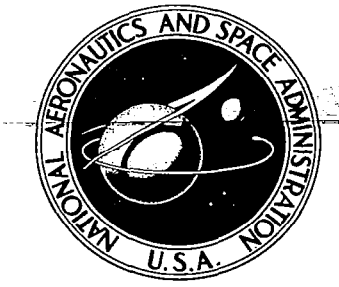
TECH LIBRARY KAFB, NM
0061434

NASA CONTRACTOR REPORT



N.A.S.A. - C.R.

LOAN COPY: RETURN TO
AFWL TECHNICAL LIBRARY
KIRTLAND AFB, N. M.

NASA CR-2730

MIDAS, PROTOTYPE MULTIVARIATE INTERACTIVE DIGITAL ANALYSIS SYSTEM FOR LARGE AREA EARTH RESOURCES SURVEYS

Volume I: System Description

D. Christenson, M. Gordon, R. Kistler, F. Kriegler, S. Lampert, R. Marshall, and R. McLaughlin

Prepared by
ENVIRONMENTAL RESEARCH INSTITUTE OF MICHIGAN
Ann Arbor, Mich. 48107
for Langley Research Center

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • APRIL 1977

TECH LIBRARY KAFB, NM
0061414

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-2730		

4. Title and Subtitle	5. Report Date
MIDAS, PROTOTYPE MULTIVARIATE INTERACTIVE DIGITAL ANALYSIS SYSTEM FOR LARGE AREA EARTH RESOURCES SURVEYS. Volume I: System Description	April 1977
	6. Performing Organization Code
7. Author(s) D. Christenson, M. Gordon, R. Kistler, F. Kriegler, S. Lampert, R. Marshall and R. McLaughlin	8. Performing Organization Report No. ERIM 108800-49-F
9. Performing Organization Name and Address Infrared and Optics Division Environmental Research Institute of Michigan P.O. Box 618 Ann Arbor, MI 48107	10. Work Unit No.
	11. Contract or Grant No. NAS1-13128
	13. Type of Report and Period Covered Final Report, April 1974-September 1975
12. Sponsoring Agency Name and Address Langley Research Center National Aeronautics and Space Administration Hampton, Virginia 23665	14. Sponsoring Agency Code

15. Supplementary Notes

Volume I of II. Mr. William Howle was Technical Monitor.
Topical report.

16. Abstract

MIDAS is a third-generation, fast, low cost, multispectral recognition system able to keep pace with the large quantity and high rates of data acquisition from large regions with present and projected sensors. MIDAS, for example, can process a complete ERTS frame in forty seconds and provide a color map of sixteen constituent categories in a few minutes. A principal objective of the MIDAS Program is to provide a system well interfaced with the human operator and thus to obtain large overall reductions in turn-around time and significant gains in throughput.

This report describes the hardware and software generated in the overall program. The system contains a midi-computer to control the various high-speed processing elements in the data path, a preprocessor to condition data, and a classifier which implements an all-digital prototype multivariate-Gaussian maximum likelihood or a Bayesian decision algorithm operating at 2×10^5 pixels/sec. Sufficient software has been developed to perform signature extraction, control the preprocessor, compute classifier coefficients, control the classifier operation, operate the color display and printer, and diagnose operation.

Volume I describes the MIDAS in detail. Volume II documents studies of several forms of advanced processing and requirements therefore.

17. Key Words	18. Distribution Statement
Real-time processing Interactive Low-cost Wirewrap MIDAS Real-time processing Pipeline Bayesian decision rule Multispectral recognition system Multivariate-Gaussian statistics	Unclassified — Unlimited Subject Category 62

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
UNCLASSIFIED	UNCLASSIFIED	135	$5.75

PREFACE

A comprehensive multispectral program devoted to the advancement of state-of-the-art techniques for earth resources surveys has been a continuing program at the Environmental Research Institute of Michigan (ERIM), formerly the Willow Run Laboratories of The University of Michigan. The basic objective of this multi-disciplinary program is to develop remote sensing as a practical tool to provide the user with processed information quickly and economically.

The importance of providing timely information obtained by remote sensing to such people as the farmer, the city planner, the conservationist, and others concerned with problems such as crop yield and disease, urban land studies and development, water pollution, and forest management must be carefully considered in the overall program. The scope of our program includes: (1) extending the understanding of basic processes; (2) discovering new applications; (3) developing advanced remote-sensing systems; (4) improving fast automatic data processing systems to extract information in a useful form; and (5) assisting in data collection, processing, analysis and ground truth verification. The MIDAS Program, with its improved data processing capability, applies directly to No. (4).

The overall program is guided by Mr. R. R. Legault, a Vice President of ERIM and Director of the Infrared and Optics Division. Work on this contract was directed by J.D. Erickson, Head of the Information Systems and Analysis Department, R. McLaughlin, Head of the Processing Systems Development Section and by F.J. Kriegler, Principal Investigator. Volume I of this report covers the system description in detail. Volume II discusses several forms of advanced processing and applications requirements.

In addition to providing the text, the authors' individual contributions were as follows: Dempster Christenson, Vernon Smith, and Michael Gordon provided system programming and diagnostic software; Roland Kistler, Rowland McLaughlin, and Seymour Lampert provided the detailed design and performed system checkout; Robert Marshall aided in overall system configuration and organized this report. Michael Schlansker and Prof. Daniel Atkins, consultants from the Electrical Engineering Department of The University of Michigan, also contributed to this program by developing the APL simulation of the processor, and by developing the

design approach to the pipeline divider sub-system in the preprocessor. The authors wish to acknowledge the direction provided by Mr. R.R. Legault and Dr. J.D. Erickson. Outstanding contributions were made by the following persons: John Baumler, Clyde Connell, William Juodawlkis, Robert Pierson, Cary Wilson, and Nancy Wilson for their efforts in system construction.

This report describes the scope of and the hardware and software generated in the overall MIDAS program. In its development certain commercial products were utilized and are identified in this report in order to specify adequately the conditions and products used in the research effort. In no case does such identification imply recommendation, endorsement, or evaluation of these products by NASA, nor does it imply that such products are necessarily the only ones or the best ones available for such applications. In many cases equivalent products are available and would probably produce equivalent results.

CONTENTS

FIGURES

TABLES

MIDAS, PROTOTYPE MULTIVARIATE INTERACTIVE DIGITAL ANALYSIS SYSTEM FOR LARGE AREA EARTH RESOURCES SURVEYS
Volume I: System Description

THE MIDAS PROGRAM

1.1 SUMMARY

The MIDAS system, while larger than its implementation in concept, has now been implemented as a prototype processor for remotely sensed data. With it, a user may load data from tape into the system, examine and analyze various portions of the scene using the color display or color printer and then perform corrections, enhancements or recognition operations on the data, obtaining these results in color maps or tabular outputs. The pipeline operates at a processing rate of 2×10^5 elements/second, or less if desired, for any processing operations to be performed. At this rate, for example, MIDAS could classify a LANDSAT -1 or 2 frame of data in about 40 seconds and provide color maps in a time ranging from about five minutes to about an hour depending on the resolution desired. In fact, however, the time to carry out such a procedure could range between a minimum of about one-half hour to about four hours, depending for the most part on the user's familiarity with his problem and the data. MIDAS now provides a valuable tool for multispectral processing, enabling and facilitating progress in applying remotely sensed data to the analysis, management and utilization of natural resources.

1.2 INTRODUCTION

With the launching of the Earth Resources Technology Satellite (now LANDSAT-1) in 1972 and the manned Skylab with its Earth Resources Experiment Package in 1973, the NASA Earth Resources Survey Program began employing large-scale space technology to add to the previous programs of remote sensing from aircraft.

Earth resources information systems to aid in the inventory, allocation, and management of Earth's recources make use of a combination of disciplines. These systems employ a priori knowledge of common practice and ecological relationships, modern sensors and data processing equipment, information theory and processing methodology, communications theory and devices, space and airborne vehicles, and large-systems theory and practice. There are, of course, many different remote sensing techniques—e.g., gravity and seismic sensors; acoustic sensors; static magnetic- and electric-field sensors; gamma- and x-ray sensors; sensors of electromagnetic radiation in the ultraviolet, visible, infrared, microwave, and radiofrequency regions of the spectrum. The remote sensing techniques which can be used from aircraft or spacecraft are, however, limited to sensing electromagnetic radiation from the ultraviolet through radio frequencies. The basic foundation of remote sensing is the use of these transducer and sensor outputs to identify automatically materials on the Earth's surface and to determine their conditions. This is referred to as the discrimination capability of remote sensing.

As used here, discrimination means the successive classification of larger classes of materials into smaller, more finely divided subclasses, as in discrimination of conditions within a type or species after the type or species of the identified object is established. The main concern is the extent to which these successively finer classifications can be made automatically, if they can be made at all. The division into classes is based upon information sensed from a distance as opposed to in situ contact measurements. The economy and convenience of the information system will vary directly with the degree to which these classifications can be made automatically from remotely sensed data.

The rationale for automatic processing of multispectral scanner data is summarized as follows:

(1) Automatic processing can be done in quasi-real time, that is, before the information content of the data can significantly decay in value.

(2) Automatic processing of large volumes of data can be accomplished more cost-effectively (not necessarily more cheaply) than manual processing and interpretation but requires non-general-purpose computers for operational systems.

(3) Although information of the desired kind may be scanty, the data volume is exceedingly high. Automating the reduction of this data volume to information frees people for other creative tasks.

(4) Automatic processing offers a potential for a greater consistency of results with objective classification standards. Here, some would also mention its higher accuracy as compared to manual processing.

(5) Derived information from automatic interpretation is in a form for quick and easy integration with other data bases in automatic information systems such as for automatic mapping and compilation of statistical records or summaries.

(6) Multispectral scanner data are generated and recorded in electronic form intended for automatic processing, in contrast to photography in which film is the recording medium (which has poor radiometric fidelity compared to that needed in automatic processing).

Basic to this process of classification or discrimination, is the concept of a signature. In general, a signature is any collection of observable features of a material or its condition that can be used for precise classification. The features that make up a signature may all be observed simultaneously or in a sequence of observations spread over a considerable time period.

Variations in four characteristics of electromagnetic radiation can be used to effect discrimination between signatures. They are: (1) spectral variations (i.e., variations in radiant power as a function of wavelength); (2) spatial variations; (3) polarization variations; and (4) time variations, which can be of two types. The first type of time variation consists of changes rapid enough to cause a Doppler shift in reflected radiation. The second type consists of slower changes such as diurnal and seasonal changes. Each of these four variations in radiant power may be employed separately in discrimination, even though they interact with each other.

However, in the research reported here, the emphasis is on spectral discrimination, because some powerful techniques have been developed which exploit the spectral variations.

A basic element of spectral discrimination theory is the realization that spectral signatures cannot be completely deterministic. That is, spectral reflectivity and emissivity measurements of natural objects exhibit some dispersion around a mean value (i.e., spectral signatures are statistical in character). This should be expected, since it is well known that taxonomy based on any characteristics shows dispersion. Thus, as we will use the term, a spectral signature is a probability density function (or set of such functions) which characterize the statistical attributes of a finite set of observations of a material and can be used to classify the material or its condition to some degree of fineness.

In this report, we are specifically concerned with nonphotographic imaging sensors which operate in the ultraviolet, visible, and infrared regions of the spectrum, i.e., multispectral scanners.

MIDAS, which stands for Multivariate Interactive Digital Analysis System, represents a breakthrough in the field of multispectral scanner image analysis by providing a low-cost regional center capability for user-oriented, interactive, near-real-time, digital analysis to produce thematic mapping with instantaneous or multitemporal data. The system is shown in Figures 1, 2 and 3. MIDAS is 10 to 40 times cheaper, and three to five times faster than ILLIAC IV, and is reliable in operation. MIDAS accepts data from multispectral scanners in the form of high-density digital tape, computer-compatible tape, or analog tape, and makes use of proven multispectral processing techniques (including signature extension) within an innovative hardware approach resulting in a cost-effective, user-controlled system for multispectral analysis and recognition. Its hardware and software are intended to require a minimum amount of instructional training for successful operation. MIDAS is intended to provide accurate multispectral analysis for applications in disciplines such as agriculture, regional planning, forestry, energy and mineral resource location, pollution detection, water resources management, and others. Features may be extracted on the basis that their radiation properties are spectral, spatial, temporal, and (possibly) polarization dependent, thus giving a very general and powerful capability.

The first MIDAS was built as part of the NASA Advanced Applications Flight Experiment program by ERIM to demonstrate the unique advantages of a modular, special-purpose multispectral processor which offers a wide selection of high-performance subsystems, peripherals, and features. In this machine the parallel digital implementation capabilities of a low-cost processor are combined with a midi-computer to achieve near-real-time operation of a complete processing system that includes multiple, user-selectable, preprocessing functions and color displays. If provided in high density form, a LANDSAT frame of 7.2×10^6 pixels (each consisting of four values of 8-bits) could be classified in about 40 seconds on MIDAS due to its 200,000 pixel/sec peak classification rate. MIDAS, however, is not limited to a frame at a

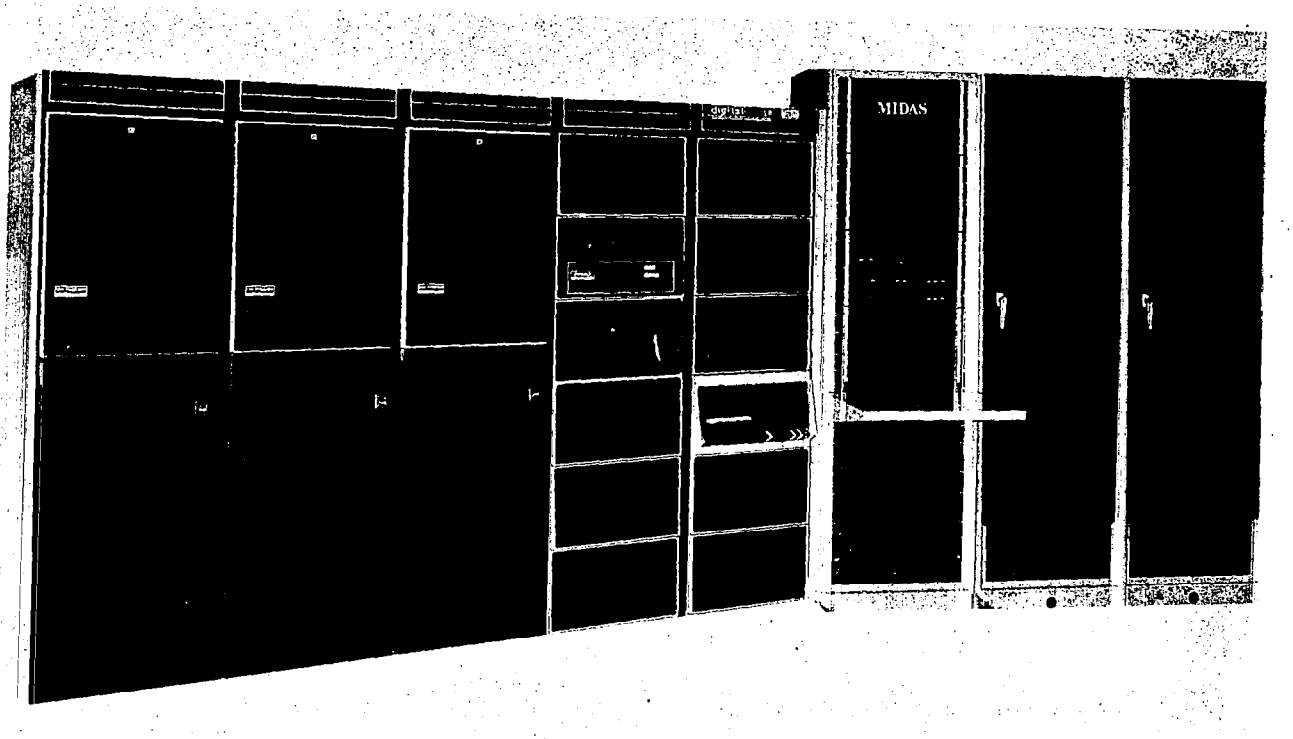


FIGURE 1. THE MIDAS MIDI-COMPUTER AND SPECIAL-PURPOSE HARDWARE CONFIGURATION

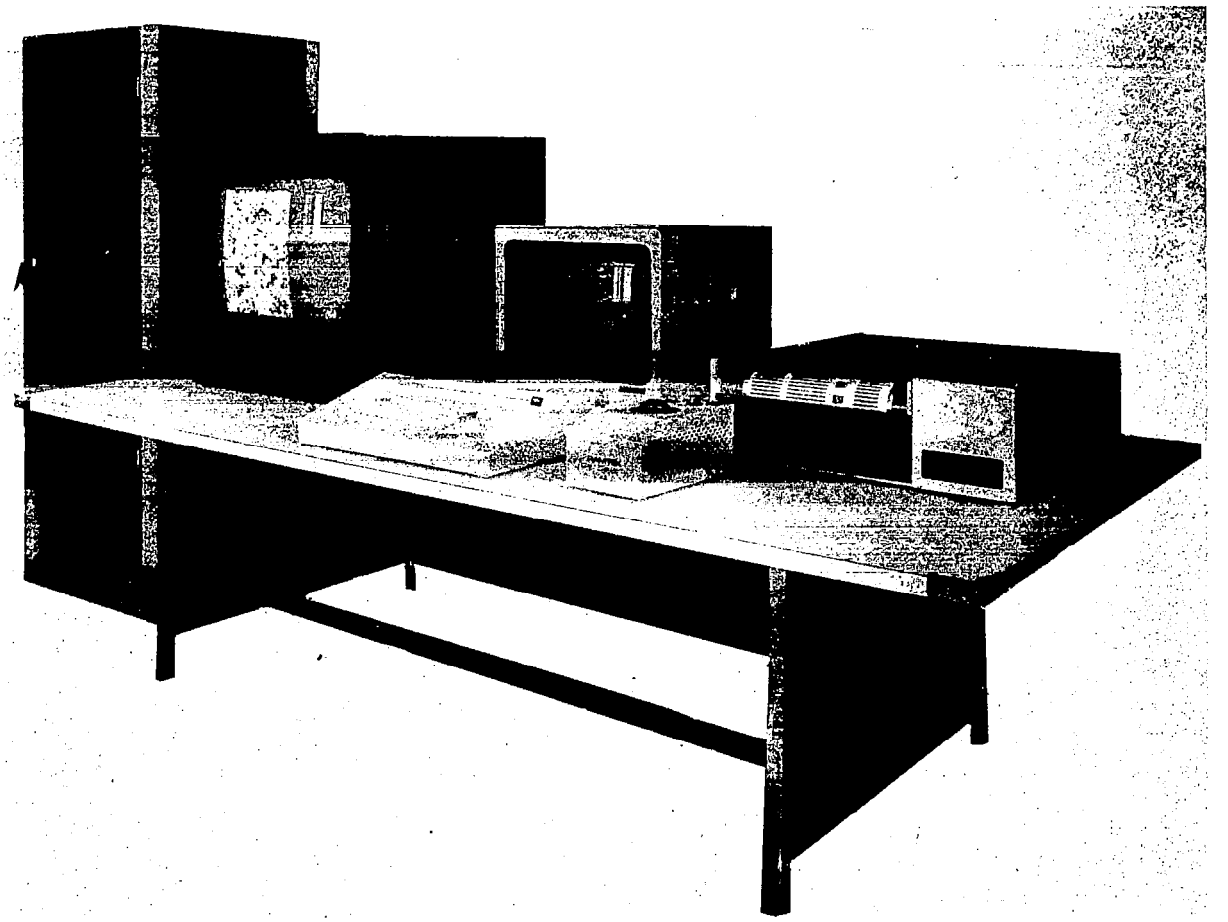


FIGURE 2. CRT MONITORS, INPUT KEYBOARD, TRACKBALL, AND INKJET PRINTER

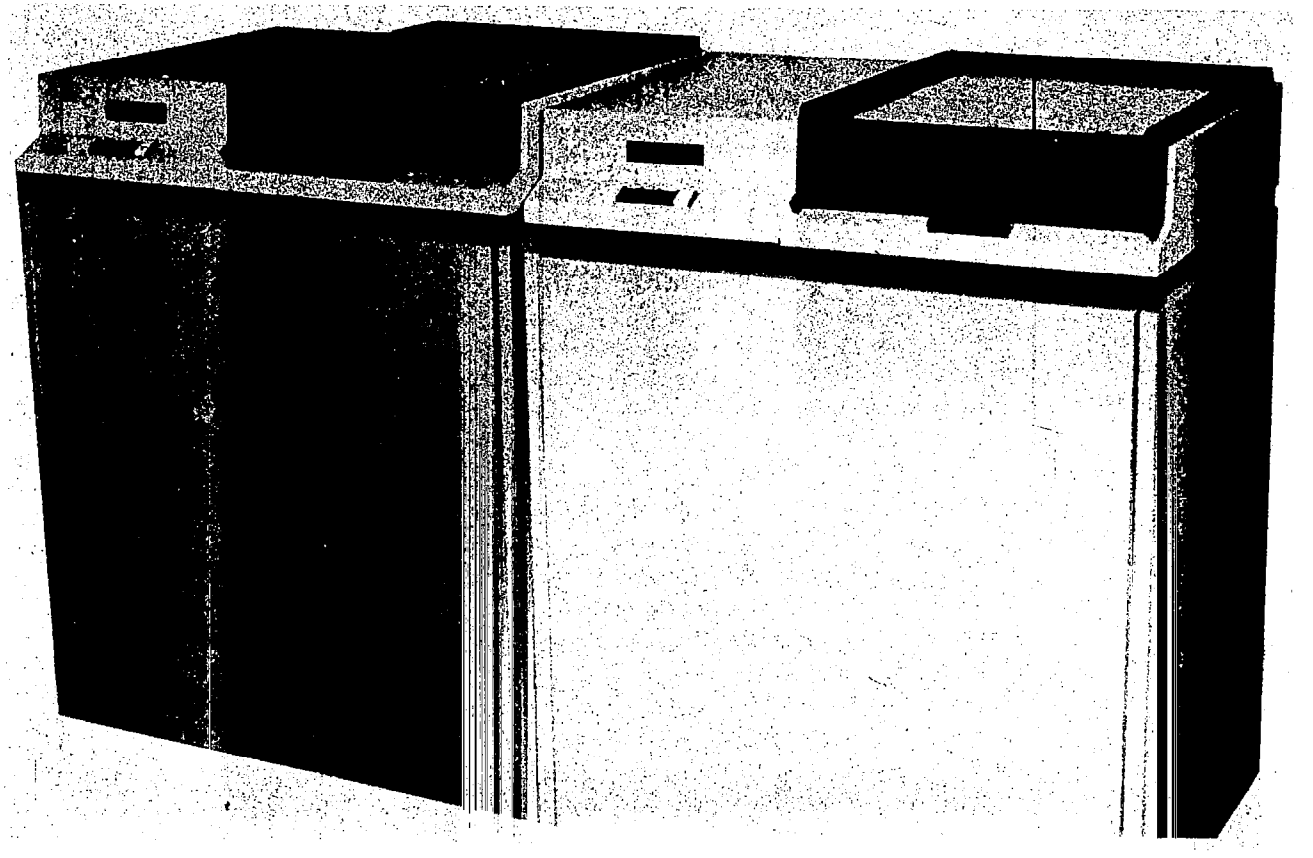


FIGURE 3. THE TWO DIVA 29-MEGABYTE STORAGE DISKS

time nor to four spectral bands at a time. Initial use demonstrates the effectiveness of this innovative approach in contrast with that of using the large general-purpose computers presently employed in the Earth Observations Program.

A rationale for MIDAS can be simply stated. A particularly important and easily overlooked aspect of applying a remote-sensing, multispectral system to aid in mapping crops, detecting pollution, or locating some ecological disturbance is that the data can be processed to provide the proper information to a user in a time short enough to meet his needs. Unfortunately, some ongoing programs do not address this aspect of the system design problem, for reasons which are rarely clear.

The magnitude of the discrepancy between the ability of a sensor to gather the data and the ability of a general-purpose computer to process it becomes the next aspect of the problem to be assessed. This enormous mismatch can probably be best appreciated by considering a brief numerical example. An airborne scanner will, typically, gather data over a 32 to 48 km flight line in about 15 minutes and record it on one reel of magnetic tape. A general-purpose digital computer can be trained and can classify this data in a time about 100 times longer than it took to collect it. Thus, the data collected in 15 minutes will require 1500 minutes of processing. This would amount to three eight-hour days of processing. Given one such computer to process the data, the aircraft should only be used for six five-hour sorties per year. For satellite systems the mismatch is worse. Clearly, the discrepancy in capabilities is unacceptable.

ERIM geometrically corrects LANDSAT data by (1) rotating the data so that N-S roads are parallel to the edges of output displays, (2) deskewing the data to correct for Earth's rotation effects, and (3) matching the aspect ratio of the output display and scaling the output image. ERIM has also received recognition for the development of computer-assisted procedures for correlating LANDSAT and geographical reference coordinate systems.

Another important objective of fast, low-cost but accurate processing is to facilitate control by the user. Classification of remotely sensed data is an interactive process in which the person and machine must, in fact, be considered as the real processing system. It thus becomes evident that well designed, interactive display and control subsystems will, in reality, offer the greatest gains in throughput. This also implies that the system should be considered as a prototype from which further refined systems may be derived for specific operational applications. To allow user tailoring of the operational system, modularity becomes an important design aspect.

The present system was conceived and constructed as a demonstration that a clearly better processing system will materially assist in practical, economic realization of the benefits of remote sensing.

1.3 BACKGROUND

The MIDAS program was to consist of two phases of development, each of a year's duration, in which the basic system was to be designed and manufactured and then, in the second phase, expanded and demonstrated. The objectives of the MIDAS program were to demonstrate

(1) that a high throughput rate can be achieved without loss of accuracy on MSS data from satellites and aircraft

(2) that large-area surveys can be classified quickly and at low cost

(3) a processing rate that was adequate to match the input rates from present or future sensors and high density digital tapes

(4) pictorial and enumerative results on data from a wide range of sources

(5) a realistic environment for reducing man-machine interaction without loss of accuracy

(6) that the regional processing center concept can be built around one relatively low-cost (of the order of $0.5 million compared to $2-3 million for large general-purpose computer system) like MIDAS and serve essentially all users with different applications

During the first phase the classifier, a pipeline machine able to accept sensor signals and produce classification results, was developed and tested along with a general-purpose controlling system based on the DEC PDP-11/45. In the second phase a pre-processor pipeline machine was added prior to the classifier and an interactive, display-based software system was developed to allow efficient operation and management of the various procedures by a user. The design approach is summarized in Table 1. As a result of these developments, MIDAS constitutes a prototype able to process data for large area surveys quickly and economically in a quasi-operational manner or, of equal importance, a device able to test the feasibility of newly proposed applications of remote sensing quickly and accurately. Summaries of the subsystem characteristics are given in Tables 2-5.

MIDAS, at this stage of development, is a prototype in other respects. Although the software system, MIDOS (MIDAS Operating System) has been designed to provide an efficient interactive control system for a user; it will require use in quasi-operational processing, and in testing the feasibility of new applications, in order to realize potential processing gains. Another principal use for a Midas system is in its function as a prototype for peripheral systems operating in conjunction with a large general-purpose system organized as a data-base and data-management center.

MIDAS can also be considered as a prototype or a test-bed system in that it mechanizes only those processing procedures now known to be not only useful, but also desirable to implement in hardware. The processing pipeline is modular, in that additional processing requirements can be met by inserting other arithmetic operators in the pipeline and by increasing or decreasing the parallelism as the need for such changes become apparent. Potential system additions indicated by current research and needs of various programs would include additional pre- and post-processor hardware for spatial array processing and multi-temporal analysis.

TABLE 1. DESIGN APPROACH USED IN THE MIDAS SYSTEM

Implement with:

- Commercial midi-computer and peripherals
- Commercial displays
- Special parallel-pipeline Preprocessor and Classifier
- Special high-density digital tape and analog tape input
- Special color hardcopy
- Special software system for control and interaction

TABLE 2. FUNCTIONAL CHARACTERISTICS OF MIDAS GENERAL-PURPOSE COMPUTER AND SOFTWARE

General-Purpose Computer:

- Control element, CCT, Data disk, Diagnostics

Software:

- Image manipulation and display
- Training data labeling
- Preprocessing analysis and application
- Signature manipulation
- Classification
- Post-classification analysis with test area verification of performance accuracy and N levels of statistical aggregation
- Diagnostics

TABLE 3. FUNCTIONAL CHARACTERISTICS OF MIDAS DISPLAY

Unlimited arbitrary shaped training and test sets
Digital zoom and move
32 colors on 512 × 512 CRT
B&W menu with trackball cursor and user prompting
Color level-slicing
Solid-state memory refresh
864 × 1314-point inkjet in 90 seconds on 21.6 × 27.9 cm paper for "walkaway" color images
Graphs
Histograms
Cluster plots

TABLE 4. FUNCTIONAL CHARACTERISTICS OF MIDAS PREPROCESSOR

User-selectable options on 16 channels
1-D and 2-D multiplicative and additive corrections
General matrix transform for linear combinations
Ratios of eight transform data variables
All at 2×10^5 pixels/sec rate
Diagnostic bus for intermediate results

TABLE 5. FUNCTIONAL CHARACTERISTICS OF MIDAS CLASSIFIER

Bayesian decision rule (maximum likelihood); easily changed
Multi-modal distributions (sum of Gaussians)
$(1 + \epsilon)$ pass classification into 16 + 1 classes or 8 + 1 classes
Strip processing, not frame limited
Rate is 2×10^5 pixels/sec or landsat frame in 40 seconds
Diagnostic bus

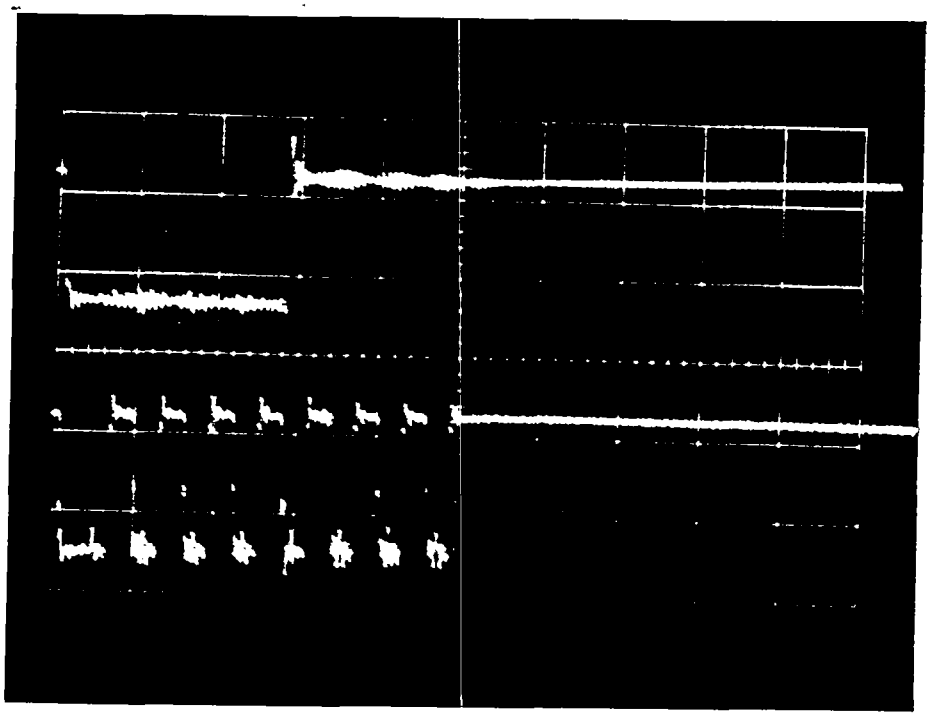
Lastly, MIDAS and arithmetic structures based on MIDAS, have been and can be considered as a prototype for sensor platform processing as a means of compressing telemetry data. This may take the form of pre-processing sensor data into a smaller amount of almost equivalent data by linear and non-linear transformations or by processing the data into classified results encoded for transmission. Such applications may be tested or developed on MIDAS as a means of proving such techniques or as a means of specifying the operations which may be performed effectively in a spacecraft. Such investigations are already being done.

1.4 AN OVERVIEW OF THE SYSTEM

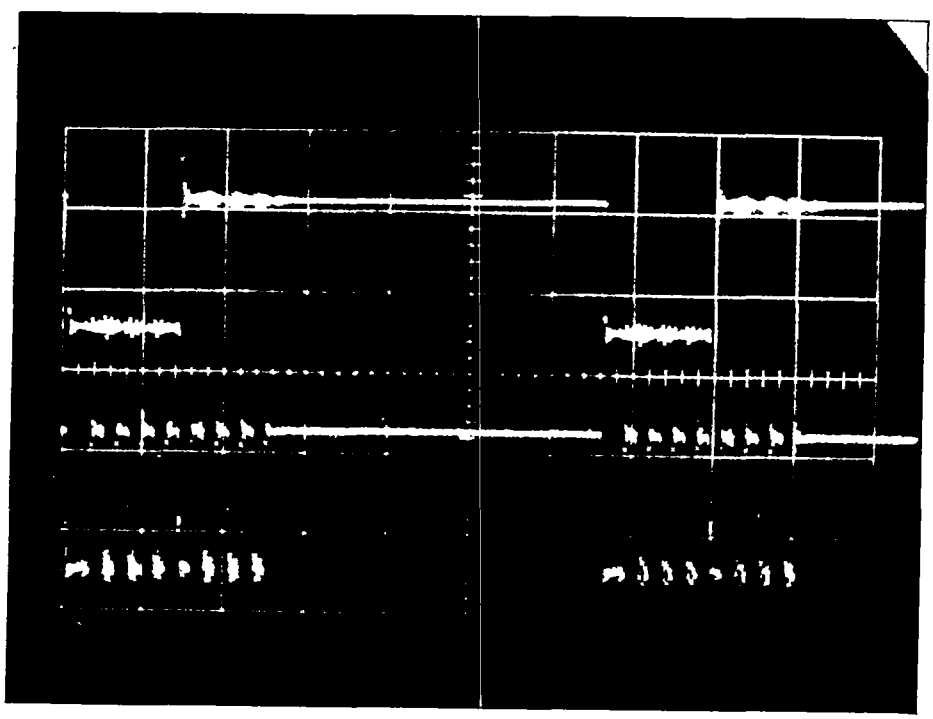
Some of the demonstrated characteristics of MIDAS are illustrated in the following discussion. The speed of the pipeline processor is shown by the test oscilloscope traces in Figure 4. Two pulse sequences from the master pipeline clock are shown in the dual trace pictures. The master clock is obtained from a crystal oscillator through a chain of count down circuits. The clocking rate of the pipe may be selected by computer control during the initial setup of classifier parameters. The clock is described in Section 2.5.1 and shown in Figure 30. The fastest rate is illustrated in the traces of Figure 4. Figure 4(a) shows the pulses on a horizontal time scale of 1 μsec/cm. The upper trace pulse has a period of 5 μsec (the basic pipeline rate of 200,000 pixels per second) while the lower trace contains 16 axis crossings in the same 5-μsec period. This is the time required to perform the function at any station in the pipe. Figure 4(b) shows the same two-pulse trains on a horizontal time scale of 2 μsecs/cm and contains two processing steps. The clock is restarted (or bursted) each time a pixel is sent from the data source. In this case the data is being sent at the rate of 1 pixel each 13 μsecs. In this illustrated case the pipeline processor is waiting 8 μsecs for data and then processing the data in 5 μsecs.

The interactive process of using MIDAS is illustrated in Figures 5 through 11. Figure 5 shows the master menu displayed on the small black and white CRT (see Figure 2) at the start of a data processing session. Assume that the desired data set has been loaded onto one of the 29-megabyte high-speed discs shown in Figure 3. The cursor (the cross near the middle of Figure 5) is moved by means of the track ball and is used to select "Data Display" by superimposing the cursor upon the box to the left of "Data Display" on the menu. The data display overlay is then read into the computer and the menu shown in Figure 6 is displayed. In this figure are also shown the required display parameters fed into the system by the user via the display keyboard (see Figure 2). An automatic-level-setting program level-slices the data into categories of grayscale. Figure 7 shows the resulting display of the raw data set on the large color CRT.

The next step returns the user to the master menu (Figure 5) at which time "Field Manipulation" is selected. This program overlay is brought into the computer, and the capability to outline training fields on the raw data image is activated. The fields are designated by moving the cursor to the vertices of an irregular figure (called a polygon) and depressing the



(a) 1 pixel, followed by dead time
(horizontal time base: 1 μsec/cm)



(b) Interval of 13 μsec between pixels caused by limitations of input device
(horizontal time base: 2 μsec/cm)

FIGURE 4. OSCILLOSCOPE TRACES ILLUSTRATING PROCESSING RATE OF 5 MICROSECONDS PER PIXEL

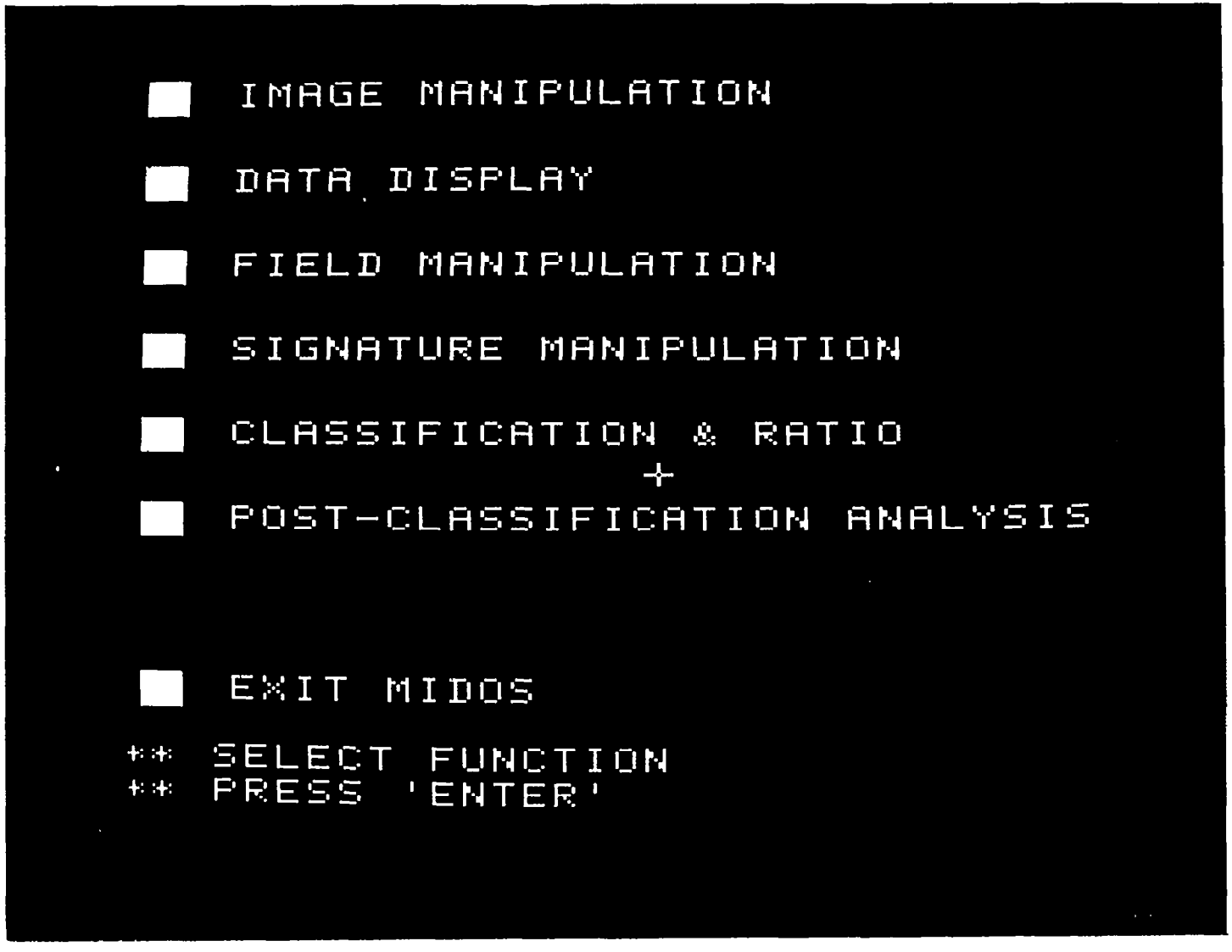


FIGURE 5. MASTER MENU DISPLAYED UPON ENTRY TO MIDOS

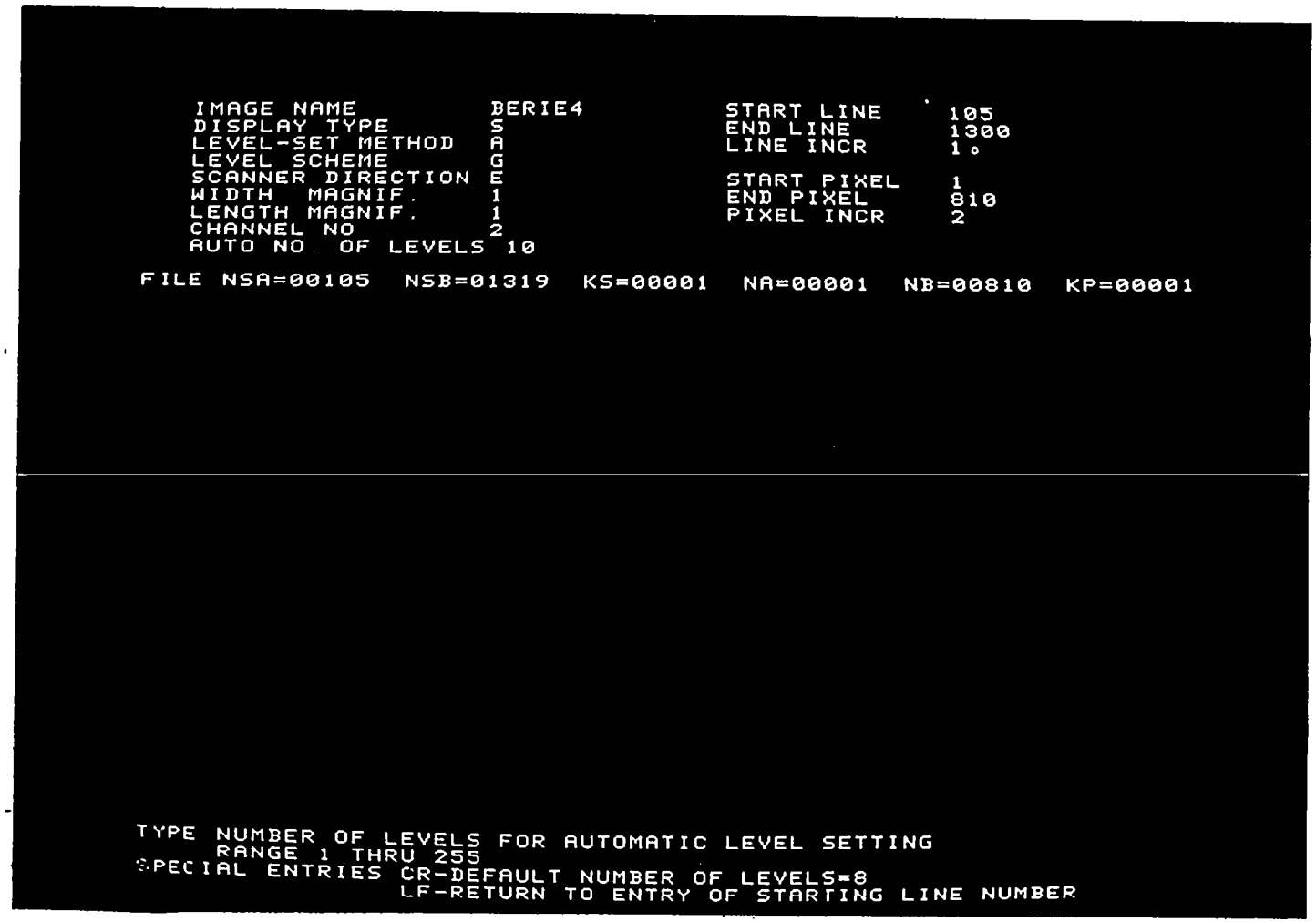


FIGURE 6. DATA DISPLAY MENU WITH PARAMETERS

FIGURE 7. RAW DATA DISPLAY ON COLOR CRT

enter button on the track ball cursor control. The figure is closed by pushing the enter button twice at the same vertex. Figure 8 shows a polygon outlining the large lake near the lower right edge of the figure. All desired training data would be designated sequentially in this manner.

The next step is to return again to the master menu and select "Signature Manipulation." This overlay calculates the statistics which describe the data contained in the training data sets outlined as described above. Figure 9 shows the results which are displayed on the black and white CRT as the statistics for each signature are determined.

Following this, the master menu is again used to choose "Classification and Ratio" as the next overlay to be called into the computer. This program accesses the disc file in which the signatures have been stored. The desired signatures are chosen from the display, and the MIDAS special purpose pipeline hardware is loaded with the appropriate coefficients from the statistics associated with each signature selected.

When classification is complete (in the order of less than a minute, depending on the size of the data set) control is once again returned to the master menu. "Data Display" is again chosen, and in a manner similar to setting up the raw data display, a classification map is shown on the large color CRT (Figure 10). If a hard copy of this classification map is desired the inkjet plotter may be used to produce a color image on paper. Such a hard copy map of the raw data is shown in Figure 11.

1.5 CONCLUSIONS AND RECOMMENDATIONS

Speed. The MIDAS system could be improved, possibly by a factor of 3, by the use of higher speed MSI and LSI technology. However, the present speed is adequate for classifying multispectral data since the principal bottleneck in such operations is the human user. Classification of a LANDSAT frame, for example, can be done in the MIDAS pipeline in about 40 seconds, but about an hour may be required to go through the machine-aided but human-judgment-dependent analysis and training operations. Little improvement in throughput can be gained from increased hardware speed when the user is the limiting element.

Review of the software programs and procedures in actual use of the system would best serve to streamline human operations which in turn would produce significant increases in speed, possibly by a factor of four. This is planned in the process of using the system.

Spaceborne Processing. The pre-processor and classifier pipelines of MIDAS can be put to effective use in on-board spacecraft processing as a means of bandwidth compression and/or pre-transmission analysis. Compression by a factor of two or three may be had from the pre-processing operation. Compression by factors of four to five may be had from the classification process, in that a 4- or 5-band, 8-bit pixel is compressed to a 4- or 5-bit, single code word. An overall data compression between 10 and 20 may be obtained.

FIGURE 8. RAW DATA SHOWING POLYGON ENCLOSING A TRAINING SET

STATISTICAL DESCRIPTION

CHANNEL	MIN	MAX	MEAN	STANDARD DEVIATION	COEFFICIENT OF VARIATION	NUMBER OF REJECTS
1	18	22	.1939E 002	.1000E 001	.5158E-001	19
2	8	13	.9974E 001	.1000E 001	.1002E 000	1
3	3	7	.4759E 001	.1000E 001	.2101E 000	4
4	0	3	.6270E 000	.1000E 001	.1595E 001	0

(a)

COVARIANCE MATRIX

CHANNEL	CHANNEL 1	2	3	4
1	.10E 001	.19E 000	.10E 000	-.44E-001
2	.19E 000	.10E 001	.39E-001	.10E 000
3	.10E 000	.39E-001	.10E 001	-.22E 000
4	-.44E-001	.10E 000	-.22E 000	.10E 001

(b)

CORRELATION MATRIX

CHANNEL	CHANNEL 1	2	3	4
1	.10E 001	.19E 000	.10E 000	-.44E-001
2	.19E 000	.10E 001	.39E-001	.10E 000
3	.10E 000	.39E-001	.10E 001	-.22E 000
4	-.44E-001	.10E 000	-.22E 000	.10E 001

(c)

EIGENVECTORS (IN COLUMNS)

	CHANNEL 1	2	3	4
	.52E 000	-.66E 000	.30E 000	.43E 000
	.27E 000	.41E 000	-.53E 000	.68E 000
	.63E 000	.56E 000	.46E 000	-.23E 000
	-.48E 000	.25E 000	.63E 000	.54E 000

(d)

EIGENVALUES

CHANNEL	
1	.12E 001
2	.80E 000
3	.72E 000
4	.11E 001

(e)

FIGURE 9. SIGNATURE STATISTICS

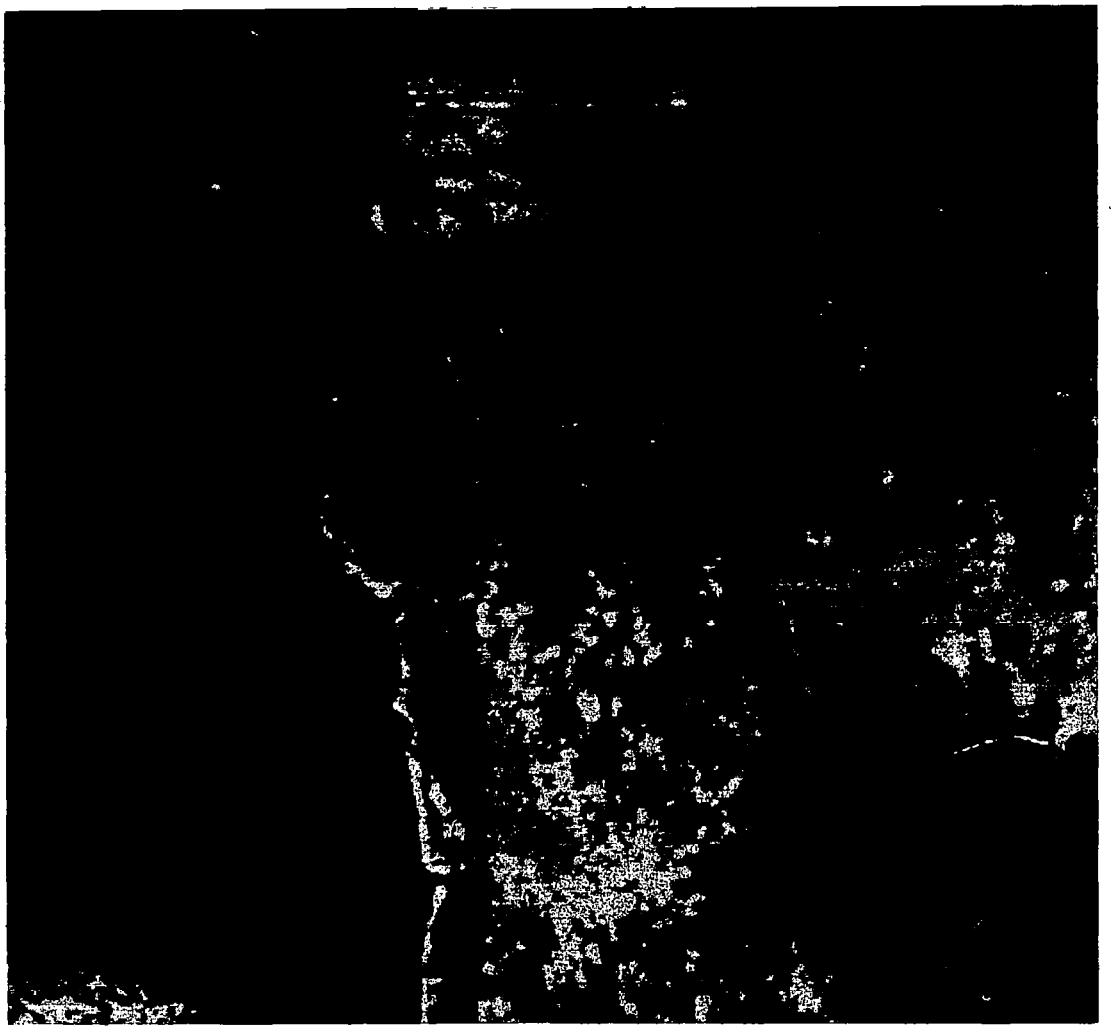


FIGURE 10. SAMPLE CLASSIFICATION MAP FROM COLOR CRT

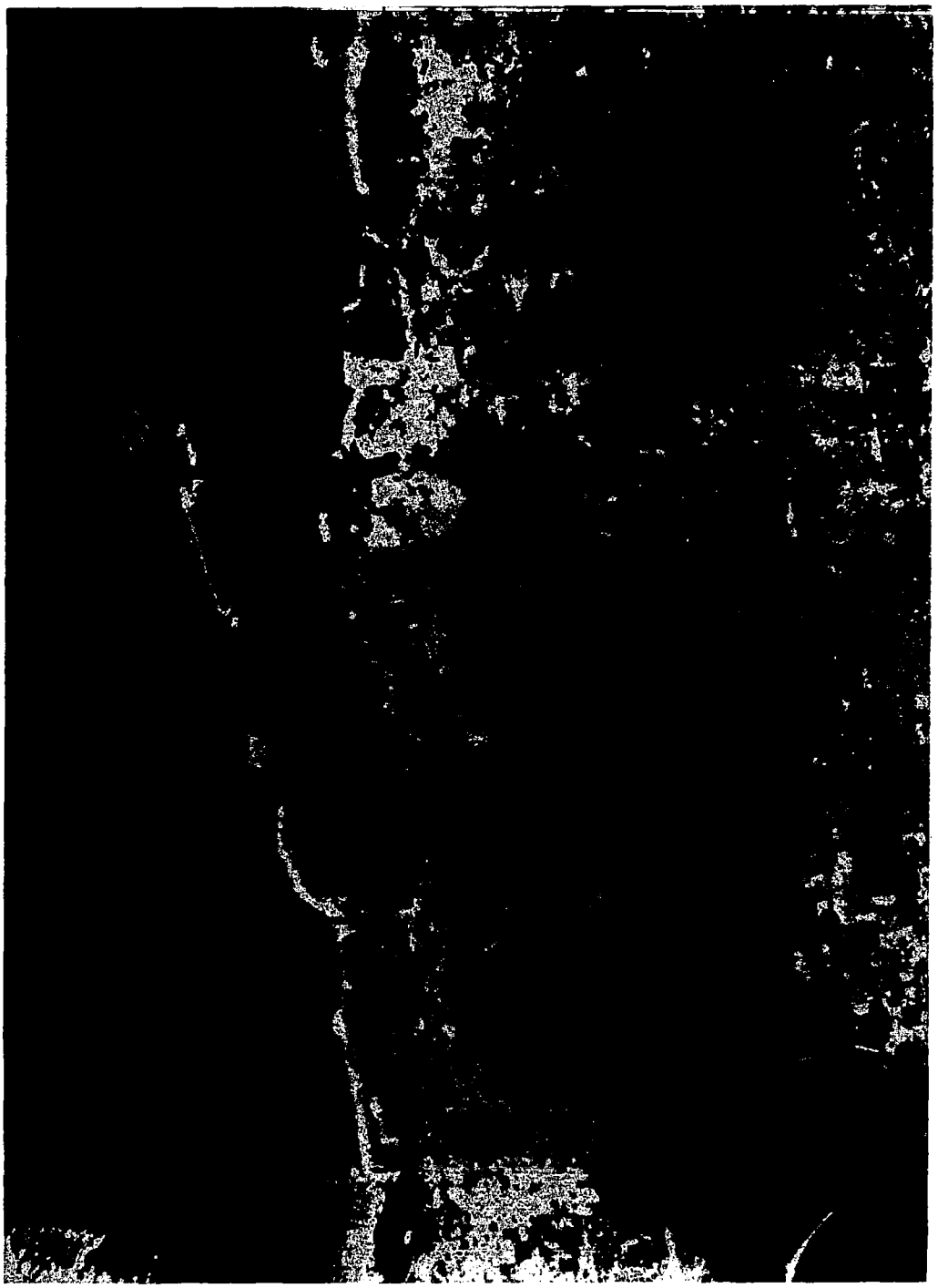


FIGURE 11. SAMPLE CLASSIFICATION MAP FROM INKJET PRINTER

A significant problem exists in such usage, however, and this is the need for in situ or a a posteriori processing. Interaction by a user to specify signatures and areas of interest is required, thus making the compression gain, dependent on the overall system configuration including the user.

Bulk analysis of a frame prior to transmission seems feasible. In this manner a review of a frame for cloud cover, for example, could be done on-line and the system could be set to respond with a suitable code to convey only this information, or using a run length code procedure, to transmit only non-cloud data, if such were desired.

Rates of 100×10^6 bits/second and higher are needed for some sensor systems. Depending on the number of bands and the element rate, an advanced technology MIDAS could process such data. MIDAS presently processes 25×10^6 bits/second (16 channels $\times$ 8 bits $\times$ 2×10^5). Using the same organization and faster components, data at a rate of 100×10^6 bits/sec and could be processed.

A Prototype. MIDAS is a prototype. The system is aimed at use in an average regional processing center, sized to meet the throughput needs for an area of 1/5 to 1/10 the continental U.S.A. Its use in smaller or larger systems may be anticipated, yielding systems with more arithmetic modules or less depending on the station size and regional demand. A great portion of effort needed yet is to match the machine to the user. The present software system is a first-pass design in what must be considered an iterative design process. Also, some portions of the analysis related to optimum linear and ratio transformations, for example, are not well enough understood at present to fix the shape of analytic programs, although the hardware now exists. There is a need then, for study of an iterative human-factor process and also of the analytic nature of data transformation for subsequent classification.

Finally, with the recent advent of micro-processors able to operate in the speed range below 100 nanoseconds, it seems likely that the MIDAS pipelines could be fabricated in the future using this technology to obtain smaller size and greater economy, using, basically, networks of microprocessors. MIDAS, however, will provide valuable information to obtain a refined system architecture for both hardware and software improvements for a period of several years, thus functioning as a true and useful prototype system.

Advanced Processing. Recent developments indicate the need for greater accuracy in classification and in geometric correction; which creates the need for overlaying imaged data from various times of the year and from various sensors, and providing image output to some specified geometric accuracy. Given data which is overlaid, MIDAS can classify or enhance such data with no modifications, assuming the spectral dimensionality of the data is not greater than 16 (as would occur for four LANDSAT images, for example). Also, enlargement of MIDAS capacity to greater than 16 dimensions is not a major change, if needed.

However, the integration of geometric and spectral image processing is very necessary and will require expansion of a MIDAS-like system. This can be done in a gradual, modular manner using MIDAS as a base system. It thus appears that evolutionary development of such forms of processing can be facilitated with MIDAS affording a test-bed facility well matched to the developments anticipated, as the use of remote sensing becomes of greater importance and practicality.

MIDAS HARDWARE

MIDAS hardware is specially constructed to fit the needs of classifying multispectral, remotely-sensed data. The core of this classification task is the specially-designed classification pipeline (see Figure 12). Several key points along the pipeline communicate with the DEC PDP-11 Unibus, a single path that connects most of the other pieces of equipment in the MIDAS system in parallel. These other pieces of equipment, which include a DEC PDP-11/45 CPU, core memory, and numerous I/Odevices, support operation of the classification pipeline by preparing its input, controlling its operation, displaying its output, and providing an easy-to-use interface with the human operator. The remaining MIDAS hardware—the direct sources of input to the classification pipeline—includes high density digital tape and analog devices.

A more detailed description of the classification pipeline section of the MIDAS system is shown in Figure 13. The figure shows the three specially-designed processors that comprise the pipeline: the Data Path Selector, the Preprocessor, and the Classifier. The additional hardware in the figure includes a high density tape unit (HDT), an analog-to-digital (A/D) unit, a digital-to-analog (D/A) unit, and DEC DR11-C and DR11-B standard Unibus interfaces. In addition to enabling operation of the classification pipeline, this hardware provides a mechanism for transferring multispectral data among several different storage devices.

The pipeline operation is outlined below, followed by descriptions of the Data Path Selector, the Classifier and the Preprocessor.

2.1 CLASSIFICATION PIPELINE

The classification pipeline, shown by the wide arrowed lines in Figure 13, is the core of the high-speed classification process. The pipeline physically consists of a one-way data flow through the three special high-speed digital processors: the Data Path Selector, the Preprocessor, and the Classifier. The Data Path Selector, a data-routing piece of hardware, performs the first step in the pipeline; it supplies the multispectral data to the remainder of the pipeline. The data supplied consists of picture elements or "pixels" where each pixel can be considered a vector of up to sixteen 8-bit data bytes or channels. The data comes from one of three sources: the Unibus via the general purpose Direct Memory Access (DMA) interface, a DEC DR11-B; the high density tape unit; or an analog tape unit. These three alternate inputs are shown by the medium wide lines in Figure 13. The data input selected proceeds through the Data Path Selector to the Preprocessor where processes such as scaling, angle correction, linear combinations, and calculations of ratios prepare the data for the key step, classification.

The actual classification of the data into categories is performed by the Classifier. Within the Classifier, the single pipeline temporarily divides into four parallel pipelines to perform fast simultaneous matrix multiplications. These multiplications are processed further, and the results are fed sequentially into a decision process wherein each former pixel is classified into one of up to 16 pre-determined categories or into a seventeenth class, meaning "none of

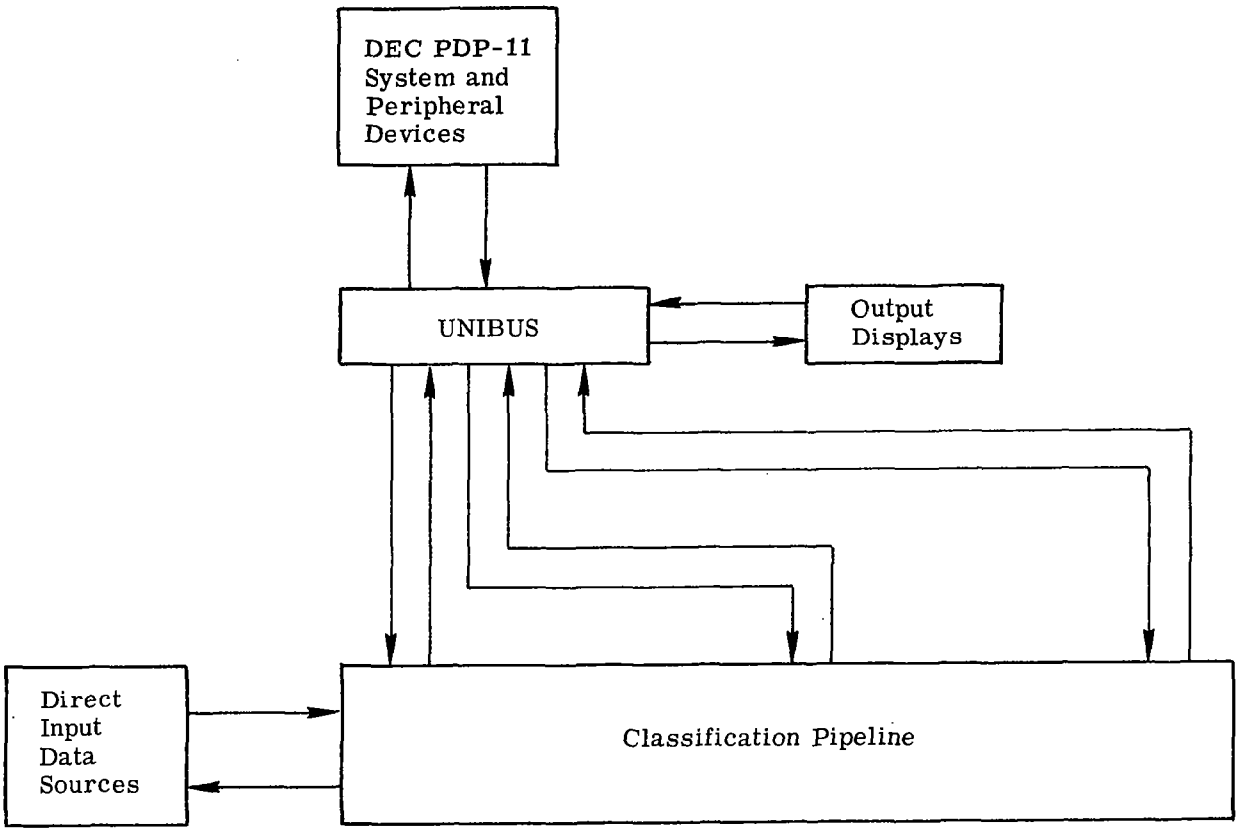


FIGURE 12. THE MIDAS SYSTEM HARDWARE

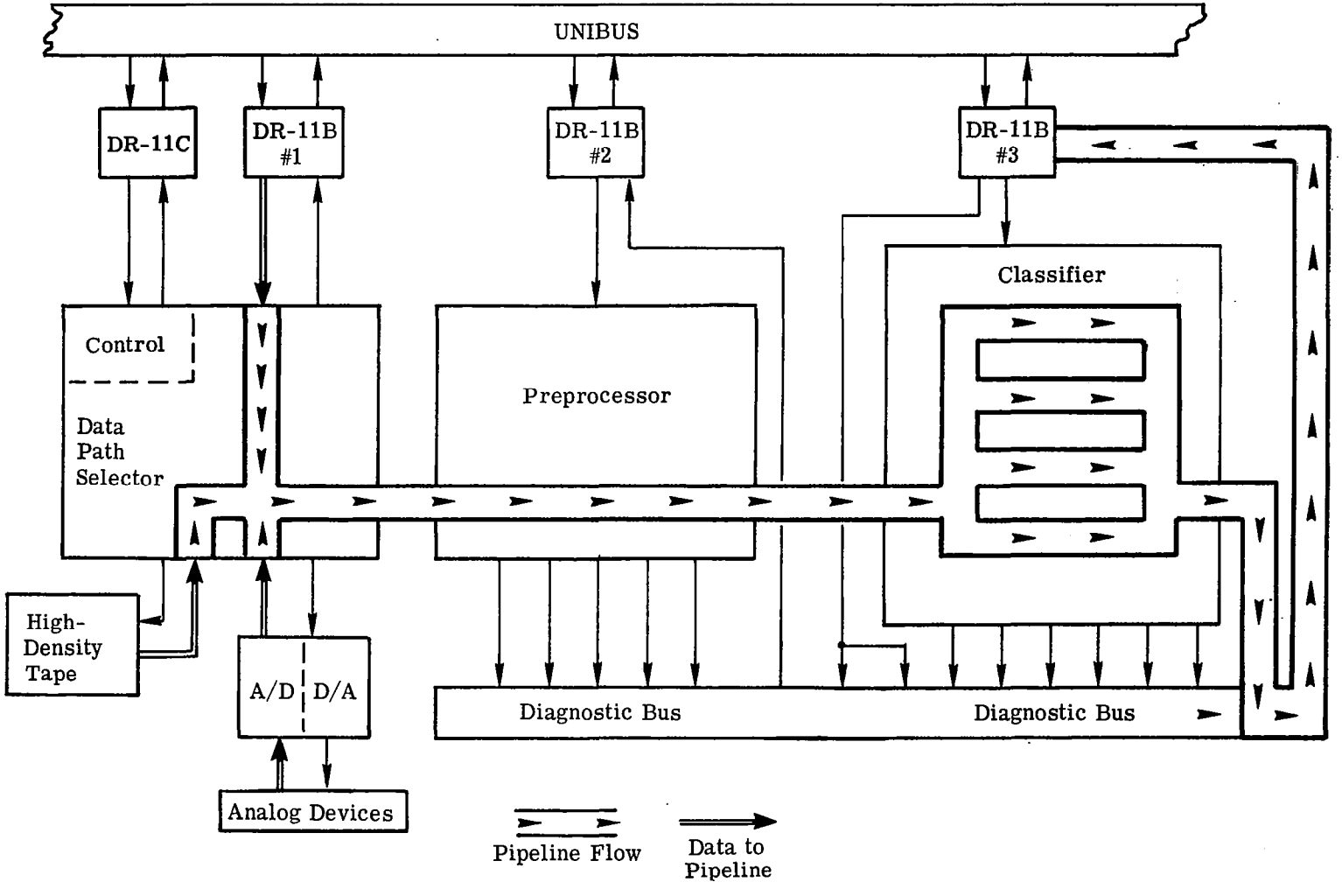


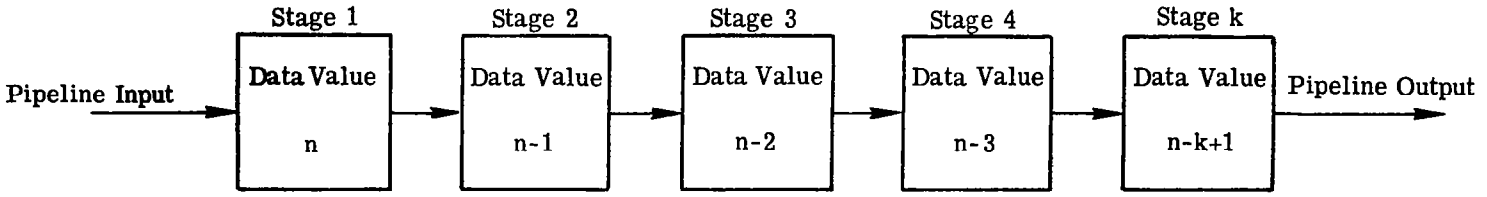
FIGURE 13. THE CLASSIFICATION PIPELINE

these." For each pixel that entered the pipeline at the Data Path Selector, only five bits, a category code, emerge from the Classifier. These five bits travel, by way of the Diagnostic Bus and the DR11-B#3, to the Unibus, along the Unibus to core memory, and from core memory to their final destination, the Diva disk. The Diva disk holds the data which may later be displayed by any of the number of output devices, including the RAMTEK Color Display and the Color Inkjet Plotter.

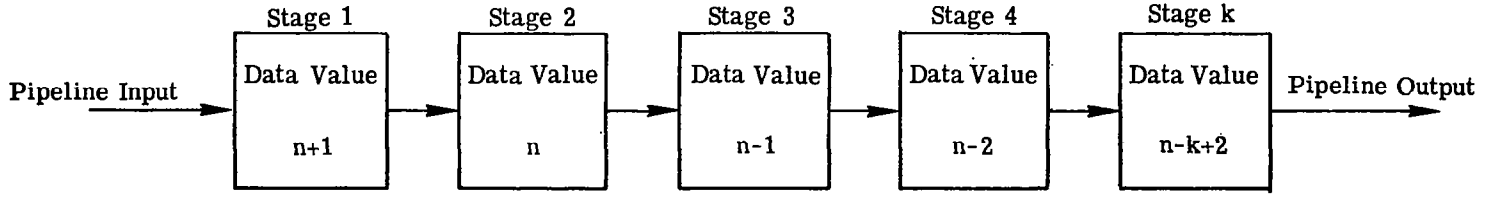
The pipeline can classify up to 200,000 pixels in this manner every second. The very fact that the structure is a parallel pipeline accounts for this high speed. The term parallel refers to the four parallel branches of the pipeline within the Classifier. This parallel structure makes possible the high speed matrix-multiplication operation. The term "pipeline" refers to the assembly line form of operation for the whole process. In general, a pipeline operates as follows: the many functions performed along the pipeline are divided into k sequential time steps, all of equal time duration. This division is illustrated for the general case in Figure 14. At any one instant,k separate stages are operating on k separate data values (Figure 14a). One time-step later (Figure 14b), each stage supplies its output to the next stage for further processing, and a new data value enters the first stage. This is accomplished by having a register between each operational stage into which the output is strobed; it is done simultaneously at all stages, thereby locking or latching up the results of each stage. These "latched" data then serve as input to the next stage, while another data value enters the first stage. A pixel is defined as always having 16 data values, therefore requiring 16 strobes or clock pulses to enter a pixel into the pipeline; if fewer than 16 data channels are available the hardware in the Data Path Selector provides the necessary filler data. The need for 16 clock cycles to enter a pixel is dictated by the fundamental operation of the Classifier and its clocking operation. The detailed description of the Classifier (Section 2.3) will explain this point.

The master clock for the MIDAS pipeline resides within the Classifier. The clock controls the stepping of data through the pipeline and the addressing of coefficients in the random access memories (RAM's). Capable of operating in four fundamental modes with seven possible internally controlled frequencies, the clock also allows external control of the frequency. It operates asynchronously, its sequencing initiated for each entering pixel by the external data source.

Operation of the pipeline as described above requires two additional steps: program-controlled setup before the process;and user-controlled diagnostics before, after and during the process. Setup in the pipeline consists of loading several RAM's in the Preprocessor and the Classifier with computer-generated numbers, numbers which include information about the data set and which specify the criteria for classifying the pixels into the particular categories. Diagnostics consist of readback of data from registers in the Preprocessor and the Classifier at several key points in the pipeline, yielding either results of mid-process calculations or clues for debugging or trouble-shooting. This information, accessed via two multi-port diagnostic



(a) Pipeline at One Instant in Time



(b) Pipeline One Time-Step Later

FIGURE 14. A PIPELINED OPERATION

buses, is available both to the CPU and directly to read-out devices. An example of usage of the diagnostics subsystem is the optional routing of ratios of channels, calculated in the Pre-processor, to the RAMTEK Color Display.

2.2 DATA PATH SELECTOR

The special-purpose hardware selector though designed specifically for the classification process, is capable of accomplishing some additional functions. In addition to playing a role in the pipeline, the selector can, by setting up the appropriate paths, transfer multispectral data among several different storage devices.

The Data Path Selector, which receives its instructions from the CPU via the DR11-C interface, can set up various alternate data paths, including:

(1) Unibus to Pipeline (via the DR11-B #1)
(2) HDT to Pipeline
(3) A/D to Pipeline
(4) Unibus to HDT
(5) Unibus to D/A
(6) HDT to Unibus
(7) A/D to Unibus

The first three alternate pathways, described in the preceding paragraph, enable operation of the pipeline with input coming from either the Unibus, the high density tape, or an analog tape. The last four pathways enable transfer of data from either the Unibus, a high density tape, or an analog tape to any one of these same storage devices. In these last four cases, the data that comes from, or is routed to, the Unibus usually is data from what is commonly called a "computer-compatible tape" (CCT), a standard-format tape compatible with DEC's TU10 magnetic tape drive as well as with IBM drives.

The control section of the Data Path Selector is responsible for receiving, interpreting, and sometimes routing most of the computer-generated control instructions. These instructions specify

(1) data paths, as described in the last paragraph
(2) parameters for selective gating of data from any of the input sources
(3) channel and frequency information for the A/D
(4) frequency and mode information for the Classifier clock
(5) other control details

The instructions are sent by the computer to the Control section via the Unibus and the DR11-C interface (the data path can be traced in Figure 13).

Since all the sources of data operate asynchronously with the MIDAS master clock, the data path selector must accept data from the data source using the source's clock and transfer

it as input to the pipeline using the MIDAS clock. This is accomplished by the dual register approach shown in Figure 15. the first rank of registers is loaded under clock control of the input device. When this loading is complete (pixel loaded) and the pipeline is ready to accept it, all data values are simultaneously loaded into the second rank registers. The outputs of these second rank registers are bussed together. The outputs are selected by feeding the 16-phase MIDAS clock to a 4 × 16 RAM, the output of which controls the selection. Since this RAM is loaded under computer control, any sequence of 16 registers (data channels) can serve as the "pixel" input to the preprocessor.

2.2.1 HIGH DENSITY TAPE SUBSYSTEM

The HDT system can accommodate up to eight channels of information. This data is recorded in digital form on separate tracks of a wide-band instrumentation tape recorder in a bi-phase code.

On playback, the outputs from the different channels of the tape recorder are fed to equalizers, one equalizer card for each tape track, to improve the signal-to-noise ratio of the P.C.M. signal. From the equalizers, the signals are passed to bit-synchronizers where the NRZ digital data and clock information are obtained. This digital information for each track is then routed to sync detect-lock circuitry. In the detect portion of this circuit, the data is collected into 16-bit words and is integrated for a valid 16-bit sync word. In the sync lock section, a lock condition is established after one or more sync words have been located. This circuitry also senses when the sync word is missing and defines an "unlock" condition which reverts the system back to the sync detect mode.

In the following sections each of the cards that make up the data handling portion of the HDT playback system is described.

2.2.1.1 Equalizer Card

There are eight equalizer cards in the HDT system, one for each of the data tracks of the tape recorder. In the playback mode, the system can operate in any one of four tape recorder speeds. Each of the cards contain four equalizer-filters which correspond to the different tape speeds. When the desired tape speed is established by the computer, the appropriate equalizer-filter is switched into the data path.

The system is conditioned in one of the four tape speeds from the computer using one of the system conditioning commands established through the DR11-C.

2.2.1.2 Bit-Synchronizer Card—Playback Circuitry

There are eight bit-synchronizer cards, one for each of the tape channels, which contain circuitry that is involved in both the record and playback modes.

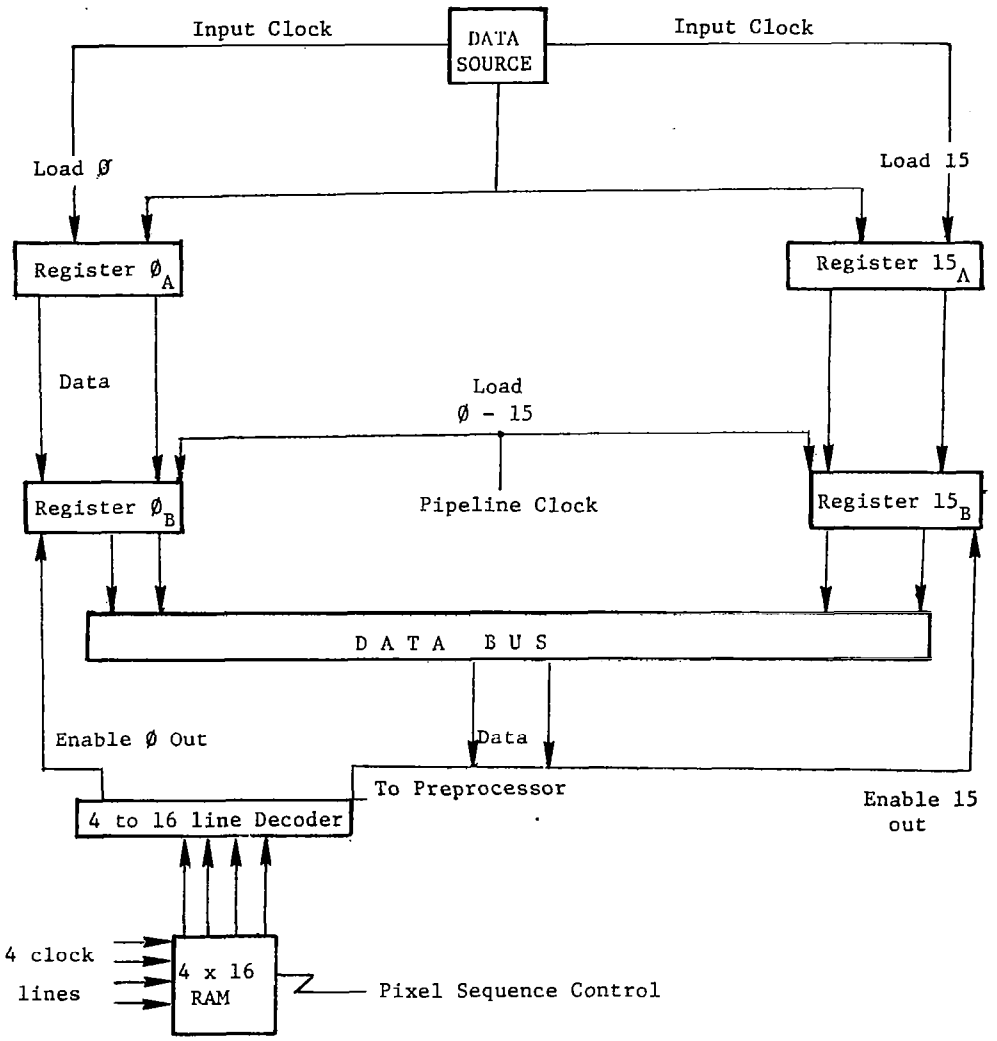


FIGURE 15. DATA FLOW TO PREPROCESSOR

In the playback mode, the bit-synchronizer card accepts a bi-phase signal from the equalizer and produces an NRZ signal and clock information. The card contains in addition to the bit-synchronizer module, some additional circuitry which is selected by field effect transistor switches as a function of the tape speed select lines. This circuitry optimizes the bit-synchronizer for the bit rate which occurs at one of the four selected tape speeds. As above the tape speed select lines are controlled from the computer through the DR11-C. The additional circuitry consists of filters and a selectable binary countdown circuit.

The output signals from the bit-synchronizer card (playback) are: data, clock, 2 × quad clock. The output signal 2 × quad clock is used in the sync lock-detect card.

The bit-synchronizer can be conditioned to operate in several different test modes which are used as diagnostic checks on the system. One test mode involves the playback circuitry. In this test mode the bi-phase input to the card from the tape recorder is removed and is replaced with a signal from a bi-phase generator under computer control.

2.2.1.3 Sync Detect-Lock Cards

There are eight of these cards in the system and the entire card is dedicated to the playback mode.

The sync detect section receives the NRZ and clock information from the bit-synchronizer card. This data is serially loaded into a 16-bit register. As each bit is transferred into the shift register, the accumulated word is interrogated to see if it has the same bit arrangement as the sync word that defines the start of line. This sync word is established by a switch register in the control section.

There are several features of the detect circuitry that allow the system to accommodate data with some adjustment to noise condition. In the section that interrogates the incoming data for the sync word, the detection circuitry can be conditioned by the control section to allow for zero error or one-bit error in the sync word for a valid detection. The bit-by-bit comparison of the incoming data with the defined sync word is accomplished by series of exclusive "or" gates. The output of these gates is fed to a ROM which encodes the number of errors in the comparison, and has as its output a binary word that represents this number. The following circuitry when conditioned by the control section recognizes only a zero or one bit error in the comparison and establishes the existence of a sync word according to the pre-assigned definition.

There are several signals from the sync lock section of the card that are brought into the detect circuitry. The function of these signals is covered in the sync lock description.

2.2.1.4 Sync Detect-Lock Card: Lock Section

The lock section of this card performs two functions. From the information obtained from the detect section concerning the presence of a sync word in the data, the lock section of each channel establishes a lock condition for the particular channel. Once the channel is in a lock mode, the circuitry then monitors the occurrence of sync and generates control signals for the FIFO card for that channel. If the sync word is missing for a pre-set number of scan lines, the circuitry then removes the lock condition and places the channel in an out-of-lock condition. The operation of the detect and lock circuitry operates for each channel independently. However, the algorithms that define the "lock" and "out-of-lock" modes are the same for all the channels and are defined by switch registers.

The lock condition is obtained in the following manner. In the lock circuitry a sync window is defined (the same for all channels), which is a group of consecutive 8-bit words at the end of the scan line and including the next sync word. In the system, the length of the window is defined by a switch register in the control section and can be from zero to 32 words. When the first sync is detected in the detect section of the card, a pulse is sent to the lock circuitry and sets the word clock flip flop. This flip flop enables the bit-clock from the bit-synchronizer to the word clock. The output of the word clock is fed to a word counter (10 bits) which keeps a count on the 8-bit words in the scan line being generated. When the word counter reaches a count equal to the total number of words in the scan line plus the number of words that make up the sync window, (this count is established by the individual switch registers located on the search lock card) a flip-flop is set which is defined as sync search. The sync search flip-flop enables the word clock to the 5-bit sync window counter. If a sync is detected within the sync window a flip-flop is set which defines the lock condition. If, however, no sync has been detected for the full window timing, the lock circuitry is reset and the channel is back to the out-of-lock condition.

Once the channel is in the lock mode it remains in that condition as long as the sync detections occur in a pre-set pattern. This pattern is determined by two bits that are defined by a switch register. The sync detections within the sync window continue as described for lock acquisition. When the channel is in the lock mode and the sync has not been detected this fact is stored in a 2-bit counter, and it is assumed that the sync occurs at the end of the video, window, i.e., where it would normally be located. If, a sync word is detected within the sync window of the next line, then the 2-bit counter is reset and the location of that detection within the window establishes the start of the proceeding line. However, if consecutive detections are missing for a number equal to the 2-bit pattern as defined by a switch register, then the channel is put in as "out-of-lock" and the sequence of detection, search and lock begins again.

2.3 THE CLASSIFIER

The Classifier performs the actual classification of the pixels into categories. The computation it performs is a maximum-likelihood decision, assuming a multimodal Gaussian

multivariate distribution. This assumption has been well justified by many experiments using multispectral data at ERIM and at LARS. Although simpler algorithms can perform well for some data sets, a significant percentage of applications demand this powerful decision rule. No penalty in speed and only a small additional cost occurs in using this algorithm. A digital implementation of the algorithms was chosen over a hybrid or analog implementation because the digital implementation

(1) costs somewhat less to fabricate,

(2) allows more exact repeatability in setup and performance,

(3) provides computer-controlled diagnostics more easily,

(4) uses the most current state-of-the-art digital techniques,

(5) produces a throughput equal to that produced by current hybrid/analog techniques.

The basic calculation to be performed is

$\ell n\{pr(X)\}$

where X is the input data vector (the vector of bytes in a pixel). The probability density function is Gaussian:

$$\ell n\{pr(X)\}_i = -\frac{1}{2}\left\{(X - M_i)^T \theta_i^{-1}(X - M_i) + \ell n|\theta_i| + n\,\ell n\left(\frac{\pi}{2}\right)\right\} \qquad (1)$$

where vector M_i is the expected value of the X vector in category i, θ_i is the variance-covariance matrix for category i, and n, called the number of channels, is the dimension of X, M, and θ. Define m as the number of categories into which the data can be classified, so that i ranges from 1 to m. Then formula (1) is calculated m times for each pixel, once for each of the m categories. The smaller the result of the i^{th} calculation, the higher the probability that the pixel belongs to the i^{th} category.

Formula (1) consists of three additive terms. The most difficult calculation in the equation is the quadratic term

$$Q_i = (X-M_i)^T \theta_i^{-1}(X-M_i) \qquad (2)$$

The term $P_i = \ell n|\theta_i|$, a constant for each of the m categories, is calculated prior to the classification process.

The choices of the number of categories, m, and the number of channels, n, are limited because the number of computational steps increases (1) in proportion to m, and (2) as the square of n. Two sets of choices are possible:

(1) m = 8 and n = 8, or

(2) m = 16 and n = 4

Note that before any processing can occur, the computer must load statistics (including M and θ^{-1}) about the data set and categories into RAM's in the Classifier. Once these have

been loaded, the Classifier accepts each pixel, calculates Q, performs a normalization step, then makes its final decision, as illustrated by Figure 16.

The design of the Q-calculating, or "Quadratic," portion of the Classifier outlined in Figure 16 follows directly from mathematical manipulation of Equation (2). The equation can be expressed in a number of ways to optimize the computation. Since the number of bits in the Classifier is limited, it is desirable to express the quadratic calculation such that the calculated result has a very limited range. The variance-covariance matrix θ can be expressed as

$$[\theta] = [\sigma]\,[\rho]\,[\sigma] \tag{3}$$

where $[\sigma]$ is a diagonal matrix of the standard deviation, and $[\rho]$ is the correlation matrix with all 1's on the diagonal and values of 0 to 1 off the diagonal (in some cases negative values may occur). Taking the inverse of Equation (3) yields

$$[\theta]^{-1} = \left[\frac{1}{\sigma}\right][\rho]^{-1}\left[\frac{1}{\sigma}\right] \tag{4}$$

Substitution of Equation (4) into (2) results in

$$Q_i = \left[\frac{\bar{X}-M_i}{\sigma_i}\right]^T [\rho_i]^{-1}\left[\frac{\bar{X}-M_i}{\sigma_i}\right] \tag{5}$$

The terms $(X-M)/\sigma$ can have a very wide range. However, if the range

$$-8 \leq (X-M_i)/\sigma_i \leq 8 \tag{6}$$

is exceeded, the value of X for that channel is too many standard deviations from the mean to be considered for classification. In this case, truncation of significant bits will occur, causing a flag to be set in the Classifier indicating this condition. This indication is used later to reject a decision that the sample is from the particular class.

The computation of Equation (5) could proceed in a straightforward manner, but can be simplified somewhat due to the symmetry of the correlation matrix and its inverse. This simplification can be accomplished in more than one way. One method is as follows:

$$Q_i = [Y_i]^T[Y_i] = \left[\frac{\bar{X}-M_i}{\sigma_i}\right]^T [B_i]^T[B_i]\left[\frac{\bar{X}-M_i}{\sigma_i}\right] \tag{7}$$

where B is an upper triangular matrix formed by the decomposition of the inverse ρ matrix. By calculating

$$[Y_i] = [B_i]\left[\frac{\bar{X}-M_i}{\sigma_i}\right] \tag{8}$$

the final matrix operation is simply

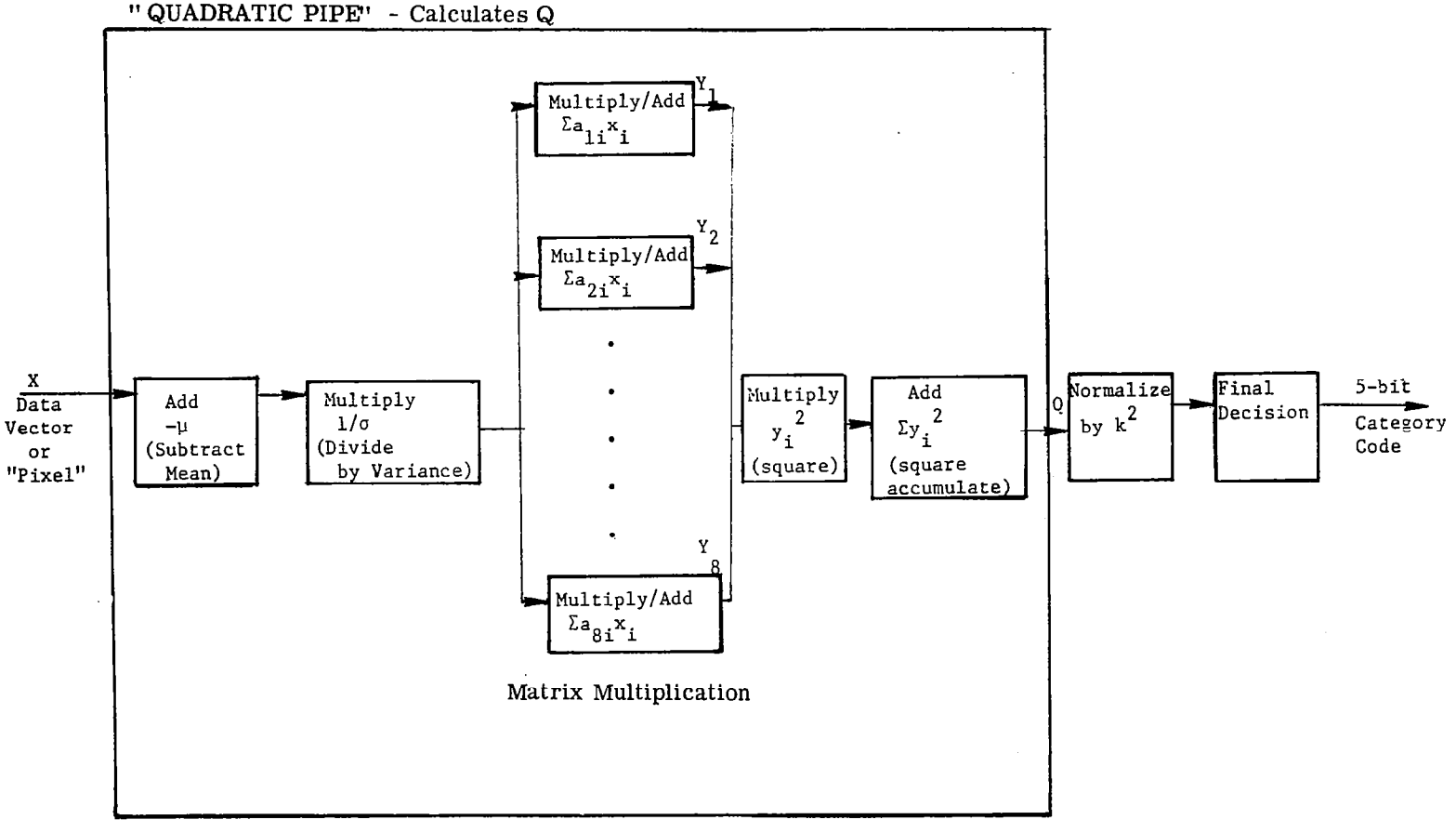


FIGURE 16. THE MIDAS CLASSIFIER

$$Q = [Y_i]^T[Y_i] = \sum_{j=1}^{n} y_{ji}^2 \tag{9}$$

where the y_{ji} are the elements of the $[Y_i]$ vector.

There are four steps implied by Equations (7-9). These are

(1) subtract the mean from each channel

(2) multiply each result by $1/\sigma$

(3) perform the Y matrix multiplication on each result of Step (2) to get Y's

(4) square each resulting Y and add the results together

Another method for calculating Equation (5) is to express the inverse of the correlation matrix ρ^{-1} in terms of its eigenvalues and eigenvectors. The correlation matrix can be expressed as

$$\rho = U\Lambda U^T \tag{10}$$

where the U matrix is comprised of eigenvectors arranged in columns, and U^T is its transpose. The Λ matrix is the set of eigenvalues on the diagonal. Taking the inverse of the correlation matrix, it can be shown that

$$\rho^{-1} = U\Lambda^{-1}U^T \tag{11}$$

which is to simply take the reciprocals of the eigenvalues and multiply by the two original eigenvector matrices. One further decomposition brings us to the desired form

$$\rho^{-1} = [U\Lambda^{-1/2}][\Lambda^{-1/2}U^T] \tag{12}$$

where $\Lambda^{-1/2}$ means $(\Lambda^{-1})^{1/2}$.

Substitution of Equation (12) into Equation (5) yields

$$Q_i = \left\{\left[\frac{\bar{X}-M_i}{\sigma_i}\right]^T U_i\Lambda_i^{-1/2}\right\}\cdot\left\{\Lambda_i^{-1/2}U_i^T\left[\frac{\bar{X}-M_i}{\sigma_i}\right]\right\} \tag{13}$$

In this case, if a vector Y is defined as

$$Y_i = \Lambda_i^{-1/2}U_i^T\left[\frac{\bar{X}-M_i}{\sigma_i}\right] \tag{14}$$

and computed as such, then the final matrix operation can be performed in the same manner as in Equation (9). The hardware required for this second calculation must perform more multiplication than in the first method. However, in order to implement the first method efficiently, a more elaborate switching scheme is needed to avoid multiplying by a large number of zeros. The details of this switching scheme were not worked out and only general consideration was given to it. Since it appeared desirable to have the flexibility and capability to do

the matrix multiplication required by either Equation (9) or (14), the hardware was designed to do full matrix multiplication. Tests were run on over 100 signatures to see if the resultant set of coefficients for either method appeared better suited to a limited-word-length multiplier. From these results it appeared that a slight advantage might be gained by using the second method.

The Quadratic Pipe portion of Figure 16 outlines the physical realization of the calculation of Q. Note how closely it follows the four steps listed above. Precision throughout this portion of the pipeline varies from 8 to 14 bits, with the significance increasing as the data progresses from beginning to end. Q itself is a 12-bit number.

Following the calculation of Q, the 12-bit value is multiplied by a set of 10-bit values called K_i^2, normalizing constants which are pre-determined parameters obtained from normalizing the inverse variance-covariance matrix, θ^{-1}.

The final decision stage of the process completes the calculation of Equation (1) by adding in the logarithm of the determinant, then decides to which, if any, of the categories the pixel belongs. This decision is made by (1) comparing the values of $\ell n\{pr(X)\}$ resulting from the repeated calculations of Equation (1) with different M and θ^{-1} values, (2) choosing the category corresponding to the smallest calculated value, if it is small enough, and (3) outputting its 5-bit code. If none of the values is small enough, a reserved code meaning "none of these" is produced.

2.3.1 SCALING OF THE CLASSIFIER

The calculation of the quadratic form given in Equation (2) above can result in a very large number. Fortunately, we are not interested in the exact result for large values and, therefore, can restrict the range, provided that an overflow condition is detected. The scaling of the quadratic pipe is summarized in Figure 17. All arithmetic operations up to the input of the squaring circuit are 2's complement arithmetic. The input data is a 9-bit word having the range of values

$$-256 \leq \times \leq 255$$

or

$$0 \leq \times 511$$

We are interested in small values of $X + (-\mu)$, but an overflow condition can be detected after adding $-\mu$ which allows us to keep the result to eight bits. Following addition, a multiplication by $1/\sigma$ is performed. A restriction on the range

$$-8 < \frac{X - \mu}{\sigma} < 8$$

is placed on the output of this multiplication. This in effect limits any data channel value to less than eight standard deviations from the mean. It is to be noted here that a lower limit on

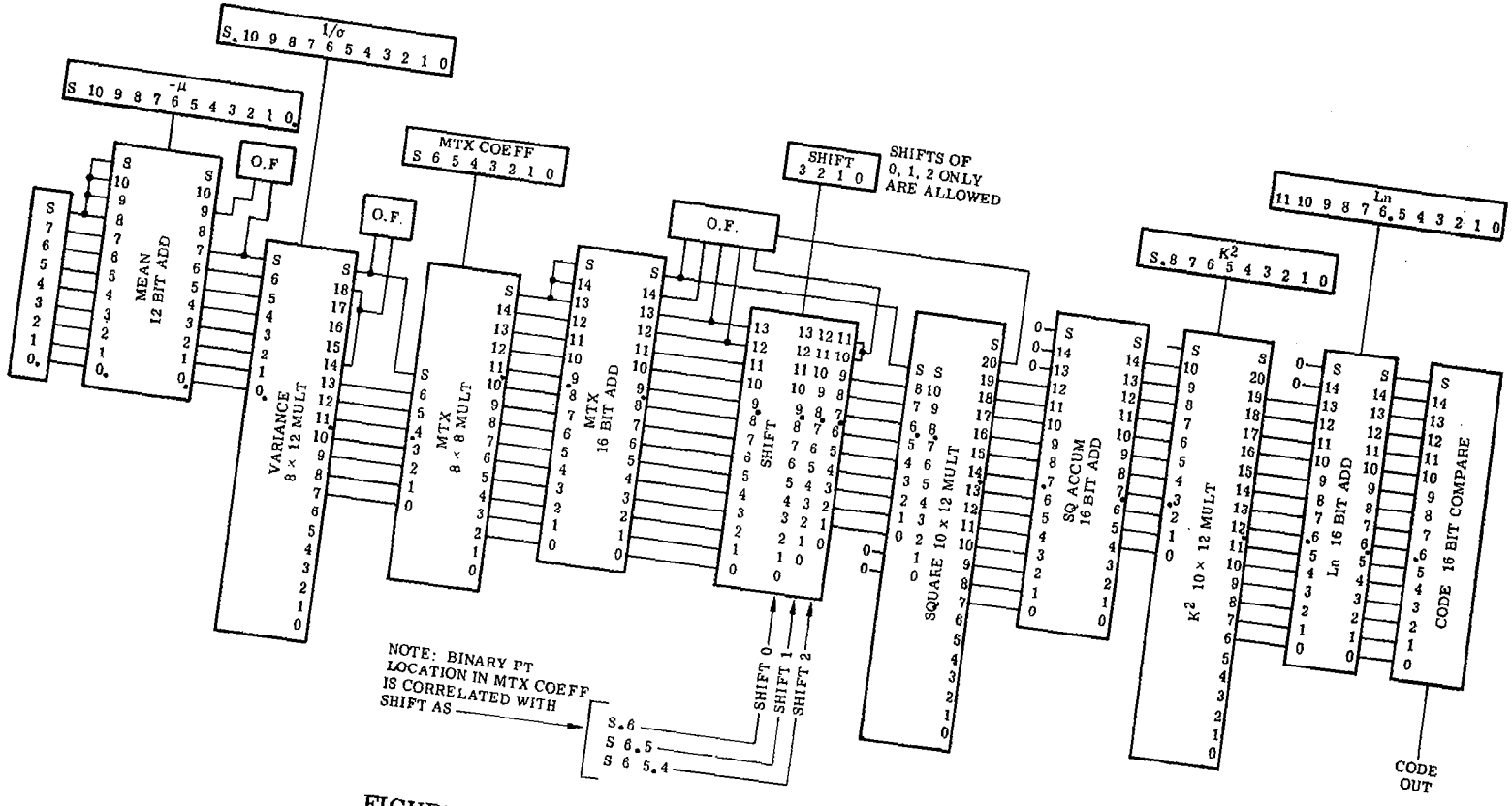


FIGURE 17. BLOCK DIAGRAM OF CLASSIFIER SCALING

σ is required and consequently an upper limit on $1/\sigma$ also. The limit selected is

$$\sigma \geq 1 + (1/2048)$$

or

$$1/\sigma \leq 1 - (1/2048)$$

which appears to be a reasonable choice based upon examining a number of signatures. Also, it would seem reasonable that variation in the data for a given object class would show changes in the two least significant bits.

The above two restrictions determine which high-order bits can be tested for overflow and discarded at the output of the $1/\sigma$ multiplication. The five bits from 2^3 to 2^7 are tested for an overflow condition. If an overflow is detected, an overflow bit is put into and clocked down a shift register, which is parallel to the pipe, in step with that data value, so that at the final test for recognition that calculation is discarded.

The input to the eight matrix multiplier cards is comprised of the sign bit and the seven bits from 2^2 to 2^{-4} from the variance card. It is to be noted that this quantity $(X-\mu)/\sigma$ has zero mean and variance of one when the input data (X) originates from the same distribution as that used to calculate μ and σ. Similarly, after the matrix multiplication is performed in the eight matrix multipliers, the y outputs have zero mean and a variance 1/k where the constant k is the largest value in the $\Lambda^{-1/2}U^T$ matrix. This normalization of the matrix is done in order to get maximum utilization of the 8-bit RAMs and their associated multiplier input. Empirical study of more than 100 signatures indicates that the value of k is from a little over 1 to about 5. If the value of k were not factored out then the variance of the y output would be 1, and a test of

$$-8 \leq y \leq +8$$

could be used. In Phase 1 the overflow test was $-8k \leq y \leq +8k$ which was found to be too large a bound. Therefore a shifter was introduced into the pipeline, as shown in Figure 17, which has the effect of making the overflow test

$$-8k' \leq y \leq 8k'$$

where

$$1 \leq k' \leq 2$$

and

$$k' = k2^{-n}$$

using the proper integer for n. The output of the matrix multiplier is tested on the appropriate bits for overflow.

The constant k which was factored out in the matrix multiplication is reintroduced in a final multiplication stage. Since there is a different k with each object class, the set of k's

can be normalized with largest k factored out. This again makes maximum use of the limited-bit multiplier and RAM.

2.3.2 PIPELINE ORGANIZATION OF THE CLASSIFIER

The pipeline can be organized in many ways. Since the fundamental calculation is the quadratic calculation for n categories of materials, there could be n pipes each performing the quadratic calculation in parallel, or there could be just one pipe performing the calculation sequentially. Speed of calculation and available integrated circuit hardware dictated the organization. It was desired to perform the quadratic calculation in 5 microseconds. The TTL integrated circuits perform arithmetic operations in 200 nanoseconds or less; therefore about 25 arithmetic operations could be performed in 5 μsec. Since almost every arithmetic operation requires a coefficient stored in a memory, 4-bit $\times$ 16-word random access memories (RAMS) were chosen as the fundamental storage element. Thus 16 arithmetic operations are performed in 5 μsec. For example, a mean value is subtracted from data entering the pipe. There are sixteen mean values stored in this operation. They could all be for one object class, or they could be two object classes of eight means each, or they could be four object classes of four means each. The last two were implemented. In the first case, the data vector must be entered twice and the whole pipeline computes the quadratic for two object classes sequentially. Eight object classes are computed with the four pipes. For the second case, where the data vector is comprised of four data values, the vector is entered four times and each pipe computes four quadratic values for four object classes thereby giving a total of sixteen classes being computed in the classifier every 5 μsec.

2.4 THE PREPROCESSOR

The MIDAS preprocessor design provides for scan angle correction of data in both additive and multiplicative modes, for taking generalized linear combinations of (angle corrected) data, and for obtaining ratios of selected pairs of such combinations. Preprocessor outputs will feed the MIDAS Classifier directly, and provisions are available to accept inputs in a variety of formats. The Preprocessor will accept up to 16 input channels, and will supply either 4 or 8 outputs to MIDAS. Figure 18, a block diagram of the Preprocessor, summarizes the operation.

Data into the preprocessor may be in either of the following formats:

(1) 10 or less bits of 2's complement

(2) 9 or less bits of (positive) magnitude

Other formats (negative magnitude, BCD, etc.) would require the addition of translators. Input data is defined as being in the range -4 to +3 127/128 units for the first case or 0 to +3 127/128 for the second case. These data formats and the entire Preprocessor scaling are shown in Figure 19.

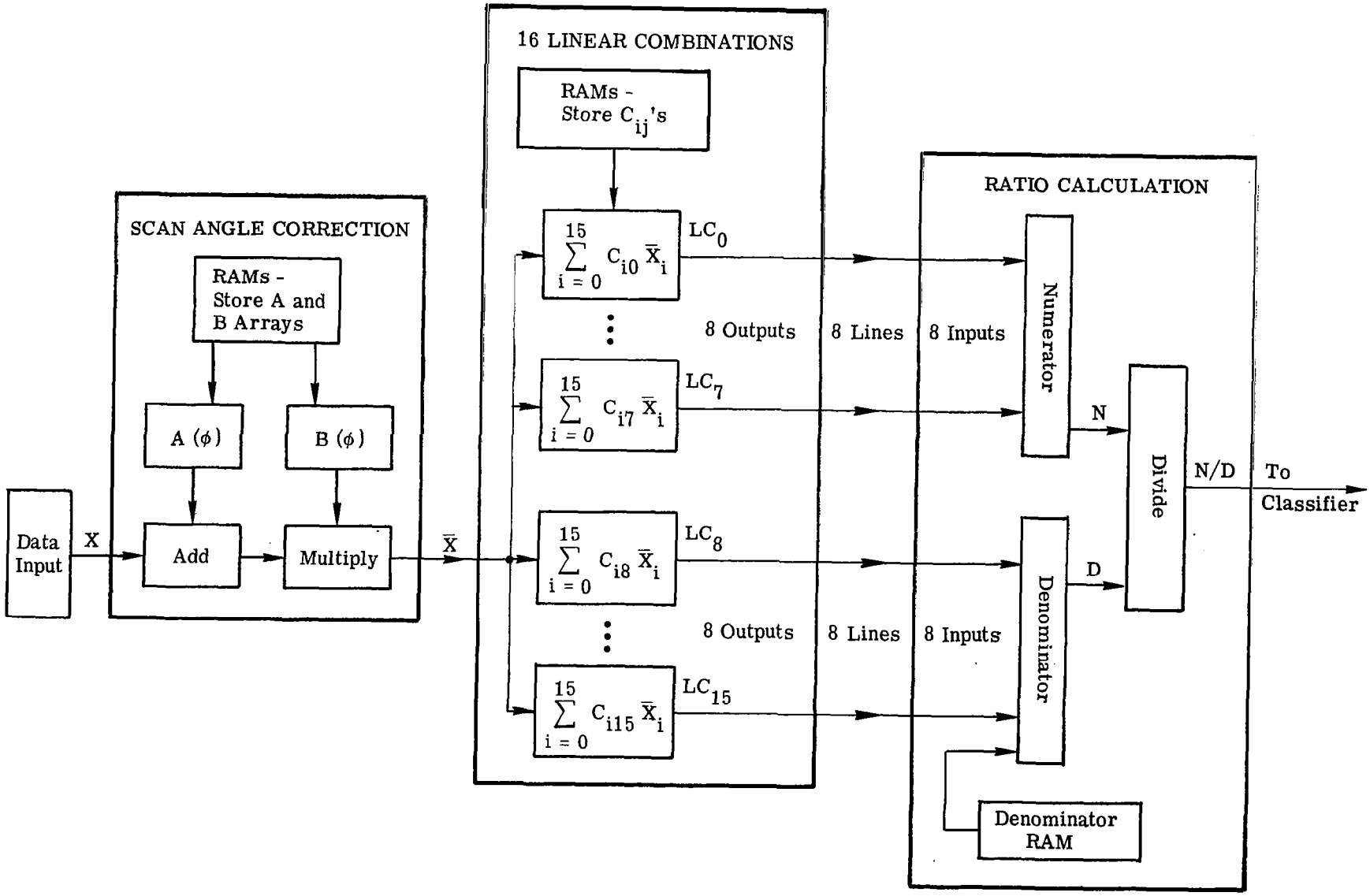


FIGURE 18. THE MIDAS PREPROCESSOR

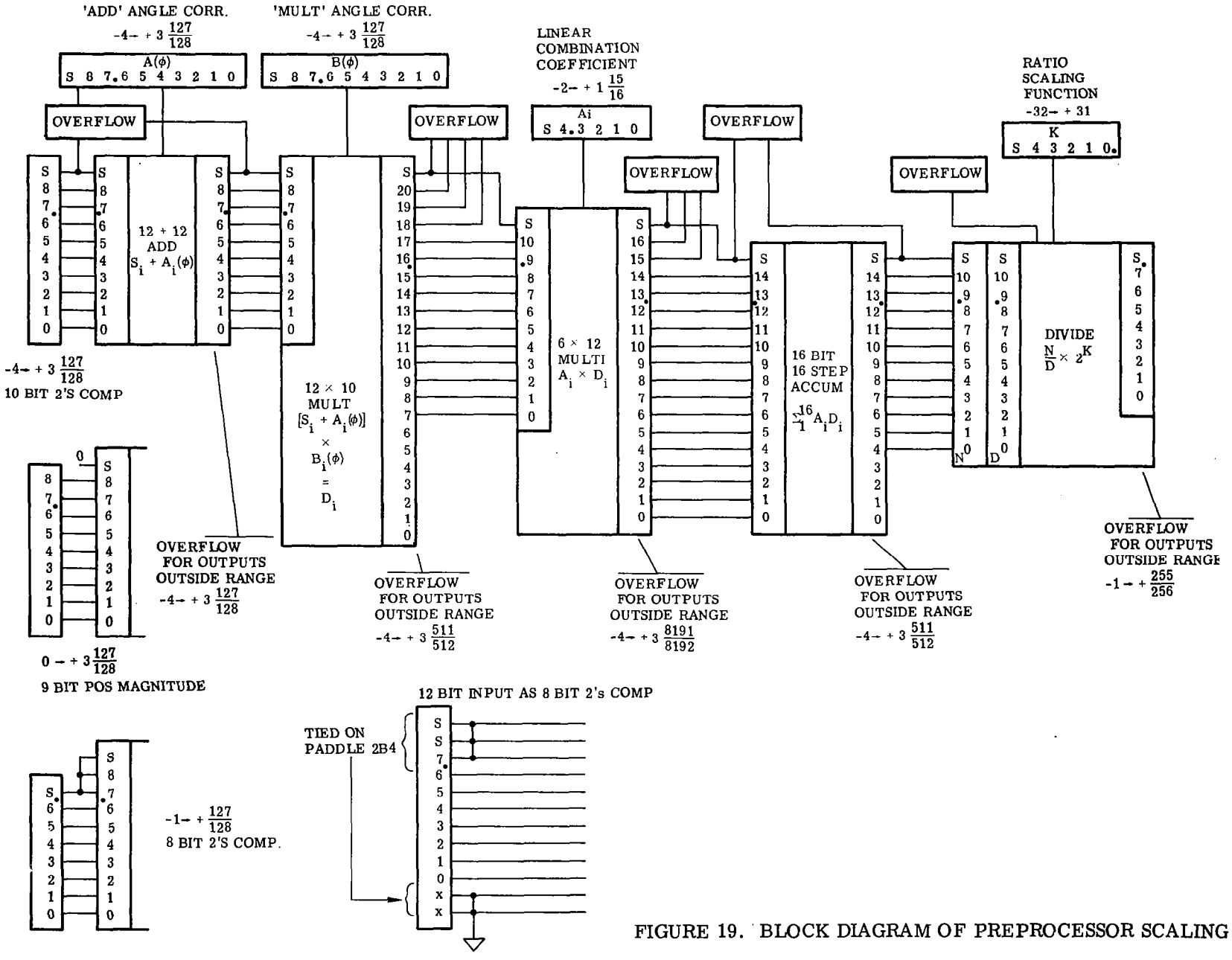


FIGURE 19. BLOCK DIAGRAM OF PREPROCESSOR SCALING

Data input maximum rate is one pixel (vector) per 5 μsec. Each pixel may be described by a vector of up to 16 channels, with maximum channel rate equal to 1 channel/300 nsec. For less than 16 channels/pixel, channels should be repeated so as to provide 16 total inputs per 5 μsec, before changing to a new pixel. All Preprocessor timing is under the control of, and synchronous with, the MIDAS clock and input rates will therefore be a function of the selected clock rate.

2.4.1 ANGLE CORRECTION

Scan angle errors are inherent in the method of collecting multispectral data using an electromechanical scanner. Figure 20 shows a scanner collecting data from one "scan line." The scanner senses points, starting at angle $\phi = -\phi_m$, sweeping to the right, and ending at angle $\phi = +\phi_m$. In a set of scanned data, consistent errors may occur as a function of the angle ϕ. Some reasons for these errors follow.

(1) The scanner aims directly perpendicular to the ground when $\phi = 0$, but at a different angle when ϕ is near $\pm\phi_m$. Thus, sensed values may differ between the middle and the end of the scan line.

(2) A longer distance must be traversed between the scanner and point A than between the scanner and point B (see Figure 20). A longer path through more dust particles and haze in the air will cause different results to be recorded than will a shorter path.

(3) The sun to the right or the left of the scanner will cause an uneven reflection pattern.

(4) Constant malfunctions or idiosyncrasies of the scanner as a function of ϕ are possible.

The Preprocessor can correct known scan angle errors in two ways.

Additive Correction. The Preprocessor can add to the pixels in every scan line a constant, pre-stored function

$$A(\phi) = A_0, A_1, A_2, \ldots A_{k-1}$$

where k is the number of additive corrections per scan line, and each A is a 16-component vector. If, for example, the result of scanning a uniform area produces a scan line of data like the solid line in Figure 21, the Preprocessor can add back to every scan line the values sketched by the dashed curve.

Output of the add function must be in the range -4 to +3 127/128. An overflow test is performed to check the add output,and an overflow bit is generated which moves in step with the data value through the Preprocessor to the Classifier where it eventually enters into the decision process.

Multiplicative Correction. The Preprocessor can multiply the pixels in every scan line by a constant, pre-stored function

$$B(\phi) = B_0, B_1, B_2, \ldots B_{k-1}$$

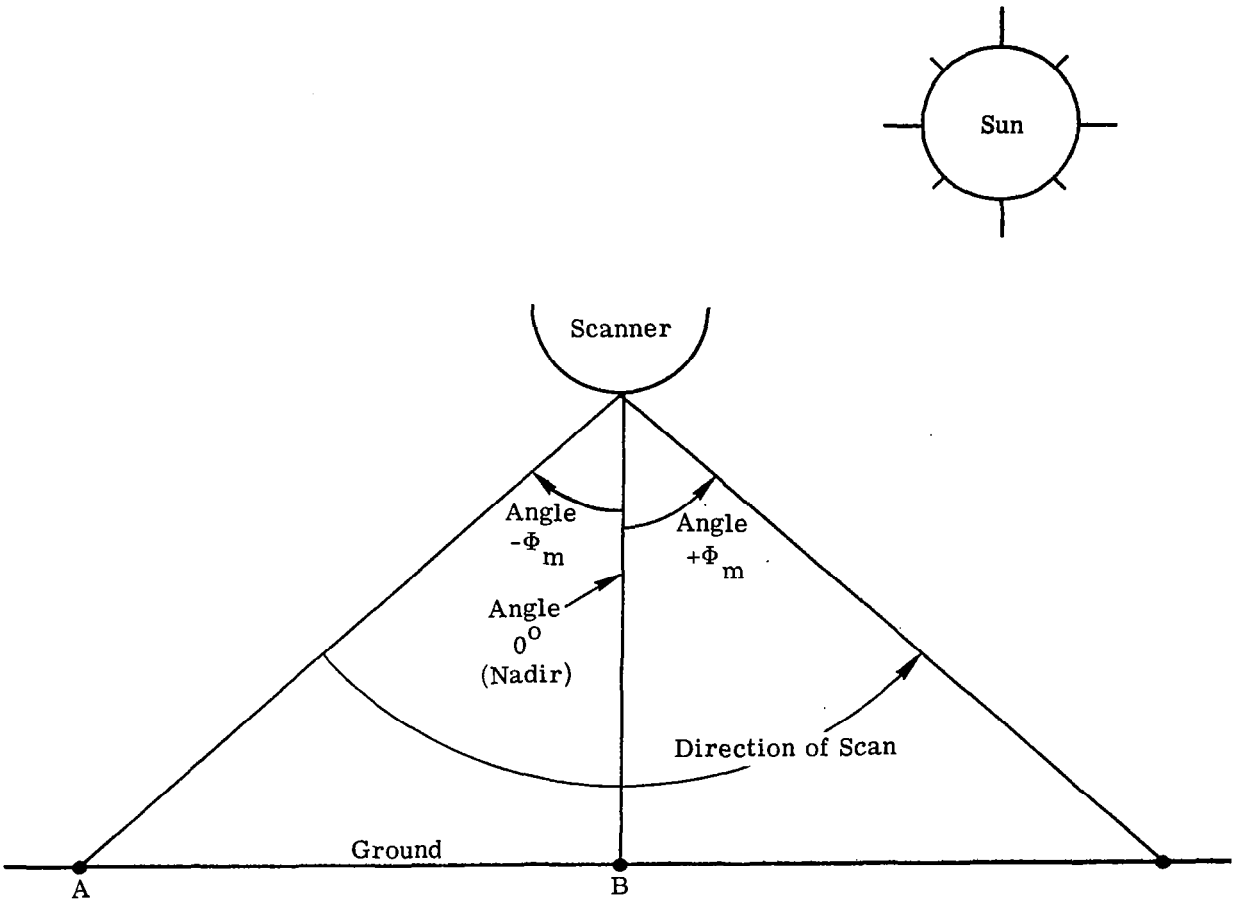


FIGURE 20. REMOTE SENSING SCANNER

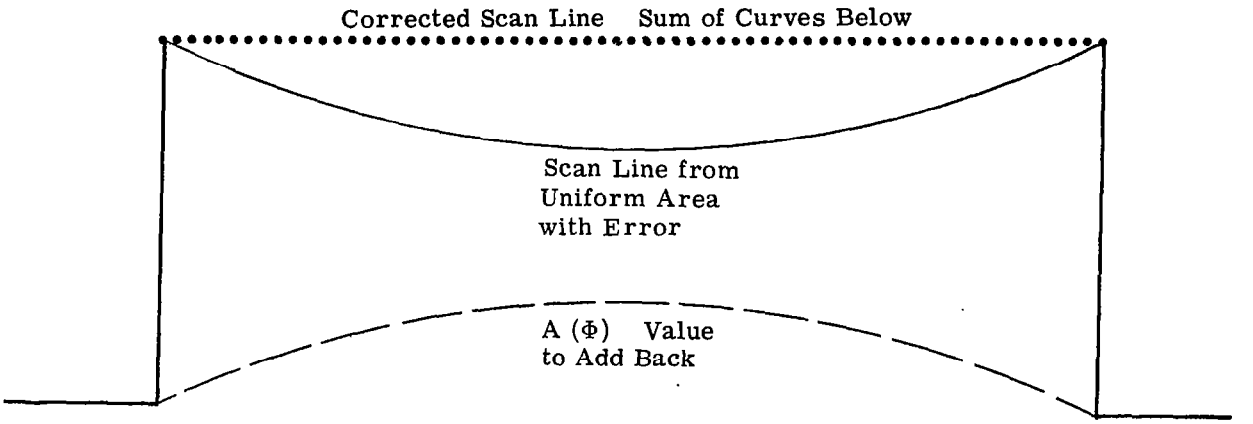


FIGURE 21. ADDITIVE SCAN ANGLE CORRECTION

where k is the number of multiplicative corrections per scan line, and each B is a 16-component vector. As Figure 18 shows, both an additive and a multiplicative correction can be made to each pixel so that the corrected X, called $\overline{X}$, is

$$\overline{X} = (X+A) * B$$

Before the classification process, the A and B arrays are stored in internal RAMs, though not in the form $A_0, A_1, A_2, \ldots A_{k-1}$. Instead, the RAMs contain $A_0, \Delta A_1, \Delta A_2, \ldots \Delta A_k$, so that $A(\phi=j) = A_0 + \Delta A_1 + \Delta A_2 + \ldots + \Delta A_j$, where j can be any number up to 1024. Note that these stored values can be chosen so that they will also scale the input channels at the multiplier output.

The angle correction described thus far may be termed a "one dimensional" correction. If the values stored in the RAMs can be updated between scan lines then the correction may be termed a "two dimensional" correction. This is possible if the input data is coming from the Unibus. However it is not possible when the data is coming from HDT or analog tape.

Data from spaceborne scanners such as LANDSAT require that preprocessing functions be updated periodically, but not necessarily for every scan line. The method of providing two-dimensional angle correction in the Preprocessor is to have two sets of RAMs containing the correction functions. One set is the active set providing the current correction functions while the second set is being updated from the Unibus.

2.4.2 LINEAR TRANSFORM

The next stage of the Preprocessor, shown in Figure 18, performs the linear combinations of channels. Input signals, having been corrected for systematic errors by $A(\phi)$ and $B(\phi)$, are applied simultaneously to sixteen linear combination modules. The outputs of these cards are wired into two groups of eight each. One group is termed the numerator and the other the denominator for subsequent ratio processing. Each LC card performs the function $\Sigma_{i=0}^{16} A_i x_i$ where x_i are incoming data and A_i are coefficients with range -2 to +1 15/16. Overflow tests are performed on individual products for the range -4 to +3 8191/8192 and on the sum of products, whose maximum range is -4 to +3 511/512. It is evident, therefore, that magnitudes of coefficients must be related to the number of non-zero coefficients; i.e., for a sum of all 16 inputs, the maximum coefficient guaranteeing no output overflow is 1/16. There are 256 values of A_i's to be stored, initially, in RAMs. Suitable coefficients can be stored such that

(1) the $\overline{x}_i$'s are unchanged
(2) the $\overline{x}_i$'s are in a rearranged order
(3) the $\overline{x}_i$'s are individually scaled
(4) there is a linear combination of any or all of the $\overline{x}_i$'s
(5) any combination of the above

2.4.3 RATIO OPERATOR

The ratio calculation stage performs eight divisions, then repeats these same eight divisions again, providing the results sequentially to the Classifier. For each division, the stage

(1) chooses any one of the eight numerators provided by the previous stage
(2) chooses any one of the eight denominators provided by the previous stage or a constant from the denominator RAM
(3) begins a pipelined divide operation
(4) supplies the results, five time-steps later, to the Classifier

The sequential choices of the numerator and denominator are pre-specified by the user in the form of a code stored in another RAM.

The linear combination and ratio calculation stages enable powerful, flexible system use. Among their applications are

(1) simulation of proposed scanners
(2) selective scaling of channels
(3) reduction of dimensionality from 16 channels to eight

2.5 MIDAS PIPELINE HARDWARE DESCRIPTION

2.5.1 THE CLASSIFIER

The classifier section of the MIDAS System consists of four bays, each containing 13 wire-wrap cards. These bays are nearly identical in that each has the circuitry for one quadratic pipe computation as described in Section 2.1. This computation requires twelve cards: (1) a mean card, (2) a variance card, (3) eight matrix multiplier cards, (4) a squaring card, and (5) a square accumulate card. These card types are described in following subsections. In addition, each bay contains one of the following one-of-a-kind wire wrap cards: k^2 card, recognition card, diagnostic/output card, and clock card; these wire-wrap cards are also described in succeeding text.

The wire-wrap hardware employs Augat 8136-URG1TG universal circuit boards and 8170-RG1 card racks and back planes. The cards can hold up to fifty 16-pin integrated circuits (ICs) or eighteen 24-pin ICs or a mix of both.

Mean Card

The mean card was updated from the one used in Phase I of the MIDAS program. It was expanded from 8-bit capability to 12-bit capability to handle the output of the preprocessor divide card. This also permits the inputting of 8-bit magnitude numbers. The prime purpose of this card is to sequentially add the stored $-\mu_i$ to the data vector component using 2's complement arithmetic. An overflow test is defined by overflow = $7 \cdot \overline{9} + \overline{7} \cdot 9$ where 7 and 9 are the adder output at those bit positions. This overflow test assumes that after the mean is subtracted from a data vector the output magnitudes must be smaller than the input magnitudes if

there is any chance that the data vector came from the distribution associated with that vector. The X inputs are quotients from the ratio section of the Preprocessor. The output of the adder is loaded into latches 74174 as processor outputs and into 8T10s for diagnostic outputs. A block diagram of this card is shown in Figure 22.

There are two selection devices on the mean card, one to select a RAM in that bay for writing into that RAM and the other to select a diagnostic bus output to the diagnostic card. In the RAM selection, a RAM write select comes from the clock card which does a bay select decode to enable the decoder on the mean card; this also sends four bits to the decoder on the mean card. A read pulse is then applied which can propagate through only one decoder input on one mean card to a RAM on one other card in that bay. In the diagnostic bus output selection an identical procedure is followed except that the origin of the signals is on the diagnostic output card.

Variance Card

The prime purpose of this card is to multiply an 8-bit number by a 12-bit number and put out an 8-bit result with overflow. There are no latches in the processing stream on this card, but the diagnostic bus has a 9-bit 8T10 latch. The input, $X-\mu$, is represented by

XXXXXXXX. (2's complement number)

The coefficient $1/\sigma$ is represented by

X.XXXXXXXXXXX (2's complement number)

Since $1/\sigma$ is never negative, the sign bit is always zero. The eight output bits of the 20-bit product are selected from

```
X-----XXX.XXXX-------
11   11        76      0
98   43
```

The overflow test simply consists of testing bits 14 through 19 to see whether they are all at the same level. If so, there is no overflow and the overflow output is high. A block diagram of this card is shown in Figure 23.

Matrix Multiplier Card

This card has two main functions: (1) to multiply $[(X-\mu)/\sigma]_i$ by a coefficient, and (2) to accumulate several products of this multiplication. The $[(X-\mu)/\sigma]_i$ input from the variance card is stored in a latch on this card. The accumulator is cleared at the start of the matrix multiplication for each signature computation. Each 16-bit product of the multiplication is truncated to the high-order 14 bits and latched before being applied as one input to the accumulator. The accumulator consists of a 16-bit adder with a feedback latch providing the second input. The two expansion bits in this accumulator are adequate for eight adds since the largest product is 2^{12} (the result of $(-2^7) \times (-2^7)$ after truncation). Added eight times, this number yields 2^{15}, which is the largest negative 12-bit number. Since the two operands were negative,

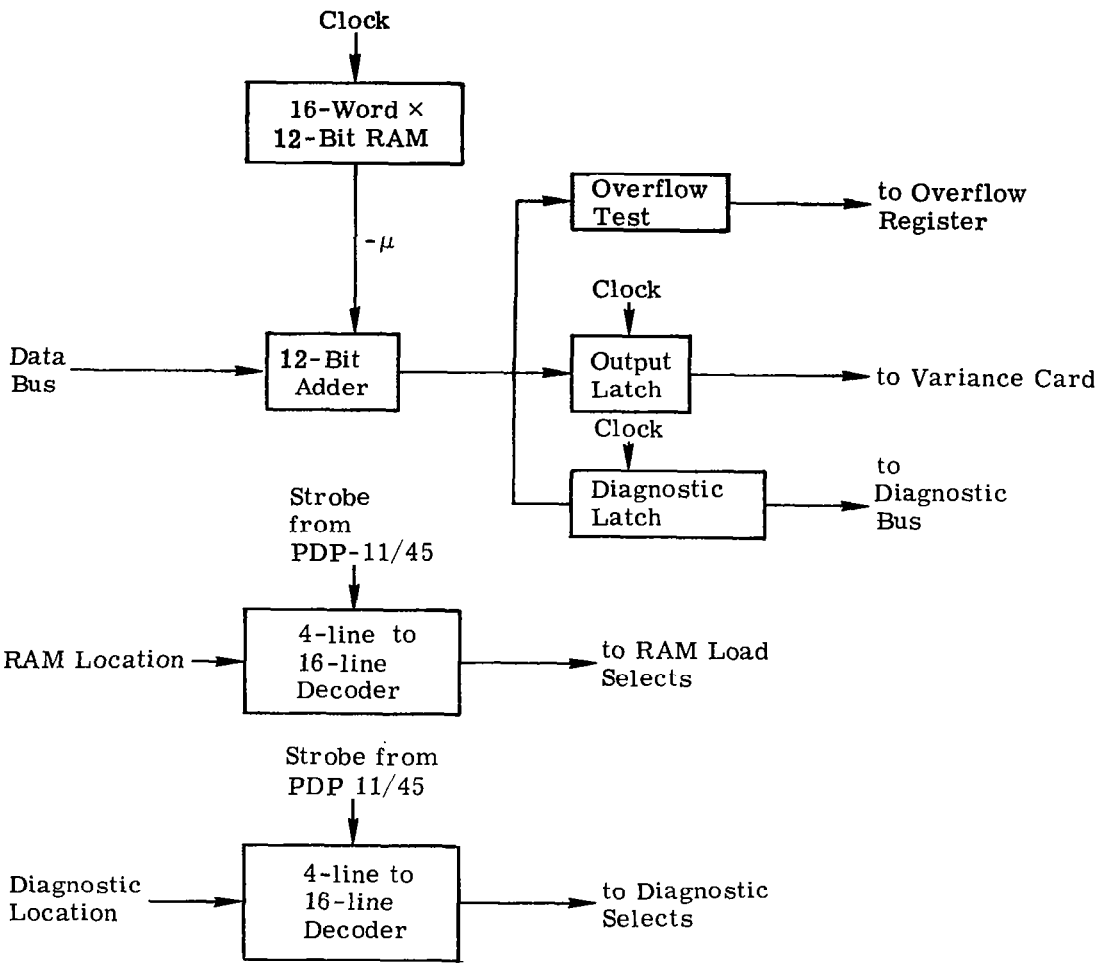


FIGURE 22. BLOCK DIAGRAM OF THE MEAN CARD

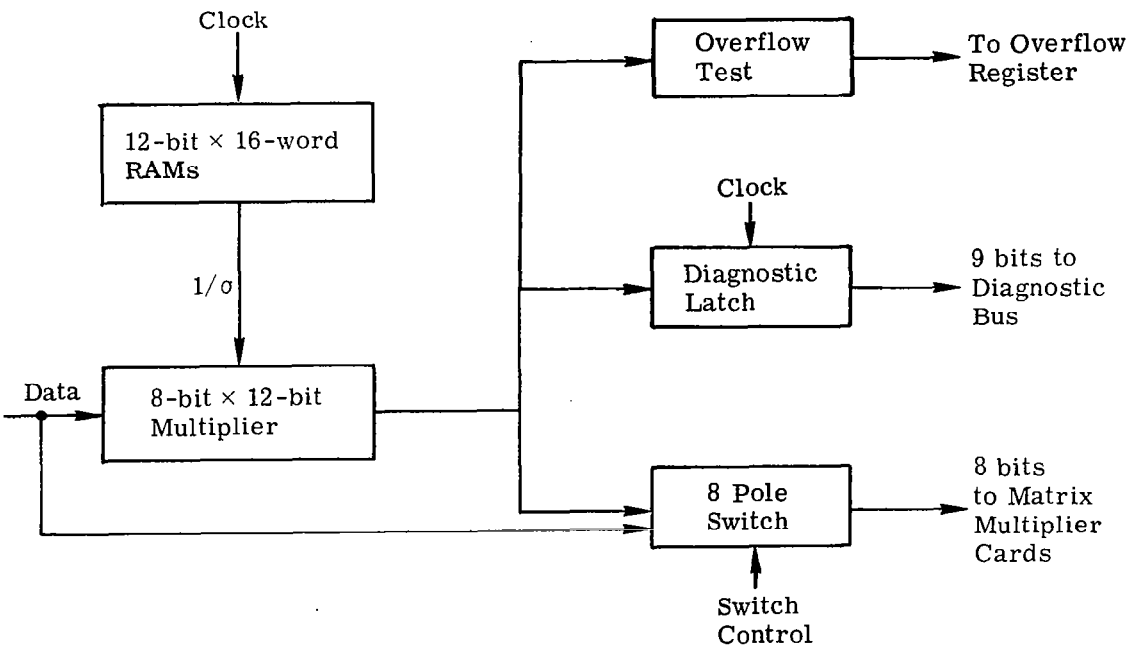


FIGURE 23. BLOCK DIAGRAM OF THE VARIANCE CARD

the accumulated products should be positive. Hence, for this special case the overflow has caused an error in sign only. However, this number is to be squared in the next step, thus correcting the only possible error. There are processor data-stream output-latches (8T10s with 16 bits) and diagnostic bus latches (8T10s, also with 12 bits). These latches are enabled at the proper time by the MTXMPX signal from the square-accumulator card and loaded by C0, the fundamental clock frequency. A block diagram of this card is shown in Figure 24.

Square Card

The first operation preparatory to squaring the matrix multiplier output, is to perform the shift as described in section 2.3.1. Next the overflow test is applied to the four to six most significant bits (depending on the shift used), and the square input is reduced to the next 10 most significant bits, not included in the overflow. A 10×10 bit square operation is then performed on the card. A block diagram of this card is shown in Figure 25.

Square Accumulator Card

This card's main function is to accumulate the squares from the square card. It has both input and output latches as well as feedback latches. It has a 13-bit input (the 2 MSB's and the low-order seven bits from the 10×10 square are not brought in). The adder is 16 bits and three expansion bits are provided for eight adds. The twelve most significant bits are brought out via 8T10s on both the processor data path and diagnostic path. The diagnostic bus is multiplexed from the diagnostic/output card, whereas the processor bus is multiplexed from the k^2 card. The square-accumulator card also has a number of control functions. It

(1) acts as a buffer for the clocks (bay buffer)
(2) multiplexes the matrix multipliers (enables 8T10s on the two output busses which are then loaded at the next C0 clock)
(3) controls the matrix multipliers' output bus to the 10×10 square card using a 3-line to 8-line decoder
(4) clears the matrix multipliers' accumulators
(5) clears its own accumulator
(6) multiplexes its own output 8T10s to the processor and diagnostic busses
(7) contains an overflow shift register. Overflows from the mean, variance, and 10×10 square cards are loaded into the shift register and then dropped into the overflow output register (8T10s) at the proper time

A block diagram of this card is shown in Figure 26.

k^2 Card

The main function of this card is to perform 10-bit by 12-bit multiply. The 12-bit data comes from the square accumulators and the 10-bit data is the k^2 normalizing constant. Its input is buffered and there are no input latches since the outputs of the square-accumulator cards are latched and are bussed into it. There is a 74174 output latch register.

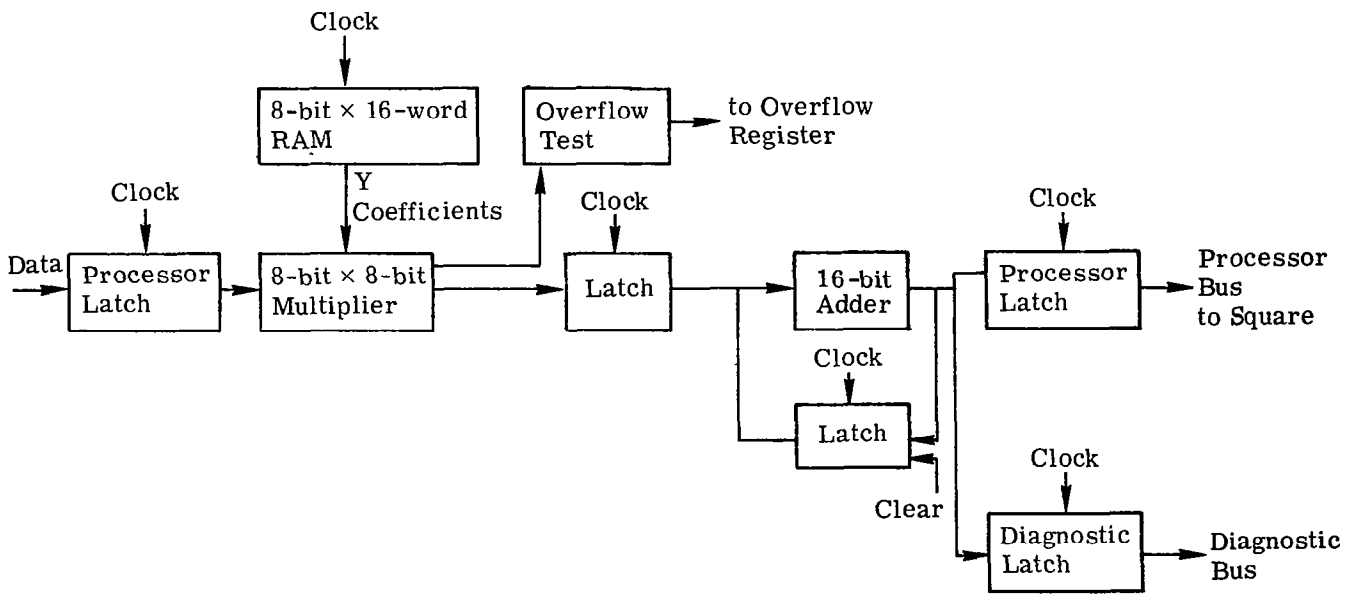


FIGURE 24. BLOCK DIAGRAM OF THE MATRIX MULTIPLIER CARD

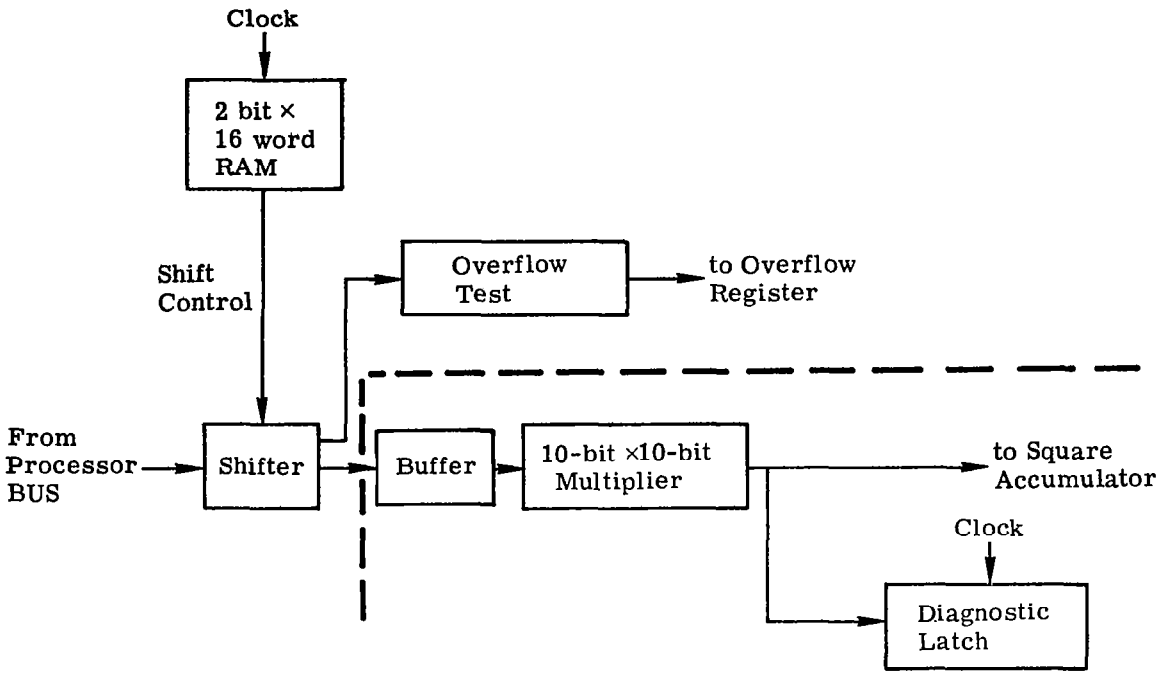


FIGURE 25. BLOCK DIAGRAM OF THE SHIFTING NETWORK AND THE SQUARE FUNCTION CARD

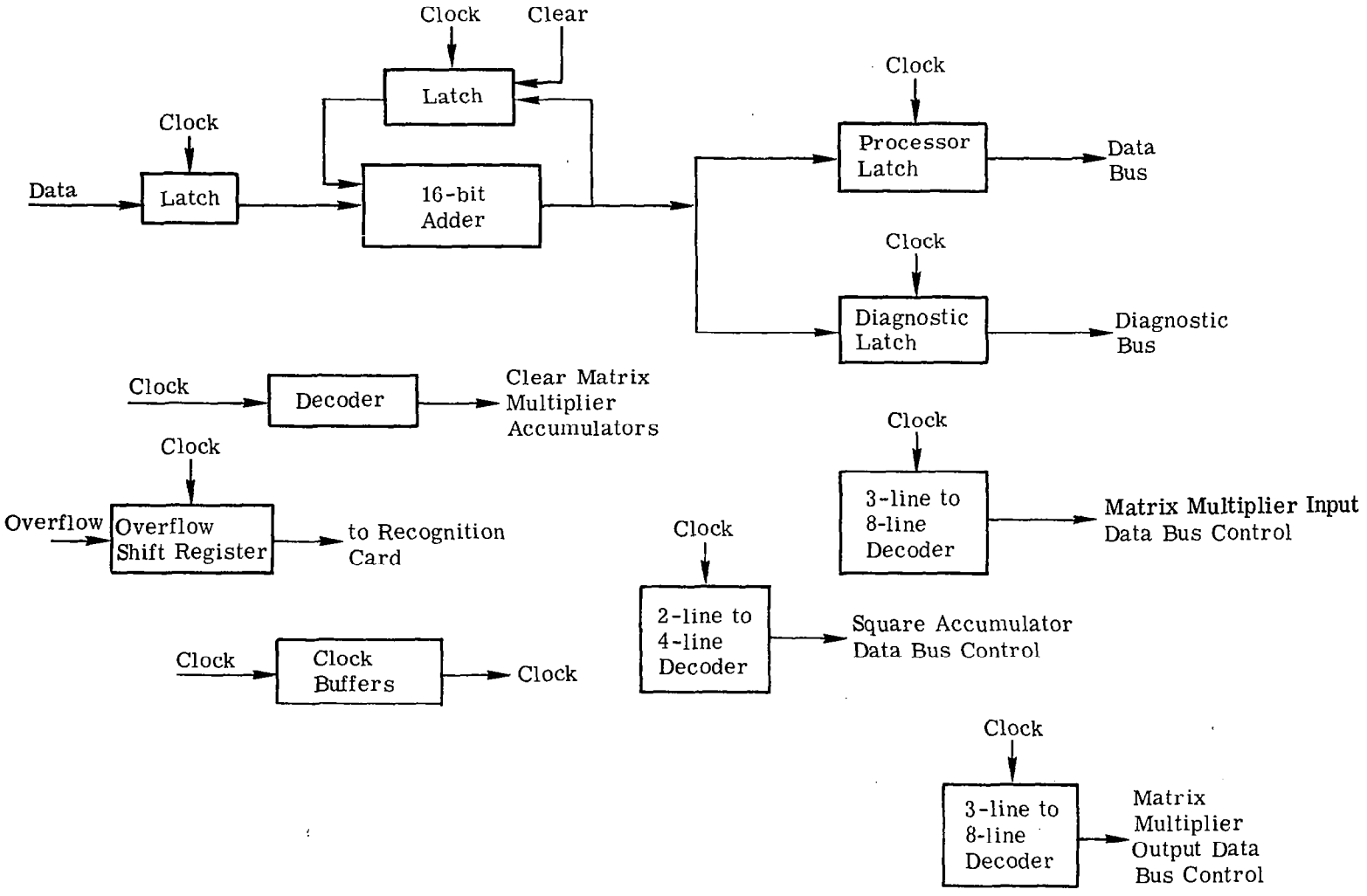


FIGURE 26. BLOCK DIAGRAM OF THE SQUARE-ACCUMULATOR CARD

Another function is to control the square accumulators' bus, which it feeds using a 2-line to 4-line decoder. The signal that multiplexes the output bus also multiplexes the overflow bus to the recognition card. A block diagram of this card is shown in Figure 27.

Recognition Card

The main function of this card is to determine which calculated material's exponent is the minimum. It also performs the Chi^2 test on the sum of the squares. At clock times 5 and 13 a new set of sum of squares from the k^2 card becomes available. A 14-bit add is performed with the $\ell n|D|$ input truncated to bits 0 through 11, while the sum of squares is a 14-bit number in bits 0 through 13.

The selection time of the overflow into the recognition logic is the same as the selection time of the square accumulation output; therefore the overflow shift register output must have two stages of latches to make up for the delay in the k^2 multiply and in the $\ell n\ |D|$ add on the recognition card. Similarly, the output of the Chi^2 test must be latched to account for the $\ell n|D|$ add delay.

A new sequence of testing begins at clock time 8 for the 8-channel set-up or at clock time 14 for the 4-channel set-up. At these times the previous recognition result is transferred to the output latches. There are two internal holding latches, one for exponent and one for material number. After loading these two quantities into the output latches, the internal ones are cleared. There is an artificial seventeenth bit set at this time which makes the stored exponent look as if it is larger than any 16-bit computed exponent. Thus, the first test will always succeed and the exponent will be stored if no overflow has occurred. When this condition is met the artificial bit is reset. The material code is a 5-bit number. It is MSB 0XXXX where the X's give the material number (0 through 15); code 16 indicates no material selected. A block diagram of this card is shown in Figure 28.

Classifier Timing Card

The Classifier consists of the four "pipeline" computers in parallel whose outputs finally converge on the circuits to scale the exponents of the density function and intercompare these exponents for a decision. The sequence of operations can be visualized as shown in Figure 29 where data is shown entering the A-D converters at t = -16 in the upper left corner of the diagram. A sample, consisting of a vector of eight elements of eight bits each is passed through the computational circuits indicated and emerges at the bottom right of the diagram as a classification code of five bits. The general appearance and function of the arithmetic operations so diagrammed is that of a "cascade" in which the breadth of the cascade in time is proportional to the computational load of a particular circuit.

Each "pipe" or "cascade" processes the computation of the quadratic form for the exponent of the Gaussian distribution for two distributions. There are, then, a sequence of alternating computations of the first exponent in cascade 1, the second exponent in cascade 1, the

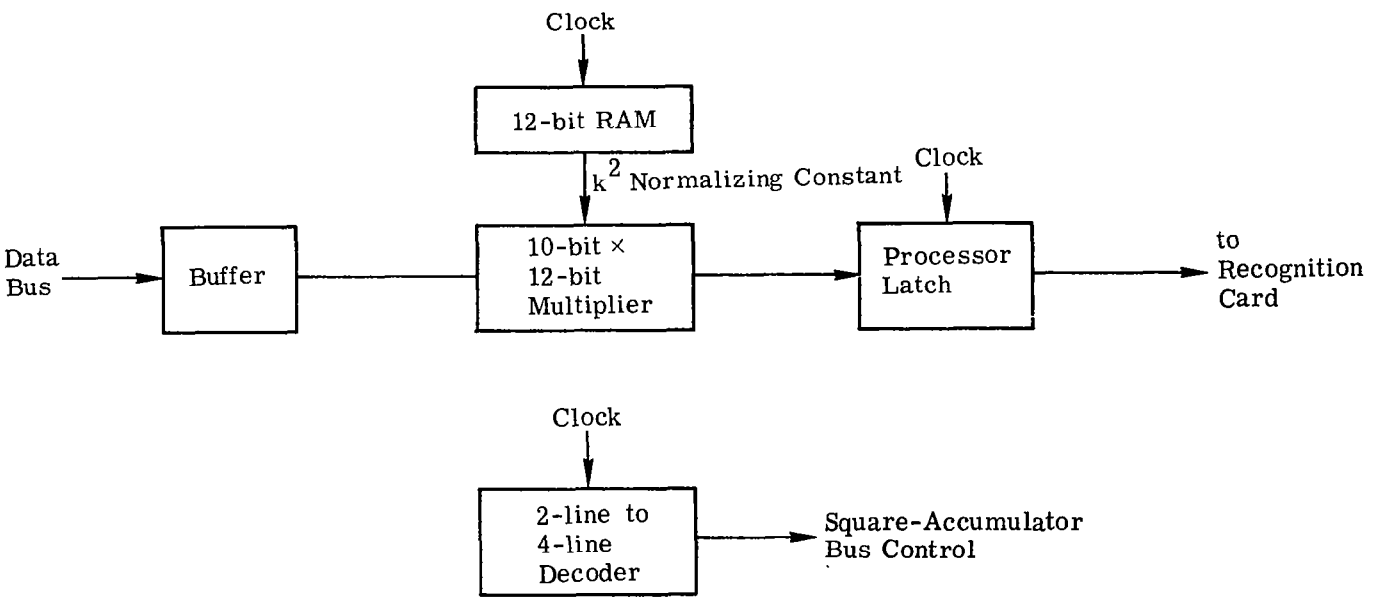


FIGURE 27. BLOCK DIAGRAM OF THE k^2 CARD

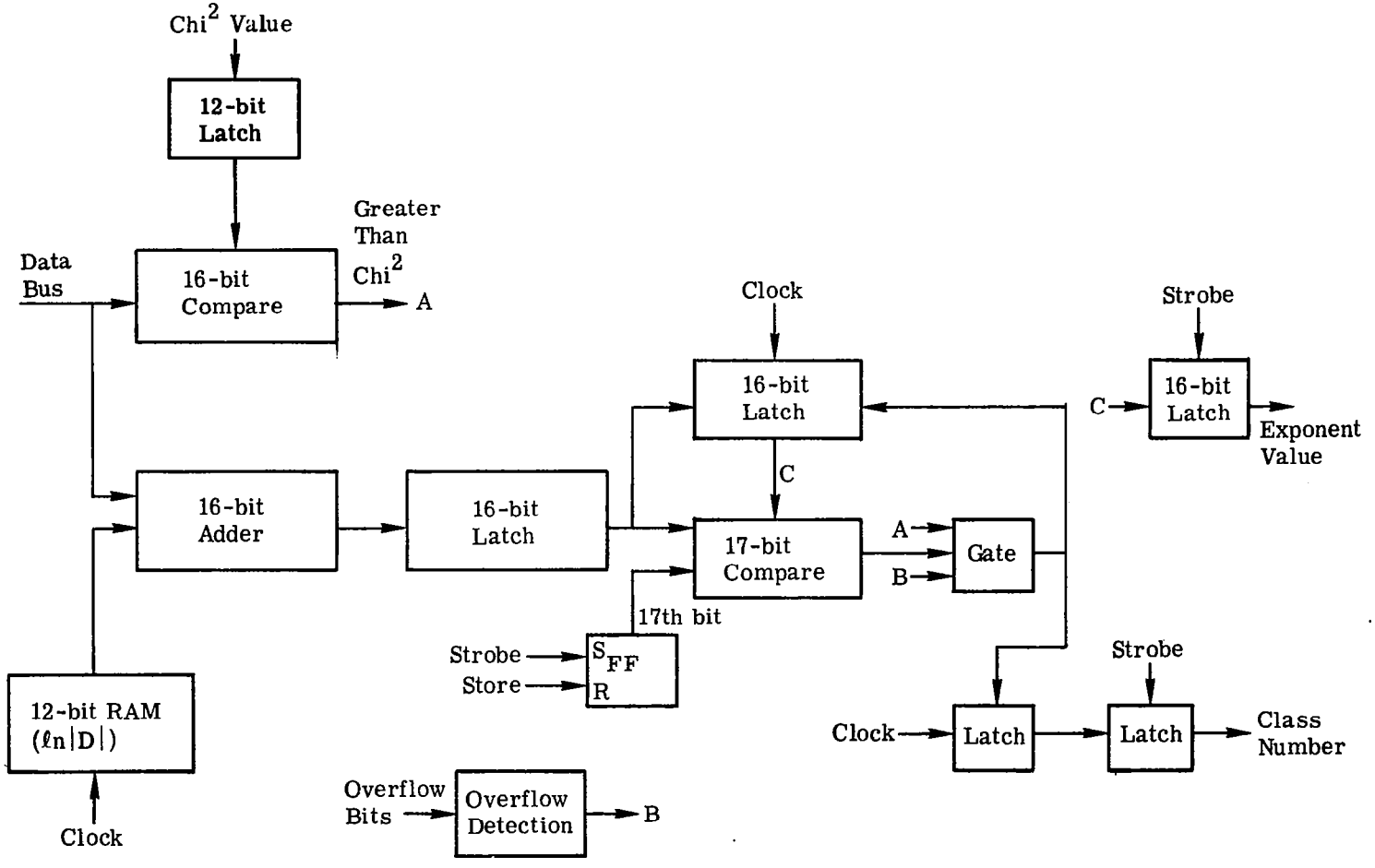


FIGURE 28. BLOCK DIAGRAM OF RECOGNITION CARD.

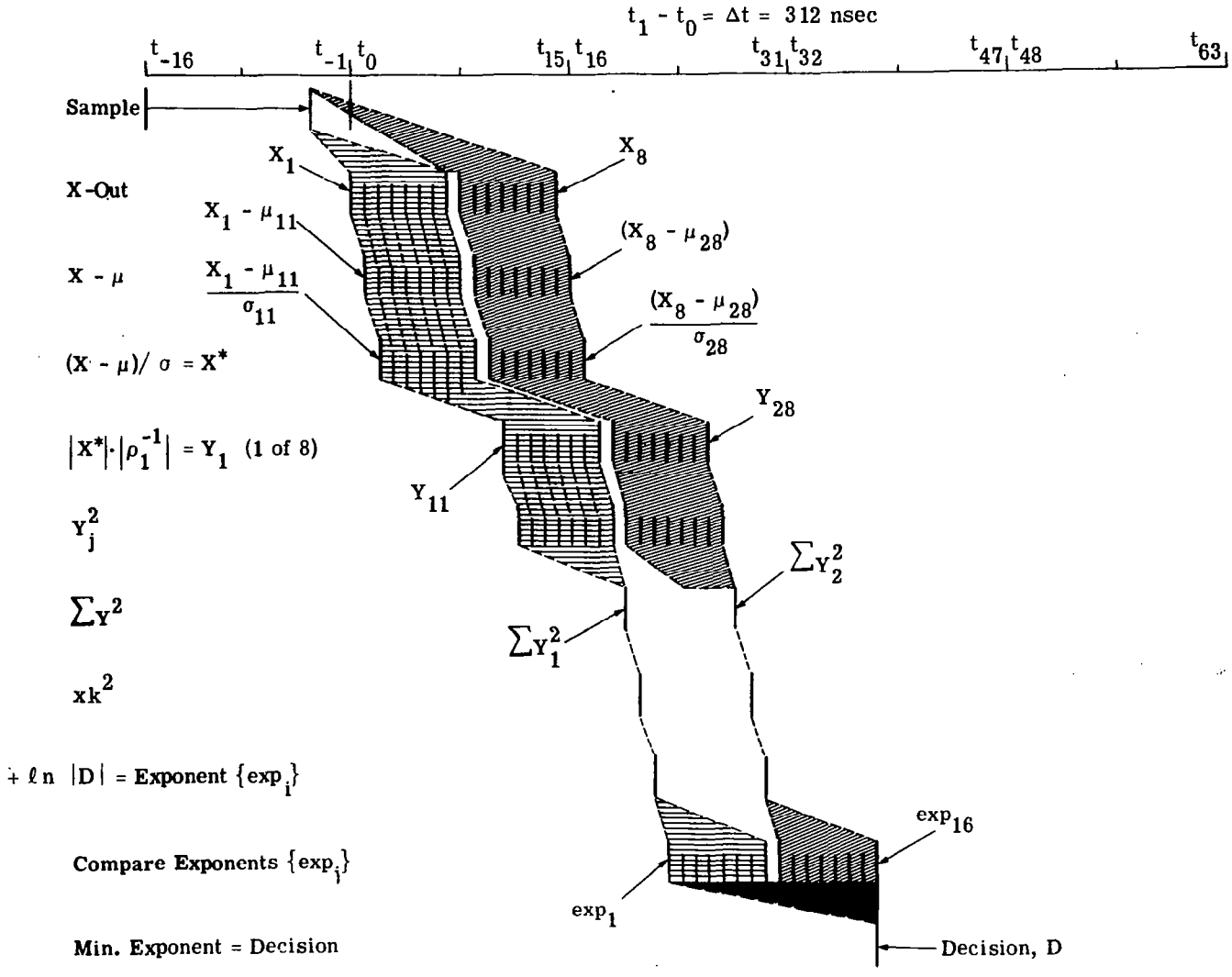


FIGURE 29. DIAGRAM OF CLASSIFIER TIMING

third exponent in cascade 2, etc. In the present machine, there are four such cascades operating in parallel, allowing the computation of eight exponents at once.

The timing diagram (Figure 29) shows the flow of two exponent computations and the resulting decision, neglecting the fact, for the sake of simplicity, that the cascade would normally contain portions of other computations for the preceding and subsequent samples. Data is entered into the A-D converters and is available for computation at the end of 16 machine cycles, or approximately 5 μsec. Data latched in the converter outputs is then supplied sequentially ($X_1, X_2, \ldots X_8$) via a multiplexer and subtracter to a latch, and one clock cycle per element is allowed for this operation. At t = 0, μ_1 is subtracted from X_1; at t = 1, μ_2 is subtracted from X_2, etc. Also, at t = 1, $(X_1 - \mu_1)$ is multiplied by $(1/\sigma_1)$ yielding X_1^*. At t = 2, X_1^* is supplied to each of eight multiplier-summers which compute the products $(X_1^* \cdot P_{11})$, $(X_1^* \cdot P_{21})$, $\ldots (X_1^* \cdot P_{81})$ and enter these into the summers. At t = 3, the multiplier-summers compute $(X_2^* \cdot P_{12})$; $(X_2^* \cdot P_{22}) \ldots (X_2^* \cdot P_{82})$ and add these products to the previous results. Thus at t = 10 the summers contain the complete sums of products for all matrix operations. Each of the eight multiplier-summers may, as a result, be considered as a row operator since it accomplishes the sequence of multiplications and summations for a particular row.

As the result of the above transform, all the elements of this vector are uncorrelated; therefore it needs only to have its elements squared and summed to obtain the normalized quadratic form for the exponent. This is accomplished during cycles t = 10 to t = 18, allowing one cycle for each squaring operation and a final cycle for storage of the summation.

At this point, the exponent must be re-scaled and the natural logarithm of the determinant of the covariance matrix added to obtain the final exponent of the density function. This requires two cycles. The comparison of these exponents, now becoming available from the normalization circuitry, begins as each exponent appears. The procedure is to examine all exponents sequentially to choose the smallest, assuming that one is less than a threshold test value which is entered first, and to retain at all times the lesser value of two sequentially examined exponents. The number of the exponent retained specifies the class of the input vector. This is available to be displayed, printed on film, or supplied to the computer for logging or subsequent processing.

The cascades may also be used to process an increased number of distributions for a lesser number of channels. Thus for a 4-channel source of such as LANDSAT, the operations of the various arithmetic units may be time-shared to provide 16 instead of eight class decisions for eight channels.

Clock Card

This card has two main purposes, to generate timing pulses to control the 16 steps in the processing of a data vector, and to control the loading of the RAMs during system setup. A block diagram is shown in Figure 30.

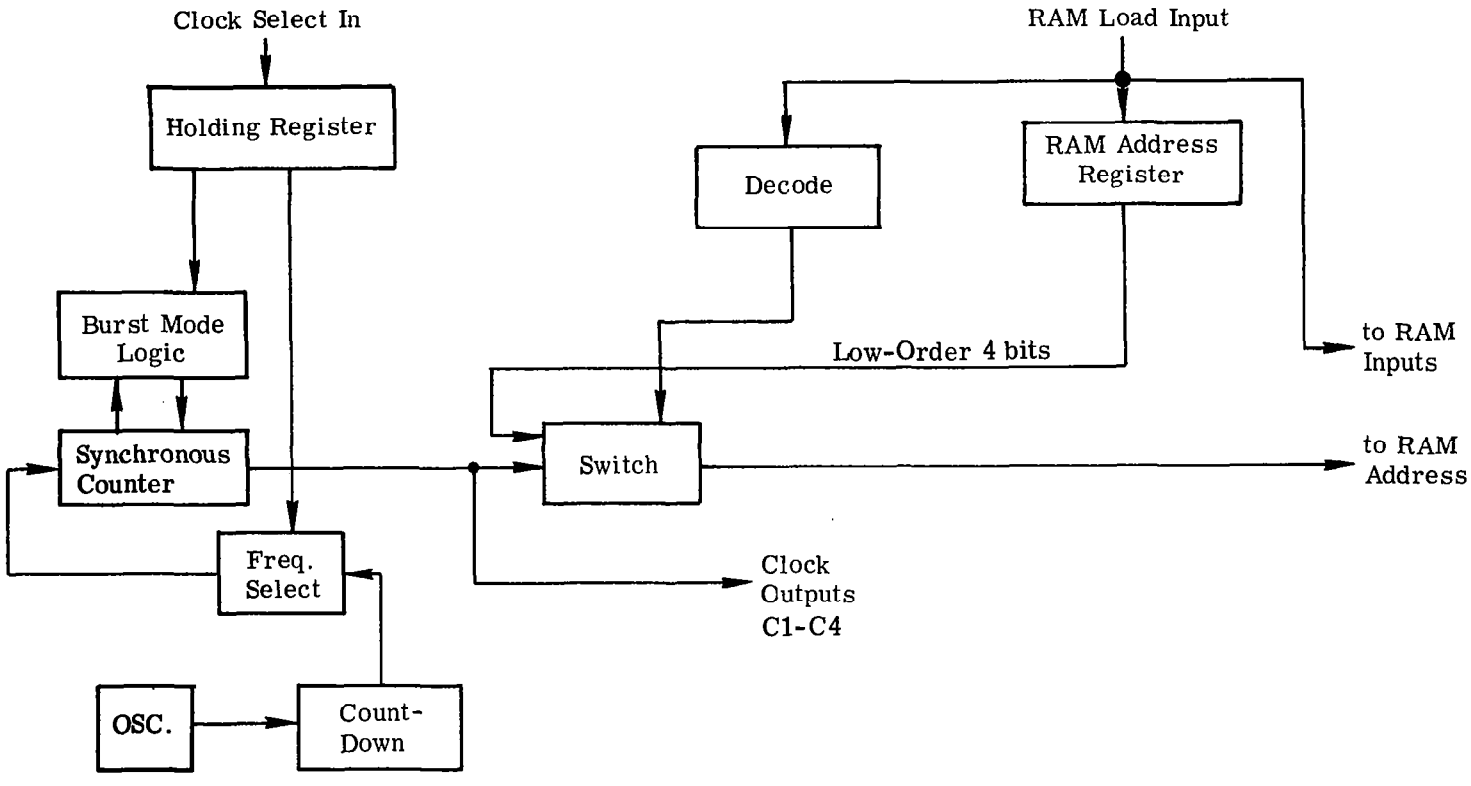


FIGURE 30. BLOCK DIAGRAM OF CLOCK CARD

The timing pulses, or clock pulses, are derived from a master oscillator. A countdown is program-selectable in six steps providing clock pulse C0 at rates from 3.2 MHz to 100 kHz. Other clock signals are labeled C1 to C4 and $\overline{C0}$ to $\overline{C4}$.

The signals C1 to C4 are used for classifier control, whereas C0 is used for loading latches at the end of a clock period. The circuitry can operate in a burst mode with or without reset to zero. The burst mode is used when operating the system using data coming from the computer. The clock can be set to count up from zero to 127. Sixteen steps are required to process a data vector. These are provided by setting the clock counter at 15, thereby allowing the sixteen steps 0 through 15. The clock is started by the computer when the computer has loaded the data vector into the second rank of the data register (latch "B" on the hybrid cards). The Classifier processes this vector and stops until the computer has loaded a new data vector. If the computer loads a new vector first, however, the logic waits for the clock to finish before a new start signal is accepted by the clock.

Control of RAM loading during system setup is accomplished by means of logic on the clock card. Timing pulses and routing control are provided (see Figure 30) which transmit the contents to any selected RAM by replacing the clock pulses C1 through C4 with the low-order four bits of the address word. The addressing and loading of RAM constants is accomplished by the following sequence:

(1) SEND WORD 1, DATA READY AND END CYCLE
(2) UPON RECEIPT OF CYCLE REQ. SEND WORD 2
(3) UPON RECEIPT OF CYCLE REQ. SEND NEW WORD 1
(4) REPEAT 2

The process stops when the word count register in the DR-11B increments to zero

The two-word transfers of RAM coefficients are

	CODE	BAY	RAM	ADD
WORD 1	0000 0	DDD	DDDD	DDDD
		MSB	MSB	MSB

	CODE	RAM CONSTANT
WORD 2	0001	DDDDDDDDDDDD
		MSB

Note that D is DATA -0 or 1, as the case may be; BAY is binary-coded 0 to 7 (only 0-3 are used); and RAM is binary-coded:

0 = MEANS
1 = VARIANCE
2 = MTX 0
3 = MTX 1
4 = MTX 2
5 = MTX 3

6 = MTX 4
7 = MTX 5
8 = MTX 6
9 = MTX 7
10 = k2 or CHI (k2 in Bay 2; CHI in Bay 3)
11 = LN (Bay 3 only)
12 = SHIFTER

ADD is binary-coded 0-15 for internal address of a RAM.

Diagnostic/Output Card

The diagnostic/output (D/O) card is the prime output port for classifier-generated signals. Inputs to the D/O may originate on the recognition card (5-bit material code and/or 16-bit exponent value) or at any access to the classifier diagnostic bus. Outputs from the D/O card are presented as a 16-bit word to the computer interface and as an 8-bit word to a display interface.

The D/O card has two input registers for holding the values of material and exponent from the recognition card and one register for holding the diagnostic value (see Figure 31). Contents of these three registers are combined and switched to the two output ports under control of the classifier clock and a computer-selected command word.

The command word establishes the desired output formats and selects a time and place for sampling the diagnostic line. The command word is stored in a register and will command the same outputs to repeat twice each classifier cycle until a new command is sent.

Available output formats are as follows.

(1) At the computer interface:
  (a) the full 16-bit exponent value
or (b) eleven most significant exponent bits plus a 5-bit material code
or (c) 12-bit diagnostic plus 4 zeros
or (d) a composite word, presented once each cycle, consisting of two 5-bit material codes

(2) At the display interface:
  (e) the eight MSBs of the exponent
or (f) the eight MSBs of the diagnostic
or (g) 5-bit material code

Any combination of the above computer and display outputs may be selected by the appropriate command except that (d) and (e) are not available simultaneously. The outputs are specified by the codes in Table 6.

2.5.2 THE PREPROCESSOR

The Preprocessor section of the MIDAS pipeline consists of four bays of wire-wrap cards. A diagram of the chassis layout and position of the various cards is shown in Figure 32. There

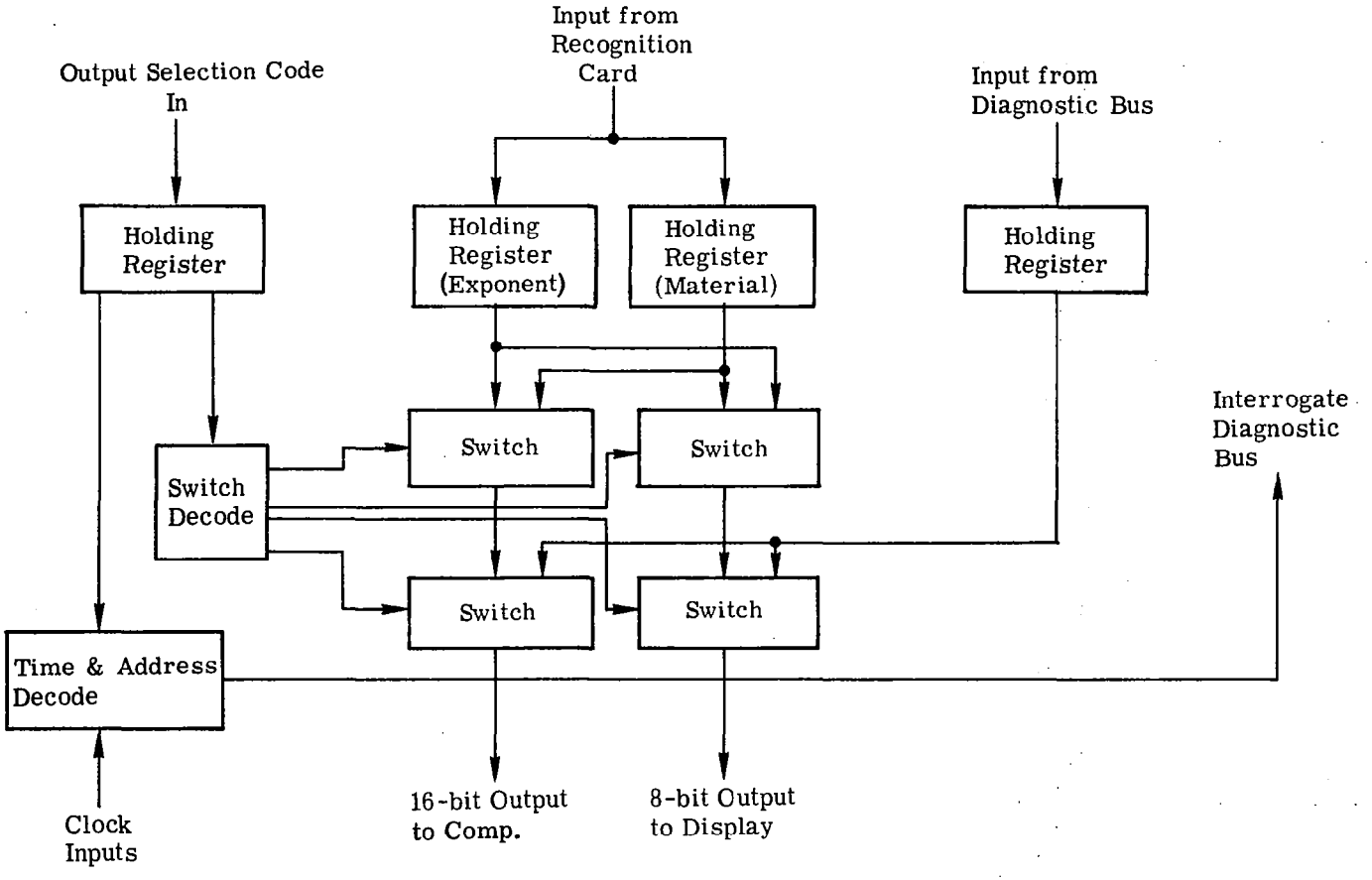


FIGURE 31. BLOCK DIAGRAM OF DIAGNOSTIC/OUTPUT CARD

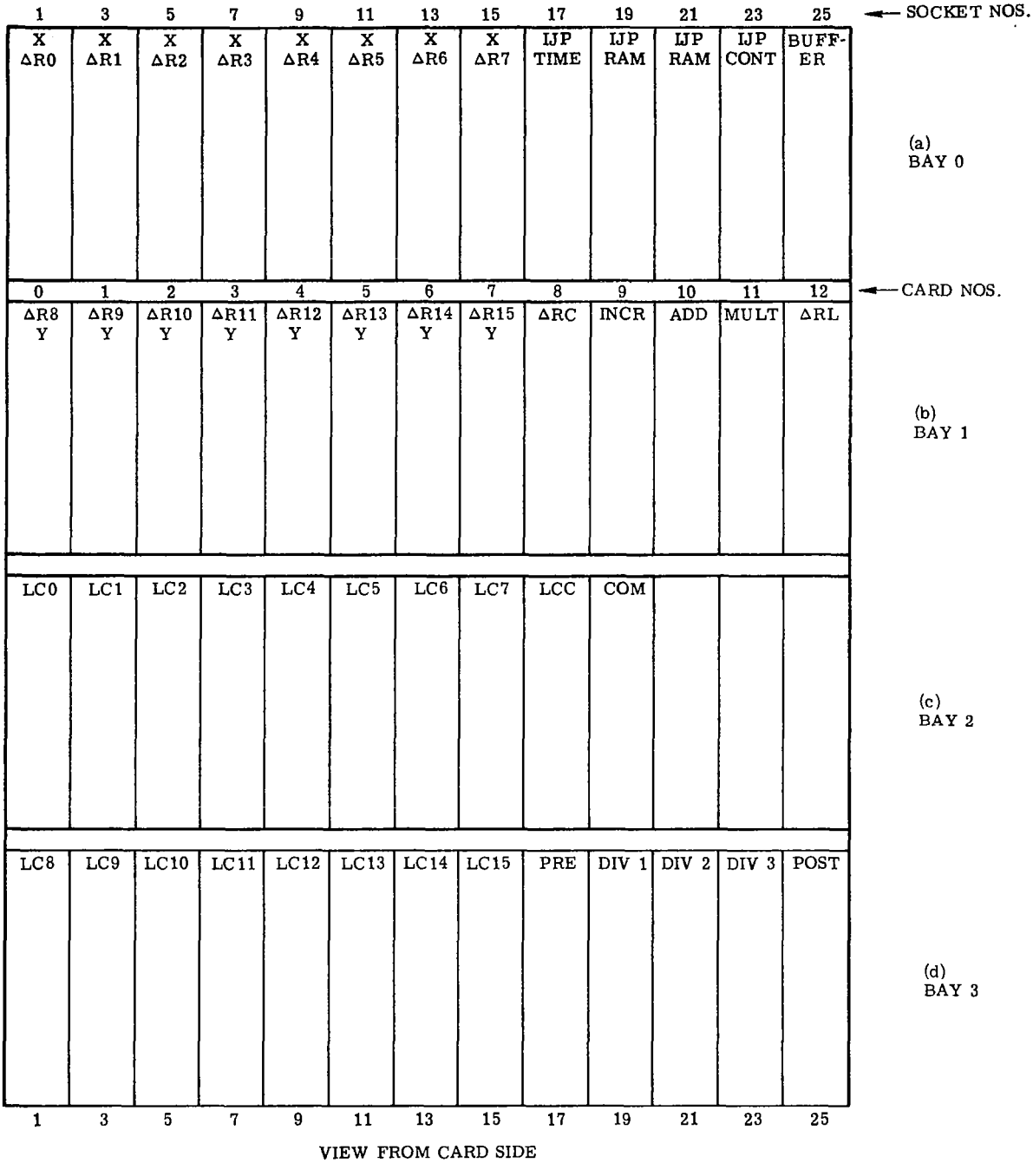


FIGURE 32. LAYOUT OF PREPROCESSOR CHASSIS

TABLE 6. CODE SELECTION FOR THE DIAGNOSTIC/OUTPUT CARD

CODE	TO COMP		TO DISPLAY
10001	000 MAT (4)	000 MAT	000 MAT
01001	000 MAT (4)	000 MAT	DIAG (1)
10101	11-BIT EXP (2)	MAT	000 MAT
01101	11-BIT EXP (2)	MAT	DIAG (1)
11101	11-BIT EXP (2)	MAT	EXP (3)
10011	DIAG	0000	000 MAT
01011	DIAG	0000	DIAG (1)
11011	DIAG	0000	EXP (3)
10111	EXP		000 MAT
01111	EXP		DIAG (1)
11111	EXP		EXP (3)

NOTES:
(1) 8 most significant bits of 12
(2) 11 most significant bits of 16
(3) 8 most significant bits of 16
(4) This output appears every second process cycle; all other outputs appear every cycle.

are a total of 52 wire-wrap-card positions that make up the Preprocessor chassis. However, not all of them are filled. Figure 32 shows 16 positions for "Delta RAM" cards designated ΔR. Only four of these cards have been constructed. The significance of this will be discussed later. Of the 52 card positions there are 12 different card designs. These are: (1) Delta RAM (ΔR), (2) Delta RAM control (ΔRC), (3) Increment (INCR), (4) ADD, (5) Multiply (MULT), (6) Delta RAM Load (ΔRL), (7) Linear Combination (LC), (8) Linear Combination Control (LCC), (9) Command (COM), (10) Prenormalization (PRE), (11) Divide (DIV), and (12) Postnormalization (POST). These card types are discussed in the following subsections.

Delta RAM Card

The Delta RAM card is basically a 4-bit by 256-word RAM repeated eight times on each card. Since the angle correction function is generated by increments at equally spaced pixels or point numbers, the angle correction function in the one-dimensional case can be thought of as a staircase type function across the scan line or "stripes" down the flight path. One card then can provide 256 increments or stripes for correcting eight data channels. Furthermore, the 4-bit word provides the 2-bit increment for both A and B functions. It takes two cards to provide the increment values for all 16 channels. There is provision in the Preprocessor chassis for 16 Delta RAM cards: up to eight cards to provide the active increment values, and up to 8 cards that are available to have updated values read into them. The updated cards can then be switched to the active state, thereby making the other eight cards available for updating. The two sets of cards are termed x and y sets respectively, and either can be used in the one-dimensional case. Not all eight cards need be used for this incremental update scheme. If eight or fewer data channels are used for input to the preprocessor only four cards are needed per set since one card contains eight RAMs; since each card contains 256 increments the maximum number of increments is 1024. If fewer than 1024 increments are needed then the number of cards may be less. For example, if eight or fewer channels and 512 increments are used, then only two cards per set are needed. The system as presently constructed has four Delta RAM cards. The integrated circuits used are 74S206 1-bit × 256-words with three-state outputs. The outputs of all x cards and all y cards are bussed together to feed the update value to the increment card. There are no diagnostic outputs on these cards.

Delta RAM Control Card

The Delta RAM control card has two major functions. The first function is to store the initial values A_o and B_o for both the active and updated function. One set of RAMs provides the initial value to the increment card when an end of line is received, thereby conditioning the angle correction function to be ready at the start of the next line. A selector switch provides the proper output of the ΔRAMs designated as active. The second major function is to select the proper ΔRAM card to put the correct increment value on the output bus. Since there are eight RAMs per card and up to 16 cards, selection circuitry for one of 128 is provided for. The circuitry disables the select for the RAMs being updated, and steers the decoded address to the 64 active RAMs.

Increment Card

Basically, the increment card performs the addition of the current value of the function to the increment and stores this updated value as the current function. Because of the pipeline nature of the MIDAS system where it takes 16 clock pulses to enter a pixel, 16 values of the current function must be stored. This is done in two 12-bit by 16-word RAMs. The output of one of the RAMs is selected through a switch as the current value of the angle correction function. This output is also fed to a switch, the use of which will be explained shortly, that normally passes the current value of the function on to the input of an adder. The other input to the adder is the increment value originating from the Delta RAM cards. Since the increment value consists of two bits, it can take on a value -2, -1, 0, +1. When the command to update is given (not necessarily every pixel) the output of the current value is added to the increment value, and the result stored in the other RAM on a 16-channel basis. Repeated values will occur if there are fewer than 16 channels of incoming data. When this RAM is loaded it becomes the angle correction value for the next pixel. The switch at the input to adder has the current value RAM as one input and the initial value as the other. At the end of the scan line this switch feeds the initial value to the adder, the other adder input being zero. The initial value drops through the adder and is loaded into the RAM. After this operation is completed (about 10 μsec) the angle correction circuitry is primed for another scan line. This card has two identical sets of the circuitry described above, one to provide the additive angle correction value $A(\phi)$, the other to provide the multiplicative angle correction value $B(\phi)$.

Add Card

The add card performs the addition of the input data with the current value of the angle correction function $A(\phi)$ which is provided by the increment card. This card has two other functions: First, the diagnostic bus terminates on this card. A 74174 latch stores the value on the diagnostic bus and feeds it to the computer. There are three diagnostic input ports on this card: (1) the input data, (2) the angle correction value $A(\phi)$, and (3) the output of the adder. Second, the overflow bits are brought in and put into a shift register at the appropriate place. Thus, when a data value appears at the output of the preprocessor, the output bit of the shift register indicates if this is valid data. Both the data values and the overflow bits are fed to the classifier.

Multiply Card

The multiply card performs a 10-bit by 10-bit multiply of the output from the add card and the multiplicative angle correction value $B(\phi)$ from the increment card. There are two diagnostic ports: $B(\phi)$ and the output of the multiplier. Three high-order bits of the multiplier are tested for overflow and an overflow bit is generated if the multiplier output is too large.

Delta RAM Load Card

The Delta RAM load card is basically a switching and selection circuit to load the Delta RAMs. The loading takes place via a DMA DR-11B interface shared with the Inkjet Plotter

Control Cards. The loading of a constant into the Delta RAMs from the computer is a two word transfer; the first word is a control word containing the Delta RAM load code (4-bits) and 12 bits of RAM address. These 12 bits are stored in a register. The second word contains the constant to be loaded into the location addressed by the 12 bits of the stored address. The RAM load pulse is steered to the proper RAM by decoding the four high-order bits of the 12 while the remaining eight bits serve as the input to all the RAMs' address lines. While one set of RAMs are being loaded the other set is being used to provide the increment values for function update. The address lines of these active RAMs are not under computer control, but have been switched to the output of a counter. This counter increases by one every n pixels where n has been loaded into a RAM and has the range $1 \leq n \leq 63$. Whenever the count increases by one, the active Delta RAMs supply a new increment value that is immediately used to update the angle correction functions as explained in the Delta RAM control card section. There are no diagnostic outputs from this card.

Linear Combination Card

The linear combination card is very similar to the matrix multiplier card in the classifier section of the pipeline. The multiplier is a 6-bit by 12-bit multiplier. The linear coefficient values are the 6-bit 2's complement numbers into the multiplier. The scaling and overflow bits tested on this card can be seen in Figure 19. There are two diagnostics on this card, one to read back the coefficients stored in the RAMs, and the other to provide the card output. There are sixteen of these cards, made identically. Eight of the cards have their outputs bussed together to form the numerator, and the remaining eight are bussed to form the denominator for the succeeding ratioing section. Any card can be accessed at any of the 16 clock times since the resultant summation is stored in a three-state holding register for the entire time a new summation is being formed. The card selection is accomplished by decoding values stored in a RAM which is addressed sequentially by the clock. At this point in the pipeline the number of data channels is reduced to a maximum of eight.

Linear Combination Control Card

The linear combination control card contains the RAM and selection circuitry for selecting which LC cards will provide the numerator and denominator inputs to the ratioing section of the preprocessor. In addition it contains a 12-bit by 16-word RAM to provide up to 16 different constants to the denominator bus. Selecting this RAM as data on the denominator bus substitutes a constant divisor instead of the ratioing of data channels. The output of this RAM can be intermixed with the LC denominators. There are two diagnostic ports on this card, one to read the denominator bus, and the other to read the contents of the linear combination card selection RAM. A large portion of the diagnostic bus output selection is decoded on this card with the control being fed to the various cards in the Preprocessor.

Command Card

The command card receives commands from the computer via a DR-11B DMA. This DMA device is a separate DR-11B from that used for Delta RAM load; it is also used to transmit

commands to the classifier section. The main purpose is to load the RAMs in the Preprocessor other than the Delta RAMs. This loading is done in a similar fashion to the Delta RAM load card described earlier. To load a RAM requires a two word computer transfer; the first, which contains the address is stored, while the second word, containing the data of the RAM to be loaded, is transferred. The diagnostic select code is also stored on this card and passed on to the linear combination control card. This was done because the fanout for RAM load selection and the fanout for Diagnostic selection required more pins than were available. The clock lines from the classifier are brought in and buffered for distribution to the Preprocessor cards. There are no diagnostic ports on this card.

2.5.3 THE PIPELINE DIVIDER

The MIDAS Preprocessor requires a division unit for the enhancement or normalization of incoming data vectors. The following describes both the process of selecting a suitable design and the actual implementation. The implementation is discussed both from the standpoint of the arithmetic theory of the division algorithm and of the hardware realization.

Specification

Some of the design specifications for the divider include

(1) a 12-bit dividend (2's complement)
(2) a 12-bit divisor (2's complement)
(3) a 9-bit quotient (2's complement)
(4) the quotient should be correct for arbitrary signs of both the dividend and divisor
(5) a 6-bit scalar (2's complement) which adjusts the quotient by shifting it right or left
(6) a 1-bit division overflow
(7) compatibility with the timing and electrical characteristics of the rest of the processor

Design Alternatives and Selection Criterion

It was immediately decided that the 300-ns clock period would require the division process to be subdivided and pipelined. The design questions which immediately arose were

(1) Which algorithm should be used for the division?
(2) Which chips should be selected to implement the division?
(3) How can the division process best be subdivided compatible with the 300-ns clock?

These questions cannot be treated sequentially. Rather, they interrelate to form one complex design problem. For example, the availability of chips strongly influences the selection of the algorithm.

The selection criterion was basically to minimize the cost of implementation. However, this did not imply minimizing chip cost. The cost of chips is a relatively small contribution to the overall design cost of the divider. It was decided that a better measure of cost was the number of cards required to implement the divider. Since the physical placement of chips on a card had already been standardized, the following were considered the major design objectives.

(1) Minimize the total number of chips where 24-pin chips are weighted equal to 2 1/2 16-pin chips.

(2) Keep the design simple to minimize the number of errors and debugging cost

Three division algorithms were considered with respect to the criterion described above.

Binary Restoring Division

Restoring division is the binary equivalent of decimal division as taught in elementary school. Normally the absolute value is taken of both input operands and their signs are used to give the quotient its correct sign. Thus, in the following discussion the dividend (numerator) and divisor (denominator) are both assumed positive. The quotient digits are selected from 0 or 1; thus only positive quantities can be represented. The partial remainder must remain positive at all times if the quotient is to be representable with digits of positive magnitude.

The dividend (numerator) becomes the first partial remainder. Each division step, which generates one quotient bit, proceeds as follows. (1) The divisor (denominator) is subtracted from the partial remainder. (2) If the result is positive, it is shifted left one bit to form the new partial remainder; the quotient bit is 1. (3) If the result of the subtraction is negative, the old partial remainder is shifted left one bit to form the new partial remainder, and the quotient bit is 0. Some of the advantages and disadvantages are listed below.

(1) The signs of the input operands must be handled by separate logic.

(2) The quotient is generated in a 2's complement form.

(3) The scheme is simple in concept.

(4) The weighted chip count was the highest of three approaches.

Binary Non-restoring Division

In non-restoring division, while the sign of the dividend is arbitrary, the sign of the divisor must be positive. The quotient digits are selected from $\{1,\bar{1}\}$ where $\bar{1}$ indicates a negative 1. Since the quotient representation allows negative quotients, the restriction that the partial remainder remain positive is now removed. The dividend again becomes the first partial remainder. Each division step computes one quotient bit and proceeds as follows. (1) The sign of the partial remainder is used to determine the quotient digit. A positive partial remainder generates a 1 while a negative partial remainder generates a $\bar{1}$. (2) The divisor is multiplied (an adder/subtractor is used) by the quotient digit and subtracted from the partial remainder. (3) The result is shifted left one bit to become the new partial remainder. Some of the advantages and disadvantages of the non-restoring scheme are

(1) only the sign of the divisor need by positive

(2) the quotient may be converted to 2's complement trivially

(3) the weighted chip count was significantly lower than that for restoring division

(4) chips were available to implement the division as a very regular array of adder/subtractors

Radix 4, Redundant Division

In the radix 4 scheme, the quotient digits are selected from $\{2,1,0,\bar{1},\bar{2}\}$. Each quotient digit represents 2-bits of result. Since there are five members in the quotient digit set, the representation is somewhat redundant. This allows the determination of the quotient digits from truncated versions (approximations) of the partial remainder and divisor. The scheme automatically takes care of input operands of arbitrary sign.

A division step proceeds as follows. (1) Some high-order bits of the partial remainder and divisor are used to generate the next quotient digit by table look-up (a ROM is employed). (2) The divisor is multiplied by the selected quotient digit (requiring an adder/subtractor and a 1-bit shifter) and subtracted from the partial remainder. (3) The result is shifted left 2 bits to form the new partial remainder. Some advantages and disadvantages of this scheme are listed below.

(1) The signs of the input operands are unrestricted
(2) The scheme for conversion of the quotient to 2's complement is non-trivial.
(3) The weighted chip count was about equal to that for non-restoring division.
(4) This scheme is significantly more complex conceptually than non-restoring division.
(5) This scheme requires the burning of ROMs.

Selection

Based on our design criterion, non-restoring binary division was selected for implementation. A major factor contributing to this selection was the availability of the Fairchild 9340 adder/subtractors with built-in carry lookahead. This chip is ideally suited for implementing the division array.

Implementation

The divider was constructed on five cards, each containing one step of the five-step divide pipeline (see Figure 33). Thus, in each of the five cards, data which was latched on the previous card flows through combinatorial logic and is then latched on the current card. All latches are tied to a common clock. The first divide card is the pre-normalization card. The second, third, and fourth are identical divide cards, each of which produces four quotient bits. The fifth card is a post-normalization card.

Pre-normalization

There are three inputs to the pre-normalization card. The inputs are all 2's complement numbers as follows:

(1) A 12-bit dividend
(2) A 12-bit divisor
(3) A 6-bit scalar.

While the dividend and divisor are held on inputs to the pre-normalization card for one clock period, the scalar value is read from a 16-word RAM (see Figure 34) on the pre-normalization card, capable of storing one scalar for each of 16 distinct Preprocessor clock cycles.

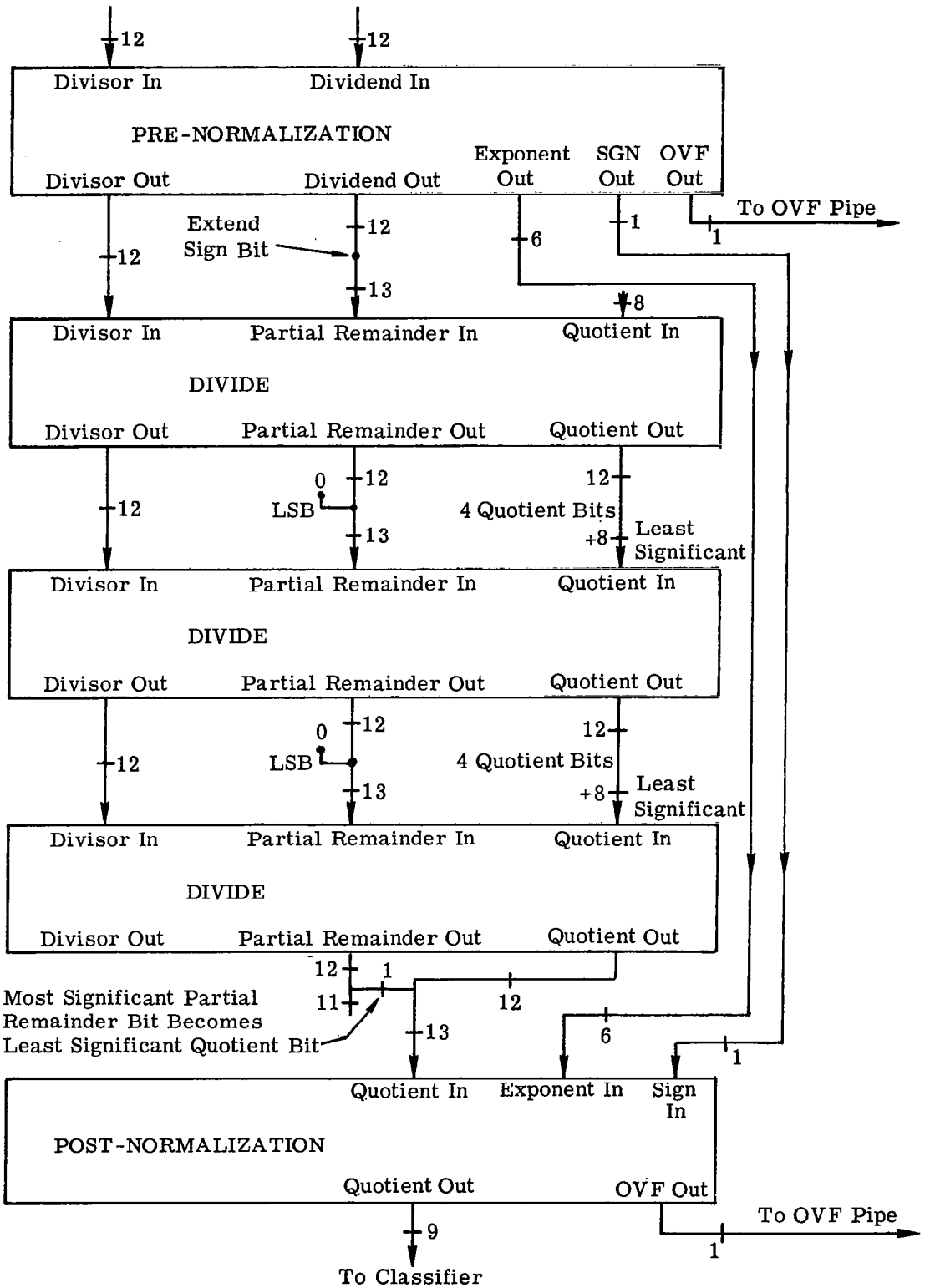
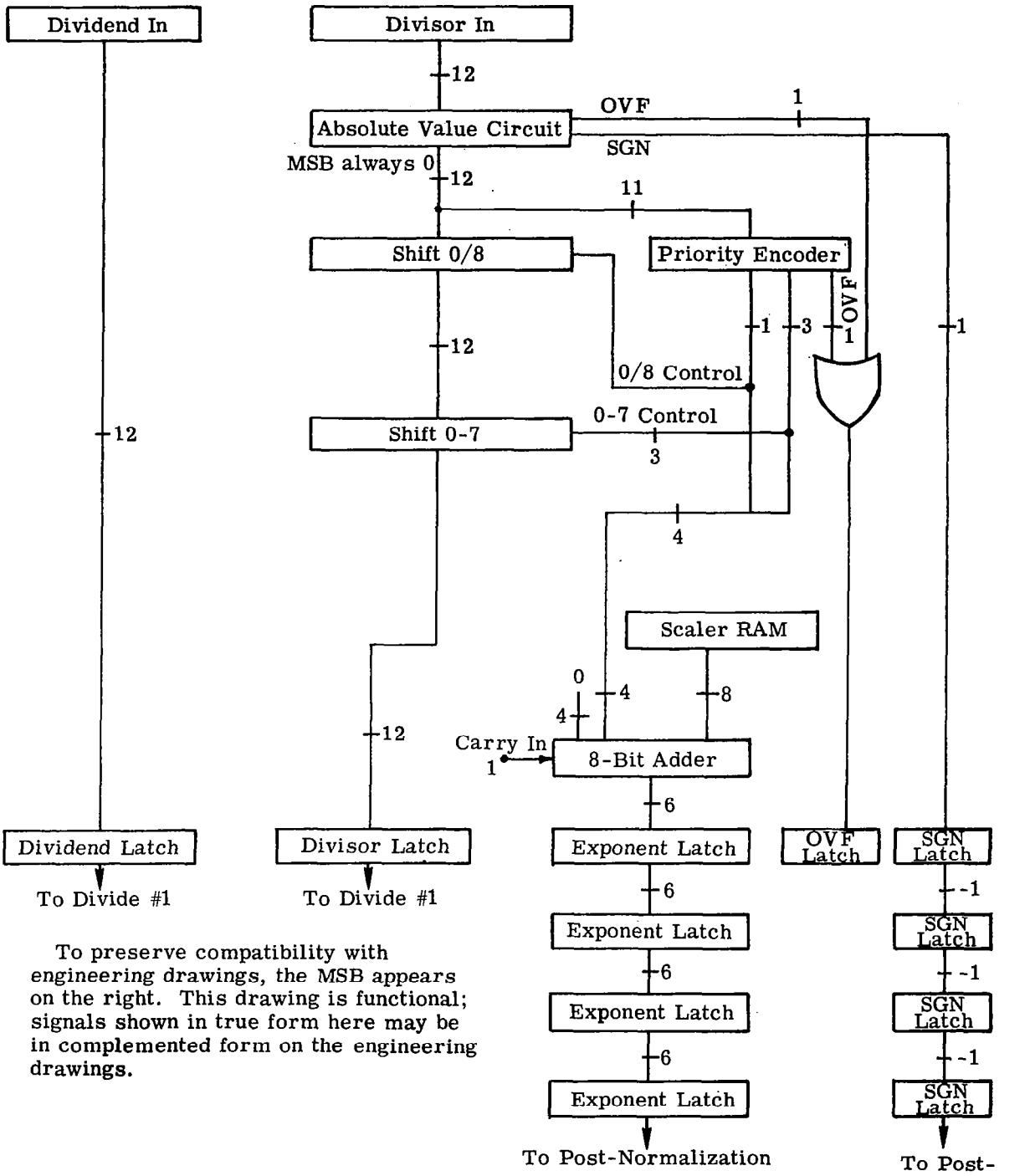


FIGURE 33. CARD INTERCONNECTIONS IN THE PREPROCESSOR RATIOING SECTION



To preserve compatibility with engineering drawings, the MSB appears on the right. This drawing is functional; signals shown in true form here may be in complemented form on the engineering drawings.

FIGURE 34. PRE-NORMALIZATION

The inputs are to be interpreted as follows: The nine bit quotient to be computed is

$$q.qqqqqqqq = nnn.\ nnnnnnnnn/ddd.ddddddddd \times 2^{ssssss}$$

where q represents a quotient bit, n represents a dividend bit, d represents a divisor bit, and s represents a scalar bit. The binary points are as indicated. Thus, the scalar adjusts the magnitude of the quotient by shifting it right or left. Note that the scalar may take on a negative value.

One can see, in the block drawing of the pre-normalization card (Figure 34), that the dividend is merely latched on the card and passed onto the first divide card (Figure 33). The divide array handles any dividend input with no overflow. The dividend is latched on the pre-normalization card to keep the dividend in step with the divisor. On the other hand, the divisor must be normalized before entering the divide array to prevent overflow within the array. The normalization process consists of taking the absolute value of the divisor and shifting left until the bit immediately to the right of the sign bit is a 1. The correct quotient value is maintained by adding to the scalar value, the number of places shifted.

In Figure 34, one can see that the absolute value of the divisor is taken at the input of the pre-normalization card. This is done by wiring 3 Fairchild 9340's as a 12-bit adder/subtractor and using the complemented sign of the input to control the add/subtract function. The output of this absolute value circuit is positive (sign bit = 0) except for the case where the divisor was the largest negative number (100.000000000). This divisor has no absolute value and generates an overflow during pre-normalization. The sign of the divisor is saved and latched four times on the pre-normalization card before being passed to the post-normalization card (Figure 33). The four latches keep the sign in step with the quotient, which is latched on three divide cards before entering post-normalization.

The next step in pre-normalization is to count the number of high-order zeros (excluding sign bit) of the positive divisor. For example, if the absolute value of the divisor looks like 000.1XXXXXXXX (X indicates either 1 or 0) then, the number of high order zero's computed is 2. This operation is done by 2 Fairchild 9318 priority encoder chips. The result appears as a 4-bit binary quantity. A zero valued divisor cannot be scaled and causes another overflow condition during pre-normalization.

The 4-bit quantity discussed above is used to control a shift network which shifts the divisor left 0-15 places. This is more than sufficient to normalize any non-zero divisor. The shift process is done in two parts. In the first part, the most significant bit of the 4-bit shift count is used to control three Signetics 8233 2-input, 4-bit digital multiplexers. These multiplexers are wired to shift 0 or 8 positions with zero's injected in the low order end. The second part of the shift process is carried out by three Signetics 8243 scalar chips which are wired as a 12-bit, 0-7 position left shifter. The low order three bits of the 4-bit shift count control the amount of shift. This normalized divisor is now latched for use on the divide cards.

The pre-normalization card also performs an exponent computation which corrects for any shift done during divisor normalization. Let C be the number of places the divisor is left-shifted during pre-normalization. Let S be the scalar as read from the RAM. In order to maintain the correct quotient value, we can see

$$Q = (N/D) \times 2^{S} = (N/(Dx2^{C})) \times 2^{(S+C)}$$

Thus, a 6-bit exponent E = S + C is computed and used to scale the quotient. This exponent computation is done with two Texas Instrument SN74283 4-bit adders. Note that the carry into this adder is set so as to actually compute E = S + C + 1. We shall explain this +1 term later.

Division

The purpose of the pre-normalization card is to eliminate the possibility of overflow during the non-restoring division by properly scaling the inputs. Clearly, the largest quotient should be generated when the largest dividendand smallest divisor are used for inputs. Pre-normalization guarantees that the divisor must be greater than or equal to 010.000 000 000. The largest dividend is equal to 011.111 111 111 positive or 100.000 000 000 negative. Thus, the largest quotients which we would like to represent are: $(4-2^{-9})/2 = 2-2^{-10}$ and, -4/2 = -2.

The dividend (pre-normalization output) becomes the first partial remainder with no shift (see Figure 33). Thus, in the intermediate non-restoring quotient (the quotient as generated by the divide array), the high order digit has weight 1. The intermediate quotient looks like: r.rrr rrr rrr rrr where $r \in \{1,\bar{1}\}$. The range of representable quotients is from 1.111 111 111 111 = $2-2^{-12}$ to $\bar{1}.\bar{1}\,\bar{1}\,\bar{1}\,\bar{1}\,\bar{1}\,\bar{1}\,\bar{1}\,\bar{1}\,\bar{1}\,\bar{1}\,\bar{1}\,\bar{1}\,\bar{1} = -2+2^{-12}$. Thus, the largest positive quotient (computed above) is within the range while, the largest negative quotient is closely approximated. We hope that this will satisfy the reader that the division array does not overflow.

The division array is constructed on three identical cards each of which computes four quotient bits. The thirteenth quotient bit is taken from the sign bit of the last partial remainder. Each non-restoring divide step is performed as follows (see Figure 35). Three Fairchild 9340 4-bit carry lookahead adder/subtractors are interconnected to form a 12-bit unit. Four rows of adder/subtractors are used to compute the four quotient bits on a divide card. Consider the output (a partial remainder) of one row of adder/subtractors. The sign bit of the partial remainder is complemented to form a quotient digit. Thus, a negative partial remainder generates a 0 ($\bar{1}$) quotient digit while a positive partial remainder generates a 1. The partial remainder is shifted left one position (0 is injected into the LSB) before entering the next row of adder/subtractors. The quotient digit times the divisor is then subtracted from this shifted partial remainder to form the new partial remainder. This is done by wiring the sign bit of the shifted partial remainder to the add/subtract control line of the next adder/subtractor row.

Between divide cards, the shift of the partial remainder is wired on the back plane which permits constructing 3 identical divide cards. The dividend becomes the first partial remainder

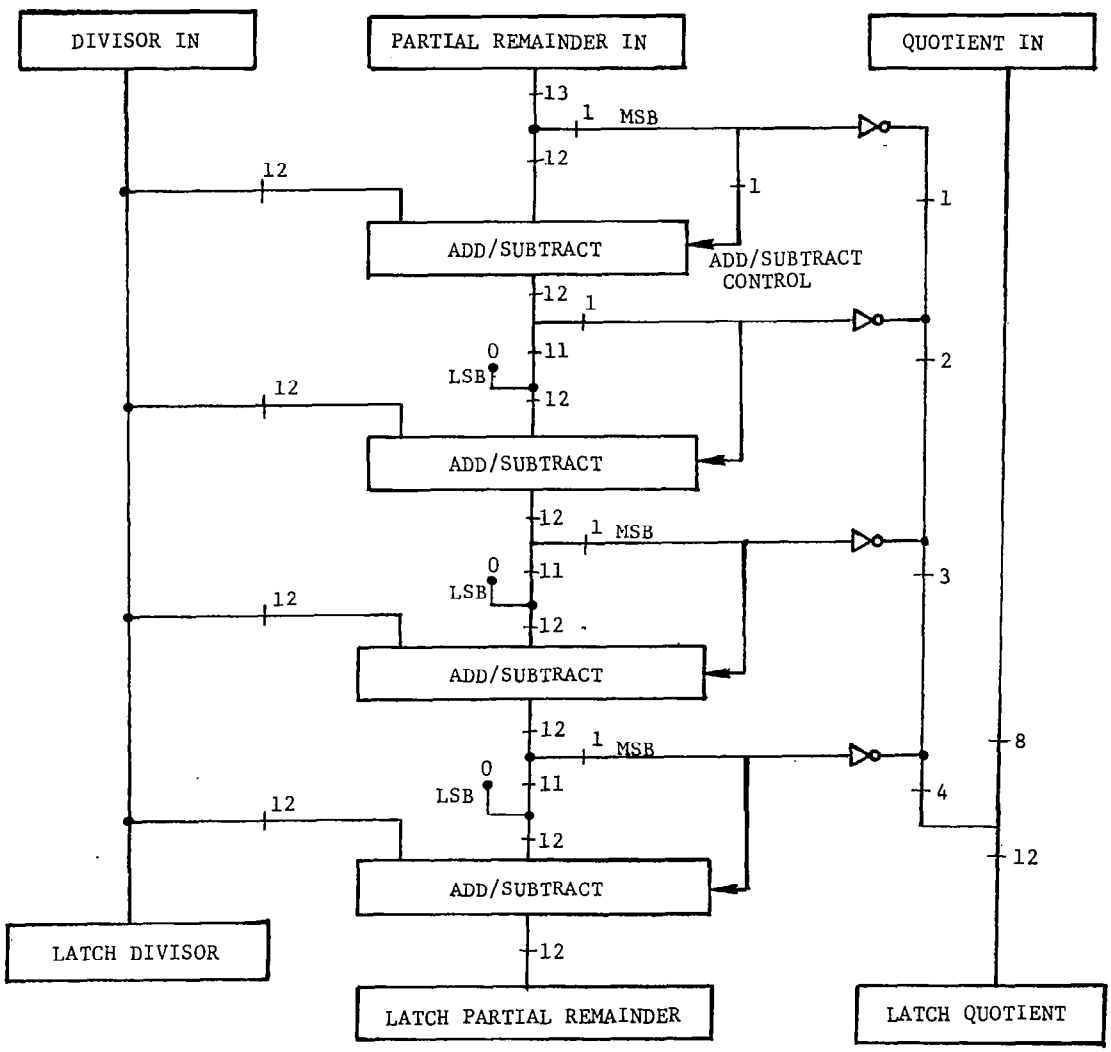


FIGURE 35. NON-RESTORING DIVIDE

with no shift, the sign bit is extended one position (see Figure 33). The partial remainder shift is wired between the first and second, and second and third divide cards.

Each divide card contains a 12-bit quotient latch. Between divide cards, a four bit left quotient shift is wired. The four quotient bits computed on a card are shifted into the low order positions of the 12-bit latch. Thus, at the third divide card, a full 12-bit quotient appears in the quotient latch. Again we mention that the 13th quotient bit is computed from the sign of the last partial remainder.

Post-normalization

Post-normalization begins with the conversion of the 13-bit non-restoring quotient to a 12-bit binary quantity with separate sign bit. Consider the non-restoring quotient R = r.rrrrrrrrrrrr where $r \epsilon \{1,\overline{1}\}$. Let A represent the weighted sum of all positive digits in the non-restoring quotient. Let B represent the weighted sum of the absolute value of the negative digits in the non-restoring quotient. For example, if R = $1.1\overline{1}1\overline{1}1\overline{1}1\overline{1}1\overline{1}1$ then A = 1.100110011111 and B = 0.011001100000. With these definitions of A and B, we can write the following equations:

$$R = A - B$$

$$2 - 2^{-12} = A + B$$

The second equation holds since the sum of the weight absolute value of all quotient digits is always $1.111111111111 = 2 - 2^{-12}$. From the equations above, we write:

$$R + 2 - 2^{-12} = 2A$$

$$R = 2A - 2 + 2^{-12}$$

$$R/2 = A - 1 + 2^{-13}$$

The addition of 2^{-13} affects the 14th digit position and can be ignored. We have:

$$R/2 = A - 1$$

Since, $\overline{1}$'s in the non-restoring quotient are represented by 0's, A is the non-restoring quotient treated as a binary number. To compute R/2, all we have to do is add -1 = 1.000000000000 to A. This corresponds to complementing the sign bit (there can not be any overflow).

Note that we have computed R/2 above. In order to scale the quotient up by a factor of 2, a carry is injected into the exponent computation on the pre-normalization card.

The post-normalization card (see Figure 36) first inverts the sign bit for reasons described above. Then the absolute value is taken of the 13-bit 2's complement quotient. Since the sign of the result is guaranteed to be zero, this process can be carried out by a 12-bit adder/subtractor group. Overflow occurs whenever the absolute value is taken of the most negative quotient. The quotient sign is saved for later use.

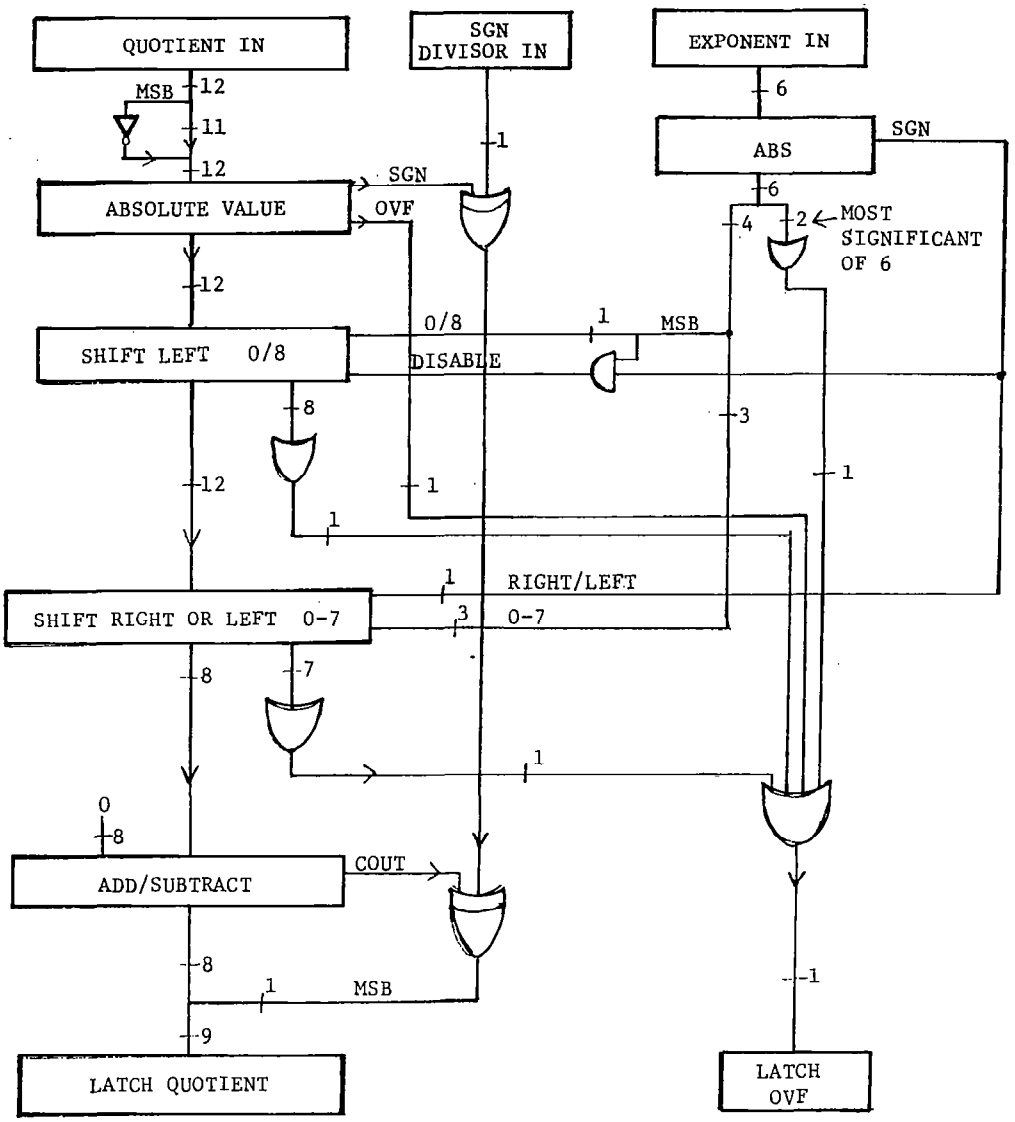


FIGURE 36. POST-NORMALIZATION

The final step of post-normalization is to compute the correct quotient from the non-restoring quotient, the exponent, the sign of the divisor, and the sign of the non-restoring quotient. The magnitude of the final quotient is computed by the scaling operation:

$$Q = R \times 2^{E}$$

This is then either complemented or not, according to its proper sign.

The exponent E is a 6-bit 2's complement number. An 8-bit adder/subtractor is used to convert E to sign and magnitude form. The sign will be used to control the direction of a shift while the magnitude will control the shift distance.

We are computing a 9-bit quotient. However, the sign is carried separately so there are only 8-quotient bits. These eight-magnitude bits are scaled by the exponent as follows: Again, the shifting is done in two stages. The first stage shifts 0 or 8 positions left. A shift right of eight (or more) positions leaves a zero value for the 8-bit quotient magnitude. Thus, the right shift is done by disabling the shifter. The left shift is done by five 2-input 4-bit digital multiplexers (SN75157). Two of the multiplexers are used to compute the eight bits shifted out of the shifter. These eight bits are or'ed into the overflow. The other three multiplexers shift the 12 quotient bits 0 or 8 places left.

The second stage of the shifter consists of three 8243 scalar chips wired to shift 12 input bits 0-7 places right or left onto an 8-bit output bus. One more scalar is used to compute the bits shifted out on a left shift. These are or'ed into the overflow. These scalars are controlled by the sign of the exponent (right/left) and the three low-order bits of the exponent magnitude (0-7). The 8-bit output of this shifter has the properly scaled 8-bit quotient magnitude (in complemented form).

The sign of the divisor and the sign of the non-restoring quotient are now exclusively or'ed to compute the final quotient sign. The 8-bit quotient magnitude along with an implicit leading 0 sign bit is added to or subtracted from a 9-bit zero according to the final quotient sign. This 9-bit operation is carried out by two 9340 adder/subtractors and an exclusive or which adds the quotient sign to the carry out of the adder/subtractors. Note that the quotient magnitude is inverted (it was in complemented form) in this same step. There is no possibility of overflow and, the result is the desired 9-bit 2's complement quotient.

Conditions for Division Overflow

The conditions which lead to division overflow can be listed as follows.

(1) If the divisor is the largest negative number (100.000000000), it cannot be complemented to form the absolute value.

(2) A zero-valued divisor cannot be properly normalized.

(3) If the non-restoring quotient is the largest negative number ($\bar{1}.\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}$), overflow occurs during the process of converting to 2's complement and taking the ab-

solute value. In order for this condition to occur, the dividend must be the largest negative number (1.00000000000) while the normalized divisor must be 010.000000000.

(4) If the computed exponent has absolute value greater than or equal to 16, the post-normalization shift network cannot properly scale the quotient.

(5) Any non-zero high-order quotient bits which are left shifted out of the quotient during the post-normalization scaling operation will result in an incorrect quotient value.

Simulation

The entire division process was simulated in detail in APL. The division algorithm is somewhat complex and difficult for the design engineer to grasp as a whole. The simulation is an effective tool for establishing that the result is indeed the quotient of the dividend and divisor inputs. Thus, the simulation establishes some confidence in the design. In addition, the simulator is useful in testing various overflow conditions and sign processing.

A number of errors were uncovered using the simulation long before the wirewrap was done. Hence, it is valuable in reducing the problems encountered during the logic debugging phase of the design process. A listing of the APL simulator and a trace of a sample simulation run are included in the Appendix.

2.6 RAMTEK COLOR DISPLAY SYSTEM

The color display system provides a means of displaying imagery on a three-color CRT. This imagery may be either unprocessed or processed data with which the operator may interact to designate areas for analysis or processing.

The display uses MOS storage for screen refresh and allows display of 512 by 512 elements on the screen at 5 bits (plus an overlay bit) per scene element. Each 5-bit element may be translated into three 4-bit signals by table-look-up memories. The overlay channel is employed to designate points as a cursor or to indicate boundary locations in designating fields for analysis.

A second black and white CRT display is employed to present the menus for control of the system. This CRT normally displays the overlay, or sixth, bit of the display element.

Control is provided by an alphanumeric keyboard and track ball which controls the cursor location.

Interfacing to the PDP-11/45 is provided by a standard DR-11B connected to the display and trackball, and a DL-11C connected to the keyboard.

2.7 INKJET PRINTER

The inkjet printer subsystem consists of two units, the printer and its control interface to the PDP-11/45 Unibus.

2.7.1 PRINTER

The inkjet printer is the device chosen for fast, hard-copy, color output for the MIDAS system. This printer allows the user to obtain a pictorial copy of the data, unprocessed or processed, in about five minutes with a picture element size of 0.2 mm × 0.2 mm (0.008 × 0.008 in.) for a picture size up to 11 1/2 × 11 1/2 in. (up to 1430 × 1430 elements per sheet). Each picture element is printed by three separate ink-jets using three color-negative primary colors (magenta, cyan and yellow). Each primary color may be controlled in its density over approximately 40 quanta at each element printed, affording a color range for each element of approximately 40^3 (64,000) separate colors.

The printer operates at a rate of approximately 50 μsec per picture element, printing a line in 66 msec (15 rev/sec) on paper held on a rotating drum. A carriage holding the three jets is moved horizontally along the drum by a helical lead screw driven by a stepper motor. The lead screw pitch is 2 mm and advances a minimum increment of 0.01 mm per motor step. Normal operation of 0.2 mm line spacing requires 20 steps per revolution and may be given continuously during printing or between lines.

The ink jets are controlled during printing by electrostatic gating of the inkjet using pulse duration modulation. Each jet produces a stream of droplets at a rate of approximately 10^6 drops per second and has a scattering distribution such that a pulse of up to 50 drops spreads over a spot of about 0.2 × 0.2 mm area. Control of element density is thus obtainable by gating a packet of droplets onto the drum within the 50 μsec pixel interval. Droplets are gated by placing a charge, either negative or zero, on each droplet by applying a voltage to the nozzle assembly. The stream is directed toward the drum through a pair of charged deflection plates. It passes through an aperture to the drum when the droplets are uncharged or is deflected into a catcher when the droplets are charged.

2.7.2 CONTROL INTERFACE

The Control Interface which connects the printer to the PDP-11/45 Unibus consists of two parts: (1) a specially designed section for electro-mechanical control of the printer and (2) a DEC direct memory access, DR-11B custom user interface for transferring digital control and data signals to the special section. The DR-11B was chosen because rapid transfer of data is required; prior system use and software support made implementation straightforward.

The special purpose control section is shown in block diagram form in Figure 37. There are ten signals passed between the printer and this unit. These are: Power On, Trigger, Left Margin, Right Margin, Drum Motor On, Stepper Motor Step, Stepper Motor Direction, Ink Color 1 On, Ink Color 2 On, and Ink Color 3 On. The first four signals originate at the printer while the other six are control signals from the computer interface. Three of the printer signals are indicators: power switch on, print head at right or left margin. The trigger signal indicates that the drum is in the proper position to start the printing of a line. The drum

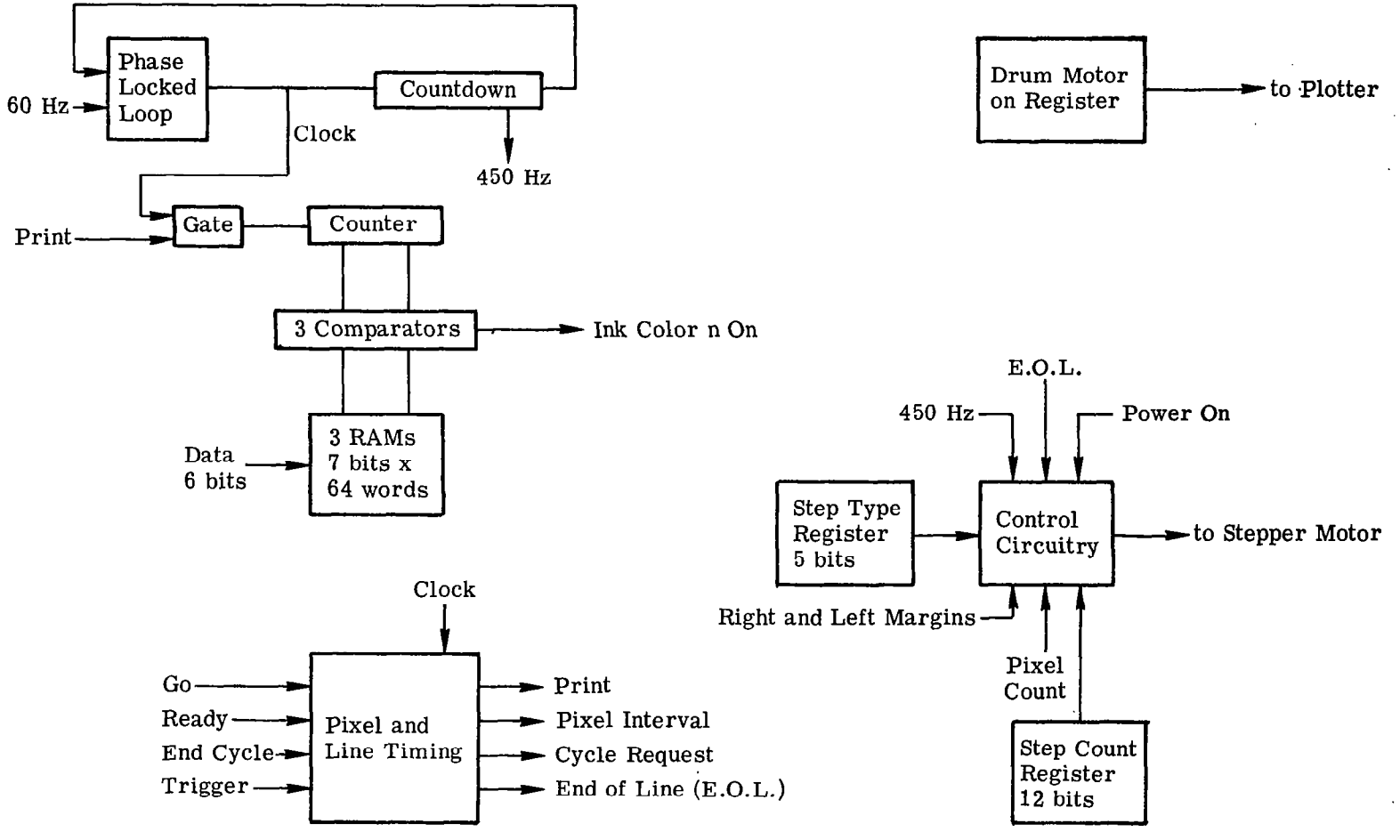


FIGURE 37. BLOCK DIAGRAM OF INKJET PRINTER CONTROL INTERFACE

motor can be turned on or off under computer control by loading the appropriate value into the drum motor register shown in Figure 37. The stepper motor moves one step every time it is pulsed. The maximum stepping rate is obtained by driving the motor with a 450-Hz square wave. The direction of rotation is controlled by the direction signal.

The special purpose interface has two major parts as shown in Figure 37. One part controls the turning on and off of the three ink jets, the other controls the stepper motor. The control of the ink jets is accomplished by turning them on for a period of $n/128 \times 50$ μsec, where the value "n" is under computer control. Within every 50 μsec interval the interface requests two 16-bit words from the computer and loads them into a first rank register. At the beginning of the next 50 μsec pixel-print cycle three 6-bit "color words" are loaded into the second rank registers from these first two words. The outputs of these registers are fed to the address lines of three 7-bit RAMs. The output of each RAM feeds one side of a digital comparator while the other side is fed by a counter. The counter starts at zero when these words are loaded and counts up to 128. As long as the counter is less than the binary value from the RAM the particular color prints until the value of "n" is reached. The cycle repeats every 50 μsec and ends when the computer indicates all data for that line has been transferred by turning off the ready line of the DR-11B. The cycle is primed again when the computer sets up another DMA transfer and issues a "GO" signal. The actual printing and data requests start after this "GO" and a "TRIGGER" signal from the printer is received at the interface. The use of the RAMs as a color lock-up table permits changing colors easily and rapidly without changing the data values.

The stepper motor control has four different modes of operation: (1) slew right or left; (2) make a large number of steps while not in the print mode; (3) step at the end of each line of print; and (4) step while printing. The slew mode is achieved by setting a bit in the step type register which permits a 450-Hz signal to be applied to the stepper motor. When a margin is reached the margin indicator signal is fed back to the computer which senses the end of the slewing action. The large-number-of-steps mode is used to position the recording head. A 12-bit register is loaded with some binary number. The interface will then send that number of pulses to the stepper motor. The maximum number of 4096 pulses moves the recording head about 40 mm (1.5 in.).

The step-at-the-end-of-line mode requires that a "GO-Trigger-End-of-Line sequence has occurred. At that time the interface sends "n" pulses to the stepper motor, where n is the number loaded in the 12-bit register. Usually this number is 20, to provide printing of contiguous lines. If the stepper motor is stepping and the interface receives the Go-Trigger combination to start printing, the interface will not print until the stepping is completed and another trigger pulse is received.

The step-while-printing mode causes the stepper motor to make one step every $n \times 50$ μsec. This will produce a skewed picture. The number "n" is loaded into the previously

mentioned 12-bit register. Usually this number is about 50 to produce contiguous lines; it cannot be less than about 25 because the pulse rate would be too high for the stepper motor to follow. The interface will halt stepping in this mode if a Go-Trigger-Trigger combination is detected because this would put gaps in the picture.

2.8 GENERAL-PURPOSE COMPONENTS

The general-purpose components now used in the system are shown in the block diagram of Figure 38. The basic computer system is configured with the DEC PDP-11/45 CPU with core, disc, and tape storage. Program development is normally done with standard units, a line printer, an alphanumeric CRT and a keyboard/printer.

MIDAS operation uses the RAMTEK components, the mass disc and the inkjet color printer for image display, storage and printing, respectively. Data input may be provided by analog tape,high-density digital (HDT) tape or computer-compatible tape (CCT) in 7- or 9-track formats. Classification and preprocessing are done in the special purpose pipelines, which are treated as Unibus-compatible peripherals.

All of the above components are interfaced to the standard DEC Unibus, principally by DMA (DR-11B) devices or single word transfer (DR-11C) devices which are not shown, for the sake of simplicity.

The general-purpose system is actually a fairly powerful "midi"-computer, employing the variety of standard hardware and software tools available in the DEC-11 system.

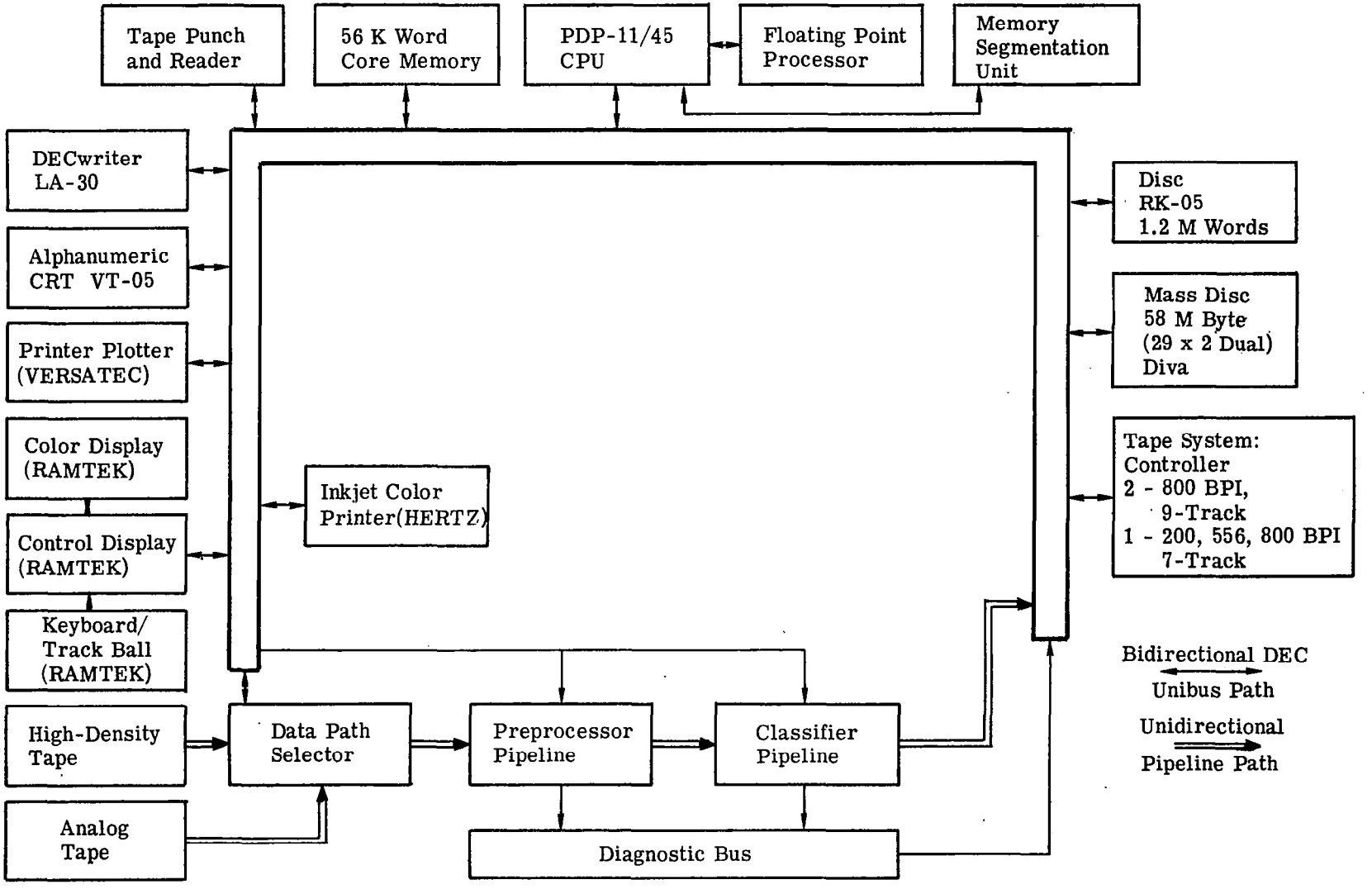


FIGURE 38. BLOCK DIAGRAM OF GENERAL-PURPOSE COMPONENTS

MIDAS SOFTWARE

3.1 INTRODUCTION

3.1.1 DIRECTION AND PURPOSE OF SOFTWARE

The software requirement on the MIDAS project consisted of the following main tasks. (1) To control the high-speed flow of data through various pieces of hardware constructed during the project, specifically input data transfers to and output data transfers from the MIDAS Preprocessor/Classifier hardware. (2) To maintain the large information system necessary for multispectral data processing and analysis, interfacing this information data-base to the hardware data-transfer control features. (3) To develop applications programs, in the various multispectral data processing areas, which are highly user-interactive and which, as a result, reduce the total number of man-hours necessary to complete a data analysis task. In brief, these were the major responsibilities of the software portions of the Phase II MIDAS project.

To expand on these general requirements, it is necessary to reiterate the hardware constructs which set MIDAS apart from other multispectral data analysis systems. Of course, the major hardware innovation is the Preprocessor/Classifier pipeline which performs the high-speed data manipulation functions of (1) multiplicative/additive scan line correction as a function of scan angle, (2) linear combinations of channels, (3) ratios of channels or of linear combinations of channels, and (4) calculation of classification type, using the multivariate quadratic decision rule. The speed of this piece of hardware (approximately 2×10^5 decision calculations per second) changes the emphasis of multispectral processing from batch-oriented data processing, where job set-up time is less than or about the same as the data-processing or classification time, to a situation where the emphasis must be on the man/machine interaction time, i.e., job set-up time. In the MIDAS system, the data-processing time has been drastically reduced by a factor of 100 to 1000, depending on the speed of the data source, compared to general-purpose computer software classification programs. In order to make use of this time/cost reduction, the man/machine interaction time must be reduced.

The method chosen to decrease the man/machine interaction time and increase the overall data processing rate (and thus the system's cost effectiveness and cumulative data throughput rate) has two main components. First, it provides the user with as much information as possible concerning various parameters of the processing task, and thus makes more effective use of the human's intuitive decision-making capabilities (which are still an integral part of most multispectral data analysis tasks). Second, it decreases the average access time to randomly-selected portions of multispectral data in order to decrease the time needed to perform certain pre-classification statistical analysis functions. One of the most powerful methods of providing information to the user is through the use of color and grayscale images of raw and processed multispectral information. These images can be displayed on two devices in the MIDAS system: (1) a color/BW CRT display system, or (2) a color image plotter for hardcopy output. These

images can provide the user with the necessary visualization of the data to guide his decision-making processes in job set-up, especially in the area of training and/or test field selection. Additionally, by using various interactive devices attached to the display system (specifically, an interactive trackball and a keyboard) to control the flow of processing, quicker response times and more complete information display can be achieved, thereby reducing the job setup time.

In order to reduce the processing time, two pieces of hardware, a 58-million-byte disk system and a high-density digital tape (HDT) system, are incorporated into the design. The disk system, which allows storage of large amounts of raw multispectral data and processed results, provides a much lower access time to random portions of a data scene than does a CCT-based system. Thus,all processing and analysis is performed using the disk database, reducing the average data-access time to the order of milliseconds from that of CCT-based systems with average access times ranging from hundreds of milliseconds to hundreds of seconds. Use of the high-density digital tape system will provide significant increases over CCT's in the efficiency of loading raw data onto the disk system, and can also be used as an input data device capable of driving the MIDAS at its maximum data processing rate. However, there is still a data output limitation due to the average data transfer rate possible with the bulk-storage disk system, and the high-density tape system will have to be slowed down by at least a factor of two to satisfy this limitation.

3.1.2 SOFTWARE OVERVIEW

In developing applications programs which perform the analysis and hardware-control functions of the MIDAS system, it was decided that the effective limit of 24K words under the DOS/BATCH operating system was inadequate for most of the applications. Two methods for alleviating this virtual memory problem were investigated: (1) modification and addition to the DEC DOS/BATCH operating system to provide a 65K-word program environment, through the use of the PDP-11/45's KT11-C memory management hardware option, or (2) use of the DEC RSX11-D multitasking operating system.

The DOS/BATCH modification and addition scheme was selected, mainly because an in-house-developed operating environment could be more finely tailored to the needs of the MIDAS system. However, the other major considerations were the shortcomings of the RSX11-D system, in the initial planning periods of Phase II MIDAS. RSX11-D was a new DEC software product, and as such was experiencing the initial field-test problems found in most new systems; in addition, the high purchase price placed on the operating system with source code and automatic DEC update service seemed unreasonably high for the budget of this project.

The modifications to the DOS/BATCH system consisted mainly of changes to the system hardware description and the insertion of several device driver routines into the monitor to control the various new pieces of hardware obtained in Phase II MIDAS. Additions to the system include a set of routines to handle the virtual addressing scheme of the KT11-C memory

management unit, to handle the interfacing of the user-program calls to monitor utility routines, and to process calls to MIDOS (MIDas Operating System) utility routines via the TRAP instruction. If more core memory had been purchased, a 65K-word program operating environment would have been possible; but with the purchase of an additional 32K word section of memory, approximately 40K words are available for a resident application program. Program and overlay loading are some of the services provided by MIDOS. Additionally, the display of interactive menus and subsequent interaction with these menus is accomplished via calls to MIDOS.

3.1.3 APPLICATION PROGRAMS

Within the MIDOS and application program system, there are presently five information systems to be maintained and used by application functions. These are

(1) the multispectral image data system (stored on the bulk storage DIVA disk system)
(2) the field-definition information system (stored on the RK11-C system disk)
(3) the scan-line correction function information system (stored on the RK11-C system disk)
(4) the statistical signature information system (stored on the RK11-C system disk)
(5) the interactive-menu/display file system (stored on the RK11-C system disk)

The first four of these information systems are modified and accessed by all applications programs, while the fifth is set-up off-line and can be thought of as an integral part of the program code associated with each application. Consequently, this information system cannot be modified by an application program; it can only be accessed for use.

The applications programs which have been developed follow to a large degree the standard functional breakdown associated with most current multispectral data analysis systems. The applications are broken down into the following areas: (1) IMAMAN, the multispectral database control program which performs image loading and file maintenance functions; (2) DISPLA, the data-display program which provides for color or grayscale image generation in both CRT and hardcopy modes; (3) FLDMAN, a field-definition information system manipulation program which allows user-definition of fields (polygonal regions within a multispectral data scene) for use in training processes and post-classification analysis; (4) STAT, a package of statistical routines which use the field-definition and multispectral data information to calculate signatures and scan line correction functions; (5) HDWCON, a MIDAS-hardware control program, which loads all hardware RAMs in the MIDAS, conditions it for a particular operation, such as multivariate classification or ratioing, and creates a disk file with the results; and (6) PANALY, a post-classification analysis program, which helps to indicate the success or accuracy of a given classification processing task. The details of the functional capabilities and the actual programs are given in Section 3.3.

All of these applications programs interact with the PDP-11/45 computer via the process-control keyboard and trackball associated with the RAMTEK CRT display system. Associated

with the process-control keyboard are several keys, designated super-function keys, which suspend the operation of a current application program, to perform some global function necessary for efficient system operation. The two current super-functions to be implemented in Phase II MIDAS are the scrolling/zooming function,and the function which terminates a current application and allows the user to select a new application function without disturbing the integrity of any of the information systems which may be in the process of modification by the current application.

The following procedure illustrates a possible processing sequence as an example of the MIDOS software system in operation.

> The raw multispectral data must be loaded into the bulk-storage disk data-base. Assume that the data set to be processed is a LANDSAT-2 data frame stored in LANDSAT-1 quarter-frame format in four files on two 9-track 800 BPI computer-compatible tapes. The user would specify the scene associated with each quarter-frame and loading of the data onto disk would proceed. A full-frame multispectral data file would be created in the bulk-storage information system, under a filename specified by the user.
>
> The next processing step might be to display various channels of the raw data on the CRT, looking for the best feature-discrimination (i.e., distinction between ground-types). A hardcopy of the color image could be produced and taken off-line to extract line and pixel number selection for field-definition, possibly the next logical processing step.
>
> In order to calculate signatures based on several areas, the boundaries of these polygonal regions must be defined by the user. These definitions may have been decided upon off-line through the use of available hardcopy and imagery, and could be punched on cards, stored on tape, and read directly into the system. Alternately, using the cursor/trackball combination in conjunction with the displayed CRT image, polygonal fields can be drawn on the displayed image and entered into the field-definition information data-base for use by subsequent processing functions.
>
> If a classification is to be performed, signatures describing each class must be entered into the signature information system, either from CCT (and thus calculated off-line or restored

from previous processing runs), or through calculation of the signature statistics using subsets of fields previously defined.

After a possible analysis of the signatures is performed, the user must select the signatures and the subset of channels of the raw multispectral data image to be used. These signatures will be loaded into the classifier, and the classification will be performed on a user-specified subset of the scene. Output of classification results will be made to a user-specified disk file. A count of the number of pixels per class will be produced.

After classification, analysis of the results might take one or both of the following methods: color display of the classification file or statistical analysis of a pre-selected subset of the classified scene. This analysis could take the form of a matrix describing the success of the classification calculations for each class, assuming homogeneity of a particular class within each of the test areas designated, and also give the number of "incorrect" classifications, as well as the overall proportions of each class in these test areas.

3.1.4 DIAGNOSTICS

Several diagnostic programs were written under MIDAS Phase II, most of which deal with the new pieces of hardware which have been added to the system. CHECK4 is a program which provides a probing mechanism to access all of the diagnostic ports within the MIDAS Preprocessor/Classifier hardware. CHECK8 is a diagnostic used to detect data transmission errors through the MIDAS hybrid circuitry as well as through the Preprocessor. IJEX and IJPTST are two diagnostic programs used to test and control the operation of the inkjet plotter and its computer interface.

3.2 OPERATING SYSTEM

As was discussed in the system overview, an immediate requirement for the Phase II MIDAS software effort was the extension of user-program storage from ~24K words toward 65K words with a minimum of 32K words. Since no operating system available at the start of Phase II satisfied this requirement, system software was devised which would meet this need as well as some of the other specialized requirements of the software effort as it was specified. The major areas of effort in the system software developed were (1) memory allocation, in conjunction with the hardware KT11C memory segmentation unit, (2) dynamic program loading, (3) interface user programs with the DOS/BATCH system, especially in the areas of

I/O, and (4) implementation of device drivers for non-standard hardware developed in Phase II. Some of the considerations that were taken into account in the design and implementation of MIDOS were: (1) keep the MIDOS components as small as possible in terms of minimum core requirements, (2) provide the mechanism for real-time response to a user interacting with a menu display system, and (3) provide a common environment-description area for inter-program-segment communication.

3.2.1 DOS/BATCH

The operating system which is used for software applications to the MIDAS project is composed of two major components: DOS/BATCH [V09-20C] and the MIDOS system components. The Digital Equipment Corporation's DOS/BATCH Operating System V09-20C is used, with subsequent released correction updates, and with the addition of several device drivers. The drivers which were written handled the following non-standard peripherals: (1) the RAMTEK imaging display CRT system, (2) the inkjet printer and preprocessor RAM loading interface, (3) the process-control keyboard device associated with the RAMTEK system, and (4) the VERSATEK line printer-plotter. No modifications to this operating system have been made to facilitate the incorporation of the MIDOS components. Both DOS and MIDOS are resident in "kernel" virtual space of the PDP-11/45. MIDOS is responsible for the set-up and control of the user-program which is resident in user virtual space.

3.2.2 MIDOS INITIALIZER

This is the first component of the MIDOS system which is brought into core as a standard load module by the DOS monitor. It is possible to run MIDOS in a special system debugging mode through console switch register settings. In a normal operation, however, this mode would not be used. Initially, INIT loads a package which contains the modules MMSERV, ERROR, and TRAPS. These routines are loaded separately from INIT (although they are part of the MIDOS system) so that in the debug mode, different versions of these modules could be loaded without confusing file manipulation prior to loading. After these three MIDOS components have been loaded, INIT initializes all link blocks in the system common area, thus making all device drivers resident in "kernel" virtual address space.

Then the EMT, TRAP, and KT11C interrupt vectors in low-core (kernel addresses 30_8, 34_8, and 250_8 respectively) are initialized to provide transfer to the proper location within the routines TRAPS and MMSERV. The existence of all non-resident MIDOS trap routines is verified and their locations on disk are stored in the trap residency table. The KT11C memory segmentation unit is then initialized by loading the "kernel," "supervisor," and "user" PARs (page address registers) and PDRs (page descriptor registers). Next a TRAP 000 instruction is issued to load the first application program. In debug mode, any user program may be specified; in non-debug mode the program USERAP, which allows the user to select an application program, is loaded.

3.2.3 MEMORY MANAGEMENT SERVICE ROUTINE (SERVE)

This MIDOS component module contains three separate routines; MMSERV, BYPASS, and SLSERV. MMSERV is the routine which services the KT11C memory segmentation unit interrupts. The only interrupts which are enabled from this hardware are those which are generated by page addressing errors, i.e., attempts to address a virtual memory which does not exist. Since the circumstances which generate an interrupt into this routine are non-recoverable, a fatal system error indicating the cause of the problem is issued. BYPASS is a routine which intercepts the interrupts which are generated by the execution of an EMT or TRAP instruction by the program executing in "user" virtual space. Since both the "kernel" space and the "user" space have separate stacks, and since these stacks are the mechanism for passing arguments in the EMT as TRAP calls, BYPASS is responsible for transferring information from the "user" stack to the "kernel" stack before entry to the routine designated by the EMT or TRAP instruction. It is also responsible for the transfer of information from the "kernel" stack to the "user" stack subsequent to execution of the called EMT or TRAP routine. SLSERV is the routine which services stack limit register violations, i.e., when the "user" or "kernel" stack falls below the limit of memory available to the virtual address space. Again, since the circumstances which generate an interrupt into this routine are non-recoverable, a fatal system error is generated.

3.2.4 ERROR HANDLER ROUTINE (ERROR)

This routine is used for all error reporting from MIDOS. It is responsible for all non-recoverable error messages, and thus it must return the state of the machine to a condition which coincides with the environment expected by DOS/BATCH. As such, this routine must disable the KT11C memory segmentation unit, reset the TRAP and EMT vectors to their former values and release all I/O link blocks and device drivers. A call is then made to the DOS/BATCH error diagnostic package (EDP) which then issues the requested error message.

3.2.5 TRAP HANDLER ROUTINE (TRAPS)

This routine is responsible for calling the user-requested TRAP routine. Since there are both resident and non-resident TRAP routines, TRAPS must determine if the requested trap routine is currently available. If the routine is non-resident and not currently residing in the trap buffer within MIDOS, it must be acquired from the disk and loaded into the trap buffer. If the routine is resident, no loading is necessary. TRAPS sets up the kernel stack to overlay the user stack and thus passes arguments to and from the stack routine. The current occupant flag is set to indicate that a routine is present in the trap buffer. Control is then passed to the trap routine.

For all non-resident traps, i.e., those trap routines which are stored in "core-image" format on the RK-05 disk, there is a fixed naming convention which is expected by the routine TRAPS. All non-resident trap routines are referenced by number and must have a filename of the format "TRAP. xxx[1,6]" where xxx=trap reference number.

3.2.6 TRAP ROUTINES (TRAP XXX)

Trap 000—Program Loader

This resident trap routine is used to call in a new segment of user-program into user-virtual space. It is capable of using three different types of input modules: LDA, LDI and LDD load modules. An LDA-module is assumed to have its instruction and data program code in a single 32K word area and,thus,in the KT11C memory segmentation unit set-up, I-space (instruction space) and D-space (data space) are overlaid. If the module requested for loading is not an LDA module, the LDI module is assumed to contain up to 32K words of instruction code and an associated LDD module is assumed to contain up to 28K words of data (the remaining 4K words allow access to the physical hardware address page). This loading scheme allows a 60K word program environment. I-space and D-space in this situation are not overlaid. This trap module is composed of eight major routines: PLOADR, RCOMD, LOADM, ASSIGN, GETHDR, GETBYT, CLEAR, and CAULK.

TRAP 000 operates as follows. After removing and saving all trap arguments from the stack,all of user core is zeroed. The COMD (communications directory) from the requested load module is obtained and the relevant information is saved in the system common area. Next, dependent on the type of modules to be loaded (either LDA or LDI/LDD), the necessary virtualpages in I- and D-spaces are computed. LOADM is called to unpack each load module segment found by GETHDR into the core allocated by ASSIGN.

Trap 001—Overlay Loader

The function of TRAP 001 is to load a module anywhere in existing core without altering the program environment of the code which has requested the overlay. This non-resident module contains the routines OVERLAY, OLOADM, and EXIT and accesses the routines RCOMD, CLEAR, GETBYT, and GETHDR, all of which are part of the resident TRAP 000.

The operation of TRAP 001 is as follows. The overlay module file is read and its COMD is unpacked for relevant information. Address checking for existence is performed. If the flag CLEAR, passed as an argument, is non-zero, the overlay buffer is zeroed. Subsequently, the overlay module is unpacked and loaded into the overlay buffer. The return address from the argument list is determined and control is passed to the calling program.

TRAP 004—Extended Address Bit Set-Up

This resident trap routine is used to convert the 16-bit virtual address associated with a DOS .TRAN block into an 18-bit physical address, necessary for DMA device set-up on control. Word 2 of the .TRAN block must contain the user-space virtual address of the DMA I/O buffer. Upon return from this trap routine, word 2 will reflect the low-order 16-bits of the physical address corresponding to the virtual address passed by the user. Bits 4 and 5 of word 3 of the .TRAN block reflect bits 16 and 17 of the physical buffer address. An error return code is made if the .TRAN block address or the virtual address specified in the block is not accessible.

TRAP 007—Virtual-to-Physical Address Conversion Routine

This resident trap routine performs the same function as TRAP 004 (i.e., conversion of a virtual user-space address into an 18-bit physical address), only this routine provides much more detailed information about the reference memory location. Since interpretation of a virtual address by the KT11C as either an I-space or a D-space address is made in the context of each particular instruction, this trap routine returns the 18-bit physical addresses associated with both the I-space and D-space virtual address. Other information which is returned are the contents of the PAR/PDR sets (both I and D-space registers) associated with the particular virtual address.

TRAP 010—Physical to Virtual Address Conversion Routine

This non-resident trap routine accepts an 18-bit physical address and returns all virtual addresses which reference this physical memory location. Since it is possible that several or all I-space and/or D-space PAR/PDR sets could reference the same memory location, up to 16 different virtual addresses might be returned to the calling program.

TRAP 011—MIDOS Exit/Restart Routine

This resident trap routine is called to generate a DOS/BATCH system error. An error code (16 bits) is passed on the stack to this trap routine. This routine then issues a call to the EDP (error diagnostic print) package within DOS/BATCH. An error message of the form 'A201 xxxxxx' is issued. At this point the user may restart MIDOS using the $CO command to DOS/BATCH or kill DOS using the $KILL command. It is not possible to restart the user-program from this point.

TRAP 012—Menu Display File Initializer

This is a non-resident trap routine which is used in conjunction with TRAP 013 to provide a simple mechanism for interactive menu display. A menu file, containing up to 50 separate display menus and created offline via a system program named CREDF, is attached by calling this trap routine. The system common area is set up to reflect the number of menus in the display file and the disk location (block address and word displacement) of each menu. Fatal system errors associated with this routine are caused by a non-existent display file, a display file format error, or a disk read error.

TRAP 013—Menu Display Routine

This non-resident trap routine attempts to display a menu, in conjunction with the MIDOS system common information which has been set up by a prior call to TRAP 012 made by the user program. The menu number requested is checked for validity and, if available, is obtained from disk and passed to the RAMTEK display system via a .TRAN call to the device driver. Associated with each menu, there may be a response table, i.e., a table which defines areas associated with menu-interactive decisions made by the user. This information (if it exists) is stored in the MIDOS system COMMON area for use by various system subroutines. Fatal system errors associated with this routine are caused when no display file has been

previously attached via TRAP 012, or by a non-existent or out-of-range menu number, a display file format error (more than 128 entries in response table), or DMA transfer errors either from disk or to the RAMTEK display system.

TRAP 014—Scanline and Pixel Number Calculation Routine

This non-resident trap routine calculates the line and pixel number for a given pixel displayed on the screen of the RAMTEK display system. In the course of image generation by the application program CRTDIS, the system COMMON area is updated to reflect the line and pixel limits currently displayed on the color monitor. It is possible to calculate the scan line and pixel number for a physical RAMTEK screen coordinate based on the magnification factor in both axes. Error return codes to the calling program are caused by a non-existent display image or a screen pixel selection for which no scanner pixel exists. No fatal system errors are possible.

3.3 APPLICATION PROGRAMS

3.3.1 DATA BASE CONTROL PROGRAM (IMAMAN)

Image Manipulation (IMAMAN) provides the user with the means to move multispectral images to or from the MIDAS DISK system (see Table 7). The transfers can be made to or from the following media: Computer Compatible Tape (CCT),High Density Digital Tape (HDT), and Analog Tape. IMAMAN performs seven functions.

(1) Load an image from a media to MIDAS
(2) UNLOAD an image from MIDAS to some media
(3) LIST the directory on the MIDAS disk system
(4) delete an image from the MIDAS disk system
(5) LIST a report of the work done on a particular file
(6) rename a file and its associated files
(7) exit IMAMAN and return to user applications menu

The load function starts by allowing a selection of the type of device to be used for input. Due to no HDT at this time, the only possible choice is CCT input. When CCT is selected, the logical unit number of the drive on which the CCT is mounted is entered, with the format of the tape and the file number desired.

The next step is to assign a nine-character name to the file which is to be generated. The first character defines which of the two MIDAS disk drives will hold the file.

Then the choice is made as to what part of the scene is to be loaded onto the disk. Scan lines can be selected starting at any line number, and go to any line number with any integer increment. The points on all selected scan line are selected in the same manner.

Channel selection is the next step. Each channel on the input medium can be selected or not for inclusion in the output file.

TABLE 7. OUTLINE OF FUNCTIONS PERFORMED BY IMAMAN

1. LOAD
   - A. Device Selection
     - (1) CCT
       - (a) UNIT #
       - (b) FORMAT
       - (c) FILE #
     - (2) HDT —NOT IMPLEMENTED
     - (3) ANALOG —NOT IMPLEMENTED
   - B. IMAGE NAME FOR OUTPUT
   - C. SCENE SELECTION
   - D. CHANNEL SELECTION
   - E. SCANNER WHICH GENERATED DATA
   - F. UPDATE TITLE RECORD
   - G. UPDATE SITE RECORD
   - H. UPDATE CHANNEL RECORD
   - I. OPTIONS
     - (1) PREVIEWING —NOT IMPLEMENTED
     - (2) STATISTICS —NOT IMPLEMENTED
     - (3) START LOADING PROCESS
   - J. PUT DATA IN FILE
2. UNLOAD —NOT IMPLEMENTED
3. DIRECTORY
4. DELETE A FILE
5. REPORT —NOT IMPLEMENTED
6. RENAME —NOT IMPLEMENTED
7. EXIT

The last piece of new information needed is the type of scanner which generated the data about to be read in. This allows the software to correct for the direction of scan when images are generated. This also makes some default information available.

The next three sections of input to the program allow the operator to change the information about the file. This includes the title, site, and channel information. The information can be changed or left unmodified.

At this point, the data is read from the input device and written on the MIDAS disk drives.

The unload function will simply take a disk file and copy that file unto the selected output device.

The directory function gives the operator a means by which he can list the directory of the two MIDAS disk drives.

The delete function allows for the removal fo a file from the disk system.

The report and rename functions have not been implemented at this time.

The EXIT function allows the operator to return to the USER-APPLICATIONS program to select a different application.

3.3.2 DATA DISPLAY PROGRAM (DISPLA)

The data display program (DISPLA) provides the user with the means to generate color or grayscale images either on the 63.5 cm CRT screen or on the hardcopy inkjet plotter (see Table 8). All data input is from the disk-data-base information system. The current algorithm, which is implemented for both CRT and IJP output, is a manual-level-setting method, which very simply assigns a particular color to a given set of input data levels. This algorithm was the simplest implementation which provided all of the necessary display capabilities initially required. Additional algorithms to provide multichannel false-color mapping and other similar, more complex color image-mapping algorithms will be implemented and incorporated into this program after user feedback provides information on the types of display functions necessary for more complicated MIDAS processing tasks.

The DISPLA program initially asks the user for the type of display to be generated, to which the user must respond with either the color CRT or the IJP as the output device. Once the device has been selected, a multispectral data file must be specified via the standard nine-character filename. If this file exists, the program inquires about the number of channels and mode of the display. Since only the single-channel, manual mode is implemented, only these responses will be treated as valid. Next, the program will request the type of display to be generated: color or grayscale. Only these two responses are considered valid.

Next, the program will interrogate the user for the magnification factors to be used in displaying the data. Two separate positive integer values must be entered, the width

TABLE 8. OUTLINE OF FUNCTIONS PERFORMED BY DATA DISPLAY

1. Select output device
    A. Color 63.5 cm CRT
    B. Inkjet plotter hardcopy product
2. Select input file for display
    A. If file exists, go to 3
    B. Go to 2
3. Select display mode
    A. Select display algorithm
    (Note: only manual level-setting implemented)
    B. Select number of channels
    (Note: only single-channel implemented)
    C. Select color scheme
        1. Color (default)
        2. Grayscale
4. Select pixel and scan line magnification factors
(Note: default for both values is 1)
5. Select scene descriptor parameters
    A. Enter channel number
    B. Enter rectangular scene descriptor array
        1. Start scan line number
        2. Ending scan line number
        3. Scan line increment
        4. Starting pixel number
        5. Ending pixel number
        6. Pixel increment
6. Enter levels associated with each color
7. Enter default color value
8. Generate display
    A. Read scan line from disk
    B. Translate and magnify data into output data line
    C. Transfer data to display device
    D. Update common area image descriptor array, only if CRT is output device
9. Go to 1

magnification (or pixel duplication factor) and the length magnification (or line duplication factor). The default value for each of these is 1, producing an unmagnified image on the imaging device.

Following this, the program will request the necessary information concerning the data to be used for input to the image-generation package. The channel to be used must be entered by the user. No default condition exists for this response. Once the existence of this channel is verified, the rectangular scene to be displayed must be entered by the user. Six integral values must be entered describing the following parameters:

(1) starting scan line number (default: first line of file)
(2) ending scan line number (default: scan line number which will maximize size of the image on the selected output device)
(3) scan line increment (default: file scan line increment value)
(4) starting pixel number (default: starting pixel number of file)
(5) ending pixel number (default: pixel number which will maximize size of the image on the selected output device)
(6) the pixel increment (default: file pixel increment value)

Since the two imaging devices have upper bounds to the size of the image that they can display, an error will be generated if the requested display is too large. The user will then have to enter a new scene descriptor array compatible with the physical limitations of the display devices. The RAMTEK is capable of generating images which are 510 pixels wide and an unlimited number of lines in length (note that any image with more than 512 lines will have some portion of the data scrolled off the screen). The inkjet plotter (IJP) is capable of generating images on 21.6 × 27.9 cm paper which are 864 pixels wide and 1314 scan lines long.

After the plot extent has been described, the assignment of colors to input data levels must be specified. A color is selected and the minimum and maximum range for the color is entered. This process continues until all levels have been specified. Finally, a default color must be specified to encompass those input data levels not assigned to a specific color.

Upon completion of this data entry phase, the color display is generated. The common area is updated after every scan line to reflect the actual limits of the currently displayed image. This information can be used by other applications programs in interacting with the currently-displayed image.

3.3.3 FIELD DEFINITION AND DESCRIPTION (FLDMAN)

Field manipulation (FLDMAN) provides the user with the means to generate polygonal training and test fields. The training field then can be used for signature extraction, and the test field can be used for post-classification analysis. There are three means by which the field definitions can be made.

(1) Draw the polygon on the color monitor.

(2) Enter the vertices from the keyboard.

(3) Read a 7-track magnetic tape with BCD card images of vertices.

Drawing on the color monitor is done by selecting the first vertex with the cursor and hitting the "ENTER" button on trackball. The same is done for the other vertices. The end of the field is signaled by hitting "ENTER" without moving the cursor. The field is checked for errors and an inter-scan-line and inter-pixel increment are entered. Next the field name, class and type are put in. The name is arbitrary, the class lumps fields together, and the type is either test or training.

Entering the vertices from the keyboard is done by typing line and point number pairs for each vertex. The end of the field is signaled by typing a carriage return with no pair of numbers preceding it. Then the increments, field name and class, and the type are entered as described above.

When reading fields from magnetic tape, the same information is read from the tape in the same order as in the other two methods until the end-of-file is found. The format of the card images is shown in Table 9.

3.4 STATISTICAL PROGRAMS

3.4.1 SIGNATURE MANIPULATION (SIGMAN)

Signature manipulation (SIGMAN) provides the user with the means to obtain signatures for use with the classification hardware. SIGMAN uses the fields generated by the field manipulation package. This program combines and scales signatures as well as extracting them.

When extracting signatures, the user selects from the current field file all the field names and field classes which he wishes to use in calculating the signature. Next he selects which channels are to be used in the calculations, and specifies the number of standard deviations from the mean data value permitted for each channel. This criterion is used to exclude pixels from the calculations. With this information, the signatures are calculated and the results displayed.

If the user accepts the signature it is given a name and is written into a file. If the user rejects the signature several alternate choices for calculating a new signature are available: (1) a new set of fields, (2) a new set of channels with the same fields, (3) different editing limits with the same channels and fields.

When combining signatures, the user must specify the weight to be given to each of the signatures. The signatures used must all have the same number of spectral channels. The combined signature is then calculated in the following way:

$$\overline{\overline{X}} = \sum_{i=1}^{m} W_i \overline{X}_i \qquad \overline{\overline{Y}} = \sum_{i=1}^{m} W_i \overline{Y}_i$$

TABLE 9. FORMAT OF CARDS FOR INPUT TO FIELD MANIPULATION

Vertices Cards

Column No.	Content	Card 1	Card 2	Card 3	Card 4
1-5 6-10	line # point #	vertex 1	8	15	22
11-15 16-20	line # point #	vertex 2	9	16	23
21-25 26-30	line # point	vertex 3	10	17	24
31-35 36-40	line # point #	vertex 4	11	18	25
41-45 46-50	line # point #	vertex 5	12	19	
51-55 56-60	line # point #	vertex 6	13	20	
61-65 66-70	line # point #	vertex 7	14	21	
71-79	field #	(Use same # on all cards of each set)			
80	1 through 4 to indicate card number for the appropriate set of vertices.				

Information Cards

Column No.	Content
1-5	Inter-scan-line increment
6-10	Inter-pixel increment
11-19	Field Name
21-29	Field Type
31	Category: T = Test G = Training
71-79	Field number as for vertices card
80	"5"

$$\text{and new } COV(X,Y) = \sum_{i=1}^{m} W_i [COV_i(X,Y) + \bar{X}_i \bar{Y}_i] - \bar{\bar{X}}\,\bar{\bar{Y}}$$

where $\bar{X}_i$ = means for signature i; channel X
$\bar{Y}_i$ = means for signature i; channel Y
W_i = weight for signature i
$\bar{\bar{X}}$ = mean for combined signature; channel X
$\bar{\bar{Y}}$ = mean for combined signature; channel Y
m = number of signatures being combined

After the signatures are comscaled, the user provides a nine-character name for the new comscaled signatures to be stored on disk. This name is later referenced by the RAM load package when setting up MIDAS.

The basic mathematical calculations are performed by a subroutine STAT.

3.4.2 SUBROUTINE STAT

If editing of the data base is requested (EDII<1), the upper and lower bounds for data values in each channel are chosen so that no more than one out of a thousand are rejected.

These editing bounds are based on the median (as an estimate of the mean) and the quartile value (as an estimate of the standard deviation). The actual standard deviation used for each channel appears in QDEV, and the upper and lower editing limits for each channel appear in EDHI and EDLO respectively. After the editing procedure, all the data points that were within the editing limits in all channels are the first NSS-NREJT data values in each channel. The remaining data values in each channel were rejected,because at least one of the corresponding data values in another channel was not within the editing limits. If fewer than two points are considered good after the editing procedure, ISW is set to 2 and the subrouting returns. If there are two or more good points after the editing procedure, the mean and median are determined for each channel. Then the covariance matrix is calculated using the following equation:

$$COV(K,L) = \frac{NESS}{NESS-1}\left\{\left[\frac{1}{NESS}\sum_{I=1}^{NESS} DATA(K,I)*DATA(L,I)\right] - MEAN(K)*MEAN(L)\right\}$$

where COV = floating point, double-precision covariance matrix
NESS = the integer number of good data points
DATA = the integer array of data values
MEAN = the floating point, single-precision vector of the mean for each channel

The standard deviation for each channel is calculated by extracting the square root of the diagonal elements of the covariance matrix. The correlation matrix is determined next through use of the following equation:

$$COR(K,L) = \frac{COV(K,L)}{STD(K)*STD(L)}$$

if STD(K)*STD(L) = 0, COR(K,L) = 1.

where COR = correlation matrix; the diagonal elements will be all 1's and all off-diagonal elements are between 0 and 1

COV = covariance matrix

STD = vector of the standard deviation for each channel

If the user has requested the eigenvectors and eigenvalues for the covariance matrix (ISW ≠ -1), these are calculated and the subroutine returns control to the calling program.

The floating point numbers which are generated by the STAT subroutine must be converted into 16-bit RAM constants suitable for loading into the MIDAS Pre-processor and Classifier. The various operations are described below for each parameter.

Means

The means used by MIDAS must be negated and scaled so that no more than eight bits in 2's complement notation are used. That is, the high-order bit (bit 7) is used as a sign. Therefore, the smallest number possible is:

$$10000000 \text{ base } 2 = -128_{10}$$

and the largest number possible is:

$$01111111 \text{ base } 2 = +127_{10}$$

The numbers coming from the signature file are in offset binary notation with the smallest integer number represented by

$$00000000 \text{ base } 2 = 0_{10}$$

and the largest integer number represented by

$$11111111 \text{ base } 2 = 255_{10}$$

The incoming double-precision, floating-point mean is negated. The constant 128_{10} is then added to the result to convert it to 2's complement notation. This number is converted to integer and put in the output buffer. If an input mean is ≥256, an error message is printed. If the data to be processed is positive magnitude, the conversion of the means to 2's complement is omitted.

Variances

The standard deviation for each channel is determined by calculating the square roots of the diagonal elements of the covariance matrix. The standard deviation is inverted, scaled, and converted to integer. MIDAS presently has only 12 bits available for this number. Therefore, to obtain as many significant bits as possible, the standard deviations are subjected to the following restriction:

$$1 < \sigma$$

With this restriction, MAX $[1/\sigma] = 1.0 - 2^{-12}$. In binary this is:

0.111111111111

This provides the maximum number of significant bits with no bits constantly 0 or constantly 1. If a standard deviation is less than 1, the standard deviation is altered to make it just enough greater than 1 so that

$$\frac{1}{\sigma} = 1.0 - 2^{-12}$$

Consequently, the processing procedure is as follows:

1. A diagonal element of the covariance matrix is found, and its square root is calculated. If no more, quit processing
2. If the result is greater than 1, go to step 3. Otherwise, multiply the diagonal element by

$$\left[\frac{1}{(1 - 2^{-12})*\sigma}\right]^2$$

   Then multiply the column of the upper diagonal covariance matrix (except the diagonal element itself) by

$$\frac{1}{(1 - 2^{-12})*\sigma}$$

   Then go back to step 1 and select the same diagonal element again. If $\sigma = 0$, σ is set equal to $\frac{1}{1 - 2^{-12}}$
3. Multiply this number by 2^{12}

$$\frac{1}{\sigma} * 2^{12}$$

4. Convert the result to integer and save it.
5. Go to step 1.

RAM Coefficients

To understand the reasons behind all the following mathematical manipulations, see the mathematical analysis in Section 2. The processing goes as follows.

1. Calculate the correlation matrix from the covariance matrix.
2. Transpose the correlation matrix. This matrix must be transposed because the next routine to use it was written in FORTRAN and expects all arrays to be stored in column-by-column instead of row-by-row form.
3. Calculate the eigenvalues and eigenvectors using the FORTRAN subroutine EIGEN.
4. Transpose the FORTRAN-form eigenvectors' matrix.
5. Treat the eigenvalues as the diagonal element of a square matrix with all off-diagonal elements set to zero.
6. Calculate the square root of each diagonal element and invert it.

7. Multiply this transformed matrix by the eigenvectors' matrix and store the results back in the eigenvectors' matrix.
8. Find the largest element in absolute value in the new matrix, negate it, divide every element of the new matrix by it, and save the constant.
9. Multiply the new matrix by 128 and convert it to integer. The low-order 8 bits are the only significant bits that MIDAS can handle.
10. Take the constant saved previously, square it, and store it in the output buffer following the coefficient matrix (stored in row-by-row form).

Determinant

This program does no scaling on the determinant. The determinant is found for the covariance matrix by performing a Gauss-Jordan inversion with partial pivotal elimination. The double-precision, floating point natural logarithm is calculated and stored in the output buffer.

3.5 HARDWARE CONTROL PROGRAM (CLASFY)

The hardware control program (CLASFY) provides the user with the mechanism to set up the MIDAS Preprocessor/Classifier hardware,and to generate the subsequent data file using this hardware (see Table 10). The set-up of the hardware determines the algorithm which is used to generate the new data file. The current implementation of this program allows the following algorithmic selections: (1) quadratic classification of either four channels with up to 16 classification types or eight channels with up to eight classification types, or (2) ratioing of two channels with the appropriate scaling necessary for retention of data significance..

The CLASFY program initially asks the user for any new set-up to be performed on the multiplicative-additive scan angle correction function hardware. The options available here are: (1) load the hardware with a user-selected function generated with the statistics package (not yet implemented), (2) load the hardware such that no correction is performed, or (3) perform no action whatsoever, retaining the previous configuration. The program will perform the desired action, and indicate in the COMMON storage area that the appropriate action has been taken.

Next the user must select the type of output desired, either classification or ratioed data generation. The classification algorithm currently implemented is a single-pass scheme with a maximum number of signature classes of either 8 or 16, dependent on the number of input channels; however, future implementations will include a multipass classification scheme with a much larger number of signature classes (possibly as high as 255) available. The ratio operation, as currently implemented, allows for generation of only one ratio between two input channels, but a multi-pass multi-ratio scheme may be implemented in the future.

If the classification mode is selected, the user must select the channel subset to be used in this classification pass. Once an acceptable channel subset has been selected, the signature file and subset of signatures (if any) must be selected. If the number of channels in the selected

TABLE 10. OUTLINE OF THE CLASSIFICATION FUNCTIONS

1. Load Preprocessor with appropriate correction functions
    A. No function (wire set-up)
    B. Function (1-dimensional) load
        1. Read correction function from disk
        2. Indicate correction function load in common area
2. Select type of output desired
    A. Classification
        1. Select number of channels and channel subset (or default to all in signature set)
        2. Select signature set (and subset of signatures if any)
        3. Load signatures, into classifier and set-up appropriate L.C. and ratio paths, storing Preprocessor description in common area
        4. Select input filename (or default to current) and output filename
        5. Specify rectangular scene to be classified
            (a) If enough disk space, go to 2.A.6
            (b) Allow user to delete unwanted disk files and go to 2.A.4
        6. Write output file title with classification description
        7. Perform classification, tallying counts
        8. Output tally counts, and go to 1
    B. Ratio
        1. Select input and output filenames
        2. Specify rectangular scene to be ratioed
            (a) If enough disk space, go to 2.B.3
            (b) Allow user to delete unwanted disk files and go to 2.B.1
        3. Select numerator and denominator channels and respective multiplicative weighting factors
        4. Load preprocessor as described
        5. Write output file title w. ratio description
        6. Perform ratioing, writing output on disk
        7. Go to 1

subset is four or less, up to 16 different signatures may be used; however, if the number of channels is between five and eight, only eight signatures may be used. The program will then load the Preprocessor linear combination and ratio hardware in such a manner that no linear combinations or ratios are formed, and then load the selected signatures into the Classifier hardware. Next the user must specify an input and an output filename. Then a scene specification for the rectangular area to be classified must be made. If enough disk space is available to hold the output classification information, a file of sufficient size will be created; otherwise, a smaller rectangular scene specification must be re-entered. The actual classification operation then begins and the classification results are stored on the disk. A running tally of the classification results for each class is kept and output at the end of the classification operation.

If the ratioing mode is selected, the user is asked to specify input and output filenames and a rectangular scene specification. If not enough disk space is available for the requested scene, a smaller rectangular scene specification must be made. Next the channel numbers for the numerator (N) and the denominator (D) channels must be specified. A scaling factor K, where $-32 < K < +31$, must also be specified. This scaling factor affects the ratio R which is formed such that

$$R = \frac{N}{D} * 2^{K}$$

After these parameters have been specified, the ratioing operation will be performed and an output file generated.

3.6 POST-CLASSIFICATION ANALYSIS (PANALY)

The post-classification data analysis program (PANALY) provides the user with a mechanism for elementary statistical analyses of classification results generated by processing data through the MIDAS classifier hardware (see Table 11). Calculation of class proportions and classification accuracy matrices upon selected subsets of a scene are the two primary functions.

Upon entry to this program, the user selects the data file to be analyzed. If this file is not a classification file as generated by the classification option of the program CLASFY, an existing filename of this type must be re-entered. The user may then select the subset of the scene to be analyzed by choosing those fields to be used from the current field definition list as formulated by the program FLDMAN. The four options available for field selection are: (1) enter up to 25 field names and up to 25 field classes to be used, (2) use all fields designated exclusively as "test fields" at time of creation in FLDMAN, (3) use all fields designated exclusively as "training fields" at time of creation in FLDMAN, or (4) use all test and training fields for analysis. Any or all of these options may be used for a given analysis task.

Next the user may optionally enter a scanner resolution element size in either hectares or acres. If this value is not entered, the scanner-type designation found in the data file title will provide the information for deciding the pixel ground area.

TABLE 11. OUTLINE OF THE POST-CLASSIFICATION ANALYSIS FUNCTIONS

1. Select input disk filename
    A. If not classification file, go to 1
2. Select fields to be used in analysis from one of the following modes
    A. Enter up to 25 field names and up to 25 field classes for analysis
    B. Use all test fields
    C. Use all training fields
    D. Use all training and test fields
3. Select scanner resolution element size (default taken from data file title)
4. Select type of analysis output
    A. Tally individual fields and/or
    B. Tally all fields together and/or
    C. Form confusion matrix based on all fields
5. Perform analysis processing and go to 1

Next the user must select the type of output to be generated on the line printer. Any or all of the following options may be selected: (1) tally output (or pixel count and ground area covered for each class) for each field separately, (2) tally output for all fields taken together, and/or (3) form classification accuracy, or confusion matrix based on all fields. This last option consists of an N × N matrix which shows the number (or proportion) of pixels of class A_i (as designated at creation time of each field used) that were actually classified as class A_j, where both i,j = 1 through N, for $0 < N \leq 16$.

Once these parameters have been entered by the user, the field definition list, stored on the RK05 system disk, is searched for the vertices of all fields included in the processing subset. The data pixels designated by these vertices are obtained from bulk disk storage, and the requested output information is computed and displayed on the line printer.

APPENDIX

THE USE OF APL IN HARDWARE SIMULATION

A.1 THE APL CLASSIFIER SIMULATION

An APL simulation was written of the MIDAS Classifier in order to validate and document the hardware design.

We can conceptually divide the pipeline into stages where each stage has the following characteristics. A stage has a memory which maintains the result of a computation (for 1 clock) while the next computation proceeds. While in the true MIDAS Classifier these memory components are edge-triggered devices, in our APL simulation they appear as master-slave devices. Thus, the output of stage 1 is computed and placed in a register called ST1M. At the end of the clock pulse, the contents of ST1M are transferred to ST1S (its slave) and stage 1 can recompute ST1M. While ST1M is being recomputed, ST1S is available for use by stage 2.

The nine classifier stages from the mean computation to exponent selection were simulated. The major objective (consistent with the goals stated above) was to correctly compute the contents of all memory elements and, thus, the state of the machine at each clock. Hence, the input and output of each stage should be identical with that of the hardware. Further, all other memory elements in the simulation (accumulators, overflow flip-flops, etc.) should maintain the exact state of the hardware. Within a stage, however, the simulation does not exactly reflect all processor activity. No attempt is made to precisely simulate all logic.

Consider stage 2 which multiplies incoming data by a variance coefficient. In hardware, this 8-bit by 12-bit multiply was performed by an array of multiplier chips. While it might be elegant to simulate these chips and their interconnections, the resultant multiplier simulation would be very complex and very slow. If this type of detailed approach had been used through the simulation, the resultant APL program would have been so slow that it would have been almost useless as a system debugging aid. The 8-bit by 12-bit multiply was actually simulated by converting both input bit strings to integers, performing an integer multiply, and converting back to a bit string. The result was identical to the result produced by the array of chips while the approach was radically different. It should be clear that, with this level of simulation, the program would be very useful in tracking down faulty multiply cards but useless for finding the error within the card.

The program listing is given on pages 115-122 and illustrated in Figure A-1. The comments should make the correspondence between stages and hardware components obvious. Note that after all nine stages have completed their operation, the end of a clock is simulated by transferring all master memory elements to their corresponding slaves and jumping to the beginning of the program. It should also be pointed out that this program (for reasons of efficiency) simulates only one of four identical pipes. Note that when a memory is specified in the first six stages, it is indexed by its pipe number.

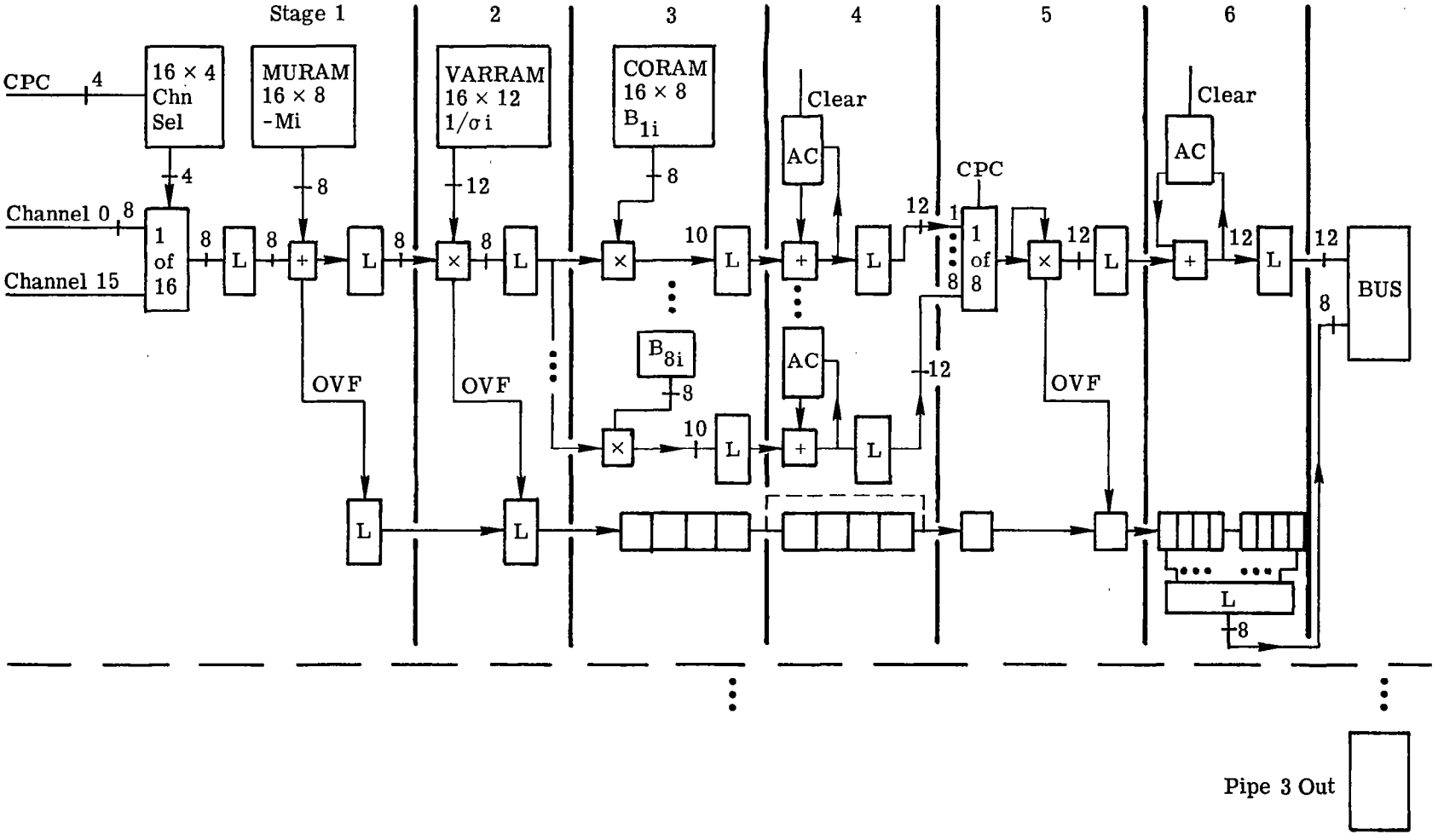


FIGURE A-1. APL MIDAS SIMULATION (Continued)

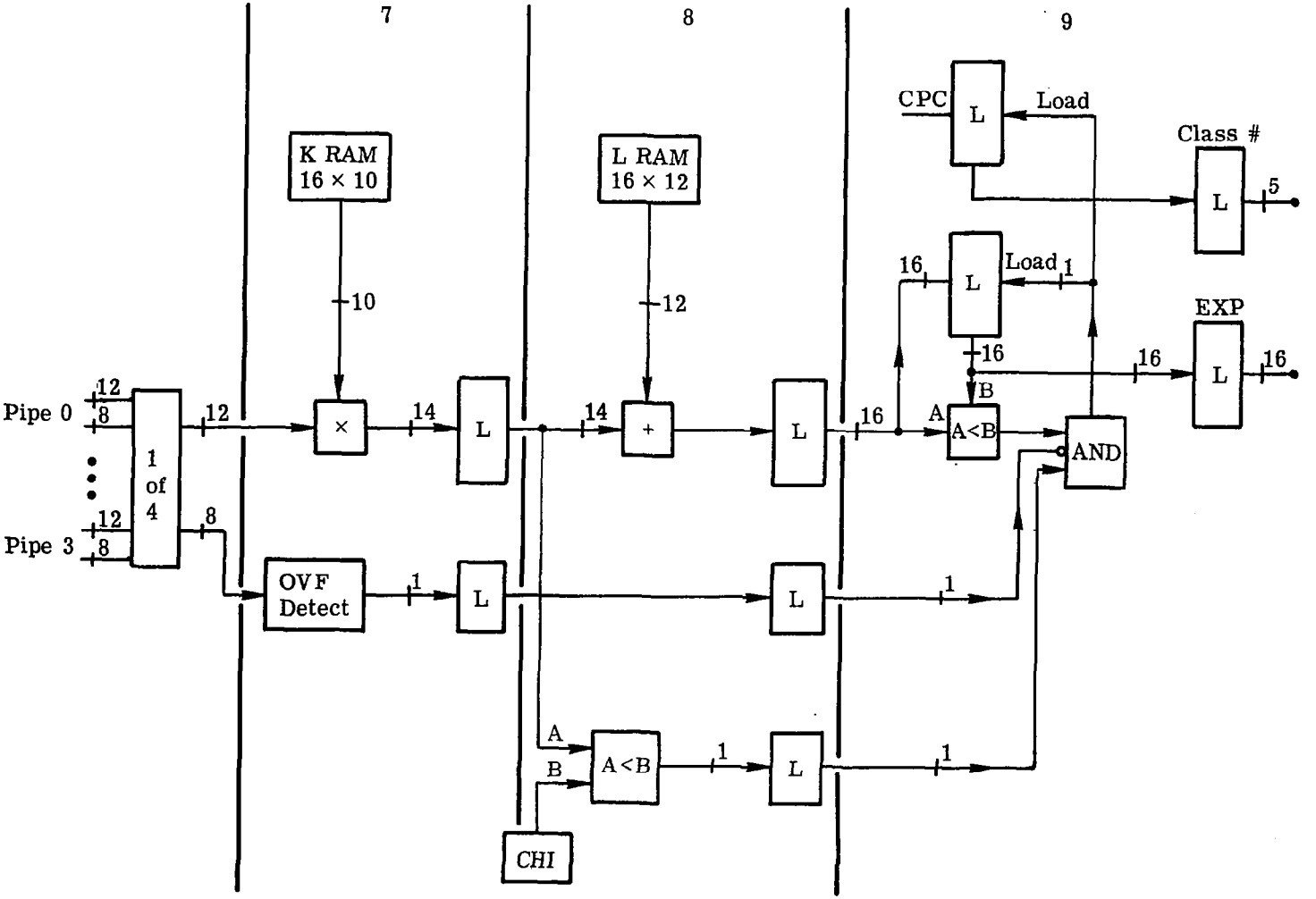


FIGURE A-1. APL MIDAS SIMULATION (Concluded)

In our simulation, the pipe number specified is always pipe zero. One could trivially change this program to do a full four-pipe simulation by indexing through the four pipes in sequence.

The first six stages of the pipeline (up through square accumulate) were actually simulated using the program described above. The simulation results were found to be exactly correct when compared to results obtained from the diagnostic system of the MIDAS with identical inputs. In fact, the simulation helped detect a small hardware failure in a portion of the Classifier thought to be operational. We feel that this simulation technique could be of great value in testing and maintaining the Classifier.

```
    ∇ MAIN
[1]   ⍝STAGE 1
[2]   ⍝MEAN STAGE
[3]   ⍝THIS STAGE IS REPLICATED 4 TIMES TO DO 4 CLASSIFICATIONS
[4]   ⍝EACH COPY HAS ITS OWN 16×8 MEMORY
[5]   ⍝MURAM←4 16 8⍴0
[6]   ⍝IRAM IS USED TO SELECT CHANNEL INPUTS
[7]   ⍝IRAM←16 4⍴0
[8]   ⍝
[9]   DATAIN←CHANNEL[((4×ICY)+2⊥CPC[2 T 3]);]
[10]  →(~∧/CPC)/N4
[11]  ICY←ICY+1
[12]  →(ICY<4)/N4
[13]  ICY←0
[14]  N4:ICY
[15]  ST1M[0;]←DATAIN ADD8 MURAM[0;2⊥CPC;]
[16]  OVSR[0;0]←OVF
[17]  ⍝
[18]  ⍝STAGE 2
[19]  ⍝VARIANCE STAGE
[20]  ⍝REPLICATED 4 TIMES
[21]  ⍝EACH HAS ITS OWN 16×12 MEMORY
[22]  ⍝VARRAM←4 16 12⍴0
[23]  ST2M[0;]←ST1S[0;]MPYET VARRAM[0;2⊥CPC;]
[24]  OVSR[0;1]←OVSR[0;1]∨OVF
[25]  ⍝
[26]  ⍝STAGE 3
[27]  ⍝MATRIX MULTIPLICATION
[28]  ⍝THERE ARE 32 SCALAR 8×8 MULTIPLIERS
[29]  ⍝IT TAKES 4,8,OR 16 T-STEPS TO FORM AN INNER PRODUCT
[30]  ⍝EACH MULTIPLIER HAS ITS OWN 16×8 RAM
[31]  ⍝CORAM←4 8 16 8⍴0
[32]  ⍝THERE ARE EIGHT OUTPUTS FOR EACH CLASSIFICATION
[33]  ⍝ST3M←4 8 10⍴0
[34]  ⍝I STEPS THROUGH THE 8 CONCURRENT MULTIPLICATIONS
[35]  I←0
[36]  MPLP:ST3M[0;I;]←ST2S[0;]MPYEE CORAM[0;I;2⊥CPC;]
[37]  ST3OVM[0;I]←OVF
[38]  I←I+1
[39]  →(I≤7)/MPLP
[40]  ⍝STAGE 4 IS THE ADDITION LOOP FOR THE INNER PRODUCT.
[41]  ⍝MMACM←4 8 12⍴0
[42]  ⍝THE INNER PRODUCT IS ACCUMULATED IN THE MMAC.
[43]  ⍝MMACS←4 8 12⍴0
[44]  I←0
[45]  MPACLP:MMACM[0;I;]←MMACS[0;I;]ADD12 ST3S[0;I;]
[46]  I←I+1
[47]  →(I≤7)/MPACLP
[48]  ⍝ LOAD THE MULTIPLY OUTPUT.
[49]  →(~MTXMXI)/(NLMO)
[50]  I←0
[51]  LMO:ST4M[0;I;]←MMACS[0;I;]ADD12 ST3S[0;I;]
[52]  I←I+1
[53]  →(I≤7)/LMO
[54]  MMACM←MMACS←CLEAR MMACM
[55]  ⍝STAGE 5.
[56]  ⍝STAGE 5 SQUARES THE Y'S
[57]  ⍝THERE ARE 4 SUCH MULTIPLIERS
[58]  ⍝ST5M←4 12⍴0
[59]  NLMO:SQIN←ST4S[0;MTXOP;]
[60]  OVF←~((∧/SQIN[0 T 3])∨~(∨/SQIN[0 T 3]))
[61]  SQIN←SQIN[0],SQIN[4 T 11],0
[62]  ST5M[0;]←SQIN MPYTT SQIN
[63]  OVSR[0;11]←OVSR[0;11]∨OVF
[64]  ⍝STAGE 6.
[65]  ⍝SQACM← 4 16 ⍴0
```

```
[66]  SQACM[0;]←SQACS[0;]SQAC(0 0 0 ,ST5S[0;],0)
[67]  ⍝DO WE LOAD THE OUTPUT?
[68]  →(~SAMXI)/(LSO+4)
[69]  LSO:SQBS←SQACS[0;]SQAC(0 0 0 ,ST5S[0;],0)
[70]  ST6M[0;]←SQBS[0 T 11]
[71]  SQACS←SQACM←CLEAR SQACM
[72]  ⍝STAGE 7
[73]  ⍝ST7M←14 ⍴0
[74]  ⍝STAGE 7 MULTIPLIES BY THE CONSTANT K
[75]  MPYKOUT←ST6S(SAMXOP;)MPY1210 KRAM[2⊥CPC;]
[76]  ST7M←MPYKOUT[2 T 15]
[77]  ⍝MPY1210 IS A 12 BY 10 MULTIPLIER.
[78]  DETECT←∨/(OVLS[SAMXOP;0 T 3],(ECHAN∧OVLS[SAMXOP;4 T 7]))
[79]  ST7OVM←ST7OVS∨DETECT
[80]  ⍝STAGE 8
[81]  ⍝STAGE 8 SUBTRACTS THE LOG TERM AND COMPUTES THE CHI BIT
[82]  ST8M←(0 0 ,ST7S)ADD16(0,LRAM[(2⊥CPC);], 0 0 0)
[83]  ST8OVM←ST7OVS
[84]  ST8CHIM←CHIRAM CHICOMP ST7S
[85]  ⍝ST8M←16 ⍴ 0
[86]  ⍝STAGE 9
[87]  STAGE 9 KEEPS THE SMALLEST EXPONENT
[88]  ⍝ALONG WITH ITS ASSOCIATED CLASS NUMBER.
[89]  ⍝ST9M←16 ⍴ 0
[90]  ⍝BESTEXP←16 ⍴ 0
[91]  CMP←ST8S EXPCOMP BESTEXPS
[92]  LOAD←CMP∧ST8OVS∧ST8CHIS
[93]  CLR←(((2⊥CPC)=14)∧~ECHAN)∨(((2⊥CPC)=8)∧ECHAN)
[94]  BESTEXPM←(~LOAD∧BESTEXPS)∨(LOAD∧ST8S)
[95]  CLASSNOM←(ECHAN∧(0,CPC[0 T 2]))∨(LOAD∧CLASSNO)
[96]  →(~CLR)/NCLR
[97]  ⍝WE MUST LOAD THE OUTPUT LATCH AND CLEAR
[98]  ST9M←BESTEXPM
[99]  ST9CNM←CLASSNOS
[100] CLASSNOM←CLASSNOS←4⍴0
[101] BESTEXPM←BESTEXPS←16⍴0
[102] ⍝MASTER TO SLAVE TRANSFERS.
[103] SHOV
[104] ST1S←ST1M
[105] ST2S←ST2M
[106] ST3S←ST3M
[107] ST4S←ST4M
[108] ST5S←ST5M
[109] SQACS←SQACM
[110] ST6S←ST6M
[111] ST7S←ST7M
[112] ST7OVS←ST7OVM
[113] ST8S←ST8M
[114] ST8OVS←ST8OVM
[115] ST9S←ST9M
[116] BESTEXPS←BESTEXPM
[117] CLASSNOS←CLASSNOM
[118] ST8CHIS←ST8CHIM
[119] OVLS←OVLM
[120] MMACS←MMACM
[121] CPC←INC CPC
[122] DISPLAY
[123] →1
      ∇
```

```
      ∇ABS[□]
    ∇ Z←ABS X;LEN
[1]   LEN←ρX
[2]   →(X[0])/NEG
[3]   Z←2⊥X
[4]   →0
[5]   NEG:Z←2⊥COMP X
[6]   SIN←~SIN
    ∇

      ∇ADD12[□]
    ∇ Z←X ADD12 Y;X1;Y1;Z1
[1]   X1←2⊥X
[2]   Y1←2⊥(Y[0],Y[0],Y)
[3]   Z1←X1+Y1
[4]   Z←(12ρ2)⊤Z1
[5]   OVF←(X[0]∧Y[0]∧~Z[0])∨((~X[0])∧(~Y[0])∧Z[0])
    ∇

      ∇ADD8[□]
    ∇ Z←X ADD8 Y;X1;Y1;Z1
[1]   X1←2⊥X
[2]   Y1←2⊥Y
[3]   Z1←X1+Y1
[4]   Z←(8ρ2)⊤Z1
[5]   OVF←(X[0]∧Y[0]∧(~Z[0]))∨((~X[0])∧(~Y[0])∧Z[0])
    ∇

      ∇CHICOMP[□]
    ∇ Z←X CHICOMP Y;X1;Y1
[1]   X1←2⊥X
[2]   Y1←2⊥Y
[3]   Z←X>Y
    ∇

      ∇CLEAR[□]
    ∇ Z←CLEAR X
[1]   Z←(ρX)ρ0
    ∇

      ∇COMP[□]
    ∇ Z←COMP X
[1]   Z←INC(~X)
```

```
     ∇ DISPLAY
[1]    ⍞←'CPC'
[2]    □←0⍴0
[3]    CPC
[4]    □←0⍴0
[5]    □←0⍴⍬
[6]    ⍞←'STAGE1'
[7]    □←0⍴0
[8]    PRINT ST1M
[9]    □←0⍴0
[10]   ⍞←'STAGE2'
[11]   □←0⍴0
[12]   PRINT ST2M
[13]   □←0⍴0
[14]   ⍞←'STAGE3'
[15]   □←0⍴0
[16]   PRINT ST3M
[17]   □←0⍴0
[18]   ⍞←'MMACM'
[19]   □←0⍴0
[20]   PRINT MMACM
[21]   □←0⍴0
[22]   ⍞←'STAGE4'
[23]   □←0⍴0
[24]   PRINT ST4M
[25]   □←0⍴0
[26]   ⍞←'STAGE5'
[27]   □←0⍴0
[28]   PRINT ST5M
[29]   □←0⍴0
[30]   ⍞←'SQACM'
[31]   □←0⍴0
[32]   PRINT SQACM
[33]   □←0⍴0
[34]   ⍞←'STAGE6'
[35]   □←0⍴0
[36]   PRINT ST6M
[37]   □←0⍴0
[38]   ⍞←'OVSR'
[39]   □←0⍴0
[40]   PRINT OVSR
[41]   □←0⍴0
[42]   □←0⍴0
     ∇

      ∇EXPCOMP[□]
     ∇ Z←X EXPCOMP Y;X1;Y1
[1]    X1←2⊥X
[2]    Y1←2⊥Y
[3]    Z←X<Y
     ∇

      ∇INC[□]
     ∇ Z←INC X;TEMP;LEN
[1]    TEMP←2⊥X
[2]    LEN←⍴X
[3]    TEMP←TEMP+1
[4]    Z←(LEN⍴2)⊤TEMP
     ∇
```

```
      ∇INPUT[□]
    ∇ INPUT;ADVEC;ADDIN;MEMADDR;CADDR;DATA
[1]   ⍝READ DATA INTO THE FUNCTIONAL MEMORIES OF THE CLASSIFIER
[2]   INLP:ADDIN←(6⍴10)⊤□
[3]   ADVEC←(16⍴2)⊤(8⊥ADDIN)
[4]   ⍝THE ADDRESS HAS BEEN CONVERTED TO A BINARY BIT STRING
[5]   MEMADDR←2⊥ADVEC[12 T 15]
[6]   CADDR←2⊥ADVEC[8 T 11]
[7]   PIPE←2⊥ADVEC[4 T 7]
[8]   →(∨/ADVEC[0 T 5])/ERROR
[9]   ⍝THE TWO HIGH ORDER OCTAL DICITS SHOULD BE ZERO.
[10]  ⍝READ THE DATA
[11]  DATA←(16⍴2)⊤(8⊥((6⍴10)⊤□))
[12]  ⍝WHICH MEMORY ARE WE LOADING?
[13]  →(CADDR=0)/MU
[14]  →(CADDR=1)/SIG
[15]  →((CADDR≥2)∧(CADDR≤9))/MPY
[16]  →(CADDR=10)/KSQ
[17]  →(CADDR=11)/LOG
[18]  →ERROR
[19]  ⍝LOAD THE MEAN RAM
[20]  MU:MURAM[PIPE;MEMADDR;]←~DATA[4 T 11]
[21]  →INLP
[22]  SIG:VARRAM[PIPE;MEMADDR;]←~DATA[4 T 15]
[23]  →INLP
[24]  MPY:CADDR←CADDR-2
[25]  CORAM[PIPE;CADDR;MEMADDR;]←~DATA[4 T 11]
[26]  →INLP
[27]  KSQ:KRAM[PIPE;]←~DATA[4 T 15]
[28]  →INLP
[29]  LOG:LRAM[PIPE;]←DATA[4 T 15]
[30]  →INLP
[31]  ERROR:□←'ERROR'
[32]  →INLP
    ∇

      ∇LCHAN[□]
    ∇ LCHAN
[1]   I←0
[2]   CHANNEL[I;]←COMP CHANNEL[I;]
[3]   I←I+1
[4]   →2
    ∇

      ∇LI[□]
    ∇ LI
[1]   I←0
[2]   IRAM[I;]←(4⍴2)⊤I
[3]   I←I+1
[4]   →2
    ∇

      ∇MPYEE[□]
    ∇ Z←X MPYEE Y;Y1;X1;Z1;TEMP
[1]   ⍝MULTIPLY ROUTINE FOR MATRIX PRODUCT
[2]   ⍝AN EIGHT BY EIGHT PRODUCT HIGH 10 BITS ARE SELECTED
[3]   SIN←0
[4]   X1←ABS X
[5]   Y1←ABS Y
[6]   Z1←X1×Y1
[7]   TEMP←(16⍴2)⊤Z1
[8]   →(~SIN)/PLUSEE
[9]   TEMP←COMP TEMP
[10]  PLUSEE:Z←TEMP[⍳10]
[11]  OVF←0
    ∇
```

```
      ∇MPYET[□]
    ∇ Z←X MPYET Y;X1;Y1;Z1;TEMP
[1]   ⍝EIGHT BY TEN MULTIPLICATION
[2]   SIN←0
[3]   X1←ABS X
[4]   Y1←ABS Y
[5]   Z1←X1×Y1
[6]   TEMP←(20⍴2)⊤Z1
[7]   ⍝FINAL PRODUCT BITS ARE SELECTED FROM TEMP
[8]   →(~SIN)/PLUSET
[9]   TEMP←COMP TEMP
[10]  PLUSET:Z←TEMP[0],TEMP[6 T 12]
[11]  ⍝OVERFLOW IS COMPUTED
[12]  OVF←~((∧/TEMP[0 T 5])∨~(∨/TEMP[0 T 5]))
    ∇

      ∇MPYTT[□]
    ∇ Z←X MPYTT Y;X1;Y1;Z1;TEMP
[1]   ⍝ TEN BY TEN MULTIPLY
[2]   X1←ABS X
[3]   Y1←ABS Y
[4]   Z1←X1×Y1
[5]   TEMP←(20⍴2)⊤Z1
[6]   Z←TEMP[1 T 12]
    ∇

      ∇MPY1210[□]
    ∇ Z←X MPY1210 Y;X1;Y1;Z1
[1]   X1←2⊥X
[2]   Y1←2⊥Y
[3]   Z1←X1×Y1
[4]   Z←(22⍴2)⊤Z1
    ∇

      ∇MTXMXI[□]
    ∇ Z←MTXMXI
[1]   Z←((~CPC[1])∨(~ECHAN))∧CPC[2]∧(~CPC[3])
    ∇

      ∇MTXOP[□]
    ∇ Z←MTXOP
[1]   Z←(2⊥CPC[1 T 3])+5
[2]   →(Z<8)/0
[3]   Z←Z-8
    ∇


      ∇OUT[□]
    ∇ Z←OUT X;LEN;TEMP
[1]   LEN←⍴X
[2]   LEN←⌈(LEN÷3)
[3]   TEMP←2⊥X
[4]   TEMP←(LEN⍴8)⊤TEMP
[5]   Z←10⊥TEMP
    ∇
```

```
    ∇ PRINT X;IMAX;JMAX;I;J;DIM;K;KMAX;OVEC
[1]   ⍝ROUTINE TO DISPLAY MEMORIES.
[2]   CASE←⍴⍴X
[3]   ⍝FIND THE RANK OF THE MEMORY
[4]   →(CASE≠1)/SNOT
[5]   ⍝A SINGLY DIMENSIONED ARRAY (LATCH)
[6]   ⎕←OUT X
[7]   →0
[8]   SNOT:→(CASE≠2)/DNOT
[9]   ⍝ A DOUBLY DIMENSIONED MEMORY
[10]  DIM←⍴X
[11]  IMAX←DIM[0]
[12]  I←0
[13]  OVEC←0⍴0
[14]  DLP:OVEC←OVEC,OUT X[I;]
[15]  I←I+1
[16]  →(I<IMAX)/DLP
[17]  ⎕←OVEC
[18]  →0
[19]  DNOT:→(CASE≠3)/TNOT
[20]  ⍝A TRIPLY DIMENSIONED ARRAY
[21]  DIM←⍴X
[22]  JMAX←DIM[0]
[23]  IMAX←DIM[1]
[24]  KMAX←1
[25]  K←1
[26]  QLP:J←0
[27]  TLP:OVEC←0⍴0
[28]  I←0
[29]  TLPP:→(CASE≠3)/SKPP
[30]  OVEC←OVEC,OUT X[J;I;]
[31]  →SKPP+1
[32]  SKPP:OVEC←OVEC,OUT X[K;J;I;]
[33]  I←I+1
[34]  →(I<IMAX)/TLPP
[35]  ⎕←OVEC
[36]  J←J+1
[37]  →(J<JMAX)/TLP
[38]  ⎕←0⍴0
[39]  K←K+1
[40]  →(K<KMAX)/QLP
[41]  →0
[42]  ⍝QUADRUPLY DIMENSIONED ARRAY
[43]  TNOT:DIM←⍴X
[44]  KMAX←DIM[0]
[45]  JMAX←DIM[1]
[46]  IMAX←DIM[2]
[47]  K←0
[48]  →QLP
[49]  4
    ∇
```

```
    ∇ RESET
[1]   ST1M←CLEAR ST1M
[2]   ST1S←CLEAR ST1S
[3]   ST2M←CLEAR ST2M
[4]   ST2S←CLEAR ST2S
[5]   ST3M←CLEAR ST3M
[6]   ST3S←CLEAR ST3S
[7]   MMACM←CLEAR MMACM
[8]   MMACS←CLEAR MMACS
[9]   ST4M←CLEAR ST4M
[10]  CPC←CLEAR CPC
[11]  ST4S←CLEAR ST4S
[12]  ST5M←CLEAR ST5M
[13]  ST5S←CLEAR ST5S
[14]  ST6M←CLEAR ST6M
[15]  ST6S←CLEAR ST6S
[16]  OVLM←CLEAR OVLM
[17]  OVLS←CLEAR OVLS
[18]  OVSR←CLEAR OVSR
[19]  SQACM←CLEAR SQACM
[20]  SQACS←CLEAR SQACS
[21]  ICY←0
    ∇

      ∇SAMXI[□]
    ∇ Z←SAMXI
[1]   Z←((~CPC[1])∨(~ECHAN))∧CPC[2]∧CPC[3]
    ∇

      ∇SHOV[□]
    ∇ SHOV
[1]   ⍝SHIFT ROUTINE FOR OVERFLOW SHIFT REGISTER/
[2]   OVLM[0;]←OVSR[0;12 T 19]
[3]   →(~ECHAN)/FOUR
[4]   OVSR[0;]←0,OVSR[0;0 T 18]
[5]   →(FOUR+2)
[6]   FOUR:OVSR[0;10 T 19]←OVSR[0;5],OVSR[0;10 T 18]
[7]   OVSR[0;0 T 5]←0,OVSR[0;0 T 4]
[8]   OVSR[0; 20 21]←DETECT,OVSR[0;20]
    ∇

      ∇SQACL□
           ∨
           [□]
    ∇ Z←X SQAC Y;X1;Y1;Z1
[1]   X1←2⊥X
[2]   Y1←2⊥Y
[3]   Z1←X1+Y1
[4]   Z←(16⍴2)⊤Z1
    ∇

      ∇T[□]
    ∇ Z←X T Y
[1]   Z←(⍳((Y-X)+1))+X
    ∇
```

A.2 THE APL PREPROCESSOR SIMULATION

A Simulation of the ratio module of the pre-processor was also written in APL. The hardware and block diagram are described in the text in Section 2.4.3. A listing of the simulation and a sample run are given in the following pages.

```
    ∇ MAIND
[1]   ⍝PRENORMALIZATION FOR DIVISION
[2]   ⍝INPUTS ARE DIVDND, DIVSOR
[3]   ⍝PRENORMALIZATION IS DONE IN ONE CLOCK
[4]   DSGN←DIVSORS[0]
[5]   S0←~DSGN
[6]   CX←DSGN
[7]   ⍝TAKE THE ABSOLUTE VALUE OF THE DIVISOR
[8]   ⍝NEGATIVE RESULT IMPLIES OVERFLOW IN ABSOLUTE VALUE COMPUTATION.
[9]   BUS←12⍴0 ACL12 DIVSORS
[10]  ABSOV←BUS[0]
[11]  ⍝PRIOE IMITATES THE ACTION OF THE PRIORITY ENCODERS.
[12]  SC←PRIOE(~BUS[1 T 11])
[13]  ⍝SC IS A 5 BIT VECTOR A0,A1,A2,A3,OVF
[14]  ⍝SHIFTING IS DONE NEXT
[15]  BUS←BUS[(SC[0]×8)T 11],(SC[0]×8)⍴0
[16]  ⍝THE ABOVE LINE SHIFTS 0 OR 8 POSITIONS
[17]  SN←2⊥SC[1 T 3]
[18]  BUS←~(BUS[SN T 11],SN⍴0)
[19]  ⍝SHIFT 0 THROUGH 7 WITH A COMPLIMENT.
[20]  ⍝INVERT SHIFTED RESULT
[21]  DST0M←~BUS
[22]  ⍝EXPONENT COMPUTATION FOLLOWS
[23]  CYF←1
[24]  EBUS←SCRAM[2 T 7]ADD6(0 0 ,SC[0 T 3])
[25]  PRST0M←DIVDNDS
[26]  ⍝THE DIVIDEND BECOMES THE FIRST PARTIAL REMAINDER
[27]  QEST0M←EBUS
[28]  DOST0M←~(SC[4]∧(~ABSOV))
[29]  DSST0M←DSGN
[30]  ⍝NON-RESTORING DIVISION
[31]  ⍝DONE IN THREE STAGES
[32]  ⍝LATCHING PROVIDED FOR DIVISOR,PARTIAL REMAINDER,AND QUOTIENT
[33]  ⍝STAGE 1
[34]  ⍝QUOTIENT IN QST0S←12 ⍴ 0
[35]  ⍝DIVIDEND IN DST0S←12 ⍴ 0
[36]  ⍝PARTIAL REMAINDER IN PRST0S←12 ⍴0
[37]  DSST1M←DSST0S
[38]  DOST1M←DOST0S
[39]  QEST1M←QEST0S
[40]  DST1M←DST0S
[41]  QEST1M←QEST0S
[42]  ⍝CARRY DIVISOR ALONG.
[43]  ⍝S0I IS PASSED ONTO THE DIVIDE CARD.
[44]  S0I←PRST0S[0]
[45]  ⍝NO SHIFT THIS TIME.
[46]  PRST1M←PRST0S DIV DST0S
[47]  ⍝COMPUTE 4 QUOTIENT BITS AND A NEW PARTIAL REMAINDER
[48]  QST1M←QST0S[4 T 11],QOUT
[49]  ⍝DIVIDE STAGE 2
[50]  DSST2M←DSST1S
```

```
[51]  DOST2M←DOST1S
[52]  QEST2M←QEST1S
[53]  DST2M←DST1S
[54]  QEST2M←QEST1S
[55]  SOI←PRST1S[0]
[56]  PRST2M←(PRST1S[1 T 11],0)DIV DST1S
[57]  QST2M←QST1S[4 T 11],QOUT
[58]  ⍝DIVIDE STAGE THREE
[59]  DSST3M←DSST2S
[60]  DOST3M←DOST2S
[61]  QEST3M←QEST2S
[62]  DST3M←DST2S
[63]  QEST3M←QEST2S
[64]  SOI←PRST2S[0]
[65]  PRST3M←(PRST2S[1 T 11],0)DIV DST2S
[66]  QST3M←QST2S[4 T 11],QOUT
[67]  ⍝POST NORMALIZATION TAKES PLACE IN ONE CLOCK.
[68]  ⍝THE INPUTS ARE QST3S,QEST3S,DSST3S,DOST3S.
[69]  ⍝FIRST WE MUST TAKE THE ABS VALUE OF Q AND COMPUTE QSGN.
[70]  MSB←~QST3S[0]
[71]  QSGN←MSB*DSST3S
[72]  S0←QST3S[0]
[73]  CX←MSB
[74]  LSB←~PRST3S[0]
[75]  QBUS←(12⍴0)ACL12(QST3S[1 T 11],LSB)
[76]  ABSOV←COUT
[77]  ⍝THIS TWO'S COMPLIMENT NUMBER IS CONVERTED TO SIGN MAGNITUDE
[78]  ⍝A SHIFT DIRECTION AND AMOUNT.
[79]  CX←QEST3S[0]
[80]  S0←~CX
[81]  ⍝QEST3S HAS A TWO'S COMP SHIFT COUNT.
[82]  LEFT←~EBUS[0]
[83]  CX←~LEFT
[84]  S0←LEFT
[85]  ADDOUT←(8⍴0)ACL8(0 0 ,QEST3S)
[86]  EBUS←ADDOUT[2 T 7]
[87]  ⍝SHIFT 0 OR EIGHT.
[88]  EIGHT←EBUS[2]
[89]  STR←(~LEFT)∧EIGHT
[90]  QBUS←(~STR)∧((EIGHT∧(QBUS[8 T 11],(8⍴0)))∨((~EIGHT)∧QBUS))
[91]  QOVF1←~((EIGHT∧QBUS[0 T 7])∨((~EIGHT)∧8⍴0))
[92]  QBUS1←QBUS,4⍴0
[93]  ⍝SN IS A SHIFT COUNT
[94]  SN←2⊥EBUS[3 T 5]
[95]  QET←~((~LEFT)∧((SN⍴0),QBUS[0 T(7-SN)]))∨((LEFT)∧(QBUS1[SN T(7+SN)]))
[96]  ⍝SHIFT 0 THROUGH SEVEN RIGHT OR LEFT.
[97]  QOVF2←(~((LEFT)∧(((7-SN)⍴0),QBUS[0 T(SN-1)])))
[98]  ⍝COMPUTE THE FINAL OVERFLOW.
[99]  DOST4M←~(DOST3S∧(~ABSOV)∧(~((~∧/QOVF1)∨(~∧/QOVF2)))∧~∨/EBUS[0 T 1])
[100] CX←QSGN
```

```
[101] S0←QSGN
[102] QETS←(8ρ0)ACL8 QET
[103] QST4M←(COUT≠QSGN),QETS
[104] ⍝MASTER TO SLAVE TRANSFERS
[105] DSST0S←DSST0M
[106] DOST0S←DOST0M
[107] QEST0S←QEST0M
[108] PRST0S←PRST0M
[109] DST0S←DST0M
[110] QEST0S←QEST0M
[111] DSST1S←DSST1M
[112] DOST1S←DOST1M
[113] QEST1S←QEST1M
[114] QST1S←QST1M
[115] DST1S←DST1M
[116] PRST1S←PRST1M
[117] QEST1S←QEST1M
[118] DSST2S←DSST2M
[119] DOST2S←DOST2M
[120] QEST2S←QEST2M
[121] QST2S←QST2M
[122] DST2S←DST2M
[123] QEST2S←QEST2M
[124] PRST2S←PRST2M
[125] DSST3S←DSST3M
[126] DOST3S←DOST3M
[127] QST3S←QST3M
[128] QEST3S←QEST3M
[129] PRST3S←PRST3M
```

```
    ∇ Z←X ACL12 Y;X1;Y1;Z1;TEMP
[1]   Y←(S0∧Y)∨((~S0)∧~Y)
[2]   X1←2⊥X
[3]   Y1←2⊥Y
[4]   Z1←X1+Y1+CX
[5]   TEMP←(13ρ2)⊤Z1
[6]   COUT←TEMP[0]
[7]   Z←TEMP[1 T 12]


    ∇ Z←X ACL8 Y;X1;Y1;Z1;TEMP
[1]   Y←(S0∧Y)∨((~S0)∧~Y)
[2]   X1←2⊥X
[3]   Y1←2⊥Y
[4]   Z1←X1+Y1+CX
[5]   TEMP←(9ρ2)⊤Z1
[6]   Z←TEMP[1 T 8]
[7]   COUT←TEMP[0]


    ∇ Z←X ADD6 Y;X1;Y1;Z1
[1]   X1←2⊥X
[2]   Y1←2⊥Y
[3]   Z1←X1+Y1+CYF
[4]   Z←(6ρ2)⊤Z1


    ∇ RPP4←PR DIV DIVISOR
[1]   QOUT←~S0I
[2]   ⍝S0 DETERMINES WHETHER WE ADD OR SUBTRACT.
[3]   S0←S0I
[4]   CX←~S0
[5]   PR←PR ACL12 DIVISOR
[6]   S0←PR[0]
[7]   CX←~S0
[8]   PR←PR[1 T 11],0
[9]   ⍝ THE PARTIAL REMAINDER IS SHIFTED.
[10]  QOUT←QOUT,~S0
[11]  PR←PR ACL12 DIVISOR
[12]  S0←PR[0]
[13]  CX←~S0
[14]  PR←PR[1 T 11],0
[15]  QOUT←QOUT,~S0
[16]  PR←PR ACL12 DIVISOR
[17]  S0←PR[0]
[18]  CX←~S0
[19]  PR←PR[1 T 11],0
[20]  QOUT←QOUT,~S0
[21]  PR←PR ACL12 DIVISOR
[22]  RPP4←PR

    ∇ SC←PRIOE X;OVFC;SCN
[1]   SCN←0
[2]   LP:→((~X[SCN])∨(SCN=10))/OUT
[3]   SCN←SCN+1
[4]   →LP
[5]   ⍝FIND THE HIGHEST ORDER ZERO BIT
[6]   OUT:SC←((4ρ2)⊤SCN),~X[SCN]
[7]   ⍝IF X[SCN]=1 WE MUST HAVE OVERFLOW (STRING OF ALL ONES)


    ∇ Z←X T Y
[1]   Z←(⍳((Y-X)+1))+X
```

```
      SCRAM
1  1  1  1  1  1  1  0
      DIVDNDS
0  0  1  0  0  0  1  1  0  0  0  0
      DIVSORS
1  1  1  0  0  1  0  0  0  0  0  0
      MAIND
      MAIND
      MAIND
      MAIND
      TΔMAIND←(⍳129)+1
      MAIND
MAIND[4] 1
MAIND[5] 0
MAIND[6] 1
MAIND[9] 0  0  0  1  1  1  0  0  0  0  0  0
MAIND[10] 0
MAIND[12] 0  0  1  0  1
MAIND[15] 0  0  0  1  1  1  0  0  0  0  0  0
MAIND[17] 2
MAIND[18] 1  0  0  0  1  1  1  1  1  1  1  1
MAIND[21] 0  1  1  1  0  0  0  0  0  0  0  0
MAIND[23] 1
MAIND[24] 0  0  0  0  0  1
MAIND[25] 0  0  1  0  0  0  1  1  0  0  0  0
MAIND[27] 0  0  0  0  0  1
MAIND[28] 1
MAIND[29] 1
MAIND[37] 1
MAIND[38] 1
MAIND[39] 0  0  0  0  0  1
MAIND[40] 0  1  1  1  0  0  0  0  0  0  0  0
MAIND[41] 0  0  0  0  0  1
MAIND[44] 0
MAIND[46] 1  1  0  0  1  0  0  0  0  0  0  0
MAIND[48] 0  0  0  0  0  0  0  0  1  0  0  1
MAIND[50] 1
MAIND[51] 1
MAIND[52] 0  0  0  0  0  1
MAIND[53] 0  1  1  1  0  0  0  0  0  0  0  0
MAIND[54] 0  0  0  0  0  1
MAIND[55] 1
MAIND[56] 1  0  0  1  0  0  0  0  0  0  0  0
MAIND[57] 0  0  0  0  1  0  0  1  0  1  0  0
MAIND[59] 1
MAIND[60] 1
MAIND[61] 0  0  0  0  0  1
MAIND[62] 0  1  1  1  0  0  0  0  0  0  0  0
MAIND[63] 0  0  0  0  0  1
MAIND[64] 1
MAIND[65] 1  0  0  1  0  0  0  0  0  0  0  0
MAIND[66] 1  0  0  1  0  1  0  0  0  0  0  0
MAIND[70] 0
MAIND[71] 1
MAIND[72] 1
```

```
MAIND[73] 0
MAIND[74] 0
MAIND[75] 0  0  1  0  1  0  0  0  0  0  0  0
MAIND[76] 0
MAIND[79] 0
MAIND[80] 1
MAIND[82] 1
MAIND[83] 0
MAIND[84] 1
MAIND[85] 0  0  0  0  0  0  0  1
MAIND[86] 0  0  0  0  0  1
MAIND[88] 0
MAIND[89] 0
MAIND[90] 0  0  1  0  1  0  0  0  0  0  0  0
MAIND[91] 1  1  1  1  1  1  1  1
MAIND[92] 0  0  1  0  1  0  0  0  0  0  0  0  0  0  0  0
MAIND[94] 1
MAIND[95] 1  0  1  0  1  1  1  1
MAIND[97] 1  1  1  1  1  1  1  1
MAIND[99] 0
MAIND[100] 1
MAIND[101] 1
MAIND[102] 1  0  1  1  0  0  0  0
MAIND[103] 1  1  0  1  1  0  0  0  0
MAIND[105] 1
MAIND[106] 1
MAIND[107] 0  0  0  0  0  1
MAIND[108] 0  0  1  0  0  0  1  1  0  0  0  0
MAIND[109] 0  1  1  1  0  0  0  0  0  0  0  0
MAIND[110] 0  0  0  0  0  1
MAIND[111] 1
MAIND[112] 1
MAIND[113] 0  0  0  0  0  1
MAIND[114] 0  0  0  0  0  0  0  0  1  0  0  1
MAIND[115] 0  1  1  1  0  0  0  0  0  0  0  0
MAIND[116] 1  1  0  0  1  0  0  0  0  0  0  0
MAIND[117] 0  0  0  0  0  1
MAIND[118] 1
MAIND[119] 1
MAIND[120] 0  0  0  0  0  1
MAIND[121] 0  0  0  0  1  0  0  1  0  1  0  0
MAIND[122] 0  1  1  1  0  0  0  0  0  0  0  0
MAIND[123] 0  0  0  0  0  1
MAIND[124] 1  0  0  1  0  0  0  0  0  0  0  0
MAIND[125] 1
MAIND[126] 1
MAIND[127] 1  0  0  1  0  1  0  0  0  0  0  0
MAIND[128] 0  0  0  0  0  1
MAIND[129] 1  0  0  1  0  0  0  0  0  0  0  0
```