47888

NASA CR-132464
ERIM 195800-25-F

MIDAS, PROTOTYPE MULTIVARIATE INTERACTIVE DIGITAL ANALYSIS SYSTEM — PHASE I

Volume II: Diagnostic System

by

F. J. Kriegler et al.
Infrared and Optics Division

ENVIRONMENTAL RESEARCH INSTITUTE OF MICHIGAN
FORMERLY WILLOW RUN LABORATORIES, THE UNIVERSITY OF MICHIGAN

August 1974

prepared for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Langley Research Center
Hampton, VA 23665
Contract No. NAS1-11979

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-132464		

4. Title and Subtitle	5. Report Date
MIDAS, PROTOTYPE MULTIVARIATE INTERACTIVE DIGITAL ANALYSIS SYSTEM — PHASE I VOLUME II: DIAGNOSTIC SYSTEM	August 1974
	6. Performing Organization Code
7. Author(s) F. J. Kriegler et al.	8. Performing Organization Report No. 195800-25-F
9. Performing Organization Name and Address Infrared and Optics Division Environmental Research Institute of Michigan P. O. Box 618 Ann Arbor, MI	10. Work Unit No.
	11. Contract or Grant No. NAS 1-11979
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546	Final Report, October 1972 through February 1974
	14. Sponsoring Agency Code

15. Supplementary Notes

Volume II of III

16. Abstract

The MIDAS System is a third-generation,fast,multispectral recognition system able to keep pace with the large quantity and high rates of data acquisition from 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 turnaround time and significant gains in throughput. This goal is elaborated in this report as an objective of the Phase II program.

This report describes the hardware and software generated in Phase I of the overall program. The system contains a mini-computer to control the various high-speed processing elements in the data path and a classifier which implements an all-digital prototype multivariate-Gaussian maximum likelihood decision algorithm operating at 2×10^5 pixels/sec. Sufficient hardware has been developed to perform signature extraction from computer-compatible tapes, compute classifier coefficients, control the classifier operation, and diagnose operation.

Volume I describes the MIDAS System in detail; Volume II contains the diagnostic programs used to test MIDAS' operations; Volume III displays the MIDAS construction and wiring diagrams.

17. Key Words	18. Distribution Statement
Wirewrap Real-time processing A/D-D/A conversion MIDAS Multispectral recognition system Multivariate-Gaussian maximum likelihood decision algorithm	Initial distribution listed at the end of this report. Unclassified-Unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
UNCLASSIFIED	UNCLASSIFIED	49	

Page Intentionally Left Blank

PREFACE

A comprehensive multispectral program devoted to the advancement of state-of-the-art techniques for remote sensing of the environment 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 multidisciplinary 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 system; (4) improving fast automatic data processing systems to extract information in a useful form; and also (5) assisting in data collection, processing, analysis and ground truth verification. The MIDAS program applies directly to No. (4) with its improved data processing capability.

This document is the final report for Phase I of the MIDAS program under NASA Contract NAS1-11979 and covers the period from October 1972 through February 1974. The contract effort was monitored by Mr. William Howle of NASA-Langley. 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 Multispectral Analysis Section, and by F. J. Kriegler, Principal Investigator. The ERIM number for this report is 195800-25-F.

ERIM personnel who contributed to this project and who co-authored this report are Dempster Christenson, Michael Gordon, Roland Kistler, Seymour Lampert, Robert Marshall, and Rowland McLaughlin. In addition to providing the text, their individual contributions were as follows: Dempster Christenson and Michael Gordon provided system programming and diagnostic software; Roland Kistler and Seymour Lampert provided the detailed design and performed system checkout; Robert Marshall aided in overall system configuration; Rowland McLaughlin organized this report. 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.

CONTENTS

FIGURES

MIDAS, PROTOTYPE MULTIVARIATE INTERACTIVE DIGITAL ANALYSIS SYSTEM—PHASE I

Volume II: Diagnostic System

1 INTRODUCTION

In developing a complex system of digital hardware such as MIDAS, it is necessary to provide a method or set of methods which provide a quantitative report on the performance of the hardware in order to achieve and maintain error-free operation with minimum operational cost and downtime. This set of methods comes under the general category of system diagnostics. With the large number of integrated circuits and wiring schemes developed for the MIDAS classifier, manual check-out of the various assemblies and subassemblies is virtually impossible without the aid of the PDP-11/45 computer system, which is interfaced to the classifier. A fast, efficient, and low cost preventive maintenance diagnostic system can be implemented using the computer system and a specially designed selective interrogation hardware feature.

In Phase I of the MIDAS project, a substantial number of diagnostic programs were developed. The set of these diagnostics can be broken down into the following four types: (1) off-line individual card diagnostics, (2) classifier subassembly diagnostics, (3) an automatic in-place fault isolation diagnostic using a diagnostic bus designed into the MIDAS classifier hardware, and (4) a classification accuracy and repeatability analysis package. This analysis program will serve as an operational evaluation tool for users having test-set ground observations, but has been initially employed as a diagnostic. The development of this set of programs was coordinated with the development of the MIDAS hardware; that is, as the check-out of the classifier hardware progressed, more complex "tools" to exercise and diagnose the more complicated subassemblies were devised and used. The frequency of use of various diagnostics in a preventive maintenance schedule will be worked out in Phase II.

After we completed initial fabrication of the classifier arithmetic cards, a set of programs was developed to check out, off-line, the operation of the hybrid, variance, matrix-multiplier, square, and square accumulator cards. Special test fixture connectors were fabricated to interface each type of card to the PDP-11/45 computer via a DR-11B DMA interface. The low-order twelve bits of the DR-11B data buffer (directly accessible to the PDP-11 CPU) were connected to the lines which would normally contain input data from the previous card in the classifier pipeline. Timing and strobe pulses, normally generated by the classifier control logic, were simulated by the PDP-11 CPU using the three function lines in the DR-11B status register (also directly accessible to the PDP-11 CPU) and the high-order four bits of the DR-11B data buffer. A separate unit was used to power the integrated circuits on the card. The bi-directional data lines, in conjunction with the several control lines, were used to load the various inputs (e.g., random-access memories and input data lines) and also to obtain the results of card operation. The expected result, simulated by the PDP-11, was compared to the actual value returned from

the card. Since the total number of inputs per card for a given operation was relatively small (usually about 16 bits of input), the computer was able to perform virtually all permutations of input in a relatively short time.

If incongruities between simulated and actual results were detected, a fault isolation procedure was initiated; this consisted mainly of loading the card with a known bit configuration and then instructing a technician how to do an error-detection procedure with a manual probe. In this way, faulty integrated circuit chips (ICs) and possible design errors (during initial checkout) could be quickly detected.

In addition, the off-line diagnostic for the so-called "hybrid" card performed an additional function. After checking out the various digital components of the card, it initiated an interactive calibration procedure to calibrate the outputs of the analog-to-digital and digital-to-analog circuits.

After the error-free operation of the individual classifier arithmetic components had been verified, a series of programs to exercise the MIDAS System's control logic subassembly were developed. This subassembly consists of 20 individual and different cards; so rather than write a complex diagnostic and fabricate a connector for each of these cards, it was decided to develop a series of operator-controlled exercisers to aid the hardware debugging.

The basic functions of the control subassembly are to set various internal switches, load control RAMs, and control the flow of data into the classifier from various external sources. These functions are initiated through control commands passed from the PDP-11 to the control subassembly via the DR-11C general interface. Two routines, CSTEP and CLOAD, were written to allow the operator to enter control commands into the subassembly in various modes of operation (e.g., continuous or single entry, etc.). Then, in conjunction with the sub-assembly schematics, the operator could manually probe the various logic cards and verify their functions.

Once the control logic subassembly had been diagnosed, the classifier subassembly was checked out. This involved the development of a series of programs which performed one of two functions: (1) loaded the RAMs associated with signature coefficients, or (2) transferred "data" from the PDP-11 into the classifier pipeline in a manner similar to that which would be used in the "normal" classification operation, and also extracted results from the classifier pipeline via the diagnostic bus structure built into the classifier.* It is sufficient to note here that the diagnostic bus allows the interrogation of inputs and outputs under PDP-11 computer control at selected points along the classifier pipeline. These operator-interactive exercises allowed the display of information from the diagnostic bus structure on the display register

*For a detailed description of the diagnostic bus structure, see Volume III.

located on the front of the PDP-11/45 mainframe unit. Through use of these routines, the classification operation could be initiated and maintained under operator control to allow more intricate manual analysis of the subassembly.

The programs RAMTST and RAMAD were written to allow the loading of user-specified constants into the classifier coefficient RAMs. Also used for this purpose was the operational program RAMLD which loaded real signature coefficients into the appropriate RAMs. The programs CHECK1 and CHECK3 were used to load a particular data vector and/or sets of data vectors into the classifier pipeline and to select and interrogate the appropriate diagnostic port as instructed by the operator. These two routines simulated the actual classification operation and allowed the operator to manually probe the pipeline in greater detail by establishing a repetitive operational mode in the classifier pipeline.

After the MIDAS system was completely checked out (i.e., both the control logic subassembly and the arithmetic classifier subassembly had been verified as error-free), a general in-place fault isolation diagnostic was devised. With the large number of integrated circuit components in use in the MIDAS system, the component failure rate, while relatively small after an initial component burn-in period, could not be ignored. The failure of a single integrated circuit chip could easily result in the failure of the entire classification system; therefore, a general system test was necessary for maintenance of a perfect performance level in all subassemblies.

With a hardware system as complex as MIDAS, the problem of software simulation of the hardware configuration via a general-purpose computer is by no means trivial. Through use of the diagnostic ports both at the input to the mean card and at the outputs from the mean, variance, matrix-multiplier, square, square accumulator, and recognition cards, in conjunction with the selective loading of classifier RAMs, it is possible to exercise each successive computational stage in the arithmetic processor pipeline; in case an error condition is present, a binary-search, fault-isolation technique can then be employed to locate the particular stage (i.e., card or set of cards) responsible for producing the anomaly.

Exercising each successive stage through the processor would be simple if direct computer access to all inputs to a particular stage were available. However, the hardware in its present configuration allows only indirect access through selective loading of fabricated coefficient constants and mean data inputs. For example, in order to generate a particular bit pattern at the input to a particular matrix-multiplier card, it is necessary to mold a set of coefficient constants for the mean and variance card RAMs, along with a unique data vector, such that the combination of these three sets of variables by the mean and variance cards' arithmetic logic units produces the desired bit pattern at the matrix-multiplier input. This function becomes more complex at subsequent stages along the pipeline.

Once error isolation has been resolved to a particular card, it is then possible to use the individual card check-out diagnostics to determine any faulty components. The assumption is made that any errors not associated with the integrated circuit chips will not appear, once the individual card has been verified in the initial stages of development. However, it is relatively simple to diagnose faulty connector problems, loose wiring, etc., through manual probing of the card.

In order to verify the algorithm implemented in the MIDAS hardware as well as the algorithms used to calculate parameters for the general classification procedure, it is necessary to analyze the classified results and determine the accuracy of the technique. The standards used to determine the accuracy of the MIDAS results are of two types: (a) the results from a performance-tested maximum-likelihood classifier (e.g., generated on a general-purpose computer), and (b) ground-truth information supplied from an external source.

Analysis is via pixel-by-pixel comparison of the MIDAS-generated classification results with either other classification results or ground-truth information. A square matrix A of order N (where N is the number of possible classification results) is accumulated, such that Ai,j represents the number of pixels classified as i by the general-purpose computer (or by ground-truth information) but which were classified as j by MIDAS. Thus, the diagonal of the matrix represents the set of coincident classifications. Subsequent to verification of the algorithm used by the MIDAS classifier, this program could be used to evaluate the system's accuracy for a range of data-set types.

Because of limitations in time, personnel, etc., various diagnostic aids were conceived but not implemented in Phase I of MIDAS:

1. In Phase I of MIDAS, direct interrogation of the coefficient RAMs is not possible using the diagnostic bus structure currently implemented. It is possible to detect errors in the loading of these RAMs by the arithmetic logic of the particular card they feed; however, it is not possible to determine automatically the source of error, because the arithmetic logic may be at fault. The design for incorporation of direct RAM access has been initiated and will be implemented in the Phase-II architecture.
2. With the implementation of a preprocessing subassembly in Phase II, a diagnostic access structure and associated diagnostic software will be designed and implemented. The hardware implementation of the preprocessing functions, which as yet have not been fully detailed, will determine the architecture of both the bus structure and the software to be developed.

2
PROGRAM DESCRIPTION

2.1 CARD CHECK-OUT DIAGNOSTICS

2.1.1 ADCHK

TITLE: ADCHK

PURPOSE: To diagnose hardware errors in the so-called "hybrid" cards and to calibrate the D/A and A/D converters off-line from the MIDAS classifier hardware.

CALLING SEQUENCE: None (main program)

DESCRIPTION: Upon entry to the program, the message

'SET CSW: Ø=CHECKOUT, 1=D/A CAL, 2=A/D CAL'

is printed on the console. The console switches should then be set by the user to designate the mode of operation:

CSW bits 1 Ø

CSW value Ø Ø - Check digital path and D/A-A/D path thru hybrid card

Ø 1 - perform calibration procedure for D/A converter

1 Ø - perform calibration procedure for A/D converter

When the console switches are set properly, the operator presses the CR key on the console, and a branch is made to one of the following operating modes:

1. CHECK-OUT MODE (MODE Ø)

   In this mode it is assumed that the A/D and D/A converters have been calibrated using modes 1 and 2. The data input to the card is permuted over the range from -128 to 127. Both the digital path and the D/A-A/D path are verified against the input value. Switching is effected for each input value such that both the processor latch and the diagnostic latch are tested.

   If an error is detected, the following error message is printed:

DIGITAL PATH			D/A - A/D PATH	
Y	BUS	PORT	BUS	PORT
xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx

   where xxxxxx represents a 6-digit octal number.

2. D/A Calibrate (MODE 1)
   In this mode, a digital voltmeter is attached to the output of the D/A converter to read the output. The following message is printed on the console:

   'D/A CALIBRATION - TYPE OCTAL VALUE TO BE USED -'

   The user then enters a 6-digit octal value (noting that only the low-order 8 bits are significant) between -128 and 127. This range of values should correspond to the voltage range -5.00 V to 4.96 V. The A/D converter may be adjusted via a setscrew on the IC.

3. A/D Calibrate (MODE 2)
   In this mode, it is assumed that the D/A converter has been calibrated. The following message is printed on the console:
   'A/D CALIBRATION - TYPE OCTAL VALUE TO BE USED'
   The user then enters a 6-digit octal number (noting that only the low-order 8 bits are significant). If an error in converting from analog to digital is found, an error message is printed in the same form as the mode Ø error message. The A/D converter should be adjusted until the error message is no longer printed.

SUBROUTINES REQUIRED:

- GETCSW - a routine to return the contents of the console switch register to the calling program
- SETF2 - a routine to set the value of function line 2 in the DR-11B status register to either Ø or 1
- SETF3 - a routine to set the value of function line 3 in the DR-11B status register to either Ø or 1
- SETB14 - a routine to set the value of bit 14 in the DR-11B data buffer to either Ø or 1
- CDR12 - a routine to load a data value into the "hybrid" cards input latch
- OUTSEL - a routine to select either the diagnostic or processor three-state output latch and return the value to the calling program

STORAGE REQUIRED: 4542_8 bytes

2.1.2 VARNCE

TITLE: VARNCE

CSECT: None

PURPOSE: To test the performance of a variance card off-line from the MIDAS classifier hardware. (See Fig. 1.)

CALLING SEQUENCE: None. This is a main program.

DESCRIPTION: There are 16 RAMs that must be loaded first. Their addresses are Ø → 15. Because of the limited size of the DR-11B buffer register, only 11 bits of a number can be loaded. The least significant bit of this number will become the two least significant bits of the actual number loaded into the RAM. The bit layout of the DR-11B buffer register is as follows:

B15 - ~ (WRITE ENABLE)	B7 - MSB - SIGN
B14 - LSB + 3	B6 - LSB + 1Ø
B13 - LSB + 2	B5 - LSB + 9
B12 - LSB + 1 and LSB	B4 - LSB + 8
B11 - RAM ADDRESS	B3 - LSB + 7
B1Ø - RAM ADDRESS	B2 - LSB + 6
B9 - RAM ADDRESS	B1 - LSB + 5
B8 - RAM ADDRESS	BØ - LSB + 4

LSB - Least significant bit

MSB - Most significant bit

The numbers loaded into the DR-11B buffer register vary between -1Ø24 and 1Ø23. After the RAMs are loaded, the number in each RAM is multiplied by all the integers between -128 and 127. These numbers are also entered through the DR-11B buffer register. Two things must be specified: the address of the RAM containing the multiplicand and the multiplier. This bit layout of the DR-11B buffer register is as follows:

B15 → B12 - Not used	B4 - LSB + 4
B11 → B8 - RAM ADDRESS	B3 - LSB + 3
B7 - MSB - SIGN	B2 - LSB + 2
B6 - LSB + 6	B1 - LSB + 1
B5 - LSB + 5	BØ - LSB

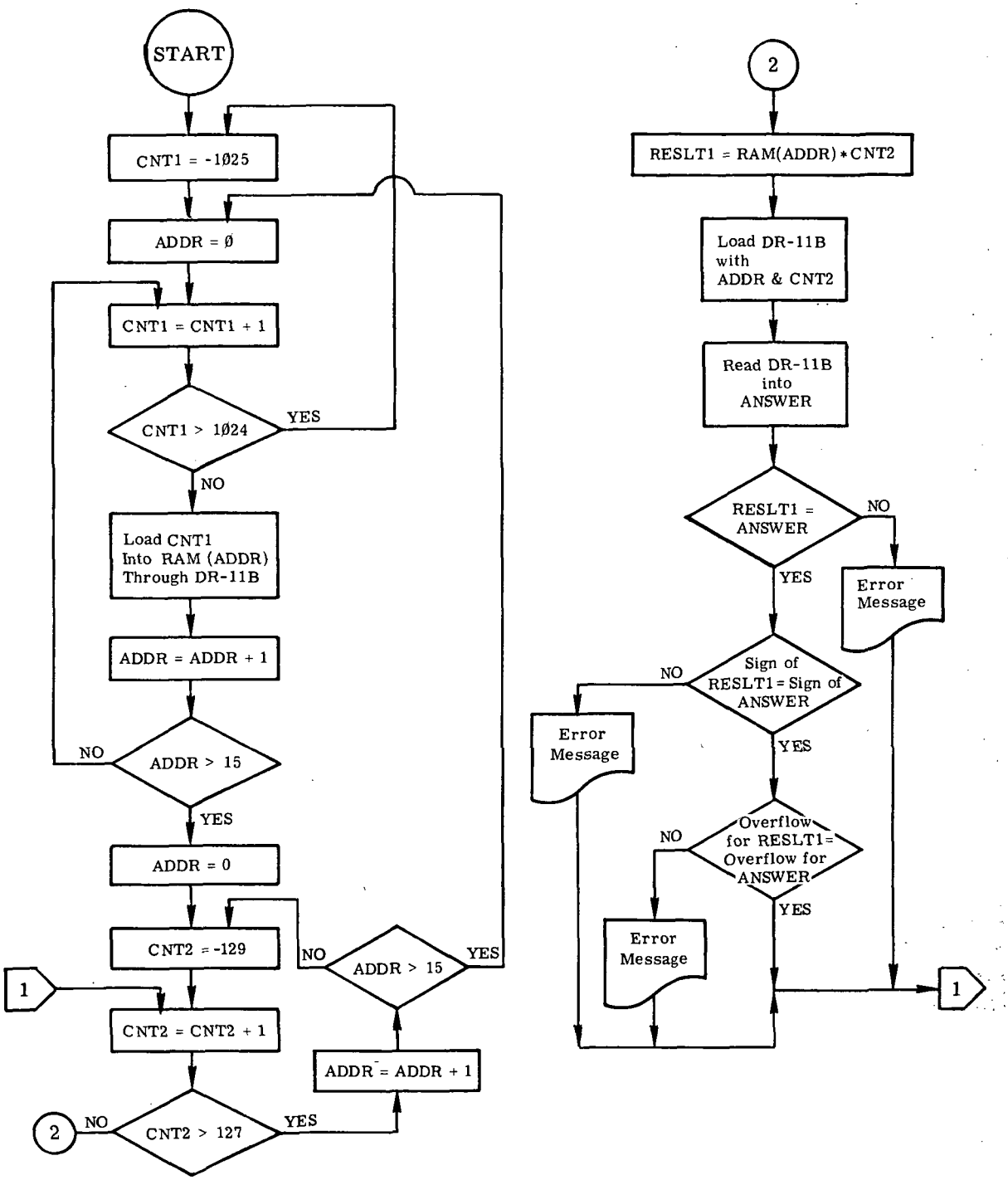


FIGURE 1. FLOWCHART FOR VARIANCE CARD DIAGNOSTIC

The results of the multiplications then return via the DR-11B buffer register. (Don't forget that when the RAMs were loaded, the 1's complement of the number put in the DR-11B buffer register was the actual number loaded.) The program then calculates what the answer should be and compares this with the result returned by the variance card. The bit layout of the number returned in the DR-11B buffer register is as follows:

B15 - N/U	B7 - LSB + 1Ø
B14 - N/U	B6 - LSB + 9
B13 - N/U	B5 - LSB + 8
B12 - N/U	B4 - LSB + 7
B11 - MSB - SIGN [LSB + 19]	B3 - N/U
B1Ø - LSB + 13	B2 - N/U
B9 - LSB + 12	B1 - N/U
B8 - LSB + 11	BØ - ~(OVERFLOW)

N/U = not used

~(OVERFLOW) = 1 when LSB + 14 → LSB + 19 are 1's or
when LSB + 14 → LSB + 19 are Ø's
= Ø for any other combination of 1's and
Ø's in LSB + 14 → LSB + 19

If B4 → B1Ø from the card are not equal to LSB + 7 → LSB + 13 of the computed answer, the following message is printed.

RAM = AAAAAA CON = BBBBBB EX = CCCCCC CD = DDDDDD

AAAAAA - octal multiplicand in RAM
BBBBBB - octal multiplier loaded into DR-11B buffer registers
CCCCCC - expected octal result LSB + 7 → LSB + 13
DDDDDD - octal result calculated by variance card B4 → B1Ø

If the results agree, then the expected sign of the result is compared with the sign determined by the variance card. If they do not agree, the following is printed.

RAM = AAAAAA CON = BBBBBB EX = CCCCCC
EX SIGN = EEEEEE CD SIGN = FFFFFF

EEEEEE - either Ø or 1; this is what the sign should be.
FFFFFF - either Ø or 1; this is the sign returned by the variance card.

If the signs agree, the overflow bit determined by the computer is compared with the overflow bit returned by the variance card. If they don't agree, the following message is printed:

RAM = AAAAAA CON = BBBBBB EX = CCCCCC
SIGN = EEEEEE EX OVR = GGGGGG CD OVR = HHHHHH

GGGGGG - either Ø or 1; this is what ~(OVERFLOW) should be.
HHHHHH - either Ø or 1; this is what the variance card returned for ~(OVERFLOW).

If the overflow bits agree, no message is printed out. The next multiplier is then loaded and multiplied by the same multiplicand stored in the same RAM address. After the multiplier exceeds $127_{1Ø}$, the next RAM is accessed and used as the multiplicand. The multiplier is set to -128 and the process repeats. After all 16 RAMs have been multiplied by all the integers between -128 and 127, the RAMs are reloaded with the next 16 numbers in the sequence (between -1Ø24 and 1Ø23). When the multiplicand exceeds 1Ø23, the RAMs are reloaded starting with -1Ø24, and a message is printed out indicating the completion of one full cycle. This message is printed once before any results are compared and also after each complete cycle. The printing of all error messages can be disabled by setting the switch register on the PDP-11/45 console to all zeroes.

Timing for One Full Cycle:

(1) The RAM multiplicands vary between -1Ø24 and 1Ø23. This gives 2Ø48 possible numbers. Sixteen RAMs are loaded at a time. The full set of RAMs is loaded 128 times. (2Ø48/16 = 128).

(2) There are 256 possible numbers for multipliers; all 256 are used on each RAM multiplicand. There are 4Ø96 multiplications performed by the card for one full set of RAMs (i.e., 256 × 16 = 4Ø96).

(3) Since the RAMs are loaded 128 times and 4Ø96 multiplications are done by the variance card for each set of RAMs, there are 524,288 multiplications done by the card for one complete cycle through all possible combinations.

(4) This normally does not take long because the card achieves the result within one PDP-11/45 instruction cycle. Unfortunately, both the juggling of the bits going to and from the DR-11B buffer register and the comparisons necessary take much longer than doing the multiplications themselves. Consequently, one full cycle of all possible combinations will take approximately 1Ø minutes (assuming error message printout is disabled).

NOTE: As this is being written, there is still a bug in the program that causes it to "bomb out" after two complete cycles. The bug has no effect on the calculations or comparisons performed. Therefore, this program can still be used to run a variance card through two complete cycles.

DEVICE REQUIRED: KEYBOARD OUTPUT

MACROS REQUIRED: .BIN2O, .INIT, .OPENO, .WRITE, .WAIT, .RLSE

STORAGE REQUIRED: $2Ø42_8$ bytes

2.1.3 MTXMTI AND MTXERR

TITLE: MTXMTI

CSECT: None

PURPOSE: To statically check the performance of a matrix-multiplier (MTX) card off-line from the MIDAS Classifier hardware.

CALLING SEQUENCE: None. This is a main program.

DESCRIPTION: This program loads the MTX card with a large number of permuted multiplicands, and insures that the card is operating properly.

A set of X values is loaded into the RAM on the card. Then eight Y values are multiplied by the X values and accumulated. The outputs of this card are checked against the computer-calculated 'correct value' and errors are flagged.

The setting of sense switches on the console affects the error printout in that errors are not printed out if any of the sense switches are set in the up position.

One complete pass of this diagnostic takes approximately 2Ø seconds.

STORAGE REQUIREMENTS: 2162_8 bytes

TITLE: MTXERR

CSECT: None

PURPOSE: To aid in locating errors in an MTX card found to be defective by the MTXMTI diagnostic program.

CALLING SEQUENCE: None. This is a main program.

DESCRIPTION: This diagnostic loads the MTX card with various values which help in isolating defective components on the card itself. Five states are possible, and a message denoting each state is printed on the console. To continue to the next state, a response of 'CO' followed by a carriage return is sufficient.

STATES:

1. Card IN ZERO STATE-TYPE CO<CR> TO CONTINUE
   AØØ5 ØØØØØØ
   In this state, the pins denoted in Table 1 should be at Ø (low state).
2. ONE STATE ** X=1, Y=-1
   AØØ5 ØØØØ1Ø
   In this state, the pins denoted in Table 2 should be at 1 (high state).
3. AØØ5 ØØØØ2Ø
   In this state, the pins denoted in Table 3 should be at 1 (high state).
4. ONE STATE ** X=-1, Y=1
   AØØ5 ØØØØ1Ø
   In this state, the pins denoted in Table 2 should be at 1 (high state).
5. AØØ5 ØØØØ2Ø
   In this state, the pins denoted in Table 3 should be at 1 (high state).

STORAGE REQUIREMENTS: $146Ø_8$ bytes

TABLE 1

IC NUMBER	PIN NUMBER
5	2,5,7,10,12,15
6	2,5
8	2-7,10-15
9	6,7,10-15
10,11,12,14,15,17	3-6,11-14
13,16	2-7,10-15
30,31,32	1,2,9-11,13,18-23

TABLE 2

IC NUMBER	PIN NUMBER
5	2,5,7,10,12,15
6	2,5
8	2-7,10-15
9	6,7,10-15
13,16	3,4,6,9,11,13,14
30,31,32	1,2,9-11,13,18-23

TABLE 3

IC NUMBER	PIN NUMBER
10,11,12,14,15,17	3-6,11-14
13,16	2,5,7,10,12,15
30,31,32	1,9-11,13,18,20, 22

2.1.4 SQARCK

TITLE: SQARCK

CSECT: [BLANK]

PURPOSE: To check the performance of square cards off-line from the MIDAS classifier hardware. All possible numbers are tested and compared with the expected result, including overflow conditions.

CALLING SEQUENCES: None. This is a main program.

DESCRIPTION: The routine starts a counter at -513, reformats the number for input to the card, loads the DR-11B buffer register, reads the result from the DR-11B buffer register, compares the actual answer with that calculated by the card, and prints out an error comment if the two numbers are not identical, or if the overflow bits are different. Overflow will occur for any number N which is outside the range

$-512 \leq N \leq 511$

The counter is then incremented by 1. If the counter is greater than 514, the counter is set to -513 and the cycle repeats.

DR-11B BIT	INPUT FORMAT CONTENTS	OUTPUT FORMAT CONTENTS
Ø	LSB + 4	LSB + 1
1	LSB + 5	LSB + 2
2	LSB + 6	LSB + 3
3	LSB + 7	LSB + 4
4	LSB + 8	LSB + 5
5	LSB + 9	LSB + 6
6	LSB + 1Ø	LSB + 7
7	MSB (SIGN)	LSB + 8
8	LSB	LSB + 9
9	LSB + 1	LSB + 1Ø
1Ø	LSB + 2	LSB + 11
11	LSB + 3	MSB (SIGN)
12	NU	OVERFLOW
13	NU	NU
14	NU	NU
15	NU	LSB

LSB = Least significant bit
MSB = Most significant bit
NU = Not used

If the user does not wish error comments to be printed, the switch register should be set to zero. If any switch is set to a 1, error comments will be printed. The only comment printed is:

I*K = XXXXX, CARD SAYS YYYYY,
OVRFLW = ZZZZZ

where XXXXX - the true square of the counter value
YYYYY - the square of the counter value as calculated by the square card
ZZZZZ - This is the overflow indicator. It is either Ø for no overflow or 1 for overflow.

If XXXXX = YYYYY, then the actual overflow is different from the overflow bit generated by the square card.

DEVICE REQUIREMENTS: keyboard output, switch register input

MACROS REQUIRED: .BI1V2D, .INIT, .OPENO, .WRITE, .WAIT, .RLSE

STORAGE REQUIREMENTS: 537_8 bytes
(See Fig. 2)

2.1.5 SQACC

TITLE: SQACC

CSECT: None

PURPOSE: To test the operation of the MIDAS square-accumulator card off-line from the MIDAS classifier hardware. (See Fig. 3.)

CALLING SEQUENCE: None. This is a main program.

DESCRIPTION: This program is used to test the operation of the MIDAS square-accumulator card (Fig. 4). Testing is done off-line using a test connector to DR-11B DMA interface #1. Data input and output go through the DR-11B data buffer (DRDB) lines. Function line 1 of the DR-11B status register (DRST) is used to provide the accumulator latch-clear signal. Latch clear is enabled when this line is low (Ø). Function line 2 of the

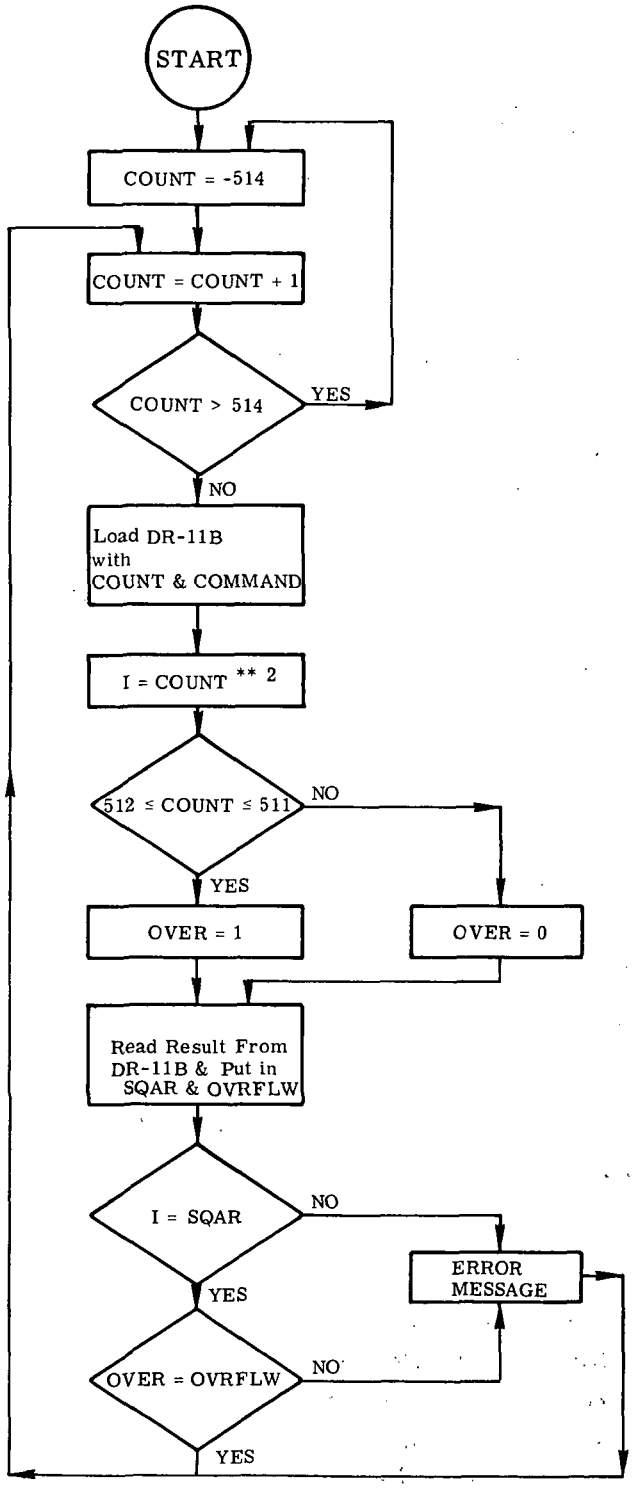


FIGURE 2. FLOWCHART FOR SQUARE CARD DIAGNOSTIC

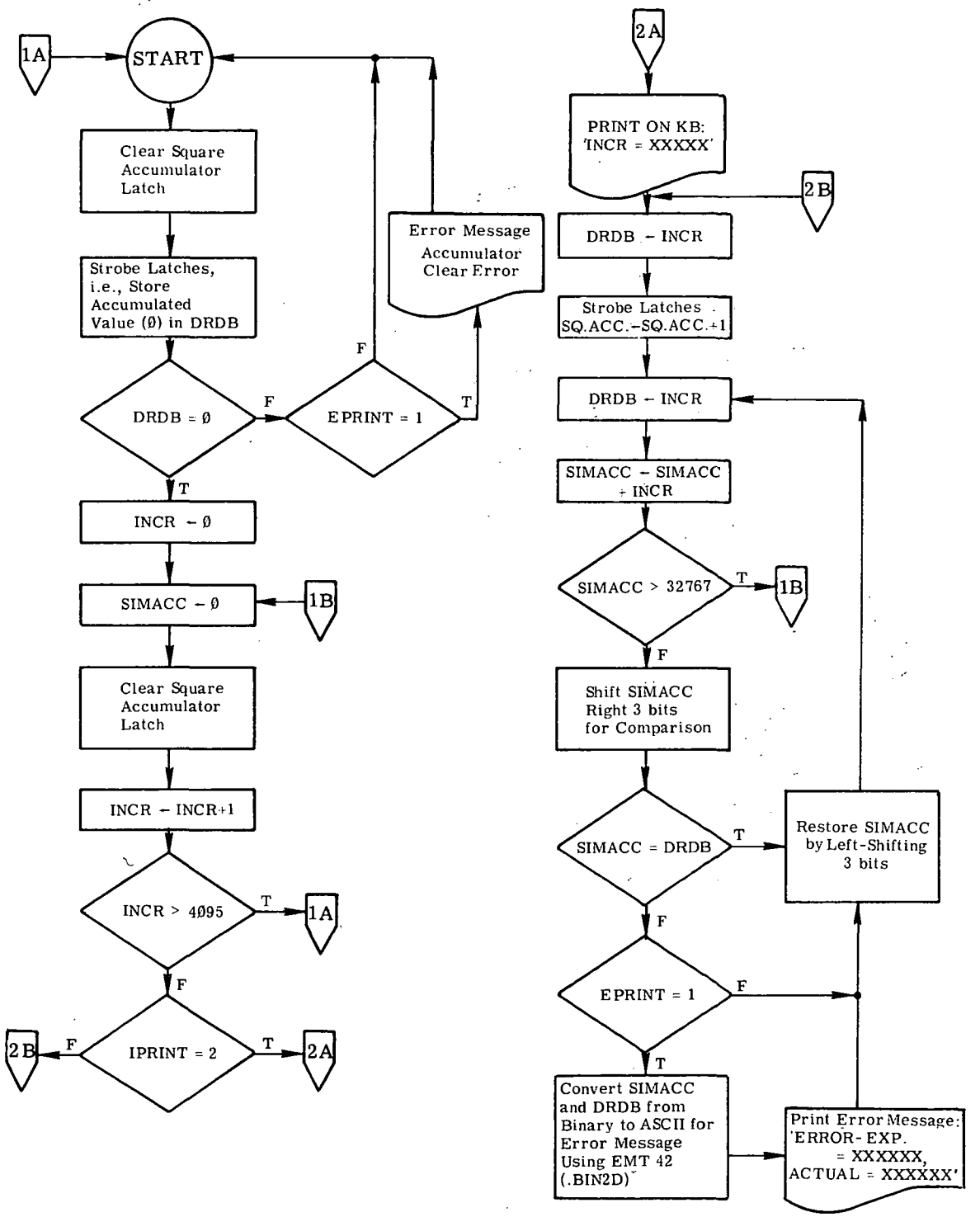


FIGURE 3. FLOWCHART FOR SQACC

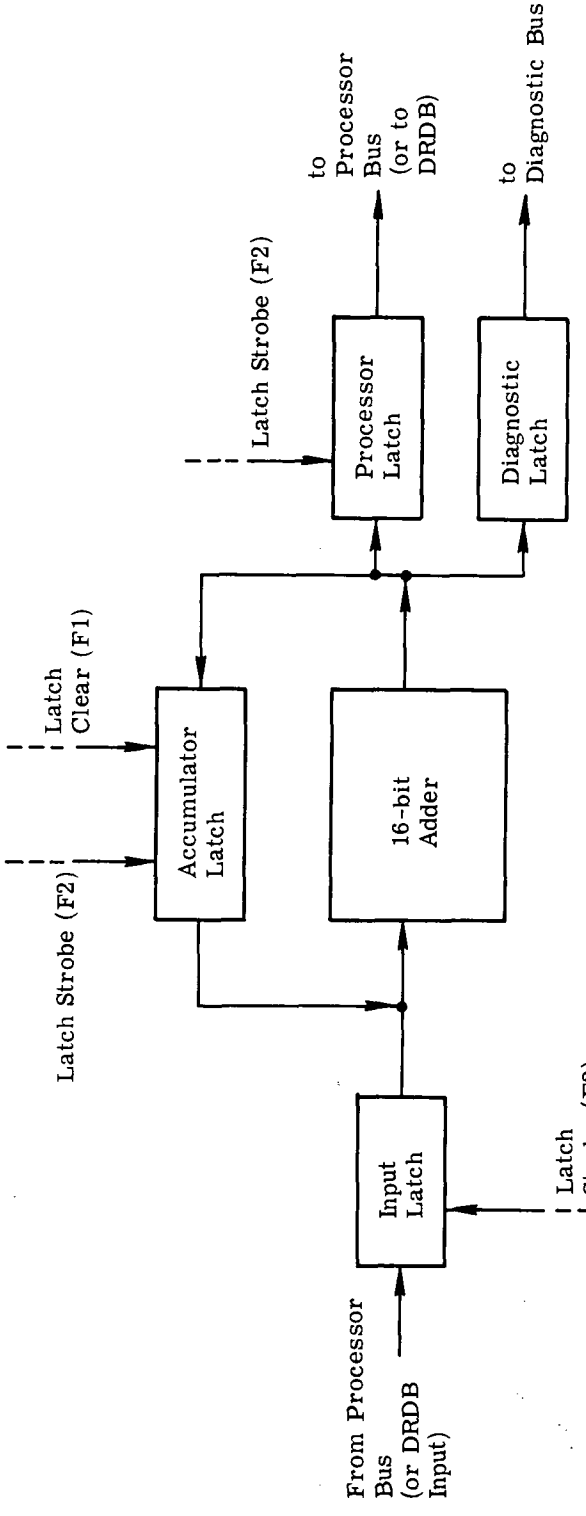


FIGURE 4. BLOCK DIAGRAM FOR SQUARE ACCUMULATOR

DRST is used to provide the latch level which stores the value on each latch's input lines in the latch. Latching is enabled when this line is high (1). This program first checks the accumulator-latch clear function. If this test fails, no branch is made to test the accumulation function of the card. If, however, this test is successful, an increment (with initial value of 1) is added to the latch, and the result is tested until overflow occurs. Then the increment is increased by one, the accumulator latch is cleared, and the process is repeated until the increment reaches a value of 4Ø95, at which time the entire diagnostic is repeated.

Sense switch Ø controls error printing (if SSW Ø = Ø, no printing; if SSW Ø = 1, print). If the expected latch value (calculated by the PDP-11/45) and the actual value returned via the DRDB do not coincide, error message #2 is printed on the keyboard. If the latch-clear test fails, error message #1 is printed on the keyboard, dependent on the value of SSW Ø. Sense switch 1 controls the printing of the increment on the keyboard. If SSW1 = Ø, no increment is printed; if SSW1 = 1, the following message is printed:

'INCR = XXXXXX'

where XXXXXX represents the 6-digit octal increment.

ERROR MESSAGES:

1. 'ACCUMULATOR CLEAR ERROR'

   occurs when accumulator clear test fails

2. 'ERR - EXP = XXXXXX, ACTUAL = YYYYYY'

   occurs when comparison of simulated and actual values fails

   where XXXXXX represents the 6-digit octal value simulated by the PDP-11/45

   and YYYYYY represents the 6-digit octal value returned from the square-accumulator card via the DRDB

MACROS USED: .BIN2O, .INIT, .RLSE, .WRITE, .WAIT, .PARAM

SUBROUTINES NEEDED: None

STORAGE REQUIREMENTS: 544_8 bytes

2.2 SUBASSEMBLY DIAGNOSTICS

2.2.1 RAMTST

TITLE: RAMTST

CSECT: None

PURPOSE: To load the contents of the MIDAS-Classifier RAMs with known values for diagnostic or set-up purposes.

DESCRIPTION: This program allows the user to load the contents of the classifier coefficient RAMs with either a single, user-specified, 12-bit constant or the address defining the RAM to be loaded. (See hardware section concerning RAM addressing.) The type of constant loading to be performed is determined by the setting of console sense switches in the following manner:

1. If SSW 15 is on (1), then the constant loaded into the MIDAS RAMs will be the address of each particular RAM. The following message will be printed on the console before any coefficient loading takes place:

   TYPE <CR> TO LOAD RAM ADDRESS IN RAMS

   The coefficient load will take place after carriage return is entered at the console. If any of SSWs Ø-14 are on (1), the program will continue loading RAM addresses into the RAMs until SSWs Ø-14 are all off (Ø). Then SSW 15 will be tested again.

2. If SSW 15 is off (Ø), the following message will be printed on the console:

   ENTER CONSTANT (4 OCTAL DIGITS)=

   The user should then enter a 4-digit octal constant which will be loaded into all of the MIDAS RAMs. Any response other than a legal octal number will result in the request being reprinted on the console until a proper response is entered. Again, if SSWs Ø-14 are not all off (Ø), the program will continue loading the constant into all the RAMs until all of SSWs Ø-14 are off (Ø). SSW 15 is then tested again and step 1 or 2 above repeated as indicated.

STORAGE REQUIREMENTS: 6464_8 bytes

2.2.2 CSTEP

TITLE: CSTEP

CSECT: None

PURPOSE: To allow the diagnosis of hardware problems in the MIDAS control logic by passing user-specified commands to the hardware via the DR-11C interface.

DESCRIPTION: This program passes the various MIDAS control commands to the hardware logic for diagnostic and set-up purposes (see Volume I, Section 6.1.)

Settings of the console switches (SSW Ø . . . SSW15) control the logical flow of the program in the following manner:

1. If SSW 15 is off (Ø), the program will cycle through the 16 possible MIDAS commands, passing each to the control hardware. If any of the switches Ø-14 is in the on position, the program will print out the command sent and stop. A carriage return on the console keyboard will restart the program.

2. If SSW 15 is on (1), the following message will be printed on the console keyboard:

   ENTER OCTAL COMMAND =

   The proper response to this request is a 6-digit octal number representing 16 binary bits (the high-order octal digit may only have a value of Ø or 1). Any other response will cause the request to be reprinted until a proper response is made.

   If any of the switches Ø-14 is in the on (1) position, the program after passing the last command to the hardware, will ask for a new command. If all of SSWs Ø-14 are off (Ø), the program will repeatedly pass the last command entered until any of the SSWs Ø-14 are placed in the on (1) position.

STORAGE REQUIREMENTS: 674_8 bytes.

2.2.3 CHECK1

TITLE: CHECK1

CSECT: None

PURPOSE: To provide the user with a method of repetitively cycling the MIDAS processor in the digital-classify mode.

DESCRIPTION: This program assumes the coefficient RAMs of the MIDAS Classifier have already been loaded with proper constants for classification (via RAMTST diagnostic or RAMLD program). The user is then allowed to enter at any time either an output code word specifying the diagnostic port and time to be examined, or a data vector of 4 channels of data which will be passed to the classifier for classification.

The flow of the program is governed by the settings of the console sense switches in the following manner:

1. If SSW 15 is on (1), a new 6-octal digit output code word must be entered from the console. The following request will be made:

   ENTER OUTPUT CODE WORD =

   The proper response to this request is a 6-digit octal number specifying the diagnostic port selected and the time of sampling. Any improper response will cause the request to be reprinted until a proper response is made.

2. If SSW 14 is on (1), the user must enter a 4-channel data vector from the console. The following request is made via the console:

   ENTER DATA =

   The proper response to this request is four 3-digit octal numbers separated by commas, with termination by a carriage return. Any improper response will cause the request to be reprinted until a proper response is made.

After the output code word and the data vector have been entered, a "record" of data is concocted, composed of the same data vector repeated until a record 5000_{10} pixels in length has been generated. This vector is then passed to the MIDAS Classifier, and the subsequent

diagnostic outputs are returned to core and displayed in the PDP-11/45 console display register.

If both SSW 14 and SSW 15 are off (Ø), the same "record" of data is passed to the MIDAS Classifier until either a new output code word or a new data vector is to be entered.

This program is designed in such a way as to be an exact analog of the procedure used in the program CLAS2D, such that any problems in CLAS2D will manifest themselves in this diagnostic.

STORAGE REQUIREMENTS: $7ØØØØ_8$ bytes

2.2.4 CHECK2

TITLE: CHECK2

CSECT: None

PURPOSE: To sequence through classifier operation with permuted data vectors so that any data-dependent error conditions will be flagged.

DESCRIPTION: In the course of debugging the MIDAS Classifier hardware, we discovered some data-dependent errors in the system. To locate these error conditions, the program CHECK2 was written to permute the input data vector through user-supplied boundaries in each channel and thus seek out anomalous classification results.

The diagnostic program CHECK2 assumes that the MIDAS coefficient RAMs have been loaded in such a way that the 16 signatures loaded are exactly equal. Hence, the classification result should be the same for all pixels. Any non-standard result is flagged as an error. The following two requests are made at the console keyboard to the user:

1. ENTER MINIMUM VALUES =
   This request asks for the lower bound on the data vector in each of the four channels. Proper response to this request is four 3-digit octal numbers in 2's complement notation, separated by commas. Each

octal number respresents 8 bits (thus the high-order digit is actually base 4 rather than base 8). The high-order bit of the 8-bit number is assumed to be a sign bit.

2. ENTER MAXIMUM VALUES =
   This request asks for the upper bound on the data vector in each of the four channels. The proper response, as above, is four 3-digit octal numbers representing an 8-bit 2's complement number.
   The permuted data vectors are passed to the MIDAS Classifier using the same procedure as with programs CHECK1 and CLAS2D to ensure consistency of performance.

STORAGE REQUIREMENTS: 15134_8 bytes

2.2.5 CHECK3

TITLE: CHECK3

CSECT: None

PURPOSE: To exercise the MIDAS Classifier in a number of areas by varying (a) the clock frequency, (b) the data vectors transferred to the classifier, (c) the number of vectors transferred, or (d) the order of the data vectors. (See Fig. 5.)

CALLING SEQUENCE: None. This is a main program.

DESCRIPTION: This program allows the user to set the following variables that parameterize a classifier test run:

1. The clock command word which describes the classifier clock frequency, clock operation mode (i.e., burst/stop, burst/reset, free run), and the burst count.
2. The number of vectors to be used (i.e., entered from the keyboard and then duplicated to form a "line" of data).
3. The actual data vectors to be passed to the classifier (four values or channels per vector).
4. The total number of data vectors to be transferred per "line," where a "line" can be defined as the set of vectors

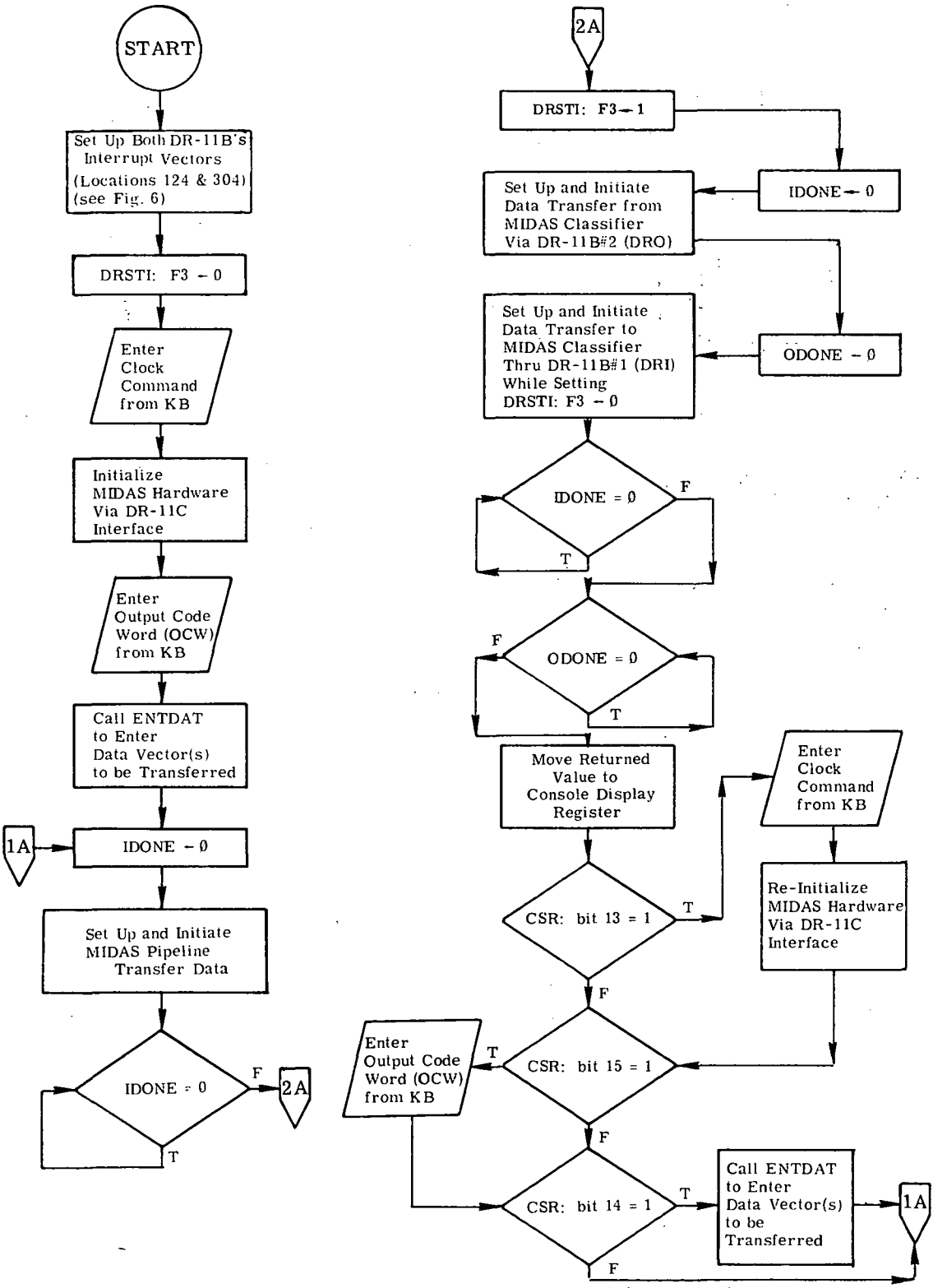


FIGURE 5. FLOWCHART FOR CHECK3 (Continued)

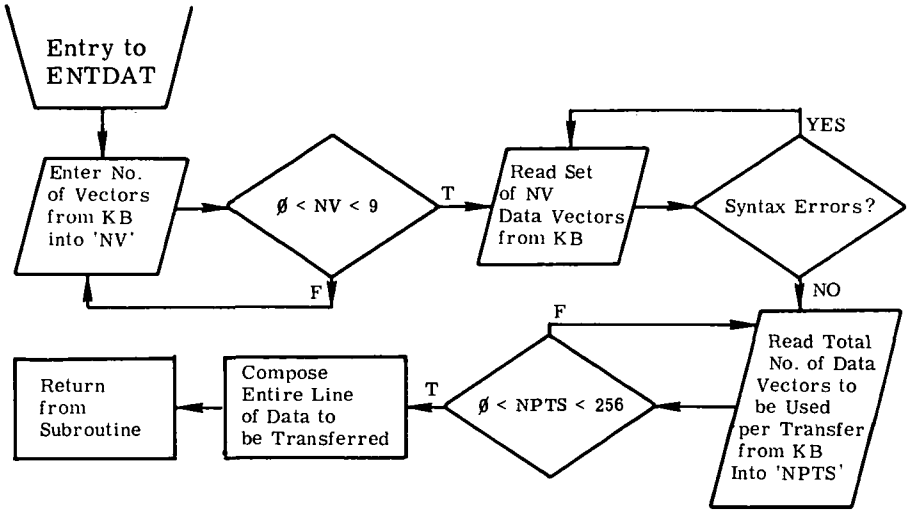


FIGURE 5. FLOWCHART FOR CHECK3 (Concluded)

DR-11B Interrupt Routines

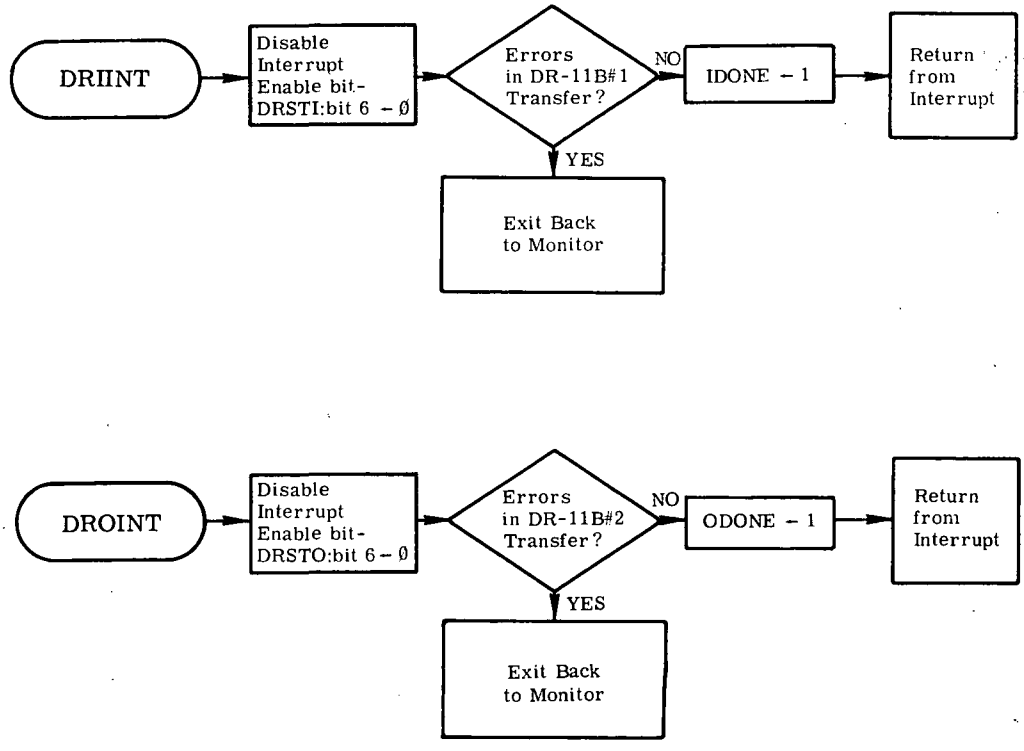


FIGURE 6. DR-11B INTERRUPT ROUTINES

transferred to the classifier in a single DMA operation of the input DR-11B.

Upon initiation of program execution, the user is interrogated for his specification of these four parameter groups. Once initially set, they can be changed at any time via console sense switches 13-15.

SSW 13 - If this switch is on (1), the clock parameter is requested at the keyboard by the following message:

'CLOCK='

The user should enter a 6-digit octal number, followed by a CR (carriage return). This number represents the clock command word as described in Vol. I, Section 6.1.1.

SSW 14 - If this switch is on (1), the data vector parameters may be changed. The user must respond to the following sequence of messages:

'NO. OF VECTORS='

The number of different data vectors the user wishes to use should be entered. This number must be within the range 1 through 8, otherwise the question will be repeated.

'VECTOR='

This message will be printed N times (where N is the response made to the previous question) on the console, once per vector input. The user should enter four 3-digit octal numbers, separated by intervening commas (e.g., '24Ø, 226, 2Ø1, 256'). Each 3-digit octal number represents an 8-bit two's complement data value to be associated with channels 1, 2, 3, and 4 respectively. Only the low-order 8 bits are significant, with the most significant bit of the eight representing the sign.

If a syntax error is encountered, the following message will be printed on the keyboard:

'ERR ** VECTOR='

The user should then re-enter the correct data vector.

'NO. OF POINTS/LINE='

The user should enter the total number of data vectors composing a "line." For example, if data values were entered for 3 vectors, and the response to this question was '9,' the "line" composed and subsequently transferred to the classifier would be of the form:

vector 1	2	3	1	2	3	1	2	3

If the number of points per line is not an even multiple of the number of data vectors entered, the number of points used is the nearest even multiple less than the number entered.

The maximum value for the number of points per line is 256. If the number entered is greater than 256, the question will be repeated.

MACROS NEEDED: .PARAM, .INIT, .RLSE, .READ, .WRITE, .WAIT, .O2BIN, .EXIT, .D2BIN

SUBROUTINES NEEDED: COMAND, GETARG

STORAGE USED: 5626_8 bytes

2.3 OPERATIONAL MAINTENANCE DIAGNOSTIC - TEST 1

TITLE: TEST1

CSECT: None

PURPOSE: To exercise the various points along the MIDAS Classifier in all bays, using selective loading of coefficient RAMs and the classifier data input lines, in order to detect errors and isolate these errors to a particular classifier component.

CALLING SEQUENCE: None. This is a main program.

DESCRIPTION: This program moves through the MIDAS-Classifier pipeline and verifies the operational status of each of the arithmetic functions. The following functions, or cards, are exercised

and examined: (1) the mean function card, (2) variance multiplication function, (3) matrix-multiplier function, (4) square function, (5) square-accumulator function, and (6) recognition function.

A set of subroutines we developed calculate the coefficient RAM constants and the input data vector necessary to establish a particular bit pattern at a particular point in the pipeline. These routines are:

FMEAN - Calculates input vectors and mean constants for specific mean output.

FVAR - Calculates variance coefficients and a variance input vector for a particular variance output.

FMTX - Calculates matrix multiplier coefficients and a matrix card input vector for a particular matrix multiplier output.

FSQ - Calculates square card input vector for particular square card output.

FSQAC - Calculates set of square accumulate input vectors necessary for particular square-accumulate card outputs.

By calling these routines in a given order it is possible to generate a set of coefficients for each card (where necessary) and the appropriate input vector to the classifier, such that all bits at a given diagnostic port can be tested. For example, to generate a particular bit pattern at the output of the variance card, the routine FVAR is called with the output bit configuration as an argument. The coefficients and mean outputs necessary are returned to the calling program. The coefficients are loaded into the variance RAMs and the routine FMEAN is called using the mean output bit configuration (generated by FVAR) as the argument. The mean coefficients and mean inputs necessary to generate the mean output desired are returned and loaded into the classifier.

If an erroneous output is detected, the contents of the RAMs upstream in the pipeline, along with the erroneous outputs and the expected outputs, are output to either the keyboard or the line printer. (Default is to the keyboard; however, the

line printer can be selected using the DOS Monitor ASSIGN command.)

SUBROUTINES NEEDED: DCL,RLOAD,FMEAN,FVAR,FMTX,FSQ,FSQAC, DRINT

MACROS NEEDED: .INIT, .RLSE, .PARAM, .READ, .WRITE, .WAIT, .BIN2O, .O2BIN, .EXIT

2.4 PERFORMANCE ANALYSIS

2.4.1 ANAL1

TITLE: ANAL1

CSECT: REFORM

PURPOSE: This routine reads any one file of an ERIM format, 7-track, 8ØØ-bpi tape and extracts all of the pixels for any channel. All the pixel values are then replaced by a value between Ø and 127 obtained from a translation table typed in by the user. The converted channel is then written back out on 9-track, 8ØØ-bpi tape with a file header record and a record header. Each scan line appears as one record. (See Fig. 7.)

CALLING SEQUENCES: None. This is a main program.

PARAMETERS: All parameters are read from the keyboard after the program prints the request.

REQUESTS:

1. HOW MANY RECOGNITION LEVELS ARE THERE?

This requests the number of values that are to be replaced on the ERIM tape. The response should be a 5-digit decimal number between 1 and 2Ø (exactly 5 digits must be supplied).

2. SPECIFY ERIM LEVEL AND ERTS LEVEL AS TWO, FIVE-DIGIT DECIMAL NUMBERS SEPARATED BY A COMMA. TYPE TWO NUMBERS PER LINE FOR AS MANY LINES AS THERE ARE LEVELS.

This requests the user to type-in the translation table as pairs of numbers, one pair of numbers per line. The number of pairs is equal to the number of recognition levels typed in as a response to Request #1. The first number of each pair is the ERIM level # to be replaced (a 5-digit decimal number between Ø and 511). This is followed by a one-character

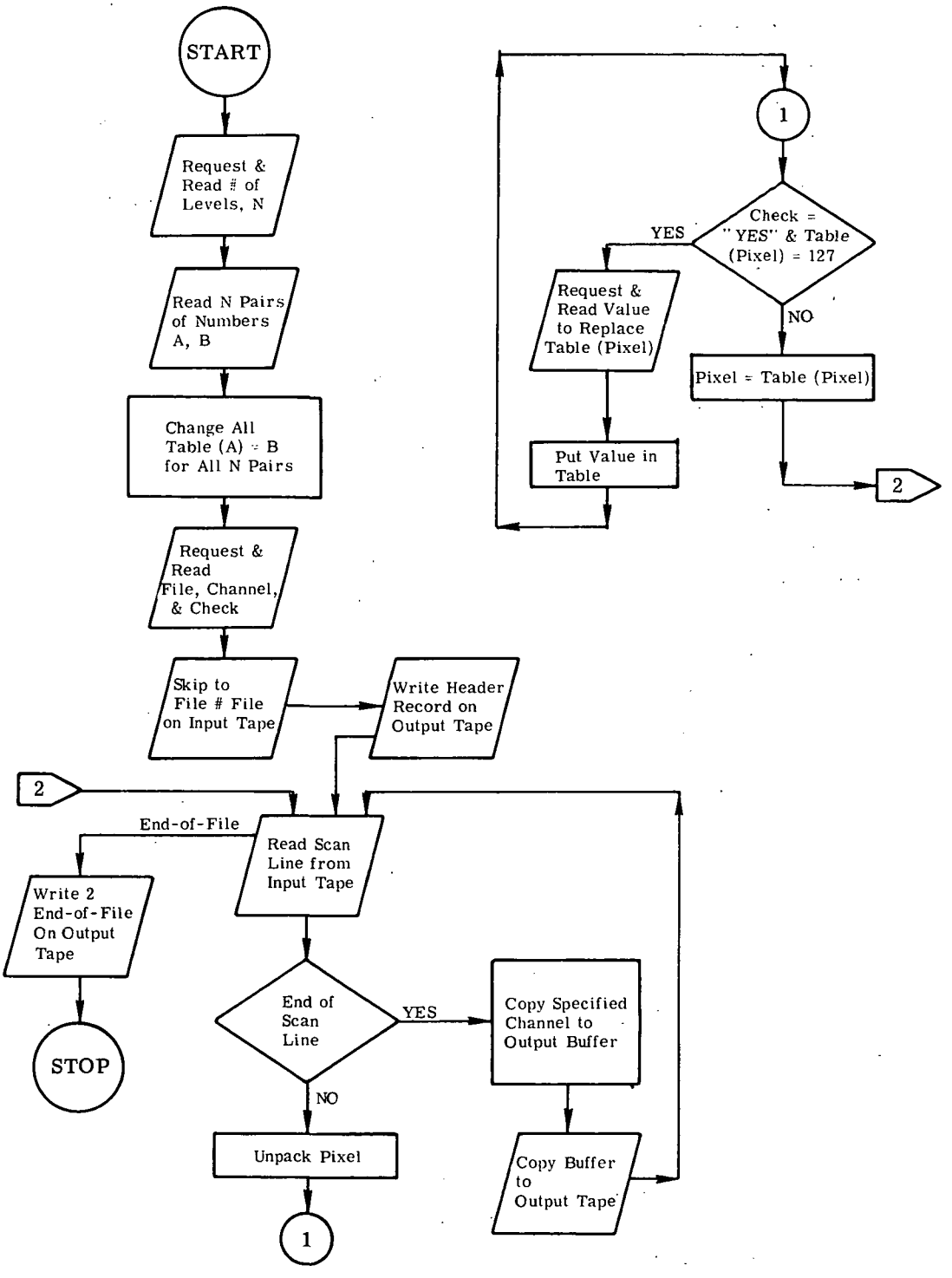


FIGURE 7. FLOWCHART FOR ANAL1

separator which is not checked. The separator is followed by the ERTS level number that is to replace every occurrence of the preceding ERIM level number (a 5-digit decimal number between Ø and 127).

After the table is read, the program prints, 'ALL RECOGNITION LEVELS READ.'

3. WHICH ERIM FILE IS WANTED?, WHICH CHANNEL IS WANTED?, *CHECK ALL CHANNELS?*

This requests the user to type the ERIM file number (a 5-digit decimal number) followed by a separator character. This is followed by the channel number that is to be converted and written on the 9-track tape (a 5-digit decimal number between 1 and the number of channels on the tape). Another separator character is typed, then the word YES or NO. This indicates whether all pixels of all channels are to be checked for ERIM levels for which no ERTS replacement was typed in. If an unspecified ERIM level is found, and YES was typed, the program requests an ERTS level to replace it with. If NO was typed, all unspecified ERIM levels are replaced by 177 octal.

DESCRIPTION: Presently, this program is designed to convert only 1 channel from 1 ERIM file. Each scan line is read in, then each pixel is unpacked and converted according to the table read in. If an unspecified ERIM level is found in any channel and checking was requested, the ERIM level is printed out and a replacement is requested. If checking was not requested, the unspecified ERIM level is replaced automatically by decimal 127. After the scan line is converted, the pixels in the requested channel are written out on 9-track tape. When all the scan lines have been read, converted, and written back out, an EOF is written on the 9-track tape, and the program terminates.

STORAGE REQUIREMENTS: 35174 octal locations

DEVICE REQUIREMENTS: keyboard-input
keyboard-output
MT1-7 track, 8ØØ bpi ERIM format input tape
MTØ-9 track, 8ØØ bpi 1-channel output tape

MACROS NEEDED: .INIT, .READ, .WRITE, .TRAN, .WAIT, .RLSE, .D2BIN, .BIN2D, .SPEC, .EXIT

SUBROUTINES NEEDED:

BCD2AS - This converts BCD to ASCII.

F3632 - This converts a 36-bit, floating-point word in 7094 format from magnetic tape into a 32-bit, floating-point number (in PDP-11/45 format).

I3616 - This converts a 36-bit, 7094-format integer from magnetic tape into a 16-bit, PDP-11/45-format integer number.

ERRORS:

ERR1,1. NONDECIMAL DIGIT FOUND. TRY AGAIN.

ERR1,2. NUMBER > 65535. TRY AGAIN.
The largest decimal number that will fit in the PDP-11/45 word is 65535.

ERR2. THERE AREN'T THAT MANY RECOGNITION LEVELS POSSIBLE. TRY AGAIN.
More than 20 recognition levels requested.

ERR3. INVALID ERIM LEVEL. TRY AGAIN.
ERIM level ≥512.

ERR4. INVALID ERTS LEVEL. TRY AGAIN.
ERTS level ≥128.

ERR5. UNSPECIFIED ERIM LEVEL FOUND.
ERIM LEVEL =
WHAT ERTS LEVEL SHOULD BE USED?
An unspecified ERIM level was found when checking was requested. Type replacement ERTS level.

ERR6. I/O ERROR.

ERR7. CHANNEL = Ø. TRY AGAIN.
Channel requested for conversion nonexistent.

ERR8. CHANNEL DESIRED > NCHAN
Channel requested for conversion nonexistent.

2.4.2 ANAL2

TITLE: ANAL2

CSECT: RESDNT

PURPOSE: This package of routines compares the output tape generated by ANAL1 with a disk recognition file generated by the classification program. The results are put into a matrix which is printed out at the end of the comparison. (See Fig. 8.)

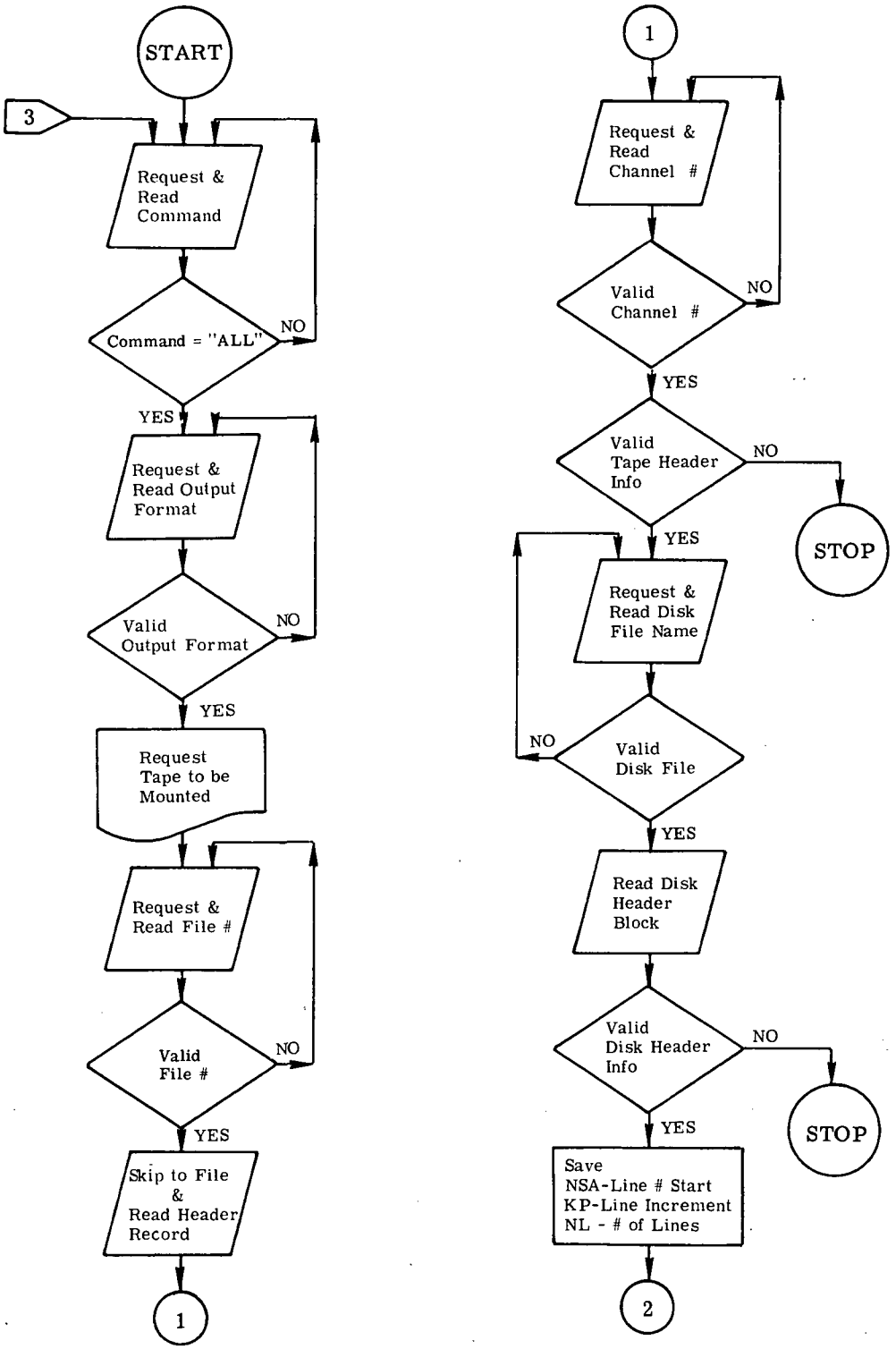


FIGURE 8. FLOWCHART FOR ANAL2 (Continued)

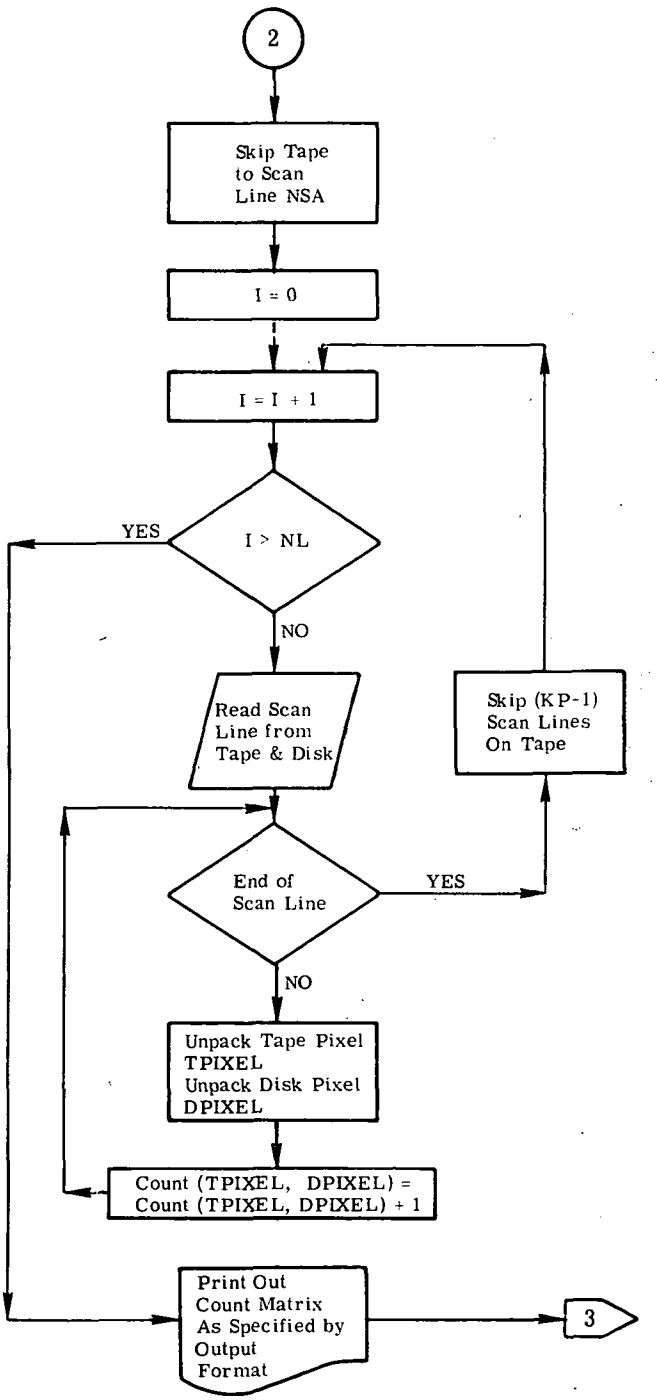


FIGURE 8. FLOWCHART FOR ANAL2 (Concluded)

CALLING SEQUENCES: None. This is a main program.

PARAMETERS: All parameters are read in from the keyboard after a printed request.

1. TYPE COMMAND

This requests the user to specify what analysis mode is to be used. Presently only the ALL command is implemented. All other commands will either be ignored or generate an I/O error.

2. TYPE OUTPUT FORMAT

This requests the user to specify the format in which the count matrix is to be dumped. After analysis has been completed, the count matrix appears as in Table 4.

$$\Sigma C_i = \sum_{i=\emptyset}^{31} \text{COUNT}(i,j)$$

$$\Sigma R_i = \sum_{j=\emptyset}^{31} \text{COUNT}(i,j)$$

COUNT(i,j) = no. of pixels with a tape recognition no. i, and a corresponding disk recognition no. j

When COUNT is typed, output will appear as follows:

TAPE	DISK	COUNT	DISK	TAPE	COUNT
Ø	Ø	COUNT(Ø,Ø)	Ø	Ø	COUNT(Ø,Ø)
Ø	1	COUNT(Ø,1)	Ø	1	COUNT(1,Ø)
Ø	2	COUNT(Ø,2)	Ø	2	COUNT(2,Ø)
⋮	⋮	⋮	⋮	⋮	⋮
Ø	31	COUNT(Ø,31)	Ø	31	COUNT(31,Ø)

TAPE	DISK	COUNT	DISK	TAPE	COUNT
1	Ø	COUNT(1,Ø)	1	Ø	COUNT(Ø,1)
1	1	COUNT(1,1)	1	1	COUNT(1,1)
1	2	COUNT(1,2)	1	2	COUNT(2,1)
⋮	⋮	⋮	⋮	⋮	⋮
1	31	COUNT(1,31)	1	31	COUNT(31,1)

TABLE 4. ARRAY OF DISK FILE AND TAPE FILE RECOGNITION LEVELS

Tape File Recognition Levels	Disk File Recognition Levels												
	Count	Ø	1	2	3	4	5	6	7	8	. . .	31	
	Ø	1ØØ	5	Ø	3	Ø	Ø	Ø	1	12		4	$\Sigma R_{Ø}$
	1	19	2ØØ	5	1	7	6	Ø	3	4		2	ΣR_{1}
	2	2	19	685	Ø	12	Ø	3	24	Ø		5	ΣR_{2}
	3	Ø	27	19	23	14	187	Ø	27	Ø		4Ø7	ΣR_{3}
	4	Ø	15	12	118	137	18	36	Ø	Ø		17	ΣR_{4}
	5	Ø	Ø	13	13	4	12	Ø	45	Ø		24	ΣR_{5}
	6	Ø	Ø	7	Ø	24	Ø	160	Ø	Ø		Ø	ΣR_{6}
	⋮												
	3Ø	7	3	Ø	Ø	3ØØ	Ø	Ø	57	12		4	$\Sigma R_{3Ø}$
	31	6	47	4	3	Ø	7	6	19	22		6	ΣR_{31}
		$\Sigma C_{Ø}$	ΣC_{1}	ΣC_{2}	ΣC_{3}	ΣC_{4}	ΣC_{5}	ΣC_{6}	ΣC_{7}	ΣC_{8}		ΣC_{31}	

When PROPORTIONS is typed, output will appear as follows:

TAPE	DISK	PROPORT	DISK	TAPE	PROPORT
Ø	Ø	COUNT(0,0)/ΣR_0	0	0	COUNT(0,0)/ΣC_0
Ø	1	COUNT(0,1)/ΣR_0	0	1	COUNT(1,0)/ΣC_0
Ø	2	COUNT(0,2)/ΣR_0	0	2	COUNT(2,0)/ΣC_0
⋮	⋮	⋮	⋮	⋮	⋮
Ø	31	COUNT(0,31)/ΣR_0	0	31	COUNT(31,0)/ΣC_0

TAPE	DISK	PROPORT	DISK	TAPE	PROPORT
1	Ø	COUNT(1,Ø)/ΣR_1	1	Ø	COUNT(Ø,1)/ΣC_1
1	1	COUNT(1,1)/ΣR_1	1	1	COUNT(1,1)/ΣC_1
1	2	COUNT(1,2)/ΣR_1	1	2	COUNT(2,1)/ΣC_1
⋮	⋮	⋮	⋮	⋮	⋮
1	31	COUNT(1,31)/ΣR_1	1	31	COUNT(31,1)/ΣC_1

When BOTH is typed, output will be the same as with PROPORTIONS except that the COUNT column of figures will be included.

3. MOUNT REFERENCE TAPE ON 9TRK UNIT 1. RING OUT. THEN TYPE <CR>

This requests the user to mount the ERIM11-formatted recognition tape (see Appendix A) on a 9-track tape drive and set its unit number to 1. After removing the ring and mounting the tape, press the CARRIAGE RETURN button on the terminal to proceed.

4. TYPE 5 DECIMAL DIGIT FILE #

This requests the user to specify which file is to be used on the reference tape. The user must type exactly 5 decimal digits.

5. TYPE 5 DECIMAL DIGIT CHANNEL #

This requests the user to specify which channel of the file is to be used for analysis. The user must type exactly 5 decimal digits.

6. WHICH DISK FILE (USE CSI FORMAT)

This requests the user to specify the name of the contiguous disk file containing the recognition results from classification. Standard CSI format should be used.

DEVICE: FILENAME.EXTENSION [CCID]

e.g., DKØ:OUTFIL.REC[6Ø,4Ø]

DESCRIPTION:

After determining the starting scan line number from the disk file, the reference tape is positioned, and the disk buffer and the tape buffer are filled. A pixel is unpacked from the tape file and the disk file. The 5 least significant bits are extracted from each pixel. The tape value is used as the first subscript i, and the disk value is used as the second subscript, j. The corresponding location in the COUNT matrix (COUNT(i,j)) is incremented, and the process repeats. After all comparisons have been made with the disk file, the COUNT matrix is printed out in the specified format.

ERROR MESSAGES:

1. ERR1, I/O ERROR
   PC = XXXXXX SUBRTN
   This indicates that an I/O error occurred at location XXXXXX within subroutine SUBRTN.

2. ERR2, NONDECIMAL DIGIT FOUND
   SUBROUTINE-SUBRTN
   When converting a 5-ASCII-character field to its binary integer form, a character was detected that was not a numerical digit in subroutine SUBRTN.

3. ERR3. NUMBER >65535.
   SUBROUTINE-SUBRTN
   After conversion of a 5-ASCII-digit number, the resultant number was too large for a PDP-11/45 word. The error occurred within subroutine SUBRTN.

4. ERR4. INVALID FILE #.
   The file number requested was less than or equal to Ø.

5. ERR5. NO EXTENTS ALLOWED FOR THIS PROGRAM VERSION.
   This program and the present computer system aren't equipped to handle scan lines with more than 8126 pixels in all channels.

6. ERR6. 0 ≥ CHANNEL # > NCHAN
   The desired channel number was less than or equal to zero, or it was greater than the number of channels on the tape.

7. ERR7. INVALID CSI FORMAT
   The desired disk file name was not requested with a format intelligible to the command string interpreter.

8. ERR8.1. FILE DOESN'T EXIST
   The desired classification disk file doesn't exist.

9. ERR8.2. FILE IS LINKED
   The desired classification disk file was a linked file, not a contiguous file.

10. ERR9. INB NOT EQUAL TO FILE SIZE
    The disk file header specified a different file size than was actually there.

11. ERR10. NPTS NOT EQUAL TO NSS
    The disk file header specified a scan line length different from the reference tape.

12. ERR11. NWB > 4Ø96
    This disk file header specified an outblock (a set of data on disk transferred as a unit) with more than 4Ø96 words.

13. ERR12. UNHANDLEABLE DATA FORMAT.
    This program is set up to handle only scan lines where the pixels are packed 2 per word.

14. ERR13. INVALID COMMAND.
    A command was specified other than AL[L]
    IN[DIVIDUAL]
    SE[ARCH]
    CH[ECK]

15. ERR14. INVALID OUTPUT FORMAT.
    An output format was specified other than CO[UNT]
    PR[OPORTIONS]
    BO[TH]

16. ERR15. DISK FILE SIZE EXCEEDED.
    A scan line was requested that didn't exist within the disk file.

17. ERR25. STARTING POINT # NOT = 1.
    The starting pixel number specified in the disk file header was not equal to 1.

18. ERR26. POINT # INCREMENT NOT = 1.
    The increment between pixels specified in the disk file header was not equal to 1.

DEVICE REQUIREMENTS: Keyboard Input
Keyboard Output
Magnetic Tape Input, 9-Track, 8ØØbpi
Disk File Input

ROUTINES REQUIRED:

RESDNT - main program-determining sequence in which the other routines are performed.

ALL - calls routines to get the appropriate scan lines off tape and disk and to load the COUNT matrix.

DECODE - reads command line and output format line, checks validity, and initializes the program.

DINIT - reads the desired disk file header, checks it for validity, and initializes the appropriate variables.

DSUBSQ - determines displacement of an arbitrary variable in an N×N array.

GETDSK - gets the block containing the desired scan line and sets a pointer to the desired pixel.

GETTAP - gets the desired scan line from the reference tape and sets a pointer to the desired pixel.

LDMTS - unpacks a pixel from the tape and disk buffer and increments the corresponding location in the COUNT matrix.

PRINT - prints out the results accumulated in the COUNT matrix according to the format specified by the user.

SEEKFL - finds the location of a specified file on the disk and returns the block # and file type.

SKIPMT - skips an arbitrary # of files and records, forward or backward on a given magnetic tape unit.

TINIT - reads the desired file's header record, checks it for validity, and initializes the appropriate variables.

FINLIB - library of FORTRAN subroutine modules.

MACROS REQUIRED: .WRITE, .WAIT, .RSLE, .INIT, .READ, .CSI1, .CSI2, .TRAN, .PARAM, .BIN20, .EXIT, .LOOK, .SPEC, .D2BIN

D. RSAV - pushes registers 5 → 0 into the stacks.

D. RRES - pops the stack into registers 0 → 5.

STORAGE REQUIREMENTS: 73762 octal locations.

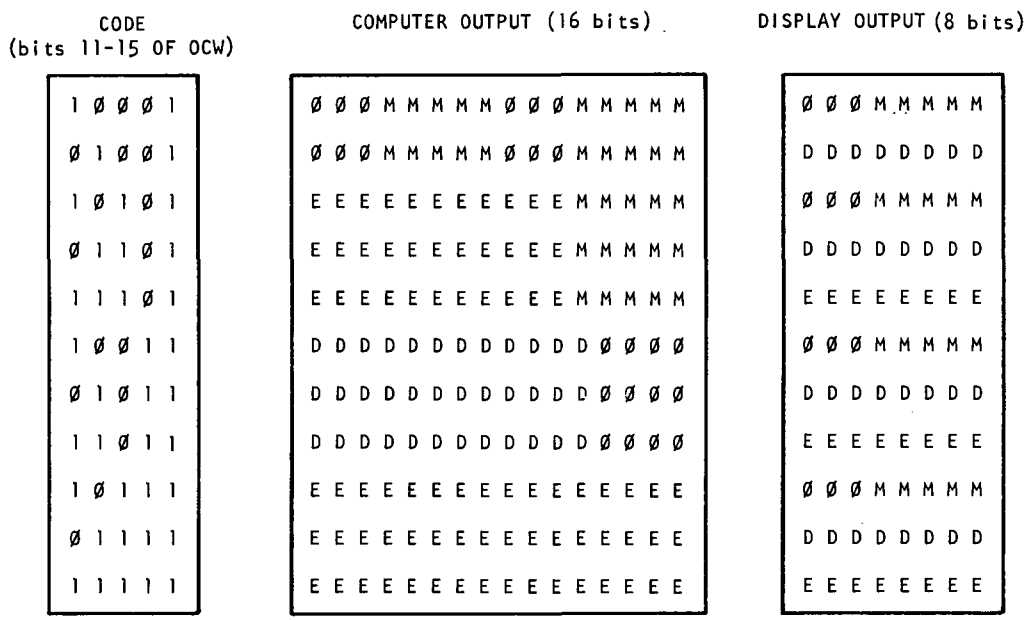
3
DIAGNOSTIC BUS STRUCTURE

Incorporated into the hardware of the MIDAS Classification System is a network which allows selective interrogation of the state of the Classifier. This hardware network is referred to in this report as the diagnostic bus.

The diagnostic bus is a relatively simple structure which operates in conjunction with the diagnostic/output card. At certain points along the processor pipeline, three-state latches are wired in parallel with the processor data lines. The ports configured in this manner are: (1) the input to the mean card, (2) the output of the mean card, (3) the output of the variance card, (4) the output of the matrix-multiplier cards, (5) the output of the square card, and (6) the output of the square-accumulator card. The diagnostic output card is conditioned via one of the DR-11B DMA interfaces to interrogate a particular latch at a given time in the cycle of the classifier clock. The interrogation description scheme is described in Fig. 9. The 16-bit word sent to the diagnostic output card via the DR-11B DMA interface is called the output code word. It is this word which selects the output(s) which will be passed from the MIDAS System to the external world.

Furthermore, this figure describes the relation of the 5 high-order bits of the output code word to the format of the data passed from the diagnostic/output card. Figure 10 details the significance of the low-order 11 bits and how the variables of bay number, card number, and time are combined in the output code word.

Thus, if one wished to have a 12-bit diagnostic returned to the computer from the classifier, such that the square card in bay 2 at time 14 was interrogated, the bit pattern comprising the output code word might be $11\emptyset11\emptyset1\emptyset1\emptyset1\emptyset111\emptyset_2 = 155256_8$.



Key: 1 - bit = 1

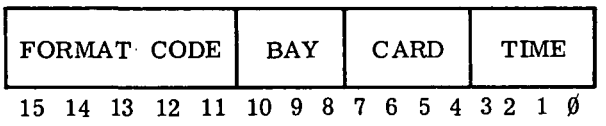
Ø - bit = Ø

M - Material Code (object class)

D - Diagnostic Output

E - Exponent Output

FIGURE 9. INTERROGATION DESCRIPTION SCHEME



Card Assignments for Diagnostic Purposes

Mean Input	- 12
Mean Output	- Ø
Variance Output	- 1
MTX #Ø Output	- 2
MTX #1 Output	- 3
MTX #2 Output	- 4
MTX #3 Output	- 5
MTX #4 Output	- 6
MTX #5 Output	- 7
MTX #6 Output	- 8
MTX #7 Output	- 9
Square Output	- 1Ø
Sq. Acc. Output	- 11

FIGURE 10. OUTPUT CODE-WORD FORMAT

Appendix

ERIM11 - MULTISPECTRAL TAPE FORMAT FOR THE PDP-11/45 SYSTEM

1. The maximum physical record length is 4Ø96 16-bit words.
2. A logical record (one scan line) consists of "extents" (physical tape records). The first extent of each logical record contains a header of control information followed by the data. Each additional extent will contain only data.
3. Each extent, with one exception, will be a physical record with 4Ø96 words. The exception is the last extent of each logical record which will be only as long as needed to complete that logical record.
4. At the beginning of each multispectral file is a 4Ø96-word header record with the title and other control information.

HEADER RECORD FORMAT:

WORD	VARIABLE NAME	DESCRIPTION
1-4	TITLE	ERIM11
5	NSA	Starting scan line number
6	KS	Scan line number increment
7	NA	Starting pixel number
8	NB	Ending pixel number
9	KP	Pixel number increment
10-11	BANG	Angle of first sample
12-13	DANG	Angle between first and second samples
14	NCHAN	Number of channels
15	NSS	Number of pixels/scan line
16	EXTENT	Number of extents minus 1
17	LENGTH	Number of words in last extent
18-21	SWTCHS	64 single-bit switches for communication between programs
22	HISTRY	Number of programs that have massaged the data in this file
23-29Ø		Program names in RADIX-50 notation
291-1296	EXTRA	Additional processing information
1297-3Ø96	CONTROL	Additional control information
3Ø97-3168	DESCRP	ASCII description of file
3169-4Ø96	ANNOTA	ASCII annotation information

FIRST EXTENT FORMAT:

WORD	VARIABLE NAME	DESCRIPTION
1-2	RECD	Digital number of next scan line
3-4	RECA	Analog number of next scan line
5-33	CONTRL	Scan line control information
34-4Ø96	DATA	Scan line data

SUBSEQUENT EXTENT FORMAT:

WORD	VARIABLE NAME	DESCRIPTION
1 → 4Ø96	DATA	Scan line data
1 → LENGTH	DATA	Scan line data (last extent of logical record)

DISTRIBUTION LIST

NASA Langley Research Center
Hampton, VA 23665

ATTN: Report & Manuscript Control Office, Mail Stop 180A (1)
ATTN: Raymond L. Zavasky, Mail Stop 115 (1)
ATTN: Technology Utilization Office, Mail Stop 193A (1)
ATTN: William M. Howle, Jr., Mail Stop 470 (10)

NASA Ames Research Center
Moffett Field, CA 94035

ATTN: Library, Mail Stop 202-3 (1)

NASA Flight Research Center
P.O. Box 273
Edwards, CA 93523

ATTN: Library (1)

NASA Goddard Space Flight Center
Greenbelt, MD 20771

ATTN: Library (1)

NASA Lyndon B. Johnson Space Center
2101 Webster Seabrook Road
Houston, TX 77058

ATTN: Library, Code JM6 (1)

Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91103

ATTN: Library, Mail 111-113 (1)

NASA Lewis Research Center
21000 Brookpark Road
Cleveland, OH 44135

ATTN: Library, Mail Stop 60-3 (1)

NASA John F. Kennedy Space Center
Kennedy Space Center, FL 32899

ATTN: Library, IS-DOC-1L (1)

NASA Marshall Space Flight Center
Huntsville, AL 35812

ATTN: Library (1)

National Aeronautics & Space Administration
Washington, D.C. 20546

ATTN: Kss-10/Library (1)
ER/NASA Headquarters (1)

NASA Scientific & Technical Information Facility
P.O. Box 33
College Park, MD 20740 (plus 1 reproducible) (27)