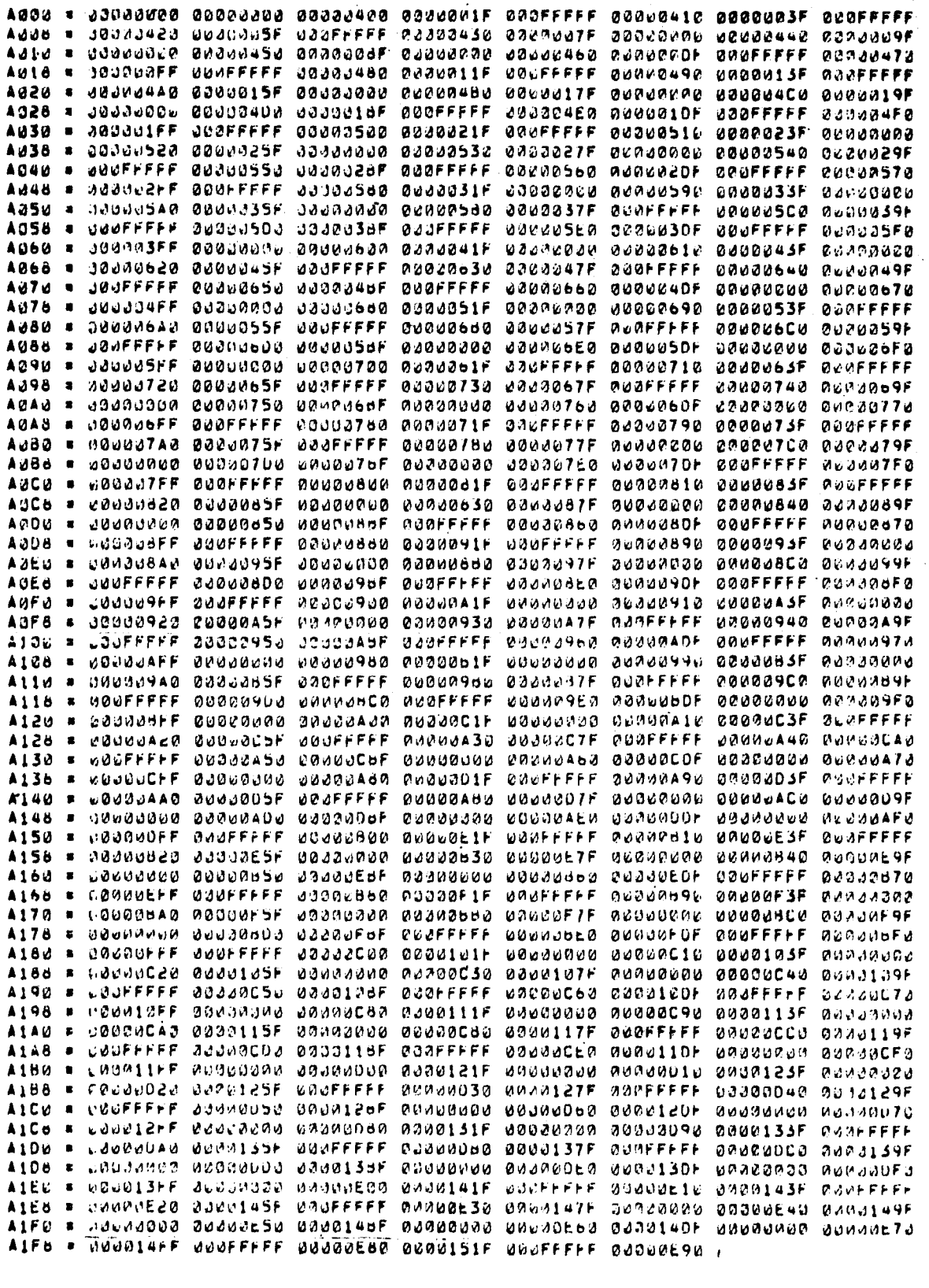


Figure A-5 - Printout of Coordinate Translation Test Results

```
APPLE V01-00 14-DEC-73 15:43:32 PAGE 00001

                                START
                        ********* AS-11-8X  POST PROCESSING      ******
                        *********                                ******
                        ********* COORDINATE TRANSFORMATION      ******
                        *********                                ******
                   0014 F1         DF              0,20
                   1414 F2         DF              20,20
                   2814 F3         DF              40,20
                   3C12 F4         DF              60,18
                   4E12 F5         DF              78,18
                   6012 F6         DF              96,18
                   7212 F7         DF              114,18
                   8418 F8         DF              132,24
                   9C18 F9         DF              156,24
                   B418 F10        DF              180,24
                   0640            ORG             X'0640'
0640 0640 00376D00 ROTM11          DC              X'00376D00'
0641 0641 00200000 ROTM12          DC              X'00200000'
0642 0642 00000000 ROTM13          DC              0
0643 0643 00E00000 ROTM21          DC              X'00E00000'
0644 0644 00376D00 ROTM22          DC              X'00376D00'
0645 0645 00000000 ROTM23          DC              0
0646 0646 00000000 ROTM31          DC              0
0647 0647 00000000 ROTM32          DC              0
0648 0648 00400000 ROTM33          DC              X'00400000'
0649 0649 00000001 SETY            DC              X'00000001'
064A 064A 00000004 SCALE           DC              X'00000004'
064B 064B 00000000 X0              DC              0
064C 064C 00000000 Y0              DC              0
064D 064D 00FFC00  Z0              DC      X'000FFC00'
064E 064E 92492492 MK11            DC              X'92492492'
064F 064F 49249249 MK12            DC              X'49249249'
0650 0650 24924924 MK13            DC              X'24924924'
0651 0651 6EDA0000 ROOT3           DC              X'6EDA0000'
                   0000            ORG     0
                   A000 INDATA     EQU     X'A000'
                   0760 SAVBL      EQU   X'0760'
                        **         INPUT CONSTANTS  SECTION  **
0000 0000 340AC000 STRT            LI              ASH,X'C000'
0001 0001 75EAFFFF                 LI              (FL1,FP3),X'FFFF'
0002 0002 00107741                 CLR             Y
0003 0003 3CLA0004                 RPT
0004 0004 1BA4AA02                 S               Y,FP3-
0005 0005 34A00001                 LI              BL,1
0006 0006 3A010760                 SR              (BL,DP),SAVBL
0007 0007 40107741                 CLR             X
0008 0008 00107741                 CLR             Y
0009 0009 3601064E                 LR              C,MK11
000A 000A 42009940                 SC              Y(0)
000B 000B 3601064F                 LR              C,MK12
000C 000C 42009940                 SC              Y(1)
000D 000D 36010650                 LR              C,MK13
000E 000E 42009940                 SC              Y(2)
000F 000F 40108BA2                 L               X,Y
0010 0010 4010B65A                 ROT             Y,96
0011 0011 4010B65A
0012 0012 40109942                 LOR             X,Y
0013 0013 4010F65A                 ROT             Y,63
```

Figure A-6 - Printout of Coordinate Translation Program (Sheet 1 of 6)

```
0014 0014 40L08B5A
0015 0015 4CU096A2          LOR             X,Y
0016 0016 46L00003          L               M,X
0017 0017 5AIF0001          S               M,255
0018 0018 00U07741          CLR             Y
0019 0019 5A7F44A2          GEN,32          X'5A7F44A2'
001A 001A 74U04000          LI              ASH,X'4000'
001B 001B 1AL70003          S       X,7
001C 001C 34UAC000          LI              ASH,X'C000'
001D 001D 32U0FFFF          LI              CL,X'FFFF'
001E 001E 3420FFFF          LI              CH,X'FFFF'
001F 001F 00U088F0          GEN,32          X'008088F0'
0020 0020 40LU9552          GEN,32          X'40E09952'
0021 0021 40L09552          GEN,32          X'40C09952'
0022 0022 1AL80002          S               Y,8
0023 0023 40L0CC5A          GEN,32          X'40C0CC5A'
0024 0024 1AL90002          S               Y,9
0025 0025 40EUCC5A          GEN,32          X'40E0CC5A'
0026 0026 1ALAA002          S               Y,10
0027 0027 40F0CC5A          GEN,32          X'40F0CC5A'
0028 0028 1ALBU002          S               Y,11
0029 0029 40F8CC5A          GEN,32          X'40F8CC5A'
002A 002A 1ALC0002          S               Y,12
002B 002B 40FCCC5A          GEN,32          X'40FCCC5A'
002C 002C 1ALD0002          S               Y,13
002D 002D 40FECC5A          GEN,32          X'40FECC5A'
002E 002E 1ALE0002          S               Y,14
002F 002F 40FFCC5A          GEN,32   X'40FFCC5A'
0030 0030 5AIF0002          S       Y,15
0031 0031 42L098A1          SET             M
0032 0032 46L00003
0033 0033 37502713          MVF             F1,F2
0034 0034 3F130037
0035 0035 43046845
0036 0036 4B0U0001
0037 0037 13A40003
0038 0038 36L100S1          LR              C,ROOT3
0039 0039 75L01313          MPC             F1,(0,32),(50,52)
003A 003A 73L01F65
0038 0038 33201F00
003C 003C 2CL00000
003D 003D 37501347          MVF             (52,20),F1
003E 003E 3F13004I
003F 003F 43046845
0040 0040 4B00U001
0041 0041 13A40003
0042 0042 37503B27          MVF             F2,F3
0043 0043 3F13U046
0044 0044 43046845
0045 0045 4B030001
0046 0046 13A40003
0047 0047 00U0B841          SET     Y
0048 0048 32U0A000          LI              DP,0
0049 0049 74AA30AA          LI              BL,170
004A 004A 431B0049  CROT    STEP
004B 004B 76L00000          LC              F1
004C 004C 47L088A5
004D 004D 40LC38B3
```

Figure A-6 - Printout of Coordinate Translation Program (Sheet 2 of 6)

```
004E 004E 25L06CF8
004F 004F 33L04A000          SR              C,INDATA(DP),2
0050 0050 76L000000          LC              F2
0051 0051 47L088A5
0052 0052 40L088B3
0053 0053 25L16CF8
0054 0054 30L04A000          SR              C,INDATA(DP),2
0055 0055 76L000C00          LC              F3
0056 0056 27L14CFD
0057 0057 30L04A000          SR              C,INDATA(DP),2
0058 0058 01L300001          DECR             BL
0059 0059 38L020000          WAIT
005A 005A 2911006A           BNZ,BL          CROT
005B 005B 081F0005           L               M,255
005C 005C 36L10640           LR              C,ROTM11
005D 005D 33L01F98           SC               (8,24),F8
005E 005E 3F170061
005F 005F 40L087A3
0060 0060 48400001
0061 0061 13740003
0062 0062 36L10641           LR              C,ROTM12
0063 0063 33L01FB3           SC               (8,24),F9
0064 0064 3F170067
0065 0065 40L087A3
0066 0066 48400001
0067 0067 13740003
0068 0068 36L10642           LR              C,ROTM13
0069 0069 33L01FC8           SC               (8,24),F10
006A 006A 3F17006D
006B 006B 40L087A3
006C 006C 48400001
006D 006D 13740003
006E 006E 40L088A1           ROT             M,1
006F 006F 40L1F8CB3
0070 0070 03L003003
0071 0071 36L10643           LR              C,ROTM21
0072 0072 33L01F98           SC               (8,24),F8
0073 0073 3F170076
0074 0074 40L087A3
0075 0075 48400001
0076 0076 13740003
0077 0077 36L10644           LR              C,ROTM22
0078 0078 33L01FB3           SC               (8,24),F9
0079 0079 3F17007C
007A 007A 40L087A3
007B 007B 48400001
007C 007C 13740003
007D 007D 36L10645           LR              C,ROTM23
007E 007E 33L01FC8           SC               (8,24),F10
007F 007F 3F170082
0080 0080 40L087A3
0081 0081 48400001
0082 0082 13740003
0083 0083 40L088A1           ROT             M,1
0084 0084 40L1F8CB3
0085 0085 03L003003
0086 0086 36L10646           LR              C,ROTM31
0087 0087 33L01F98           SC               (8,24),F8
```

Figure A-6 - Printout of Coordinate Translation Program (Sheet 3 of 6)

```
0088 0088 3F173088
0089 0089 40600A3
008A 008A 4840030001
008B 008B 13740003
008C 008C 36010647          LR              C,ROTM32
008D 008D 33L01F83          SC               (8,24),F9
008E 008E 3F170091
008F 008F 4060087A3
0090 0090 4840000001
0091 0091 13740003
0092 0092 36013548          LR              C,ROTM33
0093 0093 33L01FC8          SC               (8,24),F10
0094 0094 3F170097
0095 0095 4060087A3
0096 0096 4840000001
0097 0097 13740003
0098 0098 4200380A1         SET             M
0099 0099 0800000003
009A 009A 36013648          LR              C,SCALE
009B 009B 33L01F83          SC        (14,18),F7
009C 009C 3F11009F
009D 009D 4300087A3
009E 009E 4840000001
009F 009F 13740003
00A0 00A0 36010649          LR              C,X0
00A1 00A1 33L01F40          SC        (14,18),F4
00A2 00A2 3F1100A5
00A3 00A3 4060087A3
00A4 00A4 4840000001
00A5 00A5 13740003
00A6 00A6 3601064C          LR              C,Y0
00A7 00A7 33L01F5F          SC        (14,18),F5
00A8 00A8 3F1130AB
00A9 00A9 4060087A3
00AA 00AA 4840000001
00AB 00AB 13740003
00AC 00AC 3601364D          LR              C,Z0
00AD 00AD 33L01F71          SC        (14,18),F6
00AE 00AE 3F1100B1
00AF 00AF 4060087A3
00B0 00B0 4840000001
00B1 00B1 13740003
00B2 00B2 3800200000        WAIT
                      **    LOAD   VARIABLES  SECTION  **
00B3 00B3 3E0107B3 BTO      LR              (BL,DP),SAVBL
00B4 00B4 29L1012A          BZ,BL           EXT
00B5 00B5 411300001         DECR            BL
00B6 00B6 3901307D          SR              (BL,DP),SAVBL
00B7 00B7 4200000A1         SET             M
00B8 00B8 4800000003
00B9 00B9 0900077741        CLR             Y
00BA 00BA 3340000000        LI              FP2,0
00BB 00BB 3E03B4BC          RPT,60
00BC 00BC 1600000042        S               Y,FP2+
00BD 00BD 3800200000        WAIT
00BE 00BE 02FF8845          L               Y,255
00BF 00BF 7200000000        LI              DP,0
00C0 00C0 43F8664A BK2      STEP
```

Figure A-6 - Printout of Coordinate Translation Program (Sheet 4 of 6)

```
APPLE V01-00 14-DEC-73 15:44:00 PAGE 00005

00C1 00C1 A8000002            L               M,Y
00C2 00C2 3604A000 LOAD1      LR              C,INDATA(DP),2
00C3 00C3 4C000BA1            SCW             (12,20),F1
00C4 00C4 4FF04A9F
00C5 00C5 42000840
00C6 00C6 40F00A5A
00C7 00C7 40FC085A
00C8 00C8 57F00002
00C9 00C9 08000003
00CA 00CA 3604A200            LR              C,INDATA(DP),2
00CB 00CB 40000BA1            SCW             (12,20),F2
00CC 00CC 4FF0ACFF
00CD 00CD 42000841
00CE 00CE 42F00E52
00CF 00CF 42B00002
00D0 00D0 43F00001
00D1 00D1 42000840
00D2 00D2 40F00E52
00D3 00D3 57F00002
00D4 00D4 05B00003
00D5 00D5 3604A000            LR              C,INDATA(DP),2
00D6 00D6 40000BA1            SCW             (12,20),F3
00D7 00D7 4FF1A80F
00D8 00D8 42000840
00D9 00D9 40FC085A
00DA 00DA 4300085A
00DB 00DB 57F00002
00DC 00DC 42000203
00DD 00DD 0C000041            L               Y,M
00DE 00DE 28F100C0            BRS             BR2
00DF 00DF 38002000            WAIT
00E0 00E0 40000BA1            SET             M
00E1 00E1 08000003
00E2 00E2 33400003C           LI        FP2,60
00E3 00E3 41040001 BRL1       DECR            FP2
00E4 00E4 43406EA5            L               X,FP2
00E5 00E5 43400E45            L               Y,FP2
00E6 00E6 40FF8852            ROT             Y,1
00E7 00E7 40199A2             LOR             X,Y
00E8 00E8 40F88852            ROT             Y,1
00E9 00E9 40099A2             LOR             X,Y
00EA 00EA 18400003            S               X,FP2
00EB 00EB 2851FFE3            BNZ,FP2         BRL1
00EC 00EC 38002000            WAIT
                         *    WITHIN  ARRAY COMPUTING
00ED 00ED 40000BA1            SET             M
00EE 00EE 48000003
00EF 00EF 75F01113            SBF             F1,F4,F1
00F0 00F0 73F04C13
00F1 00F1 37201313
00F2 00F2 2C004000
00F3 00F3 75F01127            SBF             F2,F5,F2
00F4 00F4 73F05F27
00F5 00F5 37201313
00F6 00F6 2C004000
00F7 00F7 75F01138            SBF             F3,F6,F3
00F8 00F8 73F07138
00F9 00F9 37201313
```

Figure A-6 - Printout of Coordinate Translation Program (Sheet 5 of 6)

```
                APPLE V01-00 14-DEC-73 15:44:06 PAGE 00006

00FA 00FA 2C000000
00FB 00FB 75E01313          MPF   F1,F8,(205,44)
00FC 00FC 73L09BF8
00FD 00FD 33201700
00FE 00FE 2CB00000
00FF 00FF 379013E2          MVF   (207,20),F1
0100 0100 3F130103
0101 0101 43548FA5
0102 0102 4BB00001
0103 0103 13A40003
0104 0104 75E01327          MPF   F2,F9,(205,44)
0105 0105 73L0B3F8
0106 0106 33201700
0107 0107 2CB00000
0108 0108 75E01313          ADF   F1,(207,20),F1
0109 0109 73L0E213
010A 010A 37201313
010B 010B 2C000000
010C 010C 75E01338          MPF   F3,F10,(205,44)
010D 010D 73L0CBF8
010E 010E 33201700
010F 010F 2C000000
0110 0110 75E01313          ADF    F1,(207,20),F1
0111 0111 73L0E213
0112 0112 37201313
0113 0113 2C000000
0114 0114 75E01313          MPF   F1,F7,(61,38)
0115 0115 73L09362
0116 0116 33201100
0117 0117 2CB00000
0118 0118 38002000          WAIT
                       *    UNLOADING    RESULTS
0119 0119 36010649          LR                C,SETY
011A 011A 40007741          CLR                 Y
011B 011B 42009940          SC                Y(7)
011C 011C 42004402          SN                Y,254
0110 0110 5AFE0003
011E 011E 22FE8645          L                 Y,254
011F 011F 72B00000          LI                DP,0
0120 0120 43F86640 UNL1     STEP
0121 0121 70L00000 WAR2     LC       (77,20)
0122 0122 47L088A5
0123 0123 40FC8883
0124 0124 40FF8888
0125 0125 40F08808
0126 0126 4C1C8883
0127 0127 25L36CF8
0128 0128 28F10120          BRS               UNL1
0129 0129 25L10083          B                 BTO
012A 012A 38B0608C EXT      GEN,32            X'38006080'
012B 012B 38002000          WAIT
                FFFF        END
```

Figure A-6 - Printout of Coordinate Translation Program (Sheet 6 of 6)

2. Column 2 contains the execution locations of the instructions. Three locations are equal for this example since the program was executed out of page memory.
3. Column 3 displays the machine language instructions.
4. Column 4 is the label field. Tags made up of Alphanumeric characters are used here to ease the burden of the programmer.
5. Column 5 is the command field and contains various APPLE mnemonic commands.
6. Column 6 contains the argument field that the commands operate on. The arguments supply the what or where to work.
7. The remaining area may be used for comments if desired.

For the coordinate transformation program, the first 10 statements define Fields in the array. The program is originated at location 640, Base 16. Constants are then defined and a relocation to address zero takes place. After loading, the program would begin at the location that is labelled "STRT"

4. INTERPOLATION

4.1 General

The details about the test data used to validate the interpolation program, and the test results derived from it, are presented below. Included are the printouts for the test data, test results, and interpolation program.

As discussed in Section 2.2.4, the interpolation task involves a least squares quadratic fit of the five points closest to the point of interest. The form is shown in Figure A-7. By normalizing the five sample points to the point of interest (X_G, Z_G) along the X-axis ($X_G = 0$), the amplitude (Z_G) at the point of interest is equal to C of the quadratic equation. Normalizing the sample points was more simple than computing the A and B constraints for the quadratic equation and solving for Z_G.

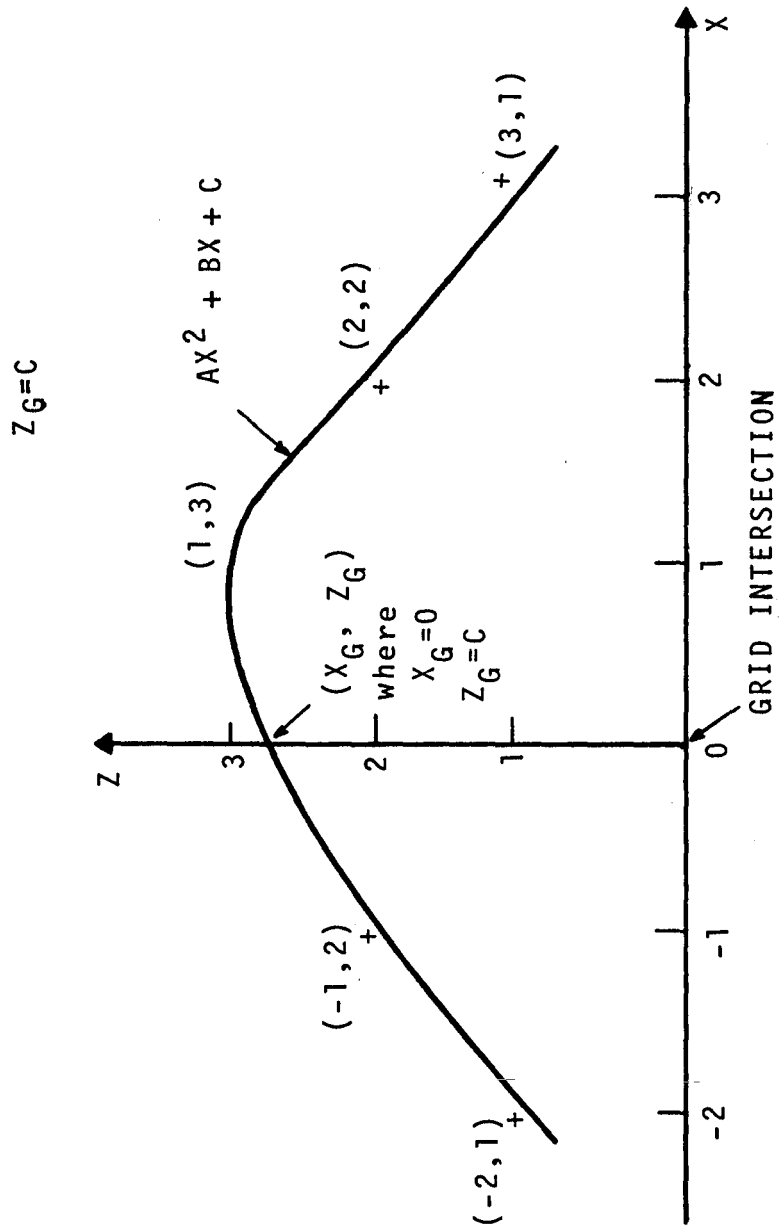


Figure A-7 - Test Data Configuration For Interpolation Program

The normal equation below represents a least squares quadratic fit to five points:

$$\begin{pmatrix} \Sigma Z_i \\ \Sigma X_i Z_i \\ \Sigma X_i Z_i \end{pmatrix} = \begin{pmatrix} 5 & \Sigma X_i & \Sigma X_i^2 \\ \Sigma X_i & \Sigma X_i^2 & \Sigma X_i^3 \\ \Sigma X_i^2 & \Sigma X_i^3 & \Sigma X_i^4 \end{pmatrix} \begin{pmatrix} C \\ B \\ A \end{pmatrix}, \quad (4)$$

where i = 1, 2, ... 5 for the five sample point values (X_i, Z_i).

Applying Cramer's Rule to solve for C (Z at X = 0) gives:

$$C = \frac{\begin{vmatrix} \Sigma Z_i & \Sigma X_i & \Sigma X_i^2 \\ \Sigma X_i Z_i & \Sigma X_i^2 & \Sigma X_i^3 \\ \Sigma X_i^2 Z_i & \Sigma X_i^3 & \Sigma X_i^4 \end{vmatrix}}{\begin{vmatrix} 5 & \Sigma X_i & \Sigma X_i^2 \\ \Sigma X_i & \Sigma X_i^2 & \Sigma X_i^3 \\ \Sigma X_i^2 & \Sigma X_i^3 & \Sigma X_i^4 \end{vmatrix}} . \quad (5)$$

4.2 Test Data

If the values of X_i, Z_i are those shown in Figure A-7, namely:

$X_1, Z_1 = -2, 1,$

$X_2, Z_2 = -1, 2,$

$X_3, Z_3 = 1, 3,$

$X_4, Z_4 = 2, 2,$ and

$X_5, Z_5 = 3, 1;$

then substituting into Equation 5 gives a value of C = 2·77.

Multiplying all X_i by some constant value will not alter the value of C. Also if the values of the Z_i are increased by some constant K, the new value of C becomes C + K.

Based upon these facts, the test data chosen had the following values:

X_1	=	-2/3	Z_1	=	N + 1
X_2	=	1/3	Z_2	=	N + 2
X_3	=	1/3	Z_3	=	N + 3

$X_4 = 2/3 \qquad Z_4 = N + 2$

$X_5 = 3/3 \qquad Z_5 = N + 1$

where N = 100, 110, 120, ... 830 for the 84 points (42 points per array; (see Section 2.2.2.4.3)) required by the two-way STARAN S-500.

The X-values were selected to aid in number scaling; keeping the size of the numbers and their respective field lengths compatible. Primarily, however, the X and Z values were chosen to provide some variation in the test data while minimizing the effort required to verify the interpolated results.

The input test data values of X_i, Z_i were taken to be 18-bit fields stored right justified into the 32-bit words of bulk core memory, starting at location 8100_{16}. Figure A-8 corresponds to the first few lines of the input test data where V_1, V_2, etc. are 32-bit memory locations containing the values shown in Table A-III.

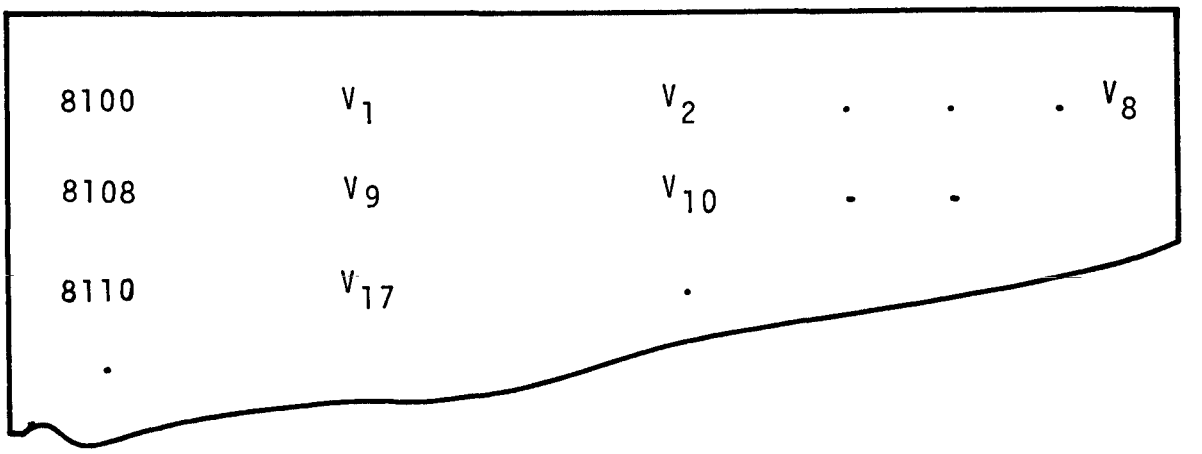


Figure A-8 - Printout Example of Interpolation Program Test Data

TABLE A-III - EXAMPLE OF INTERPOLATION PROGRAM TEST VALUES

Memory field	Sample point	Point of interest	Memory location	Decimal value	Hexadecimal value
V_1	X_1	1st	8100	-2/3	0003D556
V_2	Z_1		8101	101	00000065
V_3	X_2		8102	-1/3	0003EAAB
V_4	Z_2		8103	102	00000066
.	.		.	.	.
.	.		.	.	.
.	.		.	.	.
V_{10}	Z_5		8109	101	00000064
V_{11}	X_1	2nd	810A	-2/3	0003D556
V_{12}	X_2		810B	111	0000006F
.	.		.	.	.
.	.		.	.	.
.	.		.	.	.

A printout of the input test data values is shown in Figure A-9. These data are loaded into the two arrays and processed using the computations described in Section 2.2.4.6. The 84 results (42/array) are stored in bulk core memory.

4.3 Test Results

The test results are stored in locations 8100_{16} to 8153_{16} as 20-bit integer values (right justified in the words). The results correspond closely to the predicted values of 102.77, 112.77, ... etc. The results are shown in the printout in Figure A-10.

4.4 Program Printout

Figure A-11 is a side-by-side computer listing of the interpolation program written in the APPLE assembly language[1].

The management of this program printout is essentially the same as that described for the coordinate translation program in Section 3.4 of this Appendix.

[1] Op. cit.

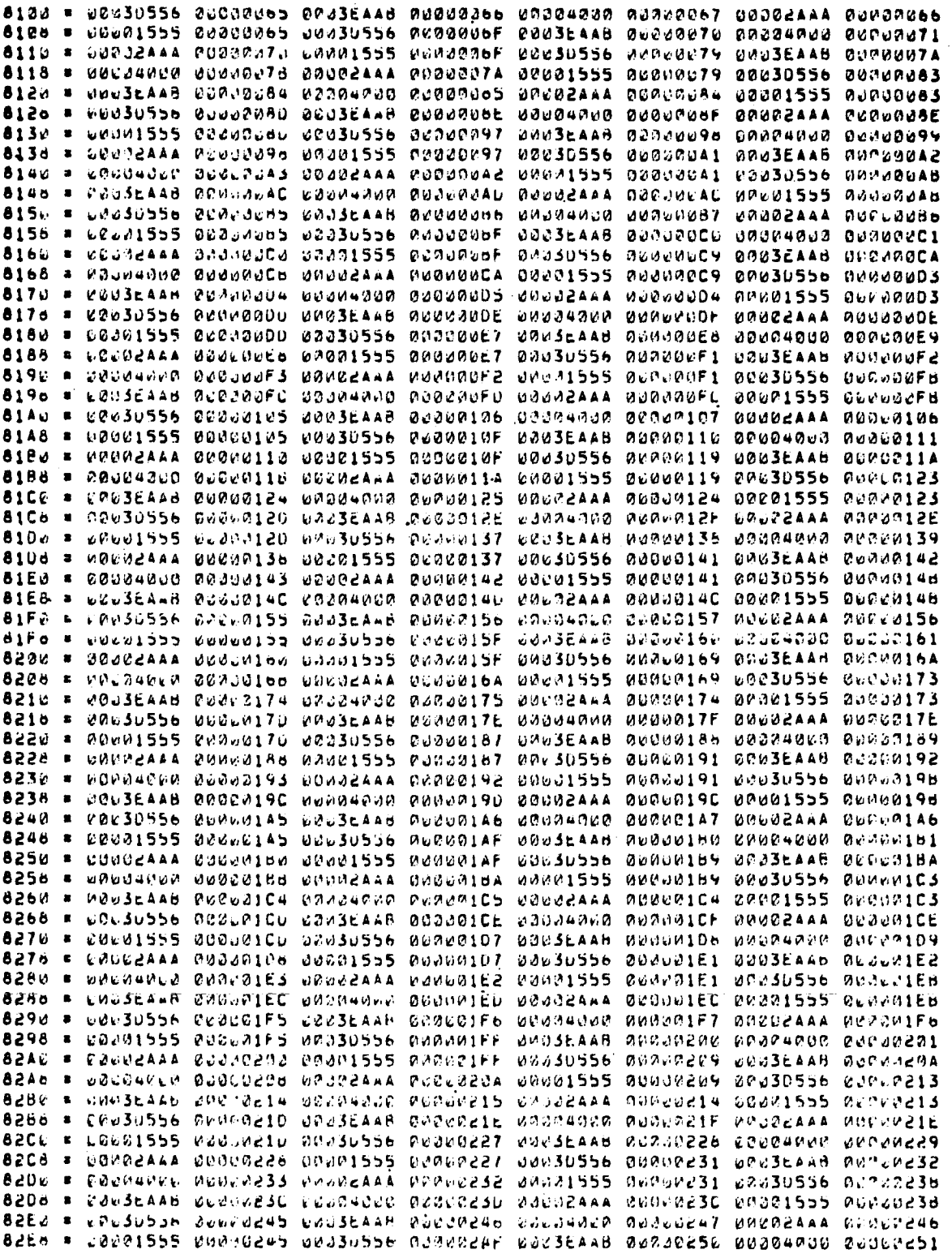


Figure A-9 - Printout of Interpolation Program Test Data
(Sheet 1 of 2)

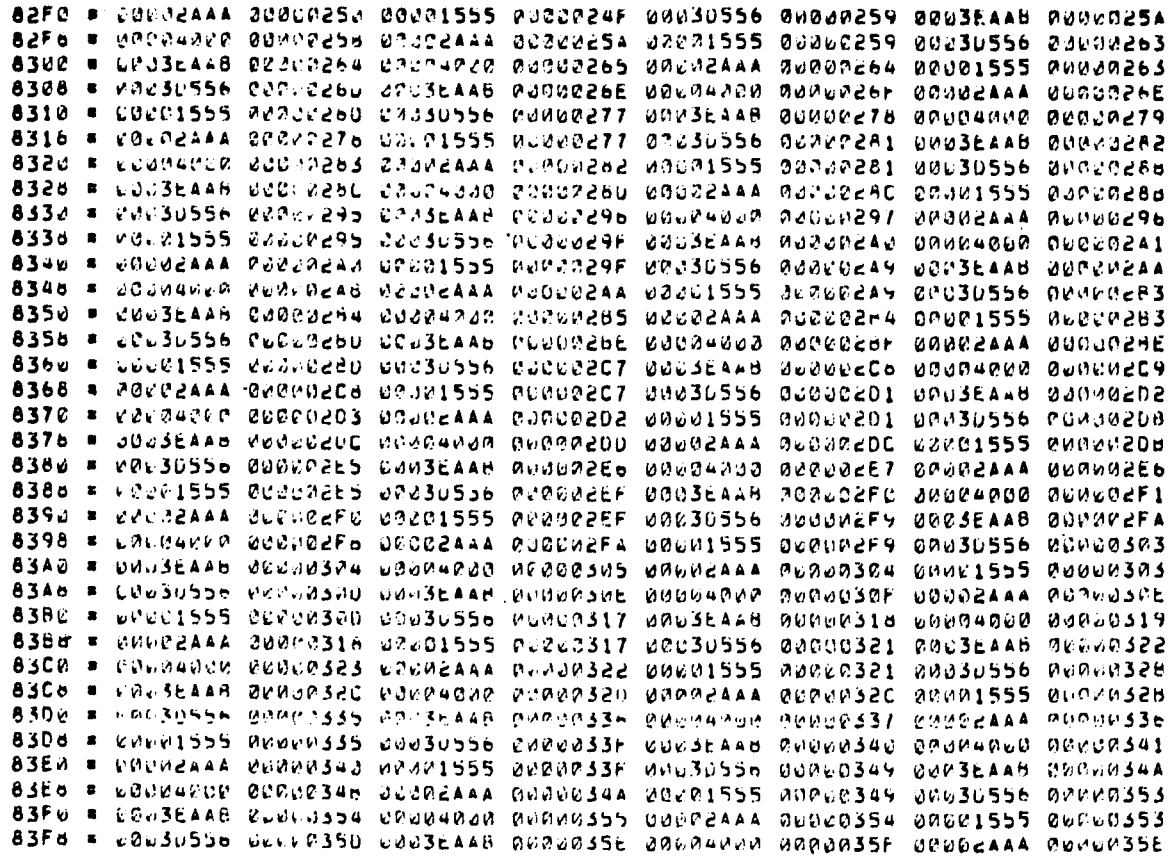


Figure A-9 - Printout of Interpolation Program Test Data (Sheet 2 of 2)

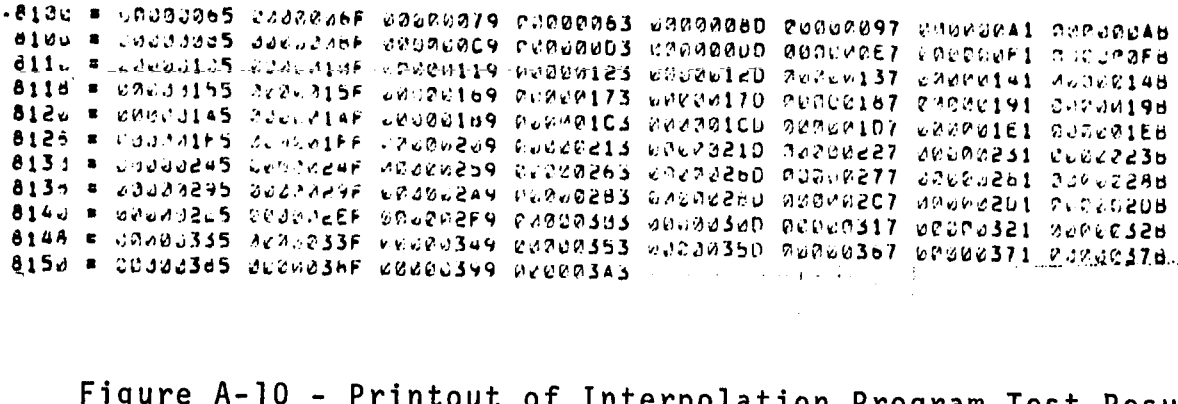


Figure A-10 - Printout of Interpolation Program Test Results

```
APPLE V01-00 16-NOV-73 14:16:09 PAGE 00001

                                START
                     ********* AS-11-8X POST  PROCESSING     *******
                     *********                                *******
                     ********* INTERPOLATION                 *******
                     *********                                *******
                0012 F1          DF                 0,18
                1212 F2          DF                 18,18
                2412 F3          DF                 36,18
                3612 F4          DF                 54,18
                4812 F5          DF                 72,18
                5A1E F6          DF                 90,30
                781E F7          DF                 120,30
                F501 MK1         DF                 245,1
                F601 MK2         DF                 246,1
                F701 MK3         DF                 247,1
                F801 MK4         DF                 248,1
                F901 MK5         DF                 249,1
                FA01 MK6         DF                 250,1
                FB01 MK7         DF                 251,1
                FC01 MK8         DF                 252,1
0000 0000 8A28A28A MASK          DC                 X'8A28A28A'          MK1
0001 0001 28A28A28               DC                 X'28A28A28'
0002 0002 A2BA28A2               DC                 X'A28A28A2'
0003 0003 82B82082               DC                 X'82082082'          MK2
0004 0004 08208208               DC                 X'08208208'
0005 0005 20820820               DC                 X'20820820'
0006 0006 30C30C30               DC                 X'30C30C30'          MK3
0007 0007 C30C30C3               DC                 X'C30C30C3'
0008 0008 0C30C30C               DC                 X'0C30C30C'
0009 0009 49249249               DC                 X'49249249'          MK4
000A 000A 24924924               DC                 X'24924924'
000B 000B 92492492               DC                 X'92492492'
000C 000C 24924924               DC                 X'24924924'          MK5
000D 000D 92492492               DC                 X'92492492'
000E 000E 49249249               DC                 X'49249249'
000F 000F 28A28A28               DC                 X'28A28A28'          MK6
0010 0010 A2BA28A2               DC                 X'A28A28A2'
0011 0011 8A28A28A               DC                 X'8A28A28A'
0012 0012 51451451               DC                 X'51451451'          MK7
0013 0013 45145145               DC                 X'45145145'
0014 0014 14514514               DC                 X'14514514'
0015 0015 1C71C71C               DC     X'1C71C71C'
0016 0016 71C71C71               DC   X'71C71C71'
0017 0017 C71C71C7               DC    X'C71C71C7'
0018 0018 00000000 SUBRT         DC                 X'00000000'
0019 0019 00000338               DC                 ROT3
001A 001A 00000334               DC                 ROT2
001B 001B 00000331               DC                 ROT1
001C 001C 0000000A SETC10        DC                 10
001D 001D 00000065 CZ1           DC                 101
001E 001E 00000066 CZ2           DC                 102
001F 001F 00000067 CZ3           DC                 103
0020 0020 00001555 CX1           DC                 X'00001555'
0021 0021 00002AAA CX2           DC                 X'00002AAA'
0022 0022 00004000 CX3           DC                 X'00004000'
0023 0023 0003D556 CX4           DC                 X'0003D556'
0024 0024 0003EAAB CX5           DC                 X'0003EAAB'
                0000             ORG    0
```

Figure A-11 - Printout of Interpolation Program
(Sheet 1 of 10)

```
APPLE V01-00 16-NOV-73 14:16:18 PAGE 00002

                A000 INDATA    EQU    X'A000'
                0760 SAVBL     EQU    X'0760'
                0761 SAVF      EQU    X'0761'
                     * COMPUTE THE    MASKS
0000 0000 340DC000 STRT        LI              ASH,X'C000'
0001 0001 75E0FFFF             LI       (FL1,FP3),X'FFFF'
0002 0002 00007741             CLR    Y
0003 0003 3C000004             RPT
0004 0004 1BA40002             S      Y,FP3-
0005 0005 34A00001             LI              BL,1
0006 0006 30B10760             SR              (BL,DP),SAVBL
0007 0007 32000000             LI              DP,0
0008 0008 33400FF5             LI              FP2,245
0009 0009 35700008             LI              FL1,8
000A 000A 36040000 BCK2        LR              C,MASK(DP),2
000B 000B 40007741             CLR             Y
000C 000C 02009940             SC              Y(0)
000D 000D 36040000             LR              C,MASK(DP),2
000E 000E 02209940             SC              Y(1)
000F 000F 36040000             LR              C,MASK(DP),2
0010 0010 42409940             SC              Y(2)
0011 0011 40008BA2             L               X,Y
0012 0012 40E0885A             ROT             Y,96
0013 0013 40D0885A
0014 0014 40A099A2             LOR             X,Y
0015 0015 40FC8F5A             ROT             Y,60
0016 0016 40E0885A
0017 0017 40D099A2             LOR             X,Y
0018 0018 1B000003             S               X,FP2+
0019 0019 01B10001             DECR            FL1
001A 001A 2891000A             BNZ,FL1         BCK2
001B 001B 40008BA1             SET             M
001C 001C 08000003
001D 001D 36010C1C             LR              C,SETC10
001E 001E 33C01F83             SC              (0,32),(100,32)
001F 001F 3F1FFF22
0020 0020 40008BA3
0021 0021 48400001
0022 0022 13740003
0023 0023 08F60E05             L               M,MK2
0024 0024 36010010             LR              C,CZ1
0025 0025 33C01F5F             SC              (0,32),(64,32)
0026 0026 3F1FFF29
0027 0027 40008BA3
0028 0028 48400001
0029 0029 13740003
002A 002A 36010023             LR              C,CX4
002B 002B 33C01F1F             SC              (0,32),(0,32)
002C 002C 3F1FFF2F
002D 002D 42008BA3
002E 002E 48400001
002F 002F 13740003
0030 0030 48008BA1             ROT             M,1
0031 0031 42FF8583
0032 0032 08000003
0033 0033 36010C1E             LR              C,CZ2
0034 0034 33C01F5F             SC              (0,32),(64,32)
0035 0035 3F1FFF38
```

Figure A-11 - Printout of Interpolation Program
(Sheet 2 of 10)

```
0036 0036 40U0B7A3
0037 0037 4B400001
0038 0038 13740003
0039 0039 36U100Z4          LR              C,CX5
003A 003A 33L01F1F          SC              (0,32),(0,32)
003B 003B 3F1F003E
003C 003C 40U0B7A3
003D 003D 4B400001
003E 003E 13740003
003F 003F 40U088A1          ROT             M,1
0040 0040 401F8883
0041 0041 08U00003
0042 0042 36U1001F          LR              C,CZ3
0043 0043 33L01F5F          SC              (0,32),(64,32)
0044 0044 3F1F0047
0045 0045 40U0B7A3
0046 0046 4B400001
0047 0047 13740003
0048 0048 36U100Z2          LR              C,CX3
0049 0049 33L01F1F          SC              (0,32),(0,32)
004A 004A 3F1F004D
004B 004B 40U0B7A3
004C 004C 4B400001
004D 004D 13740003
004E 004E 40U088A1          ROT             M,1
004F 004F 401F8883
0050 0050 08U00003
0051 0051 36U1001E          LR              C,CZ2
0052 0052 33L01F5F          SC              (0,32),(64,32)
0053 0053 3F1F0056
0054 0054 40U0B7A3
0055 0055 4B400001
0056 0056 13740003
0057 0057 36U100Z1          LR              C,CX2
0058 0058 33L01F1F          SC              (0,32),(0,32)
0059 0059 3F1F005C
005A 005A 40U0B7A3
005B 005B 4B400001
005C 005C 13740003
005D 005D 40U088A1          ROT             M,1
005E 005E 401F8883
005F 005F 08U00003
0060 0060 36U1001D          LR              C,CZ1
0061 0061 33L01F5F          SC              (0,32),(64,32)
0062 0062 3F1F0065
0063 0063 40U0B7A3
0064 0064 4B400001
0065 0065 13740003
0066 0066 36U100Z0          LR              C,CX1
0067 0067 33L01F1F          SC              (0,32),(0,32)
0068 0068 3F1F006B
0069 0069 40U0B7A3
006A 006A 4B400001
006B 006B 13740003
006C 006C 40U0BBA1          SET             M
006D 006D 08U00003
006E 006E 32000000          LI              DP,0
006F 006F 35A00002          LI              FP3,2
```

Figure A-11 - Printout of Interpolation Program
(Sheet 3 of 10)

```
APPLE V01-00 16-NOV-73 14:16:31 PAGE 00004

0070 0070 33L00000            LI                (FP1,FP2),0
0071 0071 34A00002            LI                BL,42
0072 0072 357000F5            LI                FL1,5
0073 0073 28U10079            B                 LOADC
0074 0074 75E0115F LADF       ADF               (78,18),(114,18),(78,18)
0075 0075 73L0835F
0076 0076 37200111
0077 0077 2CL00000
0078 0078 37L10761            LR     F,SAVF
0079 0079 76L00000 LOADC      LC                (14,18)
007A 007A 27L0AEFD
0078 0078 30U4A000            SR                C,INDATA(DP),2
007C 007C 76U00000            LC                (78,18)
007D 007D 27L2AEFD
007E 007E 30U4A000            SR                C,INDATA(DP),2
007F 007F 01E00000            GEN,32     X'01E00000'
0080 0080 01610001            DECR              FL1
0081 0081 28910079            BNZ,FL1           LOADC
0082 0082 01030001            DECR              BL
0083 0083 01E00000            GEN,32     X'01E00000'
0084 0084 357000F5            LI                FL1,5
0085 0085 31L10761            SR      F,SAVF
0086 0086 29110074            BNZ,BL            LADF
0087 0087 01E00000            GEN,32     X'01E00000'
0088 0088 01E00000            GEN,32     X'01E00000'
0089 0089 01E00000            GEN,32     X'01E00000'
008A 008A 01E00000            GEN,32     X'01E00000'
008B 008B 01A40001            DECR              FP3
008C 008C 34A00002            LI                BL,42
008D 008D 357000F5            LI                FL1,5
008E 008E 28010091            BZ,FP3            FPEND
008F 008F 31L10761            SR     F,SAVF
0090 0090 28U10074            B                 LADF
0091 0091 734000F0 FPEND      LI                FP2,0
0092 0092 00007741            CLR               Y
0093 0093 3ELF0094            RPT,240
0094 0094 1B000002            S                 Y,FP2+
                         *     ARRAY LOADING  PROCEDURE
0095 0095 38U08006            MVSG,PAGE1        PG1
0096 0096 28U00095
0097 0097 38U007F0
0098 0098 38U4009A
0099 0099 28U1N200            B                 PAGE1
009A 009A C13C02F0 PG1        STRTSG,PAGE1
0098 0200 36B10760 BTO        LR                (BL,DP),SAVBL
009C 0201 29U101DB            BZ,BL             EXT
009D 0202 01030001            DECR              BL
009E 0203 30B10760            SR                (BL,DP),SAVBL
009F 0204 42F688A5            L                 X,MK2
00A0 0205 42FA8845            L                 Y,MK6
00A1 0206 40U09943            LOR               Y,X
00A2 0207 42F888A5            L                 X,MK7
00A3 0208 40U09943            LOR               Y,X
00A4 0209 1AF00002            S                 Y,253
00A5 020A 72B00000            LI                DP,0
00A6 020B 43F86640 RET        STEP
00A7 020C 08U00002            L    M,Y
00A8 020D 36B4A000            LR                C,INDATA(DP),2
```

Figure A-11 - Printout of Interpolation Program
(Sheet 4 of 10)

```
00A9 020E 40000881          SCW             (14,18),F1
00AA 020F 4FL0A0BF
00AB 0210 42L08840
00AC 0211 40F08E5A
00AD 0212 40FE885A
00AE 0213 57L00002
00AF 0214 08U00003
00B0 0215 36U4A000          LR              C,INDATA(DP),2
00B1 0216 40U0881           SCW             (14,18),F3
00B2 0217 4FL1A4AF
00B3 0218 422088840
00B4 0219 40F8885A
00B5 021A 40FE8852
00B6 021B 4006885A
00B7 021C 57L00002
00B8 021D 48U00003
00B9 021E 00008841          L     Y,M
00BA 021F 28F1020B          BRS             RET
                       *    WITHIN ARRAY  COMPUTATIONS
00BB 0220 4006BBA1          SET             M
00BC 0221 48U00003
00BD 0222 75L01111          MPF     F1,F1,(160,36)
00BE 0223 73L011C3
00BF 0224 33201100
00C0 0225 2CU00000
00C1 0226 379023B5          MVF     (164,18),F2
00C2 0227 3F11022A
00C3 0228 433488A5
00C4 0229 4BB00001
00C5 022A 13A40003
00C6 022B 75L01111          MPF     F1,F2,(160,36)
00C7 022C 73L023C3
00C8 022D 33201100
00C9 022E 2CU00000
00CA 022F 37504785          MVF     (164,18),F4
00CB 0230 3F110233
00CC 0231 433488A5
00CD 0232 4BB00001
00CE 0233 13A40003
00CF 0234 75L01111          MPF     F1,F4,(160,36)
00D0 0235 73L047C3
00D1 0236 33201100
00D2 0237 2CU00000
00D3 0238 37505985          MVF     (164,18),F5
00D4 0239 3F11023C
00D5 023A 433488A5
00D6 023B 4BBF0001
00D7 023C 13A40003
00D8 023D 75L01111          MPF     F1,F3,(160,36)
00D9 023E 73L035C3
00DA 023F 33201100
00DB 0240 2CU00000
00DC 0241 379077C3          MVF     (166,30),F6
00DD 0242 3F1D0245
00DE 0243 433488A5
00DF 0244 4BB00001
00E0 0245 13A40003
00E1 0246 75L01123          MPF     F2,F3,(160,36)
```

Figure A-11 - Printout of Interpolation Program
(Sheet 5 of 10)

```
00E2 0247 73L035C3
00E3 0248 3320113D
00E4 0249 2C000000
00E5 024A 375095C3          MVF    (166,30),F7
00E6 024B 3F1D024E
00E7 024C 433488A5
00E8 024D 4B000001
00E9 024E 13A40003
                     *      COMPUTE THE  SUM OF ALL TERMS
00EA 024F 32000003          LI              DP,3
00EB 0250 35A00EA RTCS      LI              FP3,234
00EC 0251 33400048          LI              FP2,72
00ED 0252 41640001 CS1      DECR            FP2
00EE 0253 03408845          L               Y,FP2
00EF 0254 36020018          LR              C,SUBRT(DP)
00F0 0255 30010603          LRR             R3,C
00F1 0256 2C480000          BAL,R4          0(R3)
00F2 0257 4B000001          SM              Y,FP3-
00F3 0258 13A40002
00F4 0259 28510252          BNZ,FP2         CS1
00F5 025A 75E011EA          ADF             (217,18),F4,F4
00F6 025B 73L04747
00F7 025C 37201111
00F8 025D 2C000000
00F9 025E 75E011D8          ADF             (199,18),F3,F3
00FA 025F 73L03535
00FB 0260 37201111
00FC 0261 2C000000
00FD 0262 75E011C6          ADF             (181,18),F2,F2
00FE 0263 73L02323
00FF 0264 37201111
0100 0265 2C000000
0101 0266 75E011B4          ADF             (163,18),F1,F1
0102 0267 73L01111
0103 0268 37201111
0104 0269 2C000000
0105 026A 40000BA1          SET             M
0106 026B 08000003
0107 026C 33400095          LI              FP2,149
0108 026D 35A00EA           LI              FP3,234
0109 026E 34A0004E          LI              BL,78
010A 026F 41630001 CS2      DECR            BL
010B 0270 03648845          L               Y,FP2-
010C 0271 36020018          LR              C,SUBRT(DP)
010D 0272 30010603          LRR             R3,C
010E 0273 2C480000          BAL,R4          0(R3)
010F 0274 4B000001          SM              Y,FP3-
0110 0275 13A40002
0111 0276 2911026F          BNZ,BL          CS2
0112 0277 75E01DEA          ADF             (205,30),F7,F7
0113 0278 73L09595
0114 0279 37201D1D
0115 027A 2C000000
0116 027B 75E01DCC          ADF             (175,30),F6,F6
0117 027C 73L07777
0118 027D 37201D1D
0119 027E 2C000000
011A 027F 75E011AE          ADF             (157,18),F5,F5
```

Figure A-11 - Printout of Interpolation Program (Sheet 6 of 10)

```
011B 0280 73L05959
011C 0281 37201111
011D 0282 2C000000
011E 0283 2816A001          DECR            DP
011F 0284 29510250          BNZ,DP          RTCS
                      *     GENERATE FIRST WORD OF EACH GROUP IN THE OTHER WO
0120 0285 33400096          LI              FP2,150
0121 0286 41640001 GEN1     DECR            FP2
0122 0287 42168845          L               Y,MK2
0123 0288 43402245          LAND            Y,FP2
0124 0289 40C088A2          L               X,Y
0125 028A 40FF8883          ROT             X,1
0126 028B 40C09943          LOR             Y,X
0127 028C 40FF8883          ROT             X,1
0128 028D 40C09943          LOR             Y,X
0129 028E 40008882          L               X,Y
012A 028F 40FF8888          ROT             X,3
012B 0290 40FC8888
012C 0291 40009943          LOR             Y,X
012D 0292 1B400002          S               Y,FP2
012E 0293 28510286          BNZ,FP2         GEN1
                      *     GENERATE  ARRAY ELEMENTS FOR MULTIPLICATIONS
012F 0294 48180005          L               M,MK4
0130 0295 37903511          MVF             F1,F3
0131 0296 3F110299
0132 0297 435488A5
0133 0298 4B000001
0134 0299 13A40003
0135 029A 48170005          L               M,MK3
0136 029B 37903523          MVF             F2,F3
0137 029C 3F11029F
0138 029D 435488A5
0139 029E 4B000001
013A 029F 13A40003
013B 02A0 42188845          L   Y,MK7
013C 02A1 40008881          LAND    M,Y
013D 02A2 40002282
013E 02A3 48000003
013F 02A4 37904723          MVF             F2,F4
0140 02A5 3F1102A8
0141 02A6 435488A5
0142 02A7 4B000001
0143 02A8 13A40003
0144 02A9 48190005          L               M,MK5
0145 02AA 37902347          MVF             F4,F2
0146 02AB 3F1102AE
0147 02AC 435488A5
0148 02AD 4B000001
0149 02AE 13A40003
014A 02AF 481A0005          L               M,MK6
014B 02B0 37902311          MVF             F1,F2
014C 02B1 3F1102B4
014D 02B2 435488A5
014E 02B3 4B000001
014F 02B4 13A40003
0150 02B5 48150005          L               M,MK1
0151 02B6 37904759          MVF             F5,F4
0152 02B7 3F1102BA
```

Figure A-11 - Printout of Interpolation Program
(Sheet 7 of 10)

```
0153 0288 435488A5
0154 0289 4B600001
0155 028A 13A40003
0156 028B 40608BA1          SET     M
0157 028C 48600003
0158 028D 00E07741          CLR             Y
0159 028E 334000C8          LI              FP2,200
015A 028F 3E1D02C0          RPT,30
015B 02C0 18000002          S       Y,FP2+
015C 02C1 48F60005          L               M,MK4
015D 02C2 376DE577          MVF             F6,(200,30)
015E 02C3 3F1D02C6
015F 02C4 435488A5
0160 02C5 46000001
0161 02C6 13A40003
0162 02C7 48F70005          L               M,MK3
0163 02C8 376DE595          MVF             F7,(200,30)
0164 02C9 3F1D02CC
0165 02CA 435488A5
0166 02CB 4B600001
0167 02CC 13A40003
0168 02CD 42F98845          L               Y,MK5
0169 02CE 42FA88A5          L               X,MK6
016A 02CF 40600C43          LXOR            Y,X
016B 02D0 42F588A5          L               X,MK1
016C 02D1 40600C43          LXOR            Y,X
016D 02D2 48600002          L       M,Y
016E 02D3 376D0735          MVF     (38,16),(200,16)
016F 02D4 3F6F02D7
0170 02D5 435488A5
0171 02D6 4B600001
0172 02D7 13A40003
0173 02D8 34200001          LI              CH,X'0001'
0174 02D9 72604000          LI              CL,X'4000'
0175 02DA 33600F35          SC      (14,16),F3
0176 02DB 3F1102DE
0177 02DC 40600A73
0178 02DD 48400001
0179 02DE 13740003
017A 02DF 40608BA1          SET             M
017B 02E0 48600003
017C 02E1 37900012          MVF             (18,54),(0,54)
017D 02E2 3F3502E5
017E 02E3 435088A5
017F 02E4 48600001
0180 02E5 13A00003
0181 02E6 376053E5          MVF             (200,30),(54,30)
0182 02E7 3F1D02EA
0183 02E8 435488A5
0184 02E9 4B600001
0185 02EA 13A40003
                      *     MULTIPLY  ARRAY ELEMENTS
0186 02EB 75E01111          MPF     F1,F3,(90,36)
0187 02EC 73E03570
0188 02ED 33201100
0189 02EE 2C000000
018A 02EF 75E01123          MPF     F2,(90,22),(120,40)
018B 02F0 73E06F9F
```

Figure A-11 - Printout of Interpolation Program (Sheet 8 of 10)

```
018C 02F1 33201500
018D 02F2 2C000000
018E 02F3 75L01D53          MPF    (54,30),(90,22),(160,52)
018F 02F4 73L06FD3
0190 02F5 33201500
0191 02F6 2C000000
                     *      COMPUTE SUMS OF NUMERATOR AND DENOMINATOR ELEMENT
0192 02F7 48FC0005          L            M,MK8
0193 02F8 75L033D3          MVNF     (160,52),(160,52)
0194 02F9 33400FD3
0195 02FA 2C010000
0196 02FB 75L0279F          MVNF     (120,40),(120,40)
0197 02FC 33400F9F
0198 02FD 2C010000
0199 02FE 40008BA1          SET              M
019A 02FF 08000003
019B 0300 32000003          LI               DP,3
019C 0301 33400FD4 COMS2    LI   FP2,212
019D 0302 35A00F5C          LI     FP3,92
019E 0303 01040001 CS3      DECR    FP2
019F 0304 41A40001          DECR     FP3
01A0 0305 03408845          L                Y,FP2
01A1 0306 36020018          LR               C,SUBRT(DP)
01A2 0307 30010603          LRR              R3,C
01A3 0308 2C4B0000          BAL,R4           0(R3)
01A4 0309 48000001          SM               Y,FP3
01A5 030A 13000002
01A6 030B 28710303          BNZ,FP3     CS3
01A7 030C 75L02727          ADF      (0,40),(120,40),(120,40)
01A8 030D 73L09F9F
01A9 030E 37202727
01AA 030F 2C000F00
01AB 0310 75L03358          ADF      (40,52),(160,52),(160,52)
01AC 0311 73L0D3D3
01AD 0312 37203333
01AE 0313 2C000000
01AF 0314 28160F01          DECR             DP
0180 0315 29510301          BNZ,DP           COMS2                    F1 = NUM  F2
                     *      PERFORM  DIVIDE TO COMPUTE THE RESULTS
01B1 0316 379037C1          DVF    (160,34),(120,22),(0,56)
01B2 0317 3F21031A
01B3 0318 03548BA5
01B4 0319 0B000001
01B5 031A 13A40003
01B6 031B 3F140319
01B7 031C 0B000F01
01B8 031D 13A40003
01B9 031E 75L01401
01BA 031F 73L00078
01BB 0320 77202117
01BC 0321 34A08D15
01BD 0322 2C000000
01BE 0323 4A000001
01BF 0324 42004445
01C0 0325 12000002
                     *      UNLOAD THE RESULTS
01C1 0326 72000000          LI               DP,0
01C2 0327 42F68845          L                Y,MK2
```

Figure A-11 - Printout of Interpolation Program
(Sheet 9 of 10)

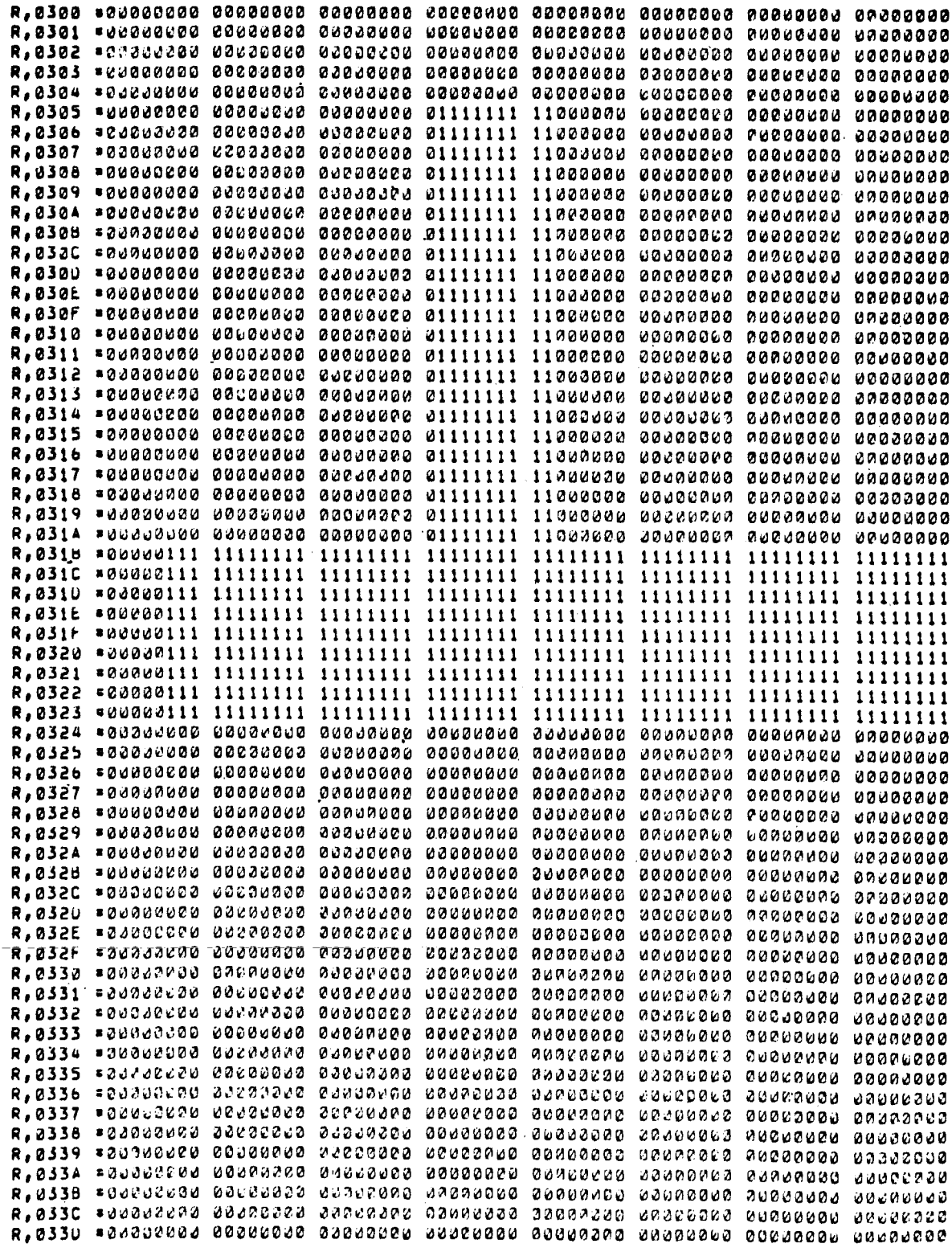


Figure B-1 - Test Pattern of Horizontal & Vertical Intersecting Lines

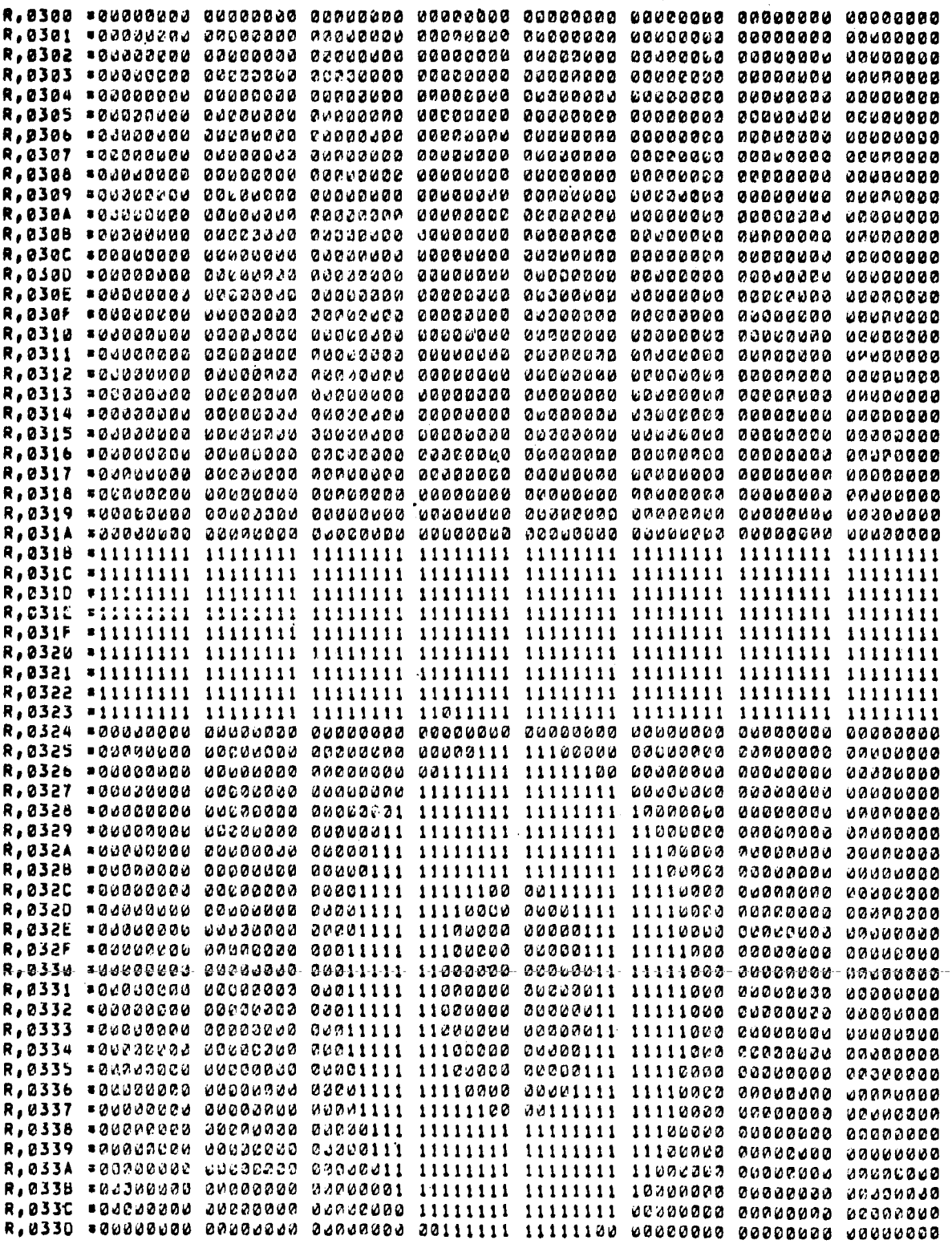


Figure B-2 - Test Pattern of Circular and Horizontal Line

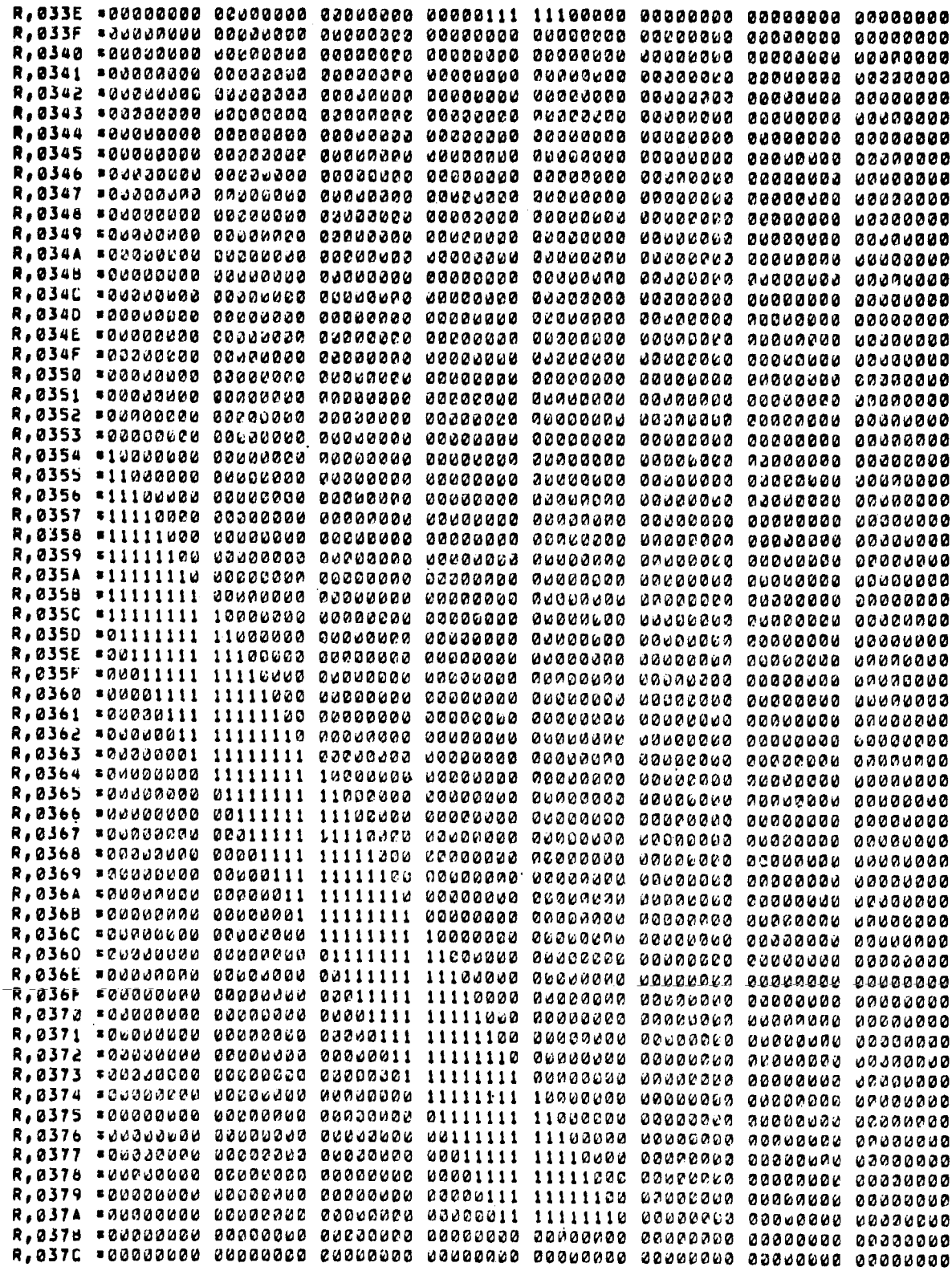


Figure B-3 - Diagonal Line Test Pattern

thinning has been performed. The thinned line is right at the centerline of the original lines. The THIN subroutine shown on sheets 4, 5, 6, 7 of B-12 was used and contains 194 assembly instructions. Figure B-5 shows the 'thinned' circle of Figure B-2 which was assumed to be a stringent test of the orthogonally implemented AAP line thinning algorithm. The cells of the thinned circle are within one bit position of the anticipated result.

The thinned diagnonal line in Figure B-6 has a slight degree of bending at the ends. However, this is truly what this thinning process should do since the diagonal line was not square at the ends. The "non-square" line endings could occur when the overlay lines that cross array boundaries are not perpendicular to the boundary. Therefore, to eliminate "pips" at the array boundaries of the final mosaicked map, each array load should contain sufficient overlap with its adjacent neighbors to permit deletion of the distorted boundary areas.

4.2 Single Line Symbol Generation

The subroutine "THIK9" is used to thicken previously thinned lines. This routine contains 49 instructions and is shown on sheets 7-11 of Figure B12. Figures B-7, B-8, and B-9 show the horizontal and vertical intersecting lines, the circle, and the diagonal line after the thickening routine was utilized.

The small distortion on the left hand edge of Figure B-7 is because these bit columns are used as part of the scratch (working storage) area. Figure B-8 is slightly out of position (up shifted) with respect to the original array image. This is part of the single array technique which restores each thickened line (array word) a few lines higher in the array so that the lines (array words) which have not been thickened at that time are not overwritten. The distortion in the thickened circle is due to the distortion introduced during the line thinning process.

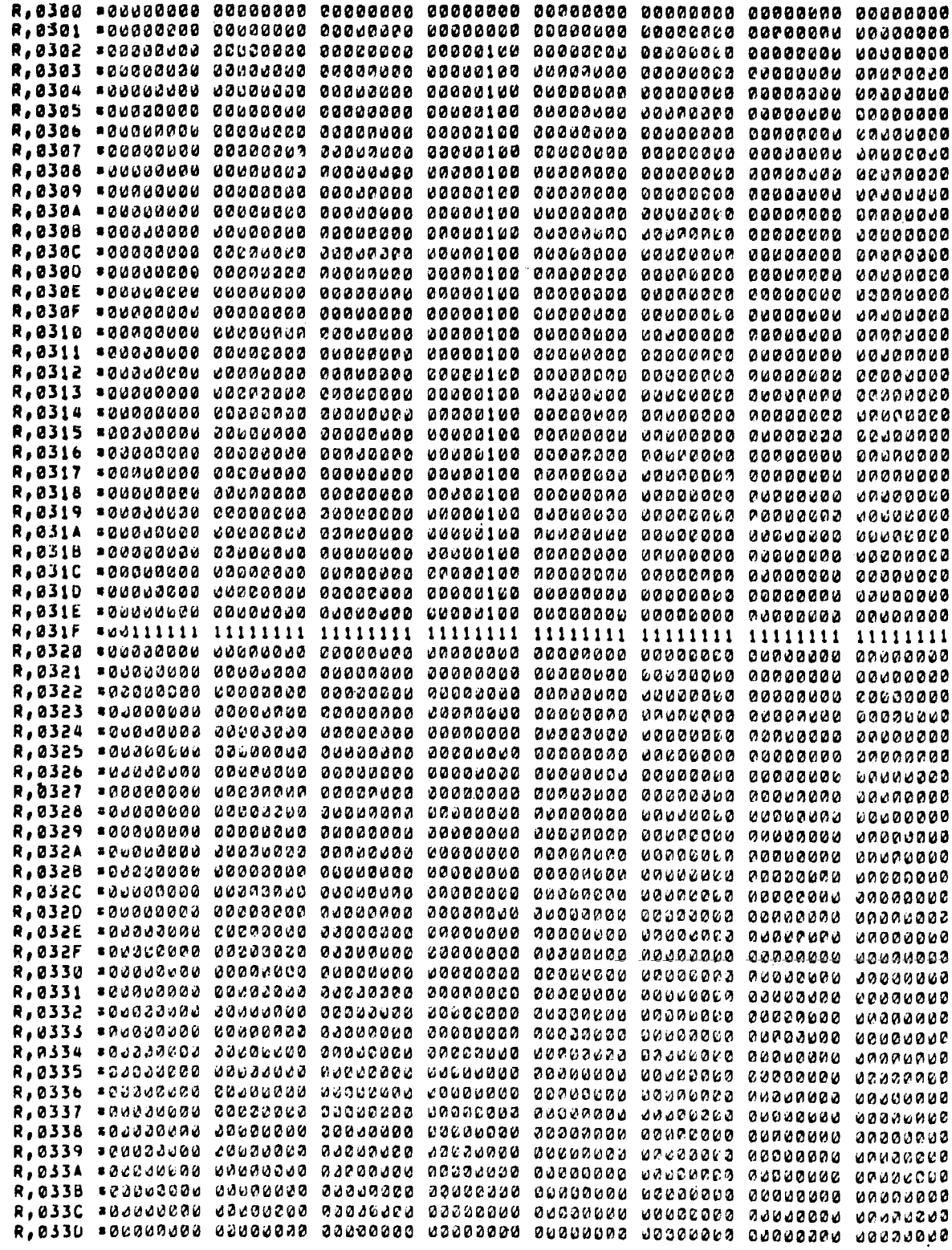


Figure B-4 - Thinned Horizontal and Vertical Lines

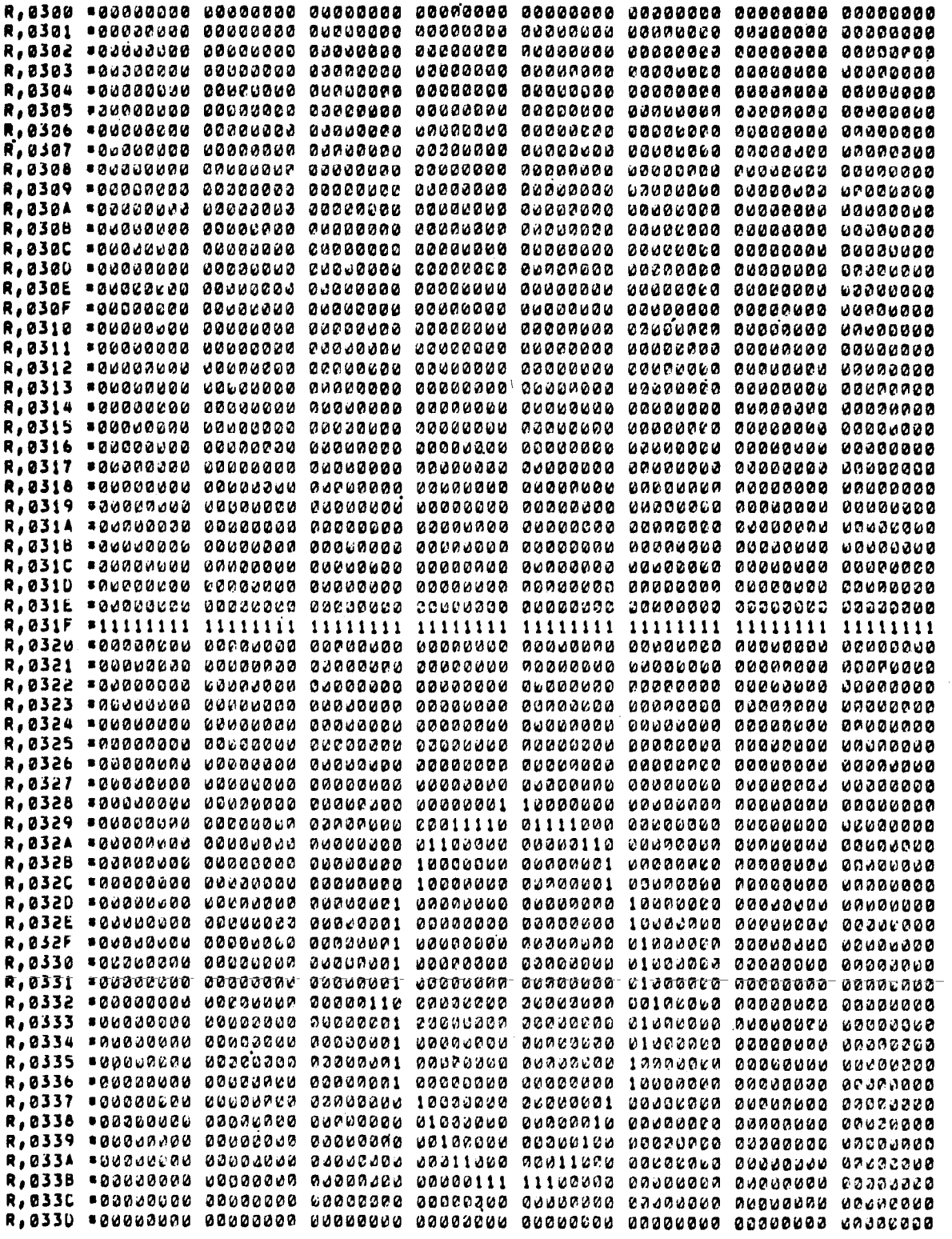


Figure B-5 - Thinned Circle and Horizontal Line

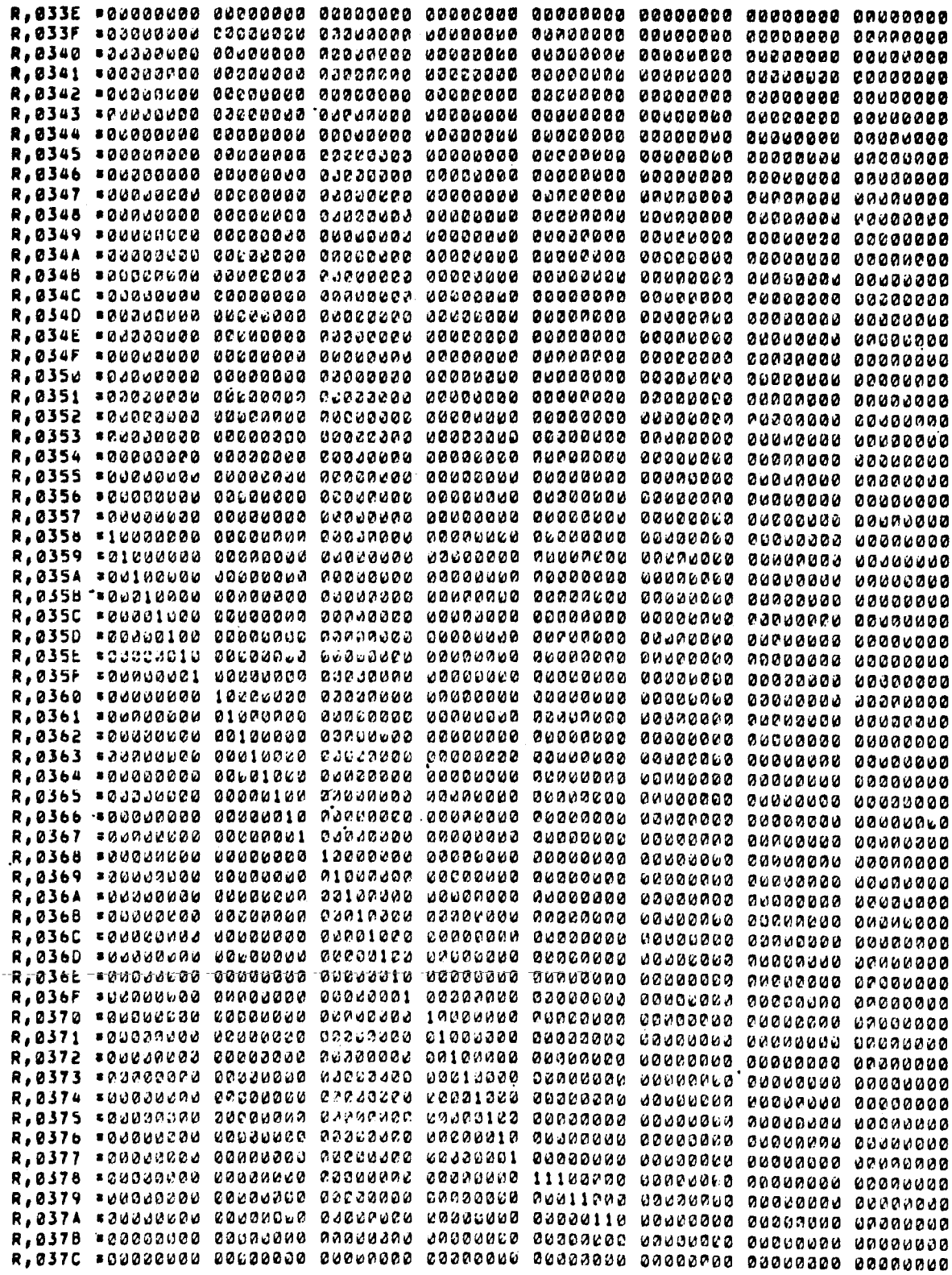


Figure B-6 - Thinned Diagonal Line

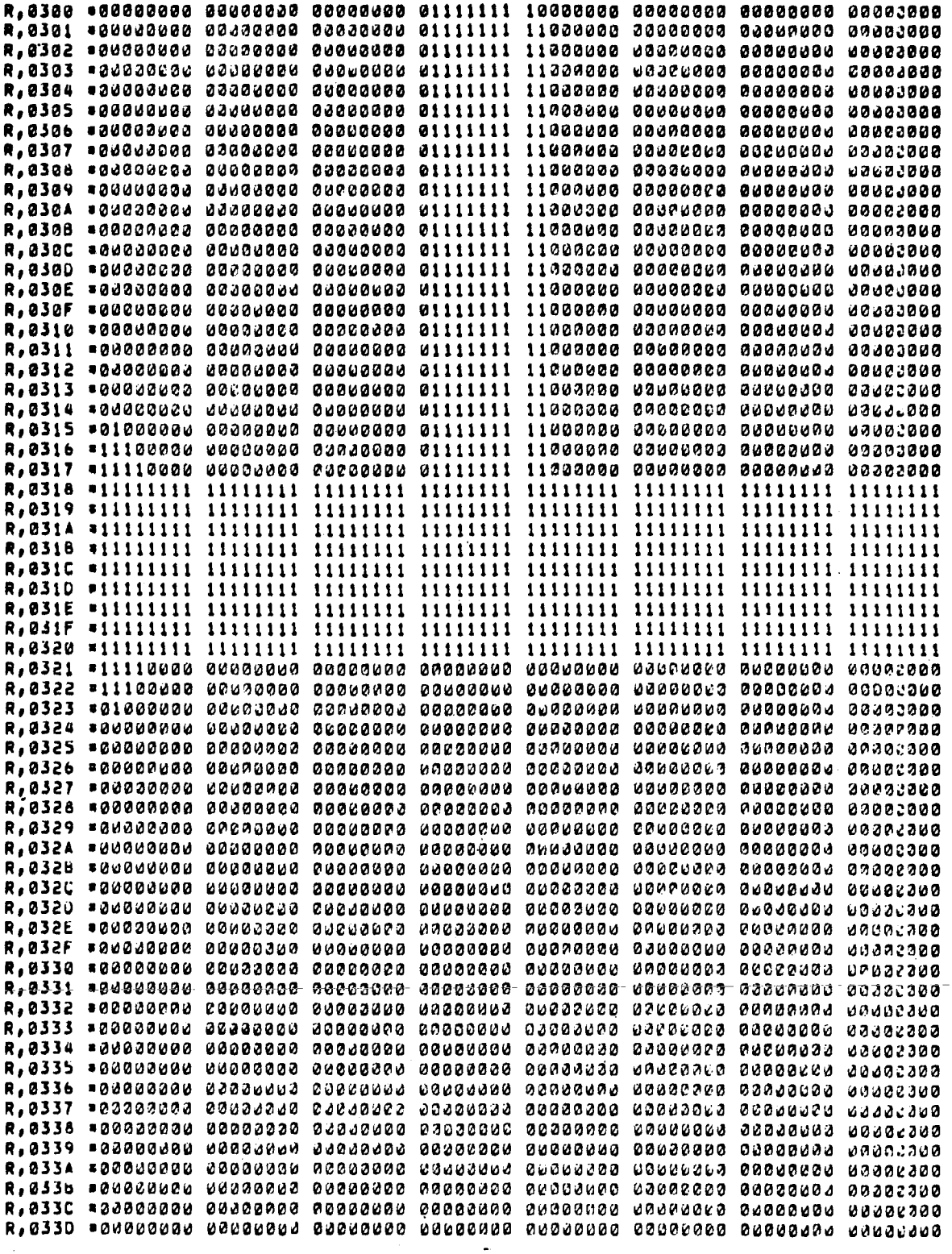


Figure B-7 - Nine Cell Thick Single Line Symbol for Horizontal & Vertical Lines

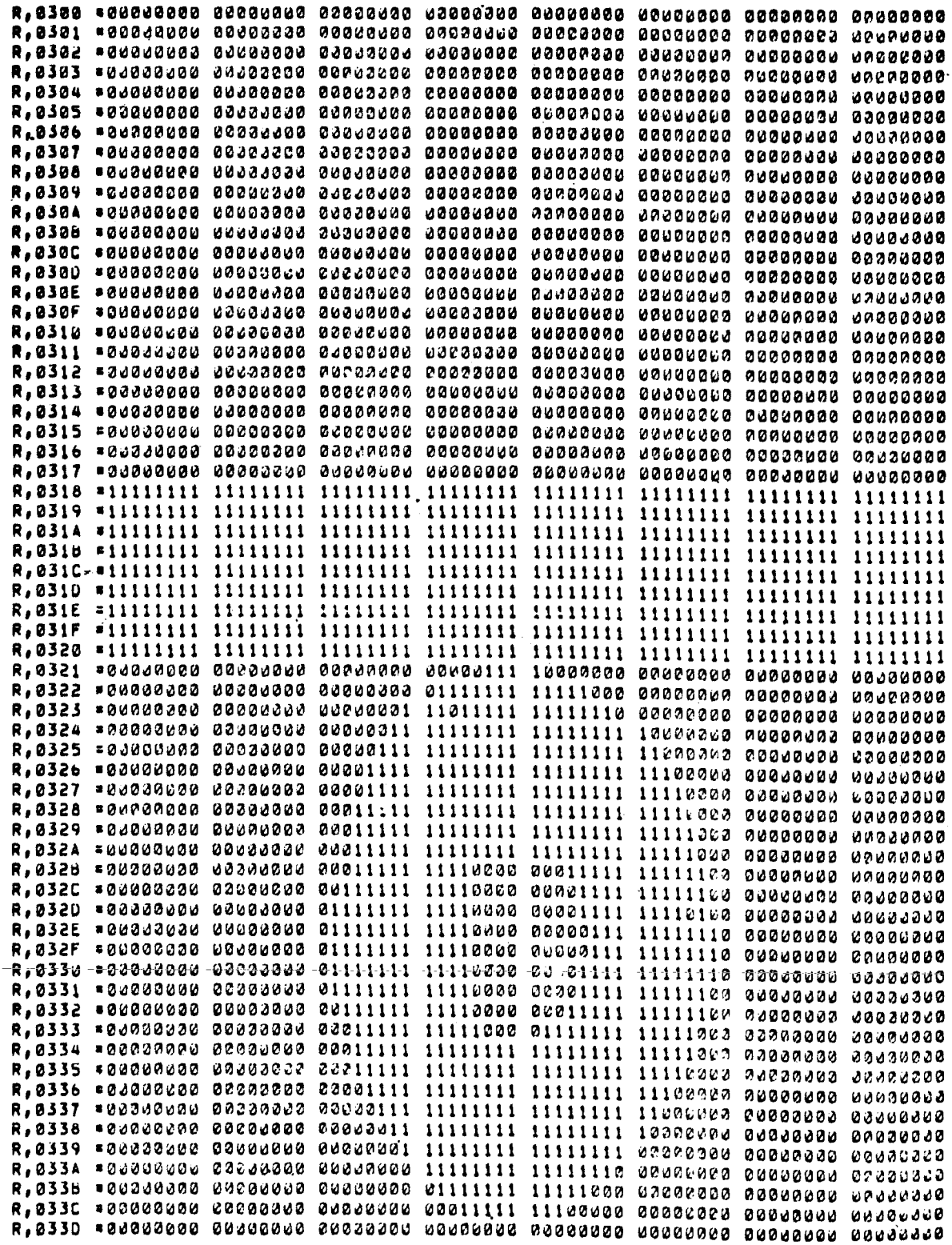


Figure B-8 - Nine Cell Thick Single Line Symbol for Circle & Horizontal Line

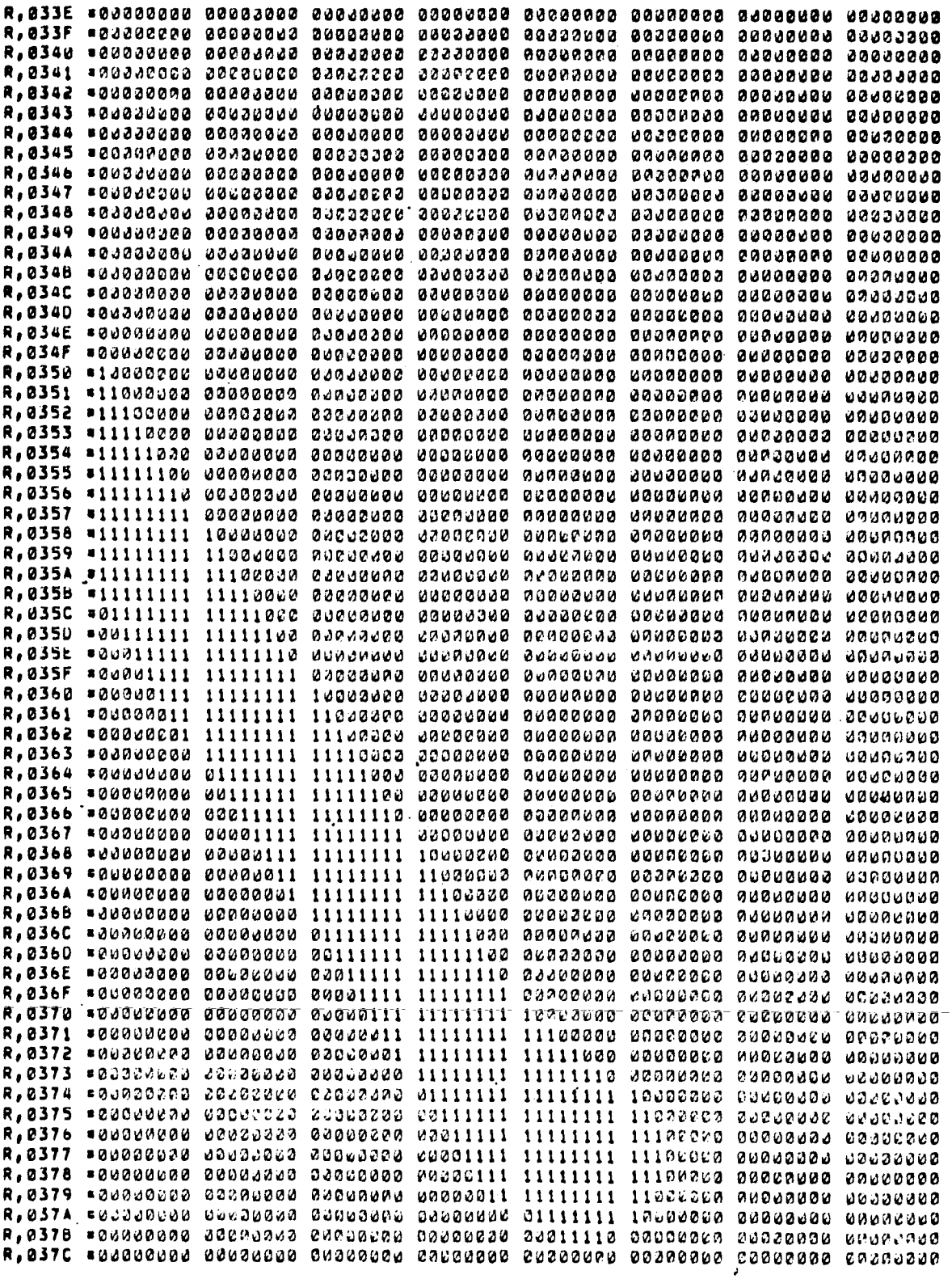


Figure B-9 - Nine Cell Thick Single Line Symbol for Diagonal Line

4.3 Double Line Symbol Generation

The "DRILL" subroutine is used in conjunction with the Line Thickening routines THIK9, THIK5, THIK3, and various array manipulations and array-to-array transfer routines to produce Figures B-10 and B-11. Routine THIK9 is used to generate the 9 bit center line which is stored in another array for later use by using the array-to-array transfer routine. Routines THIK5 and THIK3 are used to generate a 27 bit wide line in the lower half of the array. The 9 bit center line is transferred (from its storage location in the other array) to the top half of the same array. The top half of the array is then merged (word by word) with the bottom half of the array utilizing the "exclusive OR" capability of the AAP. This is the "DRILL" process and it is repeated N times, N being the number of array words containing the array image. Sheets 11-15 (Figure B-12) contain this routine.

The approach for the doubleline symbol generation implemented above uses an additional array for temporary storage which facilitated the validation effort. However, the study effort approach which utilizes one array is still a very viable technique.

5. VALIDATION PROGRAM PRINTOUTS

The Raster Processing validation program printouts are shown in Figure B-12.

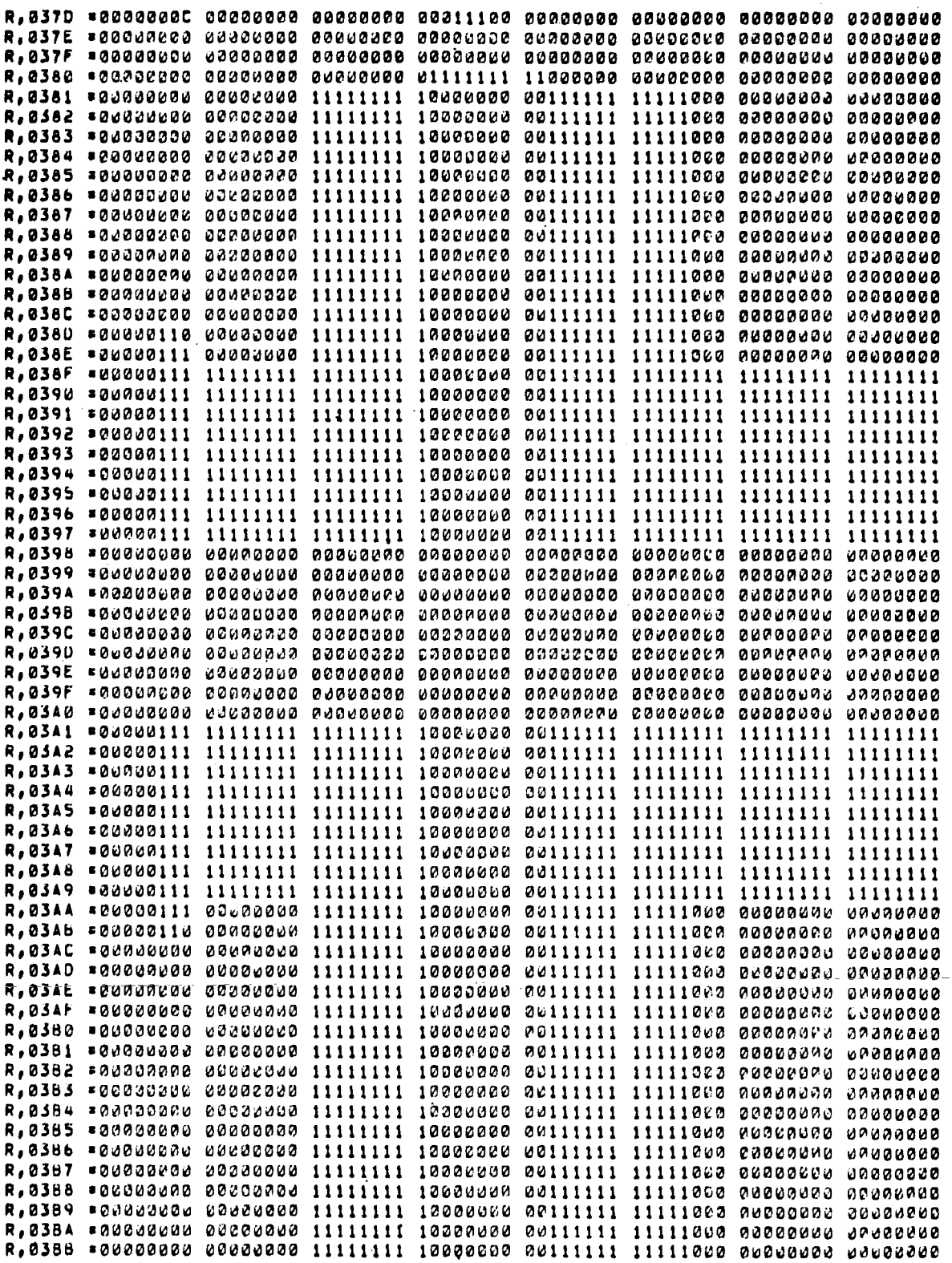


Figure B-10 - Double Line Symbol for a
Horizontal & Vertical Line Intersection

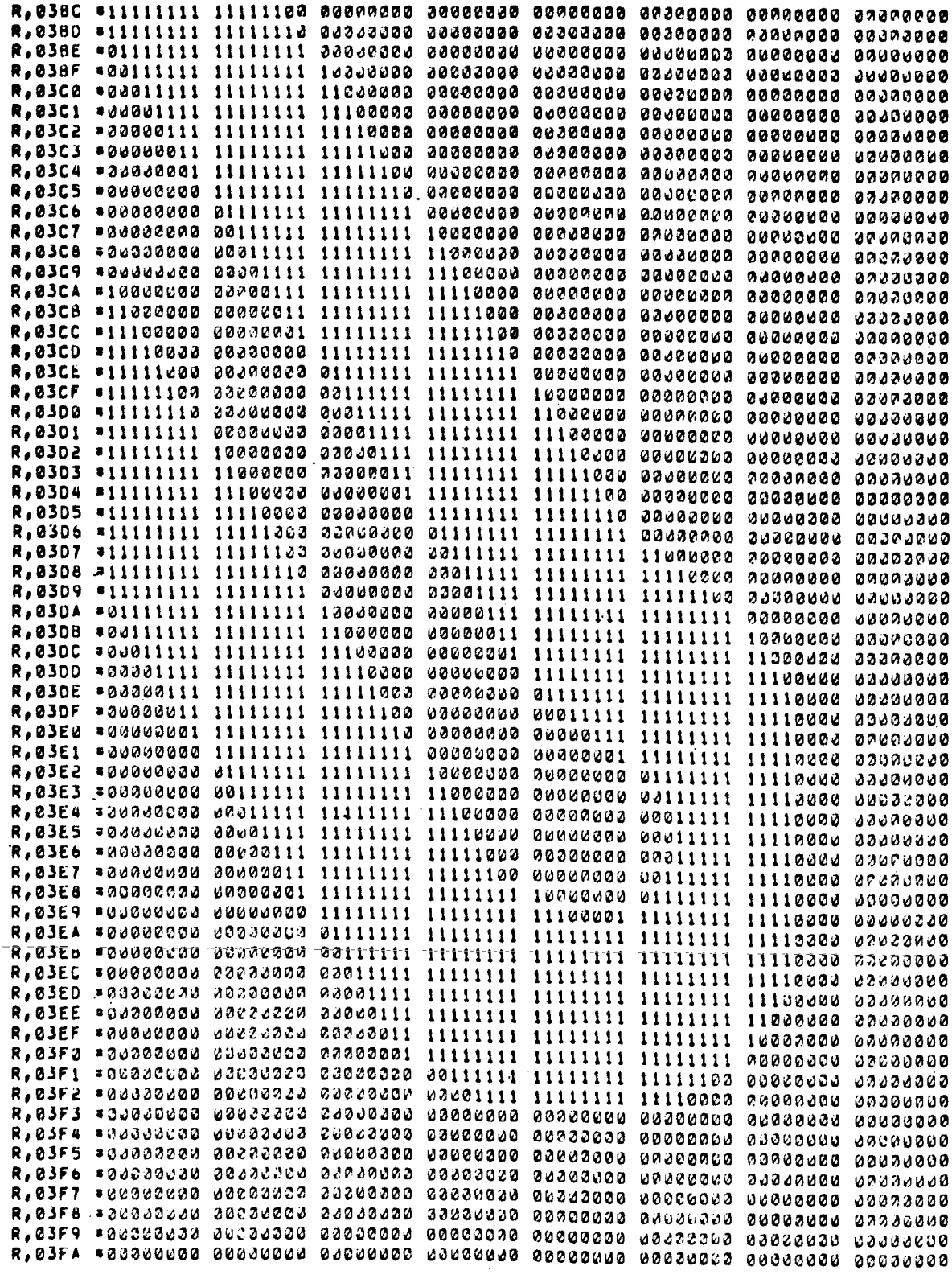


Figure B-11 - Double Line Expansion for a Diagonal Line

```
                        ******      ********      **********      ***********  ***
9048 9048 340n1f00          LI      ASH,X'1000'
9049 9049 73y00000          LI      FP1,0
904A 904A 00Ln7741          CLR     Y
904B 904B 3EFF9F4C          RPT,256
904C 904C 1B3nn002          S       Y,FP1+
904D 904D 38Un2000          WAIT
904E 904E 00Lx0001 TESTPAT  NOP
904F 904F 340nF000          LI      ASH,X'F000'
9050 9050 73y00000          LI      FP1,0
9051 9051 nnn07741          CLR     Y
9052 9052 3EFF9053        RPT,256
9053 9053 1B3n0002         S     Y,FP1+
                        ******      ********      **********      ***********  ***
9054 9054 340n2000          LI    ASH,X'2000'
9055 9055 28L19080          B       DDND
9056 9056 33y00002          LI      FP1,2
9057 9057 33400025          LI    FP2,37
9058 9058 76L00007          LI    C,7
9059 9059 40L088A1          SCW     (0,32),(160,32)
905A 905A 4FL5A0FF
905B 905B 57L50000
905C 905C 08U00003
905D 905D 01Un0001        INCR    FP2
905E 905E 76L0003F          LI    C,X'003F'
905F 905F 40L088A1          SCW     (0,32),(160,32)
9060 9060 4FL5A0FF
9061 9061 57L50000
9062 9062 08L00003
9063 9063 01Un0001        INCR    FP2
9064 9064 76L00FF           LI    C,X'00FF'
9065 9065 40L088A1          SCW     (0,32),(160,32)
9066 9066 4FL5A0FF
9067 9067 57L50000
9068 9068 08U00003
9069 9069 01Un0001        INCR    FP2
906A 906A 76L001FF          LI    C,X'01FF'
906B 906B 40L088A1          SCW     (0,32),(160,32)
906C 906C 4FL5A0FF
906D 906D 57L50000
906E 906E 08U00003
906F 906F 01Un0001        INCR    FP2
9070 9070 76L003FF          LI    C,X'03FF'
9071 9071 40L088A1          SCW     (0,32),(160,32)
9072 9072 4FL5A0FF
9073 9073 57L50000
9074 9074 08U00003
9075 9075 01Un0001        INCR    FP2
9076 9076 76L007FF          LI    C,X'07FF'
9077 9077 40L088A1          SCW     (0,32),(160,32)
9078 9078 4FL5A0FF
9079 9079 57L50000
907A 907A 08U00003
9078 9078 01Un0001        INCR    FP2
907C 907C 76L007FF          LI    C,X'07FF'
907D 907D 40L088A1          SCW     (0,32),(160,32)
907E 907E 4FL5A0FF
907F 907F 57L50000
```

Figure B-12 - Raster Processing Validation Program Printouts (Sheet 1 of 14)

```
                APPLE V01-00 04-DEC-00 15:48:26 PAGE 00015

0080 0080 08000003
0081 0081 01000001          INCR   FP2
0082 0082 7600FFC           LI     C,X'0FFC'
0083 0083 4000881           SCW      (0,32),(160,32)
0084 0084 4F05A0FF
0085 0085 5700000
0086 0086 08000003
0087 0087 01000001          INCR   FP2
0088 0088 7600FF0           LI     C,X'0FF0'
0089 0089 4F00881           SCW      (0,32),(160,32)
008A 008A 4F05A0FF
008B 008B 57000000
008C 008C 08000003
008D 008D 01000001          INCR   FP2
008E 008E 7600FE0           LI     C,X'0FE0'
008F 008F 4000881           SCW      (0,32),(160,32)
0090 0090 4F05A0FF
0091 0091 57000000
0092 0092 08000003
0093 0093 01000001          INCR   FP2
0094 0094 7601FE0           LI     C,X'1FE0'
0095 0095 4001881           SCW      (0,32),(160,32)
0096 0096 4F05A0FF
0097 0097 57000000
0098 0098 08000003
0099 0099 01000001          INCR   FP2
009A 009A 7601FC0           LI     C,X'1FC0'
009B 009B 4000881           SCW      (0,32),(160,32)
009C 009C 4F05A0FF
009D 009D 57000000
009E 009E 08000003
009F 009F 01000001          INCR   FP2
00A0 00A0 7601F80           LI     C,X'1F80'
00A1 00A1 4001881           SCW      (0,32),(160,32)
00A2 00A2 4F05A0FF
00A3 00A3 57000000
00A4 00A4 08000003
00A5 00A5 33000080      LI   FP1,191
00A6 00A6 33400040      LI  FP2,192
00A7 00A7 3F1E900A        LOOP,31   DONE
00A8 00A8 43040845          L     Y,FP1-
00A9 00A9 5B000002          S     Y,FP2+
00AA 00AA 00000001 DONE     NOP
00AB 00AB 33000031          LI    FP1,49
00AC 00AC 33400032          LI    FP2,50
00AD 00AD 3F1E9080        LOOP,31   DOND
00AE 00AE 47040845       L,W    Y,FP1-
00AF 00AF 5F002002       S,W    Y,FP2+
00B0 00B0 00000001 DOND     NOP
                        *  HORT LINE
00B1 00B1 00000841       SET    Y
00B2 00B2 33000018          LI     FP1,27
00B3 00B3 3E00900B          RPT,9
00B4 00B4 1F002002          S,W    Y,FP1+
00B5 00B5 00000001          NOP
                        * VERT  LINE
00B6 00B6 33000019       LI     FP1,25
00B7 00B7 3E00900B          RPT,9
```

Figure B-12 - Raster Processing Validation Program Printouts

(Sheet 2 of 14)

```
APPLE V01-00 04-DEC-00 15:48:06 PAGE 00016

0088 0088 18B00002          S      Y,FP1+
                         *  45 DEGREE LINE
0089 0089 00B07741          CLR  Y
008A 008A 33B00002             LI     FP1,2
008B 008B 33400001          LI   FP2,1
008C 008C 76B001FF             LI   C,X'01FF'
008D 008D 41L08840             SCW,X'8840'    Y(0)
008E 008E 00B08E52             ROT   Y,128
008F 008F 33B00053             LI     FP1,83
00C0 00CW 3F279DC3             LOOP,40    DONSL
00C1 00C1 5F3001W2             S,W    Y,FP1+
00C2 00C2 401F8852             ROT   Y,1
00C3 00C3 00B00001 DONSL    NOP
                         *******  *****   ***** ******   ****      *****
                         *          SHIFT ROUTINE
00C4 00C4 33B00023             LI     FP1,35
00C5 00C5 3F3F90CA             LOOP,96  SHIFTIT
00C6 00C6 47B08845             L,W    Y,FP1
00C7 00C7 40L08852             ROT    Y,160
00C8 00C8 40B08852
00C9 00C9 5F3001W2             S,W    Y,FP1+
00CA 00CA 00B00001 SHIFTIT  NOP
                         *******  *****   ***** ******   ****      *****
                         *  FRAME ROUTINE
00CB 00CB 00B07741             CLR    Y
00CC 00CC 33B00000             LI     FP1,0
00CD 00CD 3EB4900E             RPT,5
00CE 00CE 18B00002             S      Y,FP1+
00CF 00CF 33B000FA             LI     FP1,250
00D0 00D0 3EB49001             RPT,5
00D1 00D1 18B00002             S      Y,FP1+
00D2 00D2 33B00000             LI     FP1,0
00D3 00D3 3EB49004             RPT,5
00D4 00D4 1F300002             S,W    Y,FP1+
00D5 00D5 33B000FA             LI     FP1,250
00D6 00D6 3EB49007             RPT,5
00D7 00D7 1F300002             S,W    Y,FP1+
00D8 00D8 26LB0000             B    0(R3)
00D9 00D9 38002000             WAIT
               FFFF            END
```

Figure B-12 - Raster Processing Validation Program Printouts

(Sheet 3 of 14)

```
APPLE V01-00 04-DEC-00 15:48:06 PAGE 00001

                        START
0000 0000 40C000C01 THIN     NOP
0001 0001 40C0BBA1 COMMENCE SET      M
0002 0002 08C00003
0003 0003 33B00005          LI     FP1,5              1ST, N
0004 0004 33A00004        LI  FP2,4  N-1
0005 0005 35A00005      LI  FP3,5  NSAVE
                      *     N = THE LINE NUMBER WE ARE PRIMARILY CONCERNED WITH,
                      *      DOWN SWEEP
0006 0006 74A000F5          LI     BL,245
0007 0007 40C00001 DOWNS  NOP
0008 0008 47C088A5        L,W    X,FP1      N TO X
0009 0009 40FF6663          ROT    X,1                RT SHIFT N
000A 000A 473022A5       LAND,W  X,FP1+     N ANDED N SHIFED
000B 000B 40C088A3      L  Y,X         DUP
000C 000C 40FF888B          ROT    X,-1      LEFT SHIFT RESULT
000D 000D 40C088B8
000E 000E 40C099A2        LOR    X,Y      OR  NRS&NLS
000F 000F 48A00003     L  M,X  RESULTS TO MASK
0010 0010 474088E5            L,W    Y,FP2
0011 0011 47C088A5            L,W    X,FP1
0012 0012 40C0CCA2            LXOR       X,Y
0013 0013 40C02243            LAND       Y,X
0014 0014 40C088A2            L     X,Y
0015 0015 40FF8852            ROT    Y,1
0016 0016 473022A5            LAND,W     Y,FP1
0017 0017 40FF888B            ROT    X,-1
0018 0018 40C088B8
0019 0019 47C022A5            LAND,W     X,FP1
001A 001A 40C0CC43            LXOR   Y,X
001B 001B 40C099A1            LANDN      M,Y
001C 001C 40C066A2
001D 001D 48C00003
001E 001E 475488E5            L,W        Y,FP1+
001F 001F 47B4CC45            LXOR,W       Y,FP2+
0020 0020 40C088A1            LAND   M,Y
0021 0021 40C022A2
0022 0022 48C00003
0023 0023 5EC00001            S,W    M,A
0024 0024 40C044A1            LN   X,M
0025 0025 46C199A5            LOR,W  X,B
0026 0026 473022A5            LAND,W   X,FP1+
0027 0027 41B40001            DECR   FP2
0028 0028 5FC00003            S,W    X,FP2+
0029 0029 46C088A5            L,W    X,A
002A 002A 1EC1C003            S,W      X,B
002B 002B 41C00001            INCR   FP2
002C 002C 41C00001            INCR   FP2
002D 002D 41C30001            DECR   BL
002E 002E 29110007            BN7,BL   DOWNS
002F 002F 00C00001            NOP
                      *   LEFT  TO  RIGHT  SWEEP
0030 0030 33B00005            LI   FP1,5
0031 0031 33A00004          LI  FF2,4  N-1
0032 0032 35A00005        LI  FP3,5  NSAVE
0033 0033 74A00FF5            LI     BL,245
0034 0034 40C00001 LFRTS  NOP
0035 0035 43C088A5          L      X,FP1      N T
```

Figure B-12 - Raster Processing Validation Program Printouts

(Sheet 4 of 14)

```
0036 0036 4C1FFFFF3          ROT    X,1                          SEE
0037 0037 43JV22A5       LAND    X,FP1+     N ANDED N SHIFED
0038 0038 4CLVFF43      L  Y,X          DUP
0039 0039 4C1F8888           ROT    X,-1      LEFT SHIFT RESULT
003A 003A 4CUV8888
003B 003B 48L0994A2       LOR    X,Y      OR  NRS&NLS
003C 003C 48L00003     L  M,X  RESULTS TO MASK
003D 0C3D 43400845           L       Y,FP2
003E 003E 43LV8FA5           L     X,FP1
003F 003F 4CL0CCA2           LXOR        X,Y
0040 0040 40L02243           LAND        Y,X
0041 0041 48L0H8A2           L     X,Y
0042 0042 4C1F8852           ROT     Y,1
0043 0043 43L02245           LAND        Y,FP1
0044 0044 4C1F8888           ROT     X,-1
0045 0045 4CL08888
0046 0046 43L022A5           LAND        X,FP1
0047 0047 4CL0CCF43          LXOR     Y,X
0048 0048 4CU08BA1           LANDN       M,Y
0049 0049 4CL066A2
004A 004A 48L00003
004B 004B 43J48845           L           Y,FP1-
004C 004C 43J4CC45           LXOR           Y,FP2-
004D 004D 4CL08BA1           LAND     M,Y
004E 004E 4CUH22A2
004F 004F 48L0HF03
0050 0050 5AL00001           S   M,A
0051 0051 4CL044A1           LN       X,M
0052 0052 42L19FA5           LOR      X,B
0053 0053 43JV22A5           LAND        X,FP1+
0054 0054 41J4HH01           DECR     FP2
0055 0055 5BJH0003           S        X,FP2+
0056 0056 42L0BBA5           L        X,A
0057 0057 1AL1FF03           S          X,B
0058 0058 01J00001           INCR     FP2
0059 0059 01J00001           INCR     FP2
005A 005A 01L3CC01           DECR     BL
005B 005B 29110C34           BNZ,BL   LFRTS
005C 005C 00L00001           NOP
                        *   UP   SWEEP
005D 005D 33J0FFFA           LI     FP1,250
005E 005E 334HHFFB         LI   FP2,251  N=1
005F 005F 35AHCFFA       LI  FP3,250
0060 0060 74AHHF5            LI       BL,245
0061 0061 43J00001           NOP
0062 0062 4CL00001 UPS     NOP
0063 0063 47LV88A5         L,W     X,FP1      N TO X
0064 0064 43FFF663           ROT     X,1
0065 0065 473422A5       LAND,W  X,FP1-     N ANDED N SHIFED
0066 0066 40L08843       L  Y,X          DUP
0067 0067 4C1F8888           ROT    X,-1      LEFT SHIFT RESULT
0068 0068 4CL08B6B
0069 0069 4CLF99A2       LOR    X,Y      OR  NRS&NLS
006A 006A 48LFHF03       L  M,X  RESULTS TO MASK
006B 006B 474F8845           L,W     Y,FP2
006C 006C 47LF8BA5           L,W     X,FP1
006D 006D 4CL0CCA2           LXOR        X,Y
006E 006E 4CUV2243           LAND        Y,X
```

Figure B-12 - Raster Processing Validation Program Printouts
(Sheet 5 of 14)

```
APPLE V01-00 04-DEC-00 15:48:06 PAGE 00003

006F 006F 40L088A2          L     X,Y
0070 0070 4CFF8852          ROT     Y,1
0071 0071 47L02245          LAND,W      Y,FP1
0072 0072 40FF8888          ROT     X,-1
0073 0073 4CL08886
0074 0074 47L022A5          LAND,W      X,FP1
0075 0075 40L0CCC3          LXOR      Y,X
0076 0076 40L086A1          LANDN       M,Y
0077 0077 4CL0866A2
0078 0078 48L00003
0079 0079 47L05E45          L,W         Y,FP1
007A 007A 4704CC45          LXOR,W        Y,FP2-
007B 007B 40L058A1          LAND        M,Y
007C 007C 40L022A2
007D 007D 48L00003
007E 007E 5EL00001          S,W     M,A
007F 007F 40L044A1          LN      X,M
0080 0080 46L199A5          LOR,W     X,B
0081 0081 47L022A5          LAND,W      X,FP2+
0082 0082 41L00001          INCR      FP2
0083 0083 41L00001          INCR      FP2
0084 0084 5FL40003          S,W       X,FP2-
0085 0085 46L08BA5          L,W       X,A
0086 0086 1EL10003          S,W         X,B
0087 0087 41C40001          DECR      FP2
0088 0088 41C40001          DECR      FP2
0089 0089 41C30001          DECR        BL
008A 008A 2911FF62          BNZ,BL      UPS
008B 008B 40L00001          NOP
                      *   RT  TO  LEFT  SWEEP
008C 008C 33L00FFA          LI    FP1,250
008D 008D 334FFFF6        LI    FP2,251   N-1
008E 008E 35A00FFA      LI  FP3,250
008F 008F 74AFFFF5          LI      BL,245
0090 0090 40L00001 RTLFTS NOP
0091 0091 43L08BA5        L       X,FP1       N T
0092 0092 40FF86B3          ROT     X,1
0093 0093 435422A5       LAND     X,FP1-     N ANDED N SHIFED
0094 0094 40L08E43       L  Y,X
0095 0095 40FF8E8B          ROT     X,-1      LEFT SHIFT RESULT
0096 0096 40L08B8B
0097 0097 40L0C9A2       LOR    X,Y      OR  NRS&NLS
0098 0098 48L70003     L  M,X  RESULTS TO MASK
0099 0099 434C8845          L       Y,FP2
009A 009A 43LF86A5          L     X,FP1
009B 009B 40L0CCA2          LXOR        X,Y
009C 009C 40L02243          LAND        Y,X
009D 009D 40L08BA2          L     X,Y
009E 009E 40FF8E52          ROT     Y,1
009F 009F 43L02245          LAND        Y,FP1
00A0 00A0 40FF8B8B          ROT     X,-1
00A1 00A1 4CL08B8B
00A2 00A2 43L022A5          LAND        X,FP1
00A3 00A3 4CL2CC43          LXOR      Y,X
00A4 00A4 40L058A1          LANDN       M,Y
00A5 00A5 4CL0CFA2
00A6 00A6 48L00003
00A7 00A7 43L08E45          L         Y,FP1
```

Figure B-12 - Raster Processing Validation Program Printouts

(Sheet 6 of 14)

```
APPLE V01-00 04-DEC-00 15:48:06 PAGE 00004

  00A8 00A8 43D4CC45           LXOR        Y,FP2-
  00A9 00A9 40600A8A1          LAND   M,Y
  00AA 00AA 40000222A2
  00AB 00AB 48600003
  00AC 00AC 5AL00001           S      M,A
  00AD 00AD 40000441           LN     X,M
  00AE 00AE 42D199A5           LOR     X,B
  00AF 00AF 03000225           LAND       X,FP2+
  0080 0080 01000001           INCR    FP2
  0081 0081 41000001           INCR    FP2
  0082 0082 5BD40003           S       X,FP2-
  0083 0083 42000EA5           L       X,A
  0084 0084 1AL10003           S         X,B
  0085 0085 01D40001           DECR    FP2
  0086 0086 01D40001           DECR    FP2
  0087 0087 01030001          DECR     BL
  0088 0088 29110090          BNZ,BL   RTLFTS
  0089 0089 34D19C47          LR       BL,PASSCT
  008A 008A 01030001          DECR     BL
W 0088 0088 30D19C47          SR    BL,PASSCT
  008C 008C 29110001          BNZ,BL   COMMENCE
  008D 008D 34A00002          LI       BL,2
W 008E 008E 32D19C47          SR       BL,PASSCT       RESET PASSCOUNTER FOR THIN
  008F 008F 00007741          CLR      Y
  00C0 00C0 33000080          LI      FP1,128
  00C1 00C1 3E7F00C2          RPT,128
  00C2 00C2 1F300002          S,W      Y,FP1+
  00C3 00C3 28L80000          B     0(R3)
  00C4 00C4 38L02000          WAIT
                       ******* *****   ***** ******   ********X  *****
  00C5 00C5 00000001 THIK9    NOP
  00C6 00C6 34000078          LI       BL,120
  00C7 00C7 33500005          LI       FP1,5
  00C8 00C8 33400082          LI       FP2,130
  00C9 00C9 75A00080          LI       FP3,128  L-2
  00CA 00CA 473086A5 THICK9   L,W      Y,FP1+
  00CB 00CB 40FC65A          ROT      Y,-4
  00CC 00CC 40008E5A
  00CD 00CD 42008FA2          SOR,W    Y,FP2
  00CE 00CE 474099A5
  00CF 00CF 1F400003
  00D0 00D0 41000001          INCR     FP2
  00D1 00D1 42008FA2          SOR,W    Y,FP2
  00D2 00D2 474099A5
  00D3 00D3 1F400003
  00D4 00D4 01D40001          DECR     FP2
  00D5 00D5 41D40001          DECR     FP2
  00D6 00D6 42008FA2          SOR,W    Y,FP2
  00D7 00D7 474099A5
  00D8 00D8 5F400003
  00D9 00D9 42F88E52          ROT    Y,8
  00DA 00DA 42008EA2          SOR,W    Y,FP2
  00DB 00DB 474099A5
  00DC 00DC 1F400003
  00DD 00DD 41000001          INCR     FP2
  00DE 00DE 42100FA2          SOR,W    Y,FP2
  00DF 00DF 474099A5
  02E0 00E0 1F400003
```

Figure B-12 - Raster Processing Validation Program Printouts
(Sheet 7 of 14)

```
00E1 00E1 41000F01          INCR    FP2
00E2 00E2 42000882          SOR,W   Y,FP2
00E3 00E3 474099A5
00E4 00E4 1F4FFF03
00E5 00E5 41000F01          INCR    FP2
00E6 00E6 40F8885A          ROT     Y,-8
00E7 00E7 400F885A
00E8 00E8 42000882          SOR,W   Y,FP2
00E9 00E9 474099A5
00EA 00EA 5F400F03
00EB 00EB 42000882          SOR,W   Y,FP3
00EC 00EC 47099A5
00ED 00ED 5F000003
00EE 00EE 40FF885A          ROT     Y,7
00EF 00EF 40F8885A
00F0 00F0 42000882          SOR,W   Y,FP2
00F1 00F1 474099A5
00F2 00F2 5F400F03
00F3 00F3 42000882          SOR,W   Y,FP3
00F4 00F4 47099A5
00F5 00F5 1F000F03
00F6 00F6 01000F01          INCR    FP2
00F7 00F7 41A40001          DECR    FP3
00F8 00F8 40FFF85A          ROT     Y,-1
00F9 00F9 400F885A
00FA 00FA 42000882          SOR,W   Y,FP2
00FB 00FB 474099A5
00FC 00FC 5F400F03
00FD 00FD 42000882          SOR,W   Y,FP3
00FE 00FE 47099A5
00FF 00FF 5F000F03
0100 0100 40FC885A          ROT     Y,-5
0101 0101 40FF8852
0102 0102 400F885A
0103 0103 42000882          SOR,W   Y,FP2
0104 0104 474099A5
0105 0105 5F400F03
0106 0106 42000882          SOR,W   Y,FP3
0107 0107 47099A5
0108 0108 1F000F03
0109 0109 01000F01          INCR    FP2
010A 010A 41A40001          DECR    FP3
010B 010B 40FF8852          ROT     Y,1
010C 010C 42000882          SOR,W   Y,FP2
010D 010D 474099A5
010E 010E 5F400F03
010F 010F 42000882          SOR,W   Y,FP3
0110 0110 47099A5
0111 0111 5F000F03
0112 0112 40FF8852          ROT     Y,1
0113 0113 42000882          SOR,W   Y,FP2
0114 0114 474099A5
0115 0115 5F400F03
0116 0116 42000882          SOR,W   Y,FP3
0117 0117 47099A5
0118 0118 5F000F03
0119 0119 40FF8852          ROT     Y,1
011A 011A 42000882          SOR,W   Y,FP2
```

Figure B-12 - Raster Processing Validation Program Printouts
(Sheet 8 of 14)

```
0118 0118 474099A5
011C 011C 5F400003
011D 011D 42008BA2          SOR,W    Y,FP3
011E 011E 47009945
011F 011F 5F000003
0120 0120 00000001          NOP
0121 0121 01040001          DECR     FP2
0122 0122 41400001          INCR     FP3
0123 0123 42008BA2          SOR,W    Y,FP2
0124 0124 474099A5
0125 0125 5F400003
0126 0126 42008BA2          SOR,W    Y,FP3
0127 0127 47009945
0128 0128 5F000003
0129 0129 40FE885A          ROT   Y,-2
012A 012A 40000000
012B 012B 42008BA2          SOR,W    Y,FP2
012C 012C 474099A5
012D 012D 5F400003
012E 012E 42008BA2          SOR,W    Y,FP3
012F 012F 47009945
0130 0130 1F000003
0131 0131 01400001          INCR     FP3
0132 0132 41040001          DECR     FP2
0133 0133 40FF885A          ROT   Y,-1
0134 0134 40000000
0135 0135 42008BA2          SOR,W    Y,FP2
0136 0136 474099A5
0137 0137 5F400003
0138 0138 42008BA2          SOR,W    Y,FP3
0139 0139 47009945
013A 013A 5F000003
013B 013B 40FC8852          ROT      Y,4
013C 013C 42008BA2          SOR,W    Y,FP2
013D 013D 474099A5
013E 013E 5F400003
013F 013F 42008BA2          SOR,W    Y,FP3
0140 0140 47009945
0141 0141 1F000003
0142 0142 01400001          INCR     FP3
0143 0143 41040001          DECR     FP2
0144 0144 40FF8852          ROT   Y,1
0145 0145 42008BA2          SOR,W    Y,FP2
0146 0146 474099A5
0147 0147 5F400003
0148 0148 42008BA2          SOR,W    Y,FP3
0149 0149 47009945
014A 014A 5F000003
014B 014B 40FB885A          ROT   Y,-6
014C 014C 40FF885A
014D 014D 42008BA2          SOR,W    Y,FP2
014E 014E 474099A5
014F 014F 5F400003
0150 0150 42008BA2          SOR,W    Y,FP3
0151 0151 47009945
0152 0152 5F000003
0153 0153 00000001          NOP
0154 0154 01000001          DECR     BL
```

Figure B-12 - Raster Processing Validation Program Printouts
(Sheet 9 of 14)

```
0155 0155 29110FCA          BNZ,BL   THICK9
0156 0156 00000001          NOP
0157 0157 28000000          B    0(R3)
0158 0158 38002000          WAIT
                    ******      ********      **********      **********  ***
0159 0159 40000001 T5START  NOP
015A 015A 00000001 THICK5   NOP
015B 015B 34000078          LI      BL,120
015C 015C 33000005          LI.     FP1,5   L
015D 015D 33000081          LI      FP2,129
015E 015E 75000083          LI        FP3,131
015F 015F 07300845 THICK5   L,W  Y,FP1+
0160 0160 41000001          INCR    FP2  L
0161 0161 40FE885A          ROT     Y,-2
0162 0162 40008858
0163 0163 42008882          SOR,W    Y,FP2
0164 0164 47409945
0165 0165 5F400003
0166 0166 401F8852          ROT     Y,1
0167 0167 40FC8852
0168 0168 42008882          SOR,W    Y,FP2
0169 0169 47409945
016A 016A 1F400003
016B 016B 41040001          DECR    FP2  L=1
016C 016C 40FF885A          ROT     Y,-1
016D 016D 40008858
016E 016E 42008882          SOR,W    Y,FP2
016F 016F 47409945
0170 0170 5F400003
0171 0171 42008882          SOR,W    Y,FP3
0172 0172 47409945
0173 0173 5F000003
0174 0174 40FE885A          ROT     Y,-2
0175 0175 40008858
0176 0176 42008882          SOR,W    Y,FP2
0177 0177 47409945
0178 0178 5F400003
0179 0179 42008882          SOR,W    Y,FP3
017A 017A 47009945
017B 017B 5F000003
017C 017C 401F8852          ROT     Y,1
017D 017D 42008882          SOR,W    Y,FP2
017E 017E 47409945
017F 017F 5F400003
0180 0180 42008882          SOR,W    Y,FP3
0181 0181 47009945
0182 0182 1F000003
0183 0183 01040001          DECR     FP2
0184 0184 41000001          INCR    FP3
0185 0185 42008882          SOR,W    Y,FP2
0186 0186 47409945
0187 0187 5F400003
0188 0188 42008882          SOR,W    Y,FP3
0189 0189 47009945
018A 018A 1F000003
018B 018B 01000001          INCR    FP2
018C 018C 01000001          INCR    FP2
018D 018D 01030001          DECR    BL
```

Figure B-12 - Raster Processing Validation Program Printouts
(Sheet 10 of 14)

```
018E 018E 2911015F          BNZ,BL    THICK5
018F 018F 03000001          NOP
0190 0190 28080000          B     0(R3)
0191 0191 38002000          WAIT
                     ******     ********      **********      **********  ***
                     ******     ********      **********      **********  ***
0192 0192 00000001 THIK3     NOP
0193 0193 33000005          LI      FP1,5
0194 0194 33400081          LI      FP2,129
0195 0195 35A00083          LI      FP3,131
0196 0196 74A00078          LI      BL,120
0197 0197 47308845 THICK3   L,W     Y,FP1+
0198 0198 401F885A          ROT     Y,-1
0199 0199 40008858
019A 019A 42308852          SOR,W   Y,FP2
019B 019B 47409945
019C 019C 5F400003
019D 019D 401F8852          ROT     Y,1
019E 019E 42308852          SOR,W   Y,FP2
019F 019F 47409945
01A0 01A0 5F400003
01A1 01A1 401F8852          ROT     Y,1
01A2 01A2 42308852          SOR,W   Y,FP2
01A3 01A3 47409945
01A4 01A4 1F400003
01A5 01A5 41000001          INCR    FP2
01A6 01A6 42308852          SOR,W   Y,FP2
01A7 01A7 47409945
01A8 01A8 5F400003
01A9 01A9 401E885A          ROT     Y,-2
01AA 01AA 40008858
01AB 01AB 42308852          SOR,W   Y,FP2
01AC 01AC 47409945
01AD 01AD 5F400003
01AE 01AE 42000001          NOP
01AF 01AF 42308852          SOR,W   Y,FP3
01B0 01B0 47309945
01B1 01B1 5F300003
01B2 01B2 401F8852          ROT     Y,1
01B3 01B3 42308852          SOR,W   Y,FP3
01B4 01B4 47309945
01B5 01B5 5F300003
01B6 01B6 401F8852          ROT     Y,1
01B7 01B7 42308852          SOR,W   Y,FP3
01B8 01B8 47309945
01B9 01B9 1F300003
01BA 01BA 01A00001          INCR    FP3
01BB 01BB 01130001          DECR    BL
01BC 01BC 2911FF97          BNZ,BL    THICK3
01BD 01BD 00000001          NOP
01BE 01BE 28080000          B     0(R3)
01BF 01BF 38002000          WAIT
                     *******  *****   ***** ******   ****      *****
01C0 01C0 00000001 BTOT     NOP
                     *  BOTTEM HALF TO TOP HALF COPY
01C1 01C1 33000080          LI      FP1,128
01C2 01C2 33400000          LI      FP2,0
01C3 01C3 3F7F01C6          LOOP,128  COPY2
```

Figure B-12 - Raster Processing Validation Program Printouts
(Sheet 11 of 14)

```
01C4 01C4 47J08845          L,W     Y,FP1+
01C5 01C5 5F000002          S,W     Y,FP2+
01C6 01C6 00000001 COPY2    NOP
01C7 01C7 00007741          CLR  Y
01C8 01C8 33000080          LI      FP1,128
01C9 01C9 3E7F01CA          RPT,128
01CA 01CA 1F00FF02          S,W     Y,FP1+
01CB 01CB 28680000          B    0(R3)
01CC 01CC 38002F00          WAIT
01CD 01CD 00FF0001          NOP
01CE 01CE 2C019F4E    BAL,R3        TESTPAT
01CF 01CF 2C01FF00    BAL,R3        THIN
01D0 01D0 38002000          WAIT
01D1 01D1 2C0100C5    BAL,R3        THIK9
01D2 01D2 2C01A1C0    BAL,R3        BTOT
01D3 01D3 2C019A1F    BAL,R3        TT3
01D4 01D4 38002F00          WAIT
01D5 01D5 2C0100C5    BAL,R3        THIK9
01D6 01D6 2C01A1C0    BAL,R3        BTOT
01D7 01D7 38002F00          WAIT
01D8 01D8 28010E4     B     DLT
01D9 01D9 2C01015A    BAL,R3      THIK5
01DA 01DA 2C01A1C0    BAL,R3        BTOT
01DB 01DB 2C01015A    BAL,R3      THIK5
01DC 01DC 2C01A1C0          BAL,R3  BTOT
01DD 01DD 38002F00          WAIT
01DE 01DE 2C01A192          BAL,R3   THIK3
01DF 01DF 2C01A1C0          BAL,R3  BTOT
01E0 01E0 2C010192          BAL,R3   THIK3
01E1 01E1 2C01A1C0          BAL,R3  BTOT
01E2 01E2 2C010192          BAL,R3   THIK3
01E3 01E3 2C01A1C0          BAL,R3  BTOT
01E4 01E4 00000F01 DLT    NOP
01E5 01E5 2C019192          BAL,R3   THIK3
01E6 01E6 2C01A1C0          BAL,R3  BTOT
01E7 01E7 2C010192          BAL,R3   THIK3
01E8 01E8 00000001          NOP
01E9 01E9 2C019F28    BAL,R3        TT2
01EA 01EA 2C01D1E0    BAL,R3        DRILL
01EB 01EB 38002F00          WAIT
01EC 01EC 38002000          WAIT
                      ******      ********      **********      **********  ***
                      *   D80L  MOD2
01ED 01ED 00000001 DRILL    NOP
01EE 01EE 33000008          LI      FP2,8
01EF 01EF 35A00082        LI    FP3,130
01F0 01F0 3FE001F4        LOOP,110   D80LSTOP
01F1 01F1 47000E45        L,W   Y,FP2+
01F2 01F2 47000C45        LXOR,W   Y,FP3
01F3 01F3 5FA00002          S,W   Y,FP3+
01F4 01F4 02000001 D80LSTOP    NOP
01F5 01F5 28680000          B    0(R3)
01F6 01F6 38002000        WAIT
                      ******      ********      **********      **********  ***
01F7 01F7 33000000          LI      FP2,0
01F8 01F8 3FFF01FB          LOOP,256    HERER
01F9 01F9 47408BAD          GEN,32  X'474086AD'    X=AM(FP2) MIRROR)
01FA 01FA 5F000003          S,W     X,FP2+
01FB 01FB 00000001 HERER    NOP
                      ******      ********      **********      **********  ***
01FC 01FC 00000001          NOP
01FD 01FD 33000000          LI      FP2,0
01FE 01FE 3FFF0201          LOOP,256     HEREC
01FF 01FF 43408BAD          GEN,32  X'434086AD'    X=AM(FP2) MIRROR)
0200 0200 5B000003          S       X,FP2+
0201 0201 00000001 HEREC    NOP
0202 0202 38002000          WAIT
```

Figure B-12 - Raster Processing Validation Program Printouts
(Sheet 12 of 14)

```
                 9000           ORG   X'9000'
                        ******      ********      **********      **********  ***
  9000 9000 357000F1          LI     FL1,1
  9001 9001 35A400C0          LI     FP3,0
  9002 9002 33400000 ST1       LI     FP2,0
  9003 9003 33600003 ST2       LI     FP1,3
  9004 9004 74A00040           LI      BL,64
  9005 9005 00000001 MOV     NOP
W 9006 9006 31619C1C           SR     FP1,DUMPBUFF
W 9007 9007 31419C1C           SR     FP2,DUMPBUFF
  9008 9008 3B616C2C             ASSGN,2     12
  9009 9009 3B616C3C             ASSGN,3     12
  900A 900A 34209C1C           MAM     DUMPBUFF
  900B 900B 3BC10000
  900C 900C 3BC18C0C
  900D 900D 3BC18006           TPIO   6
  900E 900E 28C1900D           B   $-1
  900F 900F 3BC16C20             ASSGN,2    0
  9010 9010 3BC16C30             ASSGN,3    0
  9011 9011 01300001           INCR    FP1
  9012 9012 01400001         INCR   FP1
  9013 9013 01500001         INCR   FP1
  9014 9014 01600001         INCR   FP1
  9015 9015 01600001           INCR  FP2
  9016 9016 41C30001           DECR  BL
  9017 9017 00000001           NOP
  9018 9018 2911900B           BNZ,BL      MOV
  9019 9019 00C00001           NOP
  901A 901A 3BC02000           WAIT
  901B 901B 28C19023           B   ST2
  901C 901C 01C00300 DUMPBUFF  DPIO  MAM,0,2,0,3,3,1
  901D 901D 20C03001
  901E 901E 10C00000
                        ******      ********      **********      **********  ***
  901F 901F 00000001 TT3       NOP
  9020 9020 3BC16C2C           ASSGN,2      12
  9021 9021 3BC16C3C           ASSGN,3     12
  9022 9022 34209C3D           MAM     TOPTO3
  9023 9023 3BC10000
  9024 9024 3BC18C0C
  9025 9025 3BC18006           TPIO   6
  9026 9026 28C19025         B      $-1
  9027 9027 3BC16C20           ASSGN,2   0
  9028 9028 3BC16C30             ASSGN,3  0
  9029 9029 28C80000           B     0(R3)
  902A 902A 3BC02000           WAIT
  9028 9028 00C00001 TT2       NOP
  902C 902C 3BC16C2C           ASSGN,2      12
  902D 902D 3BC16C3C           ASSGN,3     12
  902E 902E 34209C40           MAM     TOPTO2
  902F 902F 3BC10000
  9030 9030 3BC18C0C
  9031 9031 3BC18006           TPIO   6
  9032 9032 28C19031         B      $-1
  9033 9033 3BC16C20           ASSGN,2   0
  9034 9034 3BC16C30             ASSGN,3  0
  9035 9035 34001000           LI      ASH,X'1000'
  9036 9036 73A00000           LI      FP1,0
```

Figure B-12 - Raster Processing Validation Program Printouts

(Sheet 13 of 14)

```
APPLE V01-00 04-DEC-00 15:48:06 PAGE 00012

9037 9037 00007741          CLR    Y
9038 9038 3EFF9039          RPT,256
9039 9039 03308845          L      Y,FP1+
903A 903A 34002000          LI     ASH,X'2000'
903B 903B 28680000          S      0(R3)
903C 903C 38102200          WAIT
903D 903D 8CL00700 TOPTO3   DPIO   MAM,1,2,0,3,0,128
903E 903E 20003701
903F 903F 10L00000
9040 9040 8CUP0000 TOPTO2   DPIO   MAM,1,3,0,2,0,128
9041 9041 10LM3701
9042 9042 20L00F00
               0000 A       EQU      0
               0001 B       EQU      1
               0004 NM1     EQU      4                    N=1
               0003 NM2     EQU      3                    N=2
               0002 NM3     EQU      2                    N=3
9043 9043 00L00000 SFP1     DC   0
9044 9044 00L00000 SFP2     DC   0
9045 9045 00L00000 SAVFP    DC   0
9046 9046 00UF0000 SAVFPE   DC   0
9047 9047 00U20000 PASSCT   DC    X'20000'


APPLE V01-00 04-DEC-00 15:48:06 PAGE 00017

ERRORS DETECTED: 0000M
WARNINGS DETECTED: 00004


                        AP SYMBOL TABLE

        SYMBOL    ADDRESS                  SYMBOL    ADDRESS

        A           0000                   B           0001
        BTOT        01C0                   COMMENCE    0001
        COPY2       01C6             EXT   DATA1$      0000
        DBOLSTUP    01F4                   DLT         01E4
        DOND        90B0                   DONE        90AA
        DONSL       90C3                   DOWNS       0007
        DRILL       01ED                   DUMPBUFF    901C
        HEREC       0201                   HERER       01FB
        LFRTS       0034             EXT   MAMB$       0000
  EXT   MAMW$       0000                   MOV         9005
        NM1         0004                   NM2         0003
        NM3         0002                   PASSCT      9047
        RTLFTS      0090                   SAVFP       9045
        SAVFPE      9046                   SFP1        9043
        SFP2        9044                   SHIFTIT     90CA
        ST1         9002                   ST2         9003
        T5SSTART    0159                   TESTPAT     904E
        THICK3      0197                   THICK5      015F
        THICK9      00CA                   THIK3       0192
        THIK5       015A                   THIK9       00C5
        THIP        0000                   TOPTO2      9040
        TOPTO3      903D                   TT2         902B
        TT3         901F                   UPS         0062
```

Figure B-12 - Raster Processing Validation Program Printouts
(Sheet 14 of 14)

APPENDIX C - SIMULTANEOUS MULTI-EXPOSURE ANALYTICAL CALIBRATION PROGRAM ALGORITHMS (PARTIAL)

1. INTRODUCTION

This appendix contains the detailed algorithms used in three of the five preliminary programs that make up part of the Simultaneous Multi-exposure Analytical Calibration [SMAC] Program. The five preliminary programs are listed below:

1. Program I - Data Preparation Program
2. Program II - General Catalog Search Program
3. Program III - Preliminary Orientation Computation Program
4. Program IV - Stellar Search Program
5. Program V - Single Camera Adjustment Program

Algorithms for Programs II, III, and IV are provided in this appendix for reference since these are the programs that were evaluated in Section 4 of this report. Each of the three programs was subdivided functionally in Section 4.2. The algorithms here are labeled and ordered in accordance with their function numbers [II-1, II-2, etc.]. This appendix accounts for each function. If there is no algorithm for a particular function, then this is so stated.

All algorithms were obtained or derived from a final report entitled Documentation for the Multi-Camera Stellar SMAC Computation Program, which was submitted by DBA Systems, Inc. of Melborne, Florida, under Contract DAAK02-69-C-0099. The contract was performed for the Department of the Army, U.S. Army Engineer Topographic Laboratories, Fort Belvoir, Virginia.

2. GENERAL CATALOG SEARCH PROGRAM ALGORITHMS

2.1 Algorithm II-1

There is no applicable algorithm.

2.2 Algorithm II-2

There is no applicable algorithm.

2.3 Algorithm II-3

This subprogram updates the cataloged star positions of 1950 to the time of observation and applies corrections for diurnal aberration [Newcomb, 1906].

The right ascension and declination are updated from 1950 to the beginning of the current year to yield the mean places of the stars, which are expressed by Taylor's theorem:

$$\alpha = \alpha_{1950} + t \frac{d\alpha}{dt} + \frac{1}{2} t^2 \frac{d^2\alpha}{dt^2} , \qquad (1)$$

and

$$\delta = \delta_{1950} + t \frac{d\delta}{dt} + \frac{1}{2} t^2 \frac{d^2\delta}{dt^2} ,$$

where $\frac{d\alpha}{dt}$, $\frac{d\delta}{dt}$ are annual variations in α and δ; $\frac{d^2\alpha}{dt^2}$, $\frac{d^2\delta}{dt^2}$ are secular variations in α and δ; and t = current year − 1950 . The annual variation due to precession alone is:

$$\frac{d\alpha}{dt_1} = m + n \sin \alpha \tan \delta = p_\alpha ,$$

and

$$\frac{d\delta}{dt_1} = n \cos \alpha = p_\delta ,$$

where m is the annual precession in α, and n is the annual precession in δ.

The annual variation due to proper motion alone is:

$$\frac{d\alpha}{dt_2} = \mu_\alpha , \quad \frac{d\delta}{dt_2} = \mu_\delta ,$$

where μ_α, μ_δ are the proper motions in α and δ.

With the last two equations, the total annual variation is expressed as:

$$\frac{d\alpha}{dt} = m + n \sin \alpha \tan \delta + \mu_\alpha = p_\alpha + \mu_\alpha,$$

and

$$\frac{d\delta}{dt} = n \cos \alpha + \mu_\alpha + p_\delta + \mu_\delta. \qquad (2)$$

Taking the derivatives of the above two equations yields the secular variations:

$$\frac{d^2\alpha}{dt^2} = \frac{dm}{dt} + \sin \alpha \tan \delta \frac{dn}{dt} + n (p_\alpha + \mu_\alpha) \cos \alpha \tan \delta + n (p_\delta + \mu_\delta) \sin \alpha \sec^2 \delta + \frac{d\mu_\delta}{dt},$$

and

$$\frac{d^2\delta}{dt^2} = \cos \alpha \frac{dn}{dt} - (p_\alpha + \mu_\alpha) n \sin \alpha + \frac{d\mu_\delta}{dt}.$$

The variation in the proper motion due to proper motion alone is:

$$\frac{d\mu_\alpha}{dt_1} = 2\mu_\alpha \mu_\delta \tan \delta,$$

and

$$\frac{d\mu_\delta}{dt_1} = - \mu_\alpha^2 \sin \delta \cos \delta.$$

The variation in the proper motion due to precession alone is:

$$\frac{d\mu_\alpha}{dt_2} = \mu_\alpha n \cos \alpha \tan \delta + \mu_\delta n \sin \alpha \sec^2 \delta$$

and

$$\frac{d\mu_\delta}{dt_2} = - \mu_\alpha n \sin \alpha.$$

With the last three equations, the total secular variation is of the form:

$$\frac{d^2\alpha}{dt^2} = \frac{dm}{dt} + \sin\alpha \tan\delta \frac{dn}{dt} + n(p_\alpha + 2\mu_\alpha)\cos\alpha \tan\delta +$$

$$n(p_\delta + 2\mu_\delta)\sin\alpha \sec^2\delta + 2\xi_\alpha \mu_\delta \tan\delta, \quad (3)$$

and

$$\frac{d^2\delta}{dt^2} = \cos\alpha \frac{dn}{dt} - n(p_\alpha + 2\mu_\alpha)\sin\alpha - \mu_\alpha{}^2 \sin\delta\cos\delta.$$

When Equations 2 and 3 are inserted in Equation 1, the result gives full expression for the mean places of the stars.

Corrections are applied to the mean places of the stars of update them to the time of exposure. The Besselian day numbers are looked up in tables for the day of the exposure and the day after the exposure, and the numbers are interpolated for the actual time of exposure. The corrections to the star portions are:

$$\Delta\alpha = A(m/n + \sin\alpha \tan\delta) + B(\cos\alpha \tan\delta) +$$
$$C(\cos\alpha/\cos\delta) + D(\sin\alpha/\cos\delta) + E + \tau(\mu_\alpha) +$$
$$J(\tan^2\delta),$$

and

$$\Delta\delta = A(\cos\alpha) - B(\sin\alpha) + C(\tan\varepsilon\cos\delta - \sin\alpha\sin\delta) +$$
$$D(\cos\alpha \sin\delta) + \tau(\mu_\delta) + J'(\tan\delta),$$

where A,B,C,D,E, and τ are the interpolated Besselian day numbers, and ε is the mean obliquity.

The corrections for diurnal aberration are expressed as:

$$\Delta\alpha = 0.319'' \cos\phi \cos h \sec\delta,$$

and

$$\Delta\delta = 0.319'' \cos\phi \sin h \sin\delta,$$

where ϕ is the latitude of the point of observation, and h is the hour angle.

The second order day numbers J, J' are computed for northern declinations as follows:

$$J = P_1 P_2,$$

and

$$J' = -.5 P_1^2;$$

for southern declinations as follows:

$$J = Q_1 Q_2,$$

and

$$J' = -.5 Q_1^2;$$

where

$$P_1 = (A + D) \sin \alpha + (B + C) \cos \alpha,$$

$$P_2 = (A + D) \cos \alpha - (B + C) \sin \alpha,$$

$$Q_1 = (A - D) \sin \alpha + (B - C) \cos \alpha, \text{ and}$$

$$Q_2 = (A - D) \cos \alpha - (B - C) \sin \alpha.$$

2.4 Algorithm II-4

The relationship of the right ascension α and declination δ [updated versions] to true zenith distance ζ_t and azimuth Z is given by:

$$t = GST + 1.00273791(UT) - \lambda - \alpha$$

$$\cos \zeta_t = \sin \delta \sin \phi + \cos \delta \cos \phi \cos t$$

$$\tan Z = -\sin t/(\tan \delta \cos \phi - \cos t \sin \phi)$$

where

t = local hour angle,

GST = Greenwich sidereal time,

UT = universal time of observation,

λ = west longitude of camera location, and

ϕ = latitude of camera location.

3. PRELIMINARY ORIENTATION COMPUTATION PROGRAM ALGORITHMS

3.1 Algorithm III-1

The apparent zenith distance of a star is given by:

$$\zeta_a = \zeta_t - \Delta\zeta \qquad (4)$$

where ζ_t is the true zenith distance, and the error, $\Delta\zeta$, caused by refraction is given by Garfinkel's expansion:

$$\Delta\zeta = \eta_1 \tan\theta + \eta_2 \tan^3\theta + \eta_3 \tan^5\theta + \eta_4 \tan^7\theta + \eta_5 \tan^9\theta.$$

In this expansion, η_i are constant coefficients depending upon atmospheric pressure, temperature, and wavelength of the observed light, and

$$\tan\theta = \frac{\frac{1}{\gamma_0}\tan\zeta_a}{1 + \sqrt{1 + (\frac{1}{\gamma_0} + \tan\zeta_a)^2}},$$

where

γ_0 is a parameter dependent upon temperature.

Equation 4 is solved iteratively for ζ_a by:

$$\zeta_a^{(p+1)} = \zeta_t - (\Delta\zeta)^{(p)}.$$

For the first approximation, $\zeta_a = \zeta_t$. The equation then is iterated until

$$|\zeta_a^{(p+1)} - \zeta_a(p)| \leq 10^{-8} \text{ or } P = 10$$

3.2 Algorithm III-2

The direction cosine of a reference star in object space is derived from the apparent zenith distance, ζ_a and azimuth, Z, of the star:

$$\lambda = \sin \zeta_a \sin Z,$$

$$\mu = \sin \zeta_a \cos Z, \text{ and}$$

$$\nu = \cos \zeta_a.$$

3.3 Algorithm III-3

The direction cosine of a reference star in image space within the camera is given by:

$$\begin{pmatrix} K \\ L \\ M \end{pmatrix} = \frac{1}{\sqrt{(x - x_p)^2 + (y - y)^2 + c^2}} \begin{pmatrix} x - x_p \\ y - y_p \\ c \end{pmatrix},$$

where

c = principal distance of camera,

(x_p, y_p) = principal point in image plane, and

(x, y) = the reference star's coordinates.

3.4 Algorithm III-4

The relation between the image space, object space, and camera's exterior orientation may be written as:

$$\begin{pmatrix} K \\ L \\ M \end{pmatrix} = T(\alpha, \omega, \eta) \begin{pmatrix} \lambda \\ \mu \\ \nu \end{pmatrix}, \qquad (5)$$

where $T(\alpha, \omega, \eta)$ is the exterior orientation matrix of the camera and is defined in terms of the three successive orientation angles α, ω, η as follows:

$$T(\alpha, \omega, \eta) = \begin{pmatrix} A & B & C \\ A' & B' & C' \\ D & E & F \end{pmatrix} = \begin{pmatrix} -\cos \eta & \sin \eta & 0 \\ \sin \eta & \cos \eta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & -\sin \omega & \cos \omega \\ 0 & \cos \omega & \sin \omega \end{pmatrix} \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Then Equation 5 may be written as:

$$\begin{pmatrix} K \\ L \\ M \end{pmatrix} = \begin{pmatrix} A & B & C \\ A' & B' & C' \\ D & E & F \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \\ \nu \end{pmatrix},$$

and

$$\begin{pmatrix} \lambda \\ \mu \\ \nu \end{pmatrix} = \begin{pmatrix} A & A' & D \\ B & B' & E \\ C & C' & F \end{pmatrix} \begin{pmatrix} K \\ L \\ M \end{pmatrix}.$$

There are three independent equations in the above two relations. For the purpose of the following discussion, two equations are chosen:

$$\lambda D + \mu E + \nu F = M,$$

and

$$KC + LC' + MF = \nu.$$

Using the above two equations, the following equation for the ith camera, jth exposure, and ℓth reference star can be written:

$$\begin{pmatrix} K & L & M & 0 & 0 \\ 0 & 0 & \nu & \lambda & \mu \end{pmatrix}_{ij\ell} \begin{pmatrix} C \\ C' \\ F \\ D \\ E \end{pmatrix}_{ij} = \begin{pmatrix} \nu \\ M \end{pmatrix}_{ij\ell}$$

A formal least squares solution for [C, C', F, D, E] is given by:

$$\begin{pmatrix} C \\ C' \\ F \\ D \\ E \end{pmatrix}_{ij} = \left\{ \sum_{\ell} \begin{pmatrix} K^2 & KL & KM & 0 & 0 \\ KL & L^2 & ML & 0 & 0 \\ MK & ML & M^2 + \nu^2 & \nu\lambda & \nu\mu \\ 0 & 0 & \lambda\nu & \lambda^2 & \lambda\mu \\ 0 & 0 & \mu\nu & \lambda\mu & \mu^2 \end{pmatrix}_{ij\ell} \right\}^{-1} \times$$

$$\left\{ \sum_{\ell} \begin{pmatrix} K\nu \\ L\nu \\ 2M\nu \\ M\lambda \\ M\mu \end{pmatrix}_{ij\ell} \right\} .$$

The preliminary approximation of the orientation angles become:

$$\begin{pmatrix} \alpha \\ \omega \\ \eta \end{pmatrix}^{(0)}_{ij} = \begin{pmatrix} \tan^{-1}\left(\frac{D}{E}\right)_{ij} \\ \sin^{-1}(F)_{ij} \\ \tan^{-1}\left(\frac{C}{C'}\right)_{ij} \end{pmatrix}$$

3.5 Algorithm III-5

The preliminary approximations $(\alpha, \omega, \eta)^{(0)}_{ij}$ are further refined by a more rigorous least squares adjustment of the photo coordinates of the reference stars. This adjustment may be expressed mathematically as:

$$\begin{pmatrix} \alpha \\ \omega \\ \eta \end{pmatrix}_{ij}^{(p+1)} = \begin{pmatrix} \alpha \\ \omega \\ \eta \end{pmatrix}_{ij}^{(p)} +$$

$$\left[\sum_{\ell} (Q^T \times Q)_{ij\ell}^{(p)} \right]^{-1} \sum_{\ell} (Q^T \varepsilon)_{ij\ell}^{(p)}$$

where

p = the iteration index,

$$Q_{ij\ell} = \frac{\partial (x, y)_{ij\ell}}{\partial (\alpha, \omega, \eta)_{ij}} ,$$

$$\varepsilon_{ij\ell} = \begin{pmatrix} x - x' \\ y - y' \end{pmatrix}_{ij\ell} ,$$

x, y = measured photo coordinates, and

x', y' = calculated star position by using $(\alpha, \omega, \eta)^{(p)}$ and (λ, μ, ν).

The above equation is iterated until the changes in the solution are negligible or the number of iterations exceeds the maximum allowable; $p = 10$ is assumed in this study for estimating the execution time.

4. STELLAR SEARCH PROGRAM

4.1 Algorithm IV-1

To find the direction cosine (λ, μ, ν) for stars, the following relation is used:

$$\begin{pmatrix} \lambda \\ \mu \\ \nu \end{pmatrix} = T(\alpha, \omega, \eta) \begin{pmatrix} x \\ y \\ c \end{pmatrix} \frac{1}{\sqrt{x^2 + y^2 + c^2}} ,$$

where

x, y, c = star's interior coordinate,

α, ω, η = preliminary camera orientation, and

$$T(\alpha, \omega, \eta) = \begin{pmatrix} -\cos\eta & \sin\eta & 0 \\ \sin\eta & \cos\eta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -\sin\omega & \cos\omega \\ 0 & \cos\omega & \sin\omega \end{pmatrix}$$

$$\begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

4.2 Algorithm IV-2

The relationship between the true zenith distance and the apparent zenith distance is given by:

$$\zeta_t = \zeta_a + \Delta\zeta$$

where

$$\Delta\zeta = \sum_{i=1}^{5} \eta_i \tan^{2i-1}\theta ,$$

and

$$\tan 2\theta = \frac{1}{\gamma_0} \tan \zeta_a .$$

Combining these equations with

$$\zeta_a = \cos^{-1} \nu$$

gives

$$\zeta_t = \zeta_a + \sum_{i=1}^{5} \eta_i \left(\frac{-\gamma_0 \nu + \sqrt{\gamma_0^2 \nu^2 + 1 - \nu^2}}{\sqrt{1 - \nu^2}} \right)^{2i-1} .$$

4.3 Algorithm IV-3

Algorithm IV-3 is similar to Algorithm II-4.

4.4 Algorithm IV-4

There is no applicable algorithm.

4.5 Algorithm IV-5

Algorithm IV-5 is similar to Algorithm II-3.

4.6 Algorithm IV-6

There is no applicable algorithm.

4.7 Algorithm IV-7

Algorithm IV-7 is similar to Algorithm II-4.

4.8 Algorithm IV-8

Algorithm IV-8 is similar to Algorithm III-1.

4.9 Algorithm IV-9

There is no applicable algorithm.

APPENDIX D - MISCELLANEOUS PHASE I INVESTIGATIONS

1. INTRODUCTION

Phase I of this study validation effort was addressed to the six tasks listed below.

1. AS-11BX postprocessing
2. Raster processing - automated cartography
3. Simultaneous multi-exposure analytical calibration (SMAC) program
4. SLAR image processing
5. Scene generation
6. Instantaneous contours

The first three tasks were selected by ETL for detailed study and are described in the main body of this report. The last three tasks are briefly discussed in this appendix.

2. SLAR IMAGE PROCESSING

The azimuth resolution of a conventional side-looking airborne radar (SLAR) system is limited by the "real aperture" resulting from the physical length of the SLAR antenna. A longer antenna can be simulated by storing the pulse-to-pulse radar returns over a period of time and processing ("compressing") these returns as though they had originated simultaneously. This process produces a "synthetic aperture" capable of an azimuth resolution much greater than that derived with the real aperture. The range data can also be "compressed" to significantly improve the range resolution.

There are several approaches that can be employed to perform the image processing. One technique is to convolve (in both the range and azimuth directions) a point target return waveform with the stored radar data. The convolution can be efficiently implemented using Fast Fourier Transforms (FFT's). Normally each convolution requires two FFT's, a product of the FFT's, and an inverse FFT of product. It is possible for this application to reduce the computations to just two FFT's, however, even with this advantage, the FFT's will account for a significant part of the total processing time.

Compared to sequential computers, STARAN has a significant advantage when processing FFT's. For example, a STARAN S-1000 can perform a 1024-point, complex input FFT approximately 50 times faster than an IBM 370/145. The 370/145 uses 32-bit fields and the STARAN times were developed for 16 bit fields. Even though the STARAN field is variable, 16 bits are assumed adequate for this application.

In general, the STARAN AAP capabilities appear to be well suited to the SLAR synthetic aperture problem. However, additional time would be required to effectively analyze the various facets of the problem such as motion compensation and range curvature, before drawing an accurate quantitative conclusion.

3. SCENE GENERATION

3.1 Description

Another potential STARAN application is in the field of the computer generation of a perspective view of a topographic data base. The input data consists of a gray level, g_i, and a relative height, Z_i for each point (X_i,Y_i) in the Cartesian data base as shown in Figure D-1. The output is a matrix of gray levels that may be used to generate a photograph of the perspective view that corresponds to an observer looking at the scene from a point (X_o,Y_o) at height Z_o.

Since each object in the scene occupies a solid angle in the perspective view, the relative locations and sizes of items in view are expressed in angular relations. Each point (X_i,Y_i) becomes $(\theta_i,\emptyset_i)$, and the height Z_i becomes angle $\Delta\theta_i$. The gray level of each point, g_i, normally depends on the viewing angle, but will be treated as a constant here. Figure D-2 shows the relationship of a point (or a cell); $P(X_i,Y_i,Z_i,g_i)$ in the perspective view. The topographic data base is in a three-dimensional space, while the perspective view is in a two-dimensional space. The same gray level, g_i, covers several points (cells) according to the 'height', $\Delta\theta_i$, in the perspective view.

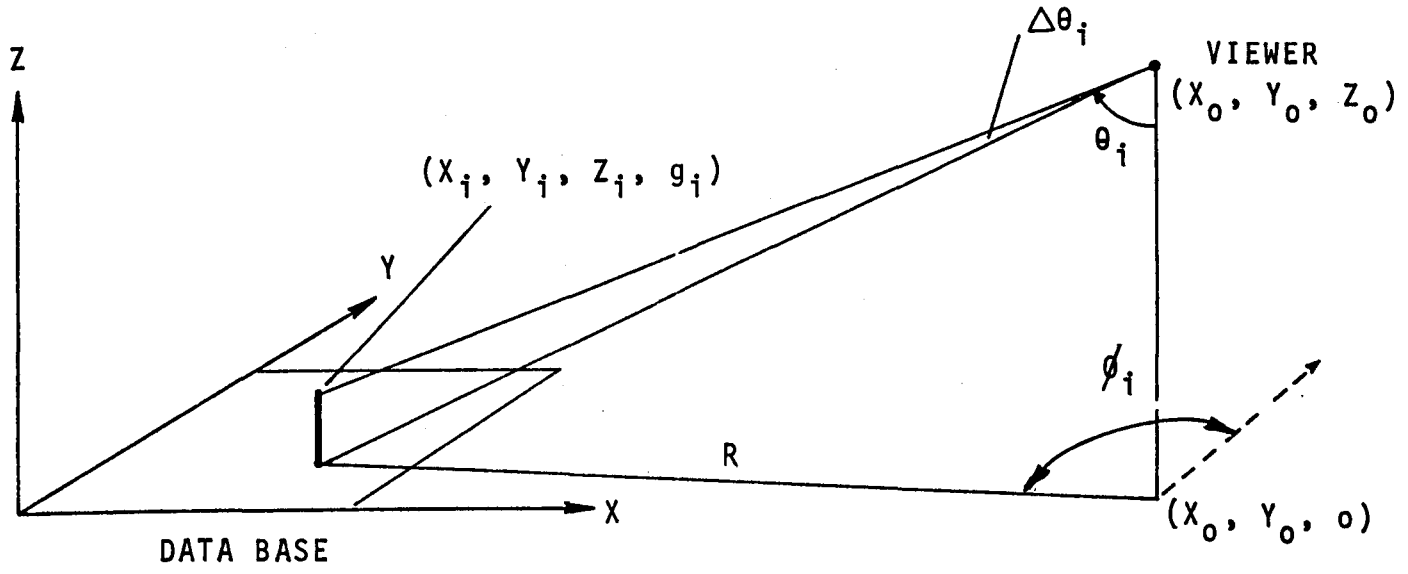


Figure D-1 - Viewer - Topographic Data Base Relationship

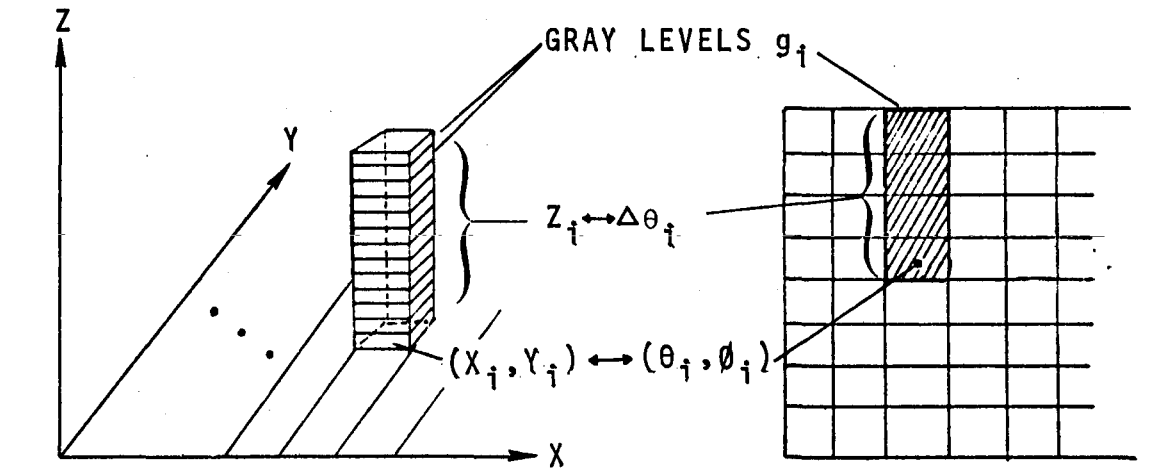


Figure D-2 - Relationship between 3-D Coordinate and 2-D Perspective Views

3.2 Algorithm

From this observation, the scene generation is seen to be just a form of coordinate transformation. Mathematically it can be expressed as:

$$\phi_i = \sin^{-1}\left(\frac{X_i - X_o}{R}\right)$$

$$\theta_i = \mathrm{ctn}^{-1}\frac{Z_o}{R} \qquad (1)$$

$$\Delta\theta_i = \sin^{-1}\left(\frac{Z_i R}{\sqrt{R^2 + Z_o^2}\cdot\sqrt{R^2 + (Z_o - Z_i)^2}}\right)$$

where

$$R = \sqrt{(X_i - X_o)^2 + (Y_i - Y_o)^2}$$

The relations may be simplified if $R >> Z_i$.

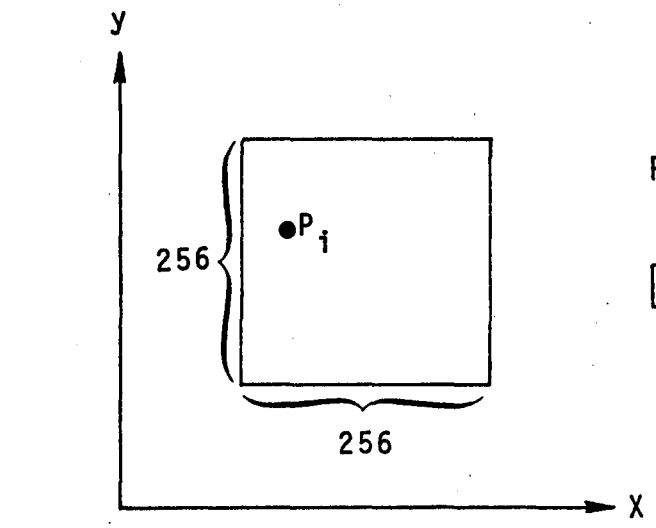
This point-to-point transformation results in an 'X-ray' type image of the perspective view; points that should be 'hidden' from this perspective view are also included. A scheme for removing these hidden objects can be developed by examining Figures D-1 and D-2. if the distant objects (higher θ) are plotted first, then the hidden objects will be 'over-plotted' by the near objects (lower θ). An algorithm based on this concept has been developed and applied in sequential computers[1]. Because the input data are in a predetermined order and the observer can be at any location, some scheme to rearrange the data to remove the hidden object is required. This rearranging can be very time consuming. A CDC 6600 computer takes 5 min to create a perspective view from a 256 x 256 data base with 64 gray levels.

3.3 STARAN approach to scene generation

Because of its associative array processing capabilities, STARAN appears well suited to the scene generation task. For comparison purposes STARAN S-1000 timing estimates will be derived for the same problem implemented on the CDC 6600.

The input data is a matrix of 256 x 256 elements with each element represented by 54-bits as shown in Figure D-3. The output buffer is a 256 x 256 matrix of 6-bit gray levels.

[1] Dr. M. Faintich, Topographic Perspective Views via Digital Image Processing, July 1973. Work performed at USAETL-CSL.



P_i IS PREPRESENTED BY 54 BITS

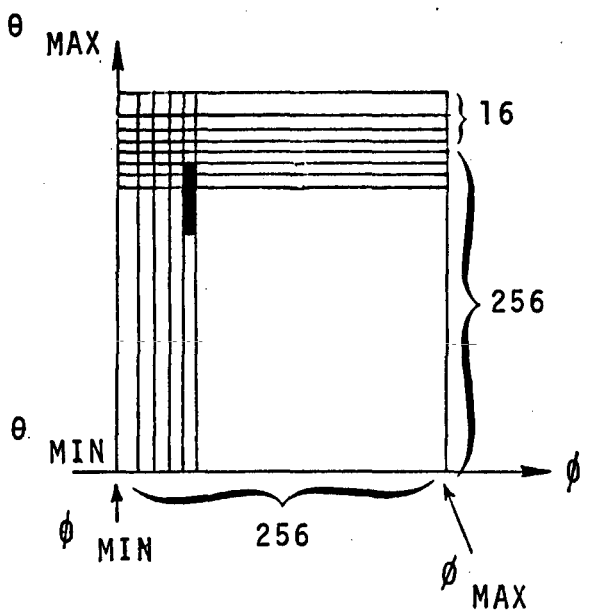


Figure D-3 - Data Definitions for Scene Generation

The perspective views are created in two passes. The first pass uses the STARAN's parallel processing capability, performs the coordinate transformation and scaling of θ and ∅ to a 256 scale for the entire input data base, and outputs ∅ (the intermediate results) to a temporary storage in the control memory. The second pass takes the intermediate file as the input and, using both the associative properties and array processing capability, performs sorts, reorders, and generates (paints) the output buffer. The total time is estimated to be less than three seconds. This amounts to less than 1 percent of the CDC 6600 time.

The functional flow chart and the timing estimates for each subtask is shown in Figure D-4. The following are brief descriptions for each block shown in the flow chart.

1. The topographic data base is brought into one 54-bit array memory (AM) field. The 1024 data points are stored in AM in one load. The total time is:

   $4 \times 1024 = 4.1$ ms

   where

   4 = μsec to load one data point (54 bits) to the AM via the common register.

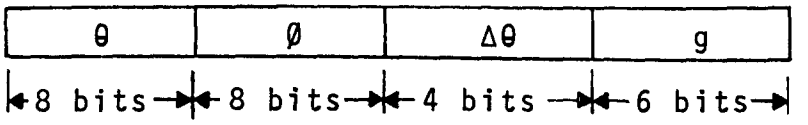
2. The coordinate transformations shown in Equation 1 may be simultaneously performed on 1024 data points. The time to perform this operation is about 8 msec. The results (θ_i, $\emptyset_i$, $\Delta\theta_i$) are then scaled to reflect:

   $\theta_{max} - \theta_{min} = 256$, and

   $\emptyset_{max} - \emptyset_{min} = 256$

   This scaling operation is done in parallel and is estimated to take about 1 msec. The total time is therefore 9 msec.

3. The intermediate results are sequentially sent to the temporary storage in STARAN control memory. Each data point is represented by a 26-bit number:



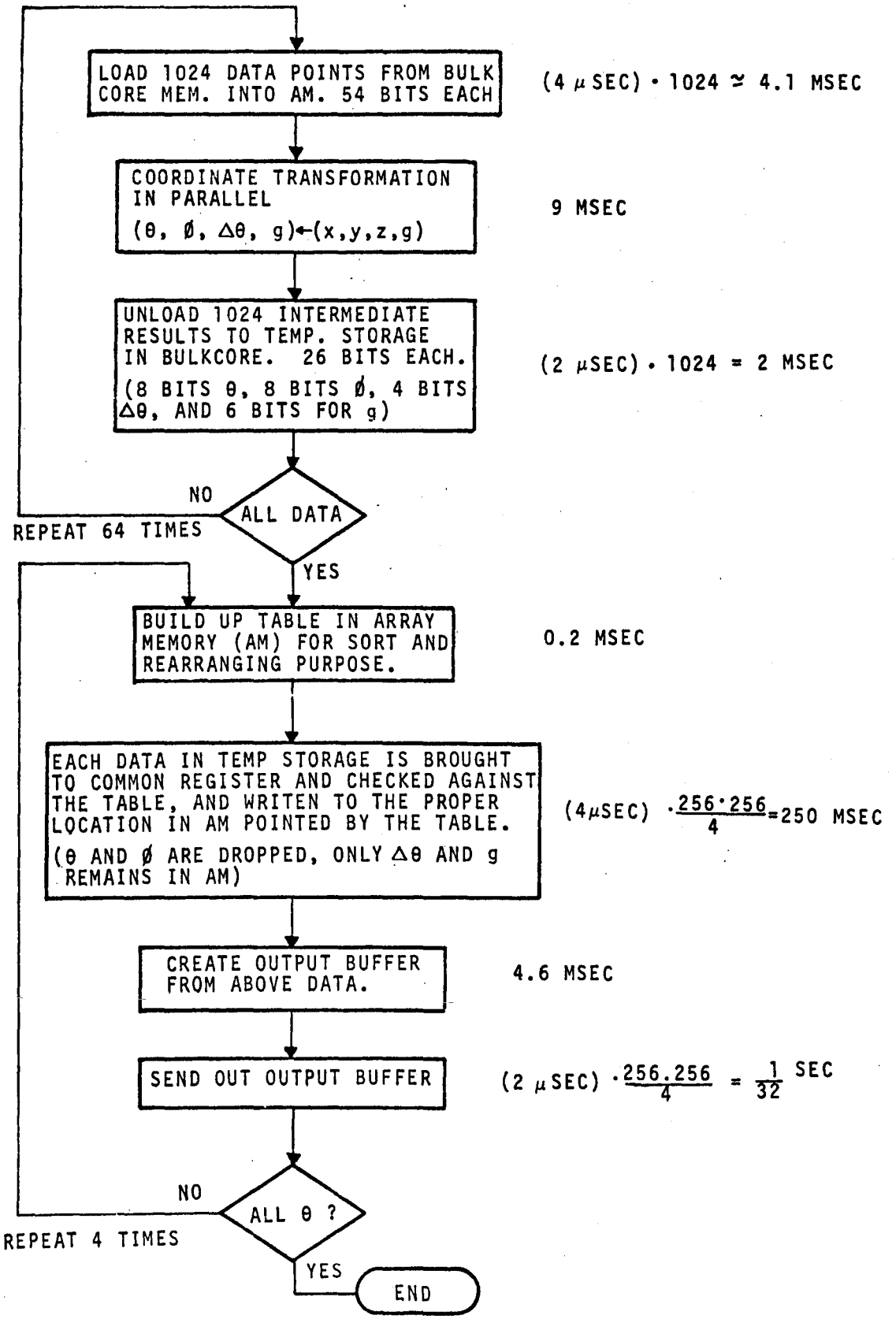


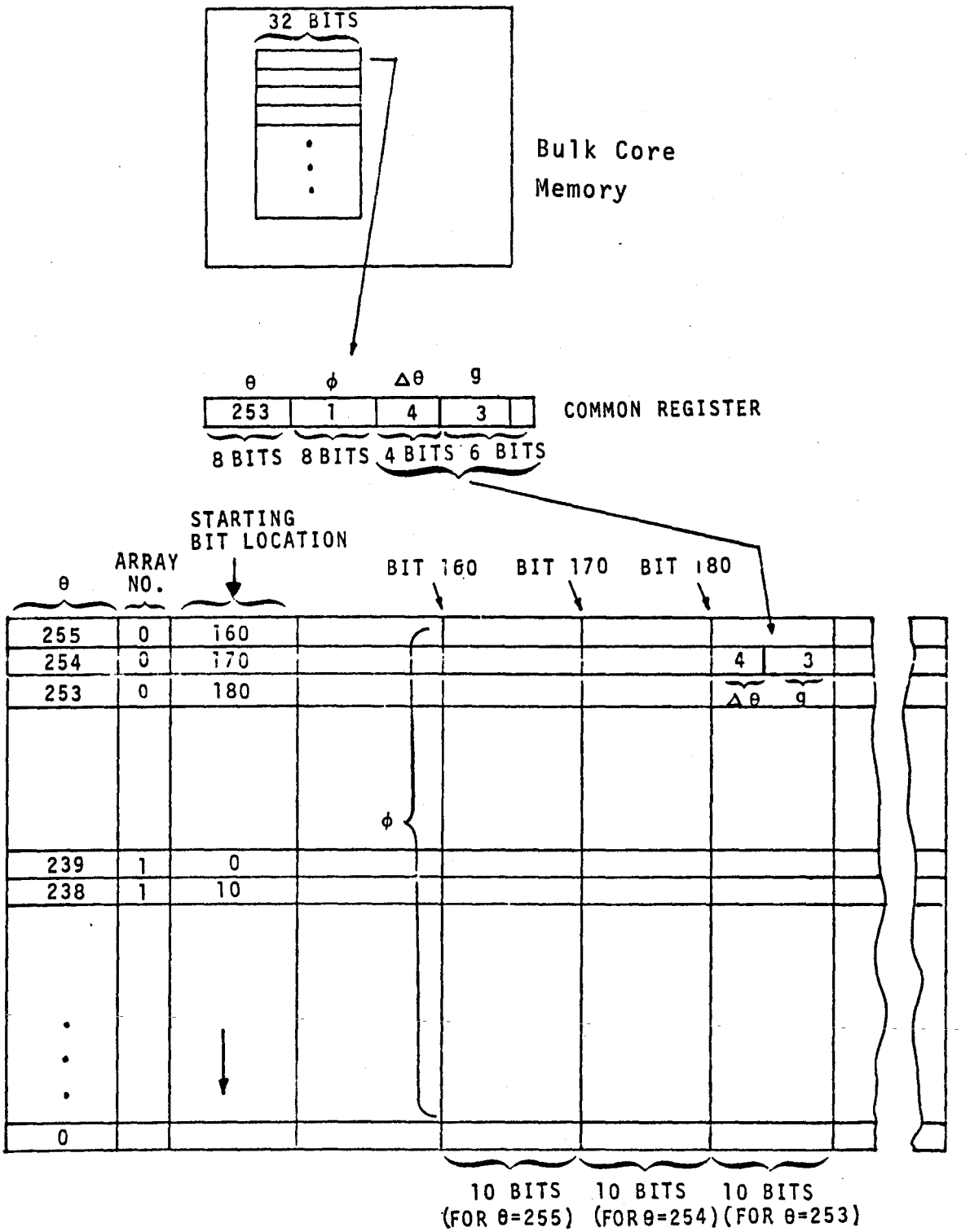
Figure D-4 - Scene Generation Flowchart and Timing

The time to unload the intermediate results is (2 μsec)·1024 = 2 msec,
where

2 μsec is the time to unload one point via the common register.

4. Due to the limited storage in AM, only 1/4 of the final output buffer is constructed at a time. A table is built for screening the intermediate file and picks up only relevant data; the first time, θ larger than 191 are picked up. The second time θ between 192 and 127, and then 128>θ>63, 64>θ. Refer to Figure D-4 for the first table. About 0.2 msec are needed to construct a table.
5. Each set of data in the entire intermediate file is brought in, one by one, to the common register and checked against the table shown in Figure D-4 for sorting and allocation. If θ falls within the specified value, the corresponding $\Delta\theta$ and g are stored in a location in AM according to the address pointed to by the table and by Ø. Figure D-4 shows an example of this operation; the table is for the case of $\theta \geq 192$. There are 256 x 256 data points to be checked; each takes an average of 4 μsec for the above operations. The total time is estimated to be

   $(4\ \mu\text{sec}) \times 256 \times 256 \simeq 1/4$ sec.
6. The final output matrix of the perspective view is structured in AM. Starting with the highest θ, the gray levels, g_i, for every Ø are written to the output buffer in AM for every θ. The number of cells to be stored depends on $\Delta\theta$. The operation is done in parallel for the Ø and sequentially according to the decreasing order of θ. The diagram in Figure D-5 will clarify this procedure.



NOTE: THIS EXAMPLE SHOWS THAT FOR ANGLE θ=253, (Δθ,g) IS STORED IN ARRAY 0, WORD 1 (DEFINED BY φ IN COMMON REGISTER), STARTING AT BIT LOCATION 180 (AS SHOWN IN THE THIRD ROW AND second column)

Figure D-4 - Tabular Storage System

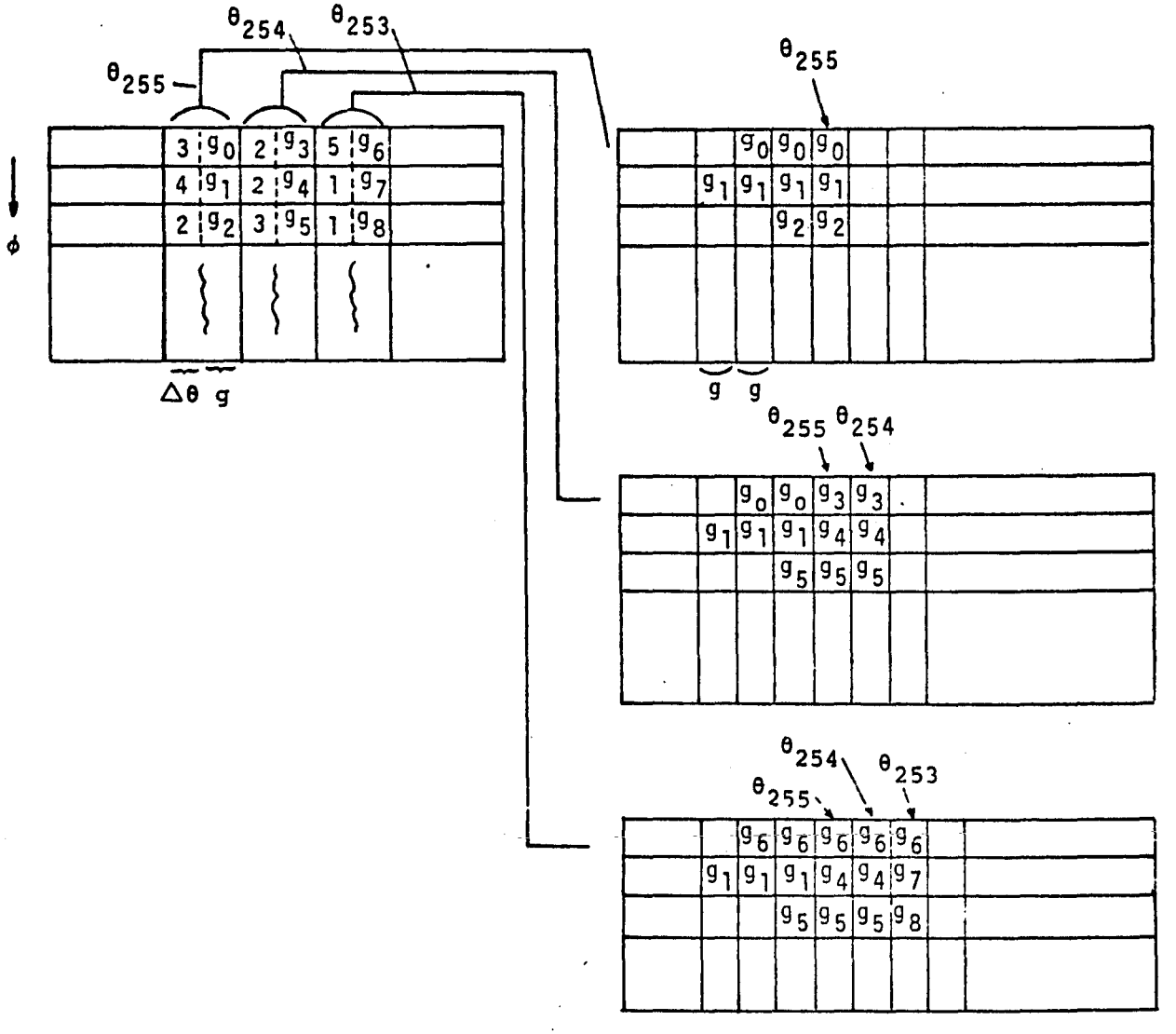


Figure D-5 - Creation of Final Output Buffer

Assuming that the average number of cells to be plotted for each set of θ is $\Delta\theta$ = 12, then the total time required is:

$$T = (\text{time for writing g}).\ \overline{(\Delta\theta)}\cdot(\text{number of } \theta \text{ to be treated})$$
$$= (6\ \mu s)(12)\cdot 64 = 4.6\ ms$$

7. Send the results from (6) to the output buffer. The time is:

$$2\ \mu s\ .\ \frac{256}{4}\ \cdot\ 256 = \frac{1}{32}\ sec.$$

By accumulating the above estimated times, the result is:

$$T = 64\ [4.1\ msec + 9\ msec + 2\ msec]\ +$$
$$4\ [0.2\ msec + 250\ msec + 4.6\ msec + 31.3\ msec]$$
$$\tilde{=} 2.11\ sec.$$

This estimated time does not include the I/O between the STARAN control memory and the outside mass storage (disc or drum). Due to the large buffer (up to 32 K words for this problem) available in the STARAN control memory and the simultaneity of I/O and STARAN control, the I/O time is essentially bounded by the transfer rate. Assume the transfer rate is 5 μsec/32-bit word (IBM 3330), than the total transfer time required for the whole operation would be 2.5 sec:

1 input of original data $\frac{10}{16}$ sec ,

1 output of intermediate file $\frac{5}{16}$ sec ,

4 inputs of intermediate file $\frac{20}{16}$ sec , and

1 final output of a data perspective view $\frac{5}{16}$ sec.

Combining this data transfer time of 2.5 sec with the STARAN processing time of 2.11 sec and taking into account the simultaneity of the operations, the total time is estimated to be less than 3 sec., which is 1 percent of the CDC 6600 processing time.

4. INSTANTANEOUS CONTOURS

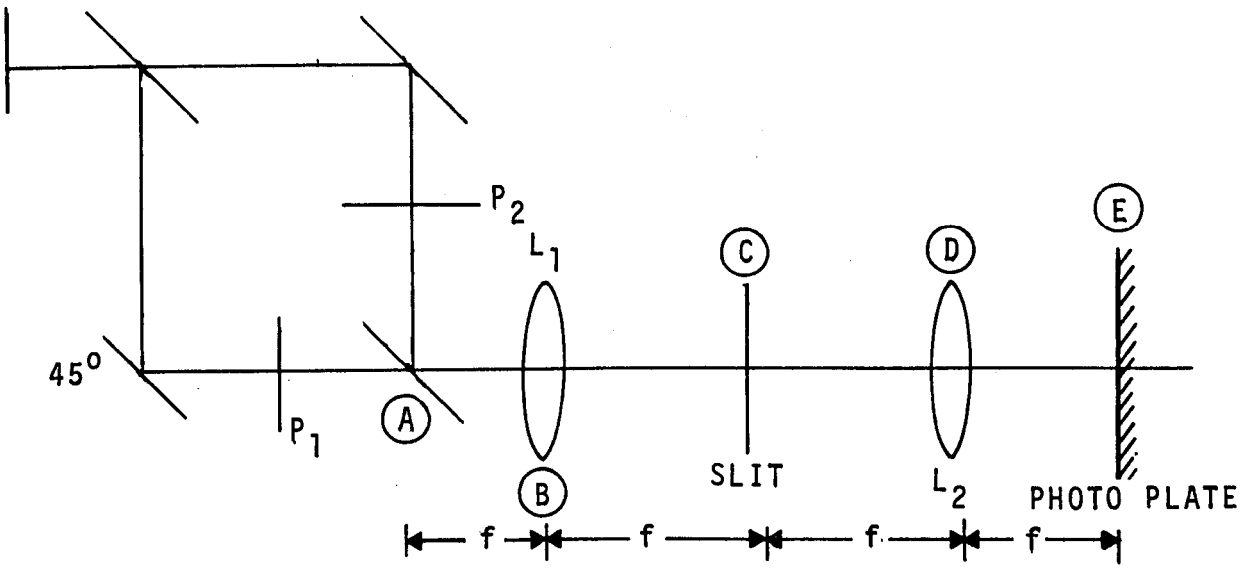
Picture processing normally done optically may be simulated in principle by a digital computer. There are several advantages to using the digital system over the optical system, however the amount of data and the large number of arithmetic operations involved in the picture processing proved to be uneconomical for the conventional sequential computer. This fact occurs because the arithmetic logic unit (ALU) is far too sophisticated to do the simple arithmetic operations normally associated with picture processing and fails to take advantage of the fact that many of the operations can be done using parallel data streams. The STARAN associative array processor with up to 8192 less sophisticated processing elements is essentially well suited to this type of application. A brief description of an optical system developed by Benton[a] to generate instantaneous contours from stereo pairs of aerial photographs is given below. A simplified diagram of the optical system is shown in Figure D-6.

The digital approach to this system using STARAN S-1000 is summarized in Table D-I. Included in the table is the estimated processing time, which is based on the assumption that each stereo pair consists of 2048 x 4096 elements and each element is represented by an 8-bit intensity (gray level).

Table D-I shows that it will take 57 sec for the STARAN to simulate the optical system. There is no corresponding data for a sequential computer approach available for comparison. But based on the FFT timing for this process (STARAN is 4 msec while sequential computer is estimated to be more than 200 msec), it appears that STARAN will outperform any sequential computer by at least a factor of 10 to 20. These times are based on a standard four-array associative array processor utilizing the 32-bit I/O channel.

---

[a]Personal communication with J. Benton, USAETL Research Institute.



P_1, P_2: THE TWO RECTIFIED STERO PAIR PROPERLY REGISTERED, X-PARALLAX ONLY

L_1, L_2: CYCLINDRICAL LENSES PERFORM THE ONE-DIMENSIONAL FOURIER TRANSFORM AND INVERSE/FOURIER TRANSFORM.

Figure D-6 - Optical System for Instantaneous Contours

In comparing the optical and the digital approach to this instantaneous contours generation problem, the STARAN S-1000 takes approximately 1 minute while the optical system obtains an immediate result. However the dedicated optical bench setup is vulnerable to the changing environmental conditions, and any change to the system parameter (say slit location or slit width) will require some careful and time-consuming realignment. The utilization of the digital system is not only insensitive to the environment conditions, but also flexible in the sense that the parameter modification to the optical system is removed from the realm of a mechanical adjustment. This removal in turn will provide a significant increase in the system's repeatability, which is important in both the laboratory and production environments.

TABLE D-I - INSTANTANEOUS CONTOURS FUNCTIONS AND TIMINGS USING STARAN S-100

Item*	Function	STARAN S-1000 Approach	STARAN S-1000 Timing
A	Sum the elements intensities of the stereo photos P_1 and P_2	The model (overlapping area of P_1, P_2 consists of 2048 x 4096 elements for each photo 2048 data points from P_1, P_2 can be loaded, and simultaneously summed. There are 4096 array loads.	(4 msec) (4096) including I/O
B	Perform a 1-D Fourier Transform in the X-direction through the cylindrical lens.	Perform a 1-D 2048-point Fast F.T. 4096 times. Assume 8-bit real input values	(4 msec) (4096)
C	Allow only a part of the resulting spectrum to pass through the slit (band-pass filter) at the FT plane.	Mask out the unwanted spectrum	negligible
D	Perform a 1-D inverse Fourier Transform through the lens.	Perform a 1-D 2048-point inverse FFT. 4096 times	(4 msec) (4096)
E	Record the resulting intensities (contours) on the photo plate. The contour interval is determined etc.	Find the intensities of above results. Output only the intensities above a given threshold.	(2 msec) (4096)
		Total	57 sec.

*Refer to the corresponding letter in Figure D-6.

APPENDIX E - STARAN PERFORMANCE DATA

This appendix contains performance data for a significant part of the STARAN APPLE mnemonics. Performance data are provided as follows:

Table E-1 - Timing for Arithmetic Operations

Table E-2 - Timing for Search Operations

Table E-3 - Timing for Logical Operations

Table E-4 - Timing for Array Data Movement

Table E-5 - Timing for Register Data Movement

Table E-6 - Timing for Input/Output Operations

Equation E-1 - FFT Subroutine Timing Extension

Table E-6 and Equation E-1 are provided for general reference and are not part of the APPLE repertoire of instructions. Equation E-1 is used to estimate the execution time of the FFT subroutine when the sample points exceed the word capacity of the STARAN selected for its execution.

Except for Table E-6, performance data represent actual execution times measured in the STARAN Evaluation and Training Facility at Goodyear Aerospace in Akron, Ohio. All instructions were executed from page memory in the speedup mode. Notes are provided with the tables and equation to define symbology and conditions pertinent to the data.

TABLE E-1 - TIMING FOR ARITHMETIC OPERATIONS

Mnemonic	Operation	Comment	Execution Time (μSec)
	Add:		
ADC	C-reg. to field-FX	$n_1 = n_2 = n_3$	$5.7+0.68n_1$
		$n_3 > n_1 = n_2$	$6.3+0.30n_1+0.38n_3$
		$n_3 \geq n_1 > n_2$	$4.1+0.19n_1+0.01n_2+0.60n_3$
ADF	Field to field-FX	$n_1 = n_2 = n_3$	$5.2+0.84n_1$
		$n_3 > n_1 = n_2$	$5.9+0.46n_1+0.38n_3$
		$n_3 \geq n_1 > n_2$	$3.7+0.35n_1+0.01n_2+0.60n_3$
FADC	C-reg to field-FL-SP		383*
FADF	Field to field-FL-SP		394*
DADF	Field to field-FL-DP		816*
	Subtract:		
SBC	C-reg. from field-FX	$n_1 = n_2 = n_3$	$5.6+0.68n_1$
		$n_3 > n_1 = n_2$	$6.4+0.30n_1+0.38n_3$
		$n_3 \geq n_1 > n_2$	$4.1+0.19n_1+0.01n_2+0.60n_3$
SBF	Field from field-FX	$n_1 = n_2 = n_3$	$5.6+0.84n_1$
		$n_3 > n_1 = n_2$	$6.4+0.46n_1+0.39n_3$
		$n_3 \geq n_1 > n_2$	$4.1+0.35n_1+0.01n_2+0.60n_3$

TABLE E-1 - TIMING FOR ARITHMETIC OPERATIONS (Cont'd)

Mnemonic	Operation	Comment	Execution Time (μSec)
	Subtract [Cont'd]:		
FSBC	C-reg. from field-FL-SP		391*
FSBF	Field from field-FL-SP		394*
DSBF	Field from field-FL-DP		816*
	Multiply:		
MPC	Field by C-reg.-FX	$(n_1 + n_2) = n_3$	~1/3 MPF
MPF	Field by field-FX	$(n_1 + n_2) = n_3$	$2.8-0.46n_1+2.0n_2+1.0n_1n_2$
FMPC	Field by C-reg.-FL-SP		~1/3 FMPF
FMPF	Field by field-FL-SP		832
DMPF	Field by field-FL-DP		3590
	Divide:		
DVC	Field by C-reg.-FX	$n_3 > n_1 > n_2$	$2.2+0.18n_1+0.43n_2+5.1(n_3-n_2)+0.89n_2(n_3-n_2)$
DVF	Field by field-FX	$n_3 > n_1 > n_2$	$4.8+0.23n_1+0.27n_2+3.0(n_3-n_2)+1.28n_2(n_3-n_2)$
FDVC	Field by C-reg.-FL-SP		772
FDVF	Field by field-FL-SP		890
DDVF	Field by field-FL-DP		3110

TABLE E-1 - TIMING FOR ARITHMETIC OPERATIONS (Cont'd)

Mnemonic	Operation	Comment	Execution Time (μSec)
	Square Root [Cont'd]:		
SQRTF	FX	$n_1 = 2n_2$	$1.1+1.6n_1+0.13n_1^2$
FSQRTF	FL-SP		632
DSQRTF	FL-DP		1968

NOTES

1. For common register operations: n_1 = array field length, n_2 = common register field length, and n_3 = field length of result.
2. For multiples: n_1 = multiplicand field length, n_2 = multiplier field length, and n_3 = product field length.
3. For division: n_1 = dividend field length, n_2 = divisor field length, and n_3 = quotient and remainder.
4. The timing for common register multiply operations is dependent upon the number of "1" groupings in the common register.
5. All above mnemonics were executed out of page memory using the "speed up bit".
6. n = field length in bits.
7. FX = fixed point.
8. FL = floating point.
9. FL-SP = floating point single precision has an 8-bit exponent and 24-bit mantissa.
10. FL-DP = floating point double precision has an 8-bit exponent and 56-bit mantissa.

* Provisional Timing

TABLE E-2 - TIMING FOR SEARCH OPERATIONS

Mnemonic	Operation	Execution Time (μSec)
	Equal:	
EQC	Field to C-reg.	0.82 + 0.19n
EQF	Field to field	0.76 + 0.48n
	Not equal:	
NEC	Field to C-reg.	0.79 + 0.19n
NEF	Field to field	0.95 + 0.48n
	Greater than:	
GTC	Field to C-reg.	0.73 + 0.19n
GTF	Field to field	3.67 + 0.46n
	Greater than or equal:	
GEC	Field to C-reg.	0.73 + 0.19n
GEF	Field to field	4.08 + 0.46n
	Less than:	
LTC	Field to C-reg.	0.73 + 0.19n
LTF	Field to field	3.68 + 0.46n
	Less than or equal:	
LEC	Field to C-reg.	0.73 + 0.19n
LEF	Field to field	4.08 + 0.46n
MAXF	Maximum field	0.84 + 0.63n
MINF	Minimum field	0.84 + 0.63n
FIND	Find first "1" in Y response store	0.24
STEP	Find the first "1" in Y and clear it	0.61
RESVFST	Find the first "1" in Y and clear all others	0.75

Notes: n = Field Length (Bits).

TABLE E-3 - TIMING FOR LOGICAL OPERATIONS

Mnemonic	Operation	Execution Time (µSec)
	Load Response Store (X, Y or M) with:	
LOR	[RS + array]	0.13§
LOR	[RS + RS]	0.13§
LORN	[RS + array*]	0.13§
LORN	[RS + RS*]	0.13§
LAND	[RS · array]	0.13§
LAND	[RS · RS]	0.13§
LANDN	[RS · array*]	0.13§
LANDN	[RS · RS*]	0.13§
LXOR	[RS ⊕ array]	0.13§
LXOR	[RS ⊕ RS]	0.13§
LXORN	[RS ⊕ array*]	0.13§
LXORN	[RS ⊕ RS]	0.13§
	Store in Associative Memory:	
SOR	[M or Y] + array	0.55
SORM	[Y + array] masked	0.68
SORN	[(M* or Y) + array]	0.55
SORNM	[Y* + array] masked	0.68
SAND	[M or Y] · array	0.55
SANDM	[Y · array] masked	0.68
SANDN	[(M* or Y*) · array]	0.55
SANDNM	[Y* array] masked	0.68

Legend:

* = Complement

\+ = Logical inclusive or

· = Logical and

⊕ = Logical exclusive or

RS = Response store

Array = Associative memory

§Add 0.26 µSec for load into M response store. When M is used for the destination, X is used for working area and therefore X cannot be a source.

TABLE E-4 - TIMING FOR ARRAY DATA MOVEMENT

Mnemonic	Operation	Execution Time (μSec)
	Load response store register:	
L	X, Y, or M from X, Y, or M	0.13
L	X, Y, or M from array	0.16
LN	X or Y from array complemented	0.16
LN	M from array complemented	0.29
	Store response store register:	
S	X, Y, or M into array	0.25
SM	X, or Y masked into array	0.39
SN	Y or M complement into array	0.39
SNM	Y complement masked into array	0.52
	Load common register:	
LC	From an associative memory word	0.72 to 1.3*
LCM	Field from associative memory word	0.45 to 1.1*
LCW	From a response store register	0.45
	Store common register into:	
SC	Associative memory	0.64 + 0.57n
SC	Response store register	0.13
SCW	Associative word	0.74 to 1.14*
SCW	Response store register	0.13
	Set response store:	
SET	X or Y	0.13
SET	M	0.26
	Clear response store:	
CLR	X or Y	0.13
CLR	M	0.26
	Rotate:	
ROT	Response store register or common register	0.13 μsec per shift**
	Move within array:	
MVF	Field	0.58 + 0.61n
MVCF	One's complement of a field	0.58 + 0.61n
MVNF	Negative of a field	3.84 + 0.69n
MVAF	Absolute value of a field	3.90 + 0.94n
INCF	Field with increment	3.83 + 0.69n
DECF	Field with decrement	3.57 + 0.81n

*If field alignment shifting is required, then the higher execution times will be experienced.

**M - response store requires 0.26 μsec additional execution time.

n = Field length (bits)

TABLE E-5 - TIMING FOR REGISTER DATA MOVEMENT

Mnemonic	Operation	Execution Time (μsec)
	Load register:	
LRR	From register	0.27*
LI	With immediate data	0.26
LR	From HSDB	0.73**
LR	From bulk core	1.35**
	Store register	
SR	In HSDB	0.70**
SR	In bulk core	1.42**
INCR	Increment register	0.25
DECR	Decrement register	0.25

* If R0 through R7 [return jump registers] is selected as either a source or destination register, then 400ns is added to the execution time.

** Address modification via the Data Pointer (DP) adds 140 nsec while the R0 through R7 address modification option adds 400 nsec.

NOTE: HSDB = High speed data buffer within control memory.

TABLE E-6 - TIMING FOR INPUT/OUTPUT OPERATIONS

Channel Name	Channel Width	Transfer Rate
Buffered input/output [BIO]	32 Bits	0.4-1.0 µsec/word[1]
Direct memory access [DMA]	32 Bits	0.3 µsec/word[2]
External function [EXF]	19 Bits	0.2-2.0 µsec/word[3]
Parallel input/output [PIO]	256 Bits/array[4]	0.3 µsec/word[5]

Note:

[1]Both the high-speed data buffer [HSDB] and bulk core can be addressed by an external device.

[2]The actual rate is a function of the external device being addressed by STARAN.

[3]This varies as a function of the EXF code and resolver usage.

[4]Can have a maximum of 32 arrays for 8192 bits.

[5]This is a maximum rate. Typical parallel head discs (PHD) have average access times of 8.5-17 msec and 256-bit channel transfer rates of 0.5-1.0 µsec/bit.

EQUATION E-1 - FFT SUBROUTINE TIMING EXTENSION

$$T_{FFT} = n\ (T_W + 0.5\ [\log_2 n]) + \underbrace{0.006Wn\ (1+[\log_2 n])}_{I/O}.$$

Where

T_{FFT} = Time to compute the FFT of a function with more points than the STARAN word capacity [msec]; applies by complex data with the real and imaginary terms each represented by 16-bit field lengths,

n = $[\ \frac{E}{W}\ ]$

E = Number of sample points

W = STARAN capacity in words (1024),

T_W = 5.0 msec for a 1024-point, 16-bit field FFT, and

I/O term = Data transfer time via common register, and

$[\ ]$ = If fractional, use next higher integer.

APPENDIX F - SPECIAL STARAN CONFIGURATIONS

1. INTRODUCTION

The flexibility of a standard STARAN* associative array processor may not be required in some multiple device production environments and a more cost effective solution may result from the development of a "stripped down" version of the AAP concept. Additionally, some applications may demand specialized options to permit effective utilization of the STARAN or enhance its performance.

In addition to marketing the standard STARAN S series AAP, Goodyear Aerospace (GAC) has extensive AAP hardware and software experience. Thus, GAC can efficiently develop and deliver specialized AAP system configurations and options that meet specific requirments.

In this Appendix, the standard STARAN S series architecture is reviewed and variations to this standard are discussed. Brief descriptions are provided for the Parallel Input/Output Module option and some approaches to a random access parallel input/output memory and hardware floating-point arithmetic.

2. STANDARD STARAN S Series Architecture

2.1 General

A top-cut diagram of the STARAN main frame is shown in Figure F-1. It consists of a conventionally addressed control memory for program storage and data buffering, a control logic unit for sequencing and decoding instructions from control memory and from one to thirty-two modular AAP arrays.

A typical AAP array is also shown in Figure F-1. This key element of the STARAN S computer system is the "main frame" memory which provides content addressability and parallel processing capabilities. Each array consists of 65,636 bits organized as a multi-dimensional access memory matrix of 256 words by 256 bits with parallel access to up to 256 bits at a time in either the word or bit direction. In addition to the storage elements, each array contains 256 bit-serial PE's often referred to in associative

---

*T. M. Goodyear Aerospace Corporation, Akron, Ohio 44315.

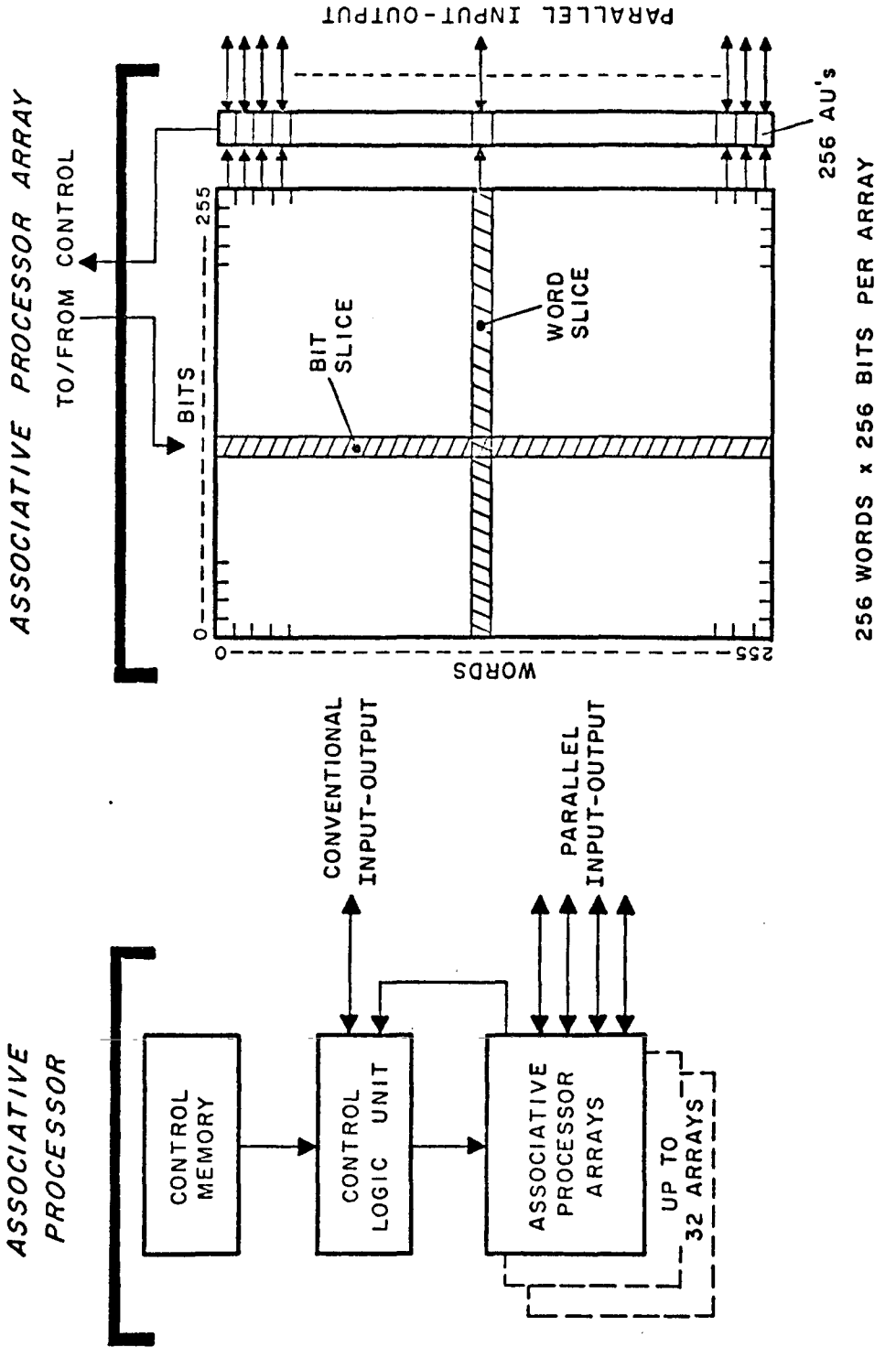


Figure F-1 - Associative Processor Diagrams

memory literature as the response store. The unique PIO capability is provided by the response store, where every PE has an independent external device I/O path. Control signals generated by the control logic unit are fed to the processing elements in parallel and all processing elements execute the instruction simultaneously. As additional arrays are added to the system these are also connected in parallel to the control logic unit, thus application programs need not be modified as the capacity of the system increases.

Major elements of the STARAN block diagram shown in Figure F-2 are described below.

2.2 AP Control Memory

The conventionally addressed and indexed AP control memory is used to store assembled AP application programs. It is also used for data storage and to act as a buffer between AP control and other elements of STARAN S. The AP control memory and associative array cycles are overlapped.

Control memory is divided into several memory blocks. Three fast "page" memories contain the current AP program segments; the slower core memory contains the remainder of the AP program. A program pager transfers program segments from the slow to the fast memory blocks. Control memory words contain 32 bits of either data or instructions.

The "page " memories use volatile, bipolar, semi-conductor elements. A page contains 512 words but can be doubled to 1024 words each on an optional basis. Page 0 may contain a library of microprograms such as arithmetic subroutines, Pages 1 and 2 are used in ping-pong fashion, with AP control executing instructions out of one page while the other is being loaded by the program pager. This permits use of page memories for selected segments of the program or for the entire program if fast execution is required.

The high-speed data buffer (HSDB), like the page memories, uses volatile, bipolar, semi-conductor elements. It contains 512 words

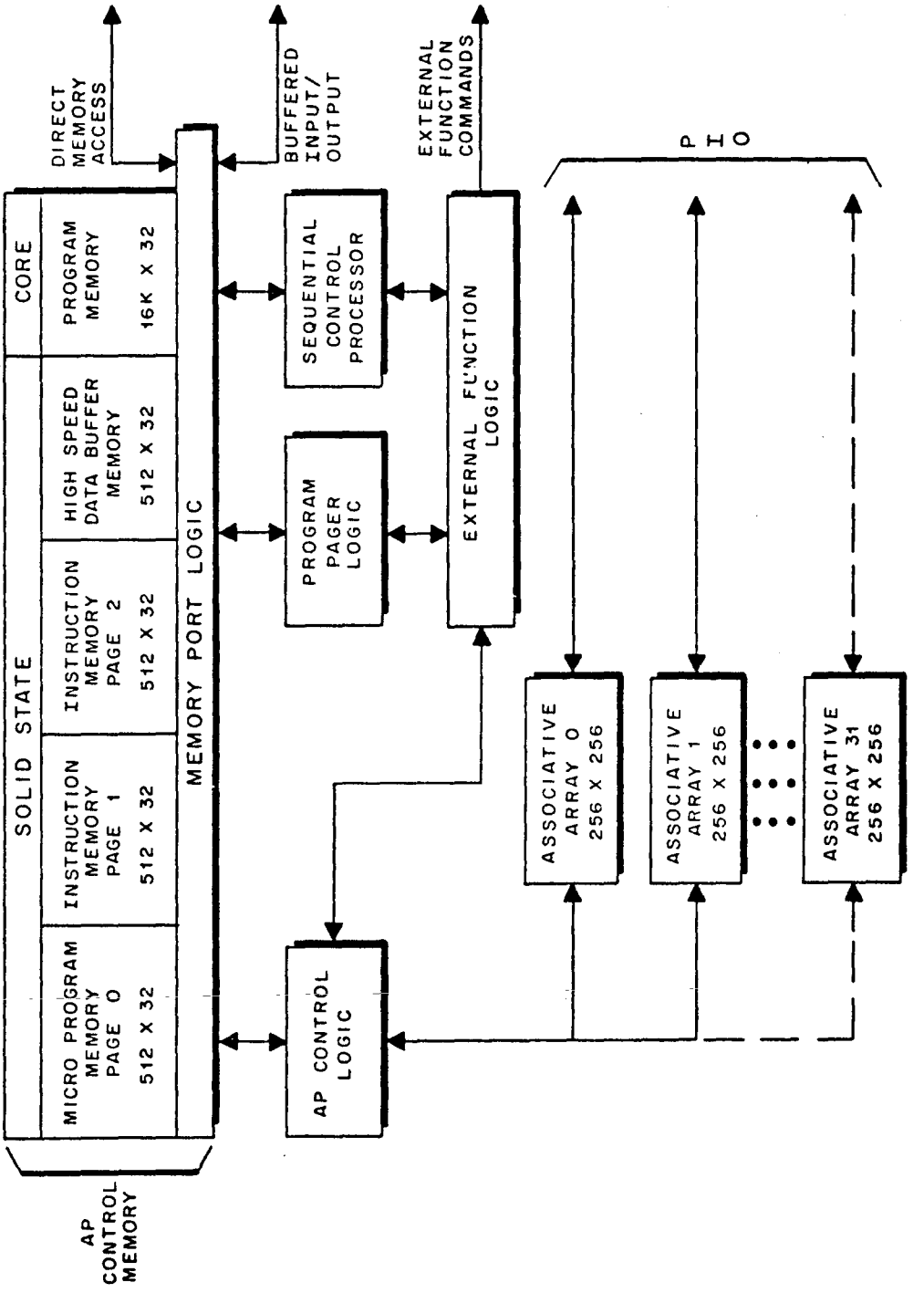


Figure F-2 - STARAN Basic Block Diagram

but also can be doubled to 1024 words. All buses can access the HSDB to store data or instruction items that need to be accessed quickly by the different STARAN elements.

The bulk core memory uses nonvolatile core storage. It contains 16,384 words and is optionally expandable to 32,768 words. It is used for storing complete AP application programs. Since the bulk core memory is accessible to all buses it is useful as a buffer for data items that do not require the high-speed of the HSDB.

A block of up to 30,720 AP control memory addresses is reserved for the direct memory access (DMA) channel to external memory. All buses can access the DMA block, thus it is possible to operate the AP solely from programs stored on external memory as, for example, the main frame memory of a conventional computer.

2.3 AP Control Logic

Executing instructions from control memory, AP control logic directly manipulates data within the associative arrays and is the data communication path between control memory and the arrays.

2.4 Program Pager Logic

The program pager loads the fast page memories from the slow core memory. While the AP control is executing a program segment out of one page, the pager can be loading the other page with a future program segment.

2.5 External Function Logic

External function (EXF) logic enables the AP control, sequential control, or an external device to control the STARAN S operation. By issuing external function codes to EXF a STARAN S element can interrogate and control the status of the other elements.

2.6 Sequential Control Processor

The sequential control (SC) portion of STARAN S consists of a sequential processor having an 8K 16-bit memory, a keyboard-printer, a perforated tape reader/punch unit, and logic capability to interface the sequential processor with other STARAN S elements. SC is used for system software programs such as assembler, operating

system, diagnostic programs, debugging, and housekeeping routines. SC peripherals which may be useful in programming aids are available as options.

2.7 Input/Output Options

A variety I/O options include conventional direct memory access (DMA), buffered I/O (BIO) channels, external function channels (EXF) and a unique interface called parallel I/O (PIO).

The direct memory access (DMA) is a 32-bit bus for STARAN to address external memory. The AP control or the sequential controller can access external memory at a rate dependent upon this memory's cycle time.

The buffered I/O (BIO) is a 32-bit bus for processors to address STARAN. Depending upon which portion of control memory is accessed, the access rate is 0.4 to 1.0 microsecond per 32-bit word.

The external function (EXF) is a bus for exchange of control signals. Discrete signals and interrupts can be both generated and accepted across this bus.

The parallel I/O (PIO) is a bus for STARAN array I/O. Up to 256 bits per array (e.g., one bit per array word) can be provided. If all 32 arrays are implemented, up to 8192 bits can be utilized in parallel at a transfer rate of approximately 0.5 microsecond, dependent upon the desired applications.

3. VARIATIONS TO STARAN ARCHITECTURE

Variations to STARAN architecture fall into two categories. In one case the major STARAN components can be added or deleted to effect a specialized configuration. For the second case, a major component can be modified to effect its capacity or performance. In general, the former would require less of a development effort than the latter. An example of each category is briefly described below.

Using the available major STARAN components. A mini-AAP can be derived by retaining bulk core memory for data buffering, one solid state memory for high speed program execution, the AP control

logic, and one or more associative arrays. Such a configuration is shown in Figure F-3. Many such "mini-AAP" (with or without the bulk core memory) could access a common data bank while performing different functions dictated by software executed from their individual solid state memories. Other variations to STARAN architecture of the type just described can be realized, with a minimal development effort, to satisfy specialized applications in a cost effective manner.

The second type of STARAN variation involves the modification of a major component such as the associative arrays. The arrays presently utilize large scale integrated circuit memory chips such as their basic storage media. With the availability of larger capacity high speed memory chips, the array word size could conceivably be increased from 256 bits to 1024 or eventually 2048 bits.

For the ETL applications considered to date the large array size does not appear to offer any significant performance advantage. However, development of the "mini-AAP" concept may very well offer a high performance, low cost advantage for deployment of multiple devices for a production application.

With the flexibility of a standard STARAN in a test bed environment such as that proposed for ETL's CSL, various "mini-AAP" configurations can be "designed" and thoroughly evaluated, thus, permitting the specification of low risk processing equipment for various production tasks.

4. PARALLEL INPUT/OUTPUT MODULE OPTION

The PIO module is a flexible, micro-programmable device with seven 256-bit ports and one 32-bit port as shown in Figure F-4. Each port is capable of communicating internally with any other port. Therefore, under software control, data can be transferred internally between any selected pair of ports with data transfers occurring simultaneously between multiple port-pairs. This arrangement permits very high speed data transfer paths (1) between associative arrays, (2) between arrays and one or more mass storage devices (such as a 256-channel parallel head disk/drum), and (3) between arrays and

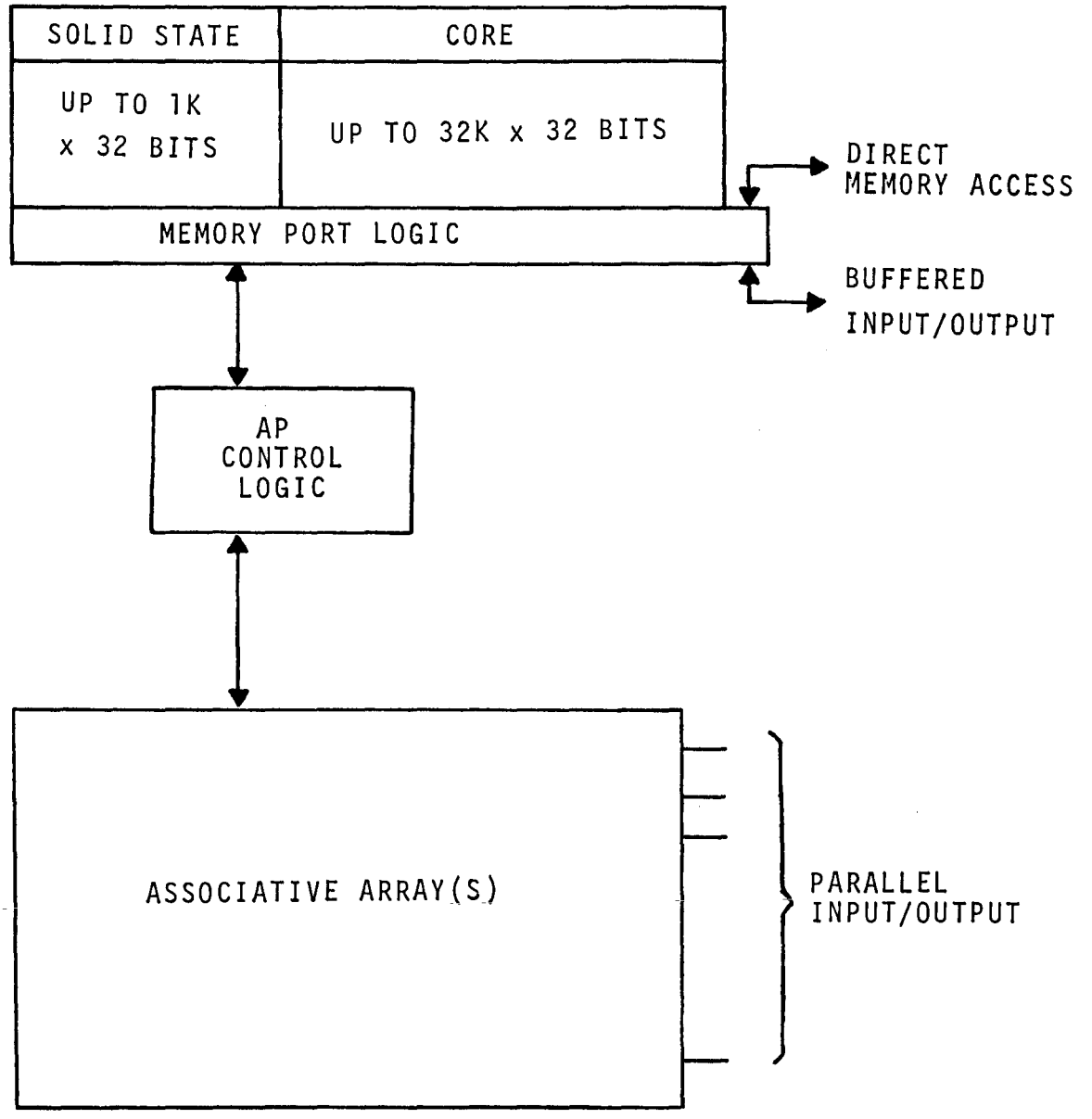


Figure F-3 - Mini-AAP Configuration

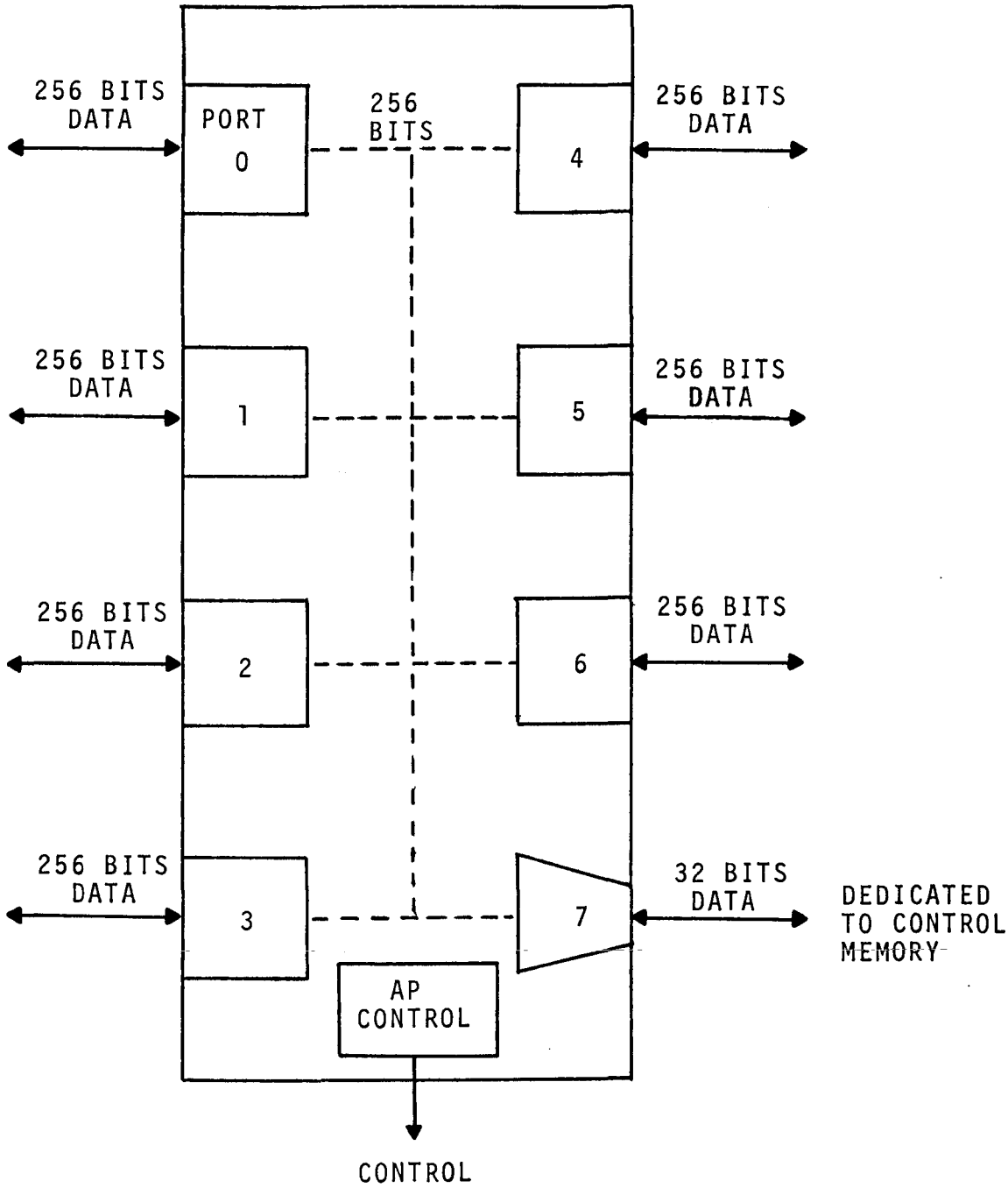


Figure F-4 - Parallel Input/Output (PIO) Module

special-purpose hardware that could be developed in the future to enhance the associative array processor performance on ETL tasks.

The PIO module and mainframe AP control can operate concurrently with their respectively selected arrays. This permits, for example, the execution of associative instructions on selected arrays while I/O operations are being performed on the remaining arrays.

The PIO module is:

1. A proven device with two existing operational systems
2. A flexible user-oriented stored program device
3. A full complement of tested software
4. A complete set of diagnostic programs
5. A complete set of documentation

5. RANDOM ACCESS PIO MEMORY

For some applications involving many transfers of array data (for example inversion of larger matrices) can be enhanced by utilizing its (PIO) interface channel with an appropriate random access memory for data storage. To be effective, the memory would have a 256 bit wide interface channel and have a transfer rate of one 256 bit word per 0.5 microsecond. There are many large memory systems presently under development using holographic and bubble memory techniques. These approaches to a mass memory system will eventually be an enhancement for the STARAN AAP system.

Presently, however, it would be necessary to develop a random access PIO memory with core or solid state memory elements. In either case, a reasonable approach to the problem would be to "gang" together multiple standard memory units. For example, eight standard 32 bit by 16,384 word memories could be read or written simultaneously to appear as one 256 bit by 16,384 word memory.

With this approach, a master control unit would be required to simultaneously drive the appropriate control lines of the individual memory units. As the cost per memory cells continues to drop, this approach will become more attractive. This approach is not intended to compete with mass memory storage systems.

6. HARDWARE FLOATING-POINT ARITHMETIC

The STARAN AAP was developed using simple processing elements and software arithmetic in order to effect a desirable cost-performance tradeoff that would be advantageous for most applications. Therefore, STARAN computation time is proportional to field size and for some tasks involving large data fields a four array STARAN becomes comparable to a high performance sequential computer such as a Univac 1108. However, some applications have critical timing constraints and can justify the increased cost required to develop and produce more complex associative array processing elements that will significantly improve STARAN performance for double-precision floating-point arithmetic.

The STARAN architecture is very conducive to the incorporation of hardware floating-point arithmetic as shown by the two approaches described below.

As illustrated in Figure F-1, each of STARAN's associative array words can simultaneously communicate with external equipment via the PIO interface channel. Therefore, the appropriate data fields in all words of all arrays can be simultaneously transferred to individual hardware arithmetic units in a bit-serial manner. The desired arithmetic operations would then be performed and the results transferred back to the arrays in a bit-serial manner. This approach would require 256 arithmetic units per array. The execution times are estimated at approximately 3 microseconds per bit.

Another approach requiring considerably less hardware would utilize STARAN's unique mixed mode operation. As shown in Figure F-5, four 64 bit fields of four array words (spaced 64 words apart) can be simultaneously read from the array as a single 256 bit data slice. In general, any 2^N (where N = 1 through 8) field size can be selected for mixed mode operation. The number of word participating in the read (or write) and the spacing between words will be $\frac{256}{2^N}$. Therefore, using mixed mode to perform double-precision floating-point arithmetic would then require the following steps:

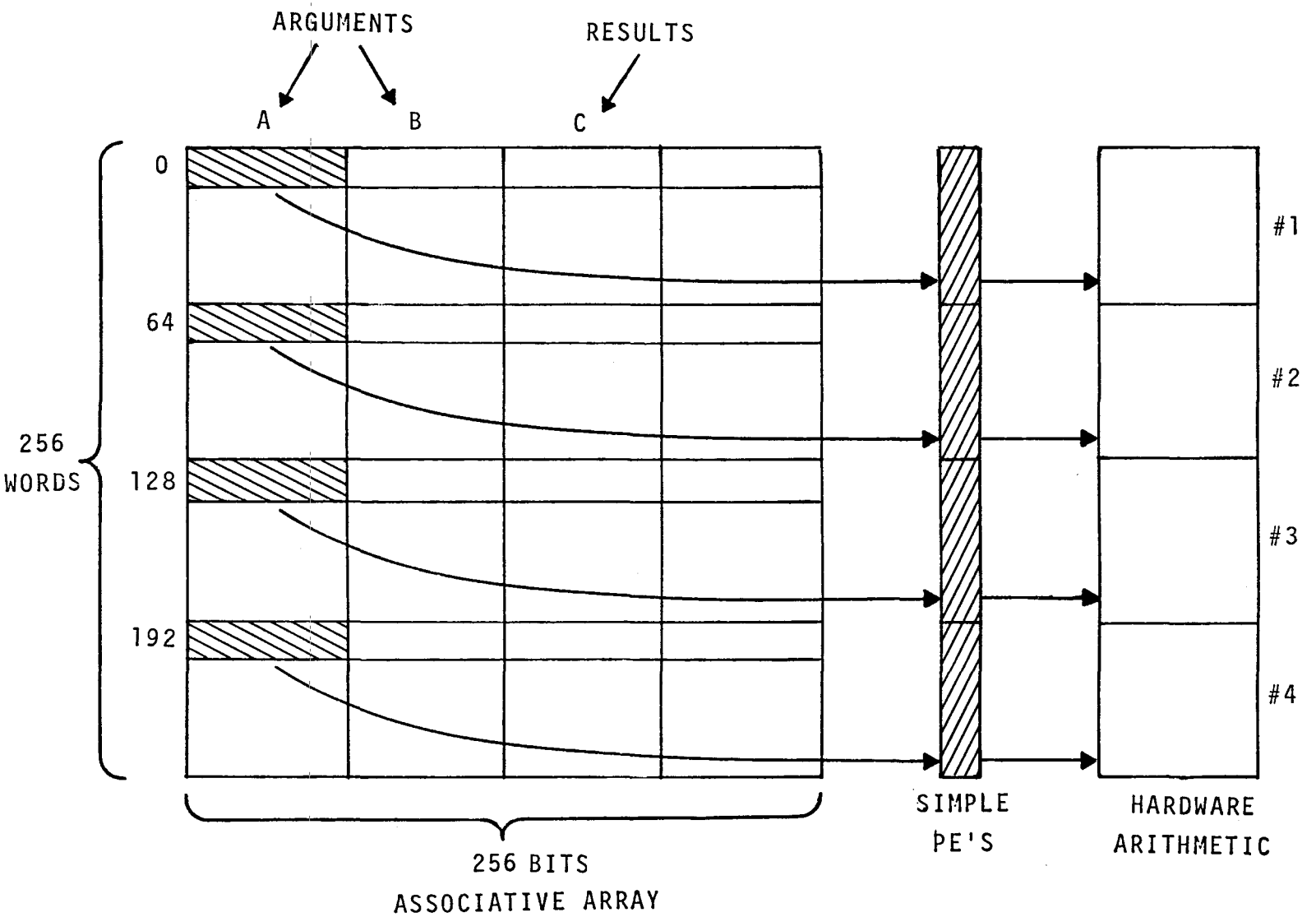


Figure F-5 - Mixed Mode Data Transfer

1. Read the four arguments from field A into the four hardware arithmetic units as shown in Figure F-5.
2. Read the four arguments from field B into the arithmetic units.
3. Perform the arithmetic operations in the hardware arithmetic units.
4. Write the four results into field C of the array. Fields A or B could be used for the results if the original arguments are not to be retained.
5. Repeat steps 1 through 4 for the other 63 groups of the array's argument pairs.

The advantages of the first hardware arithmetic approach (256 units per array) are:

1. Any field length up to the maximum size of the arithmetic unit (say 64 bits) can be selected to participate in the operation.
2. The arithmetic units need not be as sophisticated as the second approach for the same resultant speed.

The advantages of the second (mixed mode) approach are:

1. Considerably less hardware required.
2. Any 2^N (where N = 1 through 8) field size can participate in the operation while retaining maximum array utilization.

If the computation time of the hardware units could be ignored, then the overall execution time would be a function of the data transfer time and both approaches would yield the same performance. In reality however, the performance of the second approach is much more dependent upon the speed of the arithmetic units. The best approach will be a function of the application, and a tradeoff analysis between many low performance arithmetic units and considerably fewer high performance devices. This discussion describes the hardware arithmetic as an external option, however it could be incorporated as a mainframe modification.

APPENDIX G - TREE SUMMING

The effective "between word" communication inherent in the STARAN Associative Array Processor provides the user with a fast parallel technique for summing values in 2^N consecutive words. This technique is called tree summing.

The following example (Figure G-1) shows the method of adding 16 values $a_1, a_2, \ldots a_{16}$. The steps below show the operations required to perform this summation:

STEP	FUNCTION	MAJOR OPERATIONS REQUIRED
1)	Shift $a_9, a_{10}, --a_{16}$ up 8 words and add $a_1+a_9=b_1, a_2+a_{10}=b_2, \ldots a_8+a_{16}=b_8$	1 addition
2)	Shift b_5, b_6, b_7, b_8 up four words and add $b_1+b_5=c_1, b_2+b_6=c_2, \ldots b_4+b_8=c_4$	1 addition
3)	Shift c_3, c_4 up two words and add $c_1+c_3=d_1, c_2+c_4=d_2$	1 addition
4)	Shift d_2 up one word and add d_1+d_2 = result	1 addition

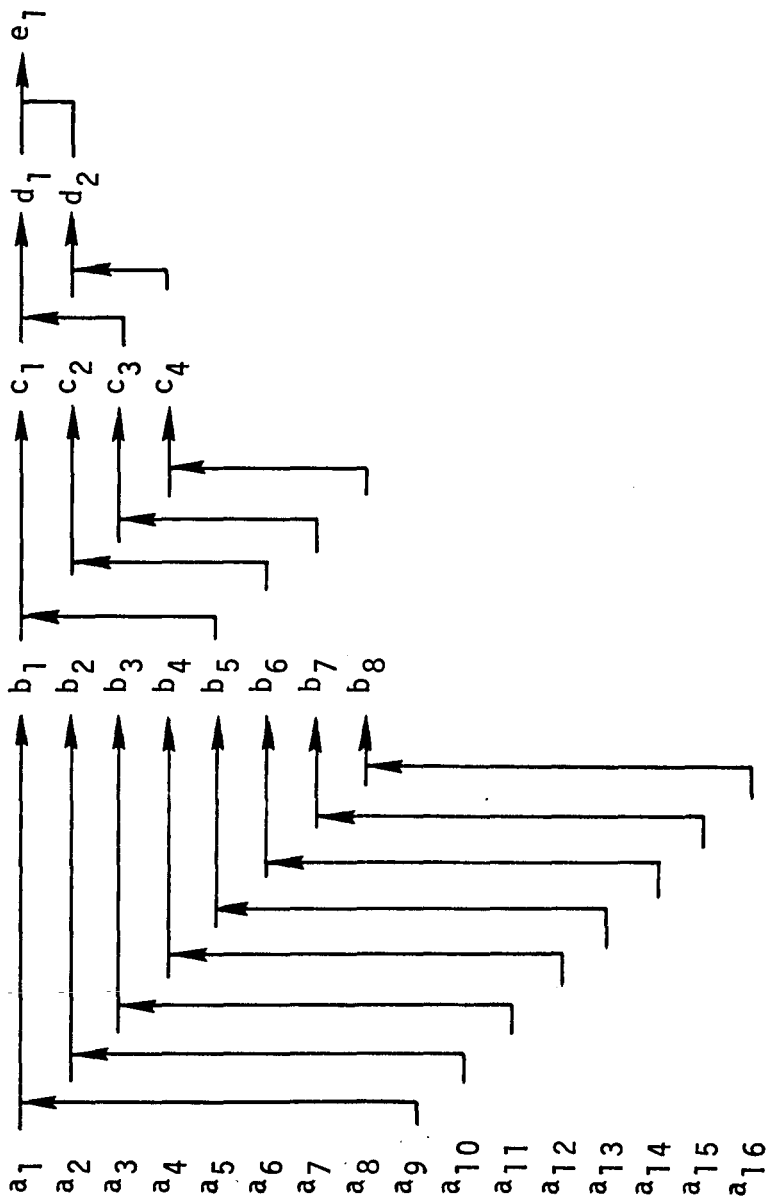


Figure G-1 - Tree Sum Example

The total number of major operations is given by:

$(1+1+1+1)=4$ additions

This "tree sum" technique may be generalized to accumulate 2^N values where N=1,2,3,...8, the number of additions being $\log_2 2^N=N$.