AD/A-001 563

DESIGN AND USE OF COMPUTER-GENERATED ELECTRONIC AREA NAVIGATION MAP DISPLAYS: HARDWARE AND SOFTWARE DEVELOPMENT

Gerald Wolfson, et al

Hughes Aircraft Company

Prepared for:

Federal Aviation Administration

April 1974

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
FAA-RD-74-129		AD/A-001563

4. Title and Subtitle	5. Report Date
Design and Use of Computer-generated Electronic Area Navigation Map Displays, Hardware and Software Development	April 1974
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Gerald Wolfson, J. William Weber	P74-165

9. Performing Organization Name and Address	10. Work Unit No.
Display Systems & Human Factors Dept., Hughes Aircraft Co., Centinela & Teale Sts., Culver City, CA. 90230	11. Contract or Grant No. DOT-FA72WA-2761
	13. Type of Report and Period Covered Final Report February 1973 - September 1974

12. Sponsoring Agency Name and Address	14. Sponsoring Agency Code
Dept. of Transportation Federal Aviation Administration Systems Research & Development Service Washington, D.C. 20590	

15. Supplementary Notes

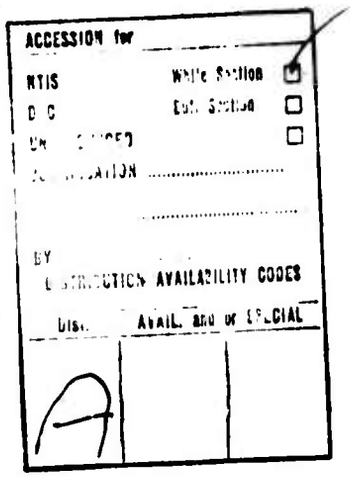
16. Abstract

This report documents hardware development of a flexible display generator and CRT including a software package to be used for the evaluation of electronic cockpit display formats as originally addressed in the Part I report, FAA-RD-72-122. The software, written for a Raytheon 704 computer, enables the generation of electronic map displays that provide both area navigation and time navigation guidance. The hardware was interfaced with a Ratheon 704 computer and designed for installation into a flight simulator.

17. Key Words	18. Distribution Statement
Area Navigation Flexible Cathode Ray Tube Display Display Generator Symbol Generator	Document is available to the public through the National Technical Information Service, Springfield, Virginia 22151.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
UNCLASSIFIED	UNCLASSIFIED	146	5.75

Form DOT F 1700.7 (8-69)



The contents of this report reflect the views of the Hughes Aircraft Company which is responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policy of the Department of Transportation. This report does not constitute a standard, specification or regulation.

FOREWORD

This report covers the work accomplished under **Part** II of Contract DOT-FA72WA-2761; Design and Use of Computer - Generated Electronic Area Navigation Map Displays. The contracting officer's representative for the efforts reported herein was D. Michael Brandewie of the Systems Research and Development Service, Federal Aviation Administration.

The work was accomplished by the Display Systems and Human Factors Department of Hughes Aircraft Company under the direction of Mr. G. Wolfson, who was Project Engineer. Special acknowledgement is given to the following individuals who contributed substantially to the technical effort:

E. A. Lawson
R. W. Lowe
J. R. Phelps
J. W. Weber

Preceding page blank

CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Continued)

LIST OF TABLES

1.0 INTRODUCTION AND SUMMARY

Reorganization of the U.S. National Airspace System for operations during the 1975-1990 period will rely heavily on airborne volumetric navigation, including horizontal (RNAV), vertical (VNAV), and time (TNAV) guidance. Ground based elements of the air traffic system for the most part will perform permissive rather than directive functions, clearing air traffic to occupy protected airspace within prescribed limits and monitoring compliance by the airborne elements. Radar vectoring services will continue to be provided, but their use will be limited; compliance with the time-referenced clearances issued will be the responsibility of the flight crews and their onboard sensors, computers, and displays.

The return of all navigation functions to the cockpit increases the premium on reliability and precision not only of the airborne sensing and computing elements but also of the controlling elements, the most critical of which are the flight crew. The Systems Research and Development Service of the Federal Aviation Administration is conducting a broad research and development program to measure empirically the magnitude and frequency of pilotage errors as a function of major display and control design variables for various classes of pilots. In support of the overall FAA program objectives, a two-part program was conducted. Part I consisted of five separate but related studies. The results of Part I have been documented in report number FAA-RD-72-122, DESIGN AND USE OF COMPUTER-GENERATED ELECTRONIC AREA NAVIGATION MAP DISPLAYS, dated August 1973. The five studies included the following:

1. Terminal Area Guidance Study
2. Data Entry Study

3. Data Presentation Study
4. Symbol-Generation Design Tradeoff Study
5. Flexible EMD Generation and Presentation System Design

Part II consisted of the design, fabrication, and checkout of a flexible display generator and CRT including a software package to be used in the evaluation of electronic cockpit display formats. The development included three tasks which are documented in this report. These tasks were:

1. Flexible Electronic Map Display (EMD) Generation and Presentation System Development
2. EMD Software Development
3. EMD System Integration

These tasks are briefly summarized in the following paragraphs.

FLEXIBLE EMD GENERATION AND PRESENTATION SYSTEM DEVELOPMENT

An Electronic Map Display (EMD) consisting of a computer specific interface, a flexible display generator and a CRT display was fabricated and integrated into a working EMD system. The EMD system which is capable of presenting computer-generated cartographic and guidance symbology was designed and fabricated to meet specific requirements and recommendations resulting from the efforts concluded under Part I of this program. The system is compatible with a Raytheon 704 computer, is suitable for installation in a flight simulator cockpit, and provides experimental flexibility in the manipulation of information content presentation variables.

EMD SOFTWARE DEVELOPMENT

Utilization of the EMD hardware required the development of an efficient EMD software system. An important task of this program was the development of such a software package designed specifically for a Raytheon 704 computer. In order to facilitate development of individual EMD features and accommodate future development efforts, a modular approach

to the programming was taken. Four software packages were developed which include the following:

1. Display Utility Routine
2. Map Data Pre-Processor
3. Data Retrieval
4. Category Processor

Completion of an entire software package would require the development of two additional functions, namely, simulator interface and overall timing and control.

EMD SYSTEM INTEGRATION

The hardware and software developed under the first two parts were effectively integrated with a Raytheon 704 computer in the flight simulation laboratory of the University of Illinois. The EMD system will be used for the conduct of area navigation studies under the sponsorship of the FAA. The unique flexibility and capability of the EMD system was demonstrated through the use of a software package designed specifically for this purpose.

2.0 FLEXIBLE EMD GENERATION AND PRESENTATION SYSTEM HARDWARE

An EMD system employing a CRT for presenting computer generated cartographic and guidance symbology was developed to meet the specific requirements and recommendations in the Part I final report. A photograph of the actual hardware is shown in Figure 2-1. The hardware consists of the following units:

1. Programmable Display Generator - This unit contains the computer specific interface for the Raytheon 704 computer, a custom designed stroke display generator and a CRT display interface.

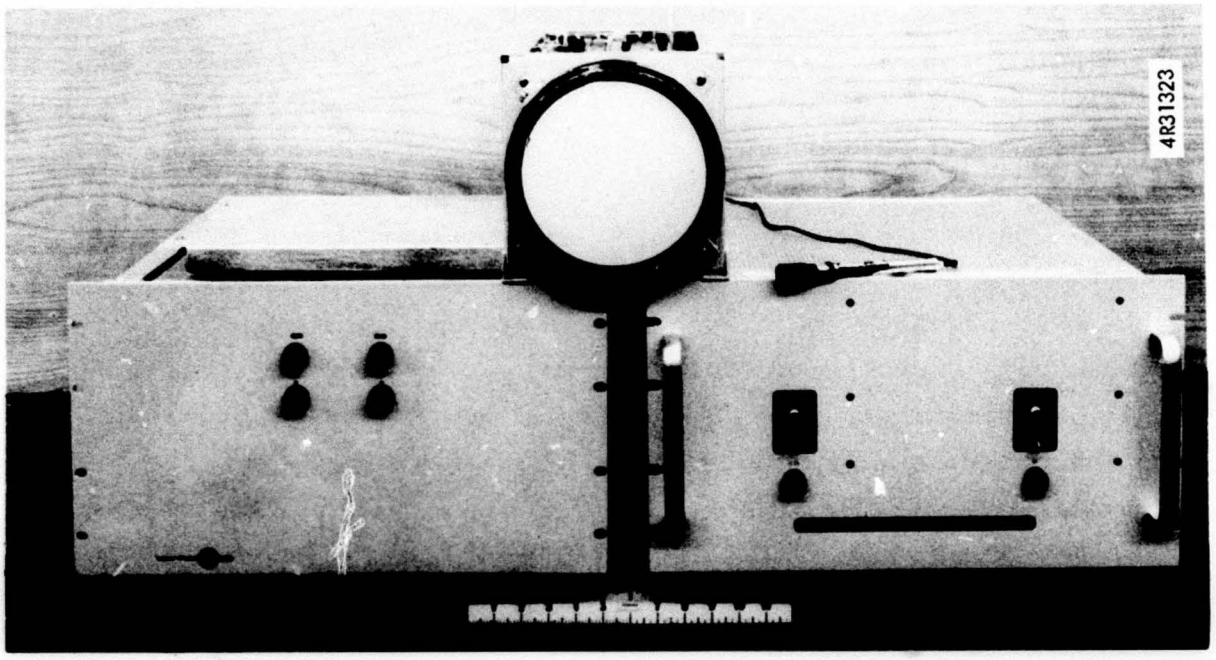


Figure 2-1. EMD system hardware.

2. EMD display - The display consists of a high bandwidth, X-Y deflection system and a remote indicator unit containing the high voltage power supply, video amplifier and 6 inch cathode ray tube.

3. Data input device - This device consists of a light pen and associated electronics used during conduct of the data input studies.

The following sections describe the system hardware. A detailed description of the display generator and hardware was presented in the Part I final report. This description has been updated in this report, reflecting the fabrication of the hardware and incorporating modifications required to achieve the specified performance.

DISPLAY GENERATOR DESIGN

Introduction

The Hughes custom display generator has been designed to meet the specific requirements and recommendations resulting from the symbol generator tradeoff study conducted under Part I. The symbol generator was designed to interface with a Raytheon 704 digital computer and to be integrated into an experimental EMD system suitable for studying pilot performance and workload reduction in civil area navigation operations.

System Design Features

The display generator is a microprogrammed display processor designed for general purpose system applications. The functional, mechanical, and electrical design is such that the system is exceptionally useful for the laboratory evaluation of display symbology.

The design concepts embodied in the Hughes display generator provide a combination of features unavailable in commercially available display generator systems. These include:

- SPEED

  The system may accept information from the computer at a maximum of 3 MHz (nominal design point 1 MHz). The internal clock determines the writing rate of the programmed dot and the system is conservatively designed to run at a maximum rate of

3 MHz (nominal design point 2 MHz). This corresponds to a maximum writing rate of 3,000,000 display elements per second and a maximum writing speed of 21,000 inches per second referenced to a 7 inch display.

- FLEXIBILITY

  The system has inherent flexibility in adapting to experimental display formats, symbol shapes, etc., since it is a computer software controlled segment chaining system.

- EFFICIENCY

  The system efficiency approaches 100 percent when the display format calls for a very large number of chained segments. A statistical analysis of chained data in complex displays, such as electronic map displays, precipitated the development of a unique segment chain encoding scheme designated, "relative chain encoding." This development, when compared with other encoding methods, significantly decreases the memory requirement for a given display list. The result of a considerable reduction in the data that must be processed. It is anticipated that the relative chain encoding will be used for 90 percent of all vectors and irregular shapes. Absolute chain encoding will be used for all alphanumerics.

- ECONOMY

  The display generator uses TTL monolithic integrated circuits, including a large number of medium scale integration types, for savings in parts and assembly time. The use of read only memories for control functions further reduces the number of circuits that might otherwise be required to provide the same functional capability.

- OPERATING CHARACTERISTICS

  16-bit instruction and data words, 2's complement setup data.

  Up to 3-MHz computer data channel word rate.

  Dual alternating 16-word input data buffers to allow efficient block data transfer when directly attached to a processor DMA port.

  Up to 4K of random access refresh memory may be used to minimize loading on the central processor. The random access feature permits updating of selected portions of the memory at more rapid rates.

  Branch and subroutine capability when processing a refresh memory display list.

  Video On-Off control for up to 4 displays.

  Eight levels of video with provision for defining hue and saturation.

Automatic blanking of symbology that exceeds the display format.

1024 display element positioning in vertical and horizontal.

Variable major positioning delay to increase display efficiency.

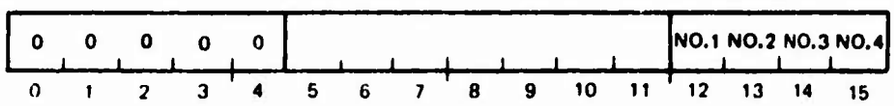
Rotation of vector about the vector start point (start point transformation under computer control).

Instruction Repertoire and Data Format

The instruction repertoire performs the following functions: Interface control, memory reference, video control, position control, chain setup, chain commands, chain data format, and end of display control. The instruction list is shown in Figure 2-2. The left hand column indicates the functions of the instruction, video setup, position setup, chain setup, chain, and interface. The second column contains an abbreviated form of the instruction name. Information from the CPU comes in 16 bit words numbered 0 to 15 as shown at top of Figure 2-2.

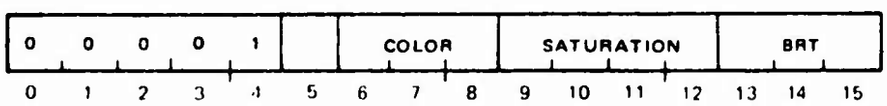
The instruction formats are described as follows:

Indicator



Bits 12 through 15 control the video drive to indicators (displays) 1 through 4. A one = video on, a zero = video off. Bits 5 through 11 are not used.

Load BRT



The lower three bits, bits 13 through 15, of the 10-bit data field are placed in the BRT register and define 8 possible brightness levels. The next four

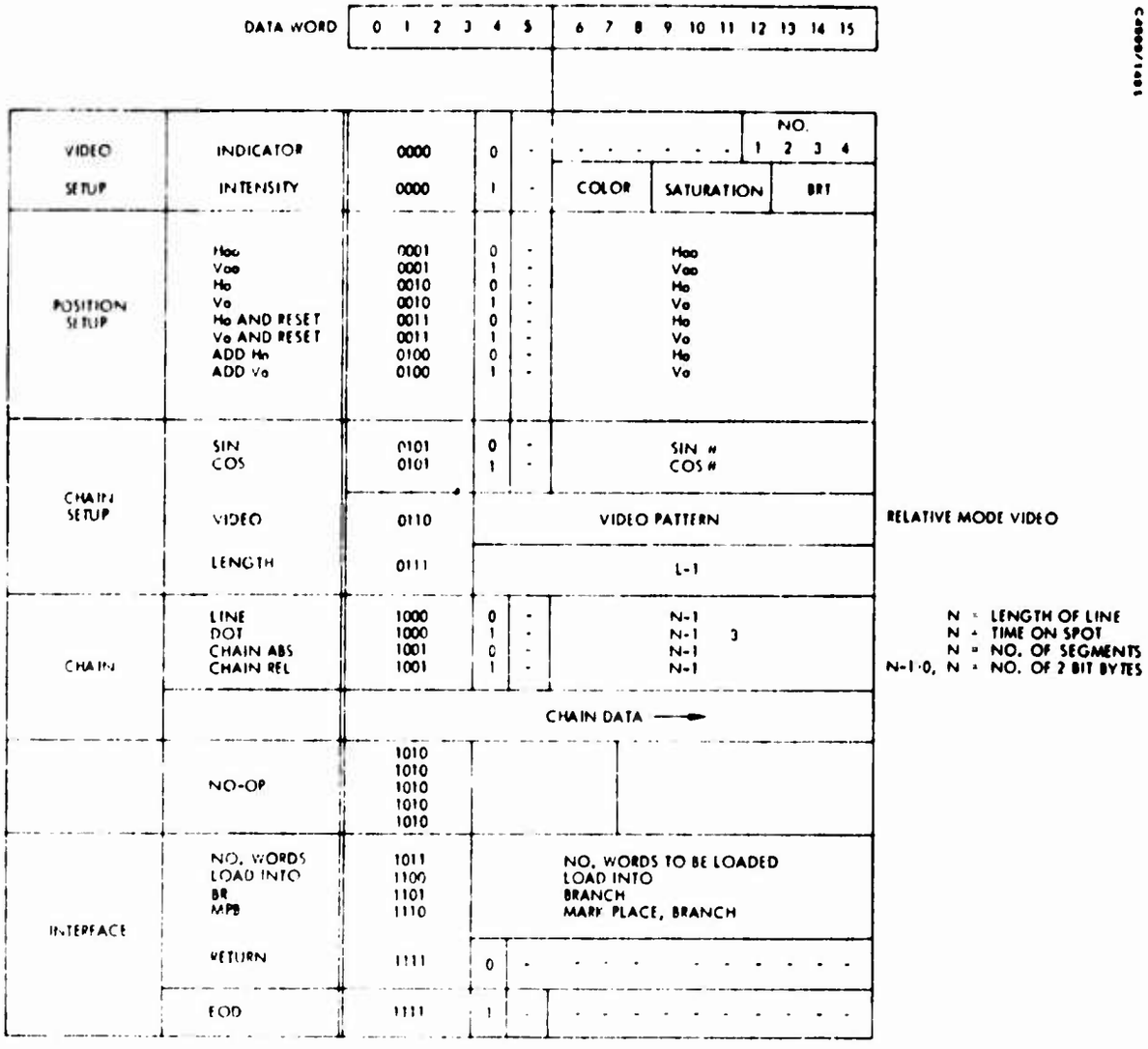
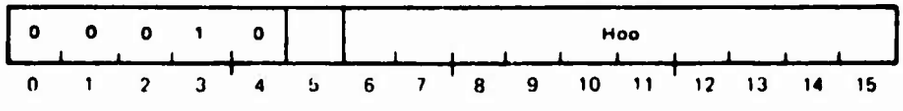


Figure 2-2. Display generator instruction list.

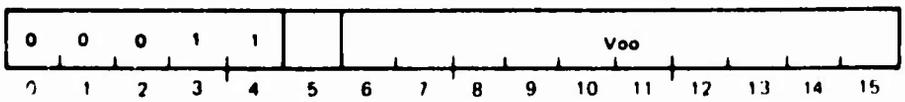
bits, bits 9 through 12, are dedicated to future color saturation capability. The next three bits, bits 6 through 8 are dedicated to color hue capability.

Load Hor Bias (Hoo)



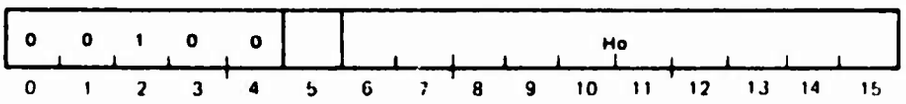
The contents of the 10-bit data field are placed in the Hoo register. This data acts to bias the display in the horizontal direction.

Load Vert Bias (Voo)



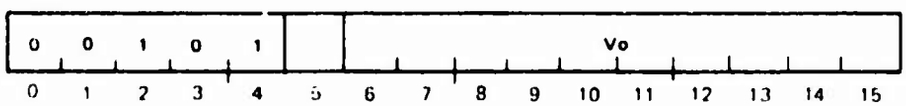
The contents of the 10-bit data field are placed in the Voo register. This data acts to bias the display in the vertical direction.

Load Hor Position (Ho)



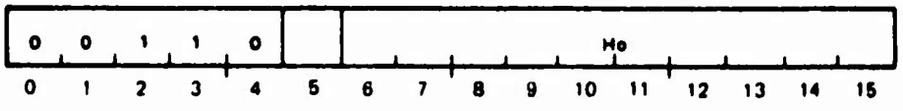
The horizontal reset flag is set to zero. The H accumulator is cleared and the contents of the 10-bit data field are added to the H accumulator via the Ho register. The data is usually a coordinate transformed position.

Load Vert Position (Vo)



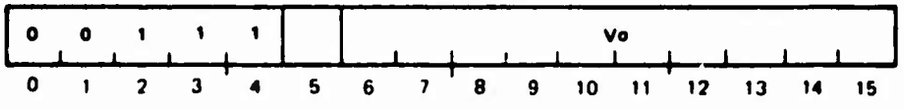
The vertical reset flag is set to zero. The H accumulator is cleared and the contents of the 10-bit data field are added to the V accumulator via the Vo register. The data is usually a coordinate transformed position.

Load "Ho With Reset"



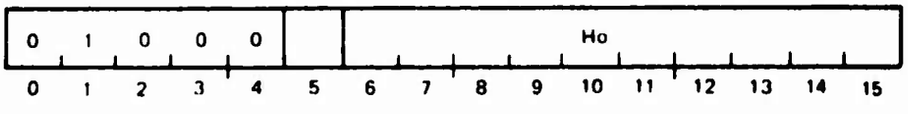
The horizontal reset flag is set to one. The H accumulator is cleared and the contents of the 10-bit data field are added to the H accumulator via the intermediate Ho register. The Reset Flag acts to reset the accumulator to the Ho data (contained in the Ho register) at the end of every display chain. The reset flag can only be cleared by using a LOAD HOR POSN instruction. The H_o data is usually a coordinate transformed position.

Load "Vo With Reset"



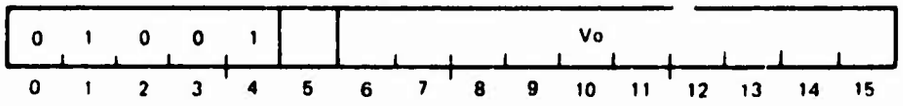
The vertical reset flag is set to one. The V accumulator is cleared and the contents of the 10-bit data field are added to the V accumulator via an intermediate Vo register. The reset flag acts to reset the accumulator to the Vo data (contained in the Vo register) at the end of every display chain. The reset flag can only be cleared by using a LOAD VERT POSN instruction. The Vo data is usually a coordinate transformed position.

Add Ho



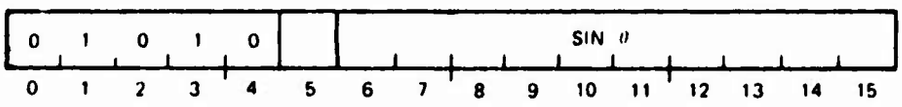
The contents of the 10-bit data field are added to the existing data in the H accumulator via an intermediate Ho register.

Add Vo



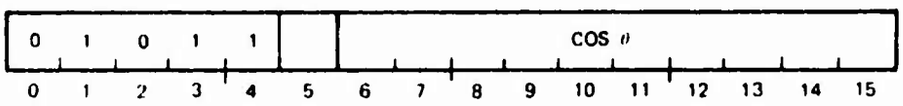
The contents of the 10-bit data field are added to the existing data in the V accumulator via an intermediate Vo register.

Load SIN



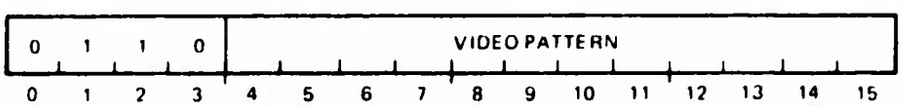
The contents of the 10-bit data field are placed in the SIN register. This data is used in creating rotated display chains. This data does not affect the values of H_o, V_o, H_{oo}, or V_{oo}. The decimal point is between bits 7 and 8.

Load COS



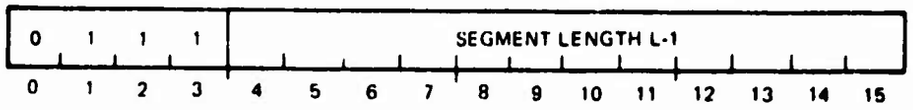
The contents of the 10-bit data field are placed in the COS register. This data is used in creating rotated display chains. This data does not affect the value of H_o, V_o, H_{oo}, or V_{oo}. The decimal point is between bits 7 and 8.

Load VID Pattern



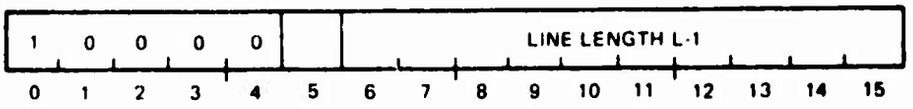
The contents of the 12-bit data field are placed in the video pattern register. This data affects only the Relative Mode video and rotates by one bit at the end of every segment. The leftmost bit is displayed first.

Load Seg Length



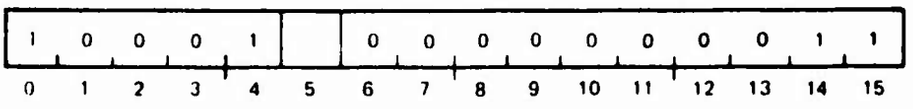
The contents of the 12-bit data field are placed in the segment length register. The binary order of the data should be the segment length minus 1 in X-Y display elements. In the relative mode of operation the value of n must be >1. In the absolute mode, n may be equal to 1.

Draw Line



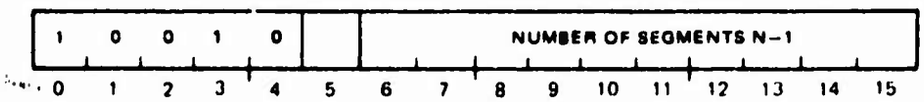
The contents of the 10-bit data field are placed in a special length counter and immediately a single chain n elements long is drawn in the x positive direction (to the right). The binary value of the data should be the line length minus 1 in X-Y display elements. The start point is wherever the beam is positioned when the instruction is implemented. If SINCOS data is defining a rotation angle θ, then the line drawn will be rotated by θ about the start point.

Draw Dot



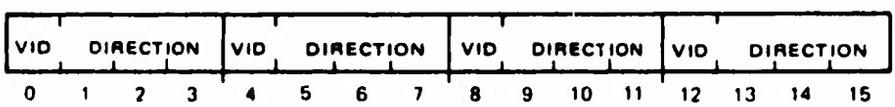
The contents of the 10-bit data field are placed in the clock time counter and used to determine the internal clock periods (n) that the dot will be displayed. The usual duration is (n - 1) = 3.

Define Chain Absolute (CH ABS)



The contents of the 10-bit data field (n - 1) are placed in the segment counter and define the number of segments (n) contained in the chain data. This control word must be immediately followed by the required number of absolute chain data words.

Absolute Data

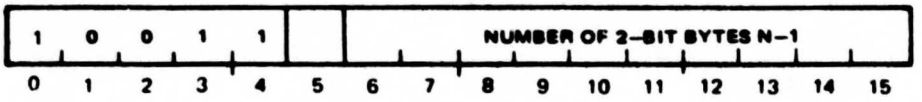


Must be preceded by a Define Chain Absolute instruction. Absolute segments are defined in 4-bit bytes where bit-0 is a video bit (1 = on, 0 = off). Bits 1-3 represent 8 possible segment directions (rotated further by θ). If North is up, the directions defined by bits 1-3 are as follows:

000	East	100	West
001	NE	101	SW
010	N	110	S
011	NW	111	SE

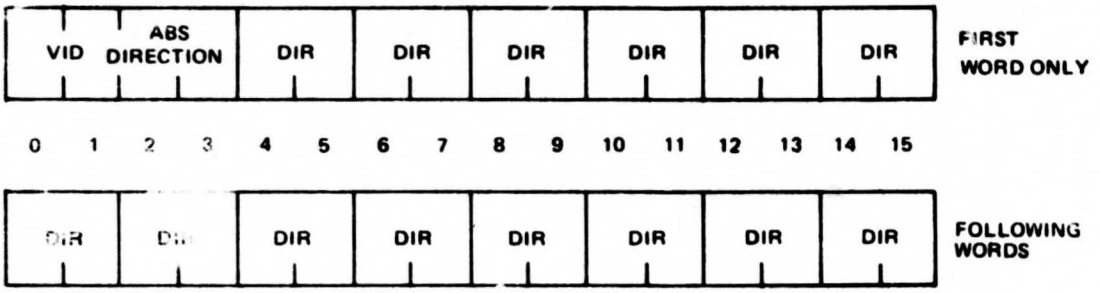
Each of the N segments defined by the control word will be of length L. (L is defined by the contents of the segment length register.)

Define Chain Relative (CH REL)



The contents of the 10-bit data field (n - 1) are placed in the nibble counter to define the number (n) of 2-bit nibbles in the chain data. (The first segment counts as two bytes.) This control word must be immediately followed by the required number of related chain data words.

Relative Data



Must be preceded by a Define Chain Relative instruction. The very first segment is coded by a 4-bit absolute code to establish an initial direction and video condition. All following segment data is encoded in the 2-bit relative code (see Figure 2-3). This permits (except for the first data word) packing of eight segments direction instructions per 16-bit word, with video being established from the repetitive use of the twelve bits contained in the video pattern register. The first bit in the video pattern register is masked by the video condition bit (bit 0) in the first segment of the first relative data word. This scheme is optimum for the defining of geographical features, where few sharp turns must be drawn. For example, if the present direction is called North, the next direction will be :

North if code is 00
NW is code is 01

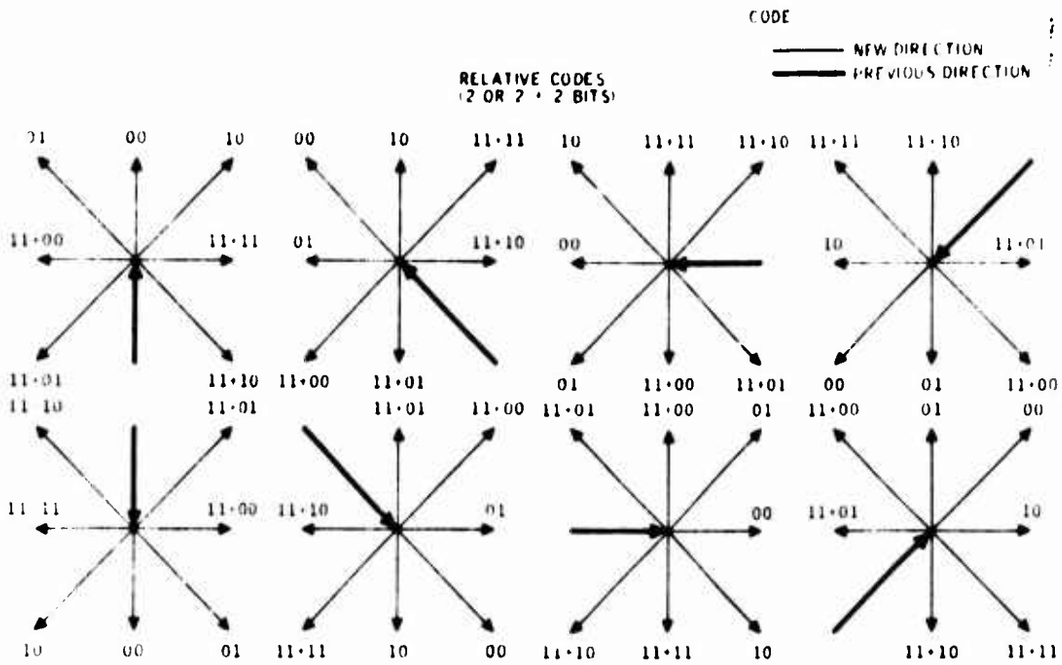
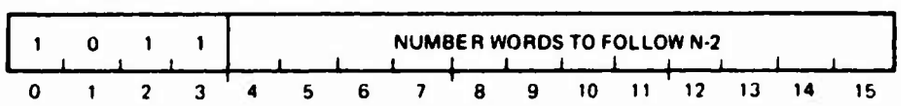


Figure 2-3. Segment chain direction encoding.

NE if code is 10
W if code is 1100
SW if code is 1101
SE if code is 1110
E if code is 1111

The segments will be of length L, where L is defined by the contents of the segment length register. The control word does not implicitly define the number of segments but rather the 2-bit nibbles used on data.

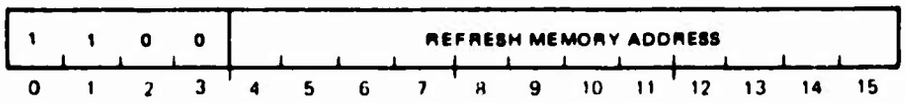
Define List



The contents of the 12-bit data field are used to define the size of the display list to be processed from the data transmitting device (processor). This

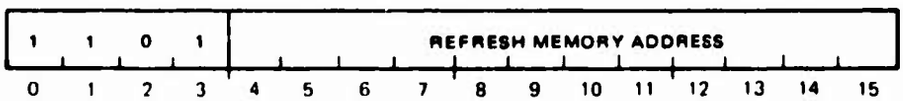
must be the very first word received following the display sync status (interrupt) sent to the processor. The magnitude of the data is n - 2 where n is the size of the list not including this word.

Load Into (Store)



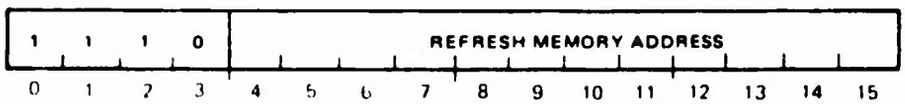
The contents of the 12-bit data field are used to define the start address for refresh memory loading. This instruction is ignored if the display generator does not have the memory option.

Branch (BR)



This instruction is a simple branch instruction. The contents of the 12-bit data field define the refresh memory location of the next data item when the refresh memory is in a readout mode. This instruction is ignored if the display generator does not have the memory option.

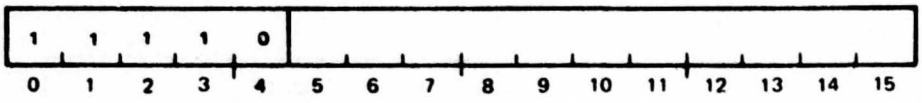
Mark Place and Branch (MPB)



The contents of the refresh memory address register are stored and the contents of the 12-bit field place in the memory address register to define the next refresh memory access location when the memory is in the readout mode. This instruction is used in conjunction with the RETURN instruction

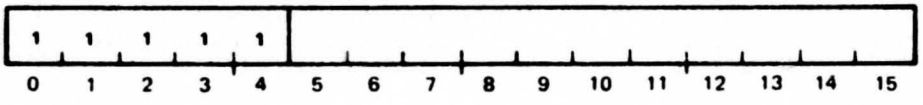
for subroutine operations. This instruction is ignored if the display generator does not have the refresh memory option.

Return



This instruction forces a return to the memory address location saved as a result of a previous MARK PLACE AND BRANCH instruction when the refresh memory is in the readout mode. This instruction is ignored if the display generator does not have the refresh memory option.

End of Display (EOD)



This instruction sets a flag to indicate the physical end of the display list and must be the very last word in any complete display list regardless of whether or not a refresh memory is used. This instruction should not be used when updating only portions of the refresh memory unless the previously issued EOD is eliminated by the new data.

System Organization

The display generator is organized as shown in Figure 2-4.

Computer/Display-Generator Interface

The function of the processor/display generator interface is to control data flow into the display generator. The interface requests processor memory cycles, utilizing the maximum output capability of the Raytheon CPU up to a 3 MHz transfer rate. Available processor timing signals are

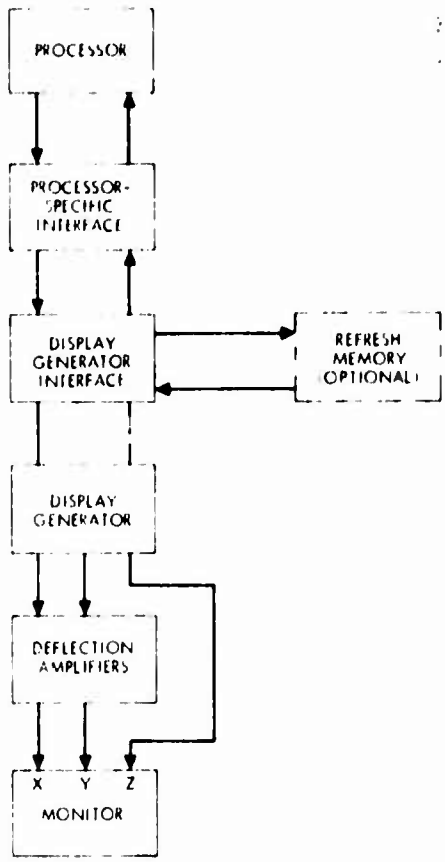


Figure 2-4. Display generator system block diagram.

used to synchronize the processor specific interface. Data is requested in blocks of 16 words and stored in a 32 word parallel alternating buffer. The data is made available to the main logic on a need basis under the control of the main logic clock and data request control. Block transfer of data results in greater display generator efficiency and allows the use of CPU block transfer instructions. Synchronization to the processor DMA is also enchanced.

The interface may be used either with or without the refresh memory option. The refresh memory option has the advantage of decreasing to a considerable degree the burden of data transmission from the CPU to the display generator. The display list need not be transmitted at the nominal 60 Hz rate, but need only be sent at major update intervals. Minor changes to the display list can still occur at the 60 Hz frame rate.

When in a memory readout mode, the data is transferred from memory in the same fashion as from the CPU. The display list is stored in the refresh memory and is transferred in blocks of 16 words to the interface output buffer where it is then available to the display generator.

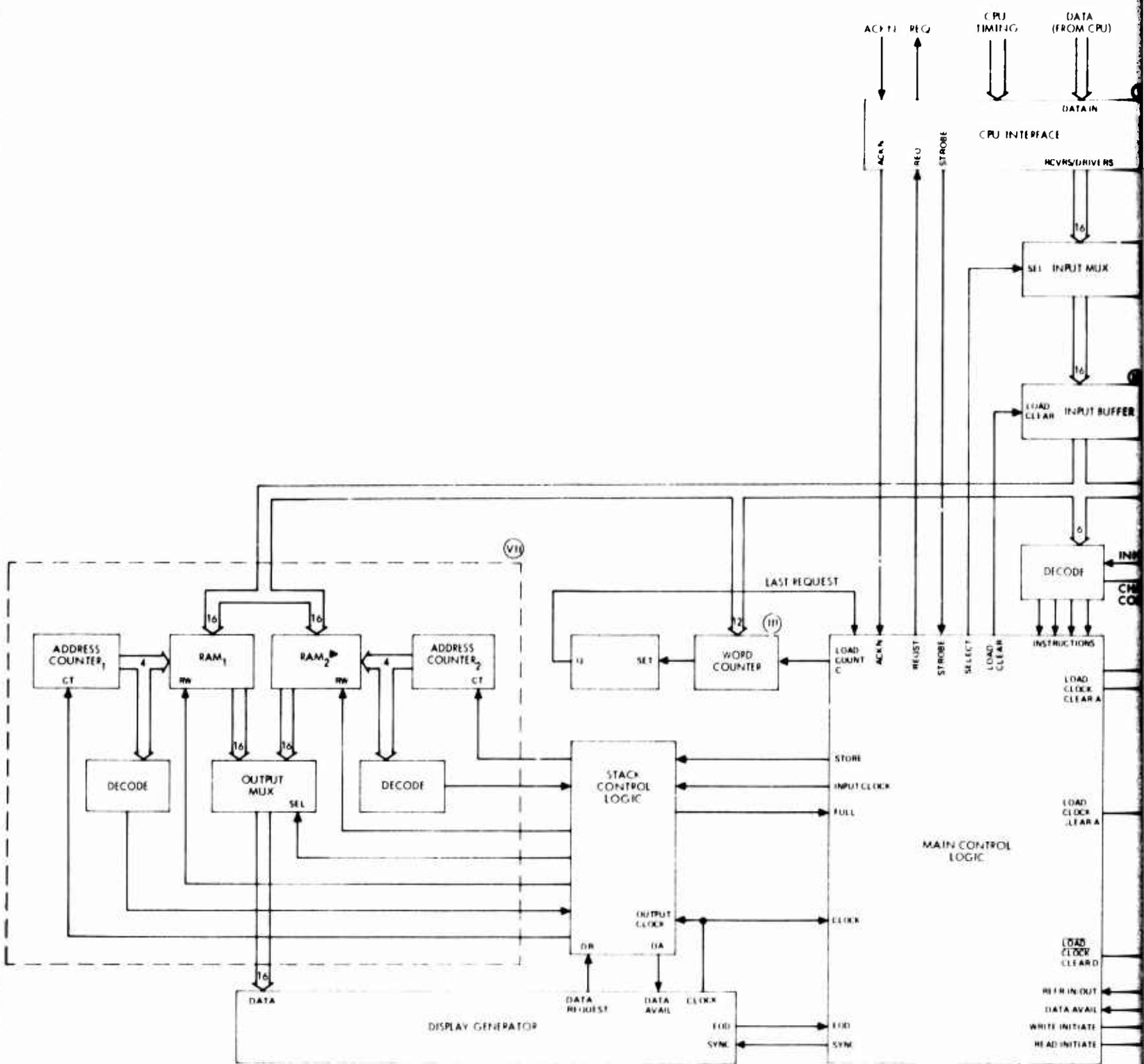
After each frame is displayed, the interface requests new data from the CPU so that the display list may be updated in part, or a complete display list transferred to the interface and stored in the refresh memory. If the display list is to remain unchanged, the CPU should ignore the cycle request. Refresh memory addressing is under the control of the CPU and several branching instructions are available.

Operation Without a Local Refresh Memory

Data from the CPU enters the interface through a line receiver/driver stage designed specifically for the computer in use (Block I of Figure 2-5). This portion of the interface also derives a data input strobe from the computer timing signals and, if necessary, modifies the existing "cycle request" and "Request Acknowledge" signals so as to be compatible with the computer and interface.

When a cycle request has been made and acknowledge, the data word is strobed into an input register (Block II). The instruction code (Bits 00 - 05) of the data word are then decoded and, approximately 200 nanoseconds after the data word enters the register one of the following operations occurs:

Instruction Code	Operation
1011 "No. Words-2"	This code indicates that the input register contains the number of remaining words in the data list minus 2. Bits 04 - 15 of the data word are loaded into the word counter (Block III). This counter is then decremented each time a data word enters the input register. When the counter reaches the zero state, cycle requests are terminated.
10010X Chain-Absolute Direction	This code indicates that the data words to follow contain chain data with absolute direction encoding. Since the chain data could be


ACK IN
REQ
CPU TIMING
DATA (FROM CPU)
DATA IN
CPU INTERFACE
ACK IN
REQ
STROBE
RCVRS/DRIVERS
16
SEL INPUT MUX
16
LOAD CLEAR
INPUT BUFFER
6
DECODE
INSTRUCTIONS
LAST REQUEST
12
WORD COUNTER
LOAD COUNT C
ACK IN
REQUST
STROBE
SELECT
LOAD CLEAR
LOAD CLOCK CLEAR A
ADDRESS COUNTER1
CT
4
RAM1
RW
RAM2
RW
4
ADDRESS COUNTER2
CT
16
16
16
16
DECODE
OUTPUT MUX
SEL
DECODE
STACK CONTROL LOGIC
STORE
INPUT CLOCK
FULL
LOAD CLOCK CLEAR A
MAIN CONTROL LOGIC
OUTPUT CLOCK
CLOCK
DR
DA
LOAD CLOCK CLEAR D
16
DATA
DATA REQUEST
DATA AVAIL
CLOCK
REFR IN/OUT
DATA AVAIL
WRITE INITIATE
READ INITIATE
DISPLAY GENERATOR
EOD
SYNC
EOD
SYNC

A

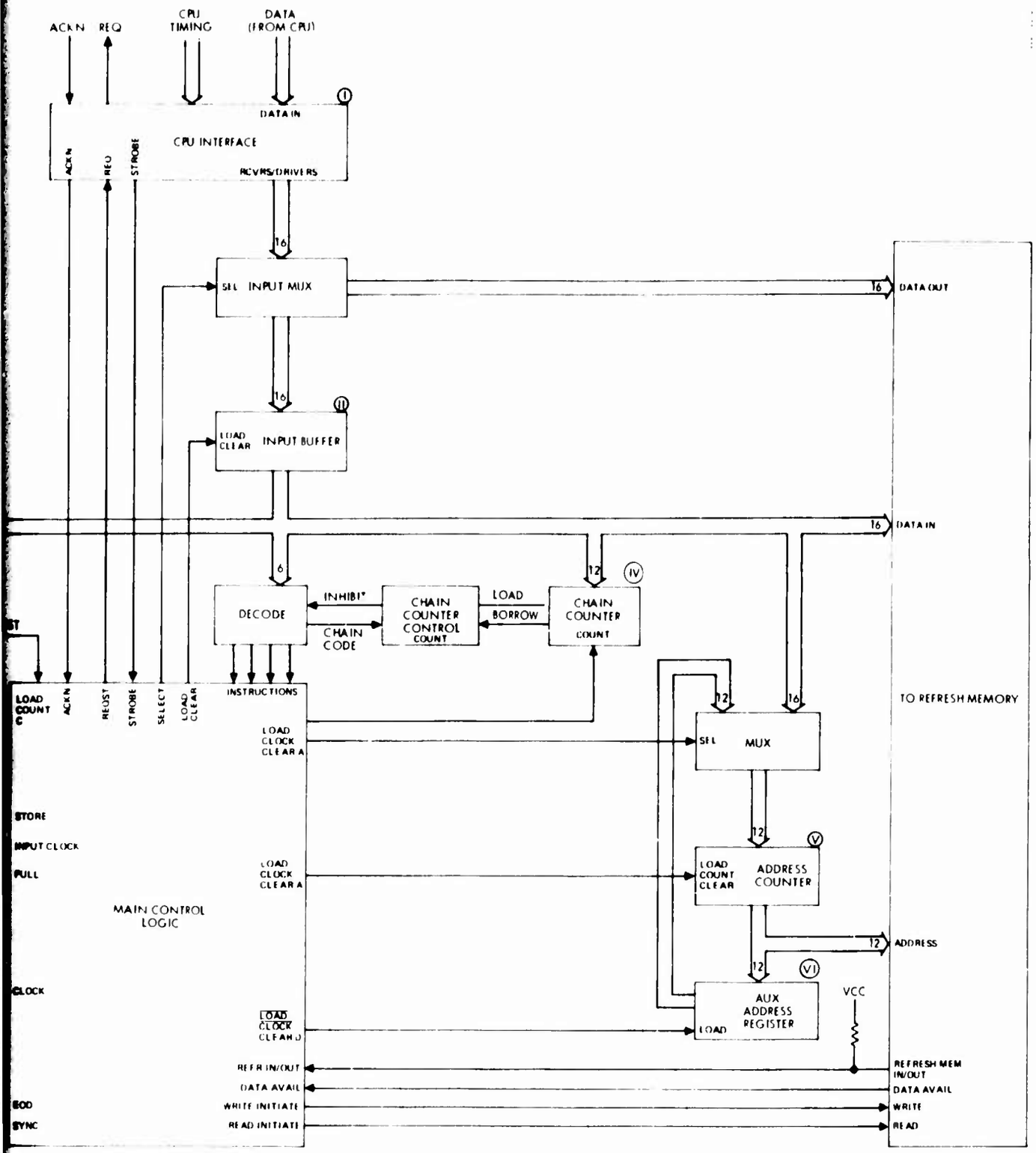


Figure 2-5. Computer/display generator interface.

B

Instruction Code	Operation
	erroneously interpreted as instruction code bits, the decoders must be disabled. Bits 06 to 15, containing the number of segments - 1, are looked at by skewing 2 bits (:4). The skewed data, which now represents the number of absolute data words to follow, is loaded into a counter (Block IV), and the decoder circuits are inhibited. The counter is decremented each time a new data word enters the input register and the decoders are enabled when the counter passes out of the zero state.
1001 1X Chain-Rel. Direction	This code indicates that the data words to follow contain chain data with relative direction encoding. Bits 06 to 15 are loaded into the chain counter and operation proceeds as explained above.
1111 1X "End of Display"	This code is used as the last data word in the data list and indicates that the entire list has been received.
All Codes Except "No. Words-2"	The absence of a "No. Words-2" code indicates that the input register contains a data word to be passed on to the main logic. The data word is then stored in the output buffer (Block VII) and is available to the main logic when the buffer is full or the last data word has been received from the CPU as indicated by the "END OF DISPLAY" code.

Operation with a Local Refresh Memory

Data from the CPU enters the interface and is strobed into the input register (Block II) as explained above. Bits 00 to 05 are decoded and approximately 200 ns after the data word is located into the input register one of the following operations occurs:

Instruction Code	Operation
1011 "No Words-2"	Same as without Refresh Memory
1001 0X "Chain-Abs"	Same as without Refresh Memory
1001 1X "Chain-Rel"	Same as without Refresh Memory

Instruction Code	Operation
1100 "Load Into"	This instruction modifies the contents of the address counter (Block V). Bits 04 to 15 are loaded into the counter.
All codes except "No. Words-2" and "Load Into"	The absence of a "Load Into" or "No. words" instruction indicates that the input register contains a data word to be stored in the refresh memory. A "Write initiated" pulse is then generated and the address counter is incremented.

When a "DISPLAY SYNC" pulse occurs the data list is retrieved from the refresh memory at either the maximum cycle rate of the particular memory used or the main logic clock frequency, whichever is less. The timing signal (Clock') used to control access to the memory is derived from the main logic clock. A "Read initiate" pulse is generated by the interface and a "data available" pulse from the refresh memory strobes the data word into the input register (BLOCK II), and the address counter is incremented.

The Bits 00 to 05 are then decoded and approximately 200 ns after the data word enters the input register one of the following occurs:

Instruction Code	Operation
1001 0X "Chain-Abs"	Same as Before
1001 1X "Chain-Rel"	Same as Before
1101 "Branch"	Bits 04 to 15 of the data word are loaded into the address counter (BLOCK IV). The next data word read from the refresh memory will be from this address.
1110 "MARK PLACE AND BRANCH"	The contents of the address counter (BLOCK V) are loaded in the auxiliary address register (BLOCK VI). On the next clock pulse bits 04 to 15 of the input register are loaded into the address counter. One memory cycle is skipped to allow the address counter to settle.
1111 0X "Return"	The contents of the auxiliary address register (BLOCK VI) are loaded into the address counter.
1111 1X "END OF DISPLAY"	Same as before.

Instruction Code	Operation
All codes except "BRANCH," "MARK PLACE AND BRANCH," "RETURN"	The absence of a branching instruction indicates that the input register contains a data word to be passed on to the display generator. The data word is then stored in the output buffer as explained previously.

Interface Timing Signals

Without Refresh Memory

Signal Name	Description
Strobe	A 50 ns pulse derived from the CPU timing signals. It is used to load data words from the CPU into the input buffer.
Delayed Strobe	A 50 ns pulse delayed approximately 200 ns from STROBE. It is used to transfer data words from the input buffer to the output buffer, word counter, or chain counter.
Display Sync	A short pulse which marks the beginning of the display period.
EOD	A short pulse generated when the main logic decodes the last word in the data list.
R	A flip-flop which indicates the current status of the interface. R is low when a display cycle is in progress and goes high when the frame is completed.
$clock_0$	The main logic clock. Data words are read from the output buffer on the low to high transition of $clock_0$.

With Refresh Memory - Refresh Cycle

Signal Name	Description
Strobe	Same as above.
Strobe delayed	A 50 ns pulse delayed approximately 200 ns from STROBE. It is used to load data words into the word counter,

Signal Name	Description
	address counter, or chain counter, or to initiate a write cycle in the refresh memory.
SYNC	Same as above.
EOD	Same as above.
R	A flip-flop which indicates the current status of the interface. R is high (1) during a refresh cycle.

With Refresh Memory - Display Cycle

Signal Name	Description
Clock'	A 50 ns pulse derived from the main logic clock. Its frequency is equal to the maximum read cycle rate imposed on the refresh memory, and it is used to initiate read cycles.
Data Available (DA)	A short pulse generated by the refresh memory. It is used to strobe data words from the refresh memory output into the input buffer.
Delayed DA	A 50 ns pulse delayed approximately 200 ns from DA. It is used to transfer data words from the input buffer into the chain counter, address counter, auxiliary address register, or output buffer.
R	Same as above. R is low (0) while the display cycle is in progress.

Output Buffer Organization and Control

The output buffer consists of two 16 word x 16 bit bipolar random access memories (RAM_1 and RAM_2), two address counters, an output multiplexer, and associated control logic.

The inputs and outputs to the buffer are as follows:

INPUTS:	16 data lines - from input buffer
	input clock - from interface control logic for write operation

"Store" - indicates that the input buffer contains a data word to be written into the buffer.

Output clock - from main logic for read operations

OUTPUTS:

data request - for read operation, indicates that the main logic is ready for a new data word

16 data lines - to display generator

"full" - indicates that the buffer is full and cannot accept data

data available - indicates to the main logic that data is available at the buffer output.

Data is written into RAM_1, while the output of RAM_2 is available to the display generator. When RAM_2 is empty, data is written into RAM_2 while the main logic reads from RAM_1.

Data flow into and out of the buffer continues alternating in this fashion until the last word in the data list is read from the buffer.

Main Logic

A functional diagram for the main logic section of the display generator is shown in Figure 2-6. The display list from the CPU passes through the interface and enters register R. Setup instructions load the chain generator with parameters Hoo, Voo, Ho, Vo, sin θ and cos θ. The control section operates the chain generator under direction of chain data from the R register. The analog interfaces convert the digital x y z information to analog x y z information for the x y z monitor.

The control unit is comprised of: the state register unit, which contains the state memories; the executive control unit, which provides control signals to all units; and a code converter and control unit which converts the relative code to absolute and the absolute code to control signals for the chain generator. The clock generator is a subunit of the control unit. A more detailed description of the functional units follows.

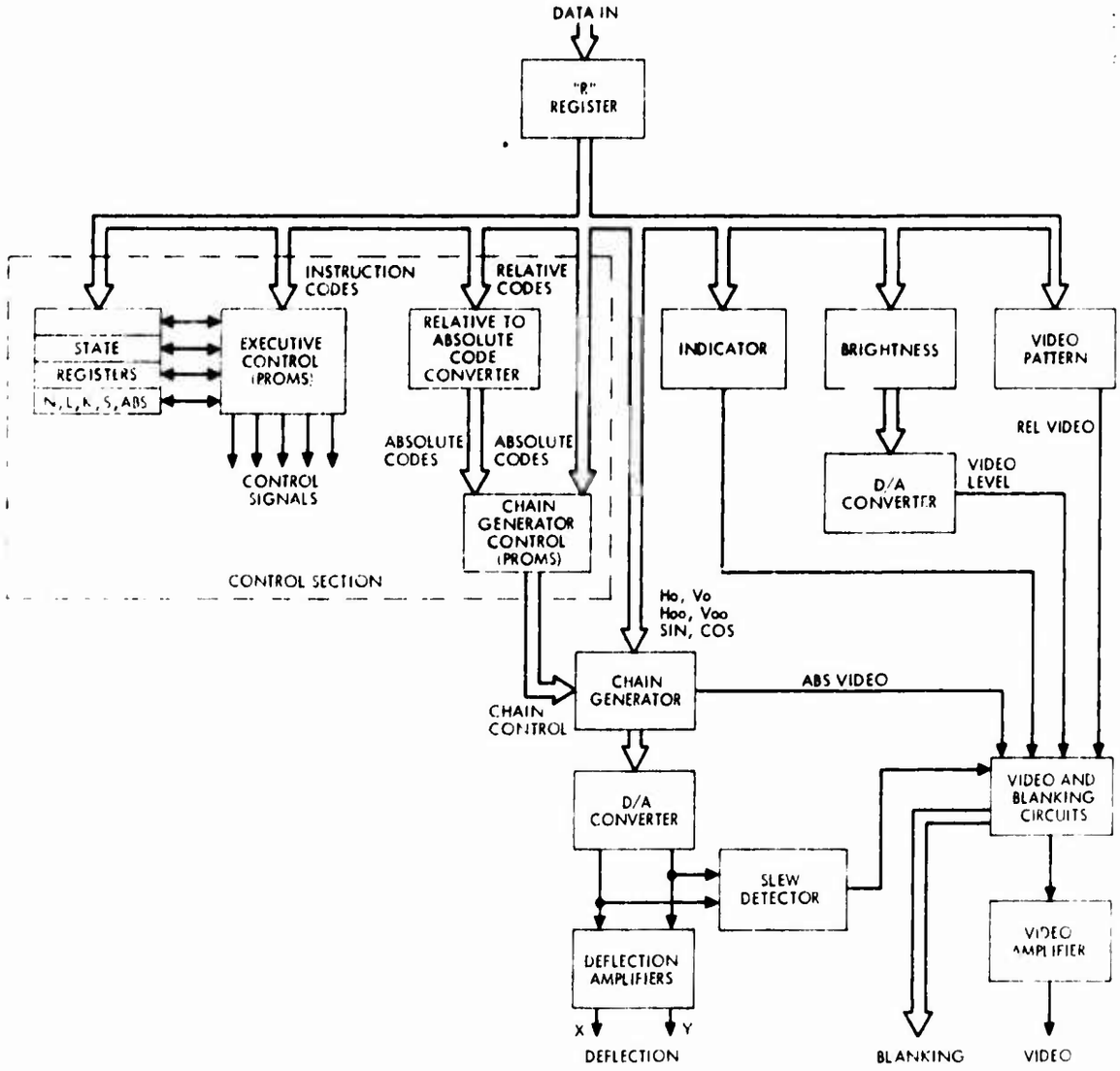


Figure 2-6. Display generator functional diagram.

...

...gister. The R register is a 16 bit buffer for incoming data, and all data ...structions from the interface unit pass through this register.

The data in the register is distributed to the chain generator and control units.

Chain Generator. The heart of the display generator consists of two accumulators which contain the horizontal and vertical position of the beam on the CRT. The remainder of the system provides data and control signals for these accumulators. At the start of a chain, the accumulators contain a position P and after one clock time contain the sum $(P + P_1)$, where P_1 is an incremental change in position. To enter a new position Po, we clear the accumulators and add Po. To generate a chain (or straight line segment) we add P_1, P_2, P_3 ..... at successive clock times. The accumulators then contain Po, $Po + P_1$, $Po + P_1 + P_2$, $Po + P_1 + P_2 + P_3$ ..... at the successive clock times. This results in a change in position, ΔP, of the beam on the CRT in the horizontal and vertical directions.

If r represents a change in the position of the beam during one clock period, then the horizontal and vertical components of r can be expressed as

$$x_i = r \cos \alpha$$

$$y_i = r \sin \alpha$$

where α is one of the eight possible chain segment directions $\frac{n\pi}{4}$ and x_i and y_i are restricted to values of +1, -1, and 0.

To rotate the chain by an angle θ, a coordinate transformation is performed on x_i and y_i. The horizontal and vertical components of r then becomes

$$x_i' = r \cos (\alpha + \theta) = r \cos \alpha \cos \theta - r \sin \alpha \sin \theta$$

$$y_i' = r \sin (\alpha + \theta) = r \sin \alpha \cos \theta + r \cos \alpha \sin \theta$$

or

$$x_i' = x_i \cos\theta - y_i \sin\alpha$$

$$y_i' = y_i \cos\theta + x_i \sin\alpha$$

where x_i' and y_i' represent the change in position of the beam during one clock period. Since the values of x_i and y_i can only be +1, -1, and 0, a chain segment is generated by adding or subtracting sin θ, cos θ or zero to the accumulators at each clock time. Chain data from the computer defines the directions θ and α by means of a series of coded instructions. The chain control circuit decodes the chain data to command the circuit to add or subtract sin θ, cos θ, or zero.

The chain generator is shown in Figure 2-7. The horizontal channel will be used as a descriptive example. The 20 bit accumulator is composed of two arithmetic logic units (ALU), and 2 registers. Only 13 bits of the 20 bits available are used in display processing. The "lsb" accumulator accumulates x_i', and the overflow from this accumulator spills into the "msb" accumulator. Output is the 13 most significant bits. The starting point of a chain is established by clearing both accumulators and adding Ho to the "msb" accumulator. Chain generation is accomplished by accumulating cos θ, -sin θ, or 0 to produce x_i'. The chain control circuit provides logic to decide which operations are to occur at each clock time.

At the output of the chain generator the horizontal bias Hoo is added to the value in the msb accumulator. Thus after k clock periods the horizontal position of the beam is

$$H_k = Hoo + Ho + \sum_{i=1}^{k} x_i'$$

R Register. The R register is a 16 bit buffer for incoming data, and all data and instructions from the interface unit pass through this register.

The data in the register is distributed to the chain generator and control units.

Chain Generator. The heart of the display generator consists of two accumulators which contain the horizontal and vertical position of the beam on the CRT. The remainder of the system provides data and control signals for these accumulators. At the start of a chain, the accumulators contain a position P and after one clock time contain the sum $(P + P_1)$, where P_1 is an incremental change in position. To enter a new position Po, we clear the accumulators and add Po. To generate a chain (or straight line segment) we add P_1, P_2, P_3 ..... at successive clock times. The accumulators then contain Po, $Po + P_1$, $Po + P_1 + P_2$, $Po + P_1 + P_2 + P_3$ ..... at the successive clock times. This results in a change in position, ΔP, of the beam on the CRT in the horizontal and vertical directions.

If r represents a change in the position of the beam during one clock period, then the horizontal and vertical components of r can be expressed as

$$x_i = r \cos \alpha$$

$$y_i = r \sin \alpha$$

where α is one of the eight possible chain segment directions $\frac{n\pi}{4}$ and x_i and y_i are restricted to values of +1, -1, and 0.

To rotate the chain by an angle θ, a coordinate transformation is performed on x_i and y_i. The horizontal and vertical components of r then becomes

$$x_i' = r \cos (\alpha + \theta) = r \cos \alpha \cos \theta - r \sin \alpha \sin \theta$$

$$y_i' = r \sin (\alpha + \theta) = r \sin \alpha \cos \theta + r \cos \alpha \sin \theta$$

or

$$x_i' = x_i \cos\theta - y_i \sin\alpha$$

$$y_i' = y_i \cos\theta + x_i \sin\alpha$$

where x_i' and y_i' represent the change in position of the beam during one clock period. Since the values of x_i and y_i can only be +1, -1, and 0, a chain segment is generated by adding or subtracting sin θ, cos θ or zero to the accumulators at each clock time. Chain data from the computer defines the directions θ and α by means of a series of coded instructions. The chain control circuit decodes the chain data to command the circuit to add or subtract sin θ, cos θ, or zero.

The chain generator is shown in Figure 2-7. The horizontal channel will be used as a descriptive example. The 20 bit accumulator is composed of two arithmetic logic units (ALU), and 2 registers. Only 13 bits of the 20 bits available are used in display processing. The "lsb" accumulator accumulates x_i', and the overflow from this accumulator spills into the "msb" accumulator. Output is the 13 most significant bits. The starting point of a chain is established by clearing both accumulators and adding Ho to the "msb" accumulator. Chain generation is accomplished by accumulating cos θ, -sin θ, or 0 to produce x_i'. The chain control circuit provides logic to decide which operations are to occur at each clock time.

At the output of the chain generator the horizontal bias Hoo is added to the value in the msb accumulator. Thus after k clock periods the horizontal position of the beam is

$$H_k = Hoo + Ho + \sum_{i=1}^{k} x_i'$$

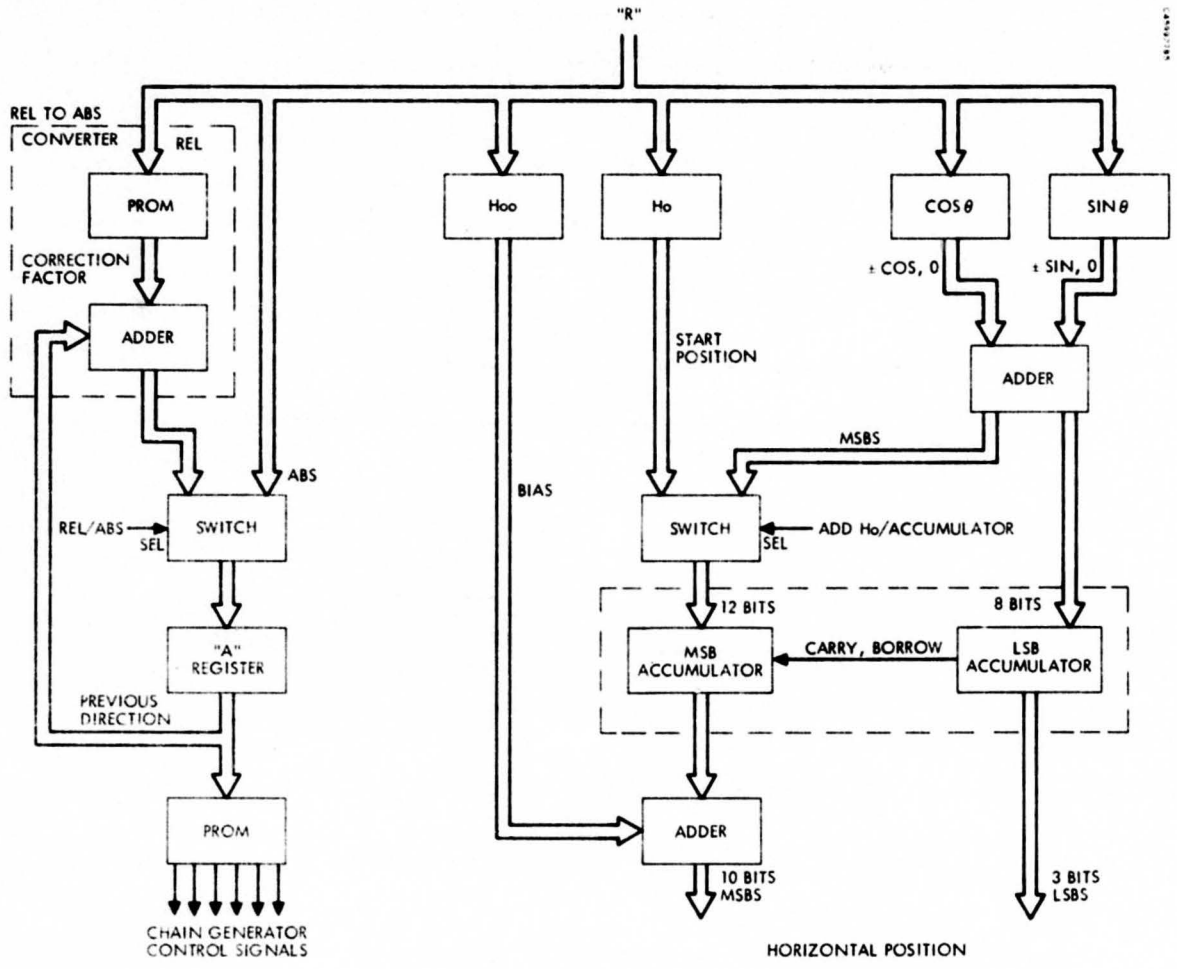


Figure 2-7. Horizontal chain generator and control.

Similarly

$$V_i = Voo + Vo + \sum_{i=1}^{i} y_i'$$

Analog interface. The analog circuit section, as shown in Figure 2-8, contains the horizontal and vertical deflection digital-to-analog (D/A) converters, video D/A converter, slope limiting circuits, and analog line drivers. The deflection drivers can drive up to four display monitors with the addition

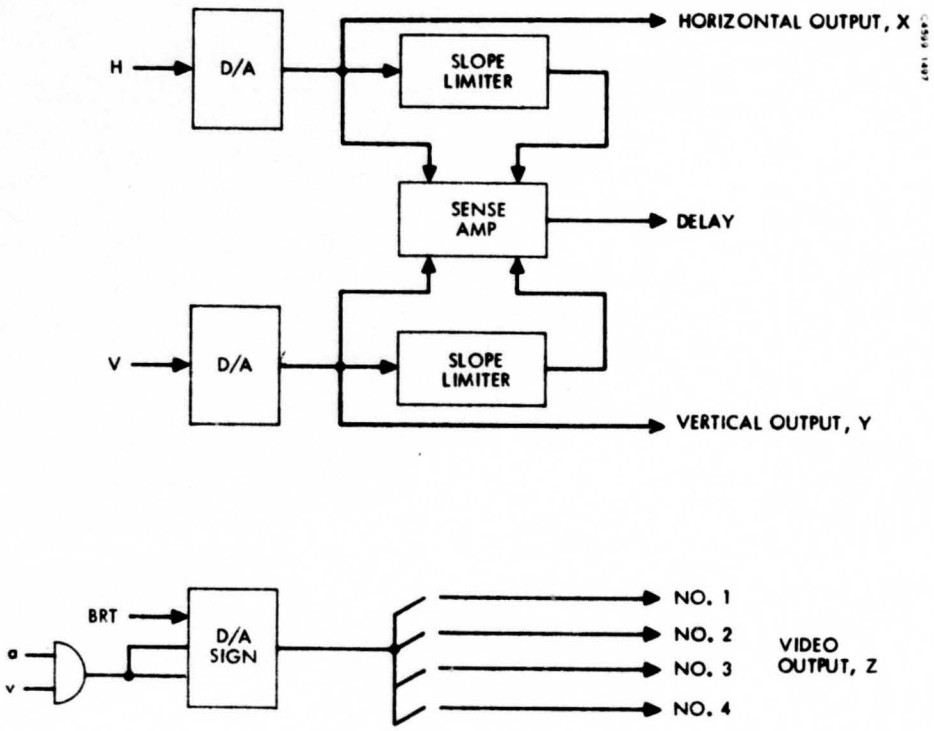


Figure 2-8. Analog interface.

of a display expansion module. The distribution control is contained in the video indicator control word.

A delay signal is generated which is a summation of a fixed delay and a variable delay which is generated when the input and output of the slope limiter differ by a fixed value. This summed delay signal stops the main logic clock if in a chain mode, blanks the display, and forces the display generator to wait until the beam settles.

Code Converter. The code converter illustrated in Figure 2-9, provides logical control signals to direct the chain generator so that proper values of x' and y' are supplied to the horizontal and vertical accumulators.

In the absolute chain mode the data switch selects the first 4 bits of chain data to be loaded into the A register. The output of the A register addresses a field-programmable read only memory (prom) which provides signal outputs that control the chain generators. After each segment is drawn the input data is shifted 4 places to the left.

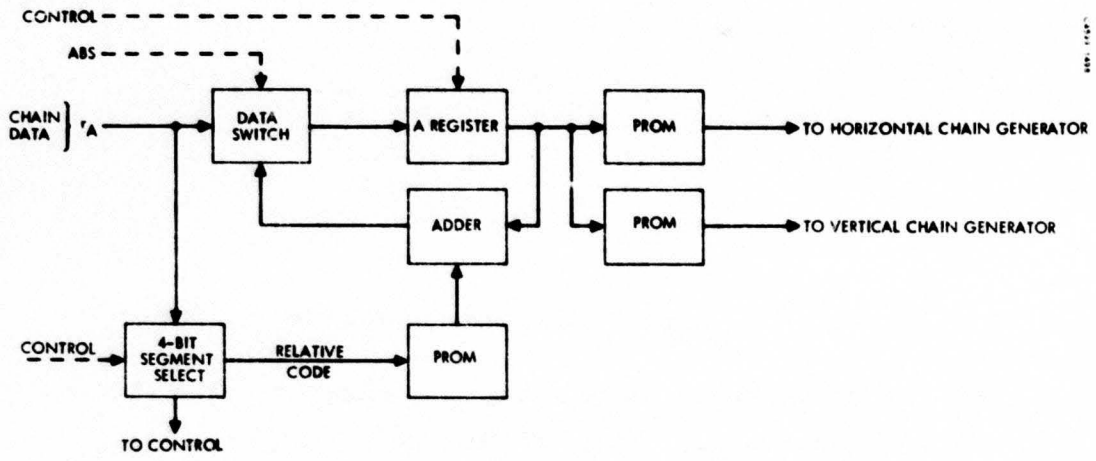


Figure 2-9. Code converter.

In the relative mode the code converter must convert the relative direction code to an absolute direction code, so that register A will always contain the absolute direction code. Control logic first selects the proper 2 or 4 bits representing the relative code, which addresses a PROM to produce a relative-to-absolute correction factor. The next direction is dependent on the present direction, and the present direction is stored in A. The contents of A added with the correction factor determines the next direction, which is loaded into the A register.

State Registers and Executive Control. The state registers and executive control units (Figure 2-10) together with the first 5 bits of the incoming data perform executive control of all the functional blocks shown in Figure 2-7. The state registers contain counters, flip flops, and control registers which define the status of operation at every step. The instruction code (first 5 bits of instruction data) together with the memory states from the state registers define 63 Q states, each Q state performing a specified set of control operations. Conversion of instruction codes and memory states to the Q states is performed in one PROM, while conversion of the Q states to control outputs is performed in another PROM. The use of PROMs for microprogram control of the display generator has resulted in a

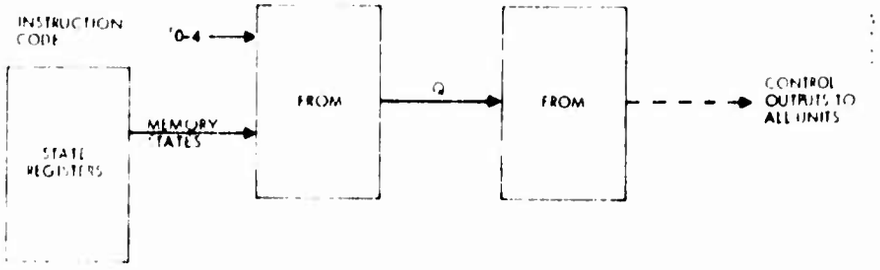


Figure 2-10. State registers and executive control.

considerable reduction in hardware and a simplification of the checkout procedures in the fabrication of the system.

Digital Overrange and Numbering System. The display is 1024 x 1024 bits. There is a 2:1 digital overrange which is not displayed but which allows a chain to go beyond the range of the display and return into view. This is the display range illustrated in Figure 2-10. The analog display range is the center square and the digital range is the larger square. The analog range is represented by the ten least significant bits. The numbering system uses a two's complement code. When V = 0, the code is all 0's. When V = -1 the code is all 1's. Figure 2-11 also shows the code for V = 511 and V = -512.

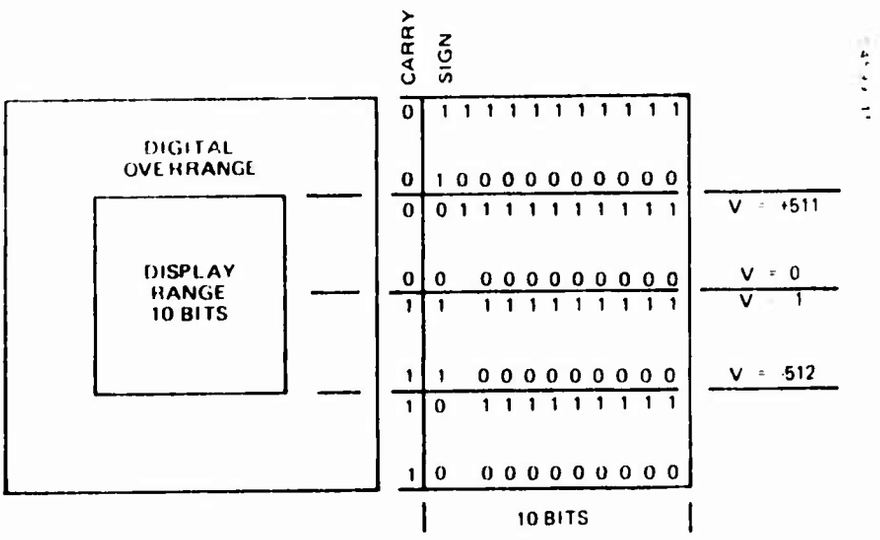


Figure 2-11. Display generator numbering system.

Mechanical Design

The display generator, figures 2-12 and 2-13 is contained in an ATR Size 1 enclosure which is mounted in a 10-1/2 H x 19 inch W x 20 inch D outer enclosure for installation in a 19 inch equipment rack. Drawer slides are mounted on the sides for easy access to the display generator.

The logic section consists of two 8 x 16 inch aluminum frames which support the wire wrap sockets and integrated circuits. The two planes are hinged on one edge and can be lifted up to expose the deflection D/A converters, power supplies, and cooling fan. The upper plane contains computer interface logic and analog circuits, and the lower plane contains the main symbol generation logic. Connections between the two planes is made by 14 pin IC socket connectors and twisted pair cables.

The outer enclosure contains the DMA buss connectors for the Raytheon 704 computer, and the remote power control and output connectors for the Kratos deflection amplifiers and CRT.

EMD DISPLAY

The selected CRT display indicator and associated electronics is capable of working in conjunction with the custom designed display generator such that it will not limit the flexibility, performance or appearance that the display generator is capable of providing. In addition the remote indicator unit is of a physical size which permits its installation and use in a Link GAT-2 or GAT-3 simulator.

A Kratos display, model RM307, was selected for the EMD display. It is a high performance, low cost XYZ graphic unit. A high quality stator deflection yoke is used with Mu metal shielding on the CRT to insure deflection accuracy. Excellent constrast, spot size and readability allow the unit to be used for display of random point plots, vectors and alphanumerics. The 2 MHz small signal bandwidth assures faithful reproduction of the commanded display.

The display is provided in two units. The cockpit installed unit consists of a CRT, deflection yoke, and high voltage power supply. The amplifier assembly containing the deflection amplifiers, video amplifier, and low voltage power supplier will be remotely located from the cockpit.

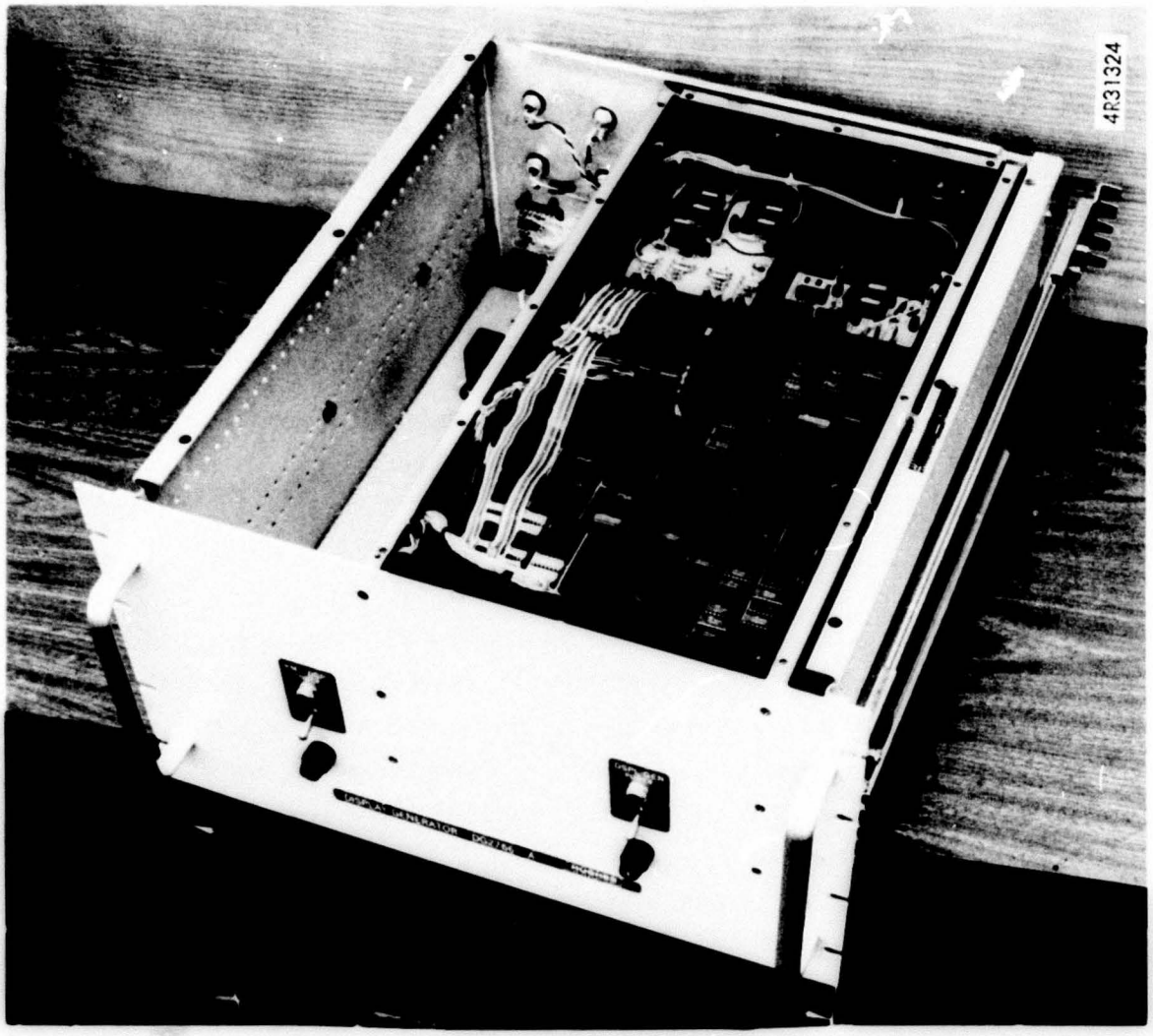


Figure 2-12. Display generator.

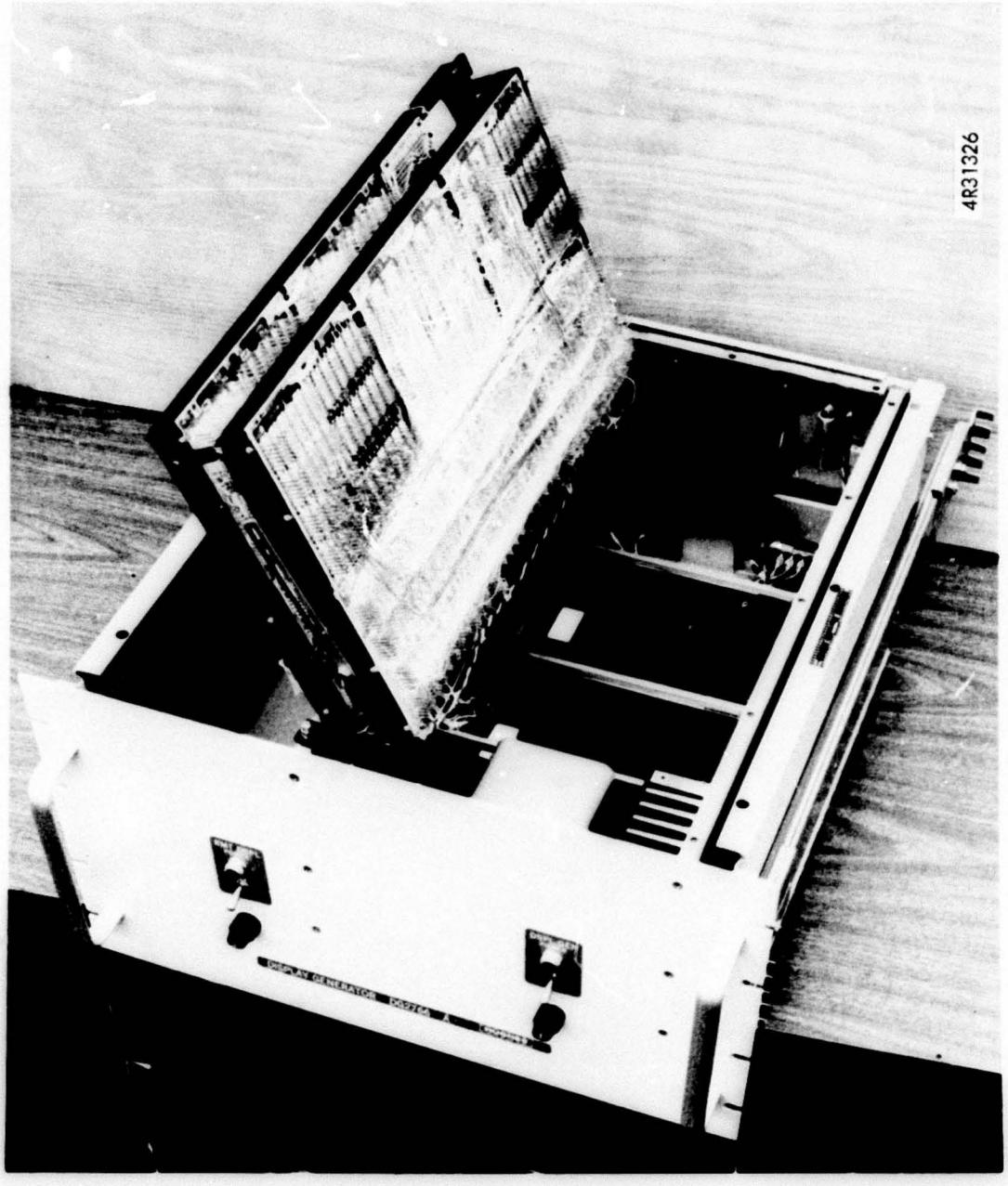


Figure 2-13. Display generator.

The KRATOS unit was selected because of its high performance, low price, compatible delivery schedule, and its proven reliability as exhibited by its wide usage as a computer readout graphic monitor. Table 2-1 is the display indicator performance specification.

TABLE 2-1. TABLE OF DISPLAY INDICATOR PERFORMANCE SPECIFICATIONS

SPOT SIZE (Note 1)	
Center of CRT	0.015 inch
Within Quality Circle	0.022 inch
SETTLING TIME (to 0.25 percent anywhere within quality circle)	12 μseconds
SMALL-SIGNAL BANDWIDTH	2 MHz
USEFUL DIAMETER (quality circle)	6 inches
TUBE SIZE	7 inches
FACEPLATE	flat
DRIFT (Note 2)	<0.1 inch per 24 hours
TEMPERATURE STABILITY (Note 3)	0.05 percent/°C max
LINEARITY (Note 2)	±2.0 percent
REPEATABILITY (Note 2)	±0.15 percent
CONTROLS (Front Panel)	
Brightness	50 ft < at 100,000 inches/sec
Contrast	4:1
DEFLECTION INPUT	
Amplitude (full scale deflection)	±5.0 volts
Impedance	1,000 ohms
UNBLANKING INPUT	
Unblank Level	-3 ±0.5 volts
Blank Level	0 ±0.5 volts
Impedance	75 ohms
AC POWER INPUT	
Voltage	117 volts
Frequency	60 Hz
DIMENSIONS	
CRT Assembly	8W x 8H x 20D inches
Amplifier Assembly	19W x 8-3/4H x 14D inches

Note 1: Measured by the two-slit analyzer method at the 50% point.

Note 2: Maximum deviation from ideal location anywhere within useful area referenced to full useful diameter.

3.0 EMD SYSTEM SOFTWARE

INTRODUCTION AND SUMMARY

The extreme flexibility of the display generation hardware makes it a highly useful display generation device for a wide variety of cockpit displays. In particular, several hardware features were designed to facilitate generation of electronic map-type displays. An important task of this program was the development of software specifically for the Raytheon 704 computer at the University of Illinois to permit generation of electronic map displays (EMDs).

Figure 3-1 shows the program and data flow for a generalized EMD software system. It includes the formatting and storage of the bulk map data base on magnetic tape by the Map Data Pre-Processor and a number of processing functions for real time EMD generation. These include bulk map data retrieval, category processing and selection and display formatting, formation of the display list through the use of display utility routines, interface with the simulator controls and the aerodynamic simulation, and overall program timing and control. The software developed under the present contract performs all of the above functions except the last two (simulator interface and overall timing and control). The following sections describe each of the four software packages in some detail:

1. Display Utility Routines
2. Map Data Pre-Processor (MDPP)
3. Data Retrieval (DATARET)
4. Category Processor (CATPROC)

Detailed flow charts and program symbol definitions are included in Appendix A for reference.

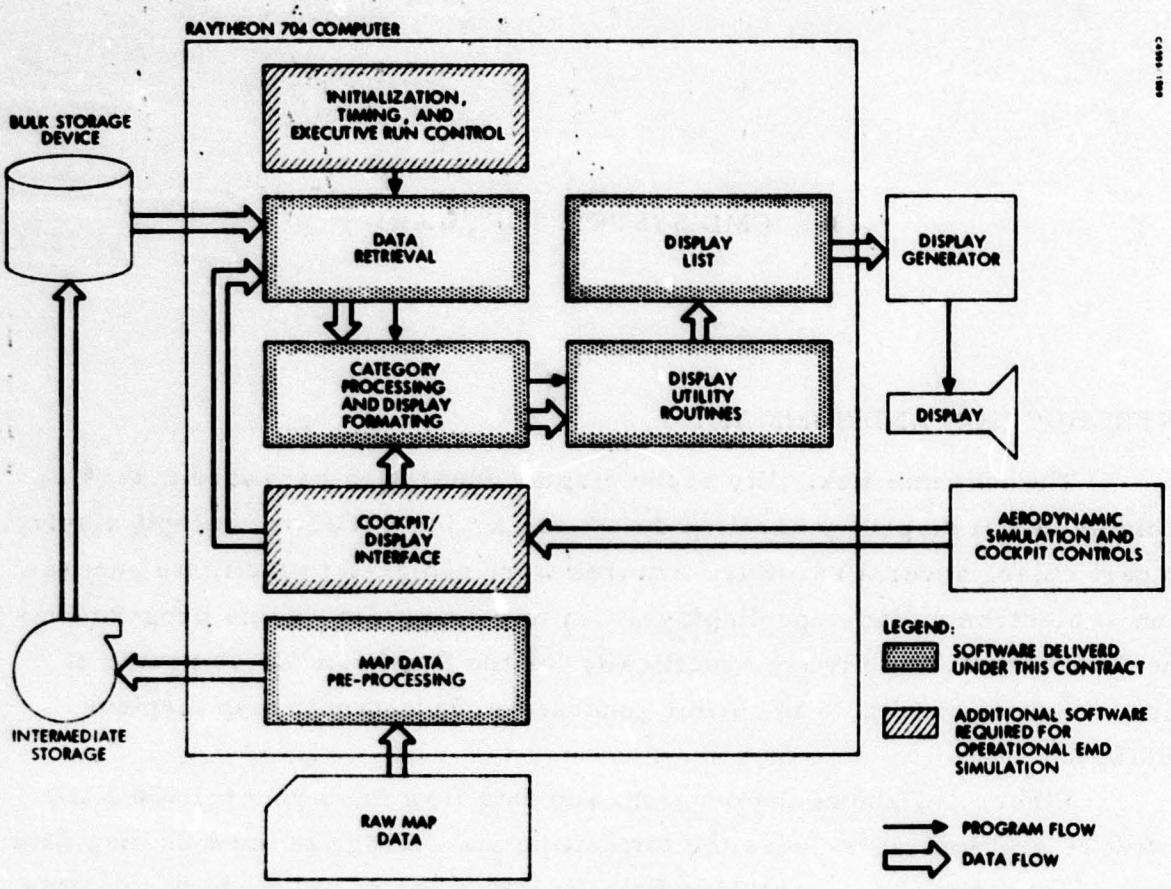


Figure 3-1. Electronic map display software system block diagram.

DISPLAY UTILITY ROUTINES

The display utility routines provide a means of converting desired display objectives, e.g., vectors, irregular shapes, alphanumerics, etc., into hardware instructions meaningful to the display generator. For instance, if the user wishes to draw a vector he may call one of the routines, specifying length and direction, and the routine stores the appropriate instructions into the display list.

There are three general categories of display utility routine:

1. Control
2. Refresh
3. Display Definition

The two routines used to control the formation of the display list are DINIT and DDSET. DINIT is used to initialize the display list parameters and to specify the memory area in which the list is to be constructed. DDSET is used to indicate that the display list is complete and to close it.

The routine, DPLAY, is used to refresh the display. It causes the display list to be output to the display via the direct memory access (DMA) channel. It should be executed at the desired refresh rate (60 Hz for a flicker-free display).

All of the other display utility routines are used to define the actual graphic contents of the display. They cause display generator control words and data to be stored into the display list defined or initiated by a call to DINIT. When the new display has been completely specified DDSET is used to cause DPLAY to refresh from the newly defined display list.

Figure 3-2 is an example of how the display utility routines are used by the electronic map display software to generate real time displays.

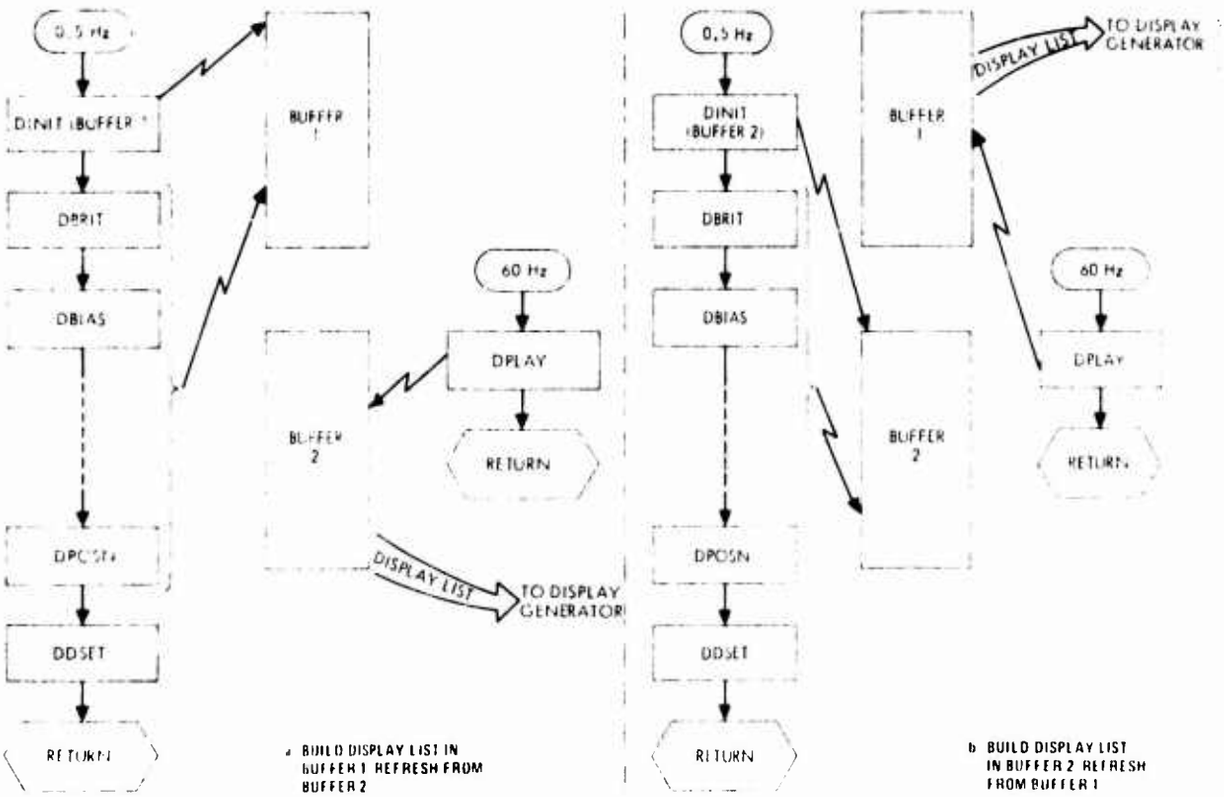


Figure 3-2. Display list initialization, construction, and output.

Alternating buffers are used to permit refresh of the display from one buffer while a new (updated) display list is being constructed in the other. Figure 3-2a shows DPLAY being executed at a 60 Hz rate, causing the display list in BUFFER 2 to be output to the display generator at that rate. At the same time, DINIT is used to initialize BUFFER 1 and other display utility routines are used to construct a new display list in that buffer area. As soon as the new display list is complete DDSET is called. When this occurs, the situation changes to that shown in Figure 3-2b. DPLAY immediately begins refreshing the display with the updated display list in BUFFER 1. BUFFER 2 is now available for reinitialization and formation of the updated display list as soon as the next 2-second update cycle begins.

There are twenty-one display utility routines to provide the user with maximum flexibility and efficiency. Each of the routines may be called from either Fortran or SYM II programs. If they are called from Fortran or if any arguments are defined in a Fortran program, the following points should be noted. The Raytheon 704 computer allocates two 16-bit words for each integer variable even though only one is actually used (the most significant 16-bits are used). Therefore, in an integer array every other 16-bit word is unused unless the array is declared double integer. The display utility routines assume there are no embedded unused words in arrays used as arguments, and all such arrays must be declared double integer.

Flow charts and symbol definitions of the display routines summarised in Table 3-1 are contained in Appendix A. The functional description and calling sequence of each of the routines is provided in the following paragraphs.

DINIT

The routine "DINIT" initializes the display list. That is to say it determines where the display list is to be stored in memory and reserves the first word of the list for the number of words (which will be supplied when the list is complete).

"DINIT" sets the pointer to the start of the display list (LSSTT) equal to the first location of the display buffer specified in the argument. The pointer to the next available list location (LSNXT) is initialized to (LSSTT)+1.

TABLE 3-1. DISPLAY UTILITY ROUTINES

Routine Name	Arguments	Scale	Function
DBIAS	IX, IY	Display Units Right Justified	Load H_{OO} and V_{OO} registers to bias display
DBRIT	IC, IS, IB	Right Justified	Specifies color, saturation and brightness of display
DPOSN	IX, IY	Display Units Right Justified	Moves beam to (IX, IY) on display
DPOSR	IX, IY	Display Units Right Justified	Moves beam to (IX, IY) and resets beam there after an operation
DPOSA	IX, IY	Display Units Right Justified	Moves beam from present position specified amount
DROTA	ISIN, ICOS	Q7	Specified rotation angle
DVPAT	IPAT	Right Justified	Specifies video pattern
DSGLN	Lngth	Display Units Right Justified	Specifies segment length for chains
DLINE	Lngth	Display Units Right Justified	Command to draw a line
DDDOT	–	–	Command to draw a dot
DABCH	NSEG, Array	Right Justified	Defines absolute chain and chain data
DRLCH	NSEG, VIDPAT, Array	Right Justified	Defines a relative chain with video pattern and data

(Continued next page)

(Table 3-1, concluded)

Routine Name	Arguments	Scale	Function
DINIT	Buffer	Right Justified	Initializes display list and reserves a word for 'Define List' word
DDSET	–	–	Set up control words for display of current buffer
DPLAY	–	–	Sets up and starts symbol generator
DWHER	Loc	Right Justified	Returns to "LOC" index of next item in list
DINST	I	Right Justified	Sets display list index to "I" (Used for inserting into display list)
DVECT	Lngth, ISIN, ICOS	Display Units (INS and COS Q7)	Draws vector of length, lngth, at specified angle
DVCTB	Lngth, ISIN, ICOS, LBLNK	Display Units (ISIN and COS Q7)	Draws vector as above except with first segment blanked
DSYMB	N, ADDR	Right Justified	Adds pre-defined symbol starting at address 'ADDR' of N words to display list
DRSET	–	–	Restores display list index after call to DINST

"DINIT" like all the display utility routines may be called from either a FORTRAN program or a SYM II program for maximum versatility. The FORTRAN calling sequence is as follows:

CALL DINIT (IBUF)

where IBUF is the array in which the display list will be stored. IBUF is an integer array and should be declared double integer if a print out is desired in the FORTRAN program.

The SYM II calling sequence for "DINIT" is as follows:

```
SGM
LDX - DINIT
JSX * 6
DATA  IBUF
```

where IBUF is the beginning of a block of memory cells large enough to contain the display list.

DDSET

The routine "DDSET" finalizes the display list since it furnishes an end-of-display instruction at the end of the list and stores the number of words in the list into the first word of the display list. "DDSET" is therefore the last routine called when building a display list.

DDSET performs some minor bookkeeping on the display list before setting up the list length and starting location (DWDCT and STTADD) for use by the routine DPLAY. The bookkeeping functions include: 1) setting the bias and position registers (Hoo, Voo, Ho, and Vo) to zero to center the beam on the display; 2) ensuring that the display list is not of a length which is an even multiple of 16; and 3) storing the length of the list minus three into the "define list" command at the start of the display list.

The FORTRAN calling sequence is

```
CALL DDSET
```

where DDSET has no arguments.

The SYM II calling sequence for "DDSET" is

```
SGM
LDX - DDSET
JSX * 6
```

DPLAY

The routine "DPLAY" outputs the display list to the display generator by connecting the DMA controller and outputting the starting address (STTADD) and number of words (DWDCT) to the controller. "DPLAY" should be called repetitively at a rate which will result in a flicker-free display.

The Fortran calling sequence is as follows:

```
CALL DPLAY
```

"DPLAY" has no arguments.

The SYM II calling sequence for "DPLAY" is

```
SGM
LDX = DPLAY
JSX * 6
```

DBRIT

The display routine "DBRIT" adds a word to the display list which specifies color, saturation and brightness of the display. Color and saturation are presently unused and are provided for future capability only. However, eight brightness levels are available to the user.

DBRIT simply packs the color, saturation, and brightness specified in the argument list into a video intensity control word. The control word is then stored into the display list.

The Fortran calling sequence for "DBRIT" is

```
CALL DBRIT (IC, IS, IB)
```

where IC — color selection from 0-7
IS — saturation level from 0-15
IB — brightness level from 0-7

The SYM II calling sequence for "DBRIT" is as follows:

```
SGM
LDX = DBRIT
JSX * 6
DATA IC
DATA IS
DATA IB
```

where IC, IS and IB are as above.

DBIAS

The routine "DBIAS" supplies a horizontal and vertical bias or offset to the entire display. Calling the routine will add two words to the display list – one for horizontal bias and one for vertical bias.

The Fortran calling sequence for "DBIAS" is

```
CALL DBIAS (IX, IY)
```

where IX and IY are integers ranging from -512 to +511 and IX is the horizontal bias and IY is the vertical bias.

"DBIAS" is called from a SYM II program with

```
SGM
LDX = DBIAS
JSX * 6
DATA IX
DATA IY
```

where IX and IY are as above.

DPOSN

The routine "DPOSN" stores two instructions into the display list to position the beam in specified horizontal and vertical positions. When the beam is moved it will not be reset to those positions except by repeating the call to "DPOSN" with the same arguments. Thus DPOSN would be used whenever the next display item to be drawn is located at the end of the previous item. An example might be in the generation of text where each character or word is located at the end of the preceding one. In this case, new position commands need not be issued for each new character or word symbol.

"DPOSN" is called in Fortran with the following statement:

```
CALL DPOSN (IX, IY)
```

where IS and IY are integers with values between -512 and +511 and are the horizontal and vertical positions, respectively.

The SYM II calling sequence for "DPOSN" is as follows:

```
SGM
LDX = DPOSN
JSX * 6
DATA IX
DATA IY
```

where IX and IY are as in the Fortran call.

DPOSR

The routine "DPOSR" also stores two instructions in the display list to position the beam horizontally and vertically as does "DPOSN". But "DPOSR" sets the horizontal and vertical reset flags to insure that the beam will be reset to the specified position after each operation, e.g., drawing a vector. A call to "DPOSN" will reset those flags.

DPOSR would generally be used wherever symbols or vectors are located at a common display location. The use of this display command obviates the requirement to add display position commands for each symbol or vector in the cluster.

The Fortran calling sequence for "DPOSR" is as follows:

```
CALL DPOSR (IX, IY)
```

where IX and IY are as in the call to "DPOSN".

The SYM II calling sequence for "DPOSR" is

```
SGM
LDX = DPOSR
JSX * 6
DATA IX
DATA IY
```

where IX and IY are as in the Fortran calling sequence.

DPOSA

The routine "DPOSA" adds two words to the display list to offset the beam from its current position on the display. One word contains the horizontal offset while the other contains the vertical offset.

The routine "DPOSA" adds two words to the display list which define a new display position. In this case, however, the arguments define incremental values to be added to the current position of the electron beam on the display. Thus, DPOSA defines a new display position relative to an established position where DPOSN, DPOSR, and OBIAS define positions or biases in absolute display coordinates.

The following statement calls "DPOSA" from a Fortran program:

```
CALL DPOSA (IX, IY)
```

where IX and IY are defined as in the call to "DPOSN".

The following sequence is used to call "DPOSA" in a SYM II program:

```
SGM
LDX - DPOSA
JSX * 6
DATA IX
DATA IY
```

where IX and IY are as in the Fortran call.

DROTA

Two words are added to the display list by "DROTA" which specify the angle of rotation for a given symbol or set of symbols. The symbol is rotated about its starting point. The two words which are added specify the sine and cosine of the angle in cartesian coordinates.

The Fortran call is as follows:

```
CALL DROTA (ISIN, ICOS)
```

where ISIN and ICOS are the sine and cosine, respectively, of the rotation angle scaled at Q7, i.e., binary point is to the right of bit 7. ISIN and ICOS are also signed integers. For example, if the rotation angle is 30 degrees

then ISIN = 128 and ICOS = 221. The values may be calculated and scaled by the following FORTRAN statements:

```
ISIN = SIN(X) * 256.
ICOS = COS(X) * 256.
```

where SIN and COS are the standard FORTRAN floating point trigonometric functions.

The SYM II calling sequence is as follows:

```
SGM
LDX = DROTA
JSX * 6
DATA ISIN
DATA ICOS
```

where ISIN and ICOS are as above.

DVPAT

The routine "DVPAT" stores into the display list a video pattern for use with relative chains. The video pattern specifies which segments of the relative chain to blank and which ones to draw. It is 12 bits long and rotates left by one bit at the end of every segment of the relative chain.

To call "DVPAT" in Fortran the following statement is used:

```
CALL DVPAT (IPAT)
```

where IPAT is the integer value of the video pattern and must not exceed 12 bits in length. For example, IPAT = 2730_{10} (= 5252_8) will set the video pattern to alternating 1's and 0's.

The SYM II calling sequence is:

```
SGM
LDX = DVPAT
JSX = 6
DATA IPAT
```

where IPAT is as above.

DSGLN

The routine "DSGLN" specifies the length of each chain segment for both relative and absolute chains. The length is in display units and must be greater than 1 for both types of chains.

The Fortran calling sequence is as follows:

```
CALL DSGLN (LNGTH)
```

where LNGTH, an integer between 2 and 511, is the segment length.

The SYM II calling sequence is as follows:

```
SGM
LDX = DSGLN
JSX * 6
DATA LNGTH
```

where LNGTH is as above.

DLINE

The routine "DLINE" adds a word to the display list which causes the symbol generator to draw a single line a specified length. The direction is defined by the sine and cosine of the last DROTA call. The length is given in display units.

The Fortran calling sequence for "DLINE" is CALL DLINE (LNGTH) where LNGTH, an integer between 1 and 1023, is the line length.

The SYM II calling sequence is as follows:

```
SGM
LDX   DLINE
JSX * 6
DATA LNGTH
```

where LNGTH is as in the Fortran call.

DDDOT

The routine "DDDOT" places in the display list a command to the display generator to draw a single dot.

The Fortran calling sequence is

```
CALL DDDOT
```

The SYM II calling sequence for "DDDOT" is

```
SGM
LDX = DDDOT
JSX * 6
```

DABCH

The routine "DABCH" stores into the display list a predefined absolute chain. In addition to the actual chain data, "DABCH" will also set up the symbol generator instruction entitled "define chain absolute", which contains the number of 4-bit segments in the chain to follow.

The Fortran calling sequence for "DABCH" is CALL DABCH (NSEG, IARRY) where NSEG is the number of 4-bit segments in the absolute chain definition and IARRY is a double integer array which contains the absolute chain. Each word of IARRY contains 32 bits of information and 8 absolute chain segments since it is a double integer array. Absolute chains are defined in Section 2.0.

The SYM II calling sequence for "DABCH" is

```
SGM
LDX = DABCH
JSX * 6
DATA NSEG
DATA IARRY
```

where NSEG and IARRY are defined as in the Fortran call.

DRLCH

The routine "DRLCH" adds to the display list a predefined relative chain. "DRLCH" also supplies the symbol generator instruction entitled "define chain relative", which contains the number of segments in the relative chain.

"DRLCH" is called in Fortran with the following instruction:

```
CALL DRLCH (NSEG, IARRY)
```

where NSEG is the number of 2-bit segments in the relative chain definition (the 4-bit absolute code is included in the count) and IARRY is a double integer array containing the relative chain. Since IARRY is a double integer array each "word" contains 32-bits of information and 16 chain segments. Relative chain data formats are described in Section 2.0.

The SYM II calling sequence for "DRLCH" is as follows:

```
SGM
LDX = DRLCH
JSX * 6
DATA NSEG
DATA IARRY
```

where NSEG and IARRY are as above.

DWHER

The routine "DWHER" returns the index of the next item in the display list. It can be used to determine the size of a display list. It is primarily used to preserve the index of an item that might be replaced later or of an area of the display list being set aside to store into later in the program. "DWHER" has no effect on the display list itself.

The Fortran calling sequence for "DWHER" is as follows:

```
CALL DWHER (LOC)
```

where LOC is the integer in which the display list index will be returned.

The SYM II calling sequence for "DWHER" is as follows:

```
SGM
LDX = DWHER
JSX * 6
DATA LOC
```

where LOC is as above.

DINST

The routine "DINST" enables the user to alter the contents of the display list by temporarily setting the display list index (LSNXT) to any position in the display list (generally obtained previously from DWHER).

The user is cautioned that he is responsible for not destroying data in the display list following the insertion or alteration. For instance, if he wishes to change five words within the list he must replace them with exactly five words.

The statement for calling "DINST" in Fortran is CALL DINST (INDEX) where INDEX contains the value of the index relative to the beginning of the list where data is to be changed.

The calling sequence for "DINST" in SYM II is

```
SGM
LDX  - DINST
JSX  * 6
DATA INDEX
```

where INDEX is as above.

DRSET

The routine "DRSET" can only be used after a call to "DINST" since "DRSET" restores the display list index (LSNXT) to the value it had before being changed by "DINST". Since "DDSET" uses the value of the index to determine the number of words in the list, "DRSET" must be called after "DINST" to ensure that the proper number of words are output to the display generator.

"DRSET" is called in Fortran with the instruction

```
CALL DRSET
```

"DRSET" is called in SYM II with the sequence

```
SGM
LDX    DRSET
JSX  * 6
```

DVECT

The routine "DVECT" adds the instructions to the display list to draw a vector of specified length and direction. Three words are added to the list - two for rotation and one to draw the line.

The Fortran calling sequence is as follows:

```
CALL DVECT (LNGTH, ISIN, ICOS)
```

where LNGTH is the vector length in absolute display units (1-1023) and ISIN and ICOS are the sine and cosine of the rotation angle scaled at Q7. See DROTA for details of ISIN and ICOS.

The SYM II calling sequence is as follows:

```
SGM
LDX  - DVECT
JSX  * 6
DATA LNGTH
DATA ISIN
DATA ICOS
```

where LNGTH, ISIN, and ICOS are as above.

DVCTB

The routine "DVCTB" adds the instructions to the display list to draw a vector of specified length and direction with the first portion blanked. Five words are added to the display list by "DVCTB" - two to offset the beam to the end of the blanked portion of the vector, two for rotation and one to draw the line.

The Fortran calling sequence for "DVCTB" is as follows:

```
CALL DVCTB (LNGTH, ISIN, ICOS, LNGBLK)
```

where LNGTH, ISIN, ICOS are as in the routine "DVECT" and LNGBLK is the length of the beginning portion of the vector to blank. LNGBLK is in absolute display units (0-1023).

The SYM II calling sequence for "DVCTB" is as follows:

```
SGM
LDX - DVCTB
JSX * 6
DATA LNGTH
DATA ISIN
DATA ICOS
DATA LNGBLK
```

where LNGTH, ISIN, ICOS, and LNGBLK are the same as above.

DSYMB

The routine "DSYMB" adds the contents of a specified array to the display list. Virtually anything can be added to the list by using this routine. However, a measure of caution must be observed when using "DSYMB". For instance, if an absolute chain is to be added to the list using "DSYMB", the first word of the chain must be the "define chain absolute" instruction with the number of 4-bit chain segments - 1 packed into it.

The Fortran calling sequence for "DSYMB" is

```
CALL DSYMB (N, IARRY)
```

where N is the total number of 16-bit words to be stored in the list and IARRY is a double integer array containing the data to be stored. IARRY may contain any number of 16-bit words.

The SYM II calling sequence is as follows:

```
SGM
LDX - DSYMB
JSX * 6
DATA N
DATA IARRY
```

where N and IARRY are as above.

MAP DATA PRE-PROCESSOR (MDPP)

The Map Data Pre-Processor (MDPP) is a stand-alone FORTRAN program designed to create a formatted map data base for storage on the Raytheon 704 disk and retrieval by the real-time Electronic Map Display (EMD) program. Input to the MDPP is on punched cards prepared from raw map data (such as air navigation charts). Output is to a magnetic tape for intermediate storage of the data base until EMD run time.

This section describes the data base, MDPP functions, preparation of input data, and the outputs from the MDPP. A glossary of program variable names, a flow chart, and program listing are included in the appendix.

Data Base

The EMD data base is a block of encoded data, catalogued by type and location on the earth (latitude and longitude). The map data are stored as strings of encoded cartography (including alphanumerics, random shapes such as coastlines and lakes, and distances such as airway vector lengths). A directory which contains the earth location of each data item, its type, and location in the data base is used to search for all map items which are to be retrieved for display at the current time. This directory is called the primary search table (PST) and is depicted in Figure 3-3. This figure shows, for example, that a section of coastline (identifier -05) located at $36^{\circ}\,58'\,47''$ N and $107^{\circ}\,13'\,10''$ W is stored as a relative chain in File 0, starting with the 33rd word.

Most of the encoded map data is stored as either relative or absolute chains. Relative chains are used to encode most of the continuous geographic data such as coastlines, lakes, etc. Absolute chains are used to encode alphanumerics and other special symbols. Alphanumerics are stored in the data base in a chain-encoded form to obviate the requirement for conversion from BCD or ASCII at run time.

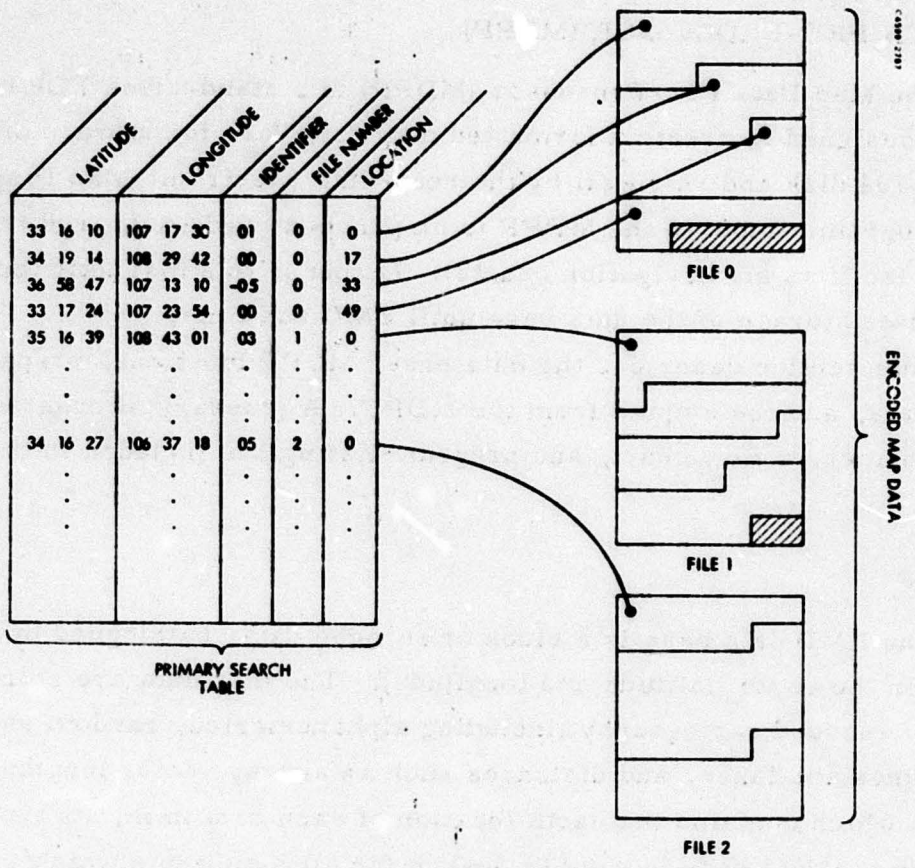


Figure 3-3. EMD data base structure.

In summary, the Map Data Pre-Processor is a program which accepts inputs which describe the contents of the map data base, encodes these data and stores them in the data base, and creates the primary search table. The following paragraphs describe the processing and the preparation of data for the MDPP.

Map Data Pre-Processor Functions

Figures 3-4 and 3-5 illustrate the functions of the MDPP very simply. In Figure 3-4, the data identifier (ID) is read from the first card of a sequence. The ID type determines the format and amount of data which follows. Latitude and longitude are read from the following card along with data corresponding to the particular ID type. The latitude and longitude are converted to seconds of arc and stored, along with the ID type into the

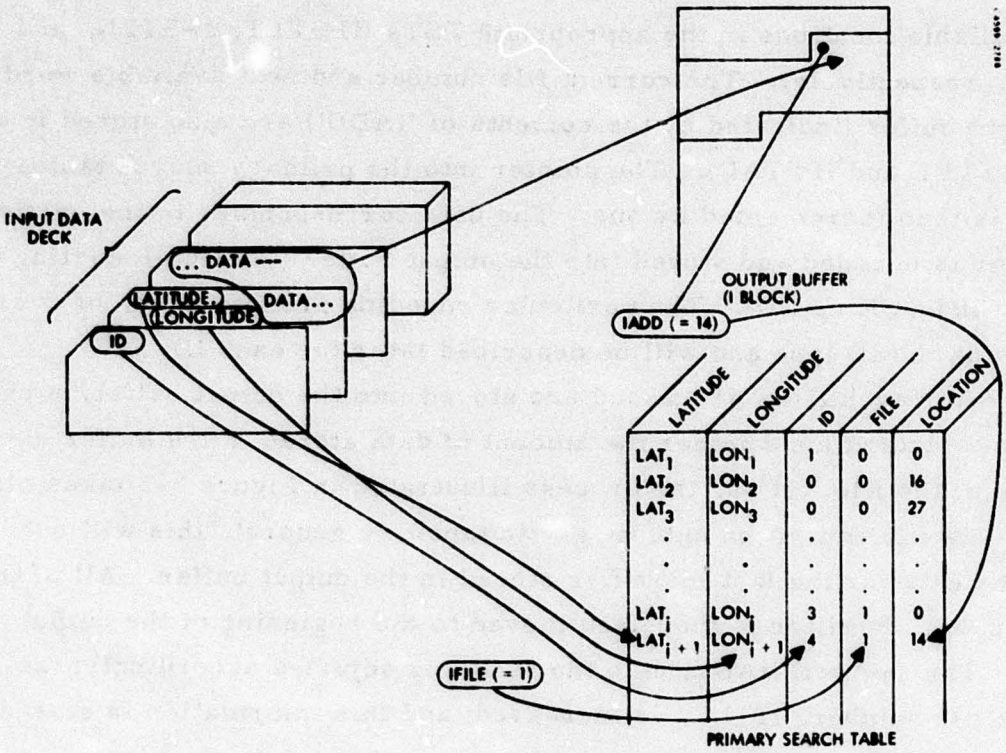


Figure 3-4. Map data pre-processing and primary search table creation.

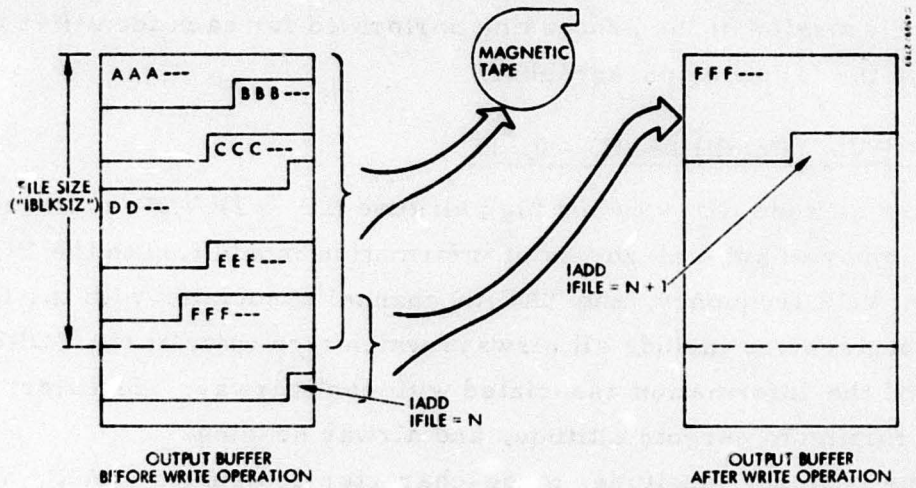


Figure 3-5. Output of map data to magnetic tape.

next available locations in the appropriate PSTs (IPSTLT, IPSTLN, and IPSTID, respectively). The current file number and next available word in the output buffer (indicated by the contents of 'IADD') are also stored in the PST (IPSTFL and IPSTAD). The pointer into the primary search tables (IPNT) is then incremented by one. The data corresponding to the particular identifier is encoded and stored into the output buffer (IBLOCK) starting with location IBLOCK (IADD). The particular encoding process used, of course, depends on the ID type and will be described later for each ID.

After each ID is processed and stored into the output buffer, a check is made to determine whether the amount of data stored in the buffer exceeds the size of the file. If so, the process illustrated in Figure 3-5 takes place. A complete file is written onto magnetic tape. In general, this will not include all of the data for the last identifier stored in the output buffer. All of the data for this identifier is therefore moved to the beginning of the output buffer. The pointer, IADD, into the buffer is adjusted accordingly; the current file number, IFILE, is increased; and this information is stored into the PSTs.

When all of the map data have been read, the last output buffer is output to tape, followed by the primary search tables and an end-of-file (EOF) mark.

A discussion of the processing performed for each identifier type is included in the following paragraphs.

VORTAC Facilities (ID - 0, 1)

Low altitude (ID = 0) and high altitude (ID = 1) VOR facilities include a large number of sub-categories of information in addition to the VORTAC identifier, VOR frequency, and TACAN channel associated with the facility. These subcategories include all airways which terminate at the VORTAC facility and the information associated with each airway: identifier, enroute mileage, minimum enroute altitude, and airway heading.

The latitude, longitude, three-character designator (LAX), VOR frequency (114.7), TACAN channel (056), and number of airways are read from the card following the identifier card. Latitude and longitude are converted to seconds of arc and stored in the PST. The designator, frequency, and channel are converted to absolute chains and stored in the data base.

If the number of airways is greater than zero, successive cards are read, one for each airway. Latitude and longitude of the other end of the airway are read and used to compute the change in X (East) and Y (North) and the length in nautical miles along the airway. These values are stored in the data base and used for computing airway vector lengths and angles on the display. Four alphanumeric fields containing airway heading (135), enroute mileage (76), minimum enroute altitude (18000), and designator (J2-18) are then converted to absolute chains and stored into the data base. Additional cards are read and converted similarly until all airway data for the VORTAC facility have been input and encoded.

Obstructions (ID = 2, 3)

Obstructions (ID = 2) and lighted obstructions (ID = 3) consist of a special symbol and an alphanumeric label indicating the height of the obstruction. In processing obstruction data, latitude and longitude are converted and stored in the PST as for VORTAC facilities. The alphanumeric label (up to 10 characters) is then converted to its absolute chain representation and stored in the data base.

Continuous Data (ID = ±4, ±5, ±6, ±7, ±8)

Continuous data are used to represent a variety of cartographic data: e.g., runway configurations, coastlines, and lakes. The data must be input to the MDPP in a chain encoded form generally obtained by overlaying a suitably scaled grid on the cartography. Positive identifiers indicate that the data are encoded in absolute chains. Negative identifiers signify relative chain encoding. Coastline and lake information can be encoded at two scales (termed high- and low-altitude) to provide more resolution at smaller display scales than would be possible if only one encoding scale were used. In the present software, scales of 0.1 and 0.8 nautical miles per segment have been selected. These scales were selected because all of the map data available at Hughes has been encoded at those scales. The rationale is as

follows: (400 nautical miles covers the display diameter at maximum scale (40 n.m./in.) on a 10 inch display. At the smallest allowable segment size (two display elements per segment) 512 segments will completely traverse the display diameter of 400 n.m. Thus, one segment is equal to 400 n.m/ 512 segments (= 0.8 n.m./segment). At the two smallest scale selections (1.25 and 5 nautical miles/inch) segmentation of the cartography becomes quite noticeable. Thus, the decision was made to encode all cartography likely to be used at the lower scales (e.g., approaching and departing terminal areas) at correspondingly greater detail, or 0.1 n.m./inch.

Thus, the following identifiers have been defined:

±4 - Terminal area (e.g., runways)
±5 - Coastlines (large scale)
±6 - Coastlines (small scale)
±7 - Lakes (large scale)
±8 - Lakes (small scale)

All of these data types are treated similarly by the MDPP. After storing the latitude and longitude into the PST, the chain representations on the source cards are re-formatted into absolute or relative chains for the 'DSYMB' display utility routine.

Alphanumeric-to-Chain Conversion

Conversion of Hollerith data chain-encoded alphanumerics is a straightforward process and is depicted in Figure 3-6. A dictionary of the chain-encoded representation of each alphanumeric symbol is stored as a table in the computer. Each alphanumeric symbol is encoded to start in the lower left-hand corner of the character font matrix and end one space to the right of the lower right-hand corner - or, where the following character will begin. Figure 3-7 illustrates several characters encoded into a 2-by 4-segment font matrix.

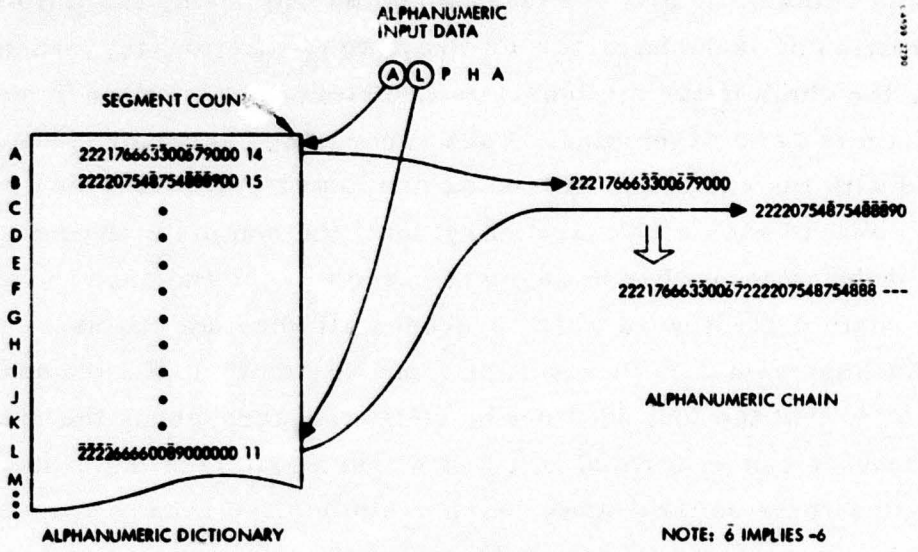


Figure 3-6. Conversion of Hollerith data to alphanumeric chains.

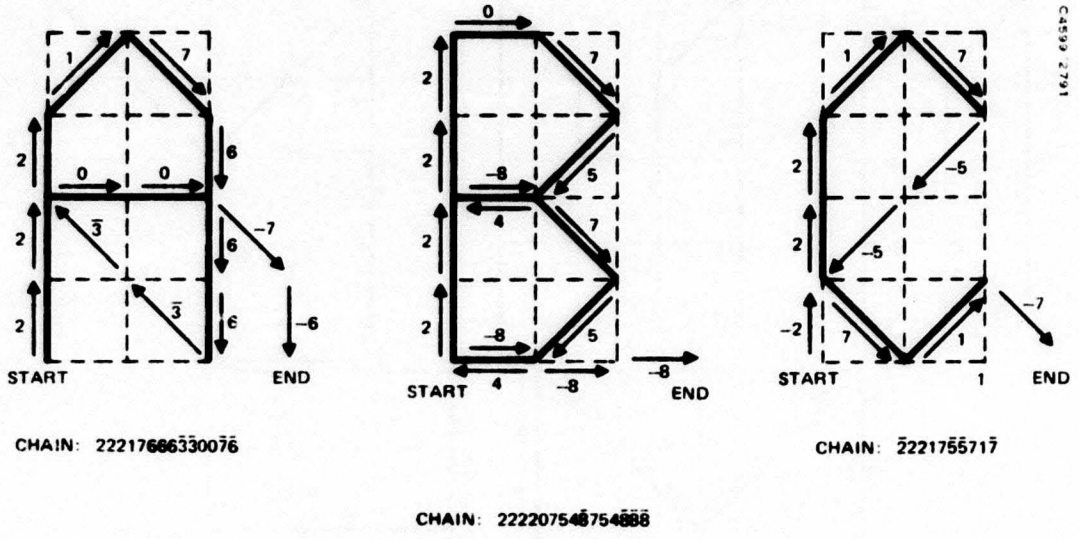


Figure 3-7. Chains generated from 2 x 4 segment matrix.

The translation process takes place by comparing the Hollerith representation of each character with keys to the dictionary. When a match is found, the chain in the dictionary is transferred to a buffer in which the entire chain is to be assembled. Each succeeding character chain is concatenated with the end of the preceding one (determined with the use of the segment count of each dictionary entry) until the complete chain is formed. A count of the total number of segments used is kept and used to form the absolute chain control word which precedes all absolute chains.

An important benefit resulting from this method of alphanumeric generation is that the font need not be consistent throughout the dictionary. Each character can be formed in a font which maximizes legibility. Figure 3-8 illustrates several fonts which could be used for the character M and the concatenation of one of them with another dissimilar font to form the word ME. The entire symbol library or individual symbols within the library can be changed simply by altering the dictionary.

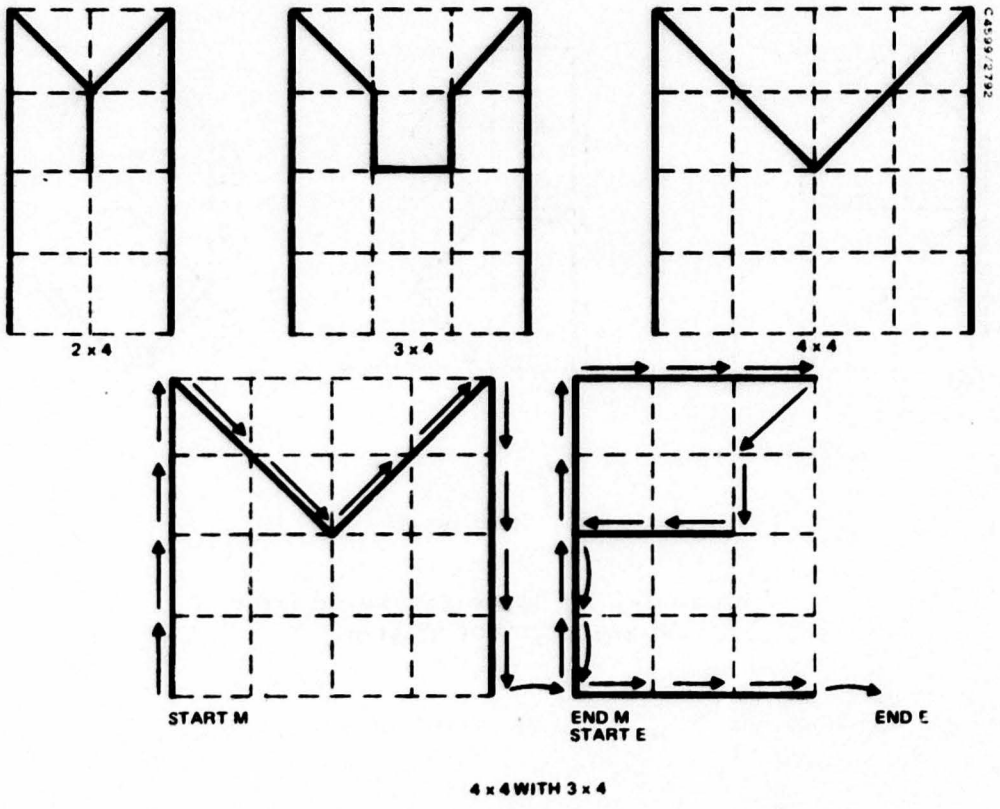


Figure 3-8. Mixing characters with different fonts.

Input Data Preparation

Preparation and input of data to the Map Data Pre-Processor is an uncomplicated process. The data are input in two sections as shown in Figure 3-9. The first section is the alphanumeric chain dictionary. A dictionary has been supplied with the software. However, the alphanumerics may be changed simply by altering the data in the first section.

The second section of input data to the MDPP contains the map data. It consists of a number of groups of cards, each group representing one map display item (VORTAC facility, coastline section, etc.). The first card in each group contains the identifier for that data item. It determines the format of the data on the following cards. The following paragraphs describe the contents of the data cards as a function of the type of data.

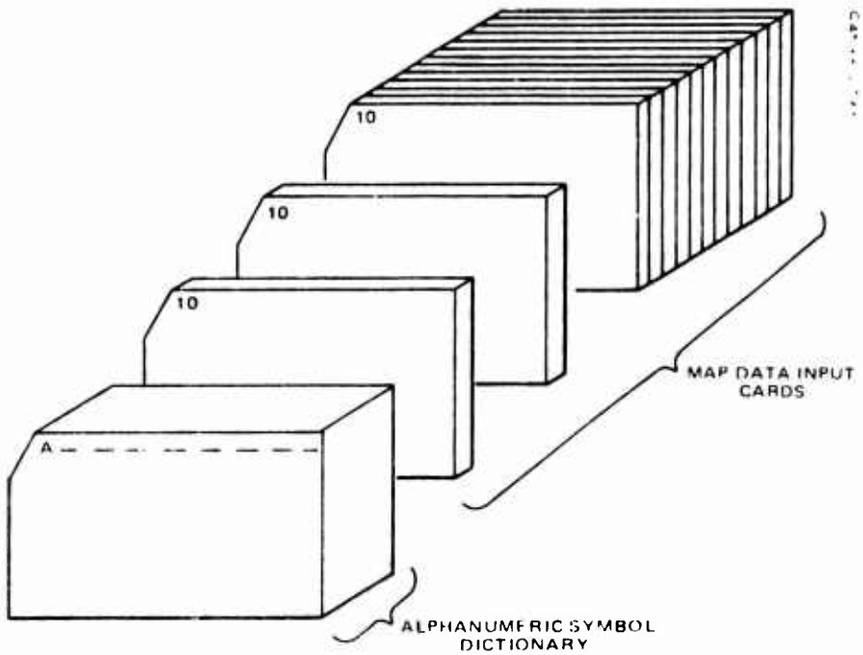


Figure 3-9. MDPP input deck structure.

VORTAC Facilities

Table 3-2 contains a description of the data and the format of the input cards for VORTAC facilities. The first card contains only the identifier number. Low altitude VORTAC facilities are identified by O and high altitude facilities by 1. The following card contains the latitude and longitude of the facility, its designator, its frequency, its TACAN channel (if it exists), and the number of airways terminating at this facility.

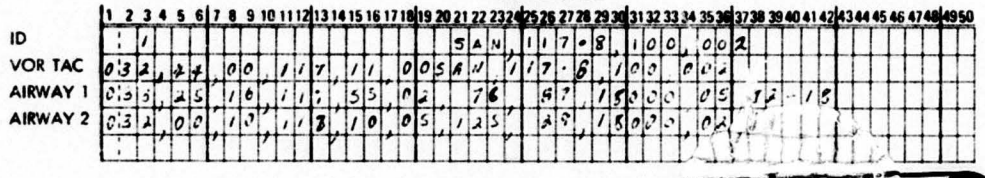
TABLE 3-2. VORTAC FACILITY INPUT CARD FORMATS

Card	Card Col.	Format	Parameter Symbol	Description
1	1-3	I3	ID	Identification No. (= 0 or 1)
2	1-3	I3	LATD	Latitude (deg)
	5-6	I2	LATM	Latitude (min)
	8-9	I2	LATS	Latitude (sec)
	11-13	I3	LOND	Longitude (deg)
	15-16	I2	LONM	Longitude (min)
	18-19	I2	LONS	Longitude (sec)
	21-23	3A1	IDES	VOR Designator
	25-29	5A1	IFREQ	VOR Frequency
	31-33	3A1	ICHAN	TACAN Channel
	35-36	I2	NAIR	Number of Airways
(Note: If NAIR equals zero no more cards are needed, otherwise there will be one card for each airway.)				
3-	1-3	I3	NDLATD	End Point Latitude (deg)
2+NAIR	5-6	I2	NDLATM	End Point Latitude (min)
	8-9	I2	NDLATS	End Point Latitude (sec)
	11-13	I3	NLOND	End Point Longitude (deg)
	15-16	I2	NLONM	End Point Longitude (min)
	18-19	I2	NLONS	End Point Longitude (sec)
	21-23	3A1	IHDGA	Heading of Airway
	25-27	3A1	MLGA	Enroute Mileage
	29-33	5A1	IALTA	Minimum Enroute Altitude
	35-36	I2	N	Number of Characters in Airway Designator
	38-47	10A1	IDESIG	Airway Designator

The third through $(2+N)^{th}$ cards (N = number of airways) contain information regarding each airway. This includes the end latitude and longitude of the airway, its heading, mileage, minimum enroute altitude and its alphanumeric designator. The end latitude and longitude is the point at which the airway vector emanating from the VORTAC facility will terminate. This point is typically an intersection or a point midway between two VORTAC facilities. This point is used to define the length and angle at which the airway vector is to be drawn (the heading designator, for example, is not accurate enough). The airway designator is preceded by an integer specifying how many characters (not to exceed 10) are in the designator. Figure 3-10 is an example of how VORTAC facility SAN with two airways might be input to the MDPP.

			EXAMPLE 001	
ID				
LATITUDE	DEG, MIN, SEC	(INTEGERS)	032, 44, 00	
LONGITUDE	DEG, MIN, SEC	(INTEGERS)	117, 11, 00	
VORTAC SYMBOL DESIGNATOR		(3 CHARACTERS)	SAN	
VORTAC FREQUENCY DESIGNATOR		(5 CHARACTERS)	117.8	
*TACAN CHANNEL DESIGNATOR		(UP TO 3 CHARACTERS)	100	
NUMBER OF AIRWAYS FROM VORTAC FACILITY		(2 DIGIT INTEGER)	02	
FOR EACH AIRWAY				
END LATITUDE	DEG, MIN, SEC		033, 25, 16	032 00, 10
END LONGITUDE	DEG, MIN, SEC		117, 55, 02	116, 10, 05
HEADING DESIGNATOR	(DEG)	(UP TO 3 CHARACTERS)	76	125
MILEAGE DESIGNATOR	(NM)	(UP TO 3 CHARACTERS)	87	29
ALTITUDE DESIGNATOR	(FT)	(UP TO 5 CHARACTERS)	18000	18000
NUMBER OF CHARACTERS IN AIRWAY DESIGNATOR			05	02
AIRWAY DESIGNATOR	(CHARACTERS)		J2 - 18	J1

CODING SHEET



*IF THERE IS NO TACAN CHANNEL LEAVE BLANK

Figure 3-10. Input data example: VORTAC facilities.

Obstructions

Table 3-3 shows the input format for cards describing obstructions (ID = 2) and lighted obstructions (ID = 3). In addition to latitude and longitude, a descriptor (generally obstruction height) is included. The descriptor is limited to 10 alphanumeric characters and is preceded by a character count. In the example of Figure 3-11 the descriptor K 787, indicating an obstacle height of 787 feet.

Continuous Data

As discussed above, continuous data include terminal area data (runways), coastlines, and lakes. All of these data are input similarly. In addition to the ID and latitude and longitude, the user inputs either a relative or absolute chain which represents the data being input. Thus, the number of cards used to input the entire chain and the number of chain segments on each are required by the MDPP. Tables 3-4 and 3-5 describe the input formats for the input of continuous data in either absolute or relative chains,

TABLE 3-3. OBSTRUCTION INPUT CARD FORMATS

Card	Card Column	Format	Parameter Symbol	Description
1	1-3	I3	ID	Identification No. (= 2 or 3)
2	1-3	I3	LATD	Latitude (deg)
	5-6	I2	LATM	(min)
	8-9	I2	LATS	(sec)
	11-13	I3	LOND	Longitude (deg)
	15-16	I2	LONM	(min)
	18-19	I2	LONS	(sec)
	21-22	I2	N	No. of characters in Descriptor
	24-33	10A1	IDESIG	Descriptor

ID			003
LATITUDE	DEG, MIN, SEC	(INTEGER)	037, 36, 00
LONGITUDE	DEG, MIN, SEC	(INTEGER)	122, 24, 00
NUMBER OF CHARACTERS IN DESIGNATOR		(INTEGER)	03
DESIGNATOR		(CHARACTERS)	787

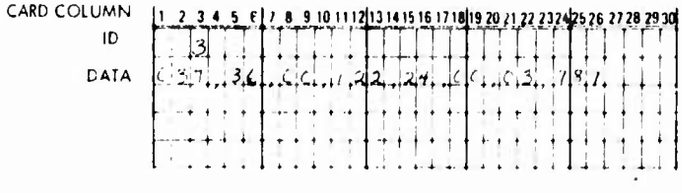


Figure 3-11. Input data example: obstructions.

TABLE 3-4. CONTINUOUS DATA INPUT CARD FORMATS (ABSOLUTE CHAINS)

Card	Card Column	Format	Parameter Symbol	Description
1	1-3	I3	ID	Identification No. (= 4, 5, 6, 7, 8)
2	1-3	I3	LATD	Latitude (deg)
	5-6	I2	LATM	(min)
	8-9	I2	LATS	(sec)
	11-13	I3	LOND	Longitude (deg)
	15-16	I2	LONM	(min)
	18-19	I2	LONS	(sec)
	21-22	I2	NCARDS	Number of Data Cards to Follow
	24-25	I2	NSPC(1)	Number of Segments on Card 1
	27-28	I2	NSPC(2)	Number of Segments on Card 2
	29-30	I2	NSPC(3)	Number of Segments on Card 3
	32-33	I2	NSPC(4)	Number of Segments on Card 4
	35-36	I2	NSPC(5)	Number of Segments on Card 5
Cards 3 to card 2 + N Cards				
3	1-2	I2	1 Seg (1)	Absolute Segments
	4-5		1 Seg (2)	
	7-8		1 Seg (3)	
	10-11		1 Seg (4)	
	13-14		1 Seg (5)	
	16-17		1 Seg (6)	
	19-20		1 Seg (7)	

(Continued next page)

(Table 3-4, concluded)

Card	Card Column	Format	Parameter Symbol	Description
	22-23		1 Seg (8)	
	25-26		1 Seg (9)	
	28-29		1 Seg (10)	
	31-32		1 Seg (11)	
	34-35		1 Seg (12)	
	37-38		1 Seg (13)	
	40-41		1 Seg (14)	
	43-44		1 Seg (15)	
	46-47		1 Seg (16)	
	49-50		1 Seg (17)	
	52-53		1 Seg (18)	
	55-56		1 Seg (19)	
	58-59		1 Seg (20)	
	61-62		1 Seg (21)	
	64-65		1 Seg (22)	
	67-68		1 Seg (23)	
	70-71		1 Seg (24)	

TABLE 3-5. CONTINUOUS DATA INPUT CARD FORMATS (RELATIVE CHAINS)

Card	Card Column	Format	Parameter Symbol	Description
1	1-3	I3	ID	Identification No. (= -4, -5, -6, -7, -8)
2	1-3	I3	LATD	Latitude (deg)
	5-6	I2	LATM	(min)
	8-9	I2	LATS	(sec)
	11-13	I3	LOND	Longitude (deg)
	15-16	I2	LONM	(min)
	18-19	I2	LONS	(sec)
	21-22	I2	IABSEG	Absolute direction code of 1st segment
	24-25	I2	NCARDS	Number of cards
	27-28	I2	NSPC (1)	Number of Segment per Card 1
	30-31	I2	NSPC (2)	Number of Segment per Card 2
	33-34	I2	NSPC (3)	Number of Segment per Card 3
	36-37	I2	NSPC (4)	Number of Segment per Card 4
	10-41	I2	NSPC (5)	Number of Segment per Card 5
3	Cards 3 to 2 + N Cards			
2 + N Cards are the same as for absolute segments (ID = 4, 5, 6, 7, 8), See Table 3-4.				

respectively. The primary distinction between the two (other than the chains) is that for relative chains the second card contains the absolute direction code of the first chain segment where, of course, this is not required for absolute chains.

The general process of encoding continuous cartographic data in either relative or absolute chains is accomplished through the use of a grid overlay. The grid (or the map source) should be appropriately scaled so that one grid unit in the x or y direction is matched to the appropriate distance on the map. For example, the data encoded for the Hughes EMD was collected by using grid paper with a 0.1 inch grid. The grid was overlaid on maps which had been photographically scaled to 1 n.mi./inch and 8 n.mi./inch. Thus, the grid could be used to generate segmented chains of 0.1 and 0.8 n.mi./segment. (Note that this applies to horizontal and vertical segments only (direction codes 0, 2, 4, 6). The diagonal segments (direction codes 1, 3, 5, 7) of course are greater by a factor of $\sqrt{2}$.)

Once the grid and map source have been obtained and are appropriately scaled, encoding proceeds by overlaying the grid on the map and tracing a continuous line on the grid which as closely as possible represents the shape of the feature being encoded. The line must be drawn from one point on the grid to one of its eight nearest neighbors. Figure 3-12 is an example of a section of coastline drawn on a grid in this way.

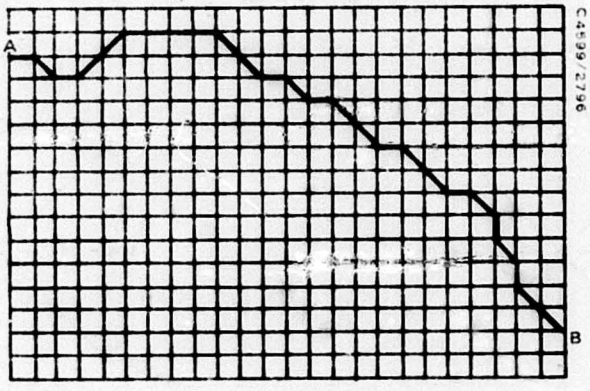


Figure 3-12. Section of coastline transferred to grid.

When the continuous shape has been transferred to the grid, it can be encoded into either a relative or absolute chain. The absolute chain is a sequence of numbers representing the absolute direction taken in going from one grid point to the next according to the diagram in Figure 3-13. For example, the data of Figure 3-12 would be encoded as an absolute chain starting at A as follows:

0, 7, 0, 1, 1, 0, 0, 0, 0, 7, 7, 0, 7, 0, 7, 7, 0, 7, 7, 0, 7, 6, 7, 6, 7, 7

The same shape, encoded starting at B is:

3, 3, 2, 3, 2, 3, 4, 3, 3, 4, 3, 3, 4, 3, 4, 3, 3, 4, 4, 4, 4, 5, 5, 4, 3, 4

Although none are shown in the example, hidden (or blanked) segments are encoded as a negative number, the absolute value of which is the direction code. The ambiguity of direction code 0 is resolved by using -8 to represent a hidden segment in the zero direction.

The relative chain code for a continuous shape is formed in much the same way. In this case, however, each segment direction is defined in terms of the direction of the preceding segment direction — relative to the preceding segment — rather than in an absolute sense. The initial segment of a relative chain must be defined as an absolute direction code. Figure 3-14 illustrates the relative direction codes. Note that mnemonic codes are accepted by the MDPP to facilitate input by the user. Translation to the appropriate two- or four-bit binary code required by the display hardware is

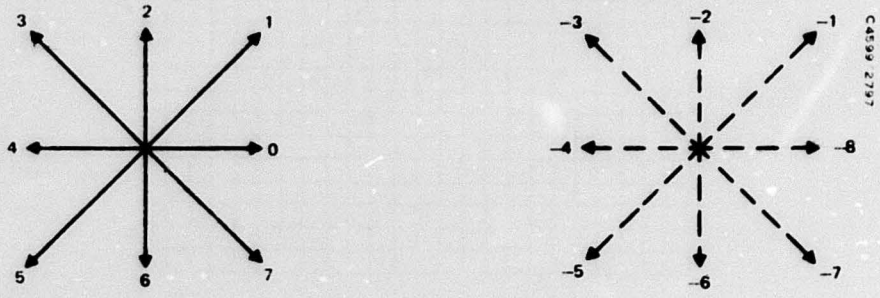
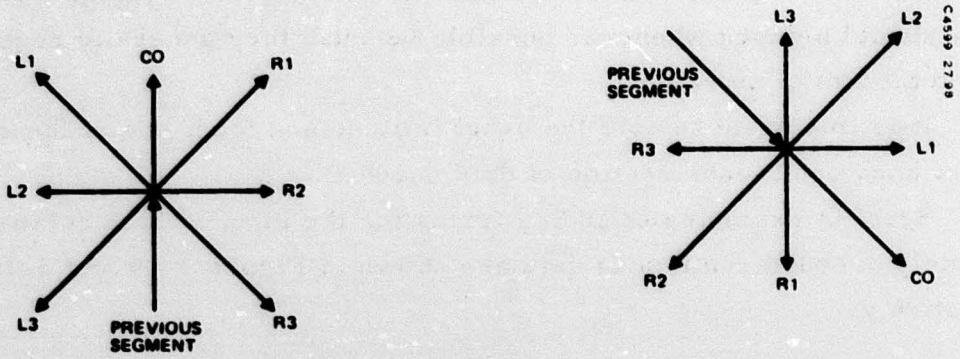


Figure 3-13. Absolute chain direction codes.



KEY:
- CO – CONTINUE STRAIGHT AHEAD
- R1 – GO 1 DIRECTION INCREMENT (45°) RIGHT
- R2 – GO 2 DIRECTION INCREMENTS (90°) RIGHT
- R3 – GO 3 DIRECTION INCREMENTS (135°) RIGHT
- L1 – GO 1 DIRECTION INCREMENT (45°) LEFT
- L2 – GO 2 DIRECTION INCREMENTS (90°) LEFT
- L3 – GO 3 DIRECTION INCREMENTS (135°) LEFT

Figure 3-14. Relative chain direction codes.

automatic. The example of Figure 3-12 would be encoded as follows starting at A;

00, R1, L1, L1, C0, R1, C0, C0, C0, R1, C0, L1, R1, L1, R1, C0, L1, R1, C0, L1, R1, R1, L1, R1, L1, C0.

Or, starting at B, its code would be:

3, C0, R1, L1, R1, L1, L1, R1, C0, L1, R1, C0, L1, R1, L1, R1, C0, L1, C0, C0, C0, L1, C0, R1, R1, L1.

The selection of relative or absolute encoding is at the user's option and depends to a large extent on the data itself. In general, continuous data such as coastlines which appears "smooth" is well-suited for relative chain encoding. The chain is simply started at one point and drawn continuously to another point. Conversely, data such as runway configurations which are more angular in appearance and which may require the use of invisible lines or doubling back can best be done with absolute chains.

The MDPP has no provision for video control for relative chains. Relative chains should be used whenever possible because they generally require fewer total bits of storage.

It is important to note the exact latitude and longitude of the starting point of each continuous section of data encoded.

Specific examples of coding forms for the input of both relatively and absolutely encoded continuous data are shown in Figure 3-15 and 3-16, respectively.

ID		4	
LATITUDE OF 1ST SEGMENT	(DEG, MIN, SEC)	034, 00, 00,	
LONGITUDE OF 1ST SEGMENT	(DEG, MIN, SEC)	118, 29, 00,	
NUMBER OF DATA CARDS TO FOLLOW		02	
NUMBER OF SEGMENTS PER CARD		24	1ST CARD
		5	2ND CARD

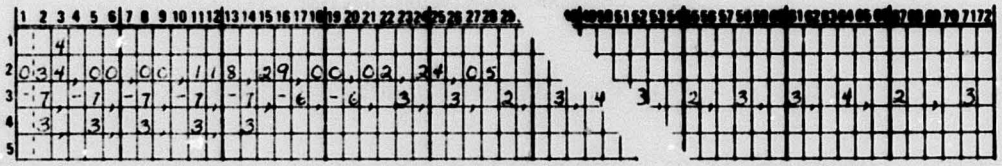


Figure 3-15. Example of absolute chain encoded continuous data.

ID		-5	
LATITUDE OF FIRST SEGMENT	(DEG, MIN, SEC)	033, 52, 00	
LONGITUDE OF FIRST SEGMENT	(DEG, MIN, SEC)	118, 23, 00	
ABSOLUTE SEGMENT	(INTERGER)	03	
NUMBER OF DATA CARDS	(INTEGER)	02	
NUMBER OF SEGMENTS PER CARD		24	1ST CARD
		10	2ND CARD

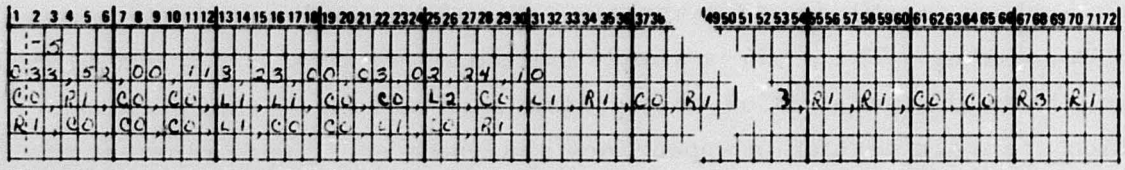


Figure 3-16. Example of relative chain encoded continuous data.

Output

As the Map Data Pre-Processor processes the input cards, the data base is assembled in the output buffer (IBLOCK). Whenever the buffer becomes full, it is output to magnetic tape as one logical record. (Each logical record is called a file in the MDPP.) The contents of each record depend entirely on type of identifiers which have been processed. Heading or identification data is not output with each record, since the primary search table contains all information necessary to locate data within a record.

When all data has been processed and output to tape, the PST is output as the last record (file) on the tape. The tape is then rewound and read to verify the write process. The contents of each record (file) are output to the line printer to provide a hard copy of the data base. The PST is also read from tape and output to the line printer.

The format of the data within a file or record is shown in Tables 3-6 through 3-8 as a function of data type. As the tables show, a large part of the data in the data base is in the form of absolute or relative chains. Figure 3-17 shows the format of each. In general, each begins with a control word containing a count of the number of segments which follow. In the case of relative chains this count isn't the number of segments but, rather, the number of 2-bit nibbles (nibble is half a byte) required to define the chain. Recall from the hardware description of Section 2.0 that relative chaining requires 2 bits per segment in some cases and 4 bits per segment in others. Relative and absolute chain control words are followed by a number of data words sufficient to include the entire chain.

DATA RETRIEVAL PROGRAM (DATARET)

The Data Retrieval Program, DATARET, performs an important part of the real time electronic map display processing functions. Its primary function is to select and retrieve all data from the bulk data base which may be processed by the Category Processor. The following paragraphs describe the operation of the Data Retrieval Program. Detailed flow charts and a symbol glossary may be found in Appendix A.

TABLE 3-6. OUTPUT DATA FORMAT FOR LOW- AND HIGH-ALTITUDE VORTAC FACILITIES

Item	Description
1	Control word for VORTAC Facility Designator contains OP code for absolute chain plu number of segments -1 in chain to follow. Example: 9015_{16}
2	VORTAC Facility Designator (absolute chain)
3	Control word for VOR Frequency Designator
4	VOR Frequency Designator (absolute chain)
5	Control word TACAN channel designator
6	TACAN Channel Designator (absolute chain)
7	Number of airways from this facility (Integer) (8 - 18 repeated for each airway)
8	End latitude of airway vector (NM) (1 word integer)
9	End longitude vector (NM) (1 word integer)
10	Length of airway vector (NM) (1 word integer)
11	Control word for heading chain
12	Heading (absolute chain)
13	Control word for mileage chain
14	Mileage (absolute chain)
15	Control word for altitude chain
16	Altitude (absolute chain)
17	Control word for designator chain
18	Airway designator (absolute chain)

TABLE 3-7. OUTPUT DATA FORMAT FOR OBSTRUCTION

Item	Description
1	Control word for absolute chain for descriptor
2	Absolute chain for obstruction descriptor

TABLE 3-8. OUTPUT DATA FORMAT FOR CONTINUOUS DATA

Item	Description
1	Control word for relative or absolute chain
2	Relative or absolute chain

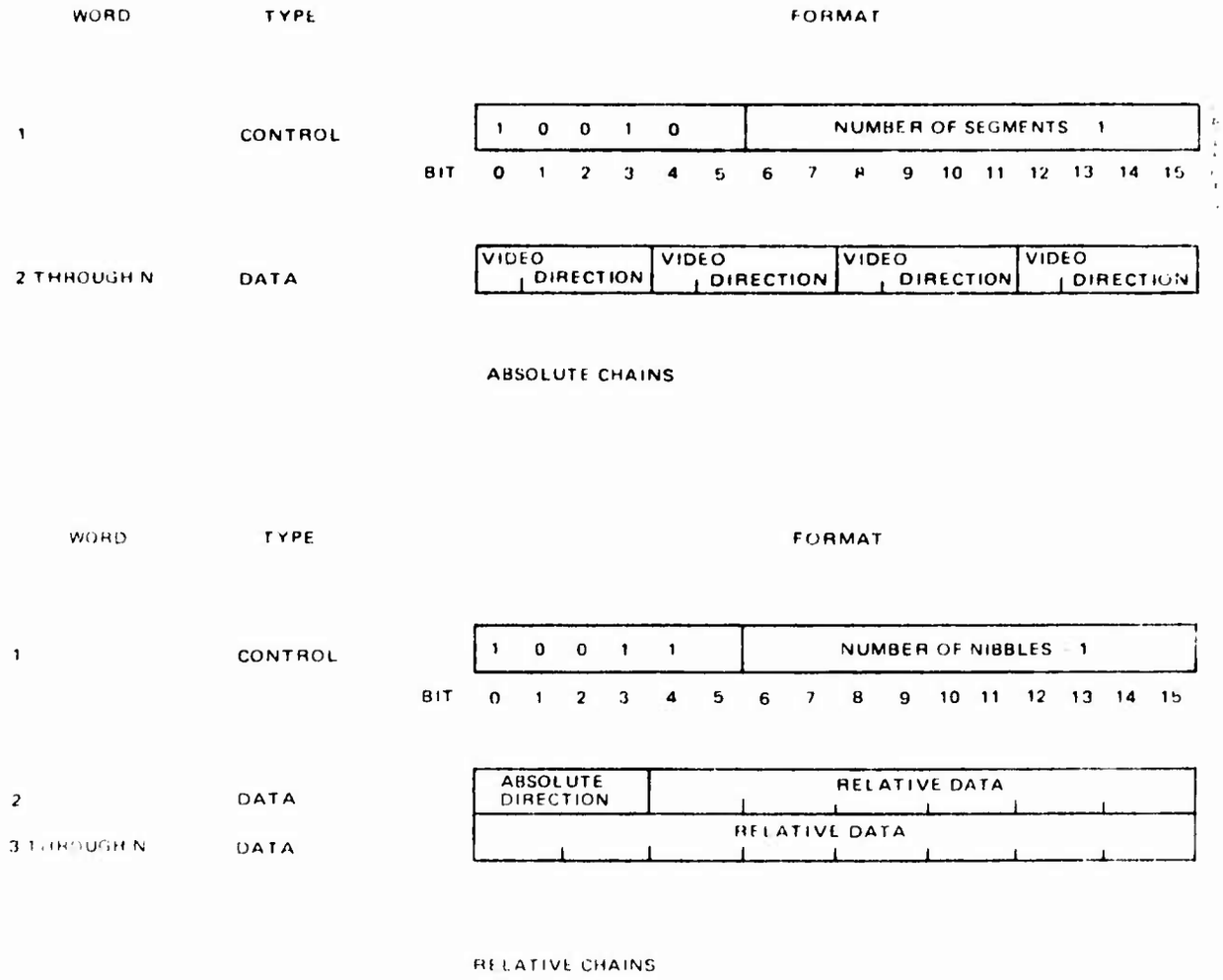


Figure 3-17. Absolute and relative chain data formats.

Figure 3-18 is a general program flow chart. Reference to it will be helpful in more easily understanding the following discussion.

Formation of the In-Window Table

The first function performed by DATARET is the computation of the search area and a search of the primary search table for all map data items within the search area.

The limits of the search area are determined by the present aircraft location and the scale selected for display. The display radius is converted to seconds of arc and is added and subtracted to aircraft latitude and longitude. This defines the latitude and longitude limits of a square search area as shown in Figure 3-19.

When the search area has been defined, the latitude and longitude of each entry in the primary search table is compared to the search area limits. Pointers to each entry in the PST which are found to be in the search area are stored into a table (the INWINDO table) as shown in Figure 3-20. Thus, the ID of the I^{th} item in the INWINDO table is referenced by the assembly language equivalent of the FORTRAN statement:

ID = IPSTID (INWINDO(I))

Data Retrieval

When the INWINDO table has been formed, DATARET initializes a new display buffer, generally different from the previous buffer.

To minimize disk access time, each file is read into memory only once per display update period. All display items in the INWINDO table and in a common file are processed before the next file in sequence is read. The process is depicted in Figure 3-18.

The retrieval of data is begun by initializing the file pointer, IFILE, to 1. The INWINDO table is searched to determine if there are any map items in the search window in the first file. If not, the file pointer is incremented and the same test is applied to successive files.

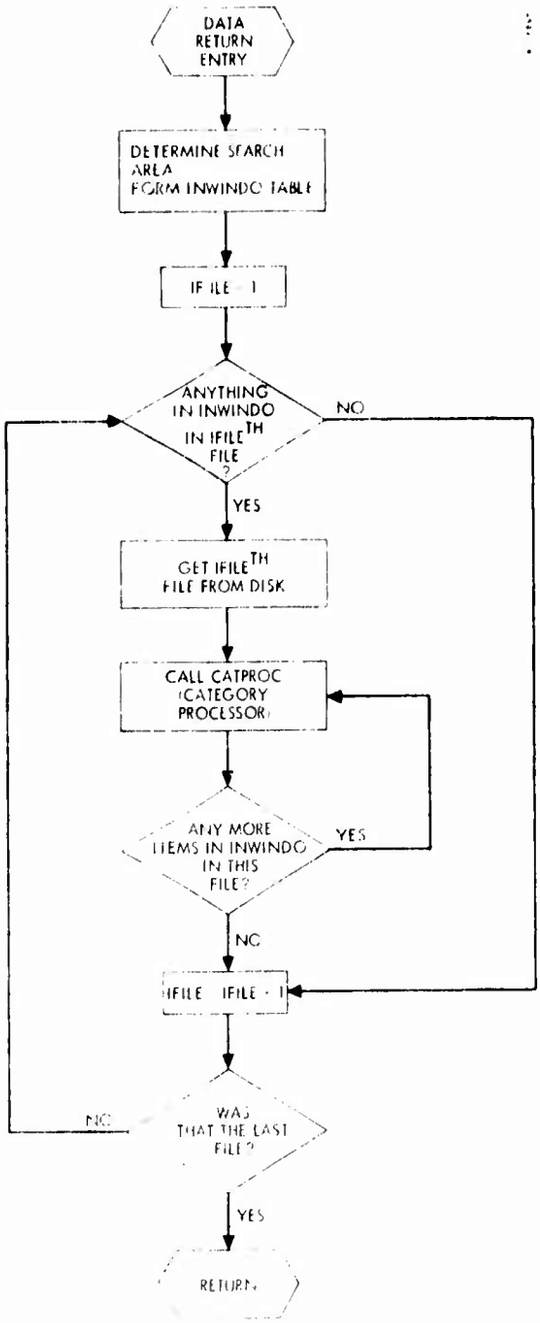


Figure 3-18. Data retrieval flowchart.

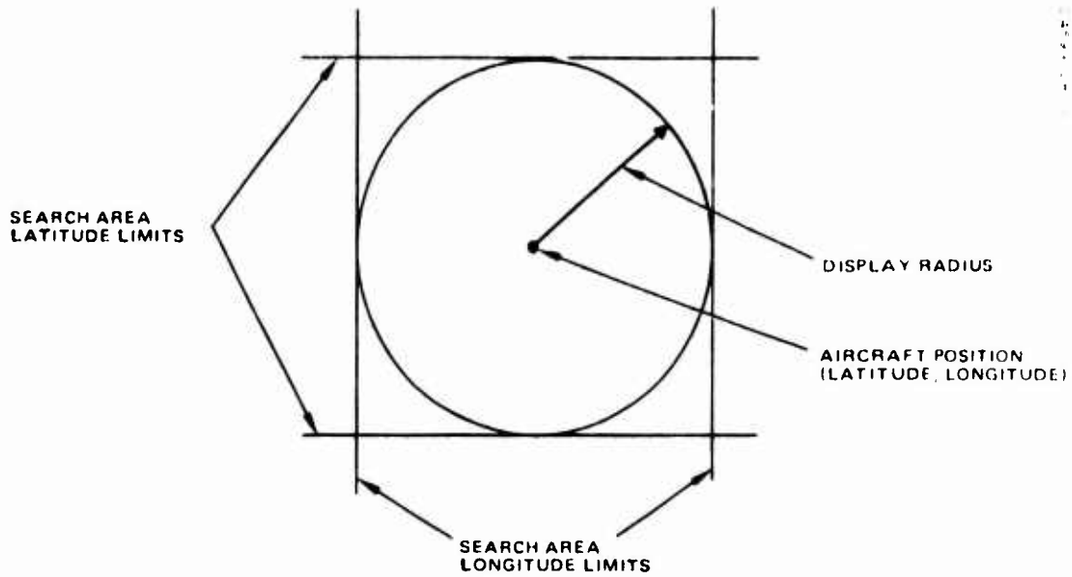


Figure 3-19. Computation of search area limits.

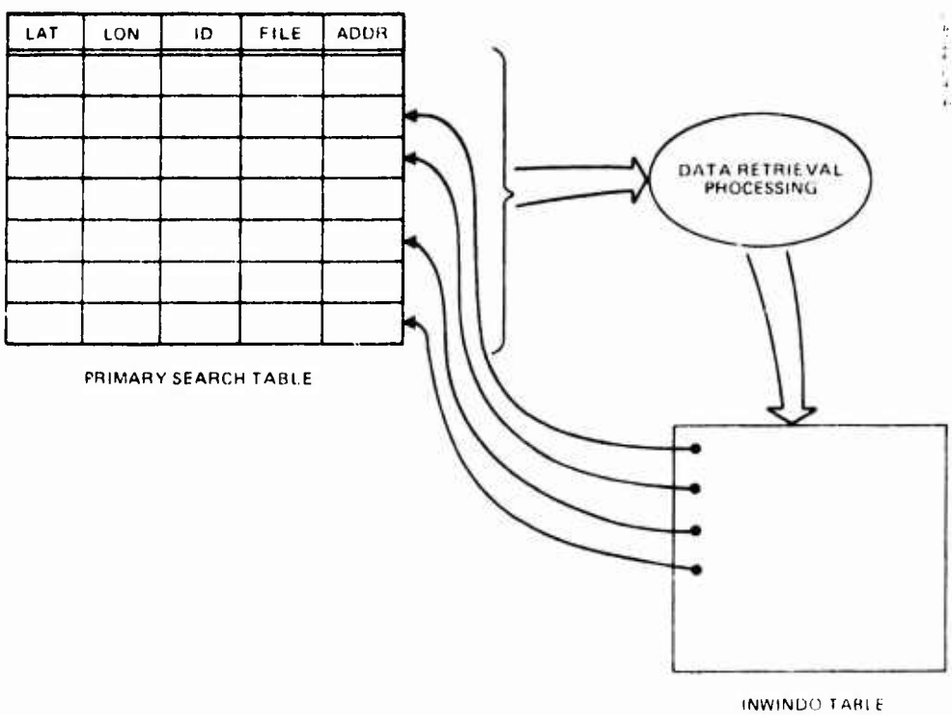


Figure 3-20. Generation of the INWINDO table.

If there is an item in the window in the first file, then that file is retrieved from the disk into memory. The category processor (CATPROC) is then called to process the display time. The INWINDO table is checked to see if there are any other map items in the file currently in memory. If there are, CATPROC is called to process each one until there are no remaining unprocessed items in the current file. The file pointer is then incremented and the entire process repeated until all map items and all files have been processed.

When the last file has been completely processed, the aircraft symbol is stored in the display list. It is positioned at the center of the display, oriented to indicate the aircraft heading. The display list is then closed and the data retrieval is completed.

CATEGORY PROCESSING AND DISPLAY FORMATTING

The category processing and display formatting program (called the category processor) computes the structure and content of the navigation display as a function of the bulk map data and the control switch settings. In order to accomplish this end, the category processor performs the following functions for each map data item.

- Computes display position relative to ownship aircraft
- Obtains chain data from map data base
- Computes placement of alphanumeric labels
- Determines display size of all labels and chains
- Adds instructions to the display list to display the data item.

Table 3-9 lists the categories of data which are currently being processed by the category processor. Display formatting features include the functions listed in Table 3-10. The category processor design allows for future expansion of the map data categories as well as the display formatting functions.

The category processor routine is called by the data retrieval program for each map item which is included in the navigation display. The category processor first computes the display position of the map item and

TABLE 3-9. MAP DATA CATEGORIES

VOR Station
VOR Frequency
TACAN Channel Number
Airway
Airway Designator
Airway Heading
Airway Mileage
Minimum Enroute Altitude
Airport Runway Configurations
Coastlines
Lakes
Obstructions
Lighted Obstructions

TABLE 3-10. MAP DISPLAY FORMATTING FUNCTIONS

1. Scale (up to 5 separate scales)
2. Altitude structure
a. Terminal area
b. Low altitude
c. High altitude
3. North Up Display Orientation
4. Aircraft at the center

verifies that it falls within the map region currently being displayed. If so, a branch is made to a specific portion of the category processor routine depending upon the classification of the map data item being processed. This portion of the category processor program then constructs the display generator instructions which will result in this data item being displayed in the proper location on the navigation display. As soon as this is accomplished, program control is returned to the data retrieval program.

A functional flowchart of the category processor is presented in Figure 3-21.

The following discussion is devoted to pertinent details of the operation of the category processor routine. These are included to enable the user to fully understand the category processor routine and to enable him to make any modifications or additions he may feel are warranted.

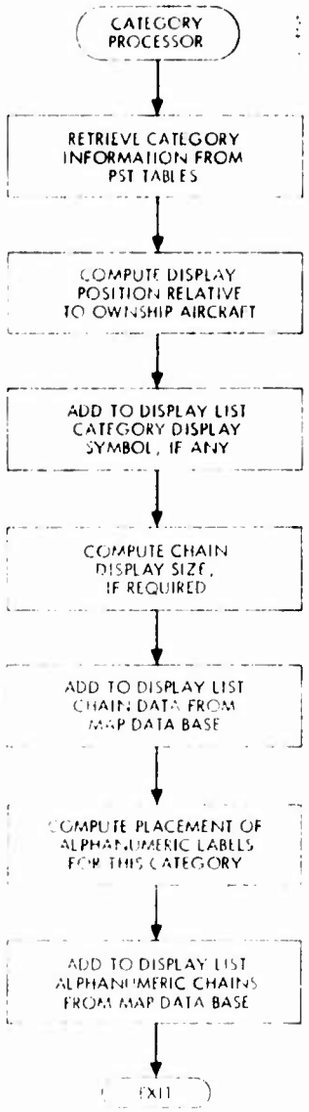


Figure 3-21. Category processor functional flowchart.

The category processor routine receives no arguments from the data retrieval program, but it does access one variable and one table which are established by the data retrieval program. The variable, ISFL, is an index to the table, INWINDO, which contains the primary search table (PST) indices of all items in the display window. By setting IDATA = INWINDO (ISFL), the PST index of the map data item currently being processed is obtained. By using IDATA as an index to the PST tables, the category processor routine can obtain the following information for the map data item under consideration: latitude, longitude, cosine of the latitude, item category, and location in the map data base input buffer. This information is then used throughout the remainder of the category processor program.

As stated above, the category processor stores the chain data corresponding to each map data category into the display list. This chain data is retrieved from the map data base by a subroutine of the category processor entitled CHADD. CHADD computes the number of words in the chain and calls the display utility routine DSYMB, which stores the chain into the display list. Each time CHADD is called, a check is made to ensure that the display buffer does not overflow. If a particular data chain is not chosen for display, a subroutine entitled CHSKIP will advance the map data base index, INDEX, around that chain. INDEX is originally set to the initial location of the map item in the map data base as determined from the PST tables. INDEX is then incremented whenever a chain is retrieved from the data base or is skipped. Not all chains, however, are in the data base. The standard symbols, such as aircraft symbol, VORTAC symbol and obstruction symbol, are encoded in separate arrays defined within the category processor data.

The placement of the VOR and airway labels and data is shown in Figure 3-22. This scheme was chosen to conform to navigation charts and to reduce display clutter. Also shown in that figure is the fact that the first portion of the airway vector is blanked. This is done to eliminate the bright spot that results at the VOR station where all the airways originate. The amount of the vector which is blanked is specified by a variable defined in the category processor program and can be altered easily. Likewise, the

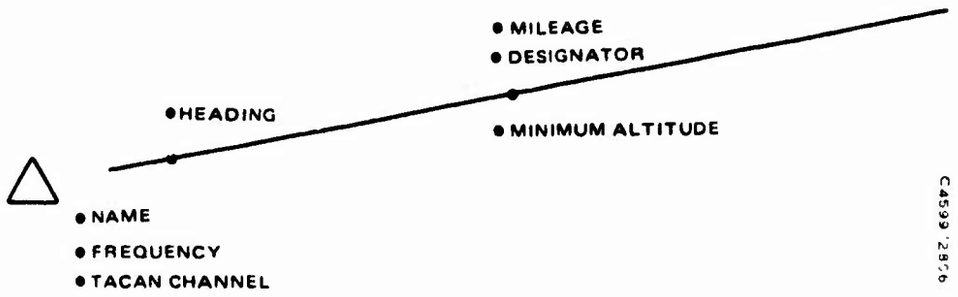


Figure 3-22. Placement of VOR and airway data.

positions of the airway data can be easily altered. Each of the labels and/or data words has a category processor defined variable associated with it that specifies the distance from it to the airway vector. In addition, the heading label has a variable associated with it that specifies the radial distance from the VOR station to its position along the vector. The other airway data is always placed halfway betwen the VOR station and the end of the airway vector or the edge of display if the end of the airway vector is off of the display. It can also be seen from Figure 3-22 that the airway data is rotated the same angle as the airway vector itself while the VOR data is always written horizontally. This is done in an attempt to reduce display clutter. The equations are written such that the alphanumerics are always written right-side-up and always maintain the same relationship to the airway vector, i.e., heading, designator and mileage always above and minimum enroute altitude always below the vector.

A list of the map data categories is given in Table 3-11. There are two ID numbers for coastlines and lakes to maintain adequate resolution for all five range scales. The difference between the two types of chain data is the segment size in which the chain is encoded.

Table 3-12 contains a list of the display selection variables used by the category processor. A value of "1" signifies the item is chosen for display while "0" indicates that the item is not to be displayed. These, of course, correspond to the control switch settings. They must appear in this order in memory and must consist of one 16-bit word each.

TABLE 3-11. MAP DATA CATEGORIES BY ID NUMBER

Item ID	Data Category
0	VORTAC - low
1	VORTAC - high
2	Obstructions
3	Lighted obstructions
4	Terminal area
5	Coastlines - 3 highest range scales
6	Coastlines - 2 lowest range scales
7	Lakes - 3 highest range scales
8	Lakes - 2 lowest range scales

TABLE 3-12. DISPLAY SELECTION VARIABLES

Data Category	Display Control Variable*
VOR Frequencies	IFREQ
TACAN Channels	ICHAN
Airways	IAIR1
Airway Headings	IAIR2
Airway Designators	IAIR3
Airway Mileages	IAIR4
Min. Enroute Alt.	IMEAS
VORTAC - Low	IVORL
VORTAC - High	IVORH
Obstructions	IOBST
Lighted Obstructions	ILOBS
Runway Configurations	IRNWY
Coastlines - 1.25, 5 n. mi./in.	ICST2
Coastlines - 10, 20, 40 n. mi./in.	ICST1
Lakes - 1.25 + 5	ILAK2
Lakes - 10, 20, 40	ILAK1

*Must appear in this order in memory. Values: 0 - Don't display category; 1 - display category. Also, must be SYM II variables.

4. 0 EMD SYSTEM INTEGRATION

INSTALLATION

The Hughes display generator has been installed in the flight simulation laboratory of the University of Illinois Aviation Research Center (see Figure 4-1).

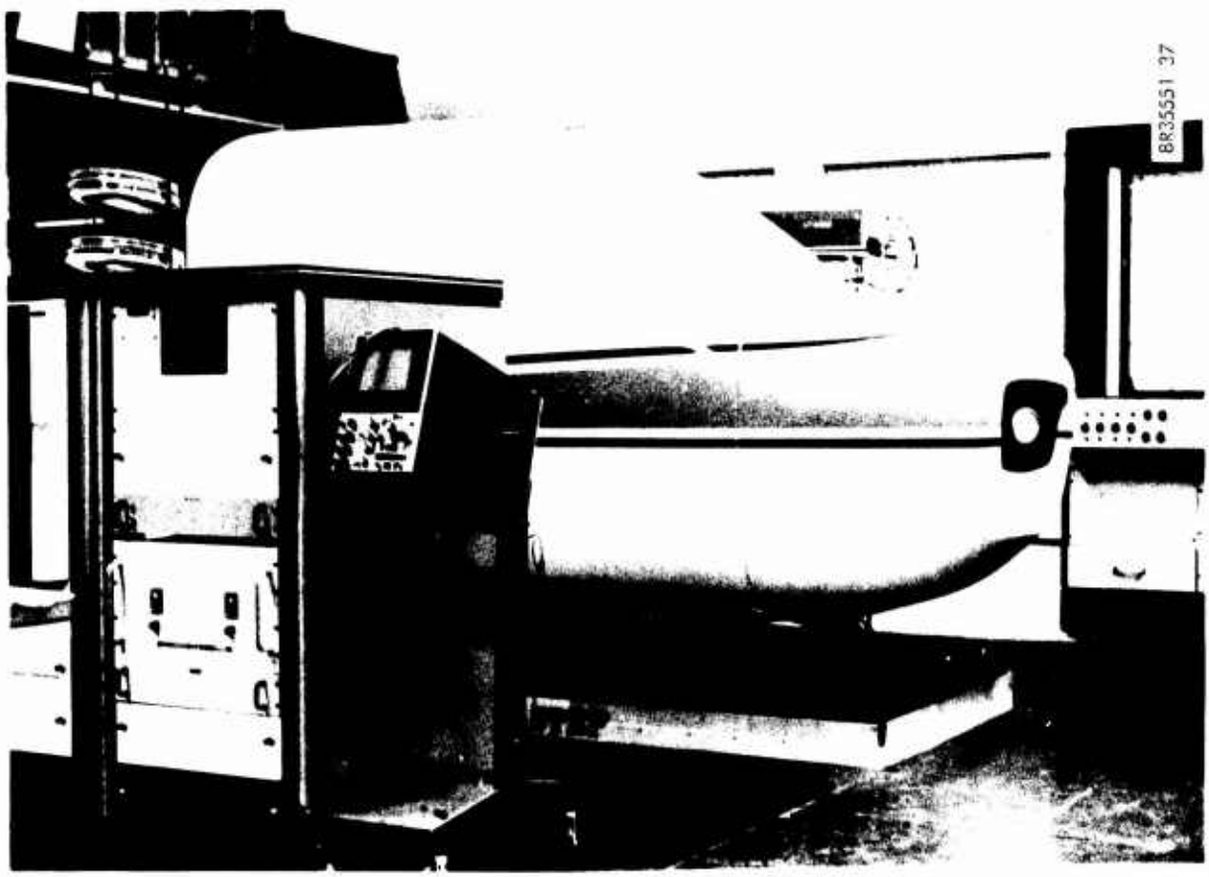


Figure 4-1. University of Illinois simulation laboratory.

A Raytheon 704 Central Processor in conjunction with a Raytheon 7002 DMA Controller provides the display generator with input data. The display generator is mounted in an external equipment rack near the computer and is connected to the DMA buss along with other high speed peripheral units.

The deflection amplifier and CRT assemblies are mounted in a flight simulator located in the laboratory. Cables of approximately twenty feet in length were provided to connect the deflection amplifiers and CRT to the display generator output.

A diagram of the University of Illinois simulation laboratory is shown in Figure 4-2.

INPUT REQUIREMENTS

The Raytheon 704 DMA buss control signal are shown in Figure 4-3. SWRT- and SRDE- are clock signals which are used to synchronize the display generator interface to the DMA buss.

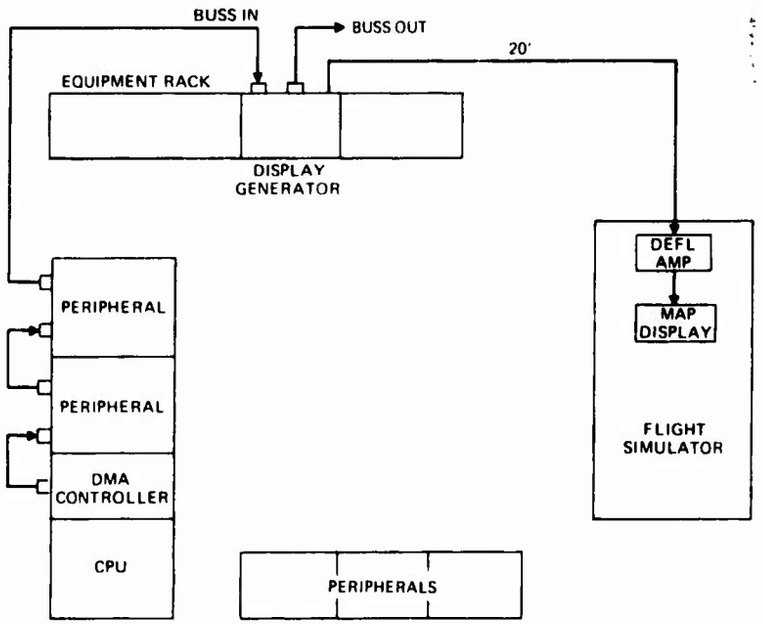


Figure 4-2. University of Illinois flight simulation laboratory.

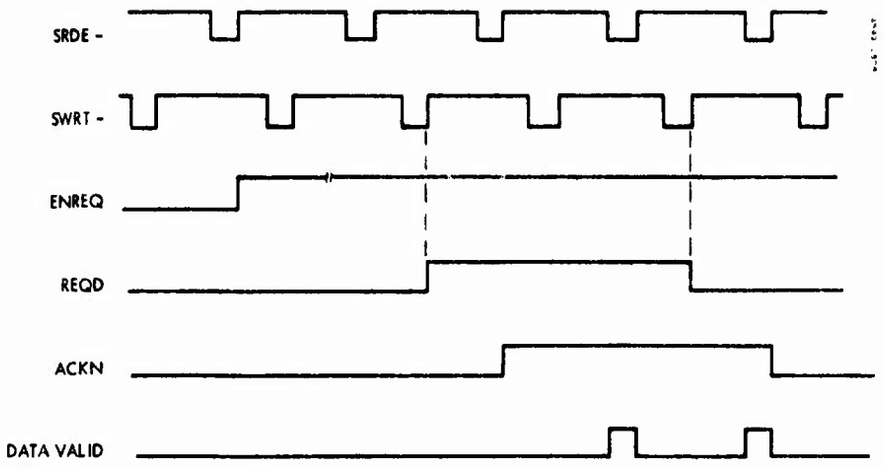


Figure 4-3. Raytheon 704 DMA buss signals.

When the display list is ready for transfer to the display generator, ENREQ goes low. The interface can then request data by placing a "one" on the REQD line. ACKN goes high during the time when the data transfer is taking place. ENREQ goes high after the last word in the display list has been transferred.

The DMA control signals are summarized in the following table.

SIGNAL NAME	DESCRIPTION
SRDE-, SWRT-	CPU clock signals used to synchronize the display generator interface to the DMA buss.
ENREQ	DMA channel enable. Allows external equipment to request memory cycles.
REQD	Memory cycle request initiated by the display generator.
ACKN	Memory cycle request acknowledged by CPU.
UN20, 21, 22	Unit address lines. Selects the display generator for data output.

OUTPUT CHARACTERISTICS

The four BNC coax connectors located on the rear panel of the display generator are the X, Y and Z axis outputs. The Z axis is controlled by

a video signal to produce various levels of brightness and a blanking signal to blank the display between line segments. The display generator output stage will drive 75 coax cables up to 20 feet in length.

OUTPUT VOLTAGE LEVELS

Video	±2.5 volts
Unblanking	0.4 volt blanks, +3 volts unblanks
X and Y Deflection	±2.5 full screen

POWER REQUIREMENTS

Display Generator	120 VAC, 60 or 400 Hz. 120 watts
Deflection Amplifier and CRT	120 VAC, 60 Hz. 600 watts average, 1200 watts maximum

OPERATING PROCEDURE

Figure 4-4 illustrates the cables necessary to connect the display generator to the Raytheon 704 computer. The following procedure should be used when operating the display generator.

1. Turn display generator and remote power switches to OFF position.
2. Connect cables as in Figure 4-4.
3. Turn intensity control fully counterclockwise.
4. Set up display list in computer.
5. Display generator power to ON position.
6. Deflection amplifier power to ON position. (Remote power switch).
7. Turn up intensity and adjust focus control.
8. Adjust size and position of display with controls located on the deflection amplifier.

EMD SYSTEM PERFORMANCE

Introduction

Correct operation and satisfactory performance of the EMD system hardware and software was verified by the performance of an approved

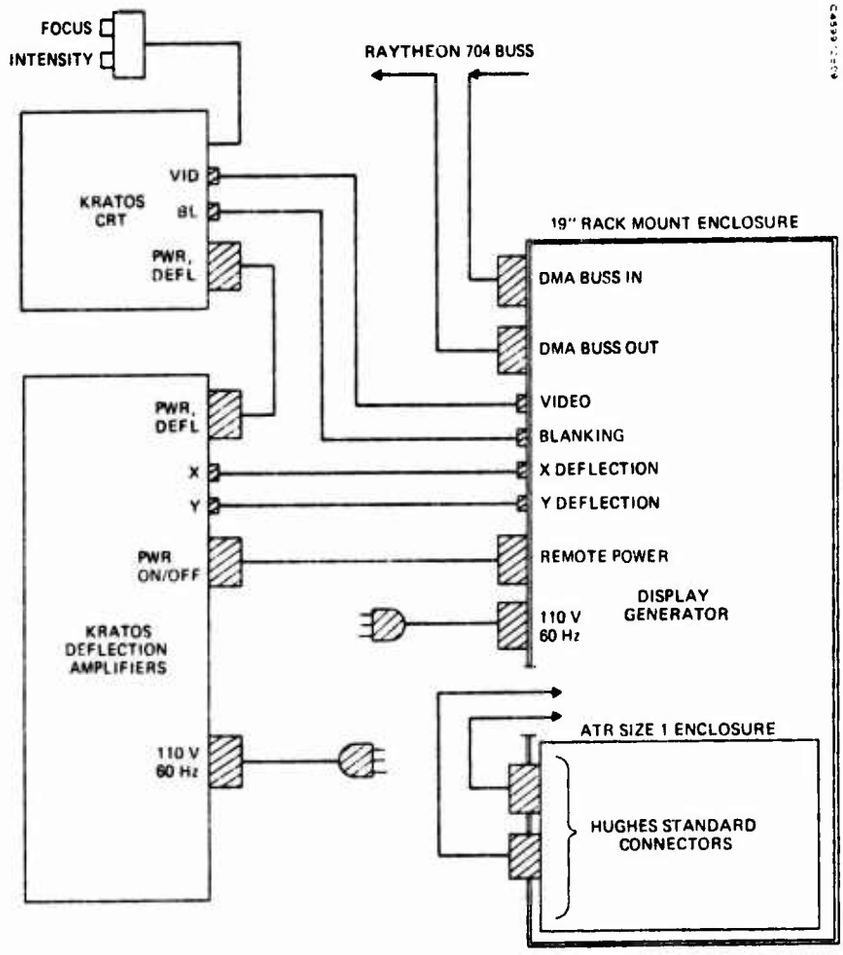


Figure 4-4. Display generator cables.

acceptance test procedure. The procedures demonstrated the flexibility and performance of the computer-driven display device required for electronic map displays. Special test computer programs were used which exercise each display system feature (such as absolute and relative chaining and symbol rotation) and graphically verify the operating speed of the device.

Deliverable software included the following routines:

1. Hardware Test Program - This program exercises all hardware display instructions to generate a special test pattern and a static map display.
2. Display Utility Routines - The routines facilitate programmer utilization of the display generator capabilities.

3. Data Retrieval, Category Processor, Map Data Pre-Processor - These routines are major elements of a dynamic electronic map display system.

Non-deliverable software has been written to exercise the Data Retrieval and Category Processor routines (which are not stand-alone programs) so that their correct operation can be demonstrated. This software will subsequently be referred to as the System Exerciser (SX). Further non-deliverable software was written to demonstrate performance characteristics such as writing speed and low system overhead. This software will be referred to as performance demonstration programs (PDP).

The figures contained in this section are actual photographs of the EMD display presentations taken at the time of EMD system performance verification.

Hardware Acceptance

Correct operation and satisfactory performance of all display generator functions was verified through the use of a special test pattern, Figure 4-5, in which the symbology reflects the use of all display generator instructions. Each instruction was used at least once in the test pattern display list. The hardware instructions verified were:

LOAD BRIGHTNESS
LOAD HORIZONTAL BIAS (H_{OO})
LOAD VERTICAL BIAS (V_{OO})
LOAD HORIZONTAL POSITION (H_O)
LOAD VERTICAL POSITION (V_O)
LOAD H_O WITH RESET
LOAD V_O WITH RESET
ADD H_O
ADD V_O
LOAD SINE
LOAD COSINE
LOAD VIDEO PATTERN
LOAD SEGMENT LENGTH
DRAW LINE

Figure 4-5. Display generator instruction vocabulary exercise.

DRAW DOT

DEFINE ABSOLUTE CHAIN (WITH ABSOLUTE DATA)

DEFINE RELATIVE CHAIN (WITH ABSOLUTE DATA)

DEFINE LIST

END OF DISPLAY

The correct operation of the video brightness instruction was verified by displaying a gray wedge, Figure 4-6, generated with a number of horizontal lines in which the intensity bit code varies in steps from 000 to 111.

Display writing speed was verified through the use of a special test pattern, Figure 4-7 and 4-8, which causes the display generator to draw continuously for 16 milliseconds. A minimum of 32,000 display elements

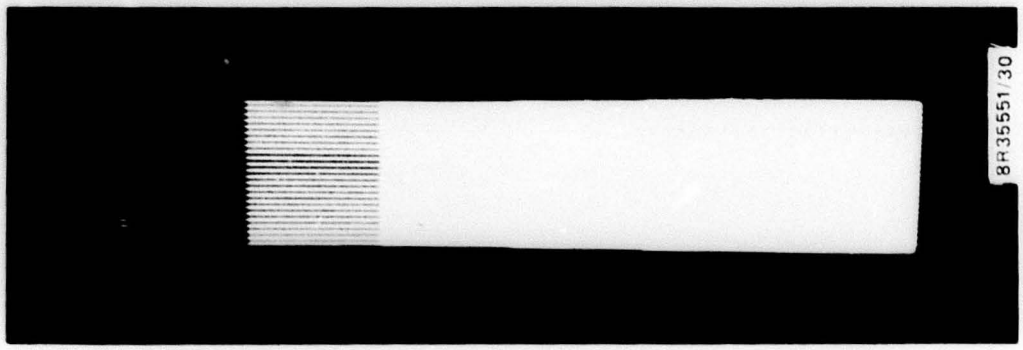


Figure 4-6. Display generator shades of gray presentation.

Figure 4-7. Display generator writing speed - relative chain encoding

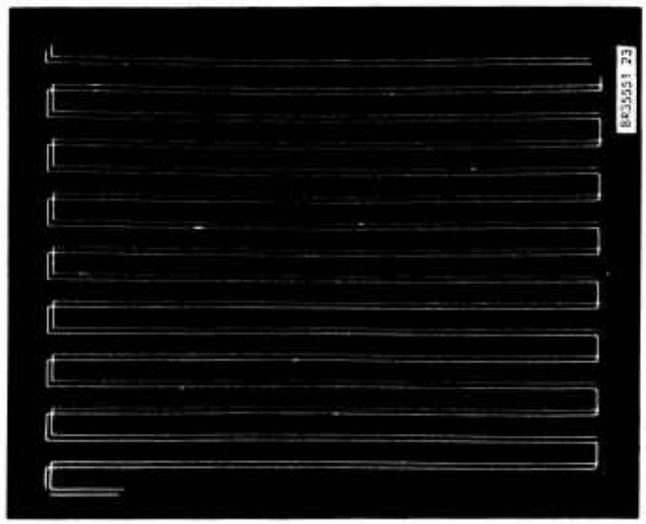


Figure 4-8. Display generator writing speed - absolute chain encoding.

was written in the 16 millisecond period, corresponding to a minimum writing rate of 13,000 inches per second on a seven-inch display. This procedure was used to verify writing speed for relative and absolute chains. A test pattern Figure 4-9, was used to demonstrate that "overhead" items (i.e., those display instructions such as LOAD H_O and V_O which do not directly cause display writing to occur) are not the limiting factor in terms of display capacity. Seven hundred and twenty short vectors were displayed. This number was limited by the size of the display buffer in the computer and not by the speed of the display generation hardware.

Verification of operation of the display generator for electronic map generation was accomplished by displaying a static map presentation Figure 4-10, in which coastlines and symbology (alphanumerics, special symbols) was present.

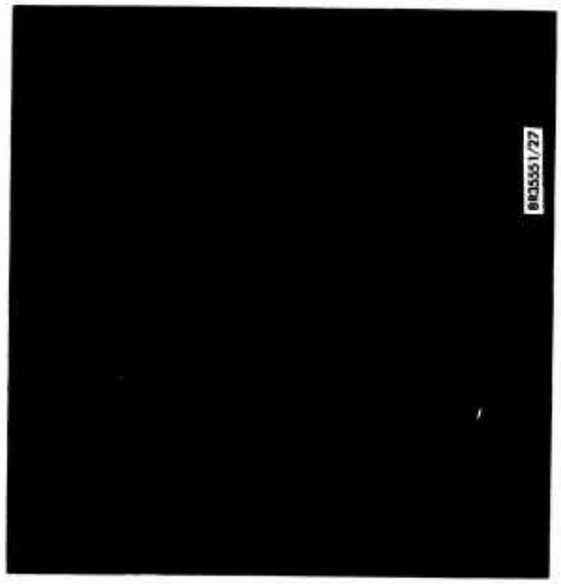


Figure 4-9. Display generator "low overhead", verification.

Software Acceptance

The deliverable software items are the display utility routines, the Hardware Test Program, the map data pre-processor, the data retrieval program, and the category processor.

The display utility routines are extensively used by the test program and the category processor. Correct operation of those programs was considered as adequate validation of the display utility routines.

The Hardware Test Program is used in the hardware acceptance procedure and its correct operation in that function was considered as verification.

Special, non-deliverable software was written to permit operation of the final three software items (map data pre-processor, data retrieval, and category processor) in an integrated interactive and dynamic mode. The following features were viewable on the display for verification of correct operation of the software. An aircraft symbol (arrow) placed at the center of the display oriented in the direction of aircraft heading. The aircraft was programmed to move in a square flight path to demonstrate selection of items from the data base, and scaling and location of the items on the

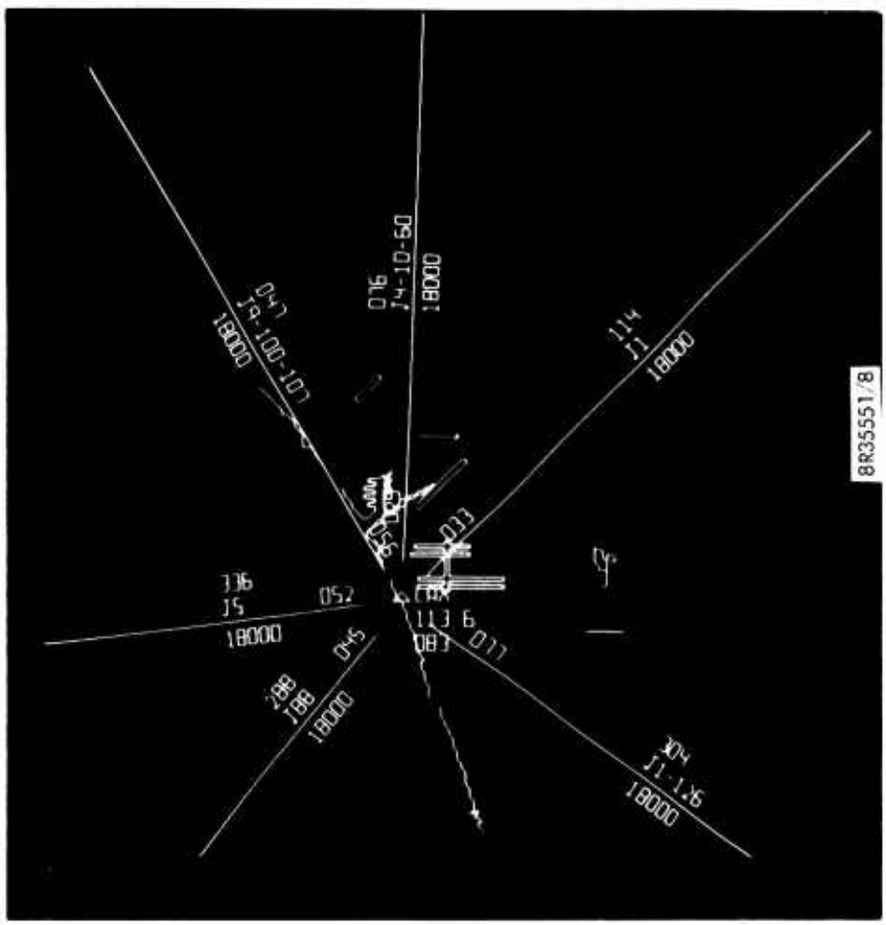


Figure 4-10. Electronic map generation.

display. A north up orientation was presented. The following data items were selectable through the computer teletype to demonstrate data retrieval, selection, location and formatting by the category processor: low and high altitude VOR stations, obstructions and lighted obstructions, runways, coastlines, lakes and rivers, VOR frequencies, airways, airway designators, airway headings, enroute mileages, minimum enroute altitudes, and TACAN channels. In addition, display scales were selectable from the computer teletype. Correct demonstration of these features verified that the data base was created by the map data pre-processor, the data was properly retrieved from bulk storage by the data retrieval program, and the display was being correctly formatted by the category processor.

The following series of photographs, Figures 4-11 through 4-16 are illustrations of the presentations on the display used to verify the correct operation of the software.

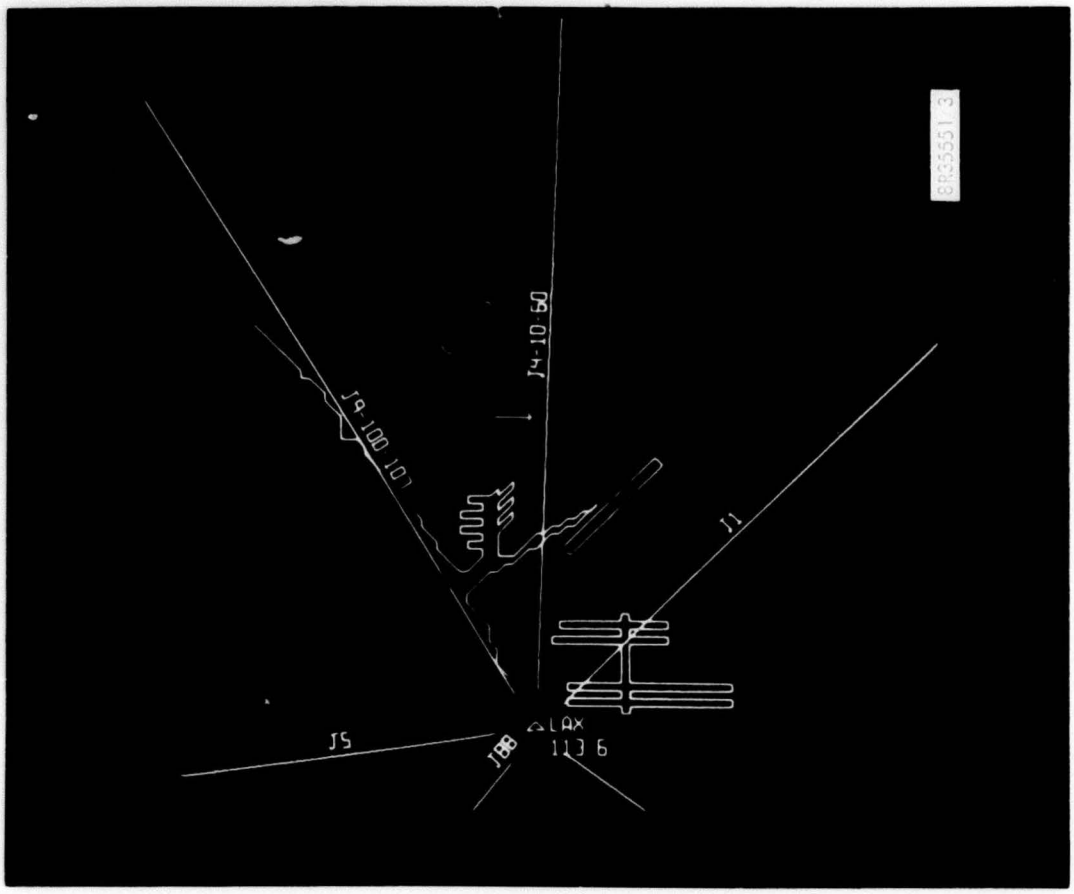


Figure 4-11. Terminal Area
Scale: 1 n.mi./inch
Presentation: VOR frequency
Airways
Airway designators

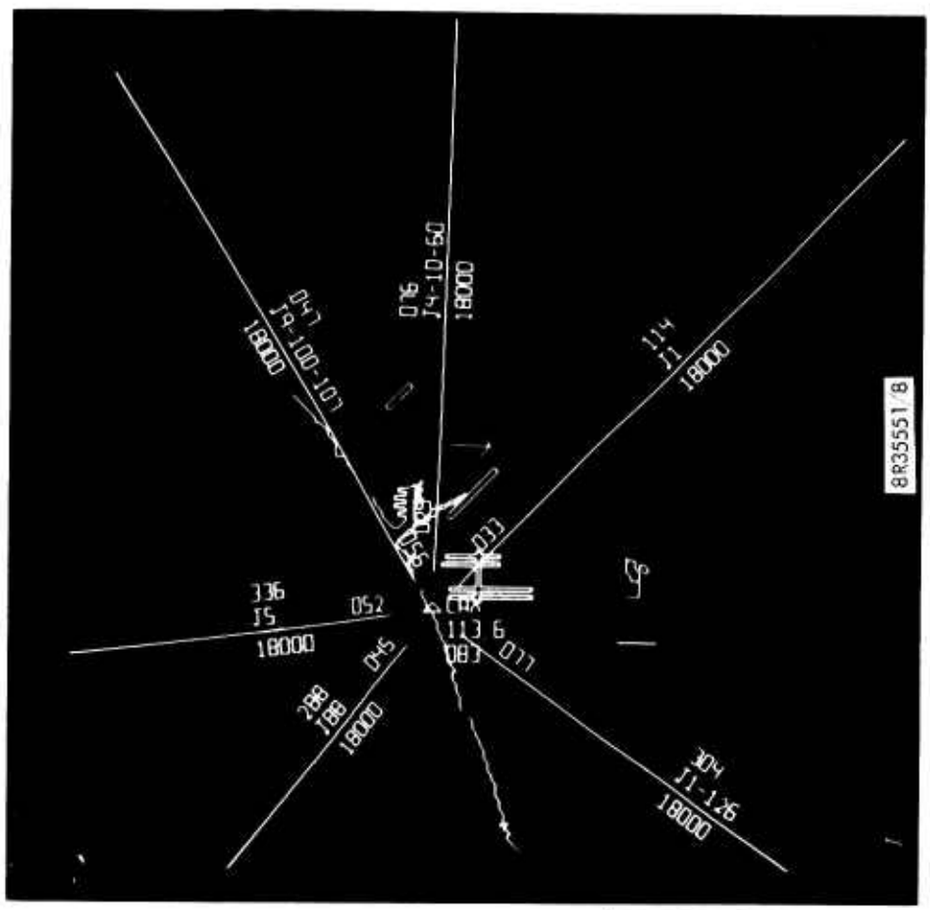


Figure 4-12. Terminal Area
Scale: 2 n.mi./inch
Presentation: VOR frequency
Airways
Airway designators
Enroute mileage to next VOR
Minimum enroute altitude

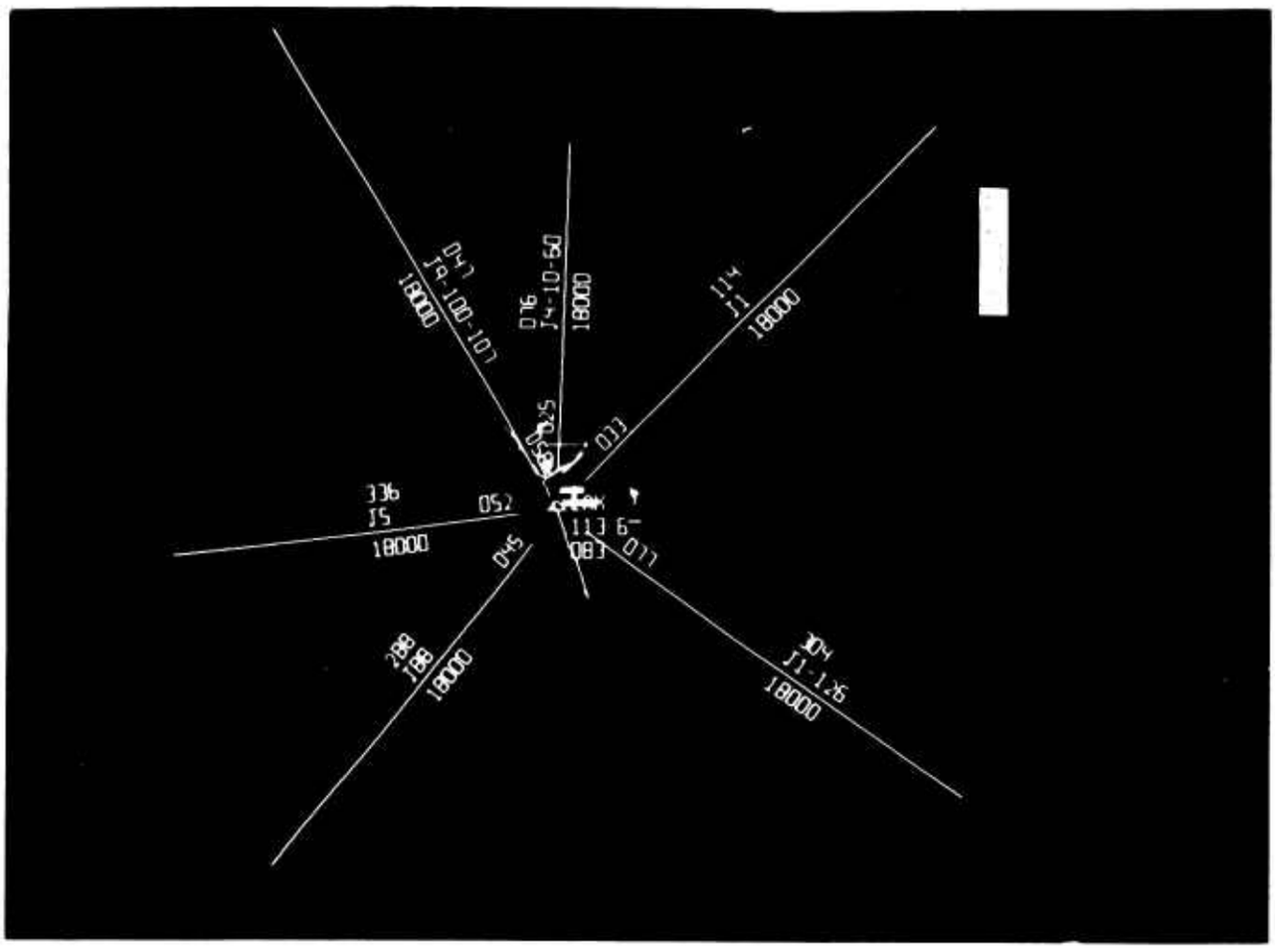


Figure 4-13. Terminal Area
Scale: 5 n.mi./inch
Presentation: VOR frequency
Airways
Airway designators
Enroute mileage to next VOR
Minimum enroute altitude
Heading
TACAN channel

gure 4-14. Low Altitude
Scale: 20 n.mi/inch
Presentation: VOR frequency
Lakes, Rivers
Obstruction

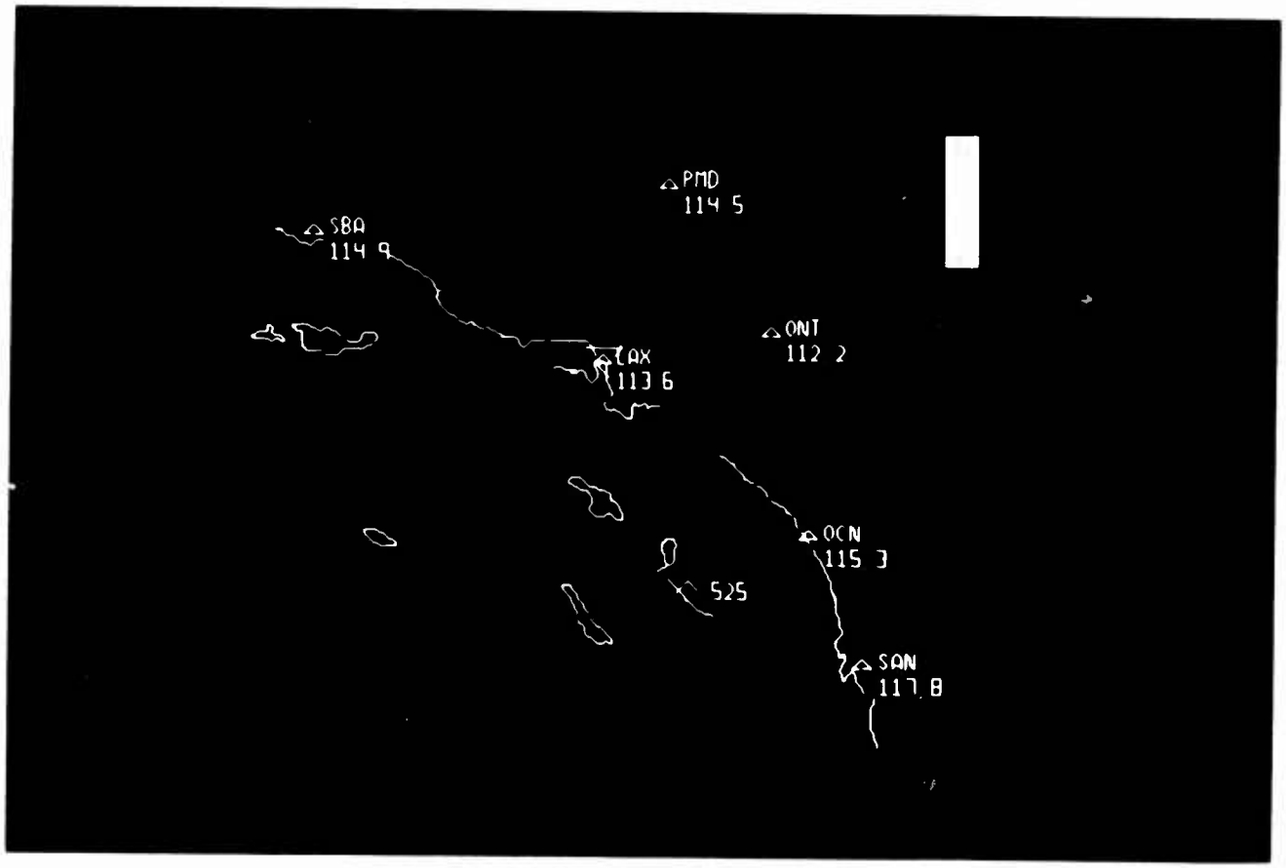


Figure 4-15. High Altitude
Scale: 20 n.mi./inch
Presentation: VOR frequency
Lakes, Rivers
Lighted obstruction

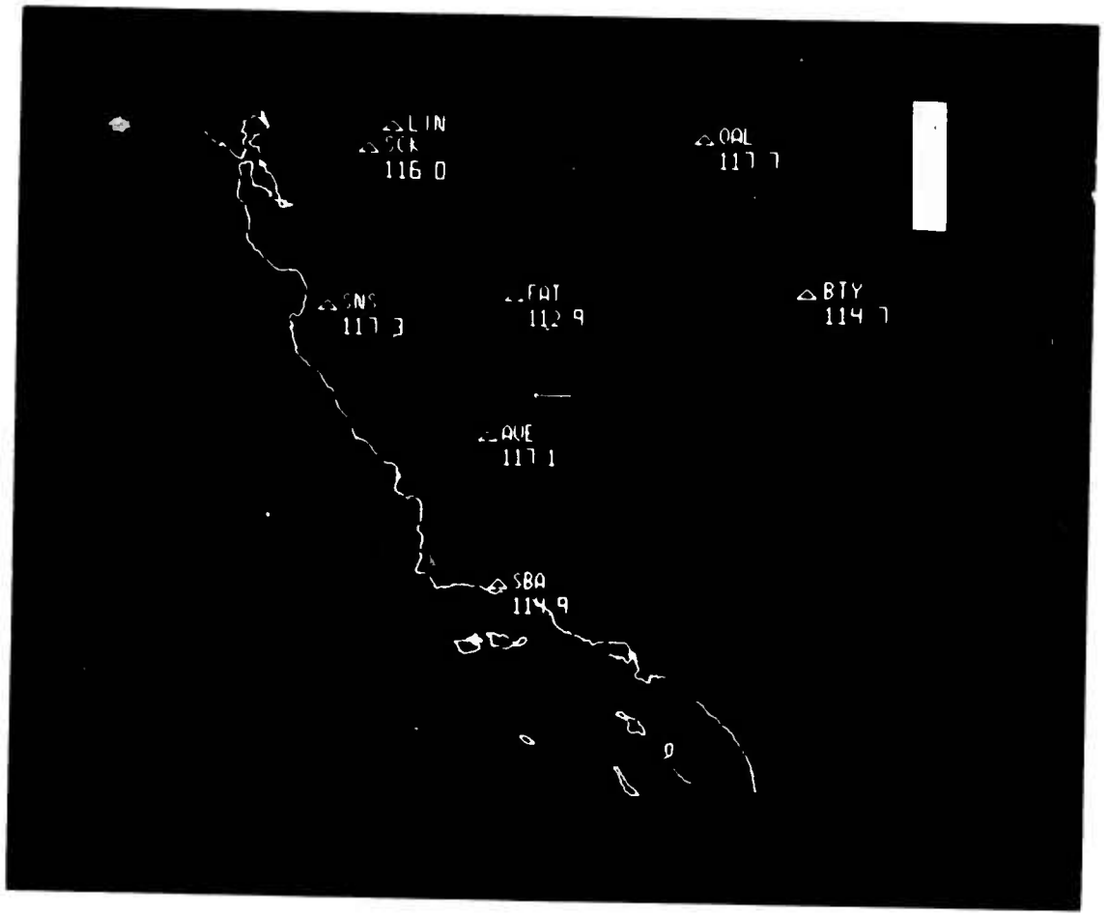
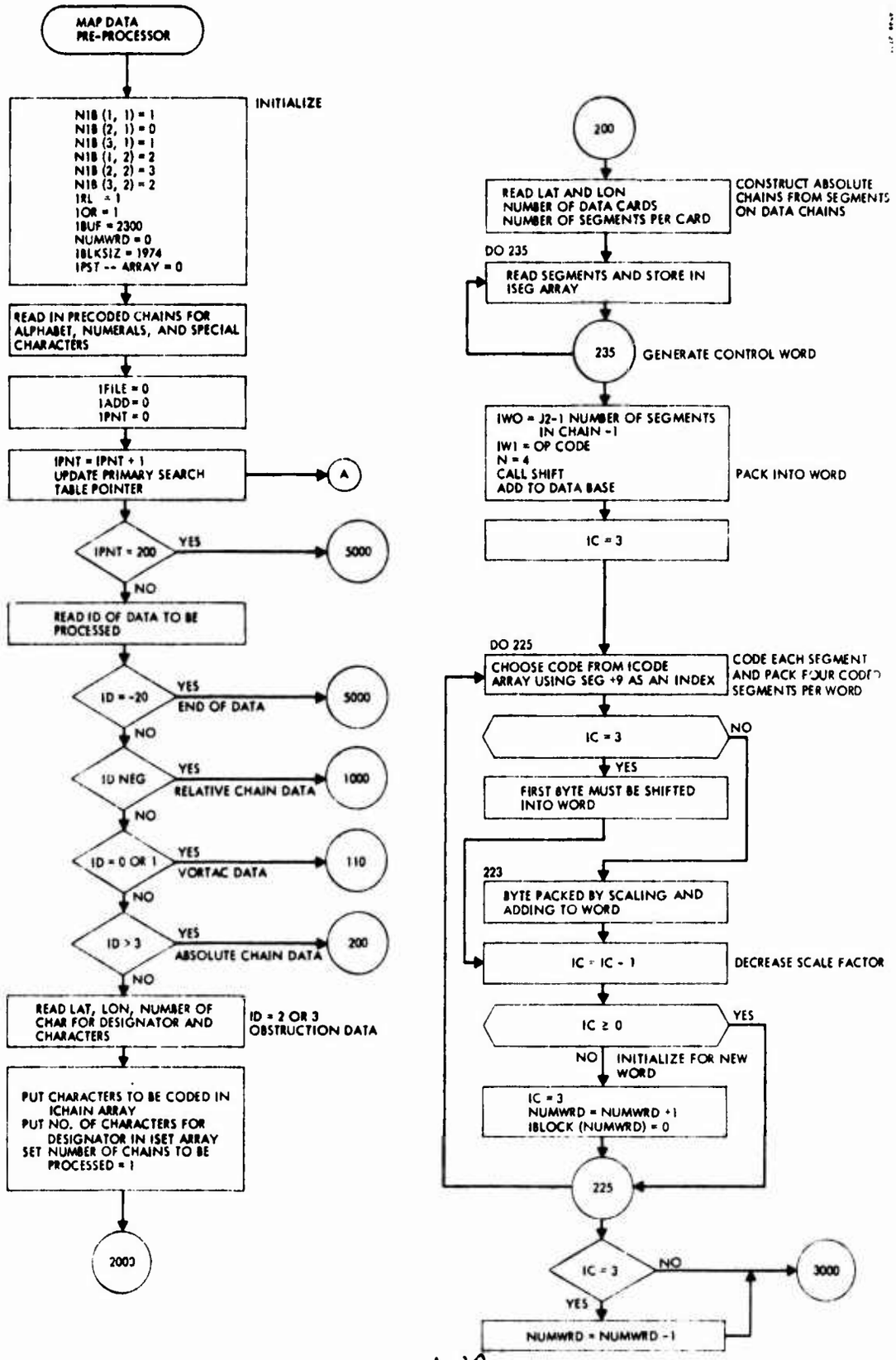


Figure 1-17. High Altitude
Scale: 30 n.mi. inch
Presentation: VOR frequency
Lakes, Rivers

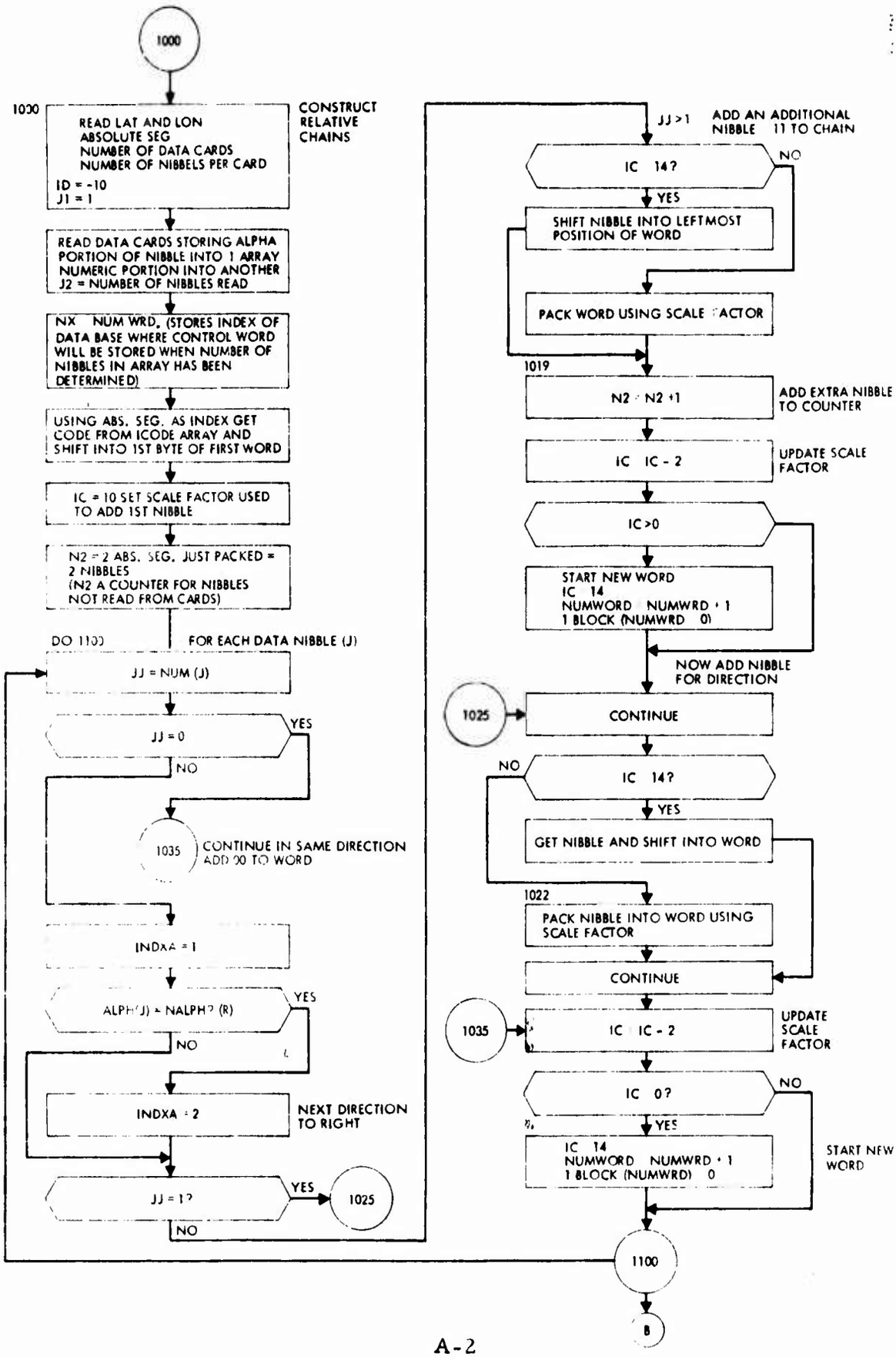
APPENDIX A

SOFTWARE FLOW CHARTS AND GLOSSARIES

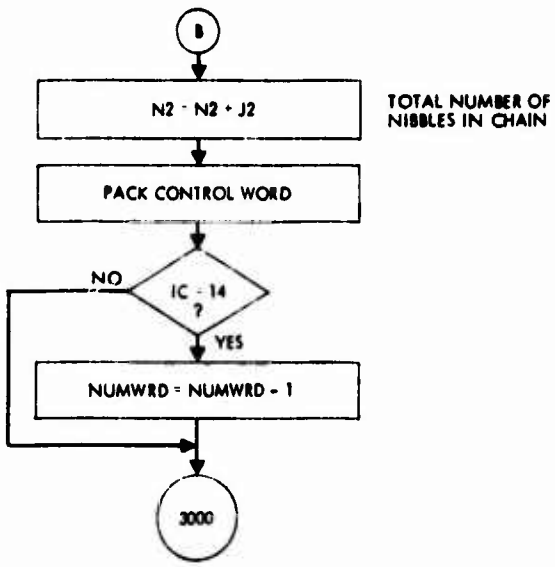
A.1 FLOW CHART FOR MAP DATA PRE-PROCESSOR PROGRAM

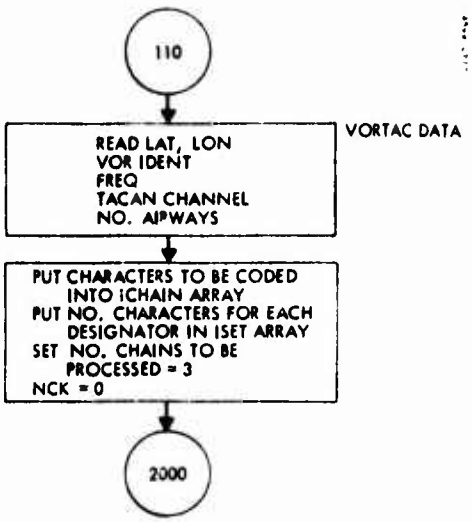


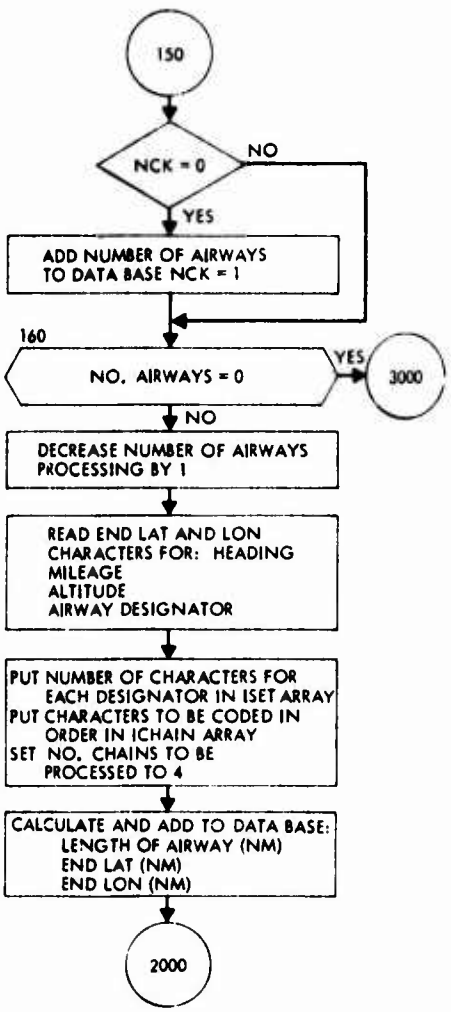
A.1 (continued)

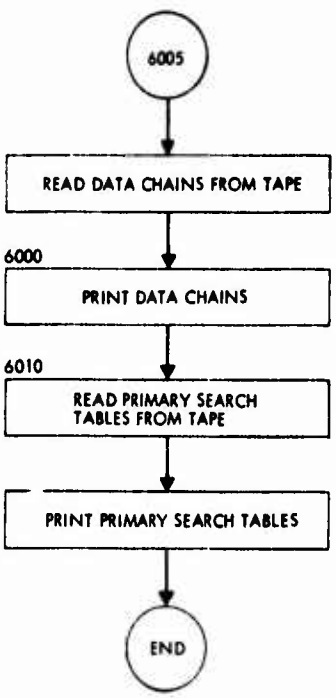


A. 1 (continued)

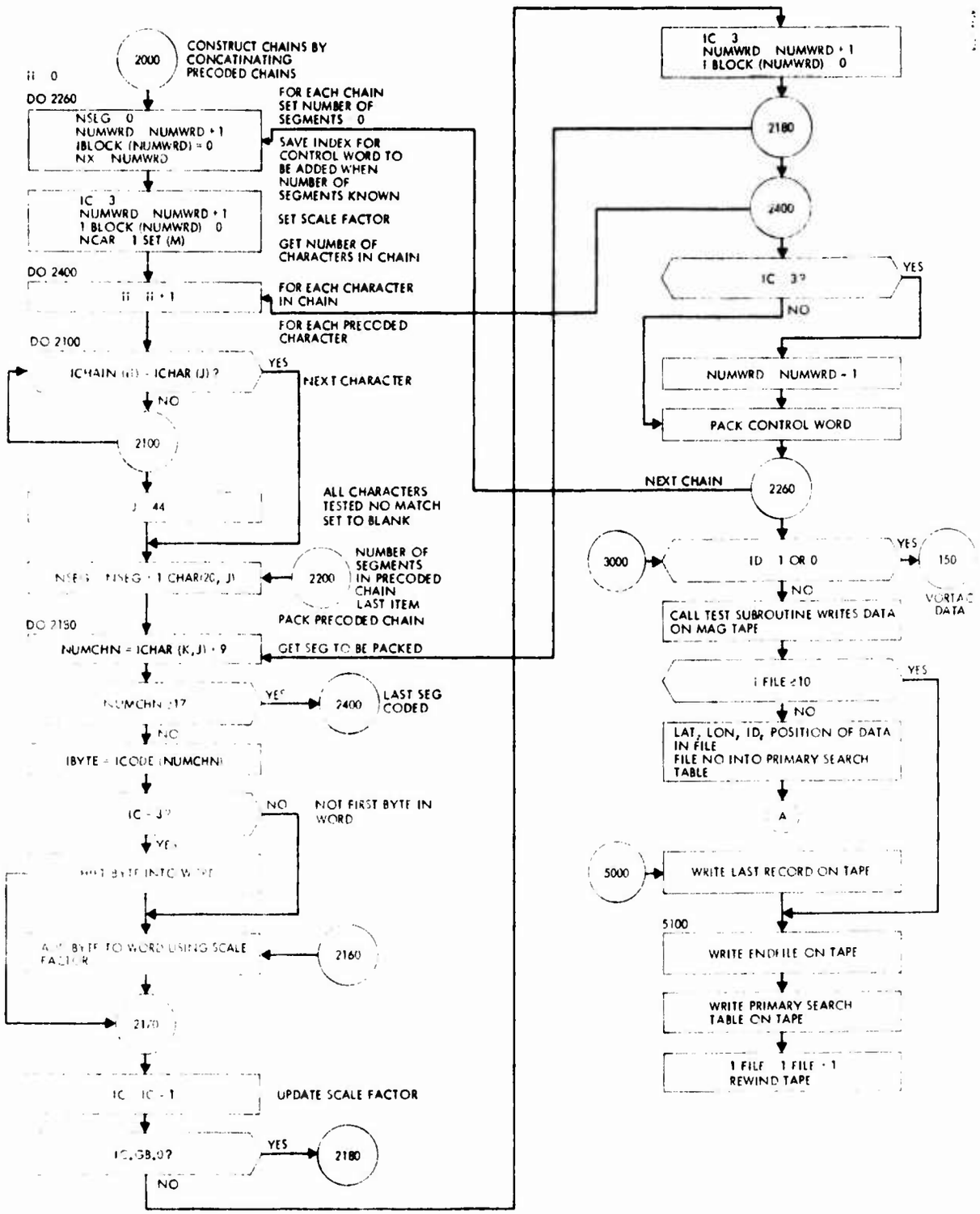








A. 1 (continued)



A.1.1 Preprocessor Glossary

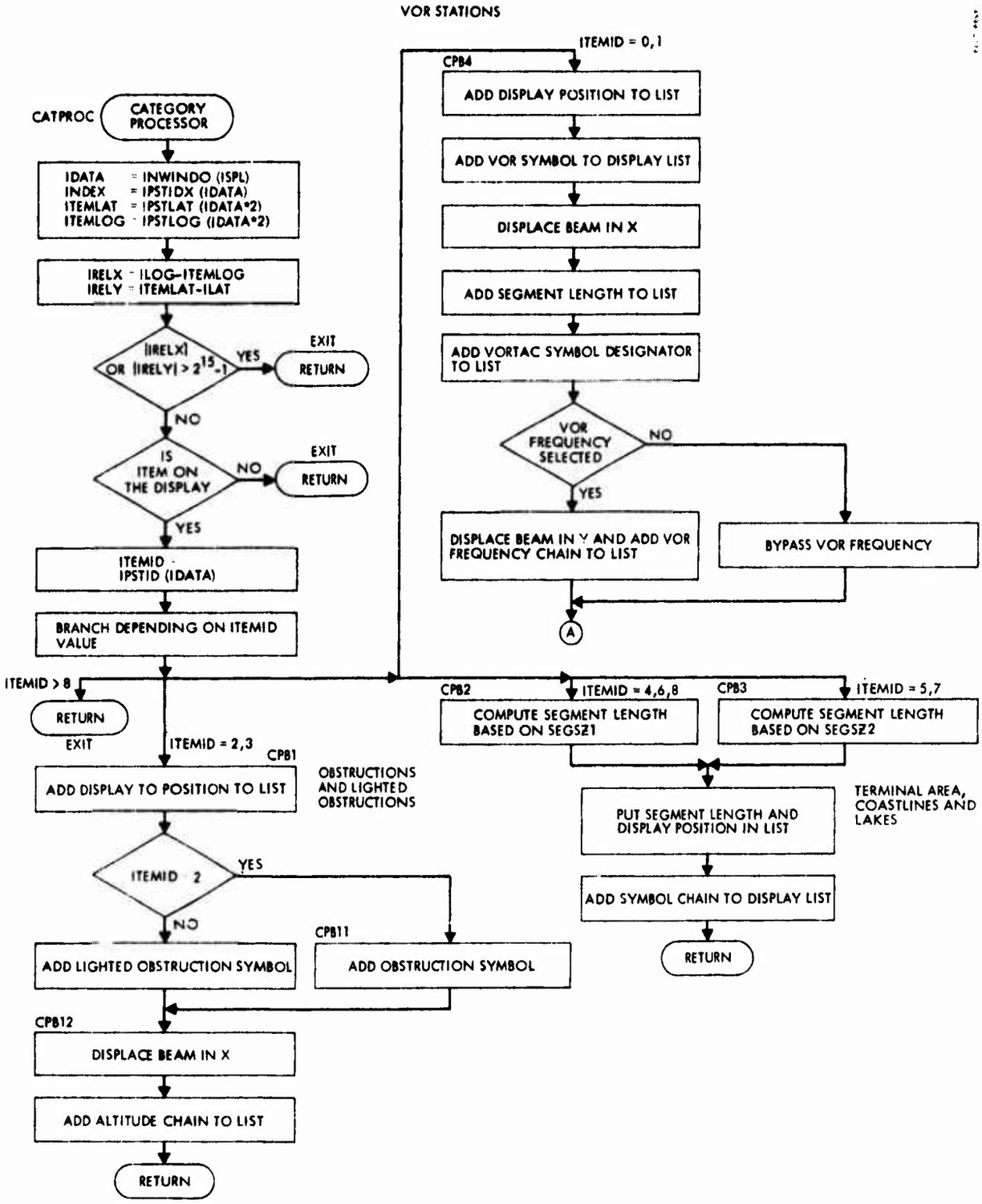
Name	Definition
D	Length of airway in nautical miles
DX	Length of latitude portion of airway vector (n.m.)
DY	Length of longitude portion of airway vector (n.m.)
IHDGA	Airway vector heading-characters to be displayed
IABSEG	The first (absolute) segment in a relative chain
IADD	Indicates position in file
IALPH	Alphabetic portion of relative data
IALTA	Altitude designator for airway
IBLKSIZ	Number of records written on mag tape at one time
IBLOCK	An array holding all data for the display
IBUF	Limit of IBLOCK array
IBYTE	Four bits of data to be packed into an "IBLOCK" data word
IC	A scale factor used in packing
ICHAIN	An array to holding characters to be converted to chains
ICHAN	TACAN channel
ICHAR	An array of coded chains representing alphabet numerals 0 - 9, and some special characters
ICOD	Hex value of absolute code Video bit is first 0 - 7 blanked 8 - 15 to be displayed
ID	Indicates type of data to follow 0 and 1 VORTAC 2 and 3 obstructions 4 - terminal areas 5 and 6 coastlines 7 and 8 lakes
IDES	Airway designator - characters to be displayed
IDESIG	Airway designator - characters to be displayed

Name	Definition
IDICT	An array of characters used to identify an input character and provide an index for accessing corresponding coded chain
IFILE	Mag tape file number = number of files - 1 on tape at end of program
IFREQ	VORTAC frequency - characters to be displayed
INAIR	Counter for number of VORTAC airways left to be processed
INDXA	An index used in encoding relative chain
IPNT	A pointer to primary search tables
IPSTAD	A primary search table indicating location of data within a file
IPSTFL	A primary search table indicating in which file data is located
IPSTID	A primary search table indicating 10 - or type of data
IPSTLN	A primary search table indicating longitude of data
IPSTLT	A primary search table indicating latitude of data
IOR	Argument for shift subroutine = 0 if no inclusive or = 1 if IFIW1 to be ORed with IW0
IRL	Argument for shift subroutine = 0 if shift is to be right = 1 if shift is to be left
ISEG	An array containing absolute chain segments read from cards
ISET	An array containing the number of characters to be processed for each chain - using characters in ICHAIN array
IW0	Argument for shift subroutine WORD to be shifted
IW1	Argument for shift subroutine WORD to be inclusively ORed with IW0
J2	Number of segments in chain to be coded

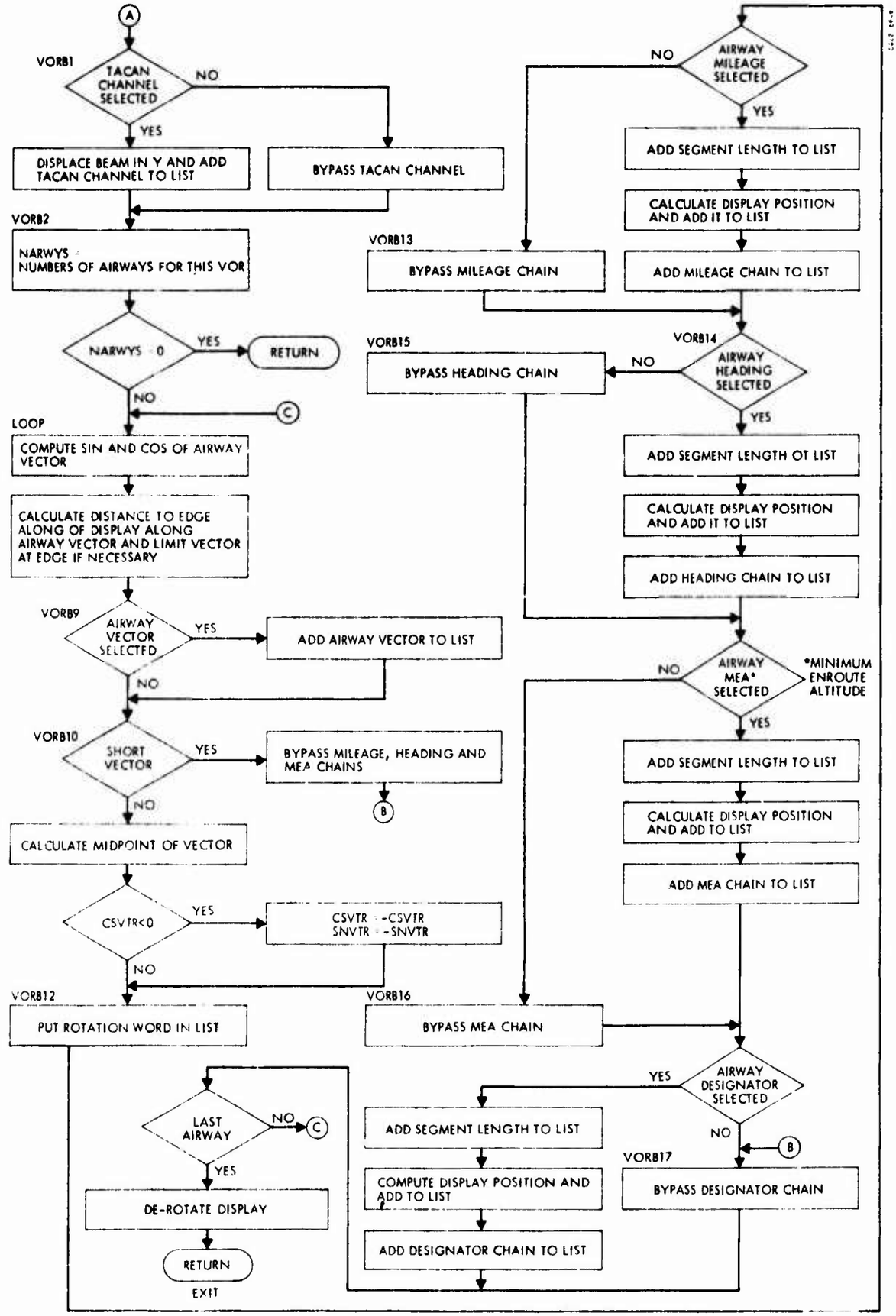
Name	Definition
LATD	Latitude (DEG)
LATM	Latitude (MIN)
LATS	Latitude (SEC)
LNGTH	Length of airway vector in nautical miles Q = 8
LOND	Longitude (DEG)
LONM	Longitude (MIN)
LONS	Longitude (SEC)
MLGA	Airway mileage — characters to be displayed
N	Argument for shift subroutine Number of bits to shift W01
NC	Temporary Index
NAIR	Number of airways associated with a VORTAC facility
NALPH	Alphabetic portion of relative chain input data
NCARDS	Number of cards to be input
NCHAINS	Number of chains to be processed
NCK	= 1 number of airways for VORTAC facility has been added to IBLOCK data array
NDLAT	Length of latitude portion of airway vector
NDLON	Length of longitude portion of airway vector
NIB	A data array used to encode relative chain data
NSEG	Number of segments — 1 in chain
NSPC	Number of segments per card
NUM	Numeric portion of relative chain input data
NUMCHIN	A pointer used in encoding absolute chains

Name	Definition
NUMWRD	An index for IBLOCK array
SHIFT	A subroutine for shifting and ORing data shift (IW0, IW1, IRL, IOR, N)
NX	A pointer for IBLOCK array so data may be entered in proper position after it has been computed.

A.2 FLOW CHART FOR DATA RETRIEVAL PROGRAM



A.2 (continued)



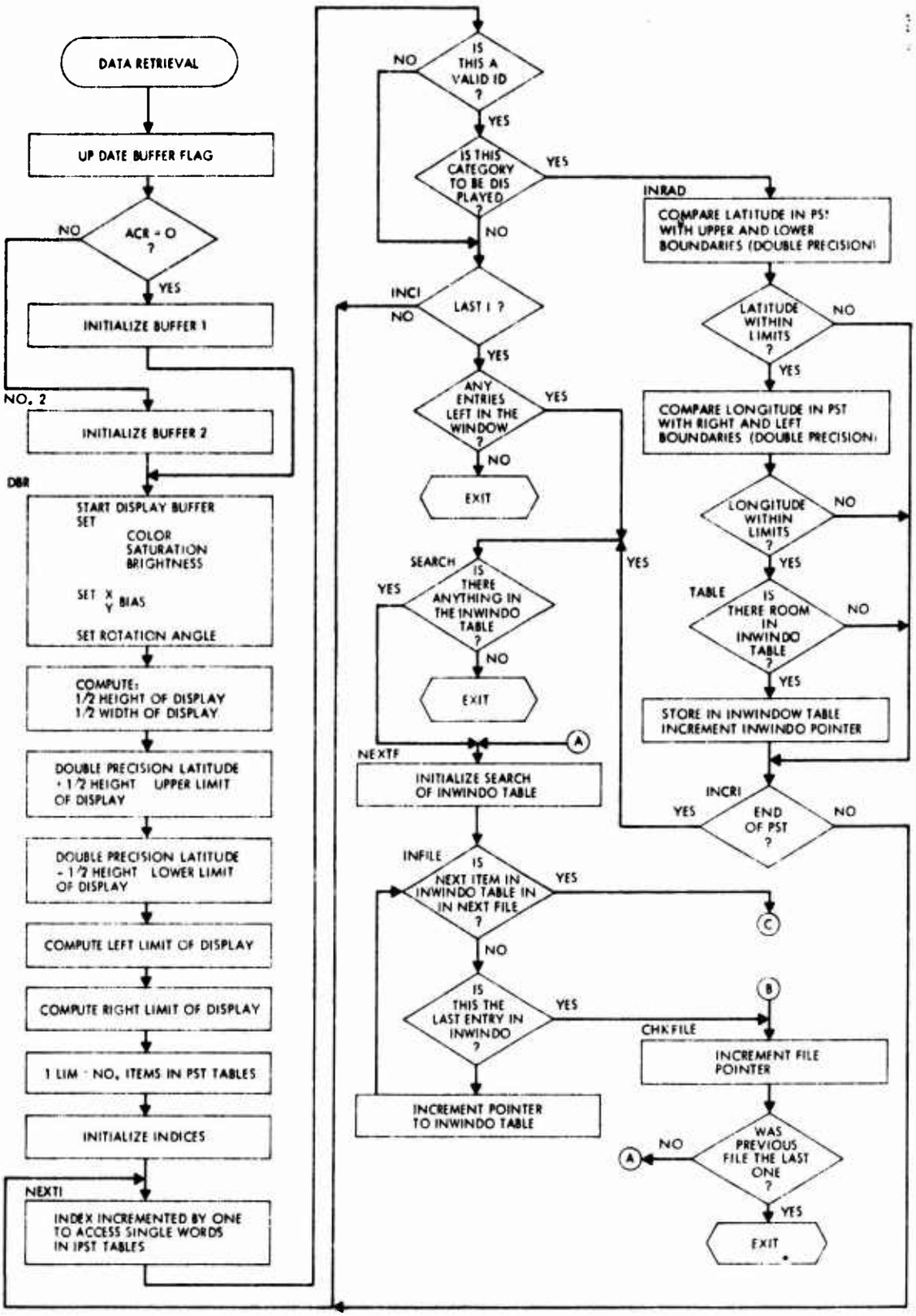
A.2.1 Data Retrieval Program Glossary

Name	Definition
AIRCRAFT	Data used to generate aircraft symbol
BRIT	Brightness level of display
BUFFLG	Flag used to determine which display buffer to initialize
DAD	Double precision fixed point add routine
DBIAS	Routine for biasing display
DBRIT	Routine for establishing color, saturation and brightness
DDSET	Routine that sets up control words for display of current buffer
DINIT	Routine that initializes display list
DISPSZ	Display diameter
DLD	Double precision load routine
DST	Double precision fixed point store routine
DSUB	Double precision fixed point subtract routine
DSYMB	Routine to add pre-defined symbol to display list
DWHER	Routine that returns index of next item in display list
I	Pointer to primary search tables
IC	Display color
ID	Data category
IDLIM	Highest valid ID number
IFILE	Pointer to disc file number
ILATD	Display latitude lower limit
ILATU	Display latitude upper limit
ILIM	Number of entries in primary search tables

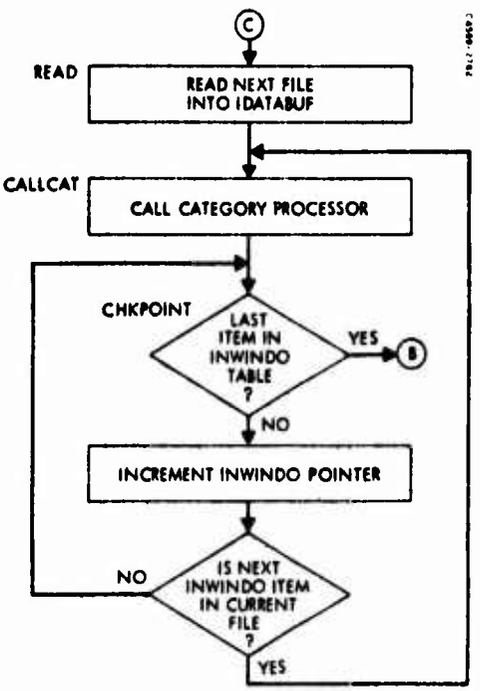
Name	Definition
ILOGL	Display longitude left limit
ILOGR	Display longitude right limit
INWINDO	Table of indices pointing to data within current display radius
IPNT	Pointer to INWINDOW table
IPSTID	Primary search table of category ID's
IPSTLAT	Primary search table of category latitudes
IPSTLON	Primary search table of category longitudes
IPSTSZ	Number of entries in PST tables
LAT1	Most significant half of latitude double word
LAT2	Least significant half of latitude double word
LATSR	One-half display height
LOC	Location of last entry to display buffer
LOG1	Most significant half of longitude double word
LOG2	Least significant half of longitude double word
LONSR	One-half display width
MAXID	Largest valid ID number
NAC	Number of segments − 1 in aircraft chain
NFILE	Number of data files stored on DISC
NSIN	Sin of display rotation angle
NCOS	Cos of display rotation angle
PNTLIM	Number of data items to be displayed
SECPNM	Seconds per nautical mile
SINHDG	Sin of aircraft heading

Name	Definition
COSHDG	Cos of aircraft heading
SIZECK	Display buffer limit
TRKSECT	Hex words containing track and sector
UNIT	File size
XBIAS	Display bias X
YBIAS	Display bias Y

A.3 FLOW CHART FOR CATEGORY PROCESSOR PROGRAM



A. 3 (continued)



A.3.1 Category Processor Glossary

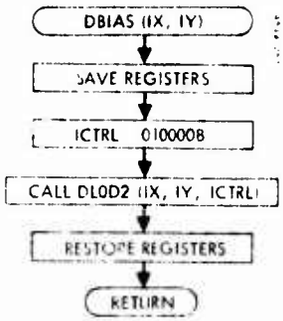
Name	Definition
AIRWY	VOR airway loop counter
BLNK	Amount of airway vector to blank
CHNAD	Absolute address of chain to be added to display list
CSVTR	Cosine of airway vector angle
DELX	X component of airway vector
DELY	Y component of airway vector
DESSZ	Airway designator pneumonic segment length
EQ0	Constant = 0
EQ1	Constant = 1
EQ2	Constant = 2
EQ512	Constant = 512
EQ1024	Constant = 1024
EQ1Q7	Constant = 1 at Q7
EQ3600	Constant = 3600
EQFF	Mask to extract last 8 bits
EQ9700	HEX constant 9700
HDGDEL	Radial distance from heading pneumonics to VOR symbol
HDGSZ	Airway heading pneumonic segment length
IAIR1	VOR airway vector selection variable
IAIR2	VOR airway heading selection variable
IAIR3	VOR airway designator selection variable
IAIR4	VOR airway mileage selection variable

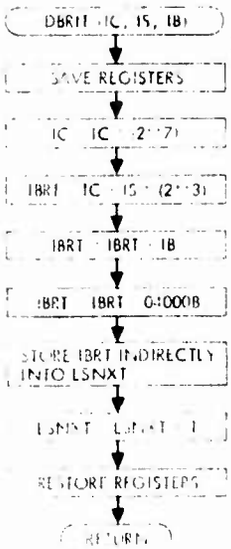
Name	Definition
ICHAN	TACAN channel number selection variable
IDATA	Absolute index of item in PST tables
IDATABUF	Buffer containing MAP data
IDELX1	Standard X offset for alphanumerics
IDELX2	Perpendicular distance from airway heading pneumonics to airway vector
IDP1	Double precision dummy variable
IFREQ	VOR frequency selection variable
ILAT	Aircraft latitude
ILOB	Lighted obstruction symbol chain
ILOG	Aircraft longitude
ILSZ	Standard alphanumeric segment length
IMEAS	VOR minimum enroute altitude selection variable
INDEX	Index of data item in IDATABUF buffer
INWINDO	Table of indices to PST table which points to items to be displayed
IOB	Obstruction symbol chain
IPSTID	Table of ID numbers of every item in data base
IPSTIDX	Table of indices of every item in each disk file
IPSTLAT	Table of latitudes of every item in data base
IPSTLOG	Table of longitudes of every item in data base
IRELX	Longitude of display item relative to display center
IRELY	Latitude of display item relative to display center
ISCALE	Display range scale

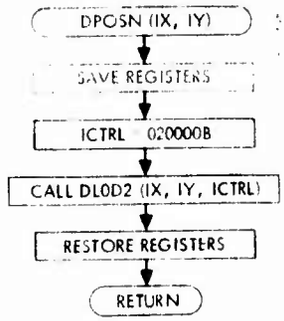
Name	Definition
ISFL	Index used to search through INWINDO table
ITEMID	ID number of display item
ITEMLAT	Latitude of display item
ITEMLOG	Longitude of display item
IVOR	VOR symbol chain
IXDISP	X-coordinate position of display item on display
IYDISP	Y-coordinate position of display item on display
IY1	Vertical display offset for displaying VOR frequencies and TACAN channel numbers
LATSR	Number of seconds in half the display in vertical dimension
LONSR	Number of seconds in half the display in horizontal dimension
MAXDU	Maximum display units in half the display
MAXID	Maximum valid ID number
MEADEL	Perpendicular distance from airway MEA pneumonics to airway vector
MEASZ	Airway MEA segment length
MILDEL	Perpendicular distance from airway mileage pneumonics to airway vector
MILSZ	Airway mileage segment length
NARWYS	Number of airways for a particular VOR station
NEG0	Mask with '1' in sign bit
NEG1	Constant − 1
NLOB	Number of words in lighted obstruction chain
NOB	Number of words in obstruction chain

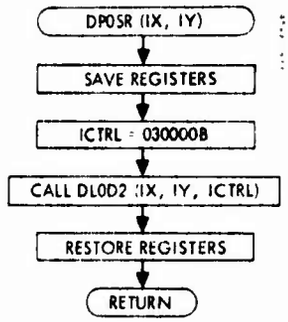
Name	Definition
NVOR	Number of words in VOR symbol chain
NWDS	Number of words in chain to be added to display list
SEGSZ1	Segment size of chains of 2 lowest range scales
SEGSZ2	Segment size of chains of 3 highest range scales
SNVTR	Sin of airway vector angle
TEMP1 – TEMP9	Temporary storage locations
XHDG	X-coordinate position of airway heading on airway vector
XMID	X-coordinate of midpoint of airway vector
YHDG	Y-coordinate position of airway heading on airway vector
YMID	Y-coordinate of midpoint of airway vector

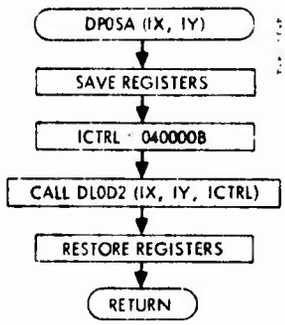
A.4 FLOW CHARTS FOR DISPLAY UTILITY SOFTWARE

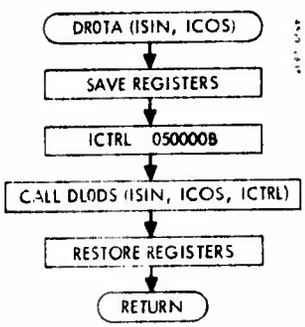


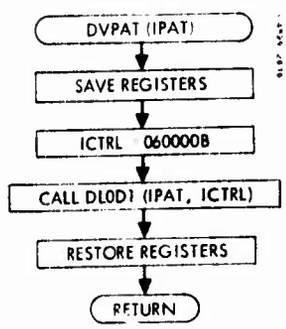


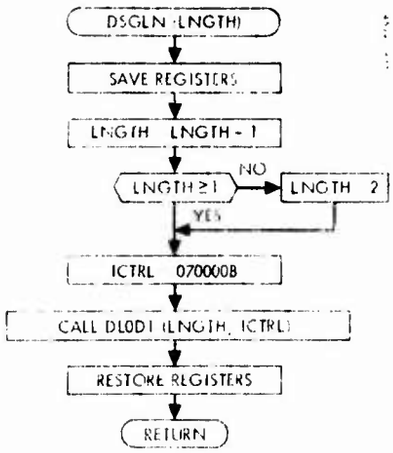


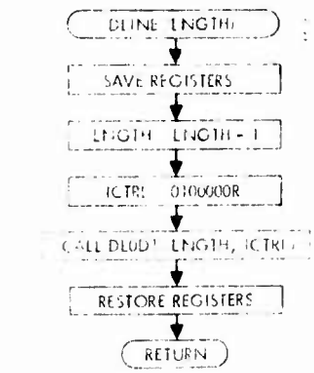

DPOSR (IX, IY)
SAVE REGISTERS
ICTRL = 030000B
CALL DLOD2 (IX, IY, ICTRL)
RESTORE REGISTERS
RETURN

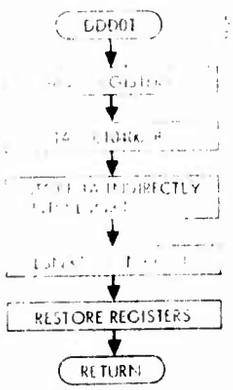

DPOSA (IX, IY)
SAVE REGISTERS
ICTRL = 040000B
CALL DLOD2 (IX, IY, ICTRL)
RESTORE REGISTERS
RETURN

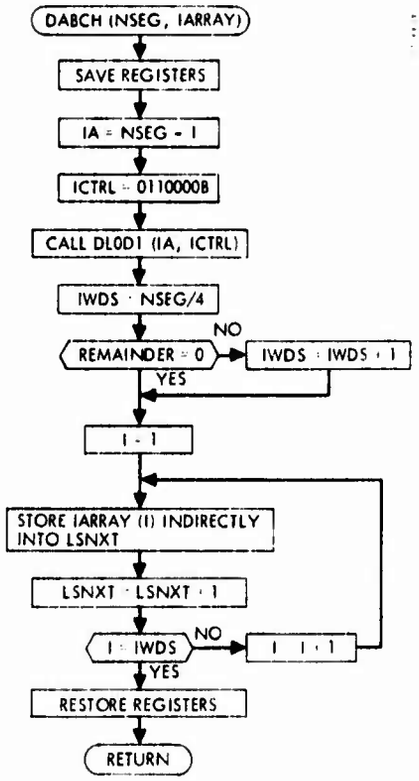

DROTA (ISIN, ICOS)
SAVE REGISTERS
ICTRL = 050000B
CALL DLODS (ISIN, ICOS, ICTRL)
RESTORE REGISTERS
RETURN

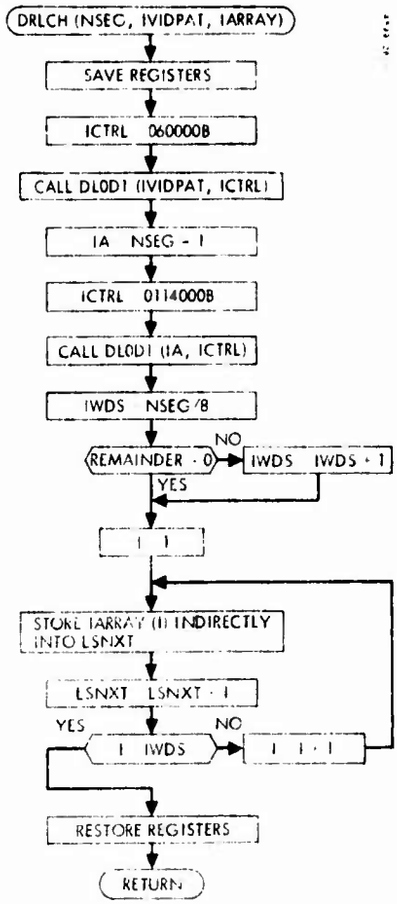

DVPAT (IPAT)
SAVE REGISTERS
ICTRL = 060000B
CALL DLOD1 (IPAT, ICTRL)
RESTORE REGISTERS
RETURN

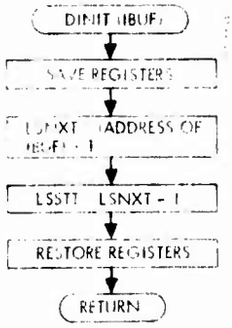

DSGLN (LNGTH)
SAVE REGISTERS
LNGTH LNGTH - 1
LNGTH ≥ 1
NO
LNGTH 2
YES
ICTRL 070000B
CALL DLOD1 (LNGTH, ICTRL)
RESTORE REGISTERS
RETURN

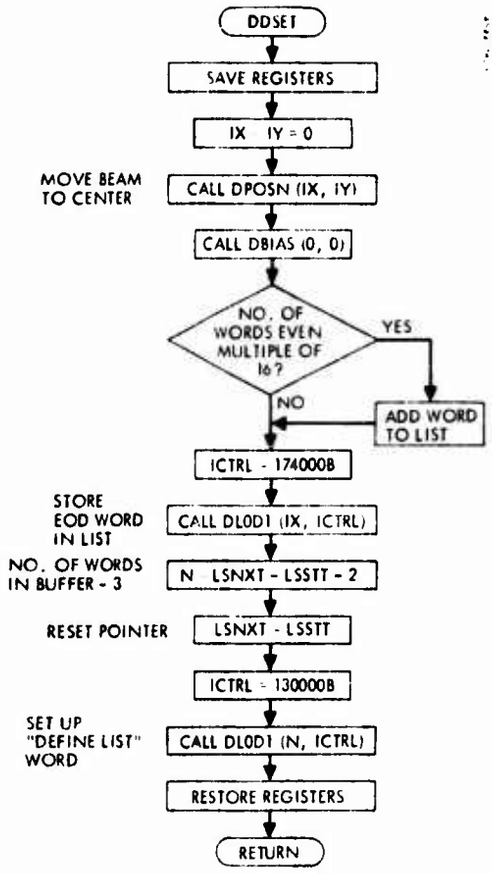

DLINE (LNGTH)
SAVE REGISTERS
LNGTH LNGTH - 1
ICTRL 030000B
RESTORE REGISTERS
RETURN

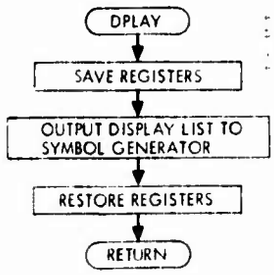

DD001
RESTORE REGISTERS
RETURN

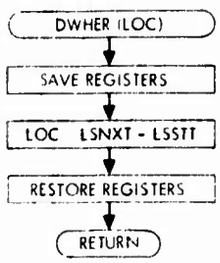

DABCH (NSEG, IARRAY)
SAVE REGISTERS
IA = NSEG - 1
ICTRL = 0110000B
CALL DL0D1 (IA, ICTRL)
IWDS = NSEG/4
REMAINDER = 0
NO
IWDS = IWDS + 1
YES
I = 1
STORE IARRAY (I) INDIRECTLY INTO LSNXT
LSNXT = LSNXT + 1
I = IWDS
NO
I = I + 1
YES
RESTORE REGISTERS
RETURN

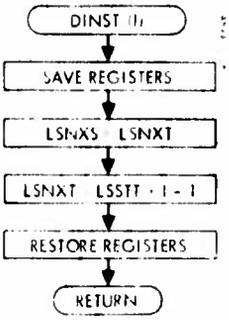

DRLCH (NSEC, IVIDPAT, IARRAY)
SAVE REGISTERS
ICTRL 060000B
CALL DLODI (IVIDPAT, ICTRL)
IA NSEG - 1
ICTRL 0114000B
CALL DLODI (IA, ICTRL)
IWDS NSEG/8
REMAINDER = 0
NO
IWDS IWDS + 1
YES
I 1
STORE IARRAY (I) INDIRECTLY INTO LSNXT
LSNXT LSNXT + 1
YES
I IWDS
NO
I I + 1
RESTORE REGISTERS
RETURN

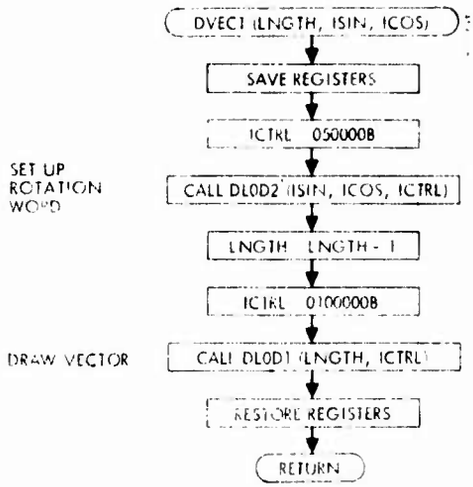

DINIT (IBUF)
SAVE REGISTERS
LSNXT ADDRESS OF IBUF + 1
LSSTT LSNXT - 1
RESTORE REGISTERS
RETURN

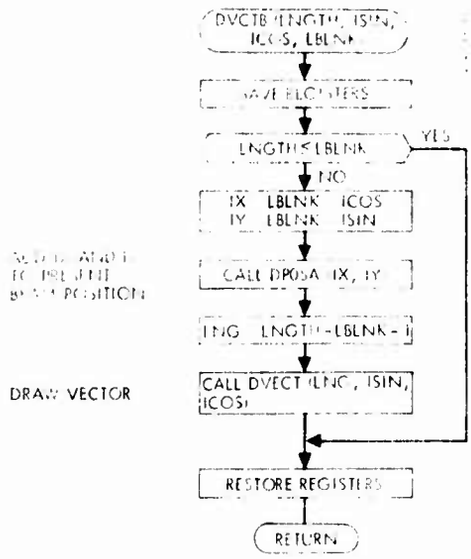

DDSET
SAVE REGISTERS
IX IY = 0
MOVE BEAM TO CENTER
CALL DPOSN (IX, IY)
CALL DBIAS (0, 0)
NO. OF WORDS EVEN MULTIPLE OF 16?
YES
NO
ADD WORD TO LIST
ICTRL - 174000B
STORE EOD WORD IN LIST
CALL DLOD1 (IX, ICTRL)
NO. OF WORDS IN BUFFER - 3
N LSNXT - LSSTT - 2
RESET POINTER
LSNXT - LSSTT
ICTRL = 130000B
SET UP "DEFINE LIST" WORD
CALL DLOD1 (N, ICTRL)
RESTORE REGISTERS
RETURN

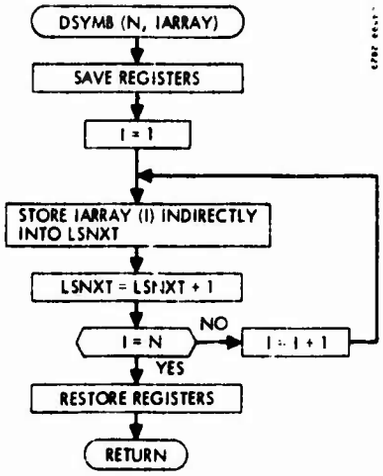

DPLAY
SAVE REGISTERS
OUTPUT DISPLAY LIST TO SYMBOL GENERATOR
RESTORE REGISTERS
RETURN

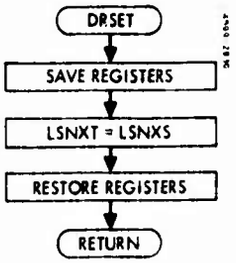

DWHER (LOC)
SAVE REGISTERS
LOC LSNXT - LSSTT
RESTORE REGISTERS
RETURN

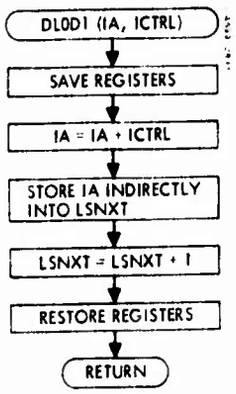

DINST (I)
SAVE REGISTERS
LSNXS = LSNXT
LSNXT = LSSTT + I - 1
RESTORE REGISTERS
RETURN

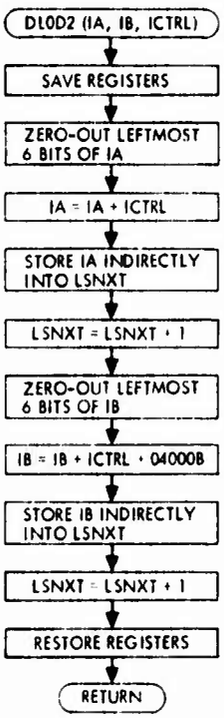

DVECT (LNGTH, ISIN, ICOS)
SAVE REGISTERS
ICTRL = 050000B
SET UP ROTATION WORD
CALL DLOD2 (ISIN, ICOS, ICTRL)
LNGTH = LNGTH - 1
ICTRL = 0100000B
DRAW VECTOR
CALL DLOD1 (LNGTH, ICTRL)
RESTORE REGISTERS
RETURN


DVCTB (LNGTH, ISIN, ICOS, LBLNK
SAVE REGISTERS
LNGTH ≤ LBLNK
YES
NO
IX = LBLNK * ICOS
IY = LBLNK * ISIN
CALL DPOSA (IX, IY
LNG = LNGTH - LBLNK - 1
DRAW VECTOR
CALL DVECT (LNG, ISIN, ICOS)
RESTORE REGISTERS
RETURN


DSYMB (N, IARRAY)
SAVE REGISTERS
I = 1
STORE IARRAY (I) INDIRECTLY INTO LSNXT
LSNXT = LSNXT + 1
I = N
NO
I = I + 1
YES
RESTORE REGISTERS
RETURN


DRSET
SAVE REGISTERS
LSNXT = LSNXS
RESTORE REGISTERS
RETURN


DLOD1 (IA, ICTRL)
SAVE REGISTERS
IA = IA + ICTRL
STORE IA INDIRECTLY INTO LSNXT
LSNXT = LSNXT + 1
RESTORE REGISTERS
RETURN


DLOD2 (IA, IB, ICTRL)
SAVE REGISTERS
ZERO-OUT LEFTMOST 6 BITS OF IA
IA = IA + ICTRL
STORE IA INDIRECTLY INTO LSNXT
LSNXT = LSNXT + 1
ZERO-OUT LEFTMOST 6 BITS OF IB
IB = IB + ICTRL + 040000B
STORE IB INDIRECTLY INTO LSNXT
LSNXT = LSNXT + 1
RESTORE REGISTERS
RETURN

A.4.1 Display Utility Software Glossary

Name	Definition
ASAV	Used to save accumulator contents
DARY1	Array address for use in DSYMB
DARY2	Array address for use in DRLCH
DARY3	Array address for use in DABCH
DBLNK	Amount of vector to blank
DBRT	Display brightness indicator
DCØLR	Display color indicator
DCØN0	Constant 0
DCØN1	Constant 1
DCØN2	Constant 2
DCØN3	Constant 3
DCØS1	Cosine for use in DVECT
DCØS2	Cosine for use in DVCTB
DCØS3	Cosine for use in DROTA
DCTL1	Symbol generator draw DOT command
DCTL2	Symbol generator brightness command
DEND1	End of array containing symbol to be added by DSYMB
DEND2	End of array containing chain to be added by DRLCH
DEND3	End of array cortaining chain to be added by DABCH
DFLAG	0 – indicates FORTRAN call 1 – indicates SYM II call
DHUE	Display saturation indicator

Name	Definition
DIDX1	Used by DSYMB to store the array into the display list
DIDX2	Used by DRLCH to store the array into the display list
DIDX3	Used by DABCH to store the array into the display list
DIX1	Horizontal display position used by DPOSN
DIX2	Horizontal display position used by DPOSR
DIX3	Horizontal display bias used by DBIAS
DIX4	Horizontal display offset used by DPOSA
DIY1	Vertical display position used by DPOSN
DIY2	Vertical display position used by DPOSR
DIY3	Vertical display bias used by DBIAS
DIY4	Vertical display offset used by DPOSA
DLNG1	Vector length used by DVECT
DLNG2	Vector length used by DVCTB
DLNG3	Segment length
DLNG4	Line length
DLØC	Dummy variable
DNSG1	Number of 2-bit chain segments − 1
DNSG2	Number of 4-bit chain segments − 1
DNUM1	Number of words in symbol chain
DPAT1	Relative chain video pattern
DSIN1	Sin for use in DVECT
DSIN2	Sin for use in DVCTB
DSIN3	Sin for use in DROTA
DVID1	Relative chain video pattern

Name	Definition
DWDCT	Number of words in display list
LSNXS	Dummy variable
LSNXT	Display list pointer
LSSTT	Display list starting address
SAVEX	Dummy variable
STTADD	Starting address of completed display list
XSAV	Used to save index contents
XSAVE	Used to save index contents

GPO 883-885