NASA Contractor Report 189603

P. 266

Advanced Transport Operating System (ATOPS) Color Displays Software Description MicroVAX System

Christopher J. Slominski
Valerie E. Plyler
Richard W. Dickson

Computer Sciences Corporation
Hampton, Virginia

Prepared For
Langley Research Center
under Contract NAS1-19038
January 1992

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

TABLE OF CONTENTS

Appendices

List of Figures

Section 1.0 INTRODUCTION

This document describes the software created for the display MicroVAX computer used on the Advanced Transport Operating System (ATOPS) project at the Langley Research Center. The software was developed by Computer Sciences Corporation (CSC) for NASA under contract NAS1-19038. This document targets the software delivery of February 27, 1991 as a baseline system. Since a few items have been accepted for the next delivery, they will also be included in this document and noted as such.

The display MicroVAX computer is the host to the Sperry microprocessor display system. The software residing in that system is addressed by another document entitled:

Advanced Transport Operating System

COLOR DISPLAY SOFTWARE
DOCUMENTATION

Microprocessor System

The display host computer works in tandem with another MicroVAX computer, referred to as the Flight Management and Flight Controls computer (FM/FC). The document

FLIGHT MANAGEMENT / FLIGHT CONTROLS
SOFTWARE DOCUMENTATION

should be referenced for information about FM/FC software.

Throughout this document, descriptions of software modules are presented in a standardized format. The basic template is shown on the next page. At the top of the form is a header block containing miscellaneous information about the module. Next appears a one or two sentence synopsis used as a quick reference stating the purpose of the module. A detailed description follows which may be a small paragraph to several pages in length. Global symbol references are listed next. These are the subroutines and common variables referenced by the particular module. Note that passed parameter variables are not shown here. Passed parameters are provided in the CALLING SEQUENCE portion of the header information block. When an asterisk is appended to the name of a data variable listed in the gloabl reference section it denotes a memory location modified by the module.

```
MODULE NAME:          ........
FILE NAME:            ........
PROCESS:              ........
CALLED BY:            ........
CALLING SEQUENCE:     ..............

PURPOSE:
     ..................................................

DESCRIPTION:
     ..................................................
..........................................................
..........................................................

GLOBAL REFERENCES:

VARIABLES
  ....... ....... .......

ARRAYS
  .......

FUNCTIONS AND SUBROUTINES
  ........ ........ ........
```

Section 2.0 SYSTEM OVERVIEW

The various sub-sections of the system overview briefly describe the overall configuration of the displays host software on the MicroVAX flight computer. The reader should be familiar with the VAX/VMS operating system. Several important key words are listed below. Detailed information about these concepts is provided by the VAX/VMS reference manuals. In particular "Introduction to VMS" and "Guide to Using VMS" are good places to start.

DEC command language (DCL)
command files
processes
images
process priorities
global sections
exceptional conditions / condition signaling
condition handlers

Section 2.1 PROCESSES AND EXECUTABLE IMAGES

There are six executable images associated with the displays host software. Three of them are utility programs and three are displays applications programs. Their names are given below with a brief description of their purposes.

	(utilities)
DDSTAR	manipulate data recording tables
SECTION	install and remove global sections
VIEW	monitor global variables
	(displays applications)
DSPFST	perform displays real-time calculations
DSPSLW	perform displays background processing
DSPHDL	perform system functions (timing, interrupts, I/O)

The environment created for the displays executable images consists of four VAX processes. They are the initial process created from the user login and three spawned sub-processes. The utility programs run in the context of the login process. Any one of the three may be activated from the terminal with the RUN command. The other three images remain active continuously under the context of their own sub-process. Since the displays applications images are always active, the VMS priority system determines how often the images actually execute. DSPHDL and DSPFST are assigned priorities within the VMS real-time range, 19 and 18 respectively. DSPSLW uses the default round-robin priority of 4.

The three displays applications images each have a well defined set of responsibilities. The remaining pages of this section list the computations performed by DSPFST, DSPSLW, and DSPHDL. The names of modules which make up each image are also included.

Section 2.1.1 IMAGE / MODULE SUMMARY

EXECUTABLE IMAGE: DSPFST

DSPFST is the displays real-time applications program. The computations performed are repeated once every 50 milliseconds (20 times per second). Its major function is the generation of real-time display data for the Sperry microprocessor color display system. This includes the data for all microprocessor formats, except the information created for the navigation displays map background. Other functions include the formatting of variables for data recording and the processing of inputs received from the microprocessor system. The following is a list of program modules which comprise DSPFST.

MODULE	FILE	PURPOSE
DSPFST	DSPFST.FOR	Executive module
DP_LOAD	DSPFST.FOR	microprocessor identification
MAPTBL	MAPTBL.MAR	global section mapping tables
DDASOT	DDASOT.MAR	data recording
DSNAP	DSNAP.FOR	variable snapshots
DUMPS	DSNAP.FOR	variable snapshots
PROJECT	PROJECT.FOR	time-box positioning
PFDEXC	PFDEXC.FOR	Sperry PFD format
SYSEXC	SYSEXC.FOR	system warning format
NAVEXC	NAVEXC.FOR	navigation format: main
SELTRK	NAVEXC.FOR	NAV: selected track
NAVMLS	NAVEXC.FOR	NAV: MLS airplane
TRENDV	NAVEXC.FOR	NAV: trend vector
RNGARC	NAVEXC.FOR	NAV: altitude range arc
TBOX	NAVEXC.FOR	NAV: time-box
ROTATE	NAVEXC.FOR	NAV: coordinate rotations
PTHPOS	PTHPOS.FOR	NAV: time-box
TIMPOS	PTHPOS.FOR	NAV: time-box
LINE	PTHPOS.FOR	NAV: time-box
TURN	PTHPOS.FOR	NAV: time-box
PASSBY	PTHPOS.FOR	NAV: time-box
INBRG	PTHPOS.FOR	NAV: time-box
PTHLEG	PTHPOS.FOR	NAV: time-box
FMTBZL	FMTBZL.FOR	NAV: bezel panel interface
PFD_NASA	PFD_NASA.FOR	Primary flight format: main
ALT_CNVRT	PFD_NASA.FOR	PFD: altitude scaling
AIRGAM	AIRGAM.FOR	PFD: aircraft/gamma symbols
STAND_OFF	AIRGAM.FOR	PFD: standoff symbols
WINDOW	WINDOW.FOR	PFD: inner window symbols
STAR	STAR.FOR	PFD: waypoint star
LIMITS	STAR.FOR	PFD: waypoint star
RWYMGR	RWYMGR.FOR	PFD: perspective runway
SCREEN	RWYMGR.FOR	PFD: perspective runway

PFDPK	PFDPK.MAR	PFD: binary packing
CASMGR	CASMGR.FOR	PFD: airspeed
MSGMGR	MSGMGR.FOR	PFD: warning messages
GUIDE	GUIDE.FOR	PFD: mode control panel interface
SCLXTK	GUIDE.FOR	PFD: mode control scaling
ENGEXC	ENGEXC.FOR	Engine format: main
EPR_F1	ENGEXC.FOR	ENG: EPR limits
EPR_F2	ENGEXC.FOR	ENG: EPR limits
FFPRC	FFPRC.FOR	ENG: fuel flow
FTEST	FTEST.FOR	ENG: lab simulation
TOPEXC	TOPEXC.FOR	TOPMS format: main
APLANE	TOPEXC.FOR	TOP: aircraft dynamics
SIMTOP	TOPEXC.FOR	TOP: simulation
STPDIS	STPDIS.FOR	TOP: stopping distance
STPREF	STPDIS.FOR	TOP: stopping distance
FNSERV	STPDIS.FOR	TOP: servo response
FILL	FILL.FOR	TOP: I/O memory formatting
ASPDCO	TOPMS.OLB	TOP: airspeed conversion
EPRF	TOPMS.OLB	TOP: EPR
FINTER	TOPMS.OLB	TOP: throttle positioning
RWYPRD	TOPMS.OLB	TOP: runway distance
THCORF	TOPMS.OLB	TOP: thrust
XLIM	TOPMS.OLB	TOP: value limiting
ANGL	UTIL.OLB	angular adjustment
ASSIGN	UTIL.OLB	logical names
GET	UTIL.OLB	Fortran address pointers
GRID	UTIL.OLB	map coordinates
LOCK	UTIL.OLB	memory residency
MAPCOM	UTIL.OLB	global section mapping
MXV	UTIL.OLB	matrix, vector product
POLAR	UTIL.OLB	X, Y, Z to polar coordinates
POSBTS	UTIL.OLB	map clipping
C_HDL	UTIL.OLB	condition handling
REPORT	UTIL.OLB	condition handling
REPORT_CHECK	UTIL.OLB	condition handling
SHOW_TT	UTIL.OLB	condition handling
UVC	UTIL.OLB	unit vector
SCOS	UTIL.OLB	sine/cosine
VCP	UTIL.OLB	vector cross product
VDP	UTIL.OLB	vector dot product
VMG	UTIL.OLB	vector magnitude
XYZ	UTIL.OLB	polar to X, Y, Z coordinates
BCDTIM	UTIL.OLB	decode system time
EXCEPTIONS	UTIL.OLB	exception messages

EXECUTABLE IMAGE: DSPSLW

DSPSLW is the displays background processing program. It executes in the spare time remaining after DSPHDL and DSPFST have completed their real-time tasks in the 50 millisecond frame. Each time DSPSLW completes its tasks it loops back to start again, like the real-time images. However there is no time constraint governing how fast it must complete one iteration of its computations. Its major function is the creation of the display data for the navigation format map background. Other functions include the TOPMS pretakeoff calculations and the PFD status announcements. The following is a list of the program modules which comprise DSPSLW.

MODULE	FILE	PURPOSE
DSPSLW	DSPSLW.FOR	executive module
PRN_AST	DSPSLW.FOR	printer completion AST
LOG_AST	DSPSLW.FOR	data log completion AST
SNAPOUT	SNAPOUT.FOR	data snap-shots
SBXMGR	SBXMGR.FOR	PFD status messages
MAPTBL	MAPTBL.MAR	global section mapping
NAVSLW	NAVSLW.FOR	map background executive
NAVUPD	NAVSLW.FOR	map background processing
BUSFMT	NAVSLW.FOR	map background processing
OPTION	OPTION.FOR	map symbology
AIRPRT	OPTION.FOR	map airports
ARPSMB	OPTION.FOR	map airports
RUNWAY	OPTION.FOR	map runways
STRIPS	OPTION.FOR	map longitudinal strips
NAVAID	OPTION.FOR	map navaids
NAVSMB	OPTION.FOR	map navaids
RADIAL	OPTION.FOR	map radials
PATHS	PATHS.FOR	map flight plan
PLAN	PATHS.FOR	map flight plan
LEG	PATHS.FOR	map flight plan
DMA	PATHS.FOR	map flight plan
TURN	PATHS.FOR	map flight plan
WPTXT	PATHS.FOR	map flight plan
BOUNDS	BOUNDS.FOR	map boundary areas
AREAS	BOUNDS.FOR	map boundary areas
NEARPT	BOUNDS.FOR	map boundary areas
TEXT	TEXT.FOR	map info lines
STORE	TEXT.FOR	map info lines
NAV_UTL	NAV_UTL.MAR	map background data formatting
PROJECT	PROJECT.FOR	map position computations
MAP_AIRWAY	MAP_AIRWAY.FOR	map airways
GET_XY	MAP_AIRWAY.FOR	map airways
NAME_SIZE	MAP_AIRWAY.FOR	map airways
PRETKF	PRETKF.FOR	TOPMS pretakeoff main

TOP_HDL	PRETKF.FOR	TOPMS condition handler
ADBW2	TOPMS.OLB	TOPMS integration
ATMOS	TOPMS.OLB	TOPMS atmospheric parameters
ASPDCO	TOPMS.OLB	TOPMS airspeed conversion
OLIMIT	TOPMS.OLB	TOPMS open-end limiting
ONED	TOPMS.OLB	TOPMS interpolation
SEARCH	TOPMS.OLB	TOPMS table search
RATE	TOPMS.OLB	TOPMS rate limiting
THCORF	TOPMS.OLB	TOPMS thrust
XLIM	TOPMS.OLB	TOPMS value limiting
DZONE	TOPMS.OLB	TOPMS dead zone
THSRVO	TOPMS.OLB	TOPMS throttle response
RWYPRD	TOPMS.OLB	TOPMS runway distance
FINTER	TOPMS.OLB	TOPMS throttle position
EPRF	TOPMS.OLB	TOPMS EPR
THROTS	THROTS.FOR	TOPMS throttle setting
AEROC	AEROC.FOR	TOPMS aircraft lift/drag
ENGINE	ENGTKF.FOR	TOPMS engine model
POLYFT	POLYFT.FOR	TOPMS curve fitting
SIMEQA	POLYFT.FOR	TOPMS solving equations
LNGFM	LNG2D.FOR	TOPMS longitudinal axis moments
STPDIS	STPDIS.FOR	TOPMS stopping distance
STPREF	STPDIS.FOR	TOPMS stopping distance
FNSERV	STPDIS.FOR	TOPMS servo response
ACTRIM	ACTRIM.FOR	TOPMS aircraft trimming
ANGL	UTIL.OLB	angle limiting
ASSIGN	UTIL.OLB	logical assignment
CLIP	UTIL.OLB	map clipping
POSBTS	UTIL.OLB	map clipping
FMTTIM	UTIL.OLB	time formatting
GET	UTIL.OLB	Fortran pointer data
GET_CHAR	UTIL.OLB	Fortran pointer data
GRID	UTIL.OLB	map projections
LOCK	UTIL.OLB	memory residency
MAPCOM	UTIL.OLB	global section mapping
OTS$FLOAT	UTIL.OLB	data formatting
POLAR	UTIL.OLB	X, Y, Z to polar coordinates
C_HDL	UTIL.OLB	condition handling
REPORT	UTIL.OLB	condition handling
REPORT_CHECK	UTIL.OLB	condition handling
SHOW_TT	UTIL.OLB	condition handling
SCOS	UTIL.OLB	sine/cosine
UVC	UTIL.OLB	unit vector
BCDTIM	UTIL.OLB	system time conversion
EXCEPTIONS	UTIL.OLB	exception messages

EXECUTABLE IMAGE: DSPHDL

DSPHDL is the displays executive program. It runs at a priority higher than the other images, therefore it has the ability to execute immediately whenever it needs to. DSPHDL sits idle waiting for clock interrupts which occur every ten milliseconds. A set of five interrupts make up one complete 50 millisecond real-time frame. The major functions for DSPHDL are performing system I/O and signaling the real-time applications program (DSPFST) when to restart a new 50 millisecond frame. It also scales and formats I/O data to the proper engineering units. The following is a list of the program modules which comprise DSPHDL.

MODULE	FILE	PURPOSE
DSPHDL	DSPHDL.MAR	timing, DMA I/O, interrupts
DISFIL	DISFIL.MAR	input formatting
DOUTIO	DOUTIO.MAR	output formatting
HDL_MESG	HDL_MESG.MAR	error messages
MAPTBL	MAPTBL.MAR	global section mapping table
ASSIGN	UTIL.OLB	logical unit assignment
LOCK	UTIL.OLB	memory residency
MAPCOM	UTIL.OLB	global section mapping
C_HDL	UTIL.OLB	condition handling
REPORT	UTIL.OLB	condition handling
REPORT_CHECK	UTIL.OLB	condition handling
SHOW_TT	UTIL.OLB	condition handling
BCDTIM	UTIL.OLB	system time conversion
EXCEPTIONS	UTIL.OLB	exception messages

Section 2.2 GLOBAL SECTIONS

Global data variables are shared within the software system through global sections. Global sections are the fastest way a multiple process software configuration can share data values.

Global sections are areas of physical memory which are mapped into the virtual address space of several active images. In the display flight software each global section consists of one relocatable program section following the standard definition of the VAX Fortran common block. The displays host software uses ten global sections. All but two, AADCOM and DISDAT, are defined as Fortran include files which contain one common block definition. The include files are needed to provide the global section templates to the Fortran compiler when compiling the Fortran modules which make up most of the displays host software. The other two global sections are macro assembly language files which are assembled directly to produce an object file containing global symbol definitions for all common variables. The following is a list of the global sections with a note on the type of memory allocations contained within each.

AADCOM	navigation data base
BIUCOM	I/O memory for SPERRY microprocessors
DISDAT	default recording list
DISNAV	input data from FM/FC MicroVAX
DLNCOM	data link information
DSPCOM	general displays variables
DSRCOM	data recording
DTCCOM	I/O memory for aircraft DATAC bus
INPCOM	formatted DATAC variables
TOPCOM	TOPMS variables

Object files are created for each of the Fortran include files by the VAX utility program BLKMAC. The Fortran compiler is not used for this since the object modules it creates do not define the individual variables of the common block as global symbols. The global symbol definitions are necessary to allow VAX macro assembly language modules efficient access to global section variables. The file COMMON.FOR exists solely for BLKMAC. This file is a Fortran "Block Data" module which includes each of the global section template files and also contains initialization statements for some of the global section variables. BLKMAC reads this file and creates an object file for each common block referenced within.

The object files created by BLKMAC are linked into the program SECTION which is used to create global sections. SECTION is an interactive program that allows the user to create, refresh, or delete the global sections. Global sections must be created in memory before any of the display flight software is started. Note that SECTION will issue a warning when attempting to delete global sections which are currently in use by applications software. The user may choose to proceed with or abort the deletion. If the delete is not aborted the VAX/VMS operating system removes the sections from its global section table but does not free the physical memory until the last image mapped to the sections has exited. This in effect changes the global sections to private sections.

Executable images gain access to the global sections through a call to the utility library module MAPCOM on start-up. MAPCOM grants read or read/write access to the various global sections depending on predefined access privileges encoded in the file MAPTBL.MAR. The utility program GLOBAL is used to define the access privileges for each process. GLOBAL creates the ".OPT" files and MAPTBL.MAR used in linking each executable image.

Section 2.3 INSPECTING GLOBAL VARIABLES WITH VIEW

The program VIEW is used to examine and modify variables in the VAX global sections defined for flight software. To use this utility the global sections must have been installed previously using the utility SECTION. The VIEW display screen contains two header lines and twenty lines for the placement of variables (see the diagram on the next page). The first header line contains the version number of VIEW, the flight system identifier to which VIEW was linked, and the date of the flight system generation. The second header line shows which of the four available display pages is currently being shown. The display lines each have the line number on the left side of the display. When variables are placed on the display line three additional fields are shown after the line number. First the format code for the variable is shown. This tells how many bytes of data comprise the selected variable, and how the binary value is interpreted. Next on the line is the value of the variable. The last part of the display line is the descriptive label used to identify what variable was placed on the line.

To start the program enter RUN VIEW on an account containing a flight system. VIEW immediately prompts for a password. The password is used to determine the read/write privilege that VIEW grants to the various global sections. VIEW maintains default privileges for users with no password. The default entry into VIEW is gained by simply entering a carriage return to the password prompt. The person responsible for the flight system build selects the access to each global section for both the default and password users. Refer to appendix B for information on the VIEW commands.

```
VIEW [V5.1]: TDWR DISPLAYS  12-MAR-1991
     Page 1
  1  I.2                0  TOPMS
  2  H.2             0020  ENABLE
  3
  4  F.4         -13.1027  ROLL
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15
 16
 17
 18
 19
 20
   ->ARRAY(16)/F=F.8/R=2/L=6
```

- FIGURE 2.1 -

Section 2.4 STARTING AND STOPPING VAX DISPLAY SOFTWARE

There are eleven files needed for a complete displays host software system. These include the six executable images described in section 2.1 and the following five files used to manage the execution of the system.

RUN.COM

This command procedure is used to start the displays host software. First it checks if an old set of log files are open (see section 2.5) and closes them if necessary. The utility program section is automatically run next to allow the user to install or refresh the global sections. Finally the sub-processes are spawned and exception log files opened by calls to GO.COM. Note that the executable image for DSPFST is run twice in RUN.COM. The first time it is run the sub-process is spawned in total control of the user terminal with the parent process placed in a wait state. This must be done to allow an interactive I/O session to confirm system configuration. DSPFST then exits and returns control to the parent process. The second time DSPFST is spawned it is made a joint process allowing the parent process to continue with control of the interactive terminal.

GO.COM

This command procedure opens the process exception log file and starts the executable image. It is called once for each of the three display software processes.

HALT.COM

This command procedure is used to properly terminate the VAX display software. The first thing it does is close the exception log files and delete all but the latest three versions of each. Since the log files are process permanent files, the logical end-of-file mark is forced to the physical end-of-file. All three sub-processes are terminated with the DCL STOP command. Finally the user is given the opportunity to select automatic removal of the installed global sections.

GBLNAME.DAT

This information file is used by the utility program SECTION as a reference to the names of all defined global sections. This file is also used when creating the software system. Refer to appendix C for its role in the system build.

SHOW_LOG.COM

This command procedure is used to review the current exception log files while the system is active. Section 2.5 (condition handling) has complete information on the log files.

Once logged into an account containing the aforementioned files the user may start the displays software by entering "@RUN" at the console terminal. The user is immediately prompted by SECTION to choose between installing the global sections or refreshing a previously installed set of global sections. When this is complete a table identifying the current microprocessor configuration is printed to the screen. This table is generated from information received from the display system. The user must review the format assignments to make sure the display system has determined correctly in which processors the various formats are loaded. If the information shown does not depict the desired display format configuration the user is given the opportunity to correct the problem and try again. Once all the user interaction is complete the remaining processes are spawned. Finally the utility program DDSTAR is automatically run to initialize the default data recording list. At this point the standard DCL "$" prompt is issued and the console terminal may be used for DCL commands and running utility programs while the display host software executes.

Section 2.5 CONDITION HANDLING

Numerous types of exceptional conditions may occur on a VAX/VMS system. These can be both hardware and software faults or traps which occur when the system detects a programming error. Without outside intervention the VMS operating system takes predefined actions through the default system condition handler. User defined condition handlers may be defined "further up the stack" to intercept exceptions before they reach the system condition handler. The displays software has defined a condition handler to perform special operations for several commonly occurring exceptions.

The system operator is notified of the occurrence of exceptions in several ways. Each process has an exception counter defined in one of the global sections (HDL_ERR, FST_ERR, SLW_ERR). These variables contain the total number of exceptions that have occurred in each process since the system was started. The utility program VIEW can be used to monitor the counters. For most exceptions an explicit notification is given at the time it occurs. The notification consists of a brief message sent to the system console terminal and a detailed description of the exception placed in the process's log file (DSPFST.LOG, DSPHDL.LOG, DSPSLW.LOG). To eliminate unnecessary I/O the terminal and log file notification will only be made once every fifteen seconds for a repeated exception. A repeated exception must have both the same error code and originate from the same machine instruction.

The following is a list of the exceptional conditions handled by the displays condition handler. Any other exceptions signaled to the displays condition handler will simply be resignaled to the the default system condition handler after the terminal and log file notifications have been posted.

(software traps from VMS math library)

MTH$_SQUROONEG - The square root of negative value error forces the math library function return value to be zero. No exception message is posted for this error.

MTH$_* - All other math library exceptions also force the function return value to zero. Terminal and log file notification are given for these.

(hardware faults)

SS$_FLTOVF_F
SS$_FLTDIV_F - These faults are modified to simulate their corresponding traps since continuation of the applications software after the fault cannot be done. VMS resignals the displays condition handler with the new trap. Note that the exception counter will be incremented twice because of this action.

SS$_ROPRAND - This fault occurs when floating point data contains an illegal binary code. There is only one undefined floating point bit pattern; the sign bit set and all other bits clear (-0). The reserved operand is changed to a value of zero and the instruction is re-started.

(hardware traps)

SS$_FLTOVF
SS$_FLTDIV
SS$_INTDIV
SS$_INTOVF - These traps are reflected in the exception counters and posted on the terminal and in the log file. The applications software continues afterward with the following instruction. Note that the integer overflow exception currently cannot occur in the software since the detection is disabled by the Fortan compiler switch /NOCHECK.

Section 2.5.1 TOPMS CONDITION HANDLER

A special condition handler is defined during the pretakeoff calculations performed in DSPSLW. This handler (TOP_HDL) receives all signaled exceptions before the main display software condition handler is activated. All normal VMS exception conditions are resignaled to the standard handler. When one of a few TOPMS related problems occurs a non-standard condition code is signaled by TOPMS pretakeoff software. When one of these errors is seen by the TOPMS handler a stack unwind is implemented which forces DSPSLW to return to the instruction immediately following the call to the pretakeoff main module. A flag variable is also set which invalidates the TOPMS display format.

Section 2.5.2 EXCEPTION LOG FILES

Exception messages are saved in log files defined for each process (DSPHDL.LOG, DSPFST.LOG, DSPSLW.LOG). Inactive log files may be viewed with DCL commands such as TYPE, COPY, or PRINT. When the display software is executing the active set of log files are accessed with the SHOW_LOG.COM command procedure. There are three forms available to use.

```
@SHOW_LOG <process_name>
@SHOW_LOG <process_name> ALL
@SHOW_LOG <process_name> SINCE
```

The "ALL" form will display on the user's terminal all exceptions posted in the file, which is empty when the software system is started. The "SINCE" form shows the user the exception messages posted since the last time the particular log file was referenced by @SHOW_LOG. The first form is equivalent to the "SINCE" form.

Each exception message on the log file consists of a header with the current MicroVAX date and time, followed by the aircraft Greenwich Mean Time (GMT). Next appears the VMS exception message followed by a traceback of the call frames.

Section 3.0 I/O COMMUNICATIONS

In order for display application software (DSPFST, DSPSLW, DDSTAR, etc.) to function correctly, real-time data from external sources must be input, and processed data must be output, in a synchronized manner. This is the responsibility of the process DSPHDL.

DSPHDL initializes system resources to allow external I/O, to schedule this external I/O, and to control the subprocesses DSPFST and DSPSLW. DSPHDL also formats I/O data for/from these processes. The executable image DSPHDL.EXE is activated either in the context of an interactive user or the context of a subprocess of an interactive user which has been created using the DCL SPAWN command. Upon activation DSPHDL raises its priority into the realtime region at level 19 which disables quantum expiration context switching. DSPHDL will then use system context to configure I/O channels for DMA with external devices (DATAC, BIU, FM/FC MicroVAX, etc). The subprocess DSPFST will be synchronized into a 50 millisecond frame by DSPHDL using an interrupt from the DATAC. The DATAC will also supply a 10 millisecond clock for synchronization of DMA I/O. This I/O data is formatted for use by the display application software by DSPHDL.

MODULE NAME: DSPHDL
FILE NAME: DSPHDL.MAR
PROCESS: DSPHDL
CALLED BY: (Main Module)
CALLING SEQUENCE: RUN DSPHDL

PURPOSE:
Initializing system resources to allow external I/O, to schedule this external I/O, and to control the subprocesses DSPFST and DSPSLW.

DESCRIPTION:
This module is very intimately tied to, and has been written around, the framework of the VAX/VMS internal architecture. As such, an understanding of VMS, in particular those portions relating to virtual memory structure and the internals of VMS I/O, will be required to follow the methodology used in the configuration of this system for DMA I/O. This understanding may be obtained from the standard VMS documentation set (in particular Programming Volume 8 - Device Support, paying particular attention to the sections mapping I/O space and connecting to an interrupt vector) and the text 'VAX/VMS Internals and Data Structures'. This understanding is assumed in this discussion and in source code comments. The module DSPHDL contains four functional parts. These include:

1.) Initialization code

2.) Main loop processing

3.) Kernel mode routines used in initialization

4.) Connect to interrupt routines

The following describes each:

1.) Initialization code - This code performs the following:

A.) Establish an error condition handler. See section 2.5 for details.

B.) Assign a channel to the default terminal so that any error messages may be reported there. Also assign a channel to the TXA5 serial port which is used for packet radio link communication.

C.) Declare an exit handler which will set process priority back to level 4.

D.) Set process priority to level 19. This puts the process DSPHDL into the real time range and will disable any quantum expiration context switching.

E.) Lock P0 process space into the working set by calling LOCK. This will reduce the possibility of page faulting during main loop execution.

F.) Assign a channel number to each of the following devices:

a.) XAA0 - DATAC DRV11

b.) XAB0 - Inter-Processor Link (IPL) DRV11

c.) XAD0 - Bus Interface Unit (BIU) DRV11

d.) KWA0 - KWV11

G.) Call MAPCOM to map to the required global sections.

H.) Use $CRMPSC to map to the physical addresses of the I/O registers for the DRV11 and KWV11 devices. This will allow the process to reference physical locations via virtual addresses.

I.) Call kernel mode routines which compute the virtual addresses of DMA buffer page table entries for use in loading Q-bus adapter mapping registers during the connect to interrupt start routines. These are discussed in section 3 below.

J.) Associate to common event flag cluster. These event flags are used for synchronization of the flight application processes.

K.) Connect to interrupt vectors. A connect to interrupt $QIO is executed for each of the three DRV11 and the KWV11 devices. This establishes connect to interrupt init, start, interrupt service, and cancel routines as discussed in detail in section 4.

L.) Wait for first 50 millisecond interrupt, then enable the 10 millisecond clock and start main loop software.

2.) Main loop processing. Main loop processing begins by waiting for either a 10 or 50 millisecond interrupt. Upon determining which interrupt occurred, DSPHDL will either execute major or minor frame processing for 50 and 10 millisecond interrupts, respectively. Major and minor frame processing is described below:

A.) Major frame - Major frame processing begins after a DATAC 50 millisecond attention interrupt. This occurs in minor frame 4 several milliseconds before the minor frame 0 interrupt. The minor frame counter variable MFRAME is set to a -1 during this interval. DSPHDL will, at the beginning of a major frame:

a.) Read 234 words of raw data from the DATAC SIR via DRV11 into DTCCOM. This includes hexadecimal SIR addresses 39 through 123.

b.) Format input data for use by display application software, using DISFIL, into INPCOM, if the variable freeze is not set.

Upon completion of I/O, DSPHDL will set event flag 64, enabling DSPFST to execute a frame.

B.) Minor frame - There are five minor frames per major frame (minor frame 0 thru 4). Minor frame zero begins with the first 10 millisecond interrupt after the 50 millisecond attention interrupt. Each 10 millisecond interrupt will signal the beginning of the next minor frame. DSPHDL will perform the following processing at the beginning of the specified minor frame:

1.) Minor frame 0 - No I/O is performed here, as it was done at the beginning of the major frame (during the MFRAME = -1 interval).

2.) Minor frame 1 - Nothing performed in this frame.

3.) Minor frame 2 - Enable IPL interrupt for transfer in frame 3.

4.) Minor frame 3 - Output 181 words (176 words for recorded data and FM/FC feedback data plus 5 words not used as a pad against corrupted data possible at the start of a transfer) in OUTCOM to the DATAC SIR. The corresponding hexadecimal SIR addresses of valid data are from 550 to 5FF. This data has been previously loaded by the routine DOUTIO. Also during this frame the FM/FC IPL transfer occurs, as initiated by an FM/FC MicroVAX interrupt. The responding handshaking software in the display MicroVAX resides in the IPL Interrupt Service Routine (ISR). This handshaking is as follows :

a.) The wordcount for the IPL transfer is loaded into the DRV11 data register by the FM/FC MicroVAX for reading by the display MicroVAX. As a flag, this count is negated if the transfer does not send the active navigation buffer. For the active buffer the count is left positive.

b.) The address for the buffer to be received is loaded into the display MicroVAX DRV11 buffer's address register.

c.) The transfer is initiated.

5.) Minor frame 4 - In this frame the BIU I/O is performed. This begins with the reception of 320 words of input data from the BIU being read into the buffer BIU_IN. Subsequent to this, 704 words are output from the buffer BIU_OUT to the BIU.

3.) Kernel mode routines - These subroutines are called from the initialization software with the $CMKRNL system service. This code must run in kernel mode in order to reference the privileged registers PR$_P0BR (P0 base register) and PR$_P0LR (P0 length register). These registers are needed in order to calculate the virtual address (in S0 space) of the DMA buffer's page table entry. This value is used in loading the Unibus adapter mapping registers.

4.) Connect to interrupt code - There is one connect to interrupt $QIO per device. A connect to interrupt $QIO has four associated parts - initialization, start, interrupt service, and cancel. These four parts are doubly mapped both in process P0 space and in system S0 space allowing them to run in system context. The role of each in this application is described below:

A.) Initialization - The only function this part has at present is to store the system mapped address of the device register block. While not used at present, this could be used to control a device's registers from another device's ISR, should the need arise.

B.) Start - The start routine is used to load the Q-bus adapter mapping registers with the physical address of the DMA buffer. This loading is achieved using the system routine IOC$LOADUBAMAP, which uses as input the virtual address of the buffer's page table entry (computed in the kernel mode routines described above). Connect to interrupt start routines normally run at IPL 6, but since the allocation and loading of mapping registers requires an IPL of 8, the IPL is raised at the beginning and then lowered back to 6 before exiting.

C.) Interrupt service routine - This code is executed when an interrupt is delivered from the associated device. Except for the BIU handshaking code described above in the section on minor frame 3, the only function the ISR is used for at present is to clear the device's CSR and optionally set an event flag. Whether an event flag is to be set after an interrupt is specified as an input flag to the connect to interrupt $QIO. Presently, the devices which will set an event flag are the KWV11 (event flag 4, the 10 millisecond clock) and the DATAC DRV11 (event flag 5, the 50 millisecond attention interrupt).

D.) Cancel - This code is executed at the time of process termination, and is used to release mapping registers that had been allocated.

GLOBAL REFERENCES:

VARIABLES
FRAME*, FRAMES*, MFRAME*, FREEZE, CNT50*, DSPST2*, HSTCNT*, HDL_ERR*, MS10ML*, MS50ML*, DTC_NRDY*, BIU_TO*, IPL_NZ*, IPL_NR*, DISNAV_BEG, ACT_WPTS, BIU_IN, BIU_OUT, DTC_IN, DTC_OUT, IPER_WCR*, IPER_BAR*, IPER_CSR*, IPER_DAT*

FUNCTIONS AND SUBROUTINES
DISFIL, DOUTIO, LOCK, MAPCOM, IOC$ALOUBAMAPN, IOC$LOADUBAMAP, IOC$RELMAPREG

MODULE NAME: DOUTIO
FILE NAME: DOUTIO.MAR
PROCESS: DSPHDL
CALLED BY: DSPHDL
CALLING SEQUENCE: JSB DOUTIO

PURPOSE:
To format 50 millisecond output data from OUTCOM for DMA to the DATAC SIR.

DESCRIPTION:
DOUTIO is called once per major frame just prior to DATAC output. It is responsible for formatting and packing data recording output data. Output data to be formatted is read from OUTCOM. Variables to be recorded are specified along with scale factors in the buffer table DASPAR. These variables are scaled and placed in the buffer DASBF for output.

GLOBAL REFERENCES:

VARIABLES
DASCAL, MXENT

ARRAYS
DASPAR, DASBF*

MODULE NAME: DISFIL
FILE NAME: DISFIL.MAR
PROCESS: DSPHDL
CALLED BY: DSPHDL
CALLING SEQUENCE: JSB DISFIL

PURPOSE:
To format 50 millisecond DATAC SIR DMA input data into INPCOM.

DESCRIPTION:
DISFIL is called once per major frame immediately following a large block input from the DATAC. It is responsible for formatting raw input data into a form usable by the display application software. DISFIL stores formatted input data into INPCOM.

DISFIL uses the following programmer defined macros:

1.) SMPLXF - Will scale, bias, and/or bit shift a 16 bit input integer source operand as specified, convert to floating point, and store the result at a destination pointed to by R0.

2.) SMPLXB - Tests the specified bit of the source and sets or clears the byte boolean pointed to by R0 conditionally.

3.) ASMPLX - Similar to SMPLXF with the addition of a validity bit test preceding the conversion. The result is stored at the address specified in R0.

4.) GSMPLX - Similar to ASMPLX but tailored for use in formatting GPS raw input data.

5.) SBOOL1 - Tests the bit specified and if set will set the boolean pointed to by R0. Otherwise the boolean is cleared.

GLOBAL REFERENCES:

VARIABLES
DTC_IN, (All variables in common INPCOM)*

FUNCTIONS AND SUBROUTINES
OTS$POWRR

Section 4.0 EXECUTIVE SOFTWARE

The three display VAX applications processes each have main modules which are entered directly from VMS when their respective executable images are started. The executable image DSPHDL is covered in section #3 and will not be mentioned again in this section. The processes DSPFST and DSPSLW each have main modules which are described on the following pages.

Main modules contain operations to setup and initialize items which effect the entire process in which they reside. They also serve as a caller of subroutines which perform the actual display tasks required to drive the microprocessor display system.

MODULE NAME: DSPFST
FILE NAME: DSPFST.FOR
PROCESS: DSPFST
CALLED BY: VMS
CALLING SEQUENCE: RUN DSPFST

PURPOSE:

DSPFST is the executive module for its process.

DESCRIPTION:

DSPFST is the main program module for the executable image of the same name. VMS transfers control to the start of this module when the RUN DSPFST command is performed. There are two distinct sections in DSPFST. The first consists of several operations performed once at the start of the executable image. The second section consists of things which are repeated cyclically. Every 50 milliseconds a sequence of operations are started. When the various tasks are completed, DSPFST enters a wait state until the next 50 millisecond frame is announced.

At the start of the executable image, DSPFST forces all the program image "pages" into memory. They are locked into memory to reduce paging I/O during image execution. The loading and locking operation is performed by the utility procedure LOCK. It is called by the VMS system procedure SYS$CMEXEC (change mode to executive). This is done because user mode does not have enough privilege to bring some of the image's pages into memory. The privileged pages are the defined exception messages for the condition handler which are normally only accessed in executive mode by the operating system.

After page locking is complete the global sections are mapped into the executable image's virtual address space. This is done with a call to the utility procedure MAPCOM. MAPCOM is passed the name of the image (DSPFST) which is used to determine which global sections will be mapped and whether write privileges are to be granted.

If DSPFST is run in interactive mode (see section 2.4) confirmation of display microprocessor formats is performed followed by exiting the image. The module DP_LOAD is called to determine which display formats are loaded into the various microprocessors. A summary is printed to the console terminal and the system operator is prompted for confirmation of its correctness.

When DSPFST is not run in interactive mode the remainder of the startup operations are performed. First the user defined condition handler is established into the initial stack frame. Any exceptions occurring in the image DSPFST will be intercepted on their way "up the stack" to the system condition handler. The standard output and error devices are logically assigned to the processes log file so exception messages may be saved in a file (see section 2.5).

The DAS recording tables are initialized to the default recording list by automatically running the utility program DDSTAR from DSPFST. Section #5 provides ample information on data recording and the role DDSTAR plays.

The final thing done before entering the real-time cyclical portion of the module is to connect to a set of event flags in the operating system to be used for inter-process signaling. The start of each new 50 millisecond frame is denoted by the setting of event flag #64.

DSPFST uses the SYS$WAITFR system service to place itself in a wait state until event flag #64 is set on. Once this occurs DSPFST will execute over any processes of lesser priority (all but DSPHDL). The major responsibility of DSPFST during the 50 millisecond frame is calling procedures which generate the data buffers used by the display micro-processors. Data recording procedures are also called at this time. In addition, several miscellaneous operations are performed. They include the processing needed when displays "freeze" has been requested (appendix B), and DSPFST timing estimates. If event flag #64 is already set when DSPFST completes the tasks required during the 50 millisecond frame, a frame time overflow is noted by incrementing the overflow counter (OVER).

GLOBAL REFERENCES:

VARIABLES
CNT CNT50 COLDST* DISPST* DP_TEST* FMTSEL* FREEZE* FSTCNT* FST_ERR* HRSS IWSFLG LABFLG MAPUPD* MAXF* MFRAME NEWFILT* OVER* TITLE TM_ADR* TOINDX TOPMS*

ARRAYS
OUTDAT

FUNCTIONS AND SUBROUTINES
ASSIGN AWAS_MGR C_HDL DDASOT DP_LOAD DSNAP ENGEXC FFPRC FOR$EXIT FTEST INSITU INSITU_MGR LIB$ESTABLISH LIB$MOVC5 LIB$SPAWN LIB$STOP LOCK MAPCOM NAVEXC PFDEXC PFD_NASA REC_ALERTS SYS$ASCEFC SYS$CLREF SYS$CMEXEC SYS$READEF SYS$WAITFR SYSEXC TOPEXC

MODULE NAME: DP_LOAD
FILE NAME: DSPFST.FOR
PROCESS: DSPFST
CALLED BY: DSPFST
CALLING SEQUENCE: CALL DP_LOAD

PURPOSE:
To confirm the display microprocessor configuration.

DESCRIPTION:
DP_LOAD is called when the process DSPFST is first started. It parses through the input buffer received from the Sperry microprocessor system to determine which display formats are loaded. This identification procedure is done for the following reasons.

. The position in the input buffer of bezel button inputs from particular formats must be known by the host software. (see appendix A for input buffer layout)

. Navigation formats have custom map background data created and sent destined to particular processors.

. The system operator may verify the versions of formats loaded by examination of the format checksums shown on the display.

The process DSPHDL must already be active when DP_LOAD is called since it is responsible for performing the I/O which fills the microprocessor input buffer.

The first thing DP_LOAD does when called is to wait two seconds. Even though the process DSPHDL is spawned before DSPFST, DP_LOAD can sometimes be called before DSPHDL has received the required inputs.

DP_LOAD steps through the input buffer examining the return status location for DP11 through DP33. When a format is loaded in a display microprocessor and operating properly a unique numeric value is stored in the return status which is used for identification purposes. DP_LOAD saves each format's ID and checksum to create the confirmation display placed on the console terminal. There are two formats which have special requirements, the Navigation format and the Primary Flight format. The MicroVAX display software needs to save the position where these formats return bezel button status. Also the microprocessor sequence number (1-9) is saved for Navigation formats so map background data may be addressed directly to individual microprocessors.

When DP_LOAD has examined all nine return status locations it prints its summary to the console terminal. A prompt message is printed requiring the operator to accept the information as correct or reject it. If accepted DP_LOAD simply returns. Otherwise a pause message is printed and DP_LOAD waits for a keyboard entry from the operator which instructs DP_LOAD to repeat the microprocessor verification procedure.

GLOBAL REFERENCES:

VARIABLES
  TDW_FOUND*

ARRAYS
  INDAT PFDBZL*

RECORD ARRAYS
  NVFMT*

FUNCTIONS AND SUBROUTINES
  FOR$CLOSE FOR$OPEN GET_WORD LIB$STOP OTS$CVT_L_TZ SYS$SETIMR
  SYS$WAITFR

MODULE NAME: DSPSLW
FILE NAME: DSPSLW.FOR
PROCESS: DSPSLW
CALLED BY: VMS
CALLING SEQUENCE: RUN DSPSLW

PURPOSE:
Serve as the executive module for its process.

DESCRIPTION:
DSPSLW is the main program module for the executable image of the same name. VMS transfers control to the start of this module when the RUN DSPSLW command is performed. There are two distinct sections in DSPSLW. The first consists of several operations performed once at the start of the executable image. The second section consists of things which are repeated cyclically. Each time it completes the required tasks it jumps to the start and begins again.

At the start of the executable image, DSPSLW forces all the program image "pages" into memory. They are locked into memory to reduce paging I/O during image execution. The loading and locking operation is performed by the utility procedure LOCK. It is called by the VMS system procedure SYS$CMEXEC (change mode to executive). This is done because user mode does not have enough privilege to bring some of the image's pages into memory. The privileged pages are the defined exception messages for the condition handler which are normally only accessed in executive mode by the operating system.

After page locking is complete the global sections are mapped into the executable image's virtual address space. This is done with a call to the utility procedure MAPCOM. MAPCOM is passed the name of the image (DSPSLW) which is used to determine which global sections will be mapped and whether write privileges are to be granted.

Next the user defined condition handler is established into the initial stack frame. Any exceptions occurring in the image DSPFST will be intercepted on their way "up the stack" to the system condition handler. The standard output and error devices are logically assigned to the process's log file so exception messages may be saved in a file (see section 2.5).

DSPSLW enters an "infinite loop" where it stays until the process is stopped by external intervention. The main function performed here is the creation of data involved with the map background updates. This is done by the call to NAVSLW. Other functions include the setting of Primary Flight Display format status information, initiation of TOPMS pretakeoff calculations, handling snap-shot data outputs, and computing background timing estimates.

GLOBAL REFERENCES:

VARIABLES
CNT DATA_LOG DAY* FRAMES HRSS MAGVAR ORGHDG* ORGLEN* PRINTER
PRN_ACTIVE RPTR RWYID* SLWCNT* SLW_ERR* SPTR TITLE TKFLEN
TM_ADR* TOPMS* TOPST*

ARRAYS
AIRPTS

FUNCTIONS AND SUBROUTINES
ASSIGN C_HDL DAY_OF_YEAR GET_REAL GET_WORD HARD_COPY
LIB$ESTABLISH LIB$SIGNAL LIB$STOP LOCK MAPCOM NAVSLW
PRETKF SBXMGR SNAPOUT SYS$ASSIGN SYS$CMEXEC

MODULE NAME: PRN_AST
FILE NAME: DSPSLW.FOR
PROCESS: DSPSLW
CALLED BY: VMS
CALLING SEQUENCE: <AST module>

PURPOSE:
Printer completion AST.

DESCRIPTION:
This Asynchronous Trap procedure (AST) is called by VMS when outputs to the system line printer have completed. The flag PRN_ACTIVE is cleared to enable use of the printer by other modules.

GLOBAL REFERENCES:

VARIABLES
PRN_ACTIVE*

MODULE NAME: LOG_AST
FILE NAME: DSPSLW.FOR
PROCESS: DSPSLW
CALLED BY: VMS
CALLING SEQUENCE: <AST module>

PURPOSE:
Log device completion AST.

DESCRIPTION:
This Asynchronous Trap procedure (AST) is called by VMS when outputs to the system logging terminal have completed. The flag LOG_ACTIVE is cleared to enable use of the printer by other modules.

GLOBAL REFERENCES:

VARIABLES
LOG_ACTIVE*

Section 5.0 DATA RECORDING

There are five data recording modules which provide the capability to record selected data on magnetic tape, paper, and strip charts. The file DISDAT.MAR, linked with the process DDSTAR, contains a default list of data items to be recorded through the Data Acquisition System (DAS). It also contains a group of alternate tables which provide lists of variables to be plotted on the strip charts. DDSTAR is an interactive program which permits the experimenter to modify the data recording tables and to set up "snap" tables for printing selected variables on the experimental systems line printer. DDSTAR processes the recording list information and stores addresses and scale factors for the DAS.

The module DSNAP works with tables generated through DDSTAR. When a user specified condition is encountered, DSNAP saves the associated set of data items. Subsequently, the background module SNAPOUT prints the data values to the line printer.

The subroutine DDASOT takes the data specified in the DAS lists, formats it, and stores it in the 50 words of DAS output memory. The strip chart data are also included in the DDASOT output, which is routed to the onboard strip charts by the DAS.

The output from DSNAP and the strip charts is available in flight. The data stored on the DAS tape is available for a "quick look" soon after the experimental flight is completed. DAS information is available over the long term for more thorough data reduction and analysis.

MODULE NAME: DDASOT
FILE NAME: DDASOT.MAR
PROCESS: DSPFST
CALLED BY: DSPFST
CALLING SEQUENCE: CALL DDASOT

PURPOSE:
To configure the alternate-tables to control the strip chart recorders, and to reformat certain data for recording.

DESCRIPTION:
DDASOT first checks the NODAS boolean to determine whether or not DDSTAR is modifying the recording tables, if true it exits immediately. DDASOT then checks the globals RECWD, RECWD1, RECWD2, and RSWADR to determine which set of alternate tables should be stored in the global DASPAR parameter list for strip chart recordings. If RSWADR is clear or if there is a boolean FALSE at the address contained in RSWADR, then the "normal" table set specified in RECWD1 is used, otherwise, RECWD2 is used. RECWD contains the current configuration. If it does not match the selected pattern, then a new set of alternate tables is loaded into DASPAR. This will happen when the alternate tables have been changed through DDSTAR and RECWD is set to -1. The values in RECWD1, RECWD2, and RSWADR are user specified through task VIEW as follows:

RSWADR: USAGE

CLEAR = The primary set of alternate tables will be written to the DASLST strip chart blocks. (RECWD1)

ADDRESS = The address of some discrete, such as MLSVAL, which will, when TRUE, cause the secondary set of alternate tables to be used. (RECWD2) (exercise extreme caution when using this option, check with the system administrator to ensure VIEW has the same virtual addresses as DSPFST.)

RECWD1/RECWD2: BIT MAP

BITS 3,2,1,0:	Value 0-7,	Use Alt Tables 0-7 for Strip Blk 1.
	Value 8-15	Reserved for future expansion.
BITS 7,6,5,4:	Value 0-7,	Use Alt Tables 0-7 for Strip Blk 2.
	Value 8-15	Reserved for future expansion.
BITS 9,8 :	Value 0-3,	Use Atl Tables 8-11 for Strip Blk 3.
	Value 4-15	Reserved for future expansion.

For a normal configuration of tables 0, 1, and 8, RECWD1 would be set to 0010 hexadecimal. For a secondary configuration of tables 4, 5, and 9, RECWD2 would be set to 0114 hexadecimal.

The most recent table configuration is recorded in RECWD. If this does not match the selected pattern, then the table addresses need to be changed. Otherwise, control passes to the data processing code at label CONT. DDSTAR selects recording table configuration 0010 hexadecimal by default.

The alternate-table setup is done at label DOIT. As appropriate, RECWD1 or RECWD2 is moved into RECWD as the new configuration record. ALTPAR is the source of the new tables. It is loaded by DDSTAR from DISDAT and/or from user input. It consists of 12 tables with 8 entries per table, 2 long-words per entry. The format and its significance are:

LWORD1:	Bits	31-25	Unused
	"	24	ON denotes an 8 bit variable.
	"	23:16	shift count (+ = Left), used to position integer data for recording.
	"	15	SET denotes NOT floating point data
	"	14:0	Scale factor for the data.

NOTE: For a floating point variable, the entire longword is a scale factor.

LWORD2: Address of the data.

The three required tables are identified and transferred to the first 16 entries (8 x 3) in DASPAR, the primary DAS recording list which includes both the alternate tables and the rest of the data list for recording. On a run where a table change has occurred, DASOT terminates at this point.

On a nominal run, when the tables are static, DASOT builds the packed discrete DISOUT and also calculates and stores the current navigation position errors. The booleans to be packed into DDISOT are listed locally at label DISLST. The sign bit of each boolean is shifted left into a register which, at the end, is shifted to place the bits at 0:9, and moved into DDISOT. Then, LAT and LON are converted to 32 bit integer data and output as LATFIN and LONFIN.

GLOBAL REFERENCES:

VARIABLES

ALTDIF* ALTPAR DASPAR* DISOUT* GUID2D GUID3D GUID4D IDDLTC* IDDLNC* LAT LATDIF* LONDIF* LONINS RECWD* RECWD1 RECWD2 RSWADR

MODULE NAME: DDSTAR (Displays DAS/Snap Access Routine)
FILE NAME: DDSTAR.FOR
PROCESS: DDSTAR
CALLED BY: A: DSPHDL (on cold start)
B: The User (manually)
CALLING SEQUENCE:

A. VAXHDL:

```
        DS_PROC_NAME:      .ASCID   /DDSTAR/
                           .......
$CREPRC_S  IMAGE=DS_PROC_NAME,-    ; DSTAR IMAGE NAME
           INPUT=TERM_DESC,-       ; USE CREATING PROCESS'S
           OUTPUT=TERM_DES,-       ;    I/O DEVICE
           BASPRI=#20,-            ; PRIORITY 20 (RUNS NOW)
           PRCNAM=DS_PROC_NAME     ; SUBPROCESS NAME
```

B. Manually: RUN DDSTAR

PURPOSE:
A utility to transfer recording parameters for the Data Acquisiton System (DAS), to accept interactive modifications to the existing parameters, and to create parameter tables for the DSNAP routine.

DESCRIPTION:
The primary function of DDSTAR is to load the tables (DASPAR & ALTPAR) used by DDASOT to select and route data to the Data Acquisition System (DAS) for recording on tape or on the aircraft strip charts. This is done automatically and transparently on system startup (cold start) when DDSTAR is called by the I/O handler (DSPHDL). In this case, DAS processing is enabled for whatever data is defined in the default DAS list, nominally DISDAT. This is an external file included in the DDSTAR process which is documented separately in this volume. (DDSTAR defines DISDAT as a common block containing structured records which correspond in format and quantity to the entries in DDATA and ATABL, the 2 global data blocks in DISDAT.)

The secondary function of DDSTAR is to run interactively and accept user input to modify the DAS list, or to create or modify the snap tables. Snap tables (SCRIT) do not pre-exist and can only be created through DDSTAR. DDSTAR also provides a mechanism for saving/returning snap tables, DAS changes, and alternate table changes to/from disk storage.

The interactive routine is menu driven and generally self explanatory. However, more detailed instructions will be displayed at various points if the user selects tutorials in response to the initial question and prompt: "Do you want tutorials? Y/N."

There are up to 50 entries in the DDATA section of the DAS list, each consisting of a name, address, and scale factor. Entries may be changed, and/or new ones added up to the limit. Entries 1 through 16 are used for the strip chart parameters and are organized in two blocks of eight entries each. Each of these blocks corresponds with 1 of the 12 alternate tables which may be read into this area by subroutine DDASOT. These alternate tables are maintained in the lower section of DISDAT, in the global data block ATABL. Tables zero through seven relate to blocks one or two. In either mode of operation, the contents of DDATA and ATABL are written to the global DASPAR and ALTPAR tables, respectively, during the DAS dump routine.

When modifying the DAS list, the user will be required to enter scale factors for each data item entered or changed. These are explained in the DISDAT documentation in this volume.

Up to eight snap tables may be created. The user should be prepared to enter the name of the variable to be used as the key for the snap, the value at which the snap should occur, the range or "window" if an exact match is not required, and the names of up to 15 variables to be "snapped" to the printer when the snap occurs.

When the name of a variable is requested, DDSTAR will recognize the name of any global variable in any global section in the system. Local variables or local common blocks cannot be referenced. Array elements can be specified with the index in parentheses. A series of array elements can be inserted as one entry by appending an asterisk and count to the index. For example, XYZ(3*15) will pick up element 3, plus the next 14, for a total of 15.

The use of the "bare" carriage return is consistent throughout DDSTAR. It will terminate the current activity, such as a series of data entries, and re-display the previous menu. From any point in the program, three or four carriage returns, at most, will bring control back to the main SNAP/DAS option. There is no limit on the direction or number of times the user may go back and forth through the various sub-options.

Program exit may be selected in response to several menus. However, at any point in the program a control-Z will cause an orderly exit. This is the usual method.

The exit routine calls the dump routines for whichever set of tables was modified during the session. For the DAS and alternate tables, the dump routines first transfer the recording parameters, then print a list of the tables on the aircraft line printer. For the snap tables, it is only necessary to print the list. During an automatic run, nothing is printed. During the first manual run after cold start, both the DAS and alternate tables will be printed, whether modified or not. Otherwise, only the modified set is printed.

The DDSTAR module includes 11 subroutines and calls one external subroutine (SYM_SEARCH).

GLOBAL REFERENCES:

VARIABLES

ALTDMP CHCNT COLDST DASDMP DDATE GETNAME* NOSNAP*
PRINTOUT RECWD1* SNAPACT* SNAPDMP SNENT TERM TUTOR

FUNCTIONS AND SUBROUTINES

DASDUMP DASPRC FOR$CLOSE FOR$DATE_T_DS FOR$OPEN MAPCOM
SNAPDUMP SNPRC

MODULE NAME: CHECK
FILE NAME: DDSTAR.FOR
PROCESS: DDSTAR
CALLED BY: GITEM, NSNAP, OLDSNAP
CALLING SEQUENCE: CALL CHECK

PURPOSE:
To return the location in an input string of a '(*)' character sequence.

DESCRIPTION:
Check returns an index, relative to the beginning of a character string, of the location of the left parenthesis (LPAREN), asterisk (ASTER), and right parenthesis (RPAREN) if they exist. The variables used by CHECK are located in a local common area. These variables include CBUF, a 14 character buffer containing the input string; CHCNT, a character count which may include blanks; and LPAREN, ASTER and RPAREN which were previously defined. In addition to the indices returned, CHCNT will be updated to reflect the elimination of any embedded blanks.

If the character string is not found, LPAREN, RPAREN, and ASTER are set to zero and a return is made to the caller. An error check is made to ensure that the left parenthesis occurs before the right parenthesis. If it does not, the following message is displayed at the user's terminal-
' NO RIGHT PAREN ! TRY AGAIN'.

GLOBAL REFERENCES:

VARIABLES
ASTER* CBUF CHCNT* ERROR* LPAREN RPAREN

MODULE NAME: DASDUMP
FILE NAME: DDSTAR.FOR
PROCESS: DDSTAR
CALLED BY: DDSTAR
CALLING SEQUENCE: CALL DASDUMP

PURPOSE:
To setup DASPAR for use by DDASOT and to print the recording, snap, and alternate tables if requested.

DESCRIPTION:
If the DASDMP flag is true, the DAS recording parameters are transferred to the DASPAR buffer from the temporary area DDATA. Since this affects the data recording process, the NODAS flag is set to inhibit data recording while DASPAR is being modified. Likewise, if the ALTDMP flag is true the alternate table data is transferred to the ALTPAR buffer.
If the PRINTOUT flag is set and DASDMP is true, the DAS list will be printed. If the PRINTOUT flag is set and ALTDMP is true, the alternate tables will be printed.

GLOBAL REFERENCES:

VARIABLES
ALTDMP DASDMP DDATE DNENT NNAME NODAS* PRINTOUT RECWD* TIME

RECORD ARRAYS
ALTPAR* ATABL DASPAR* DDATA

FUNCTIONS AND SUBROUTINES
FMTTIM FOR$IMVBITS

MODULE NAME: DASPRC
FILE NAME: DDSTAR.FOR
PROCESS: DDSTAR
CALLED BY: DDSTAR
CALLING SEQUENCE: CALL DASPRC

PURPOSE:
To serve as the controller for DAS and alternate table processing.

DESCRIPTION:
This module serves as the user interface to DDSTAR for all DAS and alternate table functions. Menus are displayed to the user from which the desired functions may be selected. These functions include modifying the DAS list, modifying the alternate tables, writing the DAS or alternate table modifications to disk, reading the specified DAS or alternate table modifications from disk, and printing the DAS list and/or alternate tables.

The user supplies inputs in response to program prompts once the desired function is selected. Error checking of inputs is performed and informational messages are displayed to guide the user through an interactive session.

GLOBAL REFERENCES:

VARIABLES
ACNT* ALTDMP* ALTFLG* CBUF CHCNT DASDMP* DASFLG* DCNT* DDATE
DNENT ERROR* GETDAS* GETOLD* ITNUM MAX* TABLE* TERM* TUTOR

RECORD ARRAYS
ASAV ATABL DDATA DSAV

FUNCTIONS AND SUBROUTINES
FOR$CLOSE FOR$OPEN GITEM UCASE

MODULE NAME: GITEM
FILE NAME: DDSTAR.FOR
PROCESS: DDSTAR
CALLED BY: DASPRC, NSNAP, SNAPMOD
CALLING SEQUENCE: CALL GITEM

PURPOSE:

To prompt the user for an item name and scale factors as necessary.

DESCRIPTION:

This module serves as the user interface for the input of any global variable, scale factor, or snap criteria data. These data may be processed from user supplied interactive inputs or previously saved changes recovered from a disk file. For each global variable name specified, a call is made to the 'SYM_SEARCH' module which searches the global symbol table to ensure that it is a valid name. SYM_SEARCH also returns the variable address, type (real,integer,etc.), and length in bytes which are used for creating the various recording tables or the snap tables.

As with other DDSTAR modules, this one also provides prompts and processes user supplied inputs. Error checking is performed and messages displayed to guide the user as necessary. Limit checking is performed on DAS entries and alternate table entries (a max of 50 entries allowed for each for saving on disk).

GLOBAL REFERENCES:

VARIABLES
ACNT ADR ALTFLG ASTER CBUF CHCNT DASFLG DCNT DNENT* ERROR*
GETDAS* GETNAME GETOLD* ITNUM* LGTH LPAREN MAX REPEAT RPAREN
SDONE SMOD SNAPACT SNENT* TABLE TERM* TYP

ARRAYS
SNAME*

RECORD ARRAYS
ASAV ATABL DDATA* DSAV* SCRIT*

FUNCTIONS AND SUBROUTINES
CHECK SYM_SEARCH UCASE

MODULE NAME: NSNAP
FILE NAME: DDSTAR.FOR
PROCESS: DDSTAR
CALLED BY: SNPRC
CALLING SEQUENCE: CALL NSNAP
Alternate entry point: CALL GETKEY

PURPOSE:
To create a new snap table or to replace the name and criteria data for a key variable.

DESCRIPTION:
This module establishes a snap table by creating the key variable as input by the user and also prompting the user for the conditions under which the snap is to occur. A call is made to 'SYM_SEARCH' to search the global variable table for the key variable to ensure that it is a valid name. As with other DDSTAR modules, prompts are supplied and user inputs processed in an interactive session. Error messages are displayed as appropriate to guide the user through the session.
The alternate entry point, GETKEY, is used whenever the user is modifying the key variable of an existing snap table. Once this determination is made, the logic path is followed as for the main entry point NSNAP.

GLOBAL REFERENCES:

VARIABLES
ADR CBUF CHCNT ERROR* ITNUM* LGTH LPAREN MAX* NNAME NNENT
RPAREN SDONE SNENT* TABLE* TERM* TUTOR TYP

ARRAYS
SNAME*

RECORD ARRAYS
SCRIT*

FUNCTIONS AND SUBROUTINES
CHECK GITEM SYM_SEARCH UCASE

MODULE NAME: OLDSNAP
FILE NAME: DDSTAR.FOR
PROCESS: DDSTAR
CALLED BY: SNPRC
CALLING SEQUENCE: CALL OLDSNAP

PURPOSE:
To read or write a snap table to or from a disk file.

DESCRIPTION:
If a file is to stored on disk, a test is made to ensure that a snap has been defined, if not a message 'NO SNAPS DEFINED, NOTHING SAVED' is displayed. If a snap has been defined, the user is requested to enter a file name. Error checking is performed for all I/O operations and an appropriate message is returned for any detected errors. A successful write operation is announced by the message ' SNAP TABLES SAVED ON FILE filename' where 'filename' is the user supplied name.

If a file is to be recovered from disk, the user is requested to enter the desired file name. As for the store function, error checking is performed for all I/O operations and appropriate messages returned for any detected error. When the data have been read from disk, each entry is processed as if it were entered from the keyboard. This ensures that data requested from older versions of the flight software are valid with the current version. Once again, an error message will be displayed for any variable not found in the current global symbol table. If no errors are detected, the message 'SNAP TABLES RECOVERED' is displayed on the user's terminal.

GLOBAL REFERENCES:

VARIABLES
ADR CBUF CHCNT ERROR LGTH LPAREN RPAREN SNAPDMP* SNENT
SNPSAV* SRST* TYP

ARRAYS
SNAME

RECORD ARRAYS
SCRIT*

FUNCTIONS AND SUBROUTINES
CHECK FOR$CLOSE FOR$OPEN SYM_SEARCH

MODULE NAME: SNAPDEL
FILE NAME: DDSTAR.FOR
PROCESS: DDSTAR
CALLED BY: SNPRC
CALLING SEQUENCE: CALL SNAPDEL

PURPOSE:
To delete a previously entered snap table.

DESCRIPTION:
This routine is used to delete a snap table from the current working set of DDSTAR snap tables. It is not used to delete previously saved snap tables from disk which may be accomplished by using the appropriate VAX/VMS commands (See VAX/VMS DCL Dictionary for details). The desired snap table number for deletion is entered by the user prior to this routine being called by SNPRC.

GLOBAL REFERENCES:

VARIABLES
DEL* SNENT*

ARRAYS
SNAME*

RECORD ARRAYS
SCRIT*

FUNCTIONS AND SUBROUTINES
FOR$BITEST

```
MODULE NAME:          SNAPDUMP
FILE NAME:            DDSTAR.FOR
PROCESS:              DDSTAR
CALLED BY:            DDSTAR, SNAPMOD
CALLING SEQUENCE:     CALL SNAPDUMP(tnum,nent)
        where:        tnum - table number to dump (zero indicates
                             all tables are to be dumped)
                      nent - output parameter containing number of
                             entries found in the table.
                             (n/a when tnum is zero)

PURPOSE:
     To print a copy of the selected snap table(s).

DESCRIPTION:
     This routine displays the specified snap table at the
user's terminal or all snap tables on the onboard line
printer if 'tnum' is zero.  It may be used to review snap
tables prior to flight or storing them on disk.  It may also
be used to obtain a listing of the current working set of snap
tables for record keeping purposes.

GLOBAL REFERENCES:

VARIABLES
  DDATE SNENT TIME

ARRAYS
  SNAME

RECORD ARRAYS
  SCRIT

FUNCTIONS AND SUBROUTINES
  FMTTIM FOR$BITEST
```

MODULE NAME: SNAPMOD
FILE NAME: DDSTAR.FOR
PROCESS: DDSTAR
CALLED BY: SNPRC
CALLING SEQUENCE: CALL SNAPMOD

PURPOSE:
To modify an existing snap table.

DESCRIPTION:
This routine permits the user to modify a previously completed snap table. The snap table to be modified is entered by the user in response to a program prompt prior to this routine being called by SNPRC. Modifications are then made in response to program prompts which guide the user through a session. Inputs are error checked and appropriate messages displayed on the user's terminal when errors are detected. Any data in the snap table may be modified including the key variable.

GLOBAL REFERENCES:

VARIABLES
CBUF CHCNT ERROR* GETNAME* ITNUM* MAX* MOD NNENT* SMOD*
TABLE* TERM*

ARRAYS
SNAME*

RECORD ARRAYS
SCRIT*

FUNCTIONS AND SUBROUTINES
GETKEY GITEM SNAPDUMP UCASE

MODULE NAME: SNPRC
FILE NAME: DDSTAR.FOR
PROCESS: DDSTAR
CALLED BY: DDSTAR
CALLING SEQUENCE: CALL SNPRC

PURPOSE:
To serve as the executive routine for all snap processing.

DESCRIPTION:
SNPRC prompts the user for desired snap table actions and calls the appropriate subroutines to accomplish them. These actions include creating/modifying snap tables, recovering snap tables from disk, and preserving snap tables on disk.

GLOBAL REFERENCES:

VARIABLES
CHCNT DEL* MOD* NNENT* SNENT SNPSAV* TERM*

FUNCTIONS AND SUBROUTINES
NSNAP OLDSNAP SNAPDEL SNAPMOD

MODULE NAME: SYM_SEARCH
FILE NAME: SYM_SEARCH.FOR
PROCESS: DDSTAR
CALLED BY: DDSTAR
CALLING SEQUENCE: CALL SYM_SEARCH(SYMBOL,ADDRESS,FORM,SIZE)

PURPOSE:
To look-up information about a flight software global variable.

DESCRIPTION:
The name of a flight software global variable is passed as a character string, by descriptor, to SYM_SEARCH as the first calling parameter. The address, format, and byte length associated with the variable are returned through the remaining three call list parameters. When the symbolic name is not found in the global symbol table, all the return values are zeroed.

The global symbol table is a group of symbol information packets having the following format.

- Name length 1 byte
- Name variable length
- Address 4 bytes
- format code 1 byte
- memory length 1 byte

The format codes have the following meaning. The first two both have the same machine data representation (floating point). The utility process VIEW differentiates between these by using floating exponential format to display variables with a format code of "2".

1 floating point
2 floating point
3 signed integer
4 unsigned hexidecimal
5 ASCII

The symbol search starts at the begining of the global symbol table. Both the start address of the table and the number of entries, a global constant, may not be accessed directly from Fortran because of their definition. They are accessed by SYM_SEARCH by declaring them external procedures and using the %LOC operator to obtain their value. The utility functions GET_BYTE and GET_LONG are used to fetch data from the table as it is searched. The library function

STR$COMPARE is used to find a match in symbol names. Since this function requires character string inputs, a descriptor is constructed and passed to STR$COMPARE to make the global symbol table name appear as a character string.

GLOBAL REFERENCES:

VARIABLES
  SYMNUM

ARRAYS
  SYMTAB

FUNCTIONS AND SUBROUTINES
  GET_BYTE GET_LONG STR$COMPARE

MODULE NAME: UCASE
FILE NAME: DDSTAR.FOR
PROCESS: DDSTAR
CALLED BY: DASPRC, GITEM, NSNAP, SNAPMOD
CALLING SEQUENCE: CALL UCASE(cbuf,chcnt)
where: cbuf - character buffer containing data
chcnt - number of characters to convert to upper case

PURPOSE:
To convert lower case ASCII characters to upper case.

DESCRIPTION:
The input characters are tested to ensure that they are in the range a - z and then converted to upper case if they are. Otherwise the characters remain unchanged.

GLOBAL REFERENCES:
none

MODULE NAME: DISDAT
FILE NAME: DISDAT.MAR
PROCESS: DDSTAR
CALLED BY: None (Components are addressed by their global names: ATABL, DDATA, DNENT)

PURPOSE:

To provide a list of data parameters for recording.

DESCRIPTION:

DISDAT is the component of the DDSTAR task which contains the default list of data from the Displays computer to be recorded by the Data Acquisition System (DAS). It contains up to 50 names and corresponding scale factors. Of these, the first 16 specify the output to the strip chart recorders. There is also a group of 12 alternate tables with eight entries each. Two of these may be selected during flight to be read into the strip chart block entries one through sixteen.

DISDAT consists of two functions and two global data blocks. The data block DDATA contains space for 50 entries and the block ATABL contains space for 12 tables of 8 entries apiece. Entries are of the form:

```
1)    .ASCII      /ZHAT_FINE        /
      REAL          ZHAT,512.

2)    .ASCII      /YHAT_COURSE      /
      INTEG         YHAT,2048.

3)    .ASCII      /EVENT7           /
      INTEG       HOLDM,2048.,B
```

The ASCII name field may contain any 12 printable characters or symbols except for a slash. The function REAL will, at assembly time, convert and store the second line as an address and a scale factor in floating point format. The function INTEG does the same except that it stores the address with a negative sign and, if there is a third parameter such as in the third example, then the scale factor is stored with negative magnitude. This indicates a one-byte variable. When the file is ultimately processed by DDSTAR, a positive address signifies that the variable is a real number, a negative address that it is an integer, and a negative scale factor that it is a boolean value.

The scale factor is a decimal number which determines how the data will appear on a strip chart, either on the aircraft or in the post-flight data reduction phase. Starting from a value of zero at the centerline of the chart, the scale is the number at which the needle will be

at the edge of the chart. When this limit is exceeded, the recorder "wraps-around"; the needle jumps to one side or the other and continues reflecting relative changes in the data. Thus, aircraft altitude, scaled at 500.0, would wrap quickly as the aircraft climbs or descends, but it would give a good record of small changes from level flight. Scaled at its maximum range, say 40,000 feet, the resolution of the altitude plot would be very poor.

Scale factor determination must consider the resolution available in the DAS and on the strip chart recorders. The DAS records the most significant 16 bits of data. The recorders can display only 12 bits, including the sign bit. The on-board recorder displays the most significant 12 bits. In post-flight analysis, the recorders can display any 12 bit string in the word.

If data of large magnitude is scaled to display small changes on the strip chart and if it is also necessary to record it at its actual magnitude, then it can be recorded twice. However, to permit post-flight reconstruction, the scale factors must be determined such that there is an overlap of significant bits. This can be done by relating the scale factors to some power of 2, up to 2 ** 15. For example:

```
REAL  POSHAT+10,100.    ;     53 ZHATF    FEET
REAL  POSHAT+10,204800. ;     54 ZHATC    FEET
```

The 100 scale factor for ZHATF will produce good resolution for small changes. At 100 * (2**11) the scale for ZHATC, 204800 feet, is slightly more than the maximum range of ZHAT (32 miles) and it provides a 5 bit overlap in the 16 bit words. Boolean data items are usually scaled at 2048, which equates to full displacement of the needle.

The comment section of a DDATA line also requires attention because some of the fields are parameters for one of the data reduction programs (CALDAS). The line length may not exceed 72 columns. The last word on the line specifies the unit of measurement. This field may not exceed 10 characters and it may not contain embedded blanks, commas, or slashes. Left-leaning slashes and underlines are acceptable. Any type of unit may be specified, but discretes must be indicated by the word "discrete." In the case of a "packed discrete," then those two words must be present. Abbreviations are not acceptable.

The alternate tables are short versions of DDATA. Tables 0 through 7 are for real or integer data. Tables 8 through 11 do not differ from tables 0 through 7. However, there is a hardware strip chart recorder interface limitation that requires these tables to contain only booleans defined as integers (INTEG). Specifically DAS channels 17-24 must contain only booleans for onboard strip chart recording.

MODULE NAME: DSNAP
FILE NAME: DSNAP.FOR
PROCESS: DSPFST
CALLED BY: DSPFST
CALLING SEQUENCE: CALL DSNAP

PURPOSE:

To record snapshot values of user specified variables according to user defined criteria.

DESCRIPTION:

The DSNAP routine records single-event values, called snapshots, for selected variables and stores them in SNAPBUF, in DSRCOM, for subsequent output to the line printer by the SNAPOUT routine. There are 5 snapshot criteria tables (SCRIT), each a structured record which contains a key variable address, the criteria under which that variable should cause a snapshot recording, and a list of up to 15 addresses for the data to be sampled when the snapshot occurs. (See the SCRIT table documentation in the DSRCOM listing.) These tables are set up by the user through the DDSTAR process.

DSNAP checks as many snap tables as have been defined in DDSTAR (indicated by SNENT). First, the type of the key variable is determined from the STYPE sub-field in the SCRIT tables. Depending on whether the key is an integer, real or single-byte, the current value is picked up through a call to GET_WORD, GET_REAL, or GET_BYTE, respectively. All three types are processed similarly. The specified criteria may be that the current value be less than, equal to, or greater than the threshold. If equality is specified, then a "window" will also have been specified and some approximation of equality will be acceptable. If the specified condition is met and a snap has not already been done for this condition, then subroutine DUMP is called and the "SNAP Done" bit is set in the type word. If a snap has been done for the specified condition, then, if that condition is no longer true, the "done" bit is cleared and that snap re-enabled. Thus, only one report is generated each time the condition is satisfied.

That single-byte key variables may be booleans or single-byte integers is irrelevant; both possibilities are checked as if they were integers. A FALSE condition is recognized by a value of exactly zero, TRUE is the least significant bit set. This could also be true for a single-byte integer but the difference is significant only for subroutine SNAPOUT to determine the labelling when the snap is printed.

Subroutine DUMP first increments SPTR (modulo-4) to tell SNAPOUT that a new snap has been recorded, then it stores the number of the snap in the 16th entry of SNAPBUF(n).SDATA. Next, for as long as there is an address (up to 15) in SCRIT(n).SLADR, the address list, it determines the type and byte count of each variable in the list, collects the value at the address through calls to GET_WORD, GET_REAL, or GET_BYTE, as appropriate, and finally stores them as integers or real numbers in the first 15 entries of SNAPBUF(n).SDATA (or .SDATR). A flag is set to indicate whether the variable is an integer or a real number, or else that there was an error or the end of the list.

GLOBAL REFERENCES:

VARIABLES
  NOSNAP RPTR* SNENT SPTR* SRST*

RECORD ARRAYS
  SCRIT*

FUNCTIONS AND SUBROUTINES
  DUMPS FOR$BITEST GET_BYTE GET_REAL GET_WORD

MODULE NAME: SNAPOUT
FILE NAME: SNAPOUT.FOR
PROCESS: DSPSLW
CALLED BY: DSPSLW
CALLING SEQUENCE: CALL SNAPOUT

PURPOSE:
To format and print snapshot recordings on the aircraft line printer.

DESCRIPTION:
SNAPOUT prints out snap data whenever new snapshots have been added to the snap buffer (SNAPBUF(n).SDATA). The global counter SPTR is set by the SNAP routine when a new snap is stored. The global counter RPTR is set by the SNAPOUT routine when the snap is printed. If the two numbers do not agree, then one or more snap lists remain to be printed and DSPSLW makes the call to SNAPOUT. Both counters are modulo-4. SNAPOUT prints one list per call.

If a snap is to be printed, SNAPOUT first increments the read counter RPTR and then formats a header line with the snap number, the name, the time, and the snap criteria, storing these in the output buffer OBUF. It then takes one entry at a time from the snap buffer, checks the form (integer, real or boolean), performs the necessary conversions, and stores the ASCII value in the output buffer. It repeats this for 5 entries per line, for 3 lines, or until the buffer is empty.

Because SNAPOUT requires a change in the I/O device, printing must be synchronized at the executive level. The flag PRN_ACTIVE is used to signal that I/O is in progress and the output is then initiated through a call to SYS$QIO. Subroutine PRN_AST, specified in the QIO statement, clears PRN_ACTIVE when the I/O is complete.

GLOBAL REFERENCES:

VARIABLES
PRINTER PRN_ACTIVE* RPTR

RECORD ARRAYS
SNAPBUF

FUNCTIONS AND SUBROUTINES
LIB$SIGNAL OTS$CNVOUT OTS$CVT_L_TI PRN_AST SYS$QIO

Section 6.0 NASA PFD SOFTWARE

The PFD format shows the current aircraft attitude and provides other critical "aircraft state" information to the pilot. Refer to the PFD format drawing at the end of this section.

The most outstanding section of the PFD format is the rectangular area around the screen center that is topped by a 106 degree arc segment. This area is referred to as the PFD view window. Within the window a number of symbols appear that depict aircraft roll, pitch, yaw, actual and reference flight path angle, angle of attack, and track angle information. Three dimensional representations of the "TO" waypoint and the destination runway are displayed in the window along with a flare guidance cue, radar altitude, and alert messages.

Angular perspective in the window is provided by the pitch grid and horizon ticks. The pitch grid has a double solid line representing the horizon which separates the sky from the ground, along with parallel grid bars spaced in 5 degree increments. Along the horizon line, tick marks are spaced to show 10 degree steps of horizontal displacement. The other window symbology is interpreted against the grid and ticks to ascertain proper angular readings. The area from the horizon line to the top of the view window is raster filled in blue to easily distinguish the sky/ground boundary formed by the horizon line. At the top of the window along the arc is a roll scale which uses a triangular pointer to designate current aircraft roll angle. The roll angle read from the scale corresponds to the amount of rotation applied to the horizon line within the view window.

On either side of the view window are gray raster filled rectangular areas called the airspeed and altitude tapes. They have tick marks and numeric values which can slide vertically giving the appearance of a rolling measurement tape.

The airspeed tape, on the left side of the view window, has the current aircraft airspeed value in the blacked out area at the center of the tape. A blue, amber, or green pointer box may also appear at the appropriate spot on the tape representing the current airspeed selection from the pilot's mode control panel. When airspeed is changing an elongated arrow will grow from the tape center vertically along the outside of the tape ticks and point to the airspeed that will be reached in ten seconds at the current rate of acceleration or deceleration. Also along the same edge of the airspeed tape is a wedge marker that indicates the upper airspeed suggested for the current aircraft flap settings. Directly below the airspeed tape the selected airspeed value, that corresponds to the airspeed pointer box, is shown in either green or amber. The aircraft mach number is shown above the airspeed tape when the value exceeds 0.5.

On the right hand side of the view window is the altitude tape. Similar to the airspeed tape, the current airplane altitude is shown in the blacked out area at the tape center. Alongside the sliding altitude tape on the right is the vertical speed scale. A yellow arrow grows from the center indicating the rate of change in altitude in units of thousands of feet per minute. A blue, amber, or green pointer box may also appear on the altitude tape representing the selected altitude from the pilot's mode control panel. An amber or green triangular pointer which represents the vertical profile of the aircraft's flight plan may also appear along the altitude tape edge. The glideslope pointer and scale are shown just to the left of the altitude tape when selected. These appear as a set of deviation dots with a diamond shaped pointer. Included in the pointer are the letters "GP" or "GS" standing for glide path (MLS) or glideslope (ILS) respectively. The dots and pointer may be green or amber. Immediately below the altitude tape area is the barometric pressure setting from the pilot's control display unit (CDU). The selected decision height value appears above the altitude tape.

The horizontal deviation indicators and scales are presented below the PFD view window. Horizontal deviation from the aircraft's flight plan is shown by an amber or green box pointer, entitled "HOR", placed above a deviation scale. MLS azimuth or ILS localizer deviations are shown by a triangular pointer placed directly below the HOR deviation scale. The pointer indicates position relative to five deviation "dots" which appear along the bottom edge of the HOR scale. The pointer and dots may be shown in amber or green.

The corners of the PFD display contain information pertaining to the current control and guidance modes of the airplane. The upper left corner shows the current control and auto-throttle mode. The destination waypoint of the flight plan is shown in the upper right corner. The currently selected horizontal and vertical guidance modes are shown in the lower left and right corners respectively. Both armed and engaged modes are announced, color coded in amber (armed) and green (engaged).

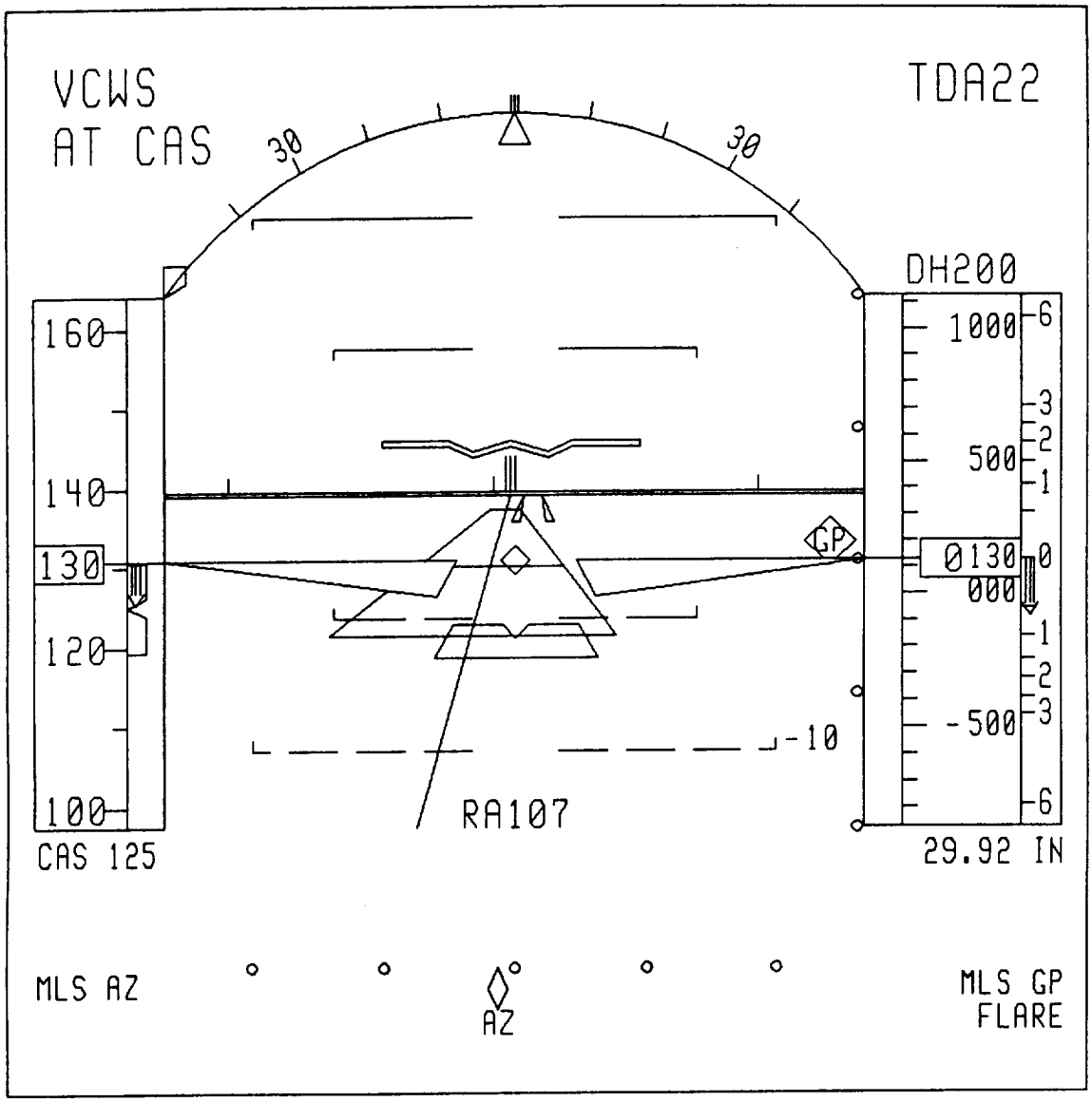
The three upper right bezel panel buttons are active on the PFD format. They are used to select the waypoint star, perspective runway, and alert messages in order from the top.

Only the right hand potentiometer, which controls the value of the decision height, is used for the PFD format.

The interpretation of some of the symbology in the PFD window is affected by the "Velocity Vector" mode. This mode changes the orientation of the symbols within the window. The display is considered in a velocity vector mode when velocity control wheel steering (VCWS) or automatic guidance with pilot selectable flight path angle (FPASEL) is being flown. The current mode can be identified from the display screen by viewing the "Gamma wedge" and "Aircraft" symbology. In the velocity vector mode the gamma wedge is a large stationary symbol, positioned at the screen center. The small aircraft symbol moves relative to the gamma wedge to depict angle of attack and drift angle. In the standard mode a large aircraft symbol is fixed .8375 inches above screen center. Angle of attack and drift angle are shown by the small moving gamma wedge. In either case the pitch and flight path angle values associated with these symbols can be read directly off the pitch grid. The orientation of three items within the view window are affected by velocity vector mode. They are the "Star Waypoint", "Perspective Runway", and "Horizon Ticks" symbols. These three objects all represent positions relative to the aircraft body in the standard mode as follows.

- STAR — The 3D position of the next waypoint.
- RUNWAY — The 3D perspective outline of the destination runway.
- TICKS — The angular displacement to the nearest ten degree heading markers.

All three are representations of what would be seen through the pilot's view window. Since the airplane typically flies in a direction slightly different from the direction pointed to by the aircraft body, the use of the above mentioned symbols is limited. When the aircraft is approaching a reference point, such as the runway, the nose of the airplane usually will be pointing away from it because of drift angle and angle of attack. Therefore the object will be displaced from the center of the screen, corresponding to being off to the side or top of the pilot's view window. This problem is rectified by the velocity vector mode. Instead of orienting symbols relative to the direction indicated by the aircraft body, they are aligned in the direction the aircraft is moving (inertial axis).



PRIMARY FLIGHT DISPLAY

-figure 6.1-

Section 6.1 NASA PFD PROCEDURES

Sixteen procedures are dedicated to the creation of display buffer data for the Nasa PFD format. The following chart lists the procedure names along with their source code language and relative size. Those modules that serve as utility subroutines to another procedure are shown with their caller. The size provided is the percentage of total PFD software memory usage.

Refer to Appendix A to identify which locations in the output buffer (OUTDAT) are used by these modules. Note that locations used for this format are tagged in the appendix by the mnemonic "PFD". The following pages include module descriptions for each of the sixteen NASA PFD software procedures.

MODULE	SOURCE	SIZE
AIRGAM STAND_OFF	FORTRAN	9%
CASMGR	FORTRAN	2%
GUIDE SCLXTK	FORTRAN	18%
MSGMGR	FORTRAN	12%
PACK	VAX MACRO	3%
PFD_NASA ALT_CNVRT	FORTRAN	16%
RWYMGR SCREEN	FORTRAN	17%
SBXMGR	FORTRAN	3%
STAR LIMITS	FORTRAN	14%
UNPACK	VAX MACRO	2%
WINDOW	FORTRAN	5%

MODULE NAME: AIRGAM
FILE NAME: AIRGAM.FOR
PROCESS: DSPFST
CALLED BY: PFD_NASA
CALLING SEQUENCE: CALL AIRGAM

PURPOSE:

To compute the positions of the aircraft and gamma wedges and their standoff symbols.

DESCRIPTION:

This module determines where the aircraft and gamma wedge symbols are positioned and calls the procedure STAND_OFF to perform the computations required for the "stand off" portions of these symbols. The gamma wedges and aircraft symbol are oriented differently in velocity vector mode (vvmode), and non-velocity vector mode. For a detailed description of when velocity vector mode is active, consult Section 6.0 in this document. If "vvmode" is active, the gamma wedges are fixed at the screen center and the aircraft symbol moves around the screen relative to the gamma wedges. The aircraft X and Y positions are based on the rotated yaw (BETAX) and rotated angle of attack (ALPHAX) values respectively, which are computed in the module PFD_NASA. The term "rotated" means that these values are transformed into the pitch axis.

The gamma wedge standoff symbol will appear in "vvmode" if the pitch column is stabilized (indicating no new flight path angle is being commanded), and the difference between actual and commanded gamma is greater than 1.5 degrees. The standoff symbol is a red dashed diamond, the same size as the diamond in the middle of the gamma wedges. The standoff symbol will be displaced directly above or below the gamma wedges by the degree difference of actual and commanded gamma. As the two values converge, the standoff symbol will move closer to the gamma wedges until it is within the 1.5 degree range - at which time it will disappear.

When "vvmode" is not valid, the aircraft symbol becomes the fixed object on the screen. It is positioned at .8375 inches above the center of the screen, which in effect defines a new screen center. The gamma wedges will now move relative to the aircraft, with its position based on the rotated angle of attack and rotated yaw values.

There is also a standoff symbol for the aircraft. It will appear if the attitude control wheel steering button has been pressed on the pilot's mode control panel, no pitch column input is being received, and the difference between actual and commanded pitch is greater than 1.5 degrees.

The aircraft standoff symbol is also composed of red dashed lines, and resembles a "w". It is actually the same size and shape of the middle section of the aircraft symbol. The aircraft standoff symbol is displaced above or below the aircraft by the actual and commanded pitch difference.

It should be noted that the moving aircraft and gamma wedge symbols are limited to the sides, top, and bottom of the PFD window, so that they will never disappear entirely from the screen. Also, the standoff symbols are limited to the top or bottom edge.

Although AIRGAM does not specify it, there are two sets of aircraft and gamma wedge symbols for "vvmode" and "non-vvmode". In "vvmode" the gamma wedges fixed at the screen center are large, while the moving aircraft is smaller. In "non-vvmode" the fixed aircraft is larger, and the moving gamma wedges are smaller. The displays microprocessor software is responsible for determining which set of symbols is displayed.

GLOBAL REFERENCES:

VARIABLES
ACWS AIRCFX* AIRCFY* ALPHAX BETAX FPAPFD GAMC GAMMA MXALF
PITCH PITFLG RLLFLG STDOFF* VVMODE

FUNCTIONS AND SUBROUTINES
STAND_OFF XLIM

MODULE NAME: STAND_OFF
FILE NAME: AIRGAM.FOR
PROCESS: DSPFST
CALLED BY: AIRGAM
CALLING SEQUENCE: CALL STAND_OFF(FLAG, DELTA, INIT, BIAS)

PURPOSE:

To compute the "stand off" symbol offsets from the gamma wedges or aircraft symbols.

DESCRIPTION:

STAND_OFF is a utility subroutine called solely by the procedure AIRGAM. Four values are passed in as formal parameters. The first is a logical value which signals when the proper conditions exist for a "stand off" symbol. The next value is the angular difference between the actual and commanded aircraft parameter being processed by AIRGAM. This value is filtered with a half second time constant to smooth transitions. When the filtered value is above 1.5 degrees the standoff symbol is enabled. A filter initialization flag is the third parameter. The last parameter passed to the module is a bias value which is added directly to the computed stand off position. This is for the 5 degree offset of the aircraft symbol used when not in a velocity vector mode.

MODULE NAME: CASMGR
FILE NAME: CASMGR.FOR
PROCESS: DSPFST
CALLED BY: PFD_NASA
CALLING SEQUENCE: CALL CASMGR

PURPOSE:

Compute data values associated with the PFD airspeed tape.

DESCRIPTION:

The calibrated airspeed (CAS) tape is located on the left side of the PFD format. The current aircraft airspeed is shown in the window fixed in the center of the tape area. The acceleration segment is a yellow arrow which starts at the actual airspeed box and extends vertically to the airspeed value which will be obtained in ten seconds at the current rate of acceleration or deceleration. Reference airspeed is depicted on the display in two ways. A pointer box is placed along the airspeed tape at the position corresponding to the desired value. Also the reference airspeed value is shown under the tape.

CASMGR computes values for the acceleration segment length, reference airspeed, and actual airspeed. The fixed point binary values sent to the displays are scaled to provide the resolution required for accurate presentation on the PFD screen. Reference airspeed is obtained directly from the value selected on the pilot's mode control panel. Aircraft calibrated airspeed is filtered to remove signal noise and used as the display CAS value. A one second lag filter is also applied to find the airspeed rate of change. The display acceleration segment length is derived from the airspeed differential.

The color used for the display of the reference airspeed is set in CASMGR. Logical values from the pilot's mode contol panel are tested to determine the correct setting of the color index sent to the PFD format.

GLOBAL REFERENCES:

VARIABLES
ACCSEG* CAS CASACT* CASF CASFLG* CASREF* IASARM IASSEL
IASSUM

MODULE NAME: GUIDE
FILE NAME: GUIDE.FOR
PROCESS: DSPFST
CALLED BY: PFD_NASA
CALLING SEQUENCE: CALL GUIDE

PURPOSE:
Process PFD horizontal and vertical modes as dictated by the pilot's mode control panel.

DESCRIPTION:
This procedure serves as the interface between the PFD format and the pilot's mode control panel. The horizontal and vertical guidance modes selected on the mode control panel are reflected in various types of color coded PFD text and symbology. The three light colors which may appear on the mode control panel are blue, amber, and green. They signify that the various modes are preselected, armed, or engaged respectively. The associated PFD format symbology follows the mode contol panel color coding to provide clear correspondence between the cockpit readout devices.
GUIDE performs its processing in the five steps listed below.

. Process the engaged vertical axis mode
. Process the armed vertical axis mode
. Process the preselected vertical axis mode
. Process the engaged horizontal axis mode
. Process the armed horizontal axis mode

Note that there are not any preselected horizontal axis modes.
The remainder of this module description shows the various modes available in the horizontal and vertical guidance axes. The PFD symbology affected by the guidance modes is also included.

VERTICAL AXIS

GUIDANCE MODES
Auto land flare
Auto land glideslope (GPS)
Auto land glideslope (MLS)
Auto land glideslope (ILS)
Vertical path
Altitude selection
Flight path angle selection

PFD SYMBOLOGY
- Glideslope bug and scale
- Vertical path deviation pointer
- Flight path angle reference bar
- Selected altitude pointer box
- Selected altitude readout
- Vertical mode readout (both armed and engaged)

HORIZONTAL AXIS

GUIDANCE MODES
- Auto land localizer (GPS)
- Auto land localizer (MLS)
- Auto land localizer (ILS)
- Horizontal path
- Track selection

PFD SYMBOLOGY
- Localizer bug and scale
- Horizontal path deviation pointer and scale
- Horizontal mode readout (both armed and engaged)

GLOBAL REFERENCES:

VARIABLES

ALTARM ALTCOR ALTREF* ALTRFV* ALTSEL ALTSUM AUTO BETAH DLBS DVBS ETAH FPABAR* FPAREF* FPASEL FPASUM FPDIAL* GPSM GSA GSE GSREF* HER HORARM HORPTH HORREF* HRAD HRZARM* HRZENG LAND LOCE LOCREF* MLSM PSTALT PSTFPA SELFPA TKSEL VDISC VERARM VERPTH VRTARM* VRTENG VRTREF*

FUNCTIONS AND SUBROUTINES

ALT_CNVRT SCLXTK

MODULE NAME: SCLXTK
FILE NAME: GUIDE.FOR
PROCESS: DSPFST
CALLED BY: GUIDE
CALLING SEQUENCE: INTEGER_VALUE = SCLXTK()

PURPOSE:
To compute the horizontal deviation pointer position.

DESCRIPTION:
SCLXTK determines the full scale width, in feet, of the horizontal deviation indicator and computes the appropriate position of the pointer. By default the horizontal deviation scale represents 7,500 feet in either direction from the center. When the aircraft is close enough to the touchdown runway's localizer shack the full scale width varies to match a localizer angular deviation. At runway threshold the width is 350 feet. The distance along the path where the width reaches the 7,500 foot maximum is dependent on the distance of the localizer shack to the runway threshold. Once the scale width has been determined the horizontal deviation pointer position is computed using the aircraft cross-track value (XTK).

GLOBAL REFERENCES:

VARIABLES
ACTCNT DTOGO RWYLEN TOWPT XTK

ARRAYS
AIRPTS

RECORD ARRAYS
ACT_WPTS

FUNCTIONS AND SUBROUTINES
XLIM

MODULE NAME: MSGMGR
FILE NAME: MSGMGR.FOR
PROCESS: DSPFST
CALLED BY: PFD_NASA
CALLING SEQUENCE: CALL MSGMGR

PURPOSE:

To display messages to the screen alerting the pilot that certain conditions exist.

DESCRIPTION:

MSGMGR contains the logic for the alert and warning messages displayed on the PFD screen. The messages, each of which consists of up to seven characters, are displayed in the lower half of the PFD window in large yellow letters. MSGMGR is executed twenty times per second, but only three messages may appear in one second. Therefore, every seventh iteration a valid warning or alert message may be shown. The text is displayed for its one-third second time frame. Priorities are given to the alert and warning indicators. The three valid messages with the highest priority are displayed in the three time slots available in one second. Each second conditions are re-checked, and again the three valid messages of highest priority are shown, so that some indicators which are also valid may not appear. On the other hand when there is no text to be displayed that area of the PFD is blanked.

A priority has been given to the messages such that warning indicators are output before alert indicators. The priority of warning text is as follows: 'AOA', 'FLAPS', 'GEAR', 'D/H', 'OM', 'MM', and 'LNK MSG' in that order. The priority for alert text is: 'ALERT', 'CLIMB' or 'DESCEND', 'TUNRL' or 'TURNR'. All of the messages except 'AOA' require that the message bezel be pressed on before they are displayed. The 'AOA' message will automatically appear in red when the maximum alpha currently computed by the flight controls software is exceeded and ground speed is greater than 64 knots.

The warning indicator 'FLAPS' has the second highest priority. It may be shown in either red or yellow. A red message indicates that either the flap handle positions for the forward and aft flight decks do not agree, or the flap settings are outside the structural limits of the airplane at the current airspeed. A yellow message will appear if the flaps are not set correctly for the current airspeed, or if the flaps are set at 40 and the airplane weight is greater than 95,000 pounds. Only the 'AOA' and 'FLAPS' text can appear in red, all other messages are shown in yellow.

The next two indicators processed are the gear and decision height warnings. If the runway bezel has been pressed on, and the flaps are set higher than 15, and the landing gear is not locked down, the text 'GEAR' is flashed. This message indicates to the pilot that the landing gear needs to be lowered. The warning 'D/H' appears when the radar altitude moves within the 30 foot range above the decision height, provided that the decision height is above 45 feet. The text 'D/H' is displayed for ten seconds.

Outer markers and mid markers are both physical positions on the ground located near runways, and are used by the pilot on runway approaches to verify his position. The text 'OM' or'MM' is displayed when the airplane passes over one of the markers, and its radio signal has been sensed.

The lowest priority warning message is 'LNK MSG'. This message indicates that a data uplink has been received, and is a cue to the pilot to check the data link display for the uplinked information.

The next set of indicators is the alert messages. They have lower priority than the warning messages, so that if all three time slots within one second have been filled already, no alert messages will be shown even if they are valid. Alert messages appear when the airplane is close to the next waypoint and they describe what general direction the airplane will take after it passes the waypoint. The actual logic to display alert indicators is: the alert flag must be on (it comes on ten seconds before the next waypoint and goes off after reaching it), the runway is not being shown, and the approaching waypoint is not the last one.

The text 'ALERT' has highest priority among the alert messages. The 'CLIMB' indicator is valid if the airplane altitude increases by 10 feet on the next flight path leg, and 'DESCEND' is valid if it decreases by 10 feet. The 'TURNL' indicator is true if the airplane turns to the left more than 2 degrees on the next leg, and 'TURNR' is true if it turns to the right more than 2 degrees. If there is a time slot available and alert messages are valid, 'ALERT' will always appear. The other alert messages are shown if there is room to show all that are valid. In other words, if ALERT/CLIMB/TURNL are all true but only two time slots are available, 'ALERT' only is shown because 'CLIMB' does not by itself adequately describe the next flight path leg. Also, once it has been determined that there is enough slots to show a complete set of alert messages, all warning messages are suppressed for the rest of that second (because warnings have higher priority and may interrupt the alert message being shown) to allow the alert messages to be displayed.

GLOBAL REFERENCES:

VARIABLES
  ACTCNT AEEF ALRTFG* CASF DECHT DLINK_FLG FLAP FLAP_LIMITS
  FLPPLC GRPOS HORPTH HRAD MIDMRK MSGBZL MXALF NAV64K OUTMRK
  RWYBZL RWYVLD TOWPT VERPTH WEIGHT WPTALR

RECORD ARRAYS
  ACT_WPTS

MODULE NAME: PACK
FILE NAME: PFDPK.MAR
PROCESS: DSPFST
CALLED BY: PFD_NASA
CALLING SEQUENCE: CALL PACK

PURPOSE:

To pack boolean values into one discrete word and send it along with the decision height value to the PFD format.

DESCRIPTION:

This procedure stores four items into the display output buffer for transmission to the PFD format. Twelve boolean values are packed as one bit discretes into a word of the display buffer. Three display buffer words are packed with twelve nibble (four bits) fields. The bit and nibble assignments are shown below.

PACKED DISCRETE WORD

BIT	VARIABLE	DESCRPTION
0	IATTV	Attitude valid
1	VVMODE	Velocity vector mode
2	ALTHLD	Altitude hold mode
3	STRVLD	Waypoint star valid
4	TRKBGF	Track bug flag
5	RWYVLD	Perspective runway valid
6	ALTRFV	Altitude referenve valid
7	HRV	Radar altitude valid
8	CASV	Calibrated airspeed valid
9	CALTV	Altitude valid
10	FMTVLD	PFD format valid (constant on)
11	TKSEL OR PSTTKA OR HORPTH	Horizontal guidance valid

NIBBLE WORD #1

CTRLMD	Control mode index
AUTHMD	Auto throttle mode index
RLLFLG	Roll command detent index
PITFLG	Pitch command detent index

NIBBLE WORD #2

RWYFLG	Perspective runway approach index
CASFLG	Selected CAS color index
HRZENG	Engaged horizontal mode index
HRZARM	Armed horizontal mode index

NIBBLE WORD #3

VRTENG	Engaged vertical mode index
VRTARM	Armed vertical mode index
FPABAR	Flight path angle bar color index
ALRTFG	Alert message index

The digital value of the right bezel panel potentiometer is fetched from the display input buffer to compute decision height. The address contained in PFDBZL points to the input buffer area corresponding to the pilot's PFD. The range of values obtained from the pot is 0FFFH (all the way left) to 0000H (all the way right). The raw pot values are translated into decision height values (feet) by the equation:

DH = .243956 * (4095 - POT_VAL)

The 'ones' digit is forced to zero to give decision height in increments of 10 feet. Note that a POT_VAL of 0FFFH produces a result of -30 feet. In actuality negative values are not obtained because none of the potentiometers provided ever reach the full 12 bit range. Typically the digitized value obtained from the pots on any DU bezel panel will only be 0F70H when turned completely to the left. This value corresponds to a decision height of zero feet.

GLOBAL REFERENCES:

VARIABLES
RWYVLD STRVLD TRKBGF VVMODE FMTVLD CASV HRV MACHV LOCVLD
CALTV IATTV PFDBZL

ARRAYS
OUTDAT*

MODULE NAME: PFD_NASA
FILE NAME: PFD_NASA.FOR
PROCESS: DSPFST
CALLED BY: DSPFST
CALLING SEQUENCE: CALL PFD_NASA

PURPOSE:

To serve as the display MicroVAX executive module to the NASA primary flight display format.

DESCRIPTION:

This module is the executive procedure for the NASA primary flight display format. Several small PFD computations are performed directly within this module, while individual procedures are called to handle larger operations associated with the PFD format.

Setting the logical value of the waypoint alert flag is the first operation performed. The flag is set when the aircraft is within ten seconds of the destination waypoint (straight leg or DMA turn), or the inbound tangent point (non-DMA turn), while traversing the active flight plan. The aircraft must be within 500 feet of the horizontal flight plan profile to be considered on the flight plan.

Four memory locations in the microprocessor output buffer are filled next. These are the actual roll, commanded roll, actual altitude, and vertical speed. they are computed from the flight controls variables ROLL, DROLL, ALTCOR, and HDCF respectively. The two roll variables are simply scaled and stored as fixed point values. The altitude is converted to a sign adjusted fixed point value by the function ALT_CNVRT. The fixed point vertical speed sent to the PFD is an exponential scaling of the altitude rate from flight controls. The screen units of length, in one-thousandths inches, for the altitude rate arrow is produced as follows.

(809.47353 * HDCF) ** .65

The flight control variables PDCOL and WHLINP are tested to determine the "out-of-detent" status of the pilot's side arm controller. Display buffer indices are set to indicate when roll and pitch commands are in progress.

The orientation of the display screen depends on modes of guidance used by the aircraft. The display screen is oriented along the aircraft "velocity vector" when the aircraft is flying velocity control wheel steering (VCWS) or automatic guidance with a manually selectable flight path angle (FPASEL). The introduction to this section describes this concept in more detail. The flag VVMODE is set to true or false by PFD_NASA to indicate which orientation will be used.

The variables FPAPFD and THETA_PFD represent the flight path angle and pitch angle used by the PFD format. The table below shows the settings of these variables under various conditions.

Condition	FPAPFD set to
. VCWS	Commanded flight path angle from control column.
. FPASEL	Selected flight path angle from AGCS mode control panel.
. OTHER	Actual flight path angle quickened by the rate of change of pitch.

Condition	THETA_PFD set to
. ACWS	Commanded pitch from control column.
. OTHER	Actual pitch angle.

The display of "Angle of Attack" and "Yaw" are then calculated by the equations:

```
AOA = FPAPFD - THETA_PFD
YAW = TRACK - HEADING
```

These values are translated into the aircraft body axis by performing a coordinate transformation with the aircraft roll angle. The following is a summary of how the values described above are used in the positioning of the "Gamma Wedge" and "Aircraft" symbols. Note some of the operations mentioned take place in the routine AIRGAM.

In Velocity Vector mode:
The Gamma Wedge symbol is fixed at the center of the display screen.

The center of the Pitch grid background is displaced from the Gamma Wedge by FPAPFD. Therefore the Gamma appears at the correct pitch grid marker.

The Aircraft symbol is drawn relative to the Gamma wedge by using the transformed angle of attack and yaw.

Not in Velocity Vector mode:
The Aircraft symbol is fixed at a position .8375 inches above screen center.

The center of the Pitch grid background is displaced from the Aircraft symbol by THETA_PFD. Therefore the aircraft symbol appears at the correct pitch grid marker.

The Gamma Wedge is drawn relative to the Aircraft symbol by using the transformed angle of attack and yaw.

Two transformation matrices are created in PFD_NASA for use by other procedures. The first is the TL2B matrix which transforms a 3D vector from geographic to aircraft body coordinates. The second, TB2E, changes body to velocity vector coordinates. See the descriptions of the procedures STAR and RWYMGR for more information on the use of these matrices.

Finally, nine PFD format procedures are called to generate the data for the PFD format in the microprocessor system.

GLOBAL REFERENCES:

VARIABLES
ACTALT* ACTROLL* ACWS ALPHAX ALTCOR AUTO BETAX CALPX CBETX CROLL DROLL DTOGO FPAPFD GAMC GAMMA GSFPS GUID2D HDCF HDGF HDOT* NAV64K PDCOL PITCH PITCH_Y* PITFLG* RLLFLG* ROLL SALPX SBETX SKYPTR* SROLL TOWPT TRKF VCWS VVMODE WHLINP WPTALR* XTK

ARRAYS
TB2E* TL2B*

RECORD ARRAYS
ACT_WPTS

FUNCTIONS AND SUBROUTINES
AIRGAM ALT_CNVRT ANGL CASMGR GUIDE MSGMGR MTH$SIGN PACK RWYMGR SCOSD STAR UNPACK WINDOW

MODULE NAME: ALT_CNVRT
FILE NAME: PFD_NASA.FOR
PROCESS: DSPFST
CALLED BY: PFD_NASA
CALLING SEQUENCE: INT_VALUE = ALT_CNVRT(FP_ALTITUDE)

PURPOSE:

To return the fixed point altitude for the microprocessor display output buffer.

DESCRIPTION:

Altitude values, in feet, for the aircraft can reach above the maximum binary word value possible (32,767). The MicroVAX uses floating point data to work with altitudes to avoid problems. However, floating point altitude values above 32,767 will cause VAX conversion errors when reformatted to 16 bit integers for transmission to the displays. The function ALT_CNVRT alleviates this problem by converting the floating point altitude to a 32 bit longword integer and saving only the lower 16 bits. No MicroVAX conversion error occurs, however high altitudes become the binary equivalent of large negative numbers. This is accounted for by the PFD format itself. The "high altitude" values are treated as unsigned 16 bit integers by the microprocessors software.

MODULE NAME: RWYMGR
FILE NAME: RWYMGR.FOR
PROCESS: DSPFST
CALLED BY: PFD_NASA
CALLING SEQUENCE: CALL RWYMGR

PURPOSE:
To compute the screen coordinates of the perspective runway PFD symbology.

DESCRIPTION:
A number of conditions must be met before the perspective runway symbology will be shown on the PFD format. These include:

- Runway selected on PFD bezel panel.
- Destination runway defined on active flight plan.
- Airplane is within the ILS zone defined around runway approach.
- Aircraft heading differs from runway heading by less than 40 degrees.
- Altitude above runway is less than 2,500 feet.

The computation of runway parameters proceeds when all the above conditions are met.

Four different coordinate systems are employed by RWYMGR while performing the calculations. They are defined as follows:

GEOGRAPHIC: X - NORTH Y - EAST Z - UP

RUNWAY: X - RWY CENTER LINE Y - RWY THRESHOLD LINE
Z - UP

BODY: X - ALONG BODY Y - THROUGH WINGS Z - THROUGH BODY, PERPENDICULAR TO WINGS

INERTIAL: BODY AXES ROTATED BY (TRACK - HEADING) AND (FLIGHT PATH ANGLE - PITCH)

First the position of the aircraft relative to the runway threshold point is computed. The values obtained represent a distance vector with each element of the vector representing the distance between the aircraft center of gravity and the runway threshold point, projected onto the geographic reference axes.

The geographic distance vector is transformed to the runway frame of reference. The "X" component of the new vector is used to perform a "Psuedo-clip" operation on the runway coordinates. Performing 3D clip calculations for each vector used to draw the runway is very time consuming. An approximation using the "X" component of the runway referenced distance vector is used instead. The idea behind the "psuedo-clip" approximation is to throw away some of the runway as the airplane approaches. The amount to throw away is determined from the "X" and "Z" components of the runway reference distance vector (see figure 6.2 at the end of this description). This technique requires that the aircraft is approaching the runway down the centerline with small pitch and yaw angles. When these criteria are not met the runway coordinates, after transformation to the body frame of reference, may cause one of two problems. If too much runway is truncated the front of the runway may disappear on the screen instead of running below the bottom edge of the view screen. A worse situation occurs when not enough is truncated. The points that are transformed behind the aircraft center of gravity actually are reflected back ahead of the airplane. This will appear as spurious lines being drawn onto the display screen. Note the conditions governing the display of the runway symbology (mentioned above) along with standard flight procedures around the touchdown runway make the runway truncation a valid procedure.

A transformation matrix is created which changes runway coordinates to body reference. The global matrix TL2B (Transform Local to Body) is adjusted by runway heading to make the new transformation matrix. The local procedure SCREEN will use this matrix when performing its computations.

Figure 6.3, at the end of this module description, shows eight points representing the outline of the runway in the runway coordinate system. These points are sent to the local procedure SCREEN for transformation. The coordinates for each point are shown in the following table.

POINT	(X)...COORDINATES (feet)...	(Y)
1	0 (or truncation point)	-100
2	length of actual runway	-100
3	length of actual runway	100
4	0 (or truncation point)	100
5	-6076 (or truncation point)	0
6	1.0E+10 (infinity)	0
7	1000 (or not shown)	-100
8	1000 (or not shown)	100

The "Z" value for all points is implicity defined as runway elevation. Note that the threshold and touchdown lines need not be drawn when the truncate point advances past them. The index RWY_INDEX is used to pass this information to the display system.

RWY_INDEX	ACTION
0	Draw threshold and touchdown lines
1	Draw touchdown line
2	Draw neither line

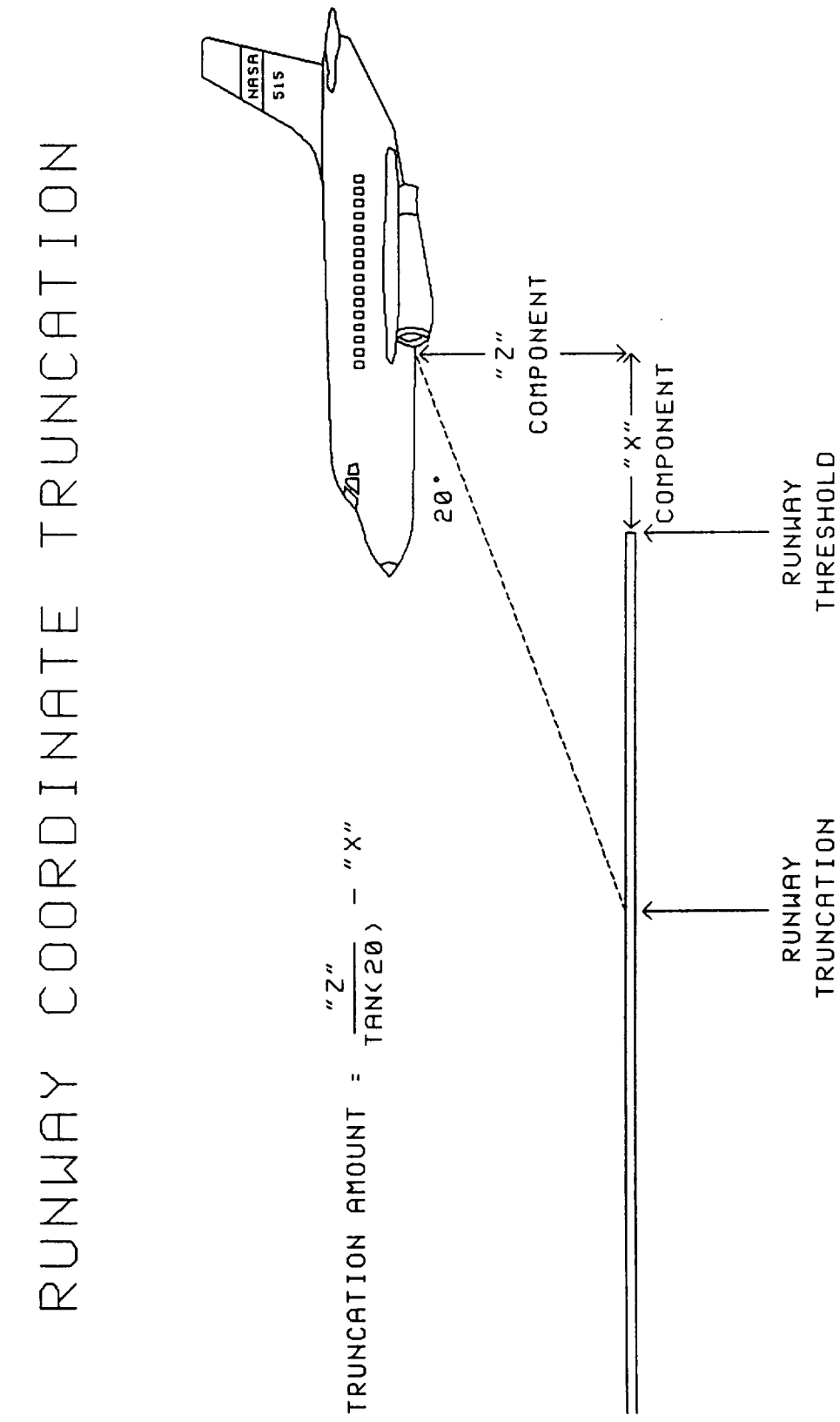
The procedure SCREEN is called to compute the X and Y screen coordinates, in one-thousandths inches, for the runway points. The screen coordinate values are sent to the display processors as 16 bit fixed point integers. The values sent during an approach to the runway vary by such a large amount that a single scale factor value for the coordinates could not be chosen. Instead the coordinates are scaled dependent on the largest value created by the procedure SCREEN. All values are multiplied by the computed scale factor before conversion to integer format takes place. The scale factor used is also sent to the display system, via the output buffer, for proper interpretation of the coordinate values sent.

GLOBAL REFERENCES:

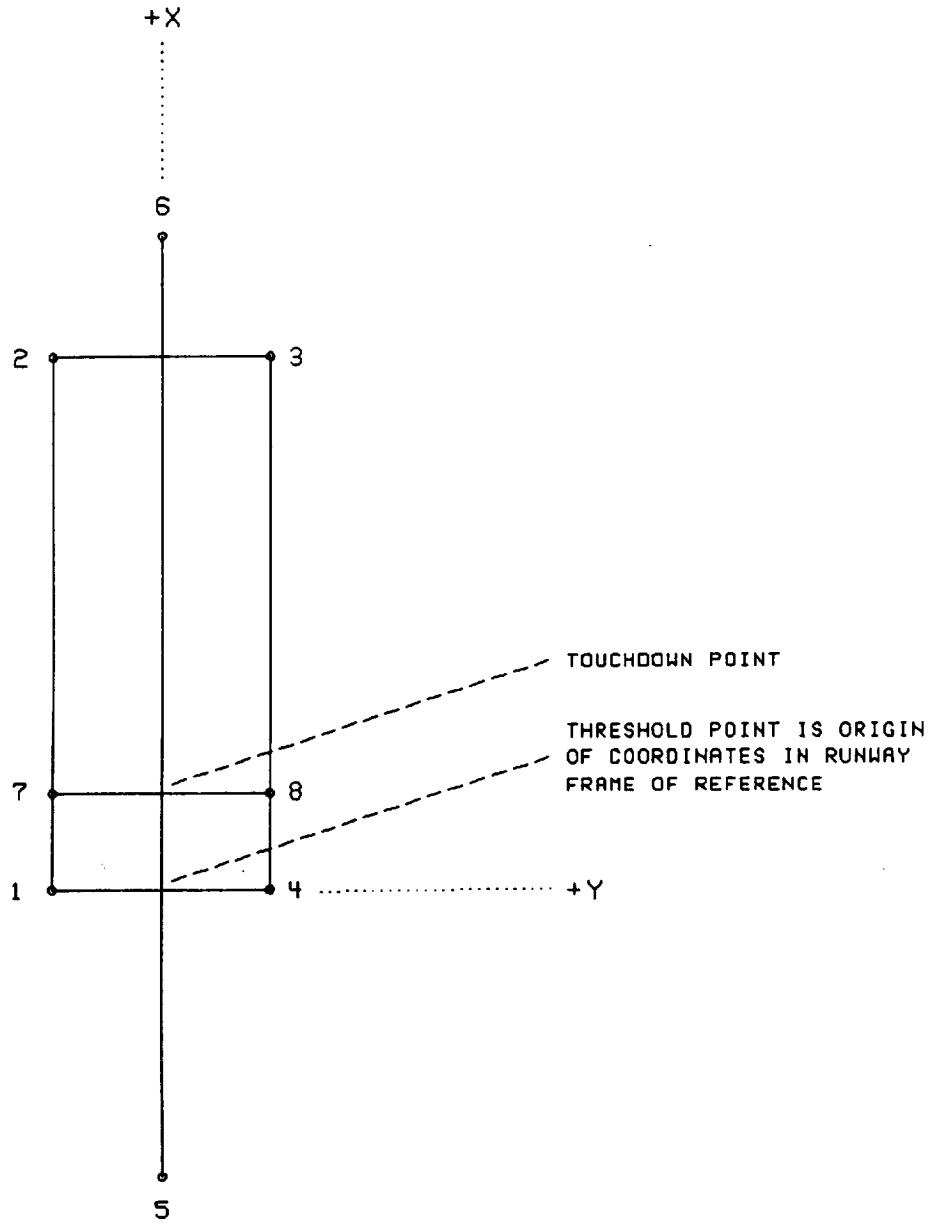
VARIABLES
ALTCOR COSRH DLATFT DLONFT DLT_ALT DLT_RWY_X DLT_RWY_Y* HDGF ILSZON INT_SCALE* LAT LON RWYBZL RWYFLG RWYHDG RWYLAT RWYLEN RWYLON RWYVLD* RYELEV SINRH TRKF VVMODE

ARRAYS
AIRPTS OUTDAT* RWY_TO_BDY* TL2B

FUNCTIONS AND SUBROUTINES
ANGL SCREEN


RUNWAY COORDINATE TRUNCATION
NASA
515
"Z"
COMPONENT
"X"
COMPONENT
20°
RUNWAY
THRESHOLD
RUNWAY
TRUNCATION
POINT
TRUNCATION AMOUNT = "Z" / TAN(20) - "X"

-figure 6.2-



RUNWAY COORDINATE POINTS

-figure 6.3-

 INTENTIONALLY BLANK

MODULE NAME: SCREEN
FILE NAME: RWYMGR.FOR
PROCESS: DSPFST
CALLED BY: RWYMGR
CALLING SEQUENCE: CALL SCREEN(RWY_X,RWY_Y,SCR_X,SCR_Y)

PURPOSE:
To compute screen unit coordinates on the PFD format for the perspective runway symbology.

DESCRIPTION:
The procedure screen receives the X and Y offsets of runway points as its first two calling parameters. These are values in feet in the runway coordinate system, centered on the runway threshold line. The position of the runway point relative to the aircraft is found by adding the aircraft offsets computed by RWYMGR (DLT_RWY_X and DLT_RWY_Y).

A vector is made that changes the coordinates origin from the runway threshold to the airplane position. This vector is transformed from the runway frame of reference to the airplane body frame. If a velocity vector mode of flight is in use the body coordinates are transformed to the system oriented in the airplane inertial frame of reference. The tangent of the pilot observation angles (up/down, left/right) are found by the ratio of the Z and Y coordinates with the depth coordinate (X). The PFD screen coordinates, in one-thousandths inches, are returned to RWYMGR through the last two calling parameters.

GLOBAL REFERENCES:

VARIABLES
DLT_ALT DLT_RWY_X DLT_RWY_Y RWYVLD* VVMODE

ARRAYS
RWY_TO_BDY TB2E

FUNCTIONS AND SUBROUTINES
MXV

MODULE NAME: SBXMGR
FILE NAME: SBXMGR.FOR
PROCESS: DSPSLW
CALLED BY: DSPSLW
CALLING SEQUENCE: CALL SBXMGR

PURPOSE:

To control and output to the PFD information concerning aircraft autothrottle and control modes, the 'TO' waypoint, and the current mach.

DESCRIPTION:

The upper left corner of the PFD is reserved for information concerning the aircraft's control and autothrottle modes. The text "AUTO", "VCWS", or "ACWS" may be displayed in the control mode slot (which is the line directly above the autothrottle text line) depending on the current flight state. If the aircraft is flying autothrottle then the following text can be shown in the autothrottle mode slot: "AT CAS" for autothrottle selected, or "AT 4D" when flying four dimensional guidance. For both modes, if the conditions are not met to display text then the slots remain blank.

The "to" waypoint name will appear in the upper right corner of the PFD whenever two dimensional guidance is possible. The name consists of five characters, and is retrieved from the active guidance buffer. When 2D guidance is not possible the area will be blanked.

SBXMGR also stores the current mach value in the display output buffer. It will appear above the airspeed tape on the PFD screen when it is .5 or greater. Mach is scaled by one thousand to provide three digits of accuracy on the display.

GLOBAL REFERENCES:

VARIABLES
ACWS AUTHMD* AUTO CTRLMD* GUID2D IASSEL MACH PFD_MACH*
PFD_TO_WPT* TIMPTH TOWPT VCWS

RECORD ARRAYS
ACT_WPTS

MODULE NAME: STAR
FILE NAME: STAR.FOR
PROCESS: DSPFST
CALLED BY: PFD_NASA
CALLING SEQUENCE: CALL STAR

PURPOSE:

To compute the screen coordinates and zoom factor for the waypoint "Star" symbol of the PFD format.

DESCRIPTION:

The waypoint "STAR" symbol on the PFD depicts a three dimensional representation of the current position of the destination waypoint on the active flight plan. When the pilot maneuvers the aircraft so that the track and flight path angle symbology lie on the STAR symbol, the aircraft is flying directly to the next waypoint on the flight plan. The size of the STAR remains constant, until the aircraft is close to the waypoint. Then the STAR zooms, to simulate a 3-D effect until the waypoint is reached.

First logical comparisons are made to determine if the proper conditions exist for displaying the STAR symbol. The following criteria must be true to proceed with the necessary calculations.

- A valid 3-D flight plan exists (altitudes for each waypoint exist)
- The "STAR" bezel button on the PFD display has been pressed.
- The 'TO' waypoint is not the last waypoint on the active flight plan.

The size of the STAR is sent to the display processor as a 16 bit integer scale factor named STARZM. The value represents the ratio of the original STAR size times 1024. For example, a value of 1024 for STARZM means display the normal sized STAR (1 * 1024). If a STAR 5.25 times as large as the normal size is desired, STARZM would be set to 5376 (5.25 * 1024). When the aircraft is within 3,500 feet the STAR will start to zoom (10 seconds before arrival at 210 knots). The amount of zoom is proportional to the aircraft's distance from the 'TO' waypoint. The STAR will grow to 20 times its original size before it is removed from the display screen until a new 'TO' waypoint can be shown.

The position of the STAR on the PFD format represents the track and flight path angle required to reach the 'TO' waypoint from the current position of the aircraft. This only corresponds to flight plan guidance when the current position of the aircraft is on the flight plan. The bearing to the destination waypoint is found using earth centered vectors pointing to both the aircraft position and the 'TO' waypoint. The normal vector to the plane formed by the earth centered vectors is compared to an "absolute west" vector to produce the desired bearing. Desired flight path angle is found using the distance to the destination waypoint and the altitude differential between the positions. The screen positions are found by rotating the angular values into the roll axis of the PFD horizon line and scaling the values to screen units (one-thousandths inches).

If the coordinates of the STAR symbol lie outside of the PFD view window, the coordinates are clipped so the STAR symbol is pegged to the side of the window by the subroutine LIMITS.

GLOBAL REFERENCES:

VARIABLES
ACTCNT ALTCOR CLON CROLL DTOGO FPAPFD GAMX GAMY GUID3D LAT LON SLON SROLL STARX STARY STARZM* STRBZL STRVLD* TK TOWPT VVMODE

ARRAYS
AIRPTS

RECORD ARRAYS
ACT_WPTS

FUNCTIONS AND SUBROUTINES
ANGL LIMITS MTH$ASIN MTH$ATAN MTH$ATAN2 VCP VDP VMG XYZ

MODULE NAME: LIMITS
FILE NAME: STAR.FOR
PROCESS: DSPFST
CALLED BY: STAR
CALLING SEQUENCE: CALL LIMITS(XPOS,YPOS,XLIM,YLIM)

PURPOSE:
To force the waypoint STAR within the boundaries of the PFD viewing window.

DESCRIPTION:
The first two calling parameters to LIMITS contain the computed offsets from the screen center of the waypoint STAR symbol. The values are limited to fall within the PFD viewport and returned in the last two calling parameters.

The limiting operation consists of calculations which determine the intersection of a line from the screen center to the STAR, and the viewport boundary. Tests are made to determine where the intersection will occur; viewport sides, bottom, or within the arc at the top. For the sides and bottom, which are either constant X or Y lines, the intersection is found from the fixed coordinate value at the boundary and the slope of the line. The arc intersection is more difficult. Both the slope of the STAR line and the arc radius are used as follows.

```
XLIM = +/- SQRT(ARC_RAD**2 / (SLOPE**2 + 1))
YLIM = SLOPE * XLIM
```

Note that the "+/-" means XLIM will follow the sign of the original unlimited X coordinate.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
MTH$SIGN MTH$SQRT

MODULE NAME: UNPACK
FILE NAME: PFDPK.MAR
PROCESS: DSPFST
CALLED BY: PFD_NASA
CALLING SEQUENCE: CALL UNPACK

PURPOSE:
To process packed discrete words.

DESCRIPTION:
UNPACK starts by creating individual boolean variables for each bit of the packed discrete word FCFLGS, which is a 32 bit longword received from the FM/FC MicroVAX computer. The following is a list of the seventeen logical byte variables formed from the individual bits of FCFLGS. Note that the variable ALTHLD is logically negated when unpacked from FCFLGS.

LOGICAL	BIT	DESCRIPTION
LOCE	0	localizer engage
ACWS	1	ACWS mode
VCWS	2	VCWS mode
AUTO	3	AUTO mode
LAND	4	LAND mode
MLSV	5	MLS valid
GEAR	6	gear flag
MLSM	7	MLS mode engaged
DVBS	8	vertical beam sensed
DLBS	9	lateral beam sensed
LANDR	10	land armed
GSE	11	glide slope engaged
TDSP	12	track dial spin
MXALF	13	AOA exceeded
ALTHLD	14	altitude hold of vcws
AEEF	15	aft flight deck engaged
GPSM	16	GPS guidance selected

The 16 buttons on the PFD display bezel panel each control one bit in a microprocessor input word. A bit is '1' while the button is depressed, otherwise it is '0'. The packed word containing the 16 bezel bits is sent to the host computer in the 320 word transmission buffer. UNPACK uses the addresses saved in the PFDBZL array to fetch the bezel words for both the pilot and co-pilot PFD formats. The bits are "one-shotted" to make long button presses appear as a one frame press. The memory word, OUTDAT(680), is used to keep the current state of each bezel button. Each time a press is detected the corresponding bit in OUTDAT(680) is toggled to the alternate state. This gives the effect of press to enable function, press again to disable function. Only the three bezel buttons on the upper right side of the display screen are used for the PFD format. Their usage is listed below.

C-2

LOGICAL	BIT	DESCRIPTION
STRBZL	0	bezel panel STAR select
RWYBZL	1	bezel panel perspective runway select
MSGBZL	2	bezel panel alert message select

The current state of the PFD bezel buttons is also sent back to the PFD since OUTDAT(680) is part of the buffer that is transmitted to the display system.

GLOBAL REFERENCES:

VARIABLES
LOCE* ACWS* VCWS* AUTO* LAND* MLSV* GEAR* MLSM* DVBS*
DLBS* LANDR* GSE* TDSP* MXALF* ALTHLD* AEEF* GPSM*
STRBZL* RWYBZL* MSGBZL* FCFLGS

ARRAYS
PFDBZL

MODULE NAME: WINDOW
FILE NAME: WINDOW.FOR
PROCESS: DSPFST
CALLED BY: PFD_NASA
CALLING SEQUENCE: CALL WINDOW

PURPOSE:

Compute position data for symbology used within the PFD format viewing window.

DESCRIPTION:

This procedure sends information to the PFD for four different symbols. These are the horizon ticks, desired track marker, track hold indicator, and the flare guidance cue.

The horizon ticks are a set of tick marks appearing at ten degree intervals along the horizon line of the PFD format. Depending on mode of operation, the center of the PFD view window is either the aircraft's current track or heading. Tick marks are all placed at even ten degree units of bearing. The offset to the first tick is computed by WINDOW, in one-thousandths inches, and stored in the microprocessor display buffer.

The desired track "T" bar rides along the top edge of the PFD horizon line. The bar is placed at the position corresponding to either the flight plan desired track, or the selected track from the pilot's mode control panel. When the horizontal path guidance is engaged the desired track of the flight plan is stored by WINDOW. Since the mode panel selected track is always stored by navigation display modules, WINDOW does not need to do so.

When track hold mode of Velocity Control Wheel Steering (VCWS) is engaged the commanded track "bananas" are shown on the lower edge of the PFD horizon line. Their position is obtained by scaling the offset value created in the flight controls software.

The last item processed by WINDOW is the flare guidance cue. The screen position of the cue, in one-thousandths inches, is computed when VCWS mode is engaged and radar altitude is less than either selected decision height or 200 feet. The computed position is proportional to radar altitude. The guidance cue will meet the PFD horizon line when altitude above the runway is zero. The rate that the cue approaches the horizon line is .05 degrees per foot of altitude.

GLOBAL REFERENCES:

VARIABLES
DECHT DSRTK FLARE* HDGF HORPTH HORTCK* HRAD MAGVAR
NOMTRK* PITCHY TKBUGX* TRKBG TRKBGF* TRKF VCWS VVMODE

FUNCTIONS AND SUBROUTINES
ANGL MTH$AMOD

Section 7.0 NAV DISPLAY SOFTWARE

The navigation display shows the position of the airplane relative to ground positions and terrain features. Figures 7.1 and 7.2 at the end of this section depict typical NAV display configurations for the two available map background orientations (Map and Plan modes). The map orientation is a selectable feature controlled by a bezel button.

The airplane chevron in Map mode is fixed 1.25 inches below the screen center and the symbology is displayed in a "track up" orientation. This means that the airplane's current track is always the center of the compass arc at the top of the display.

In Plan mode some other reference point, typically a waypoint, is used for the fixed position 1.25 inches below screen center. All other map background symbology is displayed in a "north up" orientation. The airplane chevron symbol moves about the display in this mode.

The NAV format screen is divided into three distinct areas. The major one is the airplane and map background features area. The second is the flight information area at the top of the display above the compass arc. The vertical deviation scale area is the third. The three areas exist in both Map and Plan modes. The compass arc, which forms a partition between background and flight information areas in Map mode, is not shown in Plan mode however.

The flight information area contains wind speed and direction information in the upper left corner in Map mode. Just a north pointer is displayed there in Plan mode. The right hand side of this area has four lines of flight information text. The following describes the information present on each line.

#1 'TO' waypoint information. Includes destination waypoint name and distance from present airplane position. Also the current Greenwich Mean Time is shown at the end of the line.

#2 Bezel selection indicators. Three letter mnemonics appear when bezel buttons have been used to select the map options; airports, navaids, time box, or altitude range arc.

#3 Like line #2, mnemonics are shown when terrain features, ground reference points, or the boundaries of restricted regions have been selected.

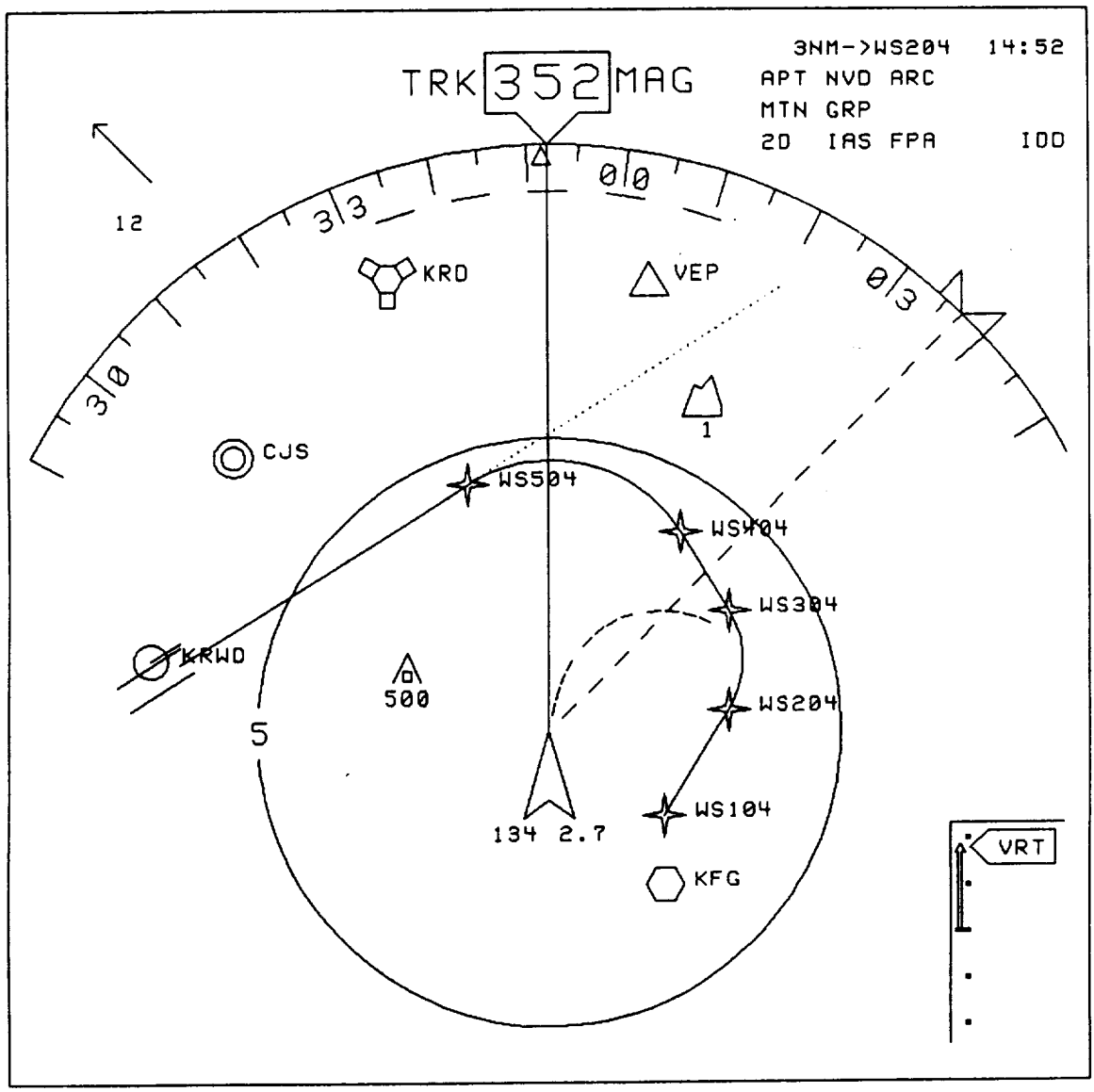
#4 Guidance and navigation modes. Aircraft modes include 2D/3D/4D guidance, track select, altitude hold, flight path angle select, and air speed select. The aircraft navigation modes are shown at the end of the line.

The aircraft and map background area shows the aircraft's present position along with selected reference points. The orientation of the symbology depends on which of the two modes is selected from the bezel panel. Flight plans are displayed as a series of straight and curved line segments with four point star symbols designating specific waypoints. The aircraft chevron has the current ground speed and altitude appended to the bottom when in Map mode. A list of the symbols that can appear in the Map background area follows in the next sections. The symbology appearing in this area is masked from the other two NAV format areas. However the masking is only performed for the vertical deviation area when the vertical deviation scale is present on the display.

The altitude profile of the aircraft is shown via the vertical deviation scale. This scale appears in the lower right hand corner of the display screen when appropriate conditions arise. A rectangular mask around the scale keeps background features from interfering when the vertical deviation scale is present.

The NAV format uses twelve of the sixteen buttons provided on the bezel panel of a DU. Neither of the two potentiometer dials is used. The following table describes the function of each NAV format bezel button. The naming conventions are: L - left hand set, R - right hand set, (1 through 8) - button number with one being the top.

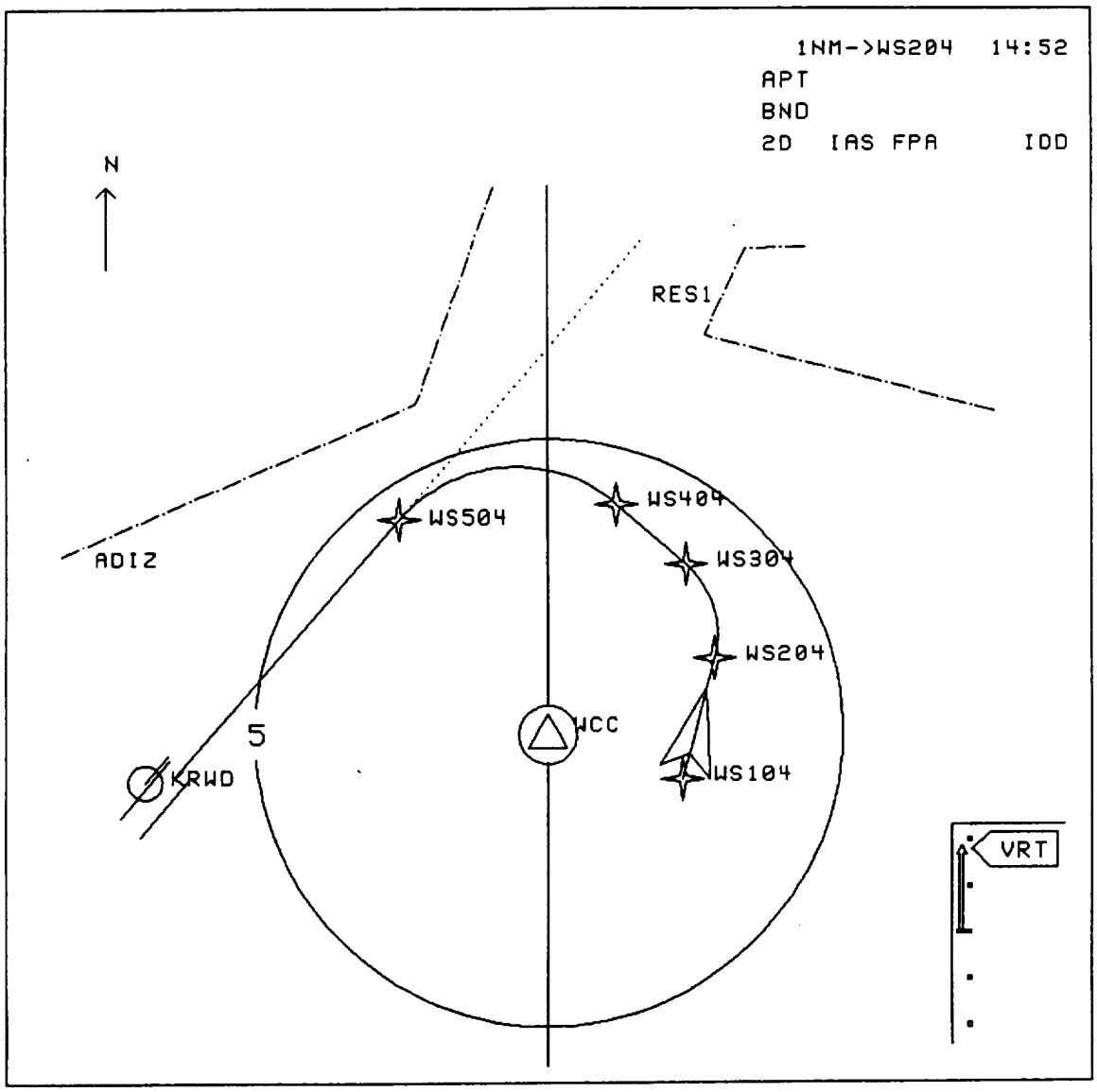
L1	not used
L2	not used
L3	not used
L4	Terrain features option
L5	GRPs option
L6	not used
L7	ADIZ boundaries option
L8	Map/Plan mode toggle
R1	MLS select/deselect
R2	Airports option
R3	Navigational aids option
R4	Time box option
R5	Altitude range arc option
R6	Path waypoint information option
R7	Zoom out (scale change)
R8	Zoom in (scale change)



NAVIGATION DISPLAY FORMAT

(MAP MODE)

-figure 7.1-



NAVIGATION DISPLAY FORMAT
(PLAN MODE)

-figure 7.2-

Section 7.1 THE NAV BACKGROUND BUFFER

The background data buffer is the first 400 words of the 704 word buffer sent to the Sperry display system from the host computer. Appendix A gives the layouts of the I/O buffers. The background data occurs in varying amounts depending on the position of the aircraft on the chosen route, and contains information on the reference points and flight plan. Unlike the rest of the data sent to the microprocessor system, the background data is a variable length stream, up to 800 words, of information which cannot be identified by its I/O buffer position. Interpretation of the data must be performed with a sequential parsing, starting at the first location. Special code words are used within the data stream to identify what type of background information follows.

Each NAV format loaded into a display processor may have unique background data. However the same 400 word data area is used by all NAV formats, therefore they must take turns using the background data buffer. Actually many of the 704 word data buffers read from the host computer contain no new data in the background area since background updates occur infrequently (every five seconds, after a NAV bezel button selection, or following a flight plan modification).

A word of data, "OUTDAT(599)" in Appendix A, is used to control the use of the background buffer. The meaning of the bits in the Map Control word are as follows:

BITS	DESCRIPTION
0-3	Display processor identifier code.
4	Part two update flag.
5-7	Unused.
8-15	Update sequence number.

The display processor identifier code is a number that designates the processor which should use this data. The next chart gives the code values used for the various processors.

CODE	DEU #	DP #
1	1	1
2	1	2
3	1	3
4	2	1
5	2	2
7	3	1
8	3	2
9	3	3

The part two flag informs the NAV format that the background buffer is to be considered a continuation of the last one received. This gives the ability to send 800 words of background data to a NAV format in two consecutive updates. The sequence number informs the NAV format that the data in the background buffer is new data. This number increments every time a new background buffer is created. This allows the NAV format to distinguish between new data designated for it and old data that has been in the buffer for a while.

All data within the background buffer is stored in groups of 2 or more words. The first word of the group is always a "group identifier word" having the group ID label, class, and word count. The diagrams at the end of this section show the various formats of background data words. Bits zero through two of the group ID word are used for the label. Presently only the following five of the possible eight group types are used.

LABEL	USAGE
0	Control group
1	Text group
2	Symbol group
3	Rotatable symbol group
4	Line segment group

The class code, bits three through seven, is used to differentiate between variants of the given group. The upper eight bits of the group ID word are the word count. The value in this byte is the total words associated with the particular data group following the ID word.

The control group is used to set up map display software prior to the processing of the remaining background data. Two classes exist in this group; start of transmission (SOT) data and mode controls. The SOT class is always the first data in the background buffer. The total number of words used in the buffer for the entire background update is stored along with the map sequence index. The index is used to pick the correct map center displacement out of the array of four sent from the host computer. Four map center values exist because up to four independent NAV formats may run simultaneously. The second class, mode controls, sets map scale, enables selectable symbology (weather radar, time box, range arc), and chooses map orientation (track up/north up).

The text group controls the placement of ASCII data on the four available text lines appearing at the upper right corner of the NAV format. The class value selects which line data is written to.

Reference position symbols are placed on the display by symbol group entries. The data for this group contains the north and east displacements from the map center and optional descriptive text to be shown with the symbol. The type of symbol (DME, GRP, AIRFIELD ...) is selected by the class code.

Symbols which require a particular orientation on the display screen are handled by the rotatable symbol group. The bearing and length (for runway only) is stored in the data group in addition to the position and text data.

The final group currently used is the line segment group. The class field of this group's ID word sets the line type and color for the following sequence of data. The two words after the identifier word are the north/east position values from map center for the line segment initial point. The remaining data in this group occurs in blocks of two or more words and is very similar to the group technique described above. Each block contains one element of the connected line/path segments being created. The first word of these blocks is the path element ID, containing the element type and word count. The element types are line, arc, on path waypoint, and off path waypoint. The line element commands a line to be drawn from the last path position to the position designated by the north/east coordinates provided. The arc type gives the subtended angle, initial inbound angle, and the arc radius. The on path waypoint element places a waypoint symbol at the last defined path position. Off path waypoints have their own north/east coordinates for placement.

The symbology created from the background data is repositioned on the display screen every 50 milliseconds by the airplane position and track data received from the I/O buffer. This allows the airplane to move smoothly over the map data. The only requirement is that the background be updated at a fast enough rate so that the display screen always covers the area defined by the background.

A sample background data buffer is shown below. The values shown are the binary words presented in hexidecimal format. The data produces a simple flight plan consisting of two straight segments, one arc segment, and a waypoint symbol entitled "COLIN". Text data for lines #1 and #4 also appear in the data. The diagram on the following page shows the path drawn from this data. Note the overlap of the physical viewing window which allows displacing the background from the physical screen center as the airplane flies.

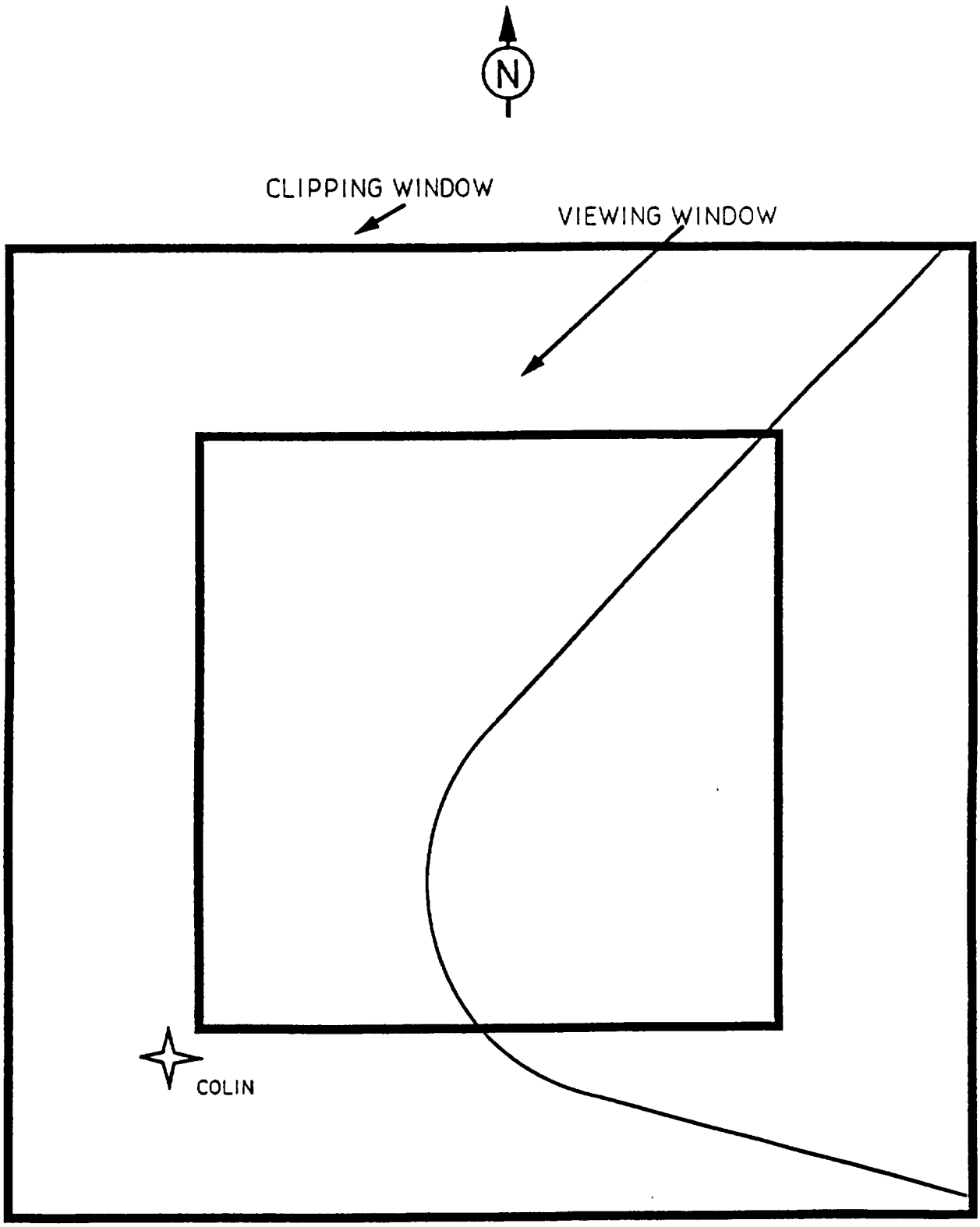
```
0100        ; SOT.
0039        ; 57 words background update.

0108        ; controls.
0001        ; scale is 5NM radius.

1204        ; line segments - 18 words.
11FE        ; initial position: 4.606 inches east
1307        ;                   4.871 inches north.
0200        ; draw line to:
FFF2        ;                .014 inches west
FFE3        ;                .029 inches south.
0301        ; draw arc with:
5977        ;                125.8 degree turn
F61D        ;                2.531 inch radius (left turn)
9F11        ;                -136.3 degree inbound bearing.
0503        ; place off path wpt at:
F298        ;                          3.432 inches west
F1E9        ;                          3.607 inches south
4F43        ;    text: "COLIN "
494C        ;
204E        ;
0200        ; draw line to:
1307        ;                4.871 inches east
ED81        ;                4.735 inches south.

0A01        ; text line #1.
2020        ; "  7NM->COLIN  12:32 ".
4E37        ;
2D4D        ;
433E        ;
4C4F        ;
4E49        ;
2020        ;
3231        ;
333A        ;
2032        ;

0A19        ; text line #4.
4433        ; "3D  IAS          IXX ".
2020        ;
4149        ;
2053        ;
2020        ;
2020        ;
2020        ;
2020        ;
5849        ;
2058        ;
```



SAMPLE PATH DATA

-figure 7.3-

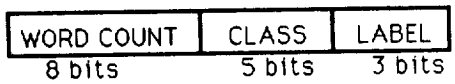
BACKGROUND BUFFER LAYOUT

Buffer consists of 5 data groups:

Label	Title
0	Control Group
1	Text Group
2	Symbol Group
3	Rotatable Symbol Group
4	Line Segment Group

Group identifier words always precede data belonging to a specific group.

Group Identifier Word:



Where:

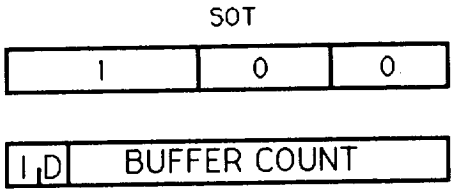
- Word count = # of words in data group following identifier.
- Class = identifies subgroup items, line types, colors, etc.
- Label = group number 0 - 4 listed above.

Notes on units for following pages:

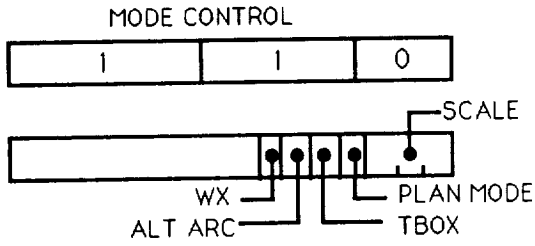
-- all distance values : 1/1000th inches.
-- all angular values: (degrees/180) * 2**15.
-- all text: standard ASCII codes in consecutive byte order.

-figure 7.4-

LABEL 0 CONTROL GROUP:



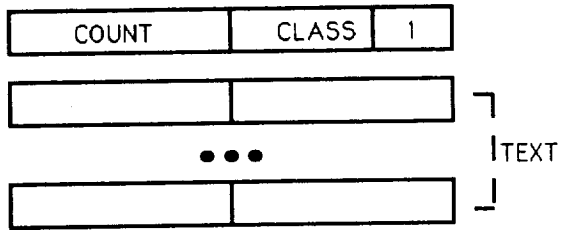
Scale	Range (nm)
0	2
1	5
2	10
3	20
4	40
5	80



LABEL 1 TEXT GROUP

CLASS

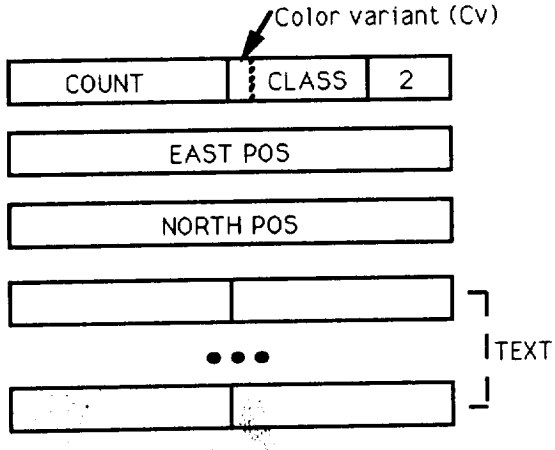
0: Text line 1
1: Text line 2
2: Text line 3
3: Text line 4



LABEL 2 SYMBOL GROUP

CLASS

0: Mountain (Cv ignored)
1: Obstruction (Cv ignored)
2: Non-directional Beacon (Cv ignored)
3: VORTAC (Cv ON = tuned)
4: VOR (Cv ON = tuned)
5: DME/TACAN (Cv ON = tuned)
6: Airfield (Cv ignored)
7: GRP (Cv ignored)
8: Marker Beacon (Cv ignored)
9: Waypoint (Cv ON = provisional)

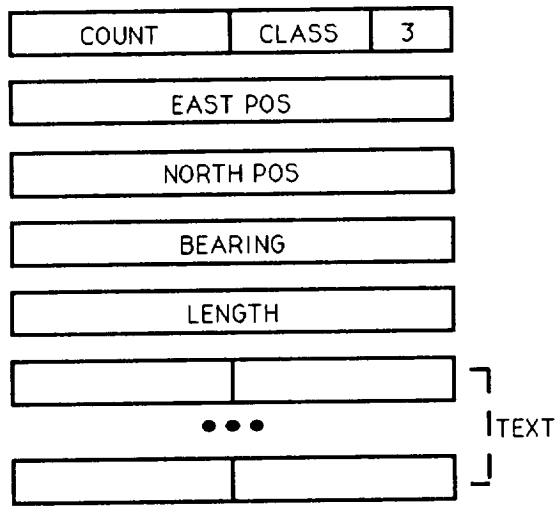


PRECEDING PAGE BLANK NOT FILMED

LABEL 3 ROTATABLE SYMBOLS GROUP

CLASS

0: AIRFIELD (Length ignored)
1: Runway
2: SRP (Length ignored)



LABEL 4 LINE SEGMENT GROUP

LINE COLOR

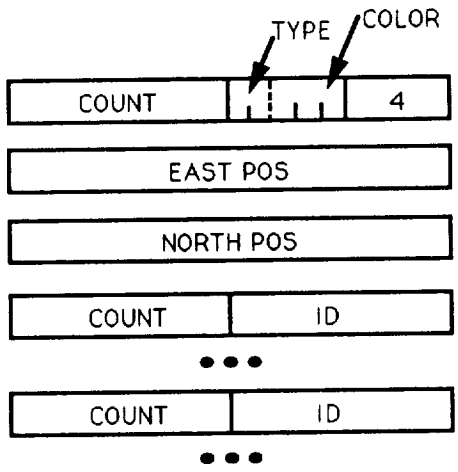
0: White
1: Green
2: Blue
3: Amber
4: Red
5: Cyan
6: Magenta
7: Yellow

LINE TYPE

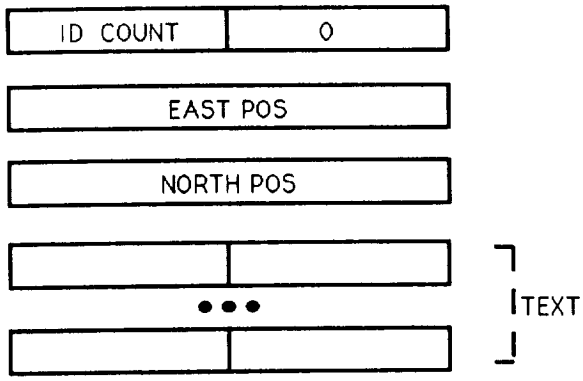
0: Solid
1: dot
2: Dash
3: Dot/Dash

PATH SUBGROUP ID:

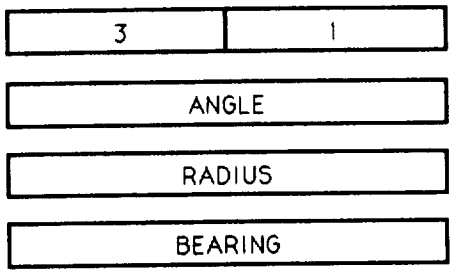
0: Position
1: Arc
2: Wpt (on path)
3: Wpt (off path)



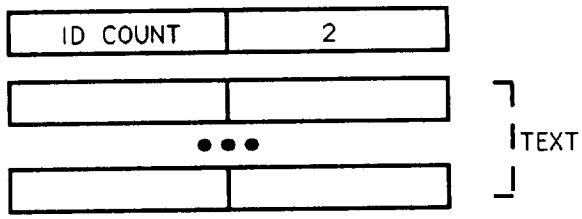
ID 0: Position



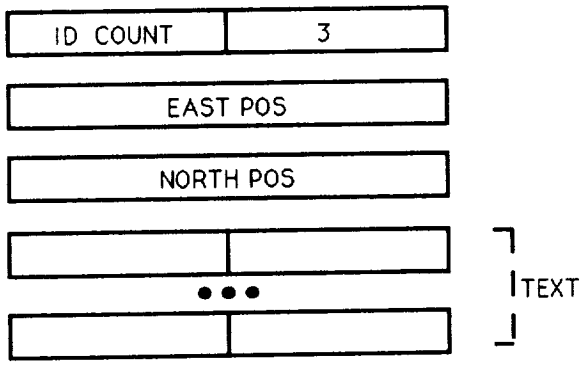
ID 1: Arc



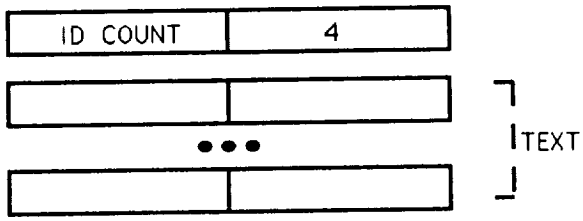
ID 2: WPT (ON)



ID 3: WPT (OFF)



ID 4: TEXT



Section 7.2 NAV BACKGROUND UTILITIES

The data stored in the Navigation format background buffer is created by calls to a set of utility procedures. All the modules described in section 7.3 make calls to these utilities to place graphic information into the map background. The set of procedures is small, totaling only four percent of all navigation format software.

There are two categories of NAV background utilities. The first group contains procedures which are called to create NAV symbology independent of other utility calls. The second group of procedures are called in sets. Each set starts with a BEG_SEG call and is terminated by a END_SEG call. The NAV background utility procedures are listed below. Detailed module descriptions are provided on the following pages.

Group #1
- NAV_TEXT
- NAV_SYMBOL

Group #2
- BEG_SEG
  - NAV_LINE
  - NAV_ARC
  - NAV_WPT
  - NAV_LABEL
- END_SEG

MODULE NAME: NAV_TEXT
FILE NAME: NAV_UTL.MAR
CALLING SEQUENCE: CALL NAV_TEXT(LINE_ID,COUNT,TEXT)

PURPOSE:
To store textual data for NAV format information lines.

DESCRIPTION:
The navigation format has four lines of textual information appearing in the upper right corner. Any of these lines can be updated with a call to NAV_TEXT. The first calling parameter for NAV_TEXT is the line identification number, from one to four. Next is the text character count followed by the actual buffer containing the ASCII text.

GLOBAL REFERENCES:

VARIABLES
NAVPTR*

ARRAYS
NAVDAT*

MODULE NAME: NAV_SYMBOL
FILE NAME: NAV_UTL.MAR
CALLING SEQUENCE: CALL NAV_SYMBOL(CLASS,X,Y,CNT,TEXT)
CALL NAV_SYMBOL(CLASS,X,Y,BRG,LEN,CNT,TEXT)

PURPOSE:
To place NAV format reference symbols on the display.

DESCRIPTION:
Navigation format symbols are placed on the display screen by calls to NAV_SYMBOL. The desired symbol is requested within the parameter list by using predefined names from the file CONSTANT.INC. The position of the symbol is passed as the X (east) and Y (north) offsets from the map background screen center (in feet). Each symbol optionally may have label text written alongside on the display screen. The character count and actual ASCII text are passed as the last two parameters. Note that commas must remain in the calling sequence when optional parameters are excluded.

The two different calling sequences pertain to the two categories of symbols available; non-rotatable and rotatable. The rotatable group uses the second calling sequence, which has bearing and length. This type of symbol maintains its orientation within the map background. Non-rotatable symbols do not rotate with the map background. Their orientation stays fixed relative to the display screen. The various types of symbols are shown below.

ROTATABLE GROUP

MNEMONIC	DESCRIPTION
FLDRWY	airfield with runway line
RWY	runway
SRP	selected reference point

NON-ROTATABLE GROUP

MNEMONIC	DESCRIPTION
MOUNT	mountain
OBSTR	obstruction
BEACON	radio beacon
VORTAC	VOR/TAC antennae
VOR	VOR antennae
DME	DME antennae
AIRFLD	airfield
GRP	ground reference point
MARKER	outer/middle marker
WPT	waypoint

Note that within the rotatable group only the runway symbol uses the length parameter.

GLOBAL REFERENCES:

VARIABLES
  NAVPTR*

ARRAYS
  NAVDAT*

MODULE NAME: BEG_SEG
FILE NAME: NAV_UTL.MAR
CALLING SEQUENCE: CALL BEG_SEG(COLOR_TYPE,X,Y)

PURPOSE:
To initiate a NAV background drawing sequence.

DESCRIPTION:
Navigation format drawing sequences are started with calls to BEG_SEG. Two items are associated with an entire drawing sequence; line type and initial position. These are set from the parameters passed to BEG_SEG. The color and line type are passed as a code value in the first parameter. The list below shows the predefined names, found in CONSTANT.INC, which are used to select the desired type. Note that the passed parameter is actually the sum of the color code and line type.

COLOR CODES

WHITE GREEN BLUE AMBER RED CYAN MAGENTA YELLOW

LINE TYPES

SOLID DASH DOT DOTDSH

Starting position of the drawing sequence is provided in the last two parameters. These represent the X (east) and Y (north) offsets from map background screen center (in feet).

After BEG_SEG has been called, no other map background utilities except NAV_LINE, NAV_ARC, NAV_WPT, and NAV_LABEL may be called until an END_SEG call is made.

GLOBAL REFERENCES:

VARIABLES
NAVPTR*

ARRAYS
NAVDAT*

MODULE NAME: NAV_LINE
FILE NAME: NAV_UTL.MAR
CALLING SEQUENCE: CALL NAV_LINE(X,Y,COUNT,TEXT)

PURPOSE:

To draw a line on the NAV display.

DESCRIPTION:

Lines are drawn as part of the map background by calls to NAV_LINE. This procedure may only be called as part of a map background drawing sequence which is initiated with a call to BEG_SEG.

The line drawn on the display starts at the last position made in the drawing sequence and terminates at the position specified in the parameter list. The first two values in the parameter list specify the X (east) and Y (north) offsets from the map background screen center (in feet). A text label may optionally be specified in the calling sequence. The number of characters is specified first, followed by the buffer containing the actual ASCII text. The text is positioned at the endpoint of the line segment. Note that commas must be supplied when option parameters are omitted.

GLOBAL REFERENCES:

VARIABLES
NAVPTR*

ARRAYS
NAVDAT*

MODULE NAME: NAV_ARC
FILE NAME: NAV_UTL.MAR
CALLING SEQUENCE: CALL NAV_ARC(ANG,RAD,BRNG,CNT,TEXT)

PURPOSE:
To generate arc segments for the NAV format display.

DESCRIPTION:
Arc segments are drawn as part of the map background by calls to NAV_ARC. This procedure may only be called as part of a map background drawing sequence which is initiated with a call to BEG_SEG.

The arc segment drawn on the display starts at the last position made in the drawing sequence. Three parameters are required to draw an arc segment. The first is the number of degrees of arc to draw. Next the radius of the arc is supplied, in feet. Note that the turn direction is established by the radius parameter. A negative value means a left turn and a positive value is used for a right turn. The third parameter is the compass bearing of the inbound tangent to the arc.

A text label may optionally be specified in the calling sequence. The number of characters is specified first (CNT), followed by the buffer containing the actual ASCII text (TEXT). The text is placed at the endpoint of the arc segment. Note that commas must be supplied when option parameters are omitted.

GLOBAL REFERENCES:

VARIABLES
NAVPTR*

ARRAYS
NAVDAT*

MODULE NAME: NAV_WPT
FILE NAME: NAV_UTL.MAR
CALLING SEQUENCE: CALL NAV_WPT(CNT,TEXT)
CALL NAV_WPT(X,Y,CNT,TEXT)

PURPOSE:

To store path waypoint symbols in the map background.

DESCRIPTION:

Path waypoint symbols are drawn as part of the map background by calls to NAV_WPT. This procedure may only be called as part of a map background drawing sequence which is initiated with a call to BEG_SEG.

A path waypoint may be placed at the last position established in the drawing sequence using the first calling format shown above. The path waypoint may be positioned independently by using the second format which has the X (east) and Y (north) offsets from the map background screen center (in feet). Note that the established drawing sequence screen position is not updated when a positioned path waypoint is entered into the map background.

A text label may optionally be specified in the calling sequence. The number of characters is specified first, followed by the buffer containing the actual ASCII text. The text is placed at the lower right side of the waypoint symbol. Note that commas must be supplied when optional parameters are omitted.

GLOBAL REFERENCES:

VARIABLES
NAVPTR*

ARRAYS
NAVDAT*

MODULE NAME: NAV_LABEL
FILE NAME: NAV_UTL.MAR
CALLING SEQUENCE: CALL NAV_LABEL(CNT,TEXT)

PURPOSE:
To store text labels into the map background.

DESCRIPTION:
Text labels may be placed at the current position in the drawing sequence by calling NAV_LABEL. This procedure may only be called as part of a map background drawing sequence which is initiated with a call to BEG_SEG. The character count and text buffer containing ASCII codes are supplied in the calling parameter list.

GLOBAL REFERENCES:

VARIABLES
NAVPTR*

ARRAYS
NAVDAT*

MODULE NAME: END_SEG
FILE NAME: NAV_UTL.MAR
CALLING SEQUENCE: CALL END_SEG

PURPOSE:
To terminate a map background drawing sequence.

DESCRIPTION:
Each time a BEG_SEG call is made, followed by other map background drawing sequence calls, a matching call to END_SEG must be made to complete the data packets stored in the background buffer. Any number of BEG_SEG/END_SEG pairs may be stored into the map background buffer.

GLOBAL REFERENCES:

VARIABLES
NAVPTR

ARRAYS
NAVDAT*

Section 7.3 NAV BACKGROUND PROCEDURES

Twenty five procedures are dedicated to the creation of display buffer data for the NAV format background buffer. The following chart lists the procedure names along with their source code language and relative size. Those modules that serve as utility subroutines to another procedure are shown with their caller. The size provided is the percentage of total NAV software memory usage.

Refer to Appendix A to identify which locations in the output buffer (OUTDAT) are used by these modules. Note that locations used for this format are tagged in the appendix by the mnemonic "NAV". The following pages include module descriptions for each of the twenty five procedures.

MODULE	SOURCE	SIZE
BOUNDS	FORTRAN	5%
AREAS		
NEARPT		
MAP_AIRWAY	FORTRAN	4%
GET_XY		
NAME_SIZE		
NAVSLW	FORTRAN	6%
NAVUPD		
BUSFMT		
OPTION	FORTRAN	27%
AIRPRT		
ARPSMB		
RUNWAY		
STRIPS		
NAVAID		
NAVSMB		
RADIAL		
PATHS	FORTRAN	18%
PLAN		
LEG		
DMA		
TURN		
WPTEXT		
TEXT	FORTRAN	8%
STORE		

MODULE NAME: BOUNDS
FILE NAME: BOUNDS.FOR
PROCESS: DSPSLW
CALLED BY: NAVSLW
CALLING SEQUENCE: CALL BOUNDS(MAP_ID)

PURPOSE:
To draw boundaries for the coastal air defense identification zones (CADIZ), air defense identification zones (ADIZ), and the restricted areas.

DESCRIPTION:
This module is the main driver for the navigation display's boundaries. There are three types of zones that can be shown on the NAV display when the boundary bezel button is selected. Included is NASA restricted areas, air defense, and coastal defense zones. Data for the boundary lines is stored in the system data base (AADCOM). The procedure "AREAS" is called with the address of a zone type and the color/line type for the display. Note the address is advanced by two to move past the start word for each zone type.

GLOBAL REFERENCES:

VARIABLES
ADZPTR CDZPTR RESPTR

RECORD ARRAYS
NVMODE

FUNCTIONS AND SUBROUTINES
AREAS

MODULE NAME: AREAS
FILE NAME: BOUNDS.FOR
PROCESS: DSPSLW
CALLED BY: BOUNDS
CALLING SEQUENCE: CALL AREAS(ADDRESS, COLOR)

PURPOSE:
To create NAV background boundary lines.

DESCRIPTION:
This procedure is passed the address of a boundary area and the desired line color/type code. The boundary area consists of a six character name followed by a series of latitude and longitude pairs. A zero word terminator marks the end of the area. The subroutine GRID converts the latitude/longitude values to north/east coordinates relative to the current map background screen center. Clipping at the screen boundaries is performed on each line formed by two sets of lat/lon values. Because of clipping, one boundary made from connected line segments may be broken into many disjoint sections. The map background utility procedures, described in section 7.2, are called to create display data for the various boundary lines processed. The boundary label is placed at the line end-point nearest the screen center. This position is found by the procedure NEARPT.

GLOBAL REFERENCES:

VARIABLES
BOTTOM LEFT NVLAT NVLON RIGHT TOP

FUNCTIONS AND SUBROUTINES
BEG_SEG CLIP END_SEG GET_REAL GET_WORD GRID NAV_LINE NEARPT

MODULE NAME: NEARPT
FILE NAME: BOUNDS.FOR
PROCESS: DSPSLW
CALLED BY: AREAS
CALLING SEQUENCE: CALL NEARPT(PTR,LABEL,TOTAL)

PURPOSE:

To select a vector end-point for boundary labeling.

DESCRIPTION:

This procedure steps through the latitude and longitude pairs stored for map boundary lines. The first item in the calling parameter list is the address of the boundary in the system database. Two items are returned to the caller. The total number of latitude/longitude pairs is returned along with the index of the point selected to receive the boundary label. The selected end-point is the one closest to the map background screen center. An approximation is used to find the distance from screen center. The absolute value of the latitude and longitude difference between the end-point and screen center are summed. This process usually selects a good place for the boundary label with little processing expense.

GLOBAL REFERENCES:

VARIABLES
NVLAT NVLON

FUNCTIONS AND SUBROUTINES
GET_REAL

MODULE NAME: MAP_AIRWAY
FILE NAME: MAP_AIRWAY.FOR
PROCESS: DSPSLW
CALLED BY: OPTION
CALLING SEQUENCE: CALL MAP_AIRWAY

PURPOSE:
To draw selected airways on the map display.

DESCRIPTION:
This procedure generates the map background display data for airways. When an airway is called up on the "NAV data" page of the pilot's control display unit (CDU), the database address is sent to the display MicroVAX. The portions of the airway which fall within the map clip window are sent as waypoint symbols and line segments. Clipping may break the airway into several disjoint segments in the map background buffer.

MAP_AIRWAY steps through the waypoint addresses stored at the airway address in the system database (AADCOM). A zero terminator word marks the end of waypoint addresses in the database. The procedure GET_XY is called to compute the X (east) and Y (north) offsets from the map background screen center (in feet). The utility CLIP is then called to determine the portion of the line segments, formed by adjacent waypoints, that falls within the viewing window. The map background utility procedures, described in section 7.2, are called to generate the line and waypoint symbols in the background buffer. The names of the waypoints are stored with the waypoint symbols for identification.

GLOBAL REFERENCES:

VARIABLES
BOTTOM LEFT RIGHT TOP

ARRAYS
LOKWPT

FUNCTIONS AND SUBROUTINES
BEG_SEG CLIP END_SEG GET_WORD GET_XY NAME_SIZE NAV_LINE
NAV_WPT

MODULE NAME: GET_XY
FILE NAME: MAP_AIRWAY.FOR
PROCESS: DSPSLW
CALLED BY: MAP_AIRWAY
CALLING SEQUENCE: CALL GET_XY(PTR,WPT_PTR,X,Y)

PURPOSE:

To compute airway waypoint positions.

DESCRIPTION:

This procedure is called with an address pointer to airway data stored in the system database. The waypoint address pointed to is fetched and returned as the second item in the calling parameter list. The waypoint address is then used to fetch the latitude and longitude of the waypoint. The procedure GRID is called to convert the latitude and longitude to X (east) and Y (north) offsets from the map background screen center (in feet). The computed values are returned to the caller through the last two calling parameters.

GLOBAL REFERENCES:

VARIABLES
NVLAT NVLON

FUNCTIONS AND SUBROUTINES
GET_LONG GET_REAL GRID

MODULE NAME: NAME_SIZE
FILE NAME: MAP_AIRWAY.FOR
PROCESS: DSPSLW
CALLED BY: MAP_AIRWAY
CALLING SEQUENCE: LENGTH = NAME_SIZE(POINTER)

PURPOSE:
To determine waypoint name length.

DESCRIPTION:
Different types of waypoints in the system database have different name lengths. Navigation aids use three character names, airfields have four, and geographic reference points have five letters in their names. This module is passed a pointer to a waypoint in the system database. It determines the type of waypoint and returns to the caller the length of the name stored at that address.

The waypoint type is determined by the format defined for the system database. This is described in detail in the flight management documentation, in the CDU section. When the fourth byte at the waypoint address is negative, the waypoint is a navigation aid. When the fifth byte is a blank character, its an airfield. Otherwise the waypoint is a geographic reference point.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
GET_BYTE

MODULE NAME: NAVSLW
FILE NAME: NAVSLW.FOR
PROCESS: DSPSLW
CALLED BY: DSPSLW
CALLING SEQUENCE: CALL NAVSLW

PURPOSE:
To control the NAV background updating for the various NAV formats running in the display system.

DESCRIPTION:
This is the main procedure for the generation of NAV display background data. The Sperry microprocessor system may have from one to four NAV formats loaded and running. Each format has unique background data requirements. When the background of a NAV format needs to be updated the corresponding update flag, UPD(1) through UPD(4), is set by one of the routines FMTBZL or NAVEXC. Section 7.1 describes the format of the map background buffer.

The same 400 word output buffer is sent to the display microprocessors to update any of the NAV formats running in the system. However the amount of data generated for a single NAV format may be up to 800 words. A two step map background update will occur when more than 400 words of data are generated. It is the job of NAVSLW to control the use of the map background buffer, restricting its use to alleviate conflicts. The buffer is not available to other NAV formats while an update is in progress.

The task DSPSLW, which contains NAVSLW, runs at the lowest priority in the display MicroVAX (see section 2.1). This means that NAVSLW does not run in a fixed time frame, but runs whenever spare time is available. Since all I/O is performed in the main 50 millisecond frame, NAVSLW must perform synchronization steps to assure data integrity. Without the proper control two problems would occur. The data could be transmitted to the display system before the background buffer is finished, or the changes to the buffer for another NAV format could start before the last one was transmitted. The first potential problem is solved by the "Map Control Word" described in section 7.1. This word is not set until the background buffer is ready to go. Even when the high priority I/O task interrupts the execution of NAVSLW and sends the incomplete buffer, no NAV format will try to use the data until a valid Map Control Word has been set. The problem of changing the data in the buffer before transmission is eliminated by requiring that two increments of the fifty millisecond counter have occurred between background updates. This assures that the transmission has been done at least once. Two increments are used, instead of one, because the I/O takes place during the last 10

milliseconds of the 50 millisecond frame (frame #4). The task DSPSLW may gain control of the system before that point if no other tasks require the system. In this case the frame counter would have incremented but the completed buffer would not have been transmitted. Waiting two frames eliminates that possibility.

NAVSLW starts by checking if the background buffer is available (BWAIT). If it is unavailable it returns, unless the two frame wait has expired. When the wait is complete the Map Control Word is cleared so any new update requests can use the background buffer, unless the second half of a two part buffer needs to be sent. BUSFMT is called to process the second part after the timer expires and the original data block was greater than 400 words.

When the map background buffer is available, the background update request flags for the NAV formats are tested to determine when an update is needed. The update process will start for the first map which has an update flag set. Other NAV formats must wait until the background buffer is available before their update request will be serviced. A background update consists of calls to the subroutines NAVUPD and BUSFMT. The responsibility of NAVUPD is to create background data designated for a particular NAV format. The generated data is saved in an 800 word scratch buffer called NAVDAT. The new data is then set up in the map background output buffer (OUTDAT) by calling BUSFMT. The map control word is managed by BUSFMT also.

While stepping through the update request flags for each map, NAVSLW checks the map orientation status in the map mode structure (NVMODE). If any of the active maps is in the "North-up" mode a flag bit is set in one of the words (DISPST) sent to the FM/FC MicroVAX computer. When a map is in "North-up" mode, the LEGS page of the CDU shows special tags to allow the flight crew to step through the flight plan.

GLOBAL REFERENCES:

VARIABLES
  BKWAIT* CNT50 DISPST* DOUBLE FSAVE

ARRAYS
  MAPID* UPD

RECORD ARRAYS
  NVMODE

FUNCTIONS AND SUBROUTINES
  BUSFMT NAVUPD

MODULE NAME: NAVUPD
FILE NAME: NAVSLW.FOR
PROCESS: DSPSLW
CALLED BY: NAVSLW
CALLING SEQUENCE: CALL NAVUPD(MAP_INDEX)

PURPOSE:

To create data for map background updates.

DESCRIPTION:

This procedure is called when one of the map background update requests has been granted. The index of the selected map is passed as the sole calling parameter. The function of NAVUPD is to oversee the creation of the background data for the selected map.

Several variables are setup for use by the various background display modules. This includes the feet to screen units (one thousandths inches) conversion factor, map clip window boundaries, and map center position.

The first four words of the background buffer contain the start of transmission group (SOT) and the mode control group (see section 7.1). The first word is the SOT header and the second is the SOT word count, which is filled in by BUSFMT after all the data is generated. The third and fourth words are the control group header and mode control bits respectively. The mode control bits are set from the map control structure which reflects the current status of the map bezel panel buttons. The remainder of the data buffer is created by calls to PATHS, BOUNDS, TEXT, and OPTION.

GLOBAL REFERENCES:

VARIABLES
BOTTOM* LAT LATCEN LEFT LON LONCEN NAVPTR* NVLAT* NVLON* NVUNIT* RIGHT TOP*

ARRAYS
NAVDAT

RECORD ARRAYS
NVFMT NVMODE

FUNCTIONS AND SUBROUTINES
BOUNDS OPTION PATHS TEXT

MODULE NAME: BUSFMT
FILE NAME: NAVSLW.FOR
PROCESS: DSPSLW
CALLED BY: NAVSLW
CALLING SEQUENCE: CALL BUSFMT(MAP_INDEX)

PURPOSE:
To store map background data in the background output areas for transmission to the microprocessor displays.

DESCRIPTION:
This procedure moves the completed map background data to the background output buffer and sets the proper codes in the map control word to enable acceptance of the data from the designated map format microprocessor. The Start of Transmission (SOT) count within the buffer is set at this time (see section 7.1). The upper two bits of the SOT count are set to the index (1-4) of the map being updated. This index allows the selected navigation format to choose the correct map center displacement from the list stored in the output buffer (by NAVEXC).

The map control word is part of the I/O memory sent to the microprocessor display system twenty times each second. The bit fields used control the usage of the map background buffer by individual map formats. The upper byte of the control word is a sequence byte. Each time a new background buffer is completed this byte is incremented to signal the navigation formats that fresh data is available. The first four bits of the lower byte of the map control word designate which of the maps should be the recipient of the newly created background buffer. A code value is placed in the four bits which is the sequence number of the micro-processor containing the destination navigation format. The second nibble of the lower byte is used to flag the selected navigation format that the set of data is the second part of a two piece background buffer.

GLOBAL REFERENCES:

VARIABLES
BKWAIT* CNT50 DOUBLE* FSAVE* NAVPTR

ARRAYS
MAPID* NAVDAT OUTDAT*

RECORD ARRAYS
NVFMT

MODULE NAME: OPTION
FILE NAME: OPTION.FOR
PROCESS: DSPSLW
CALLED BY: DSPSLW
CALLING SEQUENCE: CALL OPTION (NAV_ID)

PURPOSE:

Controls the processing of map background data for airports, GRPs, navigation aids, terrain features, origin and destination airports and runways, tuned navaids, and look-up reference points.

DESCRIPTION:

OPTION acts as an executive for processing a large subset of map background information. It calls a number of smaller modules to process specific types of information. OPTION and its set of called procedures are responsible for producing background buffer data for bezel selected features (airports, GRPs, navaids, terrain features), look-up reference points, origin and destination airport information, and reference point information selected via the CDU Fix pages.

Up to four unique NAV formats are allowed in the current display system setup. Since each NAV may select different map scales and bezels, it is necessary for OPTION to know what the selections are for the map it is currently processing. The input parameter NAV_ID provides this information.

One of the responsibilities of this procedure is to produce the data necessary to display a symbol selected on the Nav Data page. The array LOKWPT provides an address into the navigation database for the information requested, and an index indicating the type of symbol to display, respectively. The following table lists the possible values of LOKWPT(2) and the type of symbol it produces.

LOKWPT(2)	symbol
1	navaid
2	airfield
3	GRP
5	waypoint series

The last entry in the table refers to a series of waypoints that can be included in the flight plan. The module MAP_AIRWAY produces the data necessary to display the set of waypoints.

OPTION calls AIRPRT to process origin and destination airports and runways, and calls STRIPS to process the bezel selected information for airports, terrain, navaids, and GRPs. OPTION also determines if the tuned navaids, DME 2 and DME 3 are selected, and if so calls NAVAID which will store the appropriate data in the background buffer.

The last thing done in this procedure is to look for any reference points selected on the CDU Fix pages. A maximum of two fixes can be selected. Each element of the array FIXWRD corresponds to one of the fixes that can be chosen, and contains information about the fix. The least significant bit of each word indicates whether a fix has been selected. RADIAL is called to process the Fix page selections.

Module descriptions for routines mentioned above (AIRPRT, STRIPS, NAVAID, RADIAL, MAP_AIRWAY) can also be found in this section, and should be referenced if more detailed information is desired on what they do.

GLOBAL REFERENCES:

VARIABLES
BOTTOM LEFT NVAD2A NVAD3A NVID NVLAT NVLON RIGHT SCALE*
TDWR_LAT TDWR_LON TOP

ARRAYS
FIXWRD LOKWPT

RECORD ARRAYS
NVMODE

FUNCTIONS AND SUBROUTINES
AIRPRT GET_REAL GET_WORD GRID MAP_AIRWAY NAVAID NAVSMB
NAV_SYMBOL POSBTS RADIAL STRIPS

MODULE NAME: AIRPRT
FILE NAME: OPTION.FOR
PROCESS: DSPSLW
CALLED BY: OPTION
CALLING SEQUENCE: CALL AIRPRT

PURPOSE:
Controls the processing of the origin and destination airports and runways, and also look-up runways.

DESCRIPTION:
Most of the processing done in this routine is based on the values found in the array AIRPTS. This array contains addresses into the navigation database for information concerning origin, provisional destination, and active destination airports and runways. It is arranged as follows.

	airfield	runway
origin	AIRPTS(1,1)	AIRPTS(2,1)
provisional dest	AIRPTS(1,2)	AIRPTS(2,2)
active dest	AIRPTS(1,3)	AIRPTS(2,3)

Runway symbols are shown on map scales of 2, 5, 10, and 20 nautical miles if a non-zero address exists in the proper AIRPTS location for the runway requested. If an address is found, RUNWAY is called to pack the appropriate data into the map background buffer. When a valid runway address does not exist, ARPSMB is called to process an airport symbol instead. Origin and destination airports or runways are always shown. Provisional destination airport information is processed if the addresses for provisional and active destination airfields in AIRPTS differ from each other. On map scales of 40 and 80 nautical miles airport symbols only are displayed.

Runways may be looked-up via the CDU. An address into the navigation database for the runway information requested is stored in the array element LOKWPT(1). As described above, look-up runways are processed by the routine RUNWAY, and are only shown on lower map scales.

GLOBAL REFERENCES:

VARIABLES
SCALE

ARRAYS
AIRPTS LOKWPT

FUNCTIONS AND SUBROUTINES
ARPSMB RUNWAY

MODULE NAME: ARPSMB
FILE NAME: OPTION.FOR
PROCESS: DSPSLW
CALLED BY: AIRPRT
CALLING SEQUENCE: CALL ARPSMB (ARPT_ADDR)

PURPOSE:
Processes the data required to display an airfield.

DESCRIPTION:
ARPSMB accepts as input an address in the navigation database where information about an airfield to be displayed is stored. If the address is valid (non-zero), the latitude and longitude is fetched. The utility routines GRID and POSBTS convert the positions into screen coordinates, and determine if the airfield lies within the view screen. If so, the runway azimuth is retrieved as well. The procedure NAV_SYMBOL will pack the information gotten from the database, along with a four character name it fetches, into the map background buffer. For more information about NAV_SYMBOL refer to section 7.2.

GLOBAL REFERENCES:

VARIABLES
BOTTOM LEFT NVLAT NVLON RIGHT TOP

FUNCTIONS AND SUBROUTINES
GET_REAL GRID NAV_SYMBOL POSBTS

MODULE NAME: NAVAID
FILE NAME: OPTION.DOC
PROCESS: DSPSLW
CALLED BY: OPTION, STRIPS
CALLING SEQUENCE: CALL NAVAID (TUNE_FLG,NAVAID_ADDR)

PURPOSE:

Processes the data necessary to display a navaid.

DESCRIPTION:

This procedure calls the utility routines required to store information in the map background buffer for a requested navaid. Inputs passed in are a flag indicating tune status, and an address into the navigation database for information concerning the navaid. A navaid will be displayed if any one of the following conditions exist: the navaid is tuned, the map scale is 40 nautical miles or less, or it is a high altitude navaid. A bit in one of the database words fetched indicates high altitude status. Calls to GRID and POSBTS are made to determine if the navaid is within the viewing screen. If it is, NAVSMB is called to determine the appropriate symbol to display (there are three types), and the color (tuned and non-tuned navaids are different colors). The latitude and longitude positions, navaid classification, and a three-character name are stored in the background buffer.

NAVAID may be called in one of two ways: to process a single tuned navaid, or to process from the database the entire set of navaids within a particular longitudinal strip.

GLOBAL REFERENCES:

VARIABLES
BOTTOM LEFT NVLAT NVLON RIGHT SCALE TOP

FUNCTIONS AND SUBROUTINES
GET_REAL GET_WORD GRID NAVSMB NAV_SYMBOL POSBTS

MODULE NAME: NAVSMB
FILE NAME: OPTION.FOR
PROCESS: DSPSLW
CALLED BY: OPTION, NAVAID, RADIAL
CALLING SEQUENCE: CALL NAVSMB (TUNE_FLG,TYPE,CLASS)

PURPOSE:

Determines the type of navaid symbol to display.

DESCRIPTION:

Three types of navigation aids (Navaids) may be shown on the map. They are: VORs, VORTACs, and non-directional beacons. As input to this routine, a data word fetched from the area of the navigation database containing information about the navaid being processed is sent to NAVSMB in the form of the parameter TYPE. Certain bits of this word are checked to determine which one of the three types is to be displayed. Also, navaids may be tuned or non-tuned, which on the screen will result in different color symbols being shown. Tuned Navaids are depicted in green, non-tuned are white. The input parameter TUNE_FLG indicates the appropriate state. CLASS is an output parameter which contains the result of the processing done by NAVSMB, which is a single word indicating navaid type and tuned status.

MODULE NAME: RADIAL
FILE NAME: OPTION.FOR
PROCESS: DSPSLW
CALLED BY: OPTION
CALLING SEQUENCE: CALL RADIAL (INDEX)

PURPOSE:

Processes the data required to draw radials through, or a circle around, a selected reference point.

DESCRIPTION:

This procedure processes information entered on the CDU Fix pages. By means of the Fix page, a reference point may be displayed on the map with radials drawn through it at bearings selected on the CDU page. Also, a circle can be drawn around the point at a fixed radius in nautical miles. The CDU Fix page allows for up to two fixes to be selected. The input parameter to RADIAL is an index indicating which fix is being processed.

Certain global variables contain information used in drawing the selected reference point (SRP) symbology. Bits in the variable FIXWRD indicate whether the fix is active, how many bearings have been entered, and what type of SRP has been chosen. FIXADD contains the address into the navigation database of the SRP information. If a circle around the SRP has been requested on the Fix page, FIXCIR contains the radius entered in nautical miles.

The basic SRP symbol consists of a fixed size circle drawn around a selected reference point, with two radials extending from the edge of the circle in opposite directions at a bearing entered on the Fix page. Up to four bearings can be selected for a particular fix. A SRP symbol will be displayed for each bearing requested, which means that up to four SRP symbols may overlay the same reference point.

Three types of reference points may be selected on the Fix page. They are navaids, airports, and GRPs. Using FIXADD, the reference point's position is fetched from the database. The position must be within the screen viewing area before the SRP symbology can be processed. The utility procedures GRID and POSBTS are used to determine this, and also to convert the lat/lon position to screen coordinates. NAV_SYMBOL is called to pack the SRP data into the background buffer.

The Fix page also allows for a circle to be drawn, at a selected radius, around a reference point. FIXCIR contains the entered radius value. The map background utility routines BEG_SEG, NAV_ARC, and ENG_SEG produce the buffer data necessary to display the circle. For more information about these modules, and NAV_SYMBOL, reference section 7.2.

GLOBAL REFERENCES:

VARIABLES
  BOTTOM LEFT MAGVAR NVLAT NVLON RIGHT TOP

ARRAYS
  FIXADD FIXCIR FIXRAD FIXWRD

FUNCTIONS AND SUBROUTINES
  ANGL BEG_SEG END_SEG GET_REAL GET_WORD GRID NAVSMB NAV_ARC
  NAV_SYMBOL POSBTS

MODULE NAME: RUNWAY
FILE NAME: OPTION.FOR
PROCESS: DSPSLW
CALLED BY: AIRPRT
CALLING SEQUENCE: CALL RUNWAY (TYPE,RWY_ADDR,ARPT_ADDR)

PURPOSE:

Processes the data for the origin, destination, and look-up runways, and also the runway centerline.

DESCRIPTION:

RUNWAY is responsible for retrieving information about a requested runway from the navigation database, and calling the utility routines necessary to store the data in the map background buffer. Inputs to this routine are: a flag indicating runway type, a database address for the runway, and a database address for the runway's airfield respectively.

Using the runway address, the latitude and longitude position, runway length, and azimuth, are fetched. The utility routines GRID and POSBTS are called to convert the threshold position to screen coordinates, and to determine if it lies within the view screen. If it does, NAV_SYMBOL is called to pack the position, azimuth, length, and runway tag into the background buffer (see section 7.2 for more information about NAV_SYMBOL. The input parameter TYPE is used in figuring the proper tag to display. A two-character tag corresponding to the runway number is displayed for a look-up runway, while origin and destination runways are tagged with a four-character name identifying the runway's airfield. The input parameter containing the airfield's address is used to retrieve the airfield name from the database.

Origin and destination runways will have a centerline drawn in addition to the symbol. The centerline is a fixed 10 nautical mile length, and is drawn as a green dotted line. It is clipped at the screen edge if necessary. The utility routines BEG_SEG, NAV_LINE, and END_SEG put the centerline data into the background buffer. Section 7.2 describes these modules in more detail.

GLOBAL REFERENCES:

VARIABLES
BOTTOM LEFT NVLAT NVLON RIGHT TOP

FUNCTIONS AND SUBROUTINES
BEG_SEG CLIP END_SEG GET_REAL GRID NAV_LINE NAV_SYMBOL POSBTS SCOSD

MODULE NAME: STRIPS
FILE NAME: OPTION.FOR
PROCESS: DSPSLW
CALLED BY: OPTION
CALLING SEQUENCE: CALL STRIPS

PURPOSE:

Searches for and processes the longitudinal strip information in the navigation database corresponding to the region currently mapped by the NAV format.

DESCRIPTION:

Most of the information contained in the database is arranged in groups of longitudinal strips. A strip is bounded by a pair of longitudinal values which differ by two degrees (for example, 76 through 78 degrees longitude). Within it are sets of data for the airports, GRPs, navaids, terrain features, among other things, found in that region. This routine determines which strips of data lie inside the view screen, and processes any data requested within that strip.

STRIPS sequentially examines all the longitude pairs in the database. The utility routine GRID determines if a particular pair is within the screen area. If it is, then the airfield, navaid, terrain, and GRP data for that strip is processed if the corresponding bezel has been pressed on. Each strip has address pointers for the four data types, which are used in referencing the appropriate database location.

The rest of this module description assumes that a particular strip has been determined to be inside the viewing area, and the bezel for the feature being described is on. The procedures GRID and POSBTS determine if the latitude and longitude position of the individual members of each data feature are within screen limits. The map background utility routine NAV_SYMBOL packs up the information necessary to display a symbol into the background buffer. For more information on NAV_SYMBOL refer to section 7.2.

The first data feature processed in STRIPS is airports. To display an airfield the latitude and longitude position is required, and also a four-character airfield name.

Navaids are processed next. The NAV routines NAVAID and NAVSMB are called to do that. Their module descriptions, also found in this section, explain in detail the types of navaids that exist, the conditions under which they are shown, and the information required by the NAV format to display them.

There are two types of terrain features: obstructions and mountains. The mountain bezel activates both. Mountain and obstruction information is grouped together in the

database. The most significant bit in the first data word for an individual symbol is used to differentiate between the two. If the bit is set, an obstruction is processed, otherwise a mountain. Obstructions are only shown on map scales of 20 nautical miles or less. A lat/lon position converted to screen units and a tag indicating the obstruction's height in feet, are packed into the buffer. For mountains, a position and a tag representing the mountain's height in hundreds of feet are packed.

The last data feature processed in STRIPS is GRPs. GRPs are not shown on the 80 nautical mile map scale, and only high altitude GRPs are displayed at 20 and 40 nautical miles. A specific bit set in the first word from the GRP's data area indicates that it is a high altitude GRP. A screen position, and a five-character name are packed into the map background buffer.

GLOBAL REFERENCES:

VARIABLES
  BOTTOM IBPTR LEFT NVID NVLAT NVLON RIGHT SCALE TOP

RECORD ARRAYS
  NVMODE

FUNCTIONS AND SUBROUTINES
  GET_CHAR GET_LONG GET_REAL GET_WORD GRID LIB$SKPC NAVAID
  NAV_SYMBOL POSBTS

MODULE NAME: PATHS
FILE NAME: PATHS.FOR
PROCESS: DSPSLW
CALLED BY: NAVUPD
CALLING SEQUENCE: CALL PATHS(MAP_INDEX)

PURPOSE:
To initiate the generation of flight plan displays.

DESCRIPTION:
This module creates flight plan data for the navigation display format. This includes the provisional and active flight plans, when generated by the FM/FC MicroVAX computer from flight crew entries on the CDU. The module PLAN is called to draw a series of connected flight plan elements (straight line and arc segments). Only one call to PLAN is needed to create the map background data for the active flight plan since the active path is always one connected piece. The provisional path may be a set of disjoint path segments, separated by "Route Discontinuities" on the pilot's CDU. The provisional flight plan is parsed by PATHS to identify each separate flight plan section. A separate call to PLAN is made for each section found.

GLOBAL REFERENCES:

VARIABLES
ACTCNT MODCNT PMODE TXTMOD*

RECORD ARRAYS
ACT_WPTS MOD_WPTS NVMODE

FUNCTIONS AND SUBROUTINES
PLAN

MODULE NAME: PLAN
FILE NAME: PATHS.FOR
PROCESS: DSPSLW
CALLED BY: PATHS
CALLING SEQUENCE: CALL PLAN(WAYPOINTS,COUNT,TYPE)

PURPOSE:
To generate map background data for a flight plan segment.

DESCRIPTION:
This procedure is called to draw flight plan paths. The buffer containing the flight plan waypoints is passed as the first calling parameter. The number of waypoints contained in the buffer is the second parameter and the type of flight plan, active or provisional, is third.

PLAN makes calls to the utility GRID to convert latitude and longitude coordinates to X (east) and Y (north) offsets from map background screen center (in feet). Then one of three subroutines is called to create path data, dependent on the type of flight plan leg being processed. If the current waypoint is a DMA turn entry point, LEG is called to draw the straight line approach. If the current waypoint is an exit to a DMA turn, the module DMA is called to generate the DMA arc. Otherwise a straight line segment will be drawn unless the "pass-by" distance at the waypoint as shown at the current map scale is greater than .1 inches. In this situation the procedure TURN is called to create the "pass-by" turn segment.

When a flight plan is made up of only one waypoint, PLAN tests for positioning within the clipping window and if valid, calls NAV_SYMBOL to store the waypoint symbol.

GLOBAL REFERENCES:

VARIABLES
ACTCNT BOTTOM I* LEFT MODCNT NEWSEG NVLAT NVLON NVUNIT
RIGHT TOP X2 Y2

ARRAYS
AIRPTS TDAT X* Y*

FUNCTIONS AND SUBROUTINES
DMA END_SEG GRID LEG NAV_SYMBOL POSBTS TURN WPTXT

MODULE NAME: LEG
FILE NAME: PATHS.FOR
PROCESS: DSPSLW
CALLED BY: PLAN, TURN
CALLING SEQUENCE: CALL LEG(WAYPOINTS,COLOR,TEXT_FLAG)

PURPOSE:
To create a straight line segment of a flight plan.

DESCRIPTION:
This subroutine calls map background utility procedures to form a straight path segment with waypoint symbols. First the procedure CLIP is called to trim the path leg to the display window boundaries. If the end of the line is clipped the current background drawing sequence is terminated by calling END_SEG. The global flag NEWSEG is set to indicate that a BEG_SEG call must be made before any more path segments are placed in the background buffer.

When the leg endpoint is within the clip window, the module WPTXT is called to format the waypoint label for the leg endpoint. The utility procedure NAV_WPT is called to store the waypoint symbol and its label in the background buffer. When LEG is called with the NEWSEG flag on, the labeled waypoint symbol for the beginning point of the line is stored if it falls within the clipping region.

GLOBAL REFERENCES:

VARIABLES
BOTTOM I LEFT NEWSEG* RIGHT TOP

ARRAYS
TDAT X Y

FUNCTIONS AND SUBROUTINES
BEG_SEG CLIP END_SEG NAV_LINE NAV_WPT WPTXT

MODULE NAME: DMA
FILE NAME: PATHS.FOR
PROCESS: DSPSLW
CALLED BY: PLAN
CALLING SEQUENCE: CALL DMA(WAYPOINTS,COLOR)

PURPOSE:

To create a DMA turn segment of a flight plan.

DESCRIPTION:

This subroutine calls map background utility procedures to form a DMA turn path segment with waypoint symbols. First the procedure CLIP is called to determine if the entire turn segment is outside the clipping region. If it is, the current background buffer drawing sequence is terminated by calling END_SEG and the global flag NEWSEG is set to indicate that a BEG_SEG call must be made before any more path segments are placed in the background buffer. If any of the arc falls within the clipping region the entire turn segment is added to the current drawing sequence. This is done because segment clipping is only performed on straight lines.

The background utility NAV_ARC is called to store the arc segment in the background buffer. The turn angle, arc radius, and inbound bearing parameters passed to NAV_ARC are all obtained from the waypoint buffer. Note that the bearing found in the waypoint buffer for DMA turn segments is actually perpendicular to the inbound tangent point. Ninety degrees must be added or subtracted depending on the turn direction (left or right from inbound course).

When the arc endpoint is within the clip window, the module WPTXT is called to format the waypoint label for the arc endpoint. The utility procedure NAV_WPT is called to store the waypoint symbol and its label in the background buffer. When DMA is called with the NEWSEG flag on, the labeled waypoint symbol for the beginning point of the arc is stored if it falls within the clipping region.

GLOBAL REFERENCES:

VARIABLES
BOTTOM I LEFT NEWSEG* RIGHT TOP

ARRAYS
TDAT X Y

FUNCTIONS AND SUBROUTINES
ANGL BEG_SEG CLIP END_SEG NAV_ARC NAV_WPT WPTXT

MODULE NAME: TURN
FILE NAME: PATHS.FOR
PROCESS: DSPSLW
CALLED BY: PLAN
CALLING SEQUENCE: CALL TURN(WAYPOINTS,COLOR)

PURPOSE:
To create a turn segment for a flight plan.

DESCRIPTION:
This subroutine calls map background utility procedures to form an arc path segment with the "pass-by" waypoint symbol. TURN calls the utility procedure PROJECT to find the inbound and outbound tangent points which define the start and endpoints of the arc segment. Waypoint symbols are not placed at these points. One waypoint symbol is positioned off the arc at the intersection of the imaginary inbound and outbound tangent lines. A straight leg is drawn to the inbound tangent point. The display data for this leg is generated by calling the procedure LEG.
The procedure CLIP is called to determine if the entire turn segment is outside the clipping region. If it is, the current background buffer drawing sequence is terminated by calling END_SEG and the global flag NEWSEG is set to indicate that a BEG_SEG call must be made before any more path segments are placed in the background buffer. If any of the arc falls within the clipping region the entire turn segment is added to the current drawing sequence. This is done because segment clipping is only performed on straight lines.

The background utility NAV_ARC is called to store the arc segment in the background buffer. The turn angle, arc radius, and inbound bearing parameters passed to NAV_ARC are all obtained from the waypoint buffer.

When the arc segment is within the clip window, the module WPTXT is called to format the label text for the "pass-by" waypoint. The utility procedure NAV_WPT is called to store the waypoint symbol, its position, and its label in the background buffer.

GLOBAL REFERENCES:

VARIABLES
BOTTOM I LEFT NEWSEG* NVLAT NVLON RIGHT TOP X2 Y2

ARRAYS
TDAT X Y

FUNCTIONS AND SUBROUTINES
BEG_SEG CLIP END_SEG GRID LEG MTH$ATAND2 NAV_ARC NAV_WPT
PROJECT WPTXT

MODULE NAME: WPTXT
FILE NAME: PATHS.FOR
PROCESS: DSPSLW
CALLED BY: PLAN, LEG, DMA, TURN
CALLING SEQUENCE: CALL WPTXT(INDEX,WAYPOINTS,COUNT)

PURPOSE:
To format waypoint labels for the flight plan.

DESCRIPTION:
This procedure is called with a waypoint buffer and the corresponding index designating which waypoint within the buffer is being processed. Label text is generated in the global buffer TDAT and the number of characters created for the label is returned in the last calling parameter. The extent of the label depends on the global index TXTMOD. The following chart describes what is included in the waypoint label for the possible values of TXTMOD.

TXTMOD	LABEL GENERATION
0	Include waypoint name only.
1	Include waypoint name, assigned altitude and ground speed.
2	If current waypoint is the 'TO' waypoint of the active flight plan, process as TXTMOD = 0. Otherwise no label is generated.
3	If current waypoint is the 'TO' waypoint of the active flight plan, process as TXTMOD = 1. Otherwise no label is generated.

When altitude and speed values are included in the label, waypoint buffer flags are tested to determine when actual values have been assigned. The label text will have dashes instead of digits when no value is assigned.

GLOBAL REFERENCES:

VARIABLES
TDAT* TOWPT TXTMOD

FUNCTIONS AND SUBROUTINES
OTS$CVT_L_TI

MODULE NAME: TEXT
FILE NAME: TEXT.FOR
PROCESS: DSPSLW
CALLED BY: NAVUPD
CALLING SEQUENCE: CALL TEXT(MAP_INDEX)

PURPOSE:
To update map background text lines.

DESCRIPTION:
The Navigation display format allows the display of four lines of text data, 19 characters per line, in the upper right hand corner of the display screen. The text data is supplied through the map background buffer (see section 7.1). The sole calling parameter to TEXT is the index of the navigation format currently being updated.

The module TEXT builds a buffer of up to 19 characters, and uses the map background utility procedure NAV_TEXT to place the text in the background buffer. Shown below is a list of the items included on each display line.

LINE #1

ITEM	CONDITIONS
distance to go	when active flight plan exists
'TO' wpt name	when active flight plan exists
time of day	always

LINE #2

ITEM	CONDITIONS
airport indicator	'ARPT' bezel selected
navaid indicator	'NVD' bezel selected
time box indicator	'BOX' bezel selected
range arc indicator	'ARC' bezel selected

LINE #3

ITEM	CONDITIONS
terrain indicator	'MTN/OBSTR' bezel selected
GRP indicator	'GRP' bezel selected
boundary indicator	'BNDS' bezel selected

LINE #4

ITEM	CONDITIONS
2D/3D/4D	when auto guidance engaged
TRK indicator	auto-track select mode engaged
ALT indicator	auto-altitude hold mode engaged
IAS indicator	auto throttle engaged
FPA indicator	auto-flight path angle engaged
navigation mode	always

GLOBAL REFERENCES:

VARIABLES
  ALTSEL AWAS_BITS DATA_TM DAY DTOGO FE3 FPASEL GUID2D
  HORPTH HRAD_BHR IASSEL NAV TAS TASVLD TIME TIMPTH
  TKSEL TOWPT VERPTH

ARRAYS
  AWAS_DATA

RECORD ARRAYS
  ACT_WPTS NVMODE

FUNCTIONS AND SUBROUTINES
  FMTTIM NAV_TEXT OTS$CVT_L_TI OTS$FLOAT STORE

Section 7.4 NAV REAL-TIME PROCEDURES

This section is devoted to the modules that create navigation format real-time data. All the navigation format symbology, except the map background, is updated twenty times per second. The procedures described in this section format and store data into the microprocessor output buffer at this update rate.

The following chart lists the procedure names along with their source code language and relative size. Those modules that serve as utility subroutines to another procedure are shown with their caller. The size provided is the percentage of total NAV software memory usage.

Refer to Appendix A to identify which locations in the output buffer (OUTDAT) are used by these modules. Note that locations used for this format are tagged in the appendix by the mnemonic "NAV". The following pages include module descriptions for each of the fifteen procedures.

MODULE	SOURCE	SIZE
FMTBZL	FORTRAN	2%
NAVEXC	FORTRAN	12%
SELTRK		
NAVMLS		
TRENDV		
RNGARC		
TBOX		
ROTATE		
PTHPOS	FORTRAN	12%
TIMPOS		
LINE		
TURN		
PASSBY		
INBRG		
PTHLEG		

MODULE NAME: NAVEXC
FILE NAME: NAVEXC.FOR
PROCESS: DSPFST
CALLED BY: DSPFST
CALLING SEQUENCE: CALL NAVEXC

PURPOSE:
To serve as the main procedure for navigation format real-time processing.

DESCRIPTION:
This procedure initiates all the computations performed for the navigation format real-time symbology. Most of the processing is performed by the modules called from NAVEXC, however several items are handled directly within NAVEXC. The following modules are called to create and store data into the microprocessor output buffer.

FMTBZL	process bezel button inputs
SELTRK	create data for selected track lines
NAVMLS	create data for MLS airplane position
TRENDV	create data for the aircraft trend vector
RNGARC	create data for the altitude range arc
TBOX	create data for time box positioning

The first "in-line" processing performed by NAVEXC is the setting of map background update requests. The FM/FC MicroVAX computer can command updates of all map backgrounds by setting the global flag MAPUPD. This is done when new background data has been generated by CDU software or an update timer expires (5 seconds). The timer is used so the aircraft never can fly too far within the current map background to expose the clipped edge. When MAPUPD is set NAVEXC determines how many of the four possible map backgrounds are actually in use. The update request flags for each one is set on to notify map background procedures in DSPSLW to perform the updates.

A set of variables is filtered to produce smooth movement of navigation format symbology. Included are wind speed, wind direction, aircraft track, and aircraft heading. The rate of change, sine, and cosine of the filtered track are computed also.

Several microprocessor output buffer locations are filled next. Values are scaled and stored as fixed point 16 bit integers for true track, magnetic track, heading, wind speed, wind direction, and ground speed. Two words of the microprocessor output buffer are reserved for validity bits used for the navigation format. The corresponding bits for true track, magnetic track, heading, wind, ground speed, and aircraft position are always set valid. One other bit of these words can be set by NAVEXC when GPS mode is valid. The modules called by NAVEXC set other validity bits in these words.

Each navigation format running in the microprocessor system requires information about where the aircraft has moved since its last map background update. The offset from each map background center position to the current aircraft position is computed and stored in the output buffer for the microprocessors. They are stored as an array of X (east) and Y (north) offsets in one-thousandths inches. Each time a navigation format has its map background updated the map center and scale are saved to allow the background positioning described above.

GLOBAL REFERENCES:

VARIABLES
BCKWRD COSTRK DOUBLE GPSM GSFPS HDGF HDGTRU LAT LON MAGVAR MAPUPD MAPWRD MLSV NVGS* NVHDG* NVLAT NVLON NVMTRK* NVTRK* NVUNIT NVWD* NVWS* PSTTKA SINTRK TK TKSEL TOP TRKF VLD1* VLD2* WD WS

ARRAYS
APE* APN* MAPLN* MAPLT* RANGE* UNITS* UPD*

RECORD ARRAYS
NVFMT

FUNCTIONS AND SUBROUTINES
ANGL FMTBZL GRID NAVMLS POSBTS RNGARC SCOSD SELTRK TBOX TRENDV

MODULE NAME: SELTRK
FILE NAME: NAVEXC.FOR
PROCESS: DSPFST
CALLED BY: NAVEXC
CALLING SEQUENCE: CALL SELTRK(TRACK_RATE)

PURPOSE:
To store data for the selected track lines of the navigation display format.

DESCRIPTION:
This procedure processes items associated with the selected track angle from the pilot's mode control panel. It is only called when a track angle is either selected or preselected. Four symbols are controlled by this procedure.

SELECTED TRACK BUG - Shown on compass whenever selected track is valid.

TRACK LINE - Line connecting airplane and track bug shown when track error greater than one degree or the track dial spin discrete is set.

OFFSET VECTOR - Track line indicating selected track intercept point starting from offset ahead of airplane position. Shown when track line on and bit "0" of TKBITS is manually set.

EXTENDED TREND VECTOR - Same as OFFSET VECTOR but shown as an extension to the NAV trend vector. Bit "1" of TKBITS is used to manually select it.

SELTRK first stores the selected track value and valid bit into microprocessor output memory. Then the computations for either the OFFSET VECTOR or the EXTENDED TREND VECTOR are performed.

The OFFSET VECTOR intercept point along the aircraft's "straight ahead" line is computed from the equation below. The offset vector extends from this point across the screen at the bearing of the selected track. The 16 bit fixed point value sent to the navigation format is converted to nautical miles and scaled by 128 for higher resolution.

```
AHEAD = GS * GS * TAN(.5 * DEL_TRK) / (G * TAN(BANK_ANG))
```

AHEAD: distance ahead (feet)
GS: ground speed (feet/sec)
DEL_TRK: difference between actual and commanded track
BANK_ANG: nominal bank angle fixed at 25 degrees

The amount of time required to complete a desired turn is computed and the corresponding position along the trend vector is used to draw the EXTENDED TREND VECTOR. The equation for the time needed is shown below.

```
TIME = DEL_TRK / (2 * TRK_RATE)

TIME:     time in seconds
DEL_TRK:  difference between actual and commanded track
TRK_RATE: rate of change of actual track
```

Note that the factor of "2" in the equation is used since the track rate variable used by the displays MicroVAX is the number of degrees of change in one half second. The 16 bit fixed point value sent to the navigation format is scaled by 256 for greater resolution.

GLOBAL REFERENCES:

VARIABLES
GSFPS MAGVAR NVDT* NVOFS* NVSLTK* TDSP TKASUM TKBITS TKREL
VLD1* VLD2*

FUNCTIONS AND SUBROUTINES
ANGL MTH$TAN

MODULE NAME: NAVMLS
FILE NAME: NAVEXC.FOR
PROCESS: DSPFST
CALLED BY: NAVEXC
CALLING SEQUENCE: CALL NAVMLS

PURPOSE:

To perform computations for the MLS airplane symbol of the navigation format.

DESCRIPTION:

This procedure is called by NAVEXC when the Microwave Landing System (MLS) has been determined valid. A bit is set in the navigation format discrete word to indicate MLS valid. If MLS mode is engaged another bit is set in the discrete word for the display format. If MLS mode is not engaged the position of the MLS aircraft, as indicated by the MLS beam, is used to compute the offset from the aircraft position derived from the current navigation mode. The subroutine GRID computes the north and east offset in feet. ROTATE is called to convert these coordinates into navigation format "track-up" X and Y screen coordinates in feet. The values are then converted to nautical miles and scaled by a factor of 2048 for resolution.

When the MLS airplane position deviates from the desired flight plan position by more than 50 feet vertically and 500 feet horizontally another discrete word bit is set to command a MLS aircraft symbol color change.

GLOBAL REFERENCES:

VARIABLES
ALTCOR HER LAT LON MLSALT MLSLAT MLSLON MLSM MLSX* MLSY* VLD2* XTK

FUNCTIONS AND SUBROUTINES
GRID ROTATE

MODULE NAME: TRENDV
FILE NAME: NAVEXC.FOR
PROCESS: DSPFST
CALLED BY: NAVEXC
CALLING SEQUENCE: CALL TRENDV

PURPOSE:
To compute trend vector parameters for the navigation format.

DESCRIPTION:
Two items are stored by TRENDV into the microprocessor output buffer for the navigation format trend vector. The aircraft's cross-track acceleration and the ratio of cross-track acceleration with ground speed. They are scaled by factors of 512 and 65536 respectively for resolution when converted to the 16 bit fixed point values sent to the navigation format.

The cross-track acceleration used comes from one of two sources. If in a control-wheel steering mode the current roll command is used to compute the cross-track acceleration as follows.

```
CROSS_TRACK_ACC = G * TAN(COMMANDED_ROLL)
```

The constant "G" above is the gravitational acceleration value. In other modes the cross-track acceleration measured by flight controls is used. This value is filtered before use to provide smooth movement of the trend vector.

GLOBAL REFERENCES:

VARIABLES
ACWS DROLL GSFPS NVACN* VCWS XTACC XTKGS*

FUNCTIONS AND SUBROUTINES
MTH$TAN

MODULE NAME: RNGARC
FILE NAME: NAVEXC.FOR
PROCESS: DSPFST
CALLED BY: NAVEXC
CALLING SEQUENCE: CALL RNGARC

PURPOSE:

To perform computations for the altitude range arc of the navigation format.

DESCRIPTION:

Altitude range logic is performed by a sequence of events starting in the flight management MicroVAX computer. When the flight crew has selected a new altitude via the AGCS mode control panel, the altitude attained flag is made false (ALTATT). This flag is sent to the display MicroVAX computer in the block of data transferred across the inter-processor I/O link. The range to the intercept point of the desired altitude is computed by RNGARC until the difference between commanded and actual altitude is less than 5 feet. At this time the desired altitude is considered attained, but the variable ALTATT can not be set directly since it arrives at the display MicroVAX as input. Instead a bit in a packed discrete word that is sent to the flight management MicroVAX through the DATAC bus is set. This bit informs the flight management MicroVAX to set ALTATT to true.

The distance to the point where the altitude will be attained is computed differently depending on the current guidance mode of the aircraft. When the current guidance mode maintains a commanded flight path angle, all but attitude control-wheel steering, the distance is computed as follows.

DISTANCE = (SELECTED_ALT - ACTUAL_ALT) / TAN(COMMANDED_FPA)

Otherwise the current measured rate of change in altitude is used with the aircraft's ground speed as shown below.

DISTANCE = SPEED * (SELECTED_ALT - ACTUAL_ALT) / ALT_RATE

The computed distance is converted from feet to nautical miles and scaled by a factor of 128 for greater resolution. It is stored as a 16 bit fixed point value in the microprocessor output buffer.

GLOBAL REFERENCES:

VARIABLES
ALTATT ALTCOR ALTRNG* ALTSUM AUTO DISPST* GAMC GSFPS HDCF VCWS

FUNCTIONS AND SUBROUTINES
MTH$TAND

MODULE NAME: TBOX
FILE NAME: NAVEXC.FOR
PROCESS: DSPFST
CALLED BY: NAVEXC
CALLING SEQUENCE: CALL TBOX

PURPOSE:
To intiate position computations for Time Box and Bubbles symbology of the navigation format.

DESCRIPTION:
This procedure stores the position and orientation data for the Time Box and the three Bubbles. Each of the four items is processed the same, with calls to PTHPOS. A reference time is passed to PTHPOS for each of the four items. The data returned is stored into the array of locations in the microprocessor output buffer starting at OUTDAT(603).

The reference time used is the sum of two time offsets. Since the system time (TIME) has a resolution of one second, the fractional part of the current time is maintained by TBOX. Added to this fraction is 0, 30, 60, or 90 seconds for each of the four time box items, which are always separated by 30 seconds in time.

GLOBAL REFERENCES:

VARIABLES
TIME

ARRAYS
OUTDAT*

FUNCTIONS AND SUBROUTINES
PTHPOS

MODULE NAME: ROTATE
FILE NAME: NAVEXC.FOR
PROCESS: DSPFST
CALLED BY: NAVMLS, PTHPOS
CALLING SEQUENCE: CALL ROTATE(NORTH,EAST,X,Y)

PURPOSE:
To perform coordinate system rotations.

DESCRIPTION:
This procedure converts a pair of position coordinates in the North/East frame of reference to values in the navigation format's "track-up" frame of reference. The equations used are shown below.

```
SCREEN_X = EAST * COS(TRACK) - NORTH * SIN(TRACK)
SCREEN_Y = EAST * SIN(TRACK) + NORTH * COS(TRACK)
```

GLOBAL REFERENCES:

VARIABLES
COSTRK SINTRK

MODULE NAME: PTHPOS
FILE NAME: PTHPOS.FOR
PROCESS: DSPFST
CALLED BY: TBOX
CALLING SEQUENCE: CALL PTHPOS(TIME,X,Y,BEARING)

PURPOSE:
To compute values associated with the Time Box symbology of the navigation format.

DESCRIPTION:
This procedure computes the X and Y coordinates of a point on the active flight plan, defined for time guidance, that corresponds to a reference time passed as the first parameter. The reference time is an offset in seconds from the aircraft time stored in the variable TIME. The returned position values are in terms of feet from the current aircraft position. Also returned is an orientation angle which represents the flight plan bearing at the reference position. All three computed values are relative to a "track-up" display orientation. Therefore X is a cross-track distance, Y is an along-track distance, and BEARING is the angular difference from the aircraft's true track value.

PTHPOS positions the Time Box symbology at the aircraft, with the current track, when no flight plan exists (GUID2D is off). If a flight plan exists but has not been defined for time guidance (GUID4D is off), the position of the first waypoint on the flight plan is returned with the bearing of the first leg of the path.

When the active flight plan is defined for time guidance the procedure PTHLEG is called. It returns an index into the active waypoint buffer pointing to the first waypoint which has a planned time of arrival greater than the reference time. This waypoint and the one before it form a time reference path leg containing the desired reference position. The difference between the reference time and the beginning waypoint of the chosen path leg is also returned. This time represents the amount of time elapsed since the Time Box (or Bubble) has passed the first waypoint of the flight plan leg selected by PTHLEG. Note that PTHPOS returns the position of the first flight plan waypoint when the reference time is earlier than the beginning waypoint of the path.

The reference position is found on the reference leg by the procedure TIMPOS. This module returns a north and east offset (in feet) of the time reference position relative to the selected reference waypoint on the leg. One exception to this is when the reference leg is a staight leg (LEGFLG = TRUE). In this case the position values returned from TIMPOS are latitude and longitude coordinates. TIMPOS also returns the orientation bearing of the reference position.

The last thing performed by PTHPOS is the computation of screen offsets, in feet, of the time reference position from the current aircraft location. The utility procedure GRID is called to compute the north and east distances from the airplane to the time reference waypoint. The north and east offsets to the time reference position on the selected path leg are added to the reference waypoint positions. Finally the procedure ROTATE is called to convert the north and east coordinates to values relative to the "track-up" map coordinate system.

GLOBAL REFERENCES:

VARIABLES
  ACTCNT GUID2D GUID4D LAT LEGFLG LON TRKF

RECORD ARRAYS
  ACT_WPTS

FUNCTIONS AND SUBROUTINES
  ANGL GRID INBRG PTHLEG ROTATE TIMPOS

MODULE NAME: TIMPOS
FILE NAME: PTHPOS.FOR
PROCESS: DSPFST
CALLED BY: PTHPOS
CALLING SEQUENCE: CALL TIMPOS(INDEX,DT,DN,DE,BRNG)

PURPOSE:
To compute a time reference position on the active flight plan.

DESCRIPTION:
This procedure computes displacement coordinates from a selected waypoint to a flight plan position corresponding to the time reference in the calling parameter list. The first calling parameter is the index within the waypoint buffer of the end waypoint of the path leg containing the time reference position. The second parameter is the amount of time elapsed between the first waypoint of the path leg and the time reference position. TIMPOS returns the north and east offsets from the reference waypoint and the tangential path bearing at the time reference position.

First the distance between the "from" waypoint and the time reference position is calculated using the time displacement between the locations. To do this the planned acceleration along the path leg is computed as follows.

```
ACC = (SPEED2 - SPEED1) / LEG_TIME
```

Then the distance is derived as shown below.

```
DISTANCE = SPEED1 * DT + .5 * ACC * DT * DT
```

ACC: nominal leg acceleration (ft/sec/sec)
SPEED1: planned ground speed at beginning waypoint (ft/sec)
SPEED2: planned ground speed at ending waypoint (ft/sec)
LEG_TIME: time alloted to fly entire leg of path (sec)
DISTANCE: distance between "from" waypoint and reference position (feet)
DT: time between "from" waypoint and reference position (sec)

Depending on the type of path leg being processed, one of the modules TURN, LINE, or PASSBY will be called to compute the values of the offset coordinates and the bearing associated with the time reference position. The module TURN is called when the reference leg is a DMA turn. In this case the reference waypoint is switched from the end

waypoint to the beginning waypoint to make the computations easier. When entering a standard turn by the destination waypoint the module PASSBY is called. Note that the leg will be considered a straight leg unless the standard turn is significant (arc length greater than 1200 feet). If the reference position falls within the exit area of a significant standard turn past the beginning waypoint, the reference waypoint is switched to the "from" waypoint and PASSBY is called. In all other situations the module LINE is called to process the straight leg segment.

GLOBAL REFERENCES:

VARIABLES
  LEGFLG*

RECORD ARRAYS
  ACT_WPTS

FUNCTIONS AND SUBROUTINES
  LINE PASSBY TURN

MODULE NAME: LINE
FILE NAME: PTHPOS.FOR
PROCESS: DSPFST
CALLED BY: TIMPOS
CALLING SEQUENCE: CALL LINE(INDEX,DST,RLAT,RLON,BRNG)

PURPOSE:
To locate the time reference position on a staight path leg.

DESCRIPTION:
This module uses the position of the reference waypoint, identified by the passed parameter INDEX, and the distance from the 'FROM' waypoint on the reference leg (DST) to compute the offset coordinates and orientation bearing to the time reference position on straight leg segments.

Unless the 'TO' waypoint of the leg is a DMA turn entry point, the leg distance is adjusted to compensate for the standard turn by the 'TO' waypoint. One half the arc length is replaced by the distance from the tangent point to the 'TO' waypoint position. The distance between the time reference position to the end waypoint of the leg is found by subtracting the distance passed to LINE from the total leg distance. This value is passed to the utility procedure PROJECT to compute the time reference position's latitude and longitude using the leg waypoints and displacement from the 'TO' waypoint.

GLOBAL REFERENCES:

VARIABLES
LEGFLG*

RECORD ARRAYS
ACT_WPTS

FUNCTIONS AND SUBROUTINES
INBRG PROJECT

MODULE NAME: TURN
FILE NAME: PTHPOS.FOR
PROCESS: DSPFST
CALLED BY: TIMPOS, PASSBY
CALLING SEQUENCE: CALL TURN(INDEX,DST,DN,DE,BRNG,DME)

PURPOSE:
To locate the time reference position on a turn segment.

DESCRIPTION:
This module uses the position of the reference waypoint, identified by the passed parameter INDEX, and the distance from the "from" waypoint on the reference leg (DST) to compute the offset coordinates and orientation bearing to the time reference position on turn segments. The offsets and bearing are returned through the calling parameter list. The last parameter passed to TURN identifies turn segments which are defined as DMA arc path legs.

The distance from the turn start to the reference position is used with the turn radius to determine the subtented angle (radians) to the reference position as follows.

```
ANGLE = ARC_DISTANCE / TURN_RADIUS
```

The subtended angle is combined with the inbound bearing and turn radius to compute the offsets to the time reference position and the tangential bearing at that point. The inbound bearing is passed to TURN when called by the module PASSBY (DME_FLG=FALSE). Otherwise the function INBRG is used to find the DMA turn inbound bearing.

```
ang = ANGLE/2 - IN_BEARING + (180 +/- 90)
len = 2 * SIN(ANGLE/2) * TURN_RADIUS

NORTH = len * SIN(ang)
EAST = len * COS(ang)
TANGENT_BEARING = IN_BEARING +/- ANGLE
```

("+/-": + for left turn, - for right turn)

GLOBAL REFERENCES:

RECORD ARRAYS
ACT_WPTS

FUNCTIONS AND SUBROUTINES
INBRG MTH$SIND SCOSD

MODULE NAME: PASSBY
FILE NAME: PTHPOS.FOR
PROCESS: DSPFST
CALLED BY: TIMPOS
CALLING SEQUENCE: CALL PASSBY(INDEX,DST,DN,DE,BRNG)

PURPOSE:
To locate the time reference position on a standard turn segment (not DME arc).

DESCRIPTION:
This module uses the position of the reference waypoint, identified by the passed parameter INDEX, and the distance from the "from" waypoint on the reference leg (DST) to compute the offset coordinates and orientation bearing to the time reference position on turn segments. The offsets and bearing are returned through the calling parameter list.

Standard turns do not have waypoints at the beginning and ending of the arc segment. The latitude and longitude of the inbound tangent point of the turn must be found to use in the computation of the time reference position. The utility procedure PROJECT is called to compute the tangent point position from the positions of the 'FROM' and 'TO' waypoints and the distance to tangent value stored in the waypoint buffer. Once the start of the turn position has been established, the procedure TURN is called to compute the time reference position relative to the turn start point. The module GRID is then called to compute the north and east offsets from the turn tangent point to the 'TO' waypoint. These offsets are added to the values returned by TURN to produce the final time reference offset values relative to the reference waypoint.

GLOBAL REFERENCES:

RECORD ARRAYS
ACT_WPTS

FUNCTIONS AND SUBROUTINES
GRID MTH$ATAND2 PROJECT TURN

MODULE NAME: INBRG
FILE NAME: PTHPOS.FOR
PROCESS: DSPFST
CALLED BY: PTHPOS, LINE, TURN
CALLING SEQUENCE: BEARING = INBRG(INDEX)

PURPOSE:

To produce the inbound bearing to a flight plan waypoint.

DESCRIPTION:

This function returns the inbound bearing to a waypoint on the active flight plan. For waypoints that are not DMA turn entry points, the bearing is fetched directly from the waypoint buffer. For DMA entry waypoints the buffer value is the bearing from the turn center to the entry waypoint. Ninety degrees is either added or subtracted to the waypoint buffer bearing as an adjustment. Addition is used for right turns, subtraction for left turns.

GLOBAL REFERENCES:

RECORD ARRAYS
ACT_WPTS

FUNCTIONS AND SUBROUTINES
ANGL

MODULE NAME: PTHLEG
FILE NAME: PTHPOS.FOR
PROCESS: DSPFST
CALLED BY: PTHPOS
CALLING SEQUENCE: CALL PTHLEG(T_REF,INDEX,T_OFF)

PURPOSE:
To identify the flight plan leg containing the time reference position for Time Box positioning.

DESCRIPTION:
This procedure is called to find the flight plan leg containing the time reference position corresponding to either the Time Box or one of the Bubbles. A time offset is passed as the first calling parameter. This value is added to the aircraft GMT to produce the arrival time at the desired reference position. The arrival times at the end waypoint of each leg on the active flight plan are tested until one greater than the reference time is found. The index of the waypoint within the flight plan buffer is returned through the parameter list. Also returned is the time displacement between the reference time and the arrival time stored at the beginning waypoint of the selected path leg.

GLOBAL REFERENCES:

VARIABLES
ACTCNT TIME

RECORD ARRAYS
ACT_WPTS

MODULE NAME: FMTBZL
FILE NAME: FMTBZL.FOR
PROCESS: DSPFST
CALLED BY: NAVEXC
CALLING SEQUENCE: CALL FMTBZL

PURPOSE:
To process navigation format bezel button inputs.

DESCRIPTION:
This module processes the navigation display bezel panel inputs. One 16 bit word is received from the display containing bits corresponding to the 16 buttons on the display unit bezel panel. Up to four navigation formats may be active, so a word from input memory for each is examined. The address of each input word is stored in the structure "NVFMT(I).BZPTR". The following chart shows the usage of the various bezel buttons.

BIT	BUTTON	OPTION
0	R1	MLS select
1	R2	Airports option
2	R3	Navaid option
3	R4	Time Box option
4	R5	Altitude Range Arc option
5	R6	Path waypoint information cycle
6	R7	Zoom out map
7	R8	Zoom in map
8	L1	Weather radar select
9	L2	
10	L3	
11	L4	Terrain features option
12	L5	Ground reference point option
13	L6	
14	L7	Boundaries option
15	L8	Track up / north up toggle

The input words are processed in a loop for each of the navigation formats in use. The individual discrete words are "one-shotted" to make a button press appear to occur for just one 50 millisecond processing frame. If any selections in the word are made, the map background update flag for the corresponding navigation format is set on. The individual bits within the word are processed next. When the MLS select is on, a bit in word (DISPST) sent to the flight management MicroVAX through the DATAC is set. All other bit selections are reflected in the navigation format mode structure (NVMODE). Note that a copy of this structure is kept for each of the four navigation formats.

GLOBAL REFERENCES:

VARIABLES
  DISPST* TDW_TRGT*

ARRAYS
  UPD*

RECORD ARRAYS
  NVFMT NVMODE*

FUNCTIONS AND SUBROUTINES
  GET_WORD

Section 8.0 ENGINE DISPLAY SOFTWARE

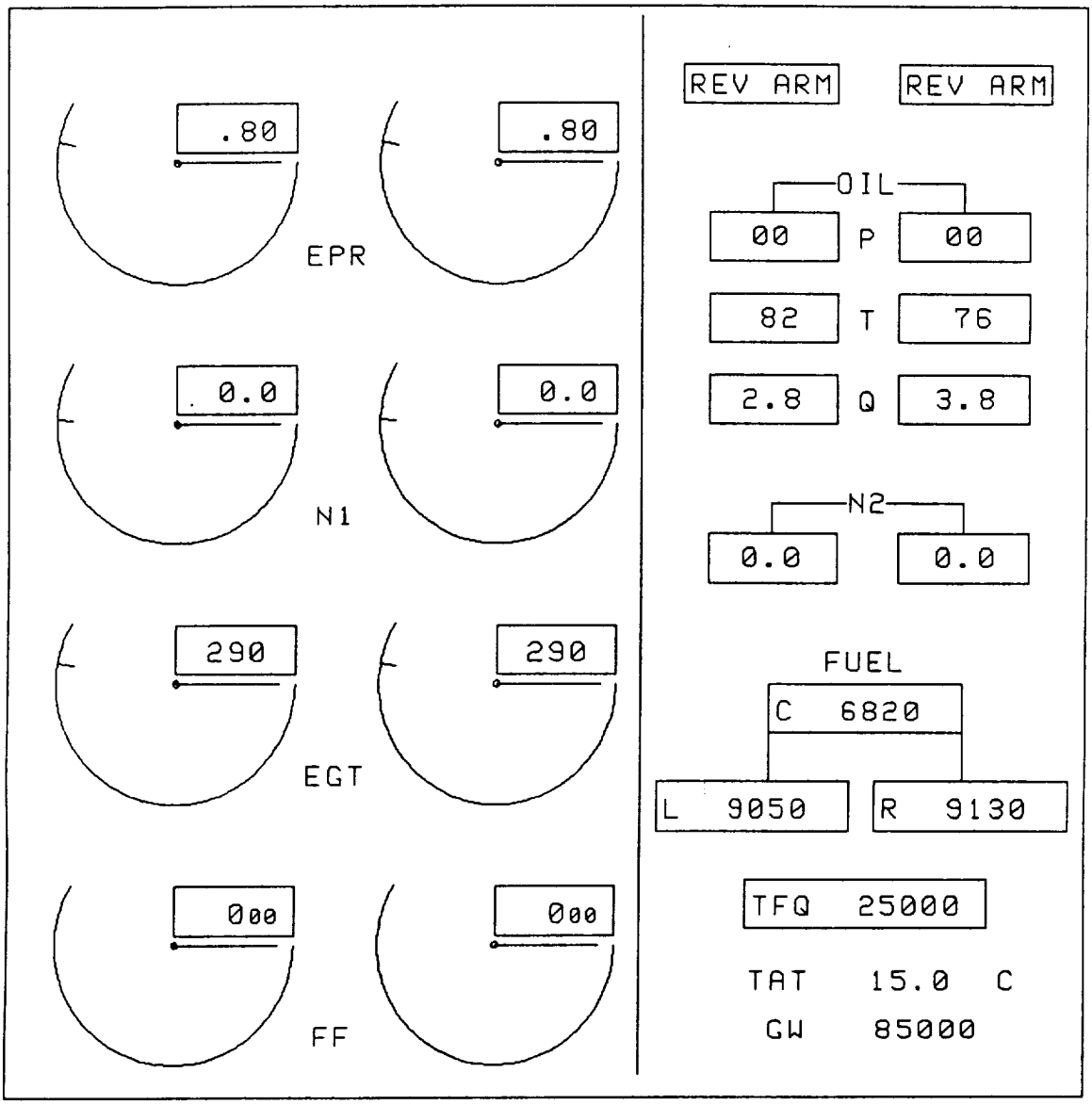
The ENG format provides a graphic representation of quantities associated with the aircraft engines. The format is split into two distinct sections; simulated engine gauges on the left and numeric displays on the right. Refer to the figure 8.1 for the engine format layout.

The gauges are displayed as four sets of pairs, situated side by side. Each set depicts the same engine parameter, but pertaining to the left and right engines. A gauge has a digital readout, an arc section representing the valid range of values for the particular quantity, and a radial pointer positioned along the arc at the appropriate location for the current value of the quantity. The four engine parameters shown in this manner are the engine pressure ratio, N1 RPM percentage, exhaust gas temperature, and fuel flow rate.

The right hand section of the display screen has numeric values of engine quantities displayed within rectangular boxes. Four miscellaneous items, left/right thrust reverser armed messages, total air temperature, and aircraft gross weight, are also shown in this section. The engine oil pressure, temperature, and quantity along with the N2 ratio are shown in left and right pairs. The amount of fuel remaining is given for the left, right, and center tanks along with the total amount of the three tanks.

This format does not utilize any of the bezel panel buttons or potentiometers.

ENGINE DISPLAY FORMAT



-figure 8.1-

Section 8.1 ENGINE PROCEDURES

There are five procedures dedicated to creating the output buffer data needed to display the engine format. The following chart lists the procedures along with the source code language and relative size. Those modules that serve as utility subroutines to another procedure are shown with their caller. The size is the percentage of total ENG software memory usage.

Refer to Appendix A to identify the memory locations in the output buffer (OUTDAT) that are used by the engine routines. The mnemonic "ENG" identifies those locations. Descriptions of the engine modules appear on the following pages.

MODULE	SOURCE	SIZE
ENGEXC	FORTRAN	67%
EPR_F1	FORTRAN	
EPR_F2	FORTRAN	
FFPRC	FORTRAN	21%
FTEST	FORTRAN	12%

MODULE NAME: ENGEXC
FILE NAME: ENGEXC.FOR
PROCESS: DSPFST
CALLED BY: DSPFST
CALLING SEQUENCE: CALL ENGEXC

PURPOSE:

To scale and store the engine format's parameters as integers in the display output buffer.

DESCRIPTION:

ENGEXC manages the processing of the data used as input parameters to the engine display format, and stores it into the display output buffer. Floating point data is scaled and stored as integers. Some of the engine parameters have values for both the left and right engines. These parameters are shown next.

Engine pressure ratio
Engine exhaust gas temperature
Engine N1
Engine fuel flow
Engine oil pressure
Engine oil temperature
Engine oil quantity
Engine N2
Engine thrust reversers armed

Many of the parameters mentioned above have fixed warning limit and caution values. They are also scaled and stored in the output buffer. The fixed values are as follows.

Exhaust gas temperature limit	(570 deg)
Exhaust gas temperature caution	(535 deg)
Engine N1 limit	(100.1 %)
Engine N1 caution	(94.0 %)
Engine oil pressure lower limit	(35 psi)
Engine oil pressure upper limit	(55 psi)
Engine oil temperature limit	(157 deg)
Engine oil quantity limit	(1.0 gal)
Engine N2 limit	(100.0 %)

The engine pressure ratio (EPR) warning limit is fixed at 2.0 when the airplane is travelling less than 64 knots, otherwise the limit is determined via lookup tables. There are two sets of tables: one for cruise limits (flaps set at zero), and the other for takeoff limits (flaps non-zero). The functions EPR_F1 and EPR_F2 perform the table lookups based on current air temperature and altitude. The EPR caution value is set to the EPR limit value minus 0.2.

To display the fuel quantities for the left, center, and right fuel tanks requires extra processing. The algorithm used is based on the premise that fuel is drawn from the center tank until less than 100 pounds remains, then it is drawn from the left and right tanks equally. The left and right tanks are initialized to 9050 and 9130 pounds respectively. The center fuel tank display will show the difference between the total fuel quantity and these two values. When the center tank display reaches 100 pounds, it will freeze at that value and fuel will be siphoned from the left and right tanks. The 80 pound difference between the two tanks remains as fuel is drawn equally from them.

The engine N1 and N2 percentages for both the left and right engines are also displayed on the engine format. The percentages indicate the ratio of the rate of change of the N1 and N2 counters to the maximum rate of change. A 70 unit per second rate of change in the counter corresponds to a 100% N1 or N2 value. Since the N1 and N2 counters change rather slowly, they are sampled only every .5 seconds. This slower rate allows for more accuracy. The sampled values minus the last counter values will yield the units changed. The ratio of these differences to the maximum 35 units per half-second rate will be the percentage displayed. Finally, the N1 and N2 percentages are filtered, scaled, and stored as integers into the display output buffer.

The left and right engine thrust reverser messages are shown on the upper right side of the engine format. They will be displayed when the thrust reversers have been armed. ENGEXC checks the two discretes which indicate whether they have been armed, and sets bits in one of the packed discrete words if they are.

Another discrete word bit is set if the aft flight deck is engaged. This bit serves as a flag to the microprocessor code to disallow the EPR radials from turning red when they enter the warning zone while the aft flight deck is engaged. This code was implemented to suppress red warning indications during takeoff.

All of the engine output parameters discussed above are assumed to have valid values. The bits in the packed discrete words used by the engine format, indicating that these parameters are valid, are always set on.

GLOBAL REFERENCES:

VARIABLES

AEEF AP_WEIGHT* EGTL EGTR EGT_CAUTION* EGT_LEFT* EGT_LIMIT* EGT_RIGHT* ENG_REVERSERS* ENG_VALID_1* ENG_VALID_2* ENG_VALID_3* EOPRSL EOPRSR EPR1 EPR2 EPR_CAUTION* EPR_LEFT* EPR_LIMIT EPR_RIGHT* FF5LF FF5RF FLAP FLOW_LEFT* FLOW_RIGHT* FTFQ FUEL_LEVEL_APU*

```
   FUEL_LEVEL_LEFT* FUEL_LEVEL_RIGHT* HBARO LREV N1_CAUTION*
   N1_LIMIT* N2_LIMIT* NAV64K OIL_LEVEL_LEFT* OIL_LEVEL_LIMIT*
   OIL_LEVEL_RIGHT* PRESSURE_HIGH* PRESSURE_LEFT*
   PRESSURE_LOW* PRESSURE_RIGHT* RREV SYS_WARN_WORD TAT
   TEMP_LEFT* TEMP_LIMIT* TEMP_RIGHT* TOTAL_AIR_TEMP* WEIGHT

ARRAYS
   ENG_DATA FUQTY OUTDAT*

FUNCTIONS AND SUBROUTINES
   EPR_F1 EPR_F2
```

MODULE NAME: EPR_F1
FILE NAME: ENGEXC.FOR
PROCESS: DSPFST
CALLED BY: ENGEXC
CALLING SEQUENCE: EPR_F1 (TAT, HBARO)

PURPOSE:
To find the appropriate flight manual maximum cruise EPR, given the current air temperature and altitude.

DESCRIPTION:
This function determines the maximum cruise EPR (engine pressure ratio) setting from the flight manual. There are two EPR values looked up in this function. One EPR value is obtained by using the current air temperature as a reference into a table of EPR values that is arranged according to temperature intervals. A second EPR value is fetched using altitude as the reference into a table based on altitude intervals. The lower of the two values looked up is used as the maximum cruise EPR setting.

MODULE NAME: EPR_F2
FILE NAME: ENGEXC.FOR
PROCESS: DSPFST
CALLED BY: DSPFST
CALLING SEQUENCE: EPR_F2 (TAT, HBARO)

PURPOSE:

To find the appropriate flight manual maximum takeoff EPR, given the current air temperature and altitude.

DESCRIPTION:

This function determines the maximum takeoff EPR (engine pressure ratio) setting from the flight manual. There are two EPR values looked-up in this function. One EPR value is obtained by using the current air temperature as a reference into a table of EPR values that is arranged according to temperature intervals. A second EPR value is fetched using altitude as the reference into a table based on altitude intervals. The lower of the two values looked up is used as the maximum takeoff EPR setting.

MODULE NAME: FFPRC
FILE NAME: FFPRC.FOR
PROCESS: DSPFST
CALLED BY: DSPFST
CALLING SEQUENCE: CALL FFPRC

PURPOSE:

To calculate fuel consumption, remaining fuel quantity, and aircraft total weight.

DESCRIPTION:

FFPRC requires a manual entry of aircraft gross weight (GRWGT) to initiate processing. If GRWGT is set to zero, processing is bypassed. Otherwise, GRWGT is compared to the previous value and if a new entry has been made then initialization processing takes place. A change in GRWGT prompts a full initialization, which begins by saving the new gross weight in I_WEIGHT, computing the aircraft empty weight (E_WEIGHT) as GRWGT minus the measured total fuel quantity (TFQ), setting the FUELUP discrete and zeroing the total fuel used (TFU). Several variables used in measuring and filtering the fuel flow are then computed. KDEN and KTF are computed as the product of the user adjustable nominal fuel density (FDEN) and the constants KKDEN and KKTF respectively. KDEN and KTF are used by the FINPT subroutine of DISFIL to produce the FU vector (lbs of fuel used in the last sample period for each engine and the APU) from the fuel temperature and fuel meter inputs. The fuel flow filter constant (KT) is computed from the user adjustable time constant (TAUFF). The fuel quantity filter constant (TAUFQ) and the lb/sample to lb/hour conversion factor (SF) are computed from the user selectable sample interval (MXITER). Finally, the interaction count (ITER) is set to the negative of MXITER, FTFQ is set to the measured fuel quantity (TFQ) and DELTA_F (the difference between TFQ and FTFQ) is set to zero.

Normal processing begins by incrementing ITER. This counter was initialized to -MXITER, which is the signal for FINPT to initialize itself. Subsequently, FINPT takes a set of readings each time ITER becomes zero. FFPRC processes these readings when ITER + 1 = 1, i.e., later in the same major frame. Total fuel used (TFU) is then computed as the integral of the sum of the FU vector. The filtered total fuel remaining (FTFQ) is then computed in a complementary filter driven by FU and corrected by TFQ with a 30 second time constant. Aircraft weight is then set to the sum of E_WEIGHT and FTFQ, and FU is rescaled to fuel flow in lbs/hr for display.

Finally, the filtered fuel flow quantities for left and right engines and the APU are computed from FU. When ITER becomes equal to MXITER, it is reset to zero, which reinitializes the cycle.

GLOBAL REFERENCES:

VARIABLES
FDEN FF5AF* FF5LF* FF5RF* FTFQ FUELUP* GRWGT ITER* KDEN* KT KTF* MXITER TAUFF TFQ TFU* WEIGHT*

ARRAYS
FU

FUNCTIONS AND SUBROUTINES
MTH$EXP

MODULE NAME: FTEST
FILE NAME: FTEST.FOR
PROCESS: DSPFST
CALLED BY: DSPFST
CALLING SEQUENCE: CALL FTEST

PURPOSE:
To simulate some of the engine parameters needed to drive the engine display format.

DESCRIPTION:
Some of the aircraft inputs used by the engine display software are not provided by the real-time flight simulation in the Experimental Avionics Systems Integration Laboratory (EASILY) Lab. The engine parameters affected by the lack of these inputs are: total fuel quantity, airplane weight, left and right engine fuel flows, and left and right engine N1 and N2 percentages. FTEST was created to simulate these values in the EASILY testing environment only. The calculations to simulate the parameters are based on the average EPR value between the left and right engines, so that the affected engine values will change along with a change in EPR. However, no attempt has been made to make these simulated values accurate, and should not be viewed as the actual values that would be displayed under the same conditions in-flight. The purpose is merely to make the engine parameters listed above seem reasonable when testing in the EASILY.

FTEST is called by DSPFST only when the boolean LABFLG has been set.

GLOBAL REFERENCES:

VARIABLES
EPR1 EPR2 FF5LF FF5RF* FTFQ GRWGT N1LEFT N1RGHT* N2LEFT* N2RGHT* WEIGHT*

Section 9.0 SYSTEM WARNING DISPLAY SOFTWARE

The data required to drive this format is created by only one procedure, SYSEXC. A description of this procedure is included on the following page. Refer to Appendix A to to find the locations in the display output buffer (OUTDAT) used by this format. The mnemonic "SYS" is used to identify buffer words used by the system warning format.

MODULE NAME: SYSEXC
FILE NAME: SYSEXC.FOR
PROCESS: DSPFST
CALLED BY: DSPFST
CALLING SEQUENCE: CALL SYSEXC

PURPOSE:

To process data and store in the output buffer the inputs needed by the System Warning format to display warning messages, and information on flap setting and gear position.

DESCRIPTION:

One of the duties of the system warning format is to display valid system messages. SYSEXC scans a series of discrete words whose bits correspond individually to a set of predefined messages available to be displayed. When a bit has been set, the appropriate text is shown. System warning messages fall into three categories: warning, caution, and special. The following table indicates the available messages for each type.

warning:	"RFD DISENGAGED"
caution:	"STB OUT OF TRM"
	"SPD BRK NO ARM"
	"FLAP LIMIT"
	"THROTTLE LIMIT"
	"SPD BRKS SYNC"
	"AILERON LIMIT"
	"ELEVATOR LIMIT"
	"RUDDER LIMIT"
	"SPD BRK ARM"
special:	"NAV 2 TUNING"
	"DME 2 TUNING"
	"COMM TUNING"

Also, SYSEXC checks to see if the flaps are moving by comparing the filtered actual flap position to the flap handle position. If the two values vary by a specified tolerance value, SYSEXC sets the appropriate bit in the display output buffer to indicate that the flaps are moving. The tolerance will be either .25 degrees for flap handle positions below 5, or 15% of the flap handle value for flap settings 5 or greater. SYSEXC also scales the value of the flap position to a Standard Angle Format (SAF) and stores the scaled position in the proper place in the display output buffer.

Lastly, SYSEXC performs some logic to determine what type of symbology should be shown to describe the position of each gear (nose gear, right gear, left gear). Different symbology is displayed, depending upon which bits are set in the display output buffer. The following chart describes which symbology is displayed for the nose gear given the following combinations of bit settings in the buffer word OUTDAT(698).

BITS 4	3	2	1	0	SYMBOL
0	0	1	1	0	GREEN 'DN'
0	0	1	0	1	WHITE 'UP'
0	0	0	1	0	RED DOWN ARROW
1	0	0	0	0	YELLOW DOWN ARROW
0	0	0	0	1	RED UP ARROW
0	1	0	0	0	YELLOW UP ARROW
ALL OTHER COMBINATIONS					RED 'X'

(Right and left gear symbology is displayed under the same bit patterns, but a different range of bits: bits 5-9 of OUTDAT(698) representing the left gear, and bits 10-14 representing the right gear.)

The following combinations of discrete values, representing the sensor data received from the airplane, yield the indicated symbols for the nose gear display.

GUPCMD gear up commanded	GDNCMD gear down commanded	NGRRED nose gear not locked	NGRDN nose gear down	SYMBOL
F	T	F	T	GREEN 'DN'
T	F	F	F	WHITE 'UP'
F	T	F	F	RED DOWN ARROW
F	T	T	T	RED DOWN ARROW
F	T	T	F	YELLOW DOWN ARROW
T	F	F	T	RED UP ARROW
T	F	T	T/F	YELLOW UP ARROW
ALL OTHER COMBINATIONS				RED 'X'

The same symbology would be displayed for right and left gear with the substitution of the following booleans as the last two items in the table above:

LEFT - LGRRED, LGRDN
RIGHT - RGRRED, GRGDN

GLOBAL REFERENCES:

VARIABLES
AILCMO ELVCMO FLAP FLPPLC FLPPOS GDNCMD GEAR_WORD GRPOS*
GUPCMD INBD LEFT_FLAP* LGRDN LGRRED NAV_VLD_2 NGRDN NGRRED
OUTBD RCOMT RCTRLD RGRDN RGRRED RIGHT_FLAP RNAV2T RUDCMO
RXPDRT SPBPLC SPDARM SPDNAR STBTRM SYS_1* SYS_2* SYS_3*
SYS_4* SYS_7* SYS_8* THRPLC

Section 10.0 SPERRY PFD SOFTWARE

The Primary Flight Display designed by Sperry requires a small amount of processing in the host computer. Only one module, PFDEXC, creates data specifically for this format. Appendix A shows the display output buffer locations dedicated to the Sperry PFD format. The mnemonic "F1" is used to refer to this format. The next page contains a description of PFDEXC.

MODULE NAME: PFDEXC
FILE NAME: PFDEXC.FOR
PROCESS: DSPFST
CALLED BY: DSPFST
CALLING SEQUENCE: CALL PFDEXC

PURPOSE:

To process and store in the NAV background buffer the parameters required by the Sperry PFD format.

DESCRIPTION:

The following aircraft parameters are required by the Sperry PFD format only. They are scaled and stored as integers in the displays output buffer (OUTDAT).

Sea level barometric pressure setting
Aircraft pitch angle
Rate of change in altitude
Vertical guidance deviation
Horizontal guidance deviation

This is not a complete set of the parameters referenced however. Some aircraft inputs used by the Sperry PFD have already been made available in the 704 word displays buffer by other formats which share the values.

Certain parameters used are always assumed to have valid values. The valids always turned on by PFDEXC are:

attitude valid
altitude valid
vertical speed valid
vertical deviation valid
lateral deviation valid

These valids correspond to bits set in the Sperry PFD discrete word PFD_VALID, also sent to the microprocessor via the output buffer.

PFDEXC does some processing for its airspeed display. If the current airspeed value is valid, bits in PFD_VALID are immediately set on corresponding to the airspeed valid, and airspeed limit valid.

The airspeed limit value is determined by PFDEXC. Fixed airspeed limits exist in a local data table for flap settings of 0, 1, 2, 5, 10, 15, 25, 30, and 40 - with the exception that the limit value for flaps 0 will be one of two values depending on whether the gear is down or up. Using the current flap position, an index into the table is figured, and interpolation is used, to compute the appropriate airspeed limit value.

One final bit of processing done for the airspeed display is to set on a valid bit in PFD_VALID when an airspeed has been selected on the pilot's CDU.

PFDEXC does some processing for the altitude display as well. If the current radar altitude value is valid, an appropriate bit in the valid word is set on, and the radar altitude value is scaled and stored in the displays output buffer. Also, a valid bit is set on in PFD_VALID when an altitude has been selected on the pilot's CDU.

GLOBAL REFERENCES:

VARIABLES
ALTSEL BARSET BETAH CASV ETAH FLAP FLPPOS GEAR HDCF HRAD HRV IASSEL PFD_BARO_SET* PFD_CASLMT* PFD_HDOT* PFD_HOR_DEV* PFD_HRAD* PFD_PITCH* PFD_VALID* PFD_VER_DEV* PITCH SYS_7

Section 11.0 TAKEOFF PERFORMANCE MONITORING SYSTEM (TOPMS)

The Takeoff Performance Monitoring System display format is used only during the takeoff phase of flight. A drawing of the format can be found on the following page. The TOPMS format provides critical takeoff information to the pilot, and allows the pilot to monitor the performance of the aircraft during the takeoff roll. It visually shows where along the runway the airplane should reach the V1 (decision speed) and VR (rotate speed) positions. The TOPMS software also performs computations to determine an appropriate takeoff advisory status, which the pilot may use in determining whether to complete the takeoff or abort it. The advisory status has four possible states, "must abort", "must takeoff", a warning state, and a state indicating everything is within normal range.

The TOPMS display has two modes, takeoff and abort. Different amounts of information are provided depending on the mode - although the runway symbol and identifier, speed value, and aircraft symbol will be shown regardless. Note that the speed value will be airspeed in takeoff mode, and ground speed in abort mode. The abort mode display is the simpler of the two. There are two critical symbols shown during the abort. One is the symbol indicating where the aircraft will stop along the runway if maximum braking is used (not including reverse thrust). It is comprised of a circle with a star in the middle. The second critical symbol (shaped like a football) represents where the stop point will be at the current level of braking.

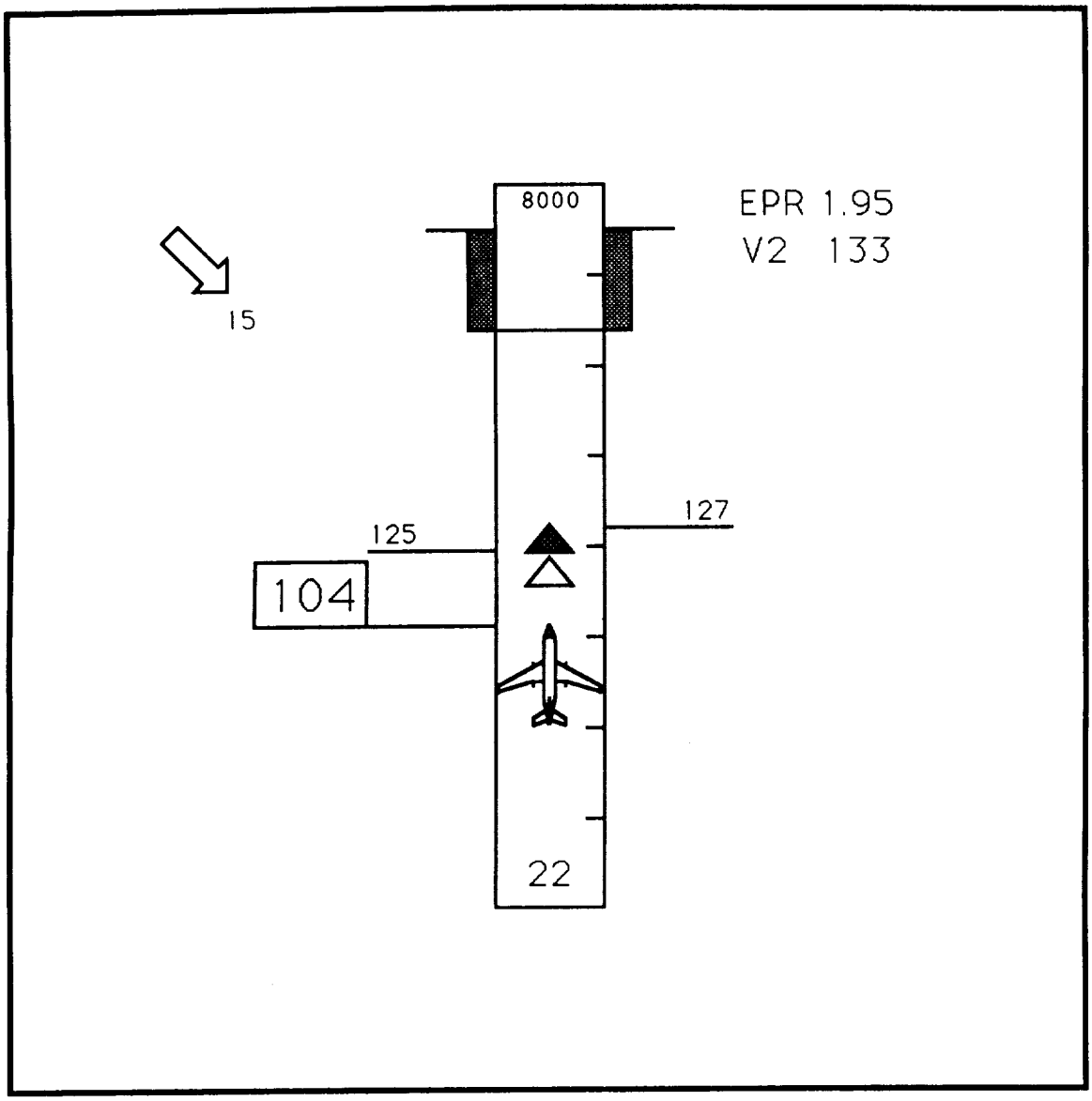
The takeoff mode display contains a lot more information than the abort mode. The parameters always displayed in this mode are as follows.

flight manual EPR setting
V1, V2, VR speeds
aircraft position on runway
wind speed, direction
takeoff advisory status
engine performance bars
takeoff roll limit

Under certain conditions the stop point using maximum braking is shown too. Figures 11.1 and 11.2 depict the takeoff and abort modes of the TOPMS display format.

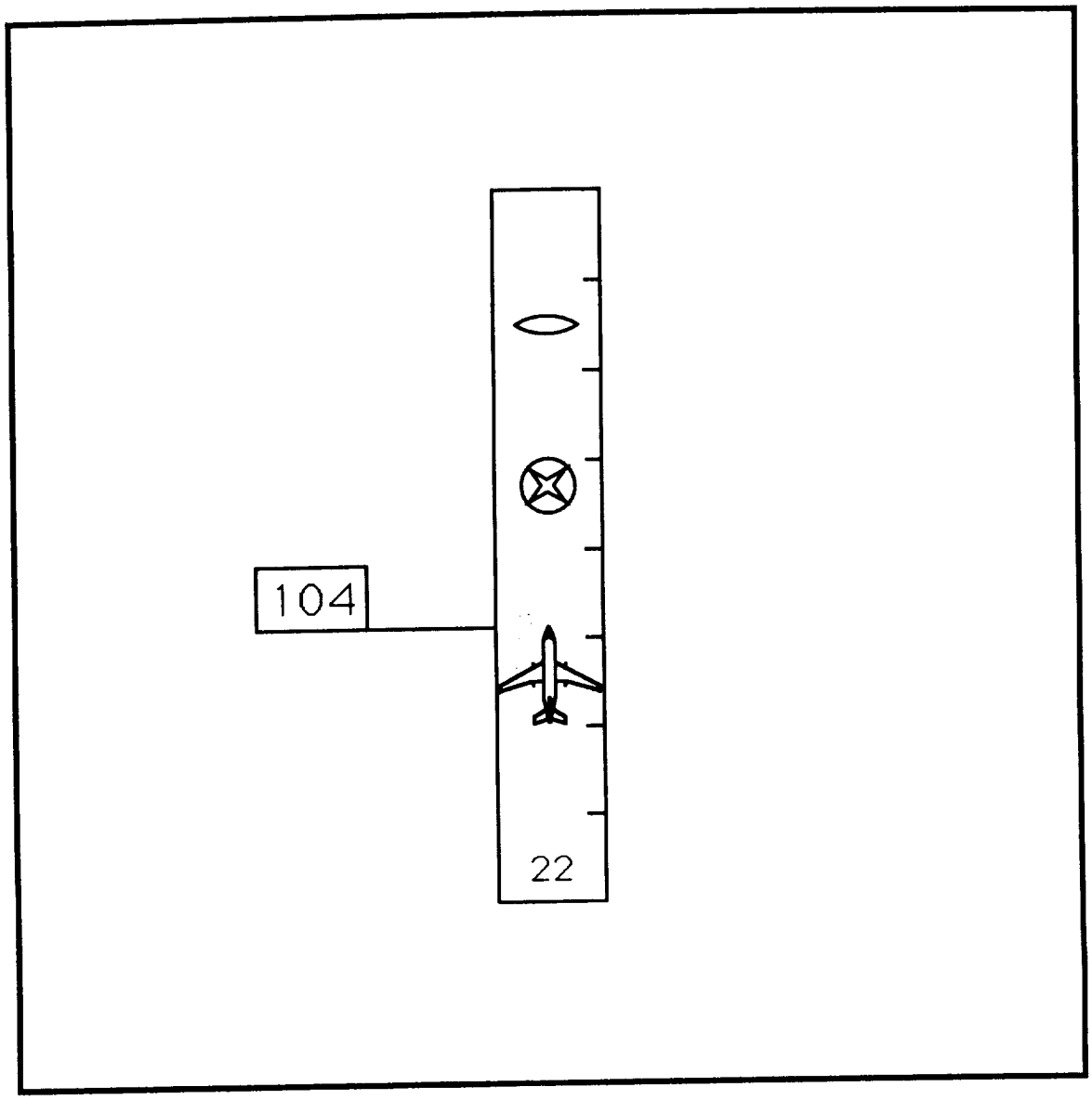
The TOPMS software is divided into two parts, the pretakeoff segment and the real-time segment. The pretakeoff software generates a takeoff profile based on CDU entries made by the pilot. The profile will be used to compare to the actual performance of the airplane during takeoff. The real-time software computes the actual takeoff

performance, and stores the required data into the displays output buffer that will generate the real-time takeoff (or abort) display. Both the pretakeoff and real-time software segments and their associated modules are discussed in sections 11.1 and 11.2. These sections should be referred to for a more detailed software description.



Takeoff Performance Monitoring System
(takeoff mode)

-figure 11.1-



Takeoff Performance Monitoring System
(abort mode)

-figure 11.2-

Section 11.1 PRETAKEOFF BACKGROUND SOFTWARE

The pretakeoff calculations for the Takeoff Performance Monitoring System (TOPMS) are performed to provide a benchmark for the TOPMS real-time software. A takeoff profile is generated using aircraft parameters and the local ambient conditions. The actual takeoff performance can be compared to the pretakeoff profile to create advisory information for the aircraft flight crew.

The TOPMS pretakeoff procedures execute after several CDU entries have been completed by the flight crew. These entries are shown below.

AIRFIELD/RUNWAY	ROUTE page.
AIRCRAFT WEIGHT	PERFORMANCE page.
FLAP SELECTION	TAKEOFF page.
TEMPERATURE	TAKEOFF page.
AIRCRAFT CG	TAKEOFF page
WIND SPEED/DIRECTION	TAKEOFF page
RUNWAY FRICTION	TAKEOFF page; default to .015
RUNWAY START OFFSET	TAKEOFF page; default to 200 feet
RUNWAY LENGTH	TAKEOFF page; default to database value

From these entries CDU software computes the stabilizer trim position, decision speed (V1), rotate speed (VR), and second segment climb speed (V2). The CDU values are all sent to the display MicroVAX computer through the interprocessor I/O link for use by the TOPMS pretakeoff modules. In addition the pretakeoff modules use the pressure altitude, obtained from the aircraft DATAC sensor bus, as a final external input parameter.

The flag TOPMS is used to control the sequence of events involved in the pretakeoff calculations. When it is set to "1" the module PRETKF is called by the background software executive DSPSLW. When the computations are complete the flag is set to "2" to indicate that the computed values are ready for the real-time TOPMS software. The following list describes the parameters that are created for use by the real-time TOPMS modules.

RWYV1	Runway distance required to reach V1.
RWYVR	Runway distance required to reach VR.
ROLLIM	Runway distance to the takeoff roll limit
COEF()	Polynomial coefficients for the takeoff profile which provides expected acceleration as a function of true airspeed.

Nine procedures are dedicated to the TOPMS pretakeoff calculations. The following chart lists the procedure names along with their source code language and relative size. Those modules that serve as utility subroutines to another procedure are shown with their caller. The size provided is the percentage of total pretakeoff software memory usage, excluding utility modules from TOPMS.OLB. The following pages include module descriptions for each of the nine procedures. The information provided is limited since much of this software was extracted from portions of the NASA's Boeing 737 aircraft simulator program. More detailed information may be obtained from the people responsible for maintaining the simulation.

MODULE	SOURCE	SIZE
ACTRIM	FORTRAN	5%
AEROC	FORTRAN	8%
ATMOS	FORTRAN	1%
ENGINE	FORTRAN	9%
LNGFM	FORTRAN	9%
POLYFT SIMEQA	FORTRAN	7%
PRETKF	FORTRAN	60%
THROTS	FORTRAN	2%

MODULE NAME: ACTRIM
FILE NAME: ACTRIM.FOR
PROCESS: DSPSLW
CALLED BY: PRETKF
CALLING SEQUENCE: CALL ACTRIM(X,Y,ICTRIM,YTOL,IFLAG)

PURPOSE:
To compute initial aircraft trim positions.

DESCRIPTION:
This procedure is used to compute aircraft trim values for pitch angle, and altitude of the center of gravity above the runway. Initial pitch and altitude values are passed as the "X" vector in the calling parameter list. The "Y" vector contains the landing gear weight distribution parameters for the current aircraft center of gravity, weight, and airplane body orientation. These arrays are used in the solving of matrix equations to produce static trim values. ACTRIM proceeds until the columns of the simultaneous equations matrix are forced close to zero. The tolerance value YTOL is used to determine when convergence has been detected. If divergence is detected the module returns with an error status in IFLAG. ACTRIM may be called iteratively from PRETKF in an attempt to gain acceptable trim values. The flag ICTRIM is set to force an initialization pass in ACTRIM the first time it is called.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
SIMEQA

MODULE NAME: AEROC
FILE NAME: AEROC.FOR
PROCESS: DSPSLW
CALLED BY: PRETKF
CALLING SEQUENCE: CALL AEROC(ALPDEG, FLAG)

PURPOSE:

To generate parameters for aeronautical derivatives.

DESCRIPTION:

This procedure is called to compute many aircraft aeronautical derivatives which are a function of angle of attack, center of gravity, and flap setting. The angle of attack value is passed as a calling parameter while the other inputs are global common variables. When the first pass variable (FLAG) is set, all derivative variables are set to the appropriate values. On subsequent calls to AEROC many of the variables do not need to be set again. This is done because these derivatives are dependent on flap setting and aircraft CG, which are static for one execution of the pretakeoff calculations. When FLAG is not set only those derivatives which are dependent on the current angle of attack are computed.

GLOBAL REFERENCES:

VARIABLES
CDBAS* CG CLAL* CLALD* CLBAS* CLDE* CLDS* CLGM* CLNZ* CLO* CLQ* CMAL* CMALD* CMBAS* CMDEM* CMDSM* CMFAC* CMGE* CMGM* CMNZ* CMO* CMQ* DCDG* DCDGE* DCLG* DCLGE* DCMG* DLCLTR* FGEL* FKA* GEARF* TOFLPS

FUNCTIONS AND SUBROUTINES
LIB$SIGNAL ONED

MODULE NAME: ATMOS
FILE NAME: ATMOS.FOR
PROCESS: DSPSLW
CALLED BY: PRETKF
CALLING SEQUENCE: CALL ATMOS

PURPOSE:
To compute atmospheric parameters.

DESCRIPTION:
This subroutine uses pressure altitude and temperature to calculate temperature ratio, pressure ratio, speed of sound, and air density.

INPUTS:

PALT	pressure altitude in feet
SOAT	temperature in degrees Celsius

OUTPUTS:

THTAMB	ambient temperature ratio
DLTAMB	ambient pressure ratio
A	speed of sound in FT/SEC
RHO	density in SLUGS/CU.FT
RTEMP	ambient temperature in degrees Rankine

GLOBAL REFERENCES:

VARIABLES
A* ALT_PRE DLTAMB RHO* RTEMP SOAT THTAMB

FUNCTIONS AND SUBROUTINES
MTH$SQRT

MODULE NAME: ENGINE
FILE NAME: ENGTKF.FOR
PROCESS: DSPSLW
CALLED BY: PRETKF
CALLING SEQUENCE: CALL ENGINE(DTH,THRUST,EPR,SPD)

PURPOSE:
To perform engine modeling computations.

DESCRIPTION:
This procedure performs computations which model the dynamics of the JT8D-7 aircraft engine. ENGINE is passed throttle position and airspeed as calling parameters. When the global simulation time variable "T" is zero all the engine parameters are initialized to their idle values. The airspeed, throttle position, and ambient atmospheric conditions are used to calculate the engine thrust and pressure ratio, which are returned to the caller in the parameter list.

GLOBAL REFERENCES:

VARIABLES
A ALT_PRE DLTAMB T THTAMB

FUNCTIONS AND SUBROUTINES
DZONE OLIMIT ONED RATE THCORF XLIM

MODULE NAME: LNGFM
FILE NAME: LNG2D.FOR
PROCESS: DSPSLW
CALLED BY: PRETKF
CALLING SEQUENCE: CALL LNGFM(OPERATE)

PURPOSE:
To compute the landing gear forces and moments.

DESCRIPTION:
This procedure computes landing gear forces and moments for the left, right, and nose gear struts. The sole input parameter is used as an initialization flag. All computed values are returned through global common memory locations.

GLOBAL REFERENCES:

VARIABLES
ALTR CG COSTHE HDOT LGLB* LGMB* LGNB* LGXB* LGYB* LGZB*
MUROL SINTHE SQLMG* SQNG* SQRMG* THEDOT TPALT

FUNCTIONS AND SUBROUTINES
ONED XLIM

MODULE NAME: POLYFT
FILE NAME: POLYFT.FOR
PROCESS: DSPSLW
CALLED BY: PRETKF
CALLING SEQUENCE: CALL POLYFT(X,Y,NPTS,COEF)

PURPOSE:
To perform polynomial curve fitting.

DESCRIPTION:
This subroutine performs a polynomial curve fit on the X-Y data sets which are passed as calling parameters. A polynomial of order "NPOLY" is fitted minimizing the sum of the squared error.

INPUTS

X(NPTS)	independent variable array
Y(NPTS)	dependent variable array
NPTS	number of sets of data points
NPOLY	order of polynomial (included constant)

OUTPUTS

COEF(NPOLY+1)	polynomial coefficients

The computed coefficients generate the solution to the following polynomial equation.

```
Y = COEF(0) + COEF(1) [ X**1 + COEF(2) [X**2 + ....
                    ......+ COEF(NPOLY) [X**NPOLY
```

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
SIMEQA

MODULE NAME: SIMEQA
FILE NAME: POLYFT.FOR
PROCESS: DSPSLW
CALLED BY: POLYFT, ACTRIM
CALLING SEQUENCE: CALL SIMEQA(B, N, C)

PURPOSE:
To solve a system of linear equations.

DESCRIPTION:
This subroutine uses matrix reduction to solve a system of linear equations. The parameter "N" is the number of rows in the matrix "B" and the number of elements in the vector "C".

GLOBAL REFERENCES:
none

MODULE NAME: PRETKF
FILE NAME: PRETKF.FOR
PROCESS: DSPSLW
CALLED BY: DSPSLW
CALLING SEQUENCE: CALL PRETKF

PURPOSE:
To perform TOPMS pretakeoff calculations.

DESCRIPTION:
This procedure is the main module for the TOPMS pretakeoff calculations. It performs internal simulations of the takeoff roll using the ambient conditions provided by the aircraft sensors and the flight crew CDU entries. Two complete simulations are made using different coefficients of runway friction (.04, .005). Each run is a simulated 45 seconds of takeoff roll with 900 data samples of airspeed and acceleration saved. At the end of a run the sampled data is passed to a curve fitting subroutine to produce a third order polynomial which approximates the airspeed versus acceleration profile of the data points. After both simulation runs are complete, the flight crew friction estimate entered on the CDU is used to interpolate a final set of polynomial coefficients from those created for the upper and lower runway friction values. These values are used to generate the remaining pretakeoff outputs described in the start of this section.

GLOBAL REFERENCES:

VARIABLES
A ALPHA ALT ALTR* ALT_PRE ARMSPB* ATOBRK* CD CDBAS CG CL CLAL CLALD CLBAS CLDE CLDS CLGM CLNZ CLO CLQ CMAL CMALD CMBAS CMDEM CMDSM CMFAC CMGE CMGM CMNZ CMO CMQ COSALP COSPHI* COSTHE DCDFSP* DCDG DCDGE DCDGSP* DCLFSP* DCLG DCLGE DCLGSP* DCMG DELA* DELE DELR* DLCLTR DTHROT* EPRC FGEL FKA GEARF HDOT* LGMB LGXB LGZB MUBC* MUROL* MURWY ORGHDG ORGLEN PB* PBDOT* PHI PHIDOT* QB QBDOT* RB* RBDOT* RHO ROLARM* ROLLIM* RWYV1 RWYVR* SINALP SINPHI* SINTHE SOAT SQLMG SQNG SQRMG T THEDOT* THETA THRIDL* THRST* TOFLPS TOSTAB TOWD TOWS TPALT UB UBDOT V1 V1STOP VB* VBDOT* VR VWK WB WBDOT WEIGHT WGHT_PRE YTOL

ARRAYS
ALTVEC* COEF* COEFS QBVEC* THVEC* UBVEC* WBVEC*

FUNCTIONS AND SUBROUTINES
ACTRIM ADBW2 AEROC ASPDCO ATMOS ENGINE LIB$ESTABLISH LIB$SIGNAL LNGFM MTH$ASIN MTH$COS MTH$SIN MTH$SQRT MTH$TAN ONED POLYFT RWYPRD STPDIS THROTS THSRVO TOP_HDL

MODULE NAME: THROTS
FILE NAME: THROTS.FOR
PROCESS: DSPSLW
CALLED BY: PRETKF
CALLING SEQUENCE: CALL THROTS(TEMP,PALT,EPRC,DTHC,DTH14)

PURPOSE:
To find the recommended takeoff EPR and the throttle setting required to reach that value.

DESCRIPTION:
This subroutine picks the EPR value recommended by the B-737 flight manual for the given ambient temperature and pressure altitude based on the tabulation on page #3 (4B-1) of the flight manual for 737-1LT dated JAN 5/70.

The throttle setting needed to achieve this EPR under static conditions is then calculated by an iterative process using a relationship between commanded EPR, throttle setting and engine inlet stagnation temperature. This relationship was obtained from the JT8D-7 engine model as developed by Dave Williams.

INPUTS
TEMP	ambient temperature in degrees Celsius
PALT	pressure altitude in feet

OUTPUTS
EPRC	EPR to be commanded
DTHC	throttle setting needed to achieve EPRC (degrees)
DTH14	throttle setting for an EPR=1.4

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
FINTER

Section 11.2 TOPMS REAL-TIME SOFTWARE

The TOPMS real-time software generates the data needed to drive the real-time takeoff display. This software computes the actual performance of the airplane during takeoff, which will be compared to the takeoff profile generated during the pretakeoff segment of the TOPMS software. The real-time software handles both the takeoff and abort mode displays.

Three procedures are dedicated to the TOPMS real-time calculations. The following chart lists the procedure names along with their source code language and relative size. Those modules that serve as utility subroutines to another procedure are shown with their caller. The size provided is the percentage of total pretakeoff software memory usage, excluding utility modules from TOPMS.OLB. The following pages include module descriptions for each of the three procedures.

MODULE	SOURCE	SIZE
TOPEXC APLANE	FORTRAN	87%
FILL	FORTRAN	13%

The real-time computations start only after the pretakeoff software has completed successfully. The global variable "TOPMS" is used as an index to coordinate processing. When the index contains a value of 2 or 3 then real-time calculations are computed. TOPEXC is the TOPMS real-time executive, and is called twenty times per second. The subroutine FILL is called at the end of each pass of TOPEXC. FILL scales, converts, packs, and stores the TOPMS data required by the display format into the output buffer - including parameters computed by the pretakeoff segment.

Another global variable "ENABLE" is important to the real-time software. Bit settings within the lower byte of this word control some of the software processing in TOPEXC. A detailed description of the individual bits and their significance can be found at the end of section 11.

To produce the real-time TOPMS display certain values must be constantly updated. A list of those parameters are as follows.

airplane position on runway
measured airspeed in takeoff mode
distance to V1 in feet
distance to VR in feet
engine performance indicators

takeoff advisory status
mode switch (takeoff to abort)
stop distance w/current braking
stop distance w/max braking
ground speed in abort mode

Note that the current EPR settings are required by the TOPMS format also, but since they are stored in the output buffer by the engine format executive ENGEXC, TOPEXC does not need to process and store those values too.

MODULE NAME: FILL
FILE NAME: FILL.FOR
PROCESS: DSPFST
CALLED BY: TOPEXC
CALLING SEQUENCE: CALL FILL

PURPOSE:
To store in the displays output buffer the data required by the TOPMS format.

DESCRIPTION:
This routine performs the scaling and storing of TOPMS related data into the displays buffer OUTDAT. It also sets flags as necessary to help regulate the proper execution of the TOPMS software, which includes signaling the completion of a run.

The global variable "TOPMS" is used as an index to indicate what phase of TOPMS processing is currently being performed. The TOPMS format will continue to be displayed while the real-time TOPMS software in TOPEXC is being processed (index 2 or 3). However, in all other cases - TOPMS format deselected (index 0), pre-takeoff calculations underway (index 1), and end of run flagged (index 4) - the uppermost bit in the displays status word DISPST will be cleared. This word sent back to the FM/FC computer will cause the appropriate flags to be set which will turn off the TOPMS format.

One particular buffer word (OUTDAT index 619) is used to pack TOPMS related flags and indices. Flags are represented as certain bits within the word. Individual bits will be set on if any of the following conditions exist.

bit	condition
0	no pretakeoff computation errors
1	left engine not failed
2	right engine not failed
5	takeoff mode (off is abort mode)
6	left or right bleeds are off
7	TOPMS,actual flap setting not same

Bits 3-4 of the packed word contain a two bit index (values of 0 to 3) for the takeoff advisory status. The advisory status determines what type of symbol will be displayed at the end of the runway (i.e. green "must go" bar, yellow caution triangle, the red "must stop" sign, or no symbol for normal conditions).

Some of the TOPMS parameters are updated and stored in the output buffer every frame, while others only need to be stored one time per run. Values constantly updated are:

airplane position along runway
min stop distance w/max braking
stop distance w/current braking
measured airspeed in takeoff mode
ground speed in abort mode
distance to VR in ft ... (as long as VR
distance to V1 in ft ... not exceeded)

TOPMS values that are computed before the run starts and never change for the course of the run, need to be stored in the output buffer only once. These values include:

wind speed
wind direction - runway heading
runway length
target EPR for takeoff roll
V2 speed
roll limit line position
VR position on runway
VR speed
V1 speed
runway number

FILL updates the TOPMS phase index variable "TOPMS", when it begins storing the values only packed once per run. This indicates the end of the first pass initialization process. The routine TOPEXC will use this flag to bypass its first pass calculations from thereon.

GLOBAL REFERENCES:

VARIABLES
CASM DISPST* DISTP EPRC FLAP GSFPS LENGFL ORGHDG ORGLEN
RENGFL ROLLIM RWY2V1 RWY2VR RWYID RWYVR STATUS STOPD STOPE
TKOFF TOFLPS TOINDX TOPMS* TOPOS TOPST TOWD TOWS V1 V2 VR
BLEEDL BLEEDR

ARRAYS
OUTDAT*

FUNCTIONS AND SUBROUTINES
ANGL

MODULE NAME: TOPEXC
FILE NAME: TOPEXC.FOR
PROCESS: DSPFST
CALLED BY: DSPFST
CALLING SEQUENCE: CALL TOPEXC

PURPOSE:
Serves as the executive for the real-time portion of the TOPMS experiment.

DESCRIPTION:
TOPEXC is the real-time executive for the TOPMS software, and most of the actual real-time processing takes place in this subroutine. The purpose of TOPEXC is to check the airplane's progress during the takeoff roll. It constantly refigures aircraft position in terms of the runway already used up, and what is additionally needed to achieve rotation speed. This routine monitors engine health, acceleration performance, computes distances in feet required to reach V1 (decision speed) and VR (rotate speed) in feet, calculates the distances required to stop, and outputs a takeoff advisory status.

TOPEXC is called by DSPFST twenty times per second if the TOPMS format is on. However, no real-time processing takes place within TOPEXC until the pretakeoff software has completed successfully. The first pass through the subroutine body performs initializations for the takeoff run. When the software sequence variable "TOPMS" has a value of three, then the continuous real-time segment is underway, and continues for the course of the run. Among the things done during the initialization process is to call APLANE to make sure the flaps are set at either 1, 5, or 15. If they are not, the TOPMS sequence index is set to "end of run" (TOPMS = 4). Another situation that will cause the run to terminate, and is checked for at the beginning of each TOPEXC pass, is if the difference between the runway heading and aircraft true heading is more than 25 degrees.

Certain bits within the global variable "ENABLE" affect TOPEXC's processing. ENABLE provides the capability to select among a small set of options that allow some control over how the run will be conducted. Bits can be set in ENABLE, using the VIEW utility, prior to the start of a run. If set, the uppermost bit will cause the TOPMS internal simulation software to activate, and the body of TOPEXC to be bypassed. Bit 3 determines whether a pitch attitude correction is computed into the along track acceleration. Bit 1 controls the internal wind update. Bit 0 enabled will will cause a ground speed bias to be figured. Bit 2 selects the failure count for a performance failure to be either 5 consecutive frames or 8. Bits 4 and 5 determine the early on "must go" advisory distance. Figure 11.3, at the end of section 11, presents a graphic representation of all the ENABLE bit settings.

TOPEXC activates and uses two time counters during its processing. These counters are referenced throughout the subroutine to determine when specific processing should take place. One is the real-time TOPMS software clock (RTIME) which is started when the EPR's are advanced beyond 1.4 and the ground speed is greater than 5 ft/sec. The other clock, XSTIME, is activated at the actual start of the run which will be when the following conditions are met: the EPR's are greater than 1.4, throttles have been steady for at least 3 seconds, and the real-time clock XSTIME has has been going for at least 10 seconds. Once XSTIME has begun counting, the takeoff run is underway.

During the course of the run, the distance travelled, the distances to reach V1 and VR, and also the amount of runway required to stop, are constantly recomputed. Two one-time updates that occur during the run affect the V1, VR, and stop distance calculations. Approximately two seconds after the run starts the runway friction coefficient (XMU) and the scheduled performance basis (COEF) are re-updated based on the current takeoff run conditions. They are initially computed during the pretakeoff segment. The other update that occurs is a wind update. It takes place when the airspeed exceeds 68 knots. The wind model can be disabled using the ENABLE word.

Two important tasks of the real-time software are to monitor engine health and acceleration performance. The TOPMS display format is designed to signal engine and performance failures. To determine if a failure has occurred, actual engine parameters are compared to predicted ones. TOPEXC computes predicted EPR and acceleration values. If the difference between the actual and predicted values differ by more than their allowable limit a specified number of consecutive iterations, then an engine failure (EPR's), or a performance failure (acceleration) is flagged.

There are two stop distances computed by TOPEXC. One is the distance required to stop using the present level of deceleration (STOPE), and the other is the stop distance using maximum braking (STOPD). The former is always figured regardless of mode, while the latter is calculated only in abort mode. Abort mode is entered when the throttles are pulled back below 1.4 after the run has already begun.

One other responsiblity of this subroutine is to output a takeoff advisory, which is used as an indication of the takeoff situation. The advisory information is displayed on the TOPMS format at the end of the runway, and comes in one of four forms: a red stop sign warning, a yellow caution triangle (early on "must go" condition), a green "must go" bar, or no symbol at all for normal conditions. The following chart describes the situations that will cause each of the advisory indicators to be displayed.

Red Stop Sign
---
(1) VR point beyond roll limit line
(2) left and right engine failures
(3) one engine failure +
airspeed below V1
(4) acceleration performance failure

Yellow Early-On "Must Go" Triangle
---
(1) one engine failure +
airspeed beyond V1 +
not enough rwy to stop

Green "Must Go" Bar
---
(1) one engine failure +
airspeed beyond V1 +
not enough rwy to stop
(2) not enough rwy to stop

No Symbol
---
(1) VR point before roll limit line +
no performance failures +
no engine failures +
enough rwy to stop

The last thing TOPEXC does before exiting is to call the subroutine FILL, which will pack the TOPMS data into the output buffer for shipment to the display microprocessors.

GLOBAL REFERENCES:

VARIABLES
A ACCLF BDXACC CAS CASM CD CL DISTP DLTAMB DTHROT* ENABLE
EPR1 EPR2 EPRPL EPRPR GSFPS GSPD GSPD1 HDGTRU LENGFL
MAGVAR MURWY ORGHDG ORGLEN PITCH RENGFL RHO ROLLIM RWY2V1*
RWY2VR RWYV1 RWYVR SOAT STATUS* STOPD STOPE* THROTL THROTR
THRST TKOFF TOPMS* TOPOS TOPST TRKACC V1 VR VWK WEIGHT XMU

ARRAYS
COEF* COEFS

FUNCTIONS AND SUBROUTINES
ANGL APLANE ASPDCO EPRF FILL FINTER MTH$COS MTH$SIN RWYPRD
SIMTOP STPDIS THCORF

MODULE NAME: APLANE
FILE NAME: TOPEXC.FOR
PROCESS: DSPFST
CALLED BY: TOPEXC
CALLING SEQUENCE: CALL APLANE

PURPOSE:

To determine the coefficients of lift and drag associated with a given flap setting.

DESCRIPTION:

APLANE is called during the first pass initialization phase of TOPEXC, provided that there were no pretakeoff computation errors. APLANE is responsible for figuring the appropriate coefficients of lift and drag for a particular flap setting. There are only three valid flap settings for a takeoff run: one, five, and fifteen. Any other flap setting will cause APLANE to set the software index flag "TOPMS" to four, which signals end of run. To determine the lift and drag coefficients the appropriate stabilizer setting must be identified first. Using the center of gravity position, table look-ups and interpolation are performed to figure the stabilizer setting.

Besides the coefficients, the rate of change of lift and drag due to the front spoilers and ground spoilers is also identified. These will be fixed values based on which of the three settings the flaps are positioned at.

GLOBAL REFERENCES:

VARIABLES

CD* CG CL* DCDFSP* DCDGSP* DCLFSP* DCLGSP* TOFLPS TOPMS*

Section 11.3 TOPMS OBJECT LIBRARY (TOPMS.OLB)

Their are seventeen subroutines on the TOPMS utility library TOPMS.OLB. Three of them are subroutines used only by other utility library modules (STPREF FNSERV SEARCH). The following pages give a brief summary of the modules on this library.

MODULE NAME: ADBW2
FILE NAME: PROCS.FOR
CALLING SEQUENCE: CALL ADBW2(STATE_VECTOR)

PURPOSE:
To perform state vector integration.

DESCRIPTION:
This utility performs integration to produce the current value of a simulated signal. A three element input array is passed as the sole calling parameter. The array elements are used as follows.

STATE_VECTOR(1): This element contains the last value of the signal. The new value computed from integration is returned in this element.

STATE_VECTOR(2): This element contains the current rate that the signal changes in one second.

STATE_VECTOR(3): This element contains the previous frames value for STATE_VECTOR(2).

The integration performed by this procedure is done 20 times per second of simulated time. The equation used to compute the new signal is given below.

SIGNAL = LAST_SIGNAL + .05 * (1.5 * RATE - 0.5 LAST_RATE)

Note that ADBW2 automatically places the current rate into the last rate element before returning to the caller.

GLOBAL REFERENCES:
none

MODULE NAME: ASPDCO
FILE NAME: PROCS.FOR
CALLING SEQUENCE: CALL ASPDCO(SPEED, TYPE)

PURPOSE:
To convert between airspeed types.

DESCRIPTION:
This procedure is passed an airspeed (knots) and a conversion type code. A type code of "0" requests conversion from true airspeed to calibrated airspeed and a type of "1" is passed for the conversion back.

GLOBAL REFERENCES:

VARIABLES
DLTAMB THTAMB

FUNCTIONS AND SUBROUTINES
MTH$SIGN MTH$SQRT

MODULE NAME: DZONE
FILE NAME: PROCS.FOR
CALLING SEQUENCE: VALUE = DZONE(SIG,LOW,HIGH)

PURPOSE:
To perform a dead-zone adjustment.

DESCRIPTION:
This utility function is called to force a dead-zone about zero on an input signal. The first calling parameter is the signal value. The remaining parameters are the dead-zone boundaries. The resulting value is the amount the input signal is outside of the dead-zone. If the input is within the dead-zone a zero value is returned.

GLOBAL REFERENCES:
none

MODULE NAME: EPRF
FILE NAME: PROCS.FOR
CALLING SEQUENCE: EPR = EPRF(THROTTLE, TEMPERATURE)

PURPOSE:
To compute EPR values.

DESCRIPTION:
This utility function is called to compute an estimated engine pressure ratio (EPR) from the throttle setting and the engine inlet stagnation temperature.

GLOBAL REFERENCES:
none

MODULE NAME: FINTER
FILE NAME: PROCS.FOR
CALLING SEQUENCE: CALL FINTER(EPR, OAT, THROTTLE)

PURPOSE:
To compute throttle setting for desired EPR.

DESCRIPTION:
This utility is passed a desired engine pressure ratio (EPR) and the outside air temperature (Deg C) to compute the throttle setting required. An iterative algorithm using the function EPRF is implemented.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
EPRF LIB$SIGNAL

MODULE NAME: FNSERV
FILE NAME: STPDIS.FOR
CALLING SEQUENCE: POS = FNSERV(RATE,COMMAND,ACTUAL,LIMIT)

PURPOSE:
To model servo response for control inputs.

DESCRIPTION:
This subroutine models servo response to control commands. The first calling parameter is the exponential servo response constant used in filtering equations. The commanded value is passed along with the current actual position. The last parameter is a rate limit which is the maximum amount the new position may differ from the last.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
MTH$SIGN

MODULE NAME: OLIMIT
FILE NAME: PROCS.FOR
CALLING SEQUENCE: VALUE = OLIMIT(VALUE,LOW,HIGH)

PURPOSE:
To perform value limiting.

DESCRIPTION:
This utility function limits an input value to the supplied boundary values. The resulting number will always be within the range LOW - HIGH, not including the boundary values. A tolerance of 1.0E-9 from the supplied range is always enforced.

GLOBAL REFERENCES:
none

MODULE NAME: ONED
FILE NAME: PROCS.FOR
CALLING SEQUENCE: VALUE = ONED(X, X_TABLE, INT_TABLE)

PURPOSE:
To perform one dimensional interpolation.

DESCRIPTION:
This utility function is passed a sample value "X" and a table of possible values for the sampled value "X_TABLE". The interval containing X in X_table is located and used to compute the interpolation value for the parallel interval in the interpolation table "INT_TABLE".

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
SEARCH

MODULE NAME: RATE
FILE NAME: PROCS.FOR
CALLING SEQUENCE: VALUE = RATE(NEW,OLD,HIGH,LOW)

PURPOSE:

To perform rate limiting.

DESCRIPTION:

This utility procedure returns the NEW value from the parameter list unless the change from the previous value exceeds the rate limits supplied. Note that the rate limits are in UNITS/SECOND, but the utility assumes the time differential between the sampled OLD and NEW is .05 seconds. Note that the global time variable "T" from the pretakeoff simulation is checked for initialization time (0.0). When the simulation time has not started the NEW values is always returned without rate limiting.

GLOBAL REFERENCES:

VARIABLES
T
FUNCTIONS AND SUBROUTINES
XLIM

MODULE NAME: RWYPRD
FILE NAME: PROCS.FOR
CALLING SEQUENCE: DISTANCE = RWYPRD(V_START,V_FINAL,V_WIND)

PURPOSE:
To compute runway distances covered.

DESCRIPTION:
This utility function returns the predicted runway distance traveled while accelerating between two airspeeds. Both airspeeds (true/knots) are supplied in the calling parameters along with the "along runway" component of the wind speed (knots).

The total distance traveled is found by a ten iteration summation of distances covered in acceleration steps. The pretakeoff acceleration versus airspeed polynomial is used to predict the acceleration at each step.

GLOBAL REFERENCES:

ARRAYS
COEF

MODULE NAME: SEARCH
FILE NAME: PROCS.FOR
CALLING SEQUENCE: INDEX = SEARCH(VALUE,TABLE)

PURPOSE:
To perform table searching for interpolation modules.

DESCRIPTION:
This utility function is called by the interpolation utilities ONED and TWOD to locate the interval within a table where a supplied value falls. Note that since no table length is passed, the last interval of the table must be large enough to contain any possible search key value.

GLOBAL REFERENCES:
none

MODULE NAME: STPDIS
FILE NAME: STPDIS.FOR
CALLING SEQUENCE: DIST = STPDIS(V_0,MU,WEIGHT,V_WIND,TK_OFF)

PURPOSE:
To compute runway stopping distances.

DESCRIPTION:
This function computes the distance required to stop the aircraft. This is done by iteratively simulating the abort process with a fixed time step (.25 seconds). The inputs are described below.

CALLING PARAMETERS

V_0	current true airspeed (FT/SEC)
MU	rolling coefficient of friction
WEIGHT	aircraft weight (POUNDS)
V_WIND	wind speed (FT/SEC)
TK_OFF	takeoff / abort mode flag

GLOBAL INPUTS

CL	coefficient of lift
CD	coefficient of drag
DCLFSP	change in CL due to forward spoilers
DCDFSP	change in CD due to forward spoilers
DCLGSP	change in CL due to ground spoilers
DCDGSP	change in CD due to ground spoilers
RHO	air density (SLUGS/CU FT)
THRIDL	idle thrust (POUNDS)
DTHROT	current throttle position (DEGREES)

The simulation performed models the forces involved in stopping the aircraft. Performance of the aircraft is modeled for .25 second intervals until the initial speed has been driven to zero. The aircraft deceleration is computed from thrust, lift, drag, weight, and runway friction. Thrust and drag values are computed from the modeling of reaction time in retarding the throttles and deploying the spoilers.

GLOBAL REFERENCES:

VARIABLES
CD CL DCDFSP DCDGSP DCLFSP DCLGSP DTHROT FSPREF GSPREF RHO THRIDL THRST TMREF

FUNCTIONS AND SUBROUTINES
FNSERV STPREF

MODULE NAME: STPREF
FILE NAME: STPDIS.FOR
CALLING SEQUENCE: CALL STPREF(TK_OFF)

PURPOSE:
To model throttle and spoiler usage in stopping.

DESCRIPTION:
This procedure is called solely by the utility STPDIS. It computes the initial positions of throttles, forward spoilers, and ground spoilers. When in takeoff mode, TK_OFF equal TRUE, the spoilers are assumed retracted and the throttles are set to takeoff position. Once a takeoff abort has started however the entire response time involved is reduced because these actions start when the abort is initiated. After a certain amount of time into the abort phase the throttles and spoilers will be considered completely in their abort positions.

The reference values for the positions are returned to STPDIS through the global variables FSPREF, GSPREF, and TMREF.

GLOBAL REFERENCES:

VARIABLES
FSPREF* GSPREF* TMREF*

FUNCTIONS AND SUBROUTINES
FNSERV

MODULE NAME: THCORF
FILE NAME: PROCS.FOR
CALLING SEQUENCE: THRUST = THCORF(MACH, EPR)

PURPOSE:
To calculate engine thrust.

DESCRIPTION:
This utility function is called to calculate sea-level engine thrust from mach number and engine pressure ratio. For current pressure altitude adjustment the returned value must be multiplied by DLTAMB, which is a global variable computed by ATMOS.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
XLIM

MODULE NAME: THSRVO
FILE NAME: PROCS.FOR
CALLING SEQUENCE: CALL THSRVO(COMMAND,POSITION)

PURPOSE:
To compute the aft flight deck throttle servo response.

DESCRIPTION:
This utility procedure is called to compute the aft flight deck throttle servo position. The calling parameters are the desired and current throttle positions respectively. The actual position of the servo is returned through the second parameter. The effect of this module is to lag the response to the command by filtering the differential between commanded and actual positions. The total amount the servo may respond in one frame is also rate limited.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
MTH$SIGN

MODULE NAME: TWOD
FILE NAME: PROCS.FOR
CALLING SEQUENCE: VALUE = TWOD(X,XST,Y,YST,NX,FST)

PURPOSE:
To perform two dimensional interpolation.

DESCRIPTION:
This utility function is used to interpolate table values dependent on two input values. This can be thought of as identifying the proper position between four points of a grid and interpolating into a corresponding value grid.

The first pair of calling parameters is the first dependent variable and its corresponding table of possible values. The next pair are used likewise for the second dependent variable. The two dimensional table of values is passed as the last calling parameter. Note that the number of rows "NX" in the two dimensional value matrix is passed for Fortran's addressing requirements.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
SEARCH

MODULE NAME: XLIM
FILE NAME: PROCS.FOR
CALLING SEQUENCE: VALUE = XLIM(VALUE,LOW,HIGH)

PURPOSE:
To perform range limiting.

DESCRIPTION:
This utility function is called to limit an input value to a range of specified values. The upper and lower boundaries are passed as calling parameters.

GLOBAL REFERENCES:
none

Section 11.4 TOPMS SIMULATION

The TOPMS display format may be driven for the purpose of demonstration by an internal simulation. The global variable ENABLE is manually modified using the VIEW utility to select the simulation. Note that the standard pilot CDU entries into the flight management computer are made as usual.

The high order four bits of ENABLE are used for the TOPMS simulation. Refer to figure 11.3 at the end of section 11 for the definition of the TOPMS ENABLE bits.

The following page contains a module description for the single routine responsible for creating the TOPMS simulation.

MODULE NAME: SIMTOP
FILE NAME: TOPEXC.FOR
PROCESS: DSPFST
CALLED BY: TOPEXC
CALLING SEQUENCE: CALL SIMTOP(PERFLG)

PURPOSE:
To simulate TOPMS variables for demonstration.

DESCRIPTION:
This procedure is called by TOPEXC when the simulation selection bit in the ENABLE word is set. It is called just before TOPMS processing begins to overwrite external input variables which are used by TOPEXC. The TOPMS performance failure flag is passed to SIMTOP to assist in determining engine failure situations.

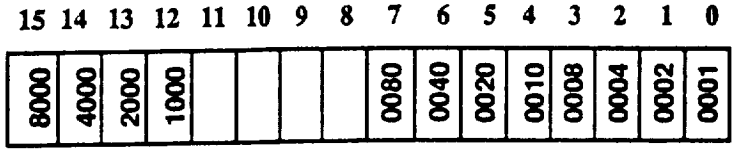
GLOBAL REFERENCES:

VARIABLES
BDXACC BLEEDL* BLEEDR* CAS DTHROT ENABLE EPR1* EPR2* EPRC FLAP* GSFPS* HDGTRU* MAGVAR ORGHDG PITCH* THROTL* THROTR* TOFLPS TOPMS* V1 VR VWK

ARRAYS
COEF

FUNCTIONS AND SUBROUTINES
ASPDCO

"ENABLE" WORD (TOPMS)



BIT(S)	USAGE
0	Set to sample stationary ground speed from IRS to create a ground speed bias term.
1	Set to disable the TOPMS internal wind update.
2	Used to determine the consecutive failure count needed to flag a performance error. When set the count is 8 frames, otherwise 5 frames. Note when the count is set to 8, performance failure checking will also occur after the aircraft reaches rotate speed (VR).
3	Set to remove pitch attitude correction to body mounted acceleration (X-axis) in the computation of along runway acceleration.
4-5	Two-bit value selects overlap distance for "early on" of must-go advisory. 00 → 0 feet 01 → 500 feet 10 → 1000 feet 11 → 1500 feet
6-7	Throttle position override bits. 01 → copy right into left 10 → copy left into right
12-15	TOPMS internal simulation bits. 15: enable simulation 12: freeze simulation 13-14: 00 → standard run 01 → poor performance 10 → engine out at 100kt 11 → engine out at V1

-figure 11.3-

Appendix A I/O BUFFER USAGE

Display data is sent to the microprocessor system in a 704 word contiguous block. Except for the Navigation display format's background buffer (first 400 words), specific locations within the data block have assigned contents. The following pages contain the information on the allocation of the data buffer. The first chart shows a composite usage layout for all six display formats. The ranges of memory used by the various formats is shown by assigning the byte offset values to ID mnemonics (NAV: Navigation Display; PFD: Primary Flight Display; ENG: Engine Display; SYS: System warning display; F1: Sperry Primary Display TOP: TOPMS display). The remaing pages have charts for each individual format, breaking down the I/O memory usage by specific variables. Note that the memory allocations for NAV and TOP overlay each other. This is because these display formats are mutually exclusive. The same microprocessor contains both formats, with only one active at a time.

The offset and size values in the charts are in terms of bytes. The index column contains the index into a word array used by the host computer. When a "+" is shown by the index the data actually resides on an odd byte boundary.

Following the I/O layouts is a detailed description of the various packed words contained in the charts.

MEMORY USAGE FOR ALL FORMATS

OFFSET	SIZE	INDEX	ID
0 - 801	802	1	NAV
802 - 1005	204	402	
1006 - 1007	2	504	ENG
1008 - 1079	72	505	
1080 - 1081	2	541	F1
1082 - 1085	4	542	
1086 - 1089	4	544	F1
1090 - 1091	2	546	PFD F1
1092 - 1097	6	547	F1
1098 - 1099	2	550	
1100 - 1103	4	551	SYS
1104 - 1107	4	553	
1108 - 1169	62	555	ENG
1170 - 1179	10	586	
1180 - 1181	2	591	PFD NAV
1182 - 1187	6	592	NAV
1188 - 1189	2	595	PFD NAV
1190 - 1193	4	596	NAV
1194 - 1195	2	598	PFD NAV
1196 - 1201	6	599	NAV
1202 - 1203	2	602	PFD NAV
1204 - 1207	4	603	NAV
1208 - 1241	34	605	NAV TOP
1242 - 1247	6	622	NAV
1248 - 1255	8	625	
1256 - 1263	8	629	PFD
1264 - 1267	4	633	PFD F1
1268 - 1271	4	635	PFD
1272 - 1275	4	637	PFD NAV
1276 - 1343	68	639	PFD
1344 - 1345	2	673	PFD NAV
1346 - 1347	2	674	
1348 - 1351	4	675	PFD
1352 - 1355	4	677	
1356 - 1357	2	679	NAV SYS
1358 - 1359	2	680	PFD F1
1360 - 1361	2	681	PFD NAV SYS F1
1362 - 1363	2	682	PFD NAV
1364 - 1365	2	683	PFD NAV SYS F1
1366 - 1369	4	684	TOP ENG SYS
1370 - 1373	4	686	
1374 - 1375	2	688	SYS F1
1376 - 1377	2	689	PFD F1
1378 - 1379	2	690	PFD NAV TOP ENG SYS F1
1380 - 1381	2	691	PFD F1

1382 - 1383	2	692	NAV TOP
1384 - 1395	12	693	SYS
1396 - 1401	6	699	ENG SYS
1402 - 1403	2	702	ENG
1404 - 1405	2	703	NAV SYS F1
1406 - 1407	2	704	NAV SYS

MEMORY ALLOCATION FOR PFD

OFFSET	SIZE	INDEX	VARIABLE	UNITS
1090 - 1091	2	546	RADAR_ALT	FEET * 6.5536
1180 - 1181	2	591	TRK_DIAL	DEG * 182.0444
1188 - 1189	2	595	TRU_HDG	DEG * 182.0444
1194 - 1195	2	598	TRU_TRK	DEG * 182.0444
1202 - 1203	2	602	MAG_TRK	DEG * 182.0444
1256 - 1257	2	629	ACC_SEGMENT	INCHES * 1000
1258 - 1259	2	630	STANDOFF	INCHES * 1000
1260 - 1261	2	631	AIRCRAFT_X	INCHES * 1000
1262 - 1263	2	632	AIRCRAFT_Y	INCHES * 1000
1264 - 1265	2	633	ACT_CAS	KNOTS * 64
1266 - 1267	2	634	CAS_REF	KNOTS * 64
1268 - 1269	2	635	DEC_HT	FEET
1270 - 1271	2	636	FPA_DIAL	DEG * 10
1272 - 1273	2	637	VRT_REF	FEET
1274 - 1275	2	638	GS_REF	FEET
1276 - 1277	2	639	HOR_TICK_CTR	INCHES * 1000
1278 - 1279	2	640	HOR_X	INCHES * 1000
1280 - 1281	2	641	MACH	RATIO * 1000
1282 - 1283	2	642	PITCH_Y	INCHES * 1000
1284 - 1285	2	643	LOC_X	INCHES * 1000
1286 - 1287	2	644	RWY_X1	INCHES * 1000
1288 - 1289	2	645	RWY_Y1	INCHES * 1000
1290 - 1291	2	646	RWY_X2	INCHES * 1000
1292 - 1293	2	647	RWY_Y2	INCHES * 1000
1294 - 1295	2	648	RWY_X3	INCHES * 1000
1296 - 1297	2	649	RWY_Y3	INCHES * 1000
1298 - 1299	2	650	RWY_X4	INCHES * 1000
1300 - 1301	2	651	RWY_Y4	INCHES * 1000
1302 - 1303	2	652	RWY_X5	INCHES * 1000
1304 - 1305	2	653	RWY_Y5	INCHES * 1000
1306 - 1307	2	654	RWY_X6	INCHES * 1000
1308 - 1309	2	655	RWY_Y6	INCHES * 1000
1310 - 1311	2	656	RWY_X7	INCHES * 1000
1312 - 1313	2	657	RWY_Y7	INCHES * 1000
1314 - 1315	2	658	RWY_X8	INCHES * 1000
1316 - 1317	2	659	RWY_Y8	INCHES * 1000
1318 - 1319	2	660	RWY_SCALE	(0-1) * 16384
1320 - 1321	2	661	PTH_TRK	DEG * 182.0444
1322 - 1323	2	662	FLARE	INCHES * 1000
1324 - 1325	2	663	STAR_X	INCHES * 1000
1326 - 1327	2	664	STAR_Y	INCHES * 1000
1328 - 1329	2	665	STAR_ZOOM	(1-20) * 1024
1330 - 1331	2	666	FPA_REF	INCHES * 1000
1332 - 1333	2	667	TRACK_BUG_X	INCHES * 1000
1334 - 1339	6	668	TO_WPT	ASCIC

1340 - 1341	2	671	NIBBLES_1	PACKED WORD
1342 - 1343	2	672	NIBBLES_2	PACKED WORD
1344 - 1345	2	673	NIBBLES_3	PACKED WORD
1348 - 1349	2	675	DISCRETES	PACKED WORD
1350 - 1351	2	676	SKY_PTR	DEG * 182.0444
1358 - 1359	2	680	ACT_ROLL	DEG * 182.0444
1360 - 1361	2	681	ACT_ALT	FEET
1362 - 1363	2	682	HDOT	INCHES * 1000
1364 - 1365	2	683	ALT_REF	FEET
1376 - 1377	2	689	FLAP_UP	KNOTS * 65.536
1378 - 1379	2	690	HOST_CNT	SECONDS * 20
1380 - 1381	2	691	BARO_SET	INCHES * 100

MEMORY ALLOCATION FOR NAV

OFFSET	SIZE	INDEX	VARIABLE	UNITS
0 - 799	800	1	BCKGND_BUF	BACKGROUND DATA
800 - 801	2	401	LINK_CMD	reserved
1180 - 1181	2	591	SEL_TRK	DEG * 182.0444
1182 - 1183	2	592	ALT_RANGE	NAUT. MILES * 128
1184 - 1185	2	593	XTK_GS_RATIO	(1/SEC) * 65536
1186 - 1187	2	594	AC_NORM	(FT/SEC/SEC) * 512
1188 - 1189	2	595	TRU_HDG	DEG * 182.0444
1190 - 1191	2	596	WIND_SPD	KNOTS
1192 - 1193	2	597	WIND_DIR	DEG * 182.0444
1194 - 1195	2	598	TRU_TRK	DEG * 182.0444
1196 - 1197	2	599	MAP_WORD	PACKED WORD
1198 - 1199	2	600	MLS_X	NAUT. MILES * 2048
1200 - 1201	2	601	MLS_Y	NAUT. MILES * 2048
1202 - 1203	2	602	MAG_TRK	DEG * 182.0444
1204 - 1205	2	603	BOX_X	FEET / 32
1206 - 1207	2	604	BOX_Y	FEET / 32
1208 - 1209	2	605	BOX_HDG	DEG * 182.0444
1210 - 1227	18	606	BUB_COORDS	3-(X,Y,HDG)
1228 - 1229	2	615	TANG_TIMEI	SEC * 256
1230 - 1231	2	616	OFF_VECTOR	NAUT. MILES * 128
1232 - 1239	8	617	AP_EAST	4-(INCHES * 1000)
1240 - 1247	8	621	AP_NORTH	4-(INCHES * 1000)
1272 - 1273	2	637	VRT_REF	FEET
1274 - 1275	2	638	GS_REF	FEET
1344 - 1345	2	673	NIBBLE3	PACKED WORD
1356 - 1357	2	679	GS_FPS	(FT/SEC) * 32
1360 - 1361	2	681	ALT_ACTUAL	FEET
1362 - 1363	2	682	HDOT_SEGMENT	INCHES * 1000
1364 - 1365	2	683	REF_ALT_VALUE	FEET
1378 - 1379	2	690	HOST_CNT	SECONDS * 20
1382 - 1383	2	692	FMT_ID	2, 7
1404 - 1405	2	703	DISC_WORD11	PACKED WORD
1406 - 1407	2	704	DISC_WORD12	PACKED WORD

MEMORY ALLOCATION FOR TOP

OFFSET	SIZE	INDEX	VARIABLE	UNITS
1208 - 1209	2	605	SPEED	KNOTS
1210 - 1211	2	606	WIND_SPD	KNOTS
1212 - 1213	2	607	WIND_DIR	DEG * 182.0444
1214 - 1215	2	608	RWY_LEN	FEET
1216 - 1217	2	609	PLANE	FEET
1218 - 1219	2	610	EPR	RATIO * 8192
1220 - 1221	2	611	V2	KNOTS
1222 - 1223	2	612	ROLL_LIMIT	FEET
1224 - 1225	2	613	VR1	FEET
1226 - 1227	2	614	VR2_POS	FEET
1228 - 1229	2	615	VR2_VAL	KNOTS
1230 - 1231	2	616	DSPEED_POS	FEET
1232 - 1233	2	617	DSPEED_VAL	KNOTS
1234 - 1235	2	618	STAR	FEET
1236 - 1237	2	619	PACK_WORD	PACKED WORD
1238 - 1239	2	620	EYE	FEET
1240 - 1241	2	621	RWY_ID	ASCII
1366 - 1367	2	684	EPR1	RATIO * 8192
1368 - 1369	2	685	EPR2	RATIO * 8192
1378 - 1379	2	690	HOST_CNT	SECONDS * 20
1382 - 1383	2	692	FMT_ID	2, 7

MEMORY ALLOCATION FOR ENG

OFFSET	SIZE	INDEX	VARIABLE	UNITS
1006 - 1007	2	504	SWITCH	reserved
1108 - 1109	2	555	ENG_EPRLIM	RATIO * 8192
1110 - 1111	2	556	ENG_EPRCAU	RATIO * 8192
1112 - 1113	2	557	ENG_EPRCOM	RATIO * 8192
1114 - 1115	2	558	ENG_N1_L	% * 163.84
1116 - 1117	2	559	ENG_N1_R	% * 163.84
1118 - 1119	2	560	ENG_N1LIM	% * 163.84
1120 - 1121	2	561	ENG_N1CAU	% * 163.84
1122 - 1123	2	562	ENG_EGT_L	DEG-C * 32.768
1124 - 1125	2	563	ENG_EGT_R	DEG-C * 32.768
1126 - 1127	2	564	ENG_EGTLIM	DEG-C * 32.768
1128 - 1129	2	565	ENG_EGTCAU	DEG-C * 32.768
1130 - 1131	2	566	ENG_FF_L	(LB/HR) * 1.6384
1132 - 1133	2	567	ENG_FF_R	(LB/HR) * 1.6384
1134 - 1135	2	568	ENG_EOP_L	PSI * 163.84
1136 - 1137	2	569	ENG_EOP_R	PSI * 163.84
1138 - 1139	2	570	ENG_EOLLIM	PSI * 163.84
1140 - 1141	2	571	ENG_EOULIM	PSI * 163.84
1142 - 1143	2	572	ENG_EOT_L	DEG-C * 91.022
1144 - 1145	2	573	ENG_EOT_R	DEG-C * 91.022
1146 - 1147	2	574	ENG_EOTLIM	DEG-C * 91.022
1148 - 1149	2	575	ENG_EOQ_L	GALLONS * 3276.8
1150 - 1151	2	576	ENG_EOQ_R	GALLONS * 3276.8
1152 - 1153	2	577	ENG_EOQLIM	GALLONS * 3276.8
1154 - 1155	2	578	ENG_N2_L	% * 163.84
1156 - 1157	2	579	ENG_N2_R	% * 163.84
1158 - 1159	2	580	ENG_N2LIM	% * 163.84
1160 - 1161	2	581	ENG_LFUELQ	POUNDS
1162 - 1163	2	582	ENG_RFUELQ	POUNDS
1164 - 1165	2	583	ENG_CFUELQ	POUNDS
1166 - 1167	2	584	ENG_TAT	DEG-C * 128
1168 - 1169	2	585	ENG_GW	POUNDS * .125
1366 - 1367	2	684	ENG_EPR_L	RATIO * 8192
1368 - 1369	2	685	ENG_EPR_R	RATIO * 8192
1378 - 1379	2	690	HOST_CNT	SECONDS * 20
1396 - 1397	2	699	DISC_WORD7	PACKED WORD
1398 - 1399	2	700	DISC_WORD8	PACKED WORD
1400 - 1401	2	701	DISC_WORD9	PACKED WORD
1402 - 1403	2	702	DISC_WORD10	PACKED WORD

MEMORY ALLOCATION FOR SYS

OFFSET	SIZE	INDEX	VARIABLE	UNITS
1100 - 1101	2	551	SYS_LFLAP	DEG * 182.0444
1102 - 1103	2	552	SYS_RFLAP	DEG * 182.0444
1356 - 1357	2	679	GS_FPS	(FT/SEC) * 32
1360 - 1361	2	681	PFD_ALT	FEET
1364 - 1365	2	683	PFD_SELALT	FEET
1366 - 1367	2	684	ENG_EPR_L	RATIO * 8192
1368 - 1369	2	685	ENG_EPR_R	RATIO * 8192
1374 - 1375	2	688	PFD_VSPD	(FT/SEC) * 49.152
1378 - 1379	2	690	HOST_CNT	SECONDS * 20
1384 - 1385	2	693	DISC_WORD1	PACKED WORD
1386 - 1387	2	694	DISC_WORD2	PACKED WORD
1388 - 1389	2	695	DISC_WORD3	PACKED WORD
1390 - 1391	2	696	DISC_WORD4	PACKED WORD
1392 - 1393	2	697	DISC_WORD5	PACKED WORD
1394 - 1395	2	698	DISC_WORD6	PACKED WORD
1396 - 1397	2	699	DISC_WORD7	PACKED WORD
1398 - 1399	2	700	DISC_WORD8	PACKED WORD
1400 - 1401	2	701	DISC_WORD9	PACKED WORD
1404 - 1405	2	703	DISC_WORD11	PACKED WORD
1406 - 1407	2	704	DISC_WORD12	PACKED WORD

MEMORY ALLOCATION FOR F1

OFFSET	SIZE	INDEX	VARIABLE	UNITS
1080 - 1081	2	541	PFD_PITCH	DEG * 182.0444
1086 - 1087	2	544	PFD_VDVDAT	FEET * 9.6
1088 - 1089	2	545	PFD_XLDDAT	FEET * 9.6
1090 - 1091	2	546	RADDAT	FEET * 6.5536
1092 - 1093	2	547	PFD_SPDDAT	knots * 500
1094 - 1095	2	548	PITCMDCP	DEG * 182.0444
1096 - 1097	2	549	ROLCMDCP	DEG * 182.0444
1264 - 1265	2	633	IASDAT	KNOTS * 64
1266 - 1267	2	634	ASTDAT	KNOTS * 64
1358 - 1359	2	680	PFD_ROLL	DEG * 182.0444
1360 - 1361	2	681	ALTITUDE	FEET
1364 - 1365	2	683	PFD_ASELDAT	FEET
1374 - 1375	2	688	VSPDAT	(FT/SEC) * 49.152
1376 - 1377	2	689	VMODAT	KNOTS * 65.536
1378 - 1379	2	690	HOST_CNT	SECONDS * 20
1380 - 1381	2	691	BAROSET	INCHES * 100
1404 - 1405	2	703	DISC_WORD11	PACKED WORD

PACKED WORD DESCRIPTIONS

INDEX	BIT(S)	USAGE
599	0-15	map background contol word (Section 7.1)
619	0	TOPMS format valid
	1	left engine o.k.
	2	right engine o.k.
	3-4	takeoff advisory status
	5	flap set error
	6	left or right bleeds off
671	0-3	control mode index
	4-7	throttle mode index
	8-11	roll controller detent status
	12-15	pitch controller detent status
672	0-3	perspective runway approach index
	4-7	reference CAS color code
	8-11	engaged horizontal mode index
	12-15	armed horizontal mode index
673	0-3	engaed vertical mode index
	4-7	armed vertical mode index
	8-11	flight path angle bar color code
	12-15	alert message index
675		
	0	attitude valid
	1	velocity vector mode enabled
	2	altitude hold sub-mode valid
	3	star symbol valid
	4	track bug symbol valid
	5	runway symbol valid
	6	reference altitude valid
	7	radar altitude valid
	8	airspeed valid
	9	altitude valid
	10	PFD format valid
	11	track select valid
693	0	rfdiu disengaged
	1-4	reserved
	5	RFD Nav #2 tuning
	6	DME #2 tuning
	7	RFD com tuning 1 & 2
694	0-6	reserved
	7	stabilizer out of trim
	8-15	reserved
695	0-9	reserved
	10	speed brake-do not arm
	11-15	reserved

696	0	flap placard
	1	throttle placard
	2	speed brake placard
	3	aileron cam out
	4	elevator cam out
	5	rudder cam out
	6-13	reserved
	14	anti-skid inoperative
	15	reserved
697	0-1	reserved
698	0	nose gear up
	1	nose gear down
	2	nose gear locked
	3	nose gear up to down
	4	nose gear down to up
	5	left gear up
	6	left gear down
	7	left gear locked
	8	left gear up to down
	9	left gear down to up
	10	right gear up
	11	right gear down
	12	right gear locked
	13	right gear up to down
	14	right gear down to up
699	0	left flap moving
	1	right flap moving
	2-3	reserved
	4	speed brake armed
	5-9	reserved
	10	left reverser armed
	11	right reverser armed
700	0	left EPR valid
	1	left N1 valid
	2	left EGT valid
	3	left fuel flow valid
	4	left oil pressure valid
	5	left oil temperature valid
	6	left oil quantity valid
	7	left N2 valid
701	0	right EPR valid
	1	right N1 valid
	2	right EGT valid
	3	right fuel flow valid
	4	right oil pressure valid
	5	right oil temperature valid
	6	right oil quantity valid
	7	right N2 valid

702	0	EPR limit valid
	1	EPR caution valid
	2	EPR command valid
	3	N1 limit valid
	4	N1 caution valid
	5	EGT limit valid
	6	EGT caution valid
	7	oil pressure limit valid
	8	oil temperature limit valid
	9	oil quantity limit valid
	10	N2 limit valid
	11	left fuel quantity valid
	12	right fuel quantity valid
	13	center fuel quantity valid
	14	total air temperature valid
	15	gross weight valid
703	0	attitude valid
	1	airspeed valid
	2	selected airspeed valid
	3	airspeed limit valid
	4	altitude valid
	5	selected altitude valid
	6	vertical speed valid
	8	vertical deviation valid
	9	lateral deviation valid
	10	radar altitude valid
	11-12	reserved
	13	true heading valid
	14	true track valid
	15	selected track valid
704	0	track line valid
	1	MLS mode engaged
	2	MLS mode valid
	3	GPS mode valid
	6	wind valid
	7	left flap valid
	8	right flap valid
	9	MLS airplane color variant
	13	ground speed valid
	14	position valid
	15	magnetic track valid

INDAT(N)	DESCRIPTION
1	DP11 FORMAT ID/STATUS
2	DP11 LEFT POT VALUE
3	DP11 RIGHT POT VALUE
4	DP11 BEZEL DISCRETE WORD
5:0-7	DP11 DP ID FOUND BY FORMAT FROM DISCRETES
5:8-15	DP11 DEU ID FOUND BY FORMAT FROM DISCRETES
6	DP11 FORMAT CHECKSUM
7	DP11 FORMAT SPARE TIME APPROXIMATION (msec)
8	DP11 FORMAT TIME FRAME OVERFLOW COUNT
9	DP11 FORMAT TO DP4 I/O INTERFACE MISSES
10 - 32	
33	DP12 FORMAT ID/STATUS
34	DP12 LEFT POT VALUE
35	DP12 RIGHT POT VALUE
36	DP12 BEZEL DISCRETE WORD
37:0-7	DP12 DP ID FOUND BY FORMAT FROM DISCRETES
37:8-15	DP12 DEU ID FOUND BY FORMAT FROM DISCRETES
38	DP12 FORMAT CHECKSUM
39	DP12 FORMAT SPARE TIME APPROXIMATION (msec)
40	DP12 FORMAT TIME FRAME OVERFLOW COUNT
41	DP12 FORMAT TO DP4 I/O INTERFACE MISSES
42 - 64	
65	DP13 FORMAT ID/STATUS
66	DP13 LEFT POT VALUE
67	DP13 RIGHT POT VALUE
68	DP13 BEZEL DISCRETE WORD
69:0-7	DP13 DP ID FOUND BY FORMAT FROM DISCRETES
69:8-15	DP13 DEU ID FOUND BY FORMAT FROM DISCRETES
70	DP13 FORMAT CHECKSUM
71	DP13 FORMAT SPARE TIME APPROXIMATION (msec)
72	DP13 FORMAT TIME FRAME OVERFLOW COUNT
73	DP13 FORMAT TO DP4 I/O INTERFACE MISSES
74 - 96	
97	DP21 FORMAT ID/STATUS
98	DP21 LEFT POT VALUE
99	DP21 RIGHT POT VALUE
100	DP21 BEZEL DISCRETE WORD
101:0-7	DP21 DP ID FOUND BY FORMAT FROM DISCRETES
101:8-15	DP21 DEU ID FOUND BY FORMAT FROM DISCRETES
102	DP21 FORMAT CHECKSUM
103	DP21 FORMAT SPARE TIME APPROXIMATION (msec)
104	DP21 FORMAT TIME FRAME OVERFLOW COUNT
105	DP21 FORMAT TO DP4 I/O INTERFACE MISSES

106 - 128	
129	DP22 FORMAT ID/STATUS
130	DP22 LEFT POT VALUE
131	DP22 RIGHT POT VALUE
132	DP22 BEZEL DISCRETE WORD
133:0-7	DP22 DP ID FOUND BY FORMAT FROM DISCRETES
133:8-15	DP22 DEU ID FOUND BY FORMAT FROM DISCRETES
134	DP22 FORMAT CHECKSUM
135	DP22 FORMAT SPARE TIME APPROXIMATION (msec)
136	DP22 FORMAT TIME FRAME OVERFLOW COUNT
137	DP22 FORMAT TO DP4 I/O INTERFACE MISSES
138 - 192	
193	DP31 FORMAT ID/STATUS
194	DP31 LEFT POT VALUE
195	DP31 RIGHT POT VALUE
196	DP31 BEZEL DISCRETE WORD
197:0-7	DP31 DP ID FOUND BY FORMAT FROM DISCRETES
197:8-15	DP31 DEU ID FOUND BY FORMAT FROM DISCRETES
198	DP31 FORMAT CHECKSUM
199	DP31 FORMAT SPARE TIME APPROXIMATION (msec)
200	DP31 FORMAT TIME FRAME OVERFLOW COUNT
201	DP31 FORMAT TO DP4 I/O INTERFACE MISSES
202 - 224	
225	DP32 FORMAT ID/STATUS
226	DP32 LEFT POT VALUE
227	DP32 RIGHT POT VALUE
228	DP32 BEZEL DISCRETE WORD
229:0-7	DP32 DP ID FOUND BY FORMAT FROM DISCRETES
229:8-15	DP32 DEU ID FOUND BY FORMAT FROM DISCRETES
230	DP32 FORMAT CHECKSUM
231	DP32 FORMAT SPARE TIME APPROXIMATION (msec)
232	DP32 FORMAT TIME FRAME OVERFLOW COUNT
233	DP32 FORMAT TO DP4 I/O INTERFACE MISSES
234 - 256	
257	DP33 FORMAT ID/STATUS
258	DP33 LEFT POT VALUE
259	DP33 RIGHT POT VALUE
260	DP33 BEZEL DISCRETE WORD
261:0-7	DP33 DP ID FOUND BY FORMAT FROM DISCRETES
261:8-15	DP33 DEU ID FOUND BY FORMAT FROM DISCRETES
262	DP33 FORMAT CHECKSUM
263	DP33 FORMAT SPARE TIME APPROXIMATION (msec)
264	DP33 FORMAT TIME FRAME OVERFLOW COUNT
265	DP33 FORMAT TO DP4 I/O INTERFACE MISSES
266 - 288	
289	BIU REAL TIME INTERRUPT COUNT

290	COUNT OF PASSES THROUGH BIU MAIN LOOP
291	BIU HIGH-SPEED BUS TRANSMITTER TIMEOUT COUNTER
292	COUNTER OF DEU RESPONSE FAILURES
293	BIU HSB RECEPTIONS WITH WRONG NUMBER OF BYTES
294	HSB CRC ERROR COUNTER
295	DISCRETE INDICATING BIU ROUTINES EXECUTED
296	NUMBER OF BIU REINITIALIZATIONS AFTER POWER ON
297	NUMBER OF BIU HARDWARE RESETS SINCE POWER ON
298	NUMBER OF UNEXPECTED INTERRUPTS IN THE BIU
299	HSB SECONDARY ADDRESS (DESTINATION OF FRAME)
300	HSB PRIMARY ADDRESS (FILTERS INCOMING FRAMES)
301	BIU HARDWARE INPUT DISCRETE
302	NORDEN TO BIU INPUT COUNTER
303	BIU TO NORDEN OUTPUT COUNTER
304	NUMBER OF TIMEOUTS DURING INPUT FROM DR11 TO BIU
305	NUMBER OF TIMEOUTS DURING OUTPUT FROM BIU TO DR11
306	CONTENTS OF DR11 ADDRESS REGISTER
307	BIU TIMEOUT COUNTER (CLEARED EACH FRAME)
308	NO DR11 ACTIVITY COUNTER
309 - 311	DP14 GENERAL PURPOSE STATUS WORDS
312 - 314	DP24 GENERAL PURPOSE STATUS WORDS
315 - 317	DP34 GENERAL PURPOSE STATUS WORDS
318	UNEXPECTED DR11 WRITE COMMAND COUNTER
319	UNEXPECTED DR11 READ COMMAND COUNTER
320	

Appendix B VIEW COMMAND ENTRIES

When entering commands to VIEW the entered text is shown below the last display line. The prompt "->" is shown while VIEW is accepting input. While the VIEW prompt is displayed the values of the variables on the display lines are not updated. This "freezes" the state of all displayed variables at the time input was started. To perform a value "freeze" when no actual entries need to be made, enter a blank space to get into update hold. A carriage return will send a null command to VIEW which will return to standard update mode.

The following pages contain a description of the commands available for VIEW users. When the complete format of a command is given, optional parts are delimited by square brackets. The last page of this section contains examples of VIEW commands with a brief description of the actions performed.

** Displaying Variables **

Flight software global variables may be placed on the VIEW display screen by entering their name followed by one or more options. The general format of this command is shown below.

<name> [([+]n)] [/L=n[/I]] [/F=a[.n]] [/R=n] [/D=a] [/S=n]

The various options are used to override default actions from VIEW.

SUBSCRIPT / OFFSET

A numeric value may be entered, enclosed in parentheses, immediately following the symbol name. VIEW interprets the number in one of two ways. If the entered value is an unsigned constant then the value is treated as an array index. VIEW uses the index to determine which of several consecutive data items should be displayed. When the value is preceeded by a "+" or "-" sign the value will be used as a direct byte offset from the address associated with the variable's name. When no subscript is supplied the base address of the entered variable is used. Note that entering "(1)" or "(+0)" after a variable shows the identical memory location as is seen when the variable's name is entered by itself.

/L
This switch is used to select the line on the display screen (1-20) where the variable will be placed. The default is the first line after the last used line. When the /L option is used any variable already show on the chosen line will be erased. If the new variable is to be inserted at the line the /I switch must be used in conjunction with /L. When the /I is used the variables on the rest of the display page are moved down to make room for the new entry. Note that variables at the bottom of the display page will be pushed of the end of the page.

/F
This switch is used to override the default format stored in the VIEW symbol table. There are two parts to this switch; The format type and the format length. The format length indicates how many bytes of memory belong to the variable and the format type defines how the data at that location is interpreted. The following table shows the five format types and their valid data lengths.

FORMAT	LENGTH	DESCRIPTION
F	4,8	floating point format
E	4,8	exponential floating point
I	1,2,4	signed decimal fixed point
H	1,2,4	unsigned hexadecimal fixed point
A	1 - 8	ASCII text

Note the byte length defaults to 4 when not supplied.

/R
The repeat count is used to display a group of consecutive memory locations each having the same data format. The default repeat count is one, which shows the symbolic address location only.

/D
This switch overrides the default description label placed alongside an entered variable. VIEW uses the entered variable name as the label by default. Any ASCII text string may be used, up to 14 characters long.

/S

This switch defines the number of lines to be used for an update sequence of the selected variable. An update sequence shows the last "n" sampled values of the chosen variable. For each update cycle of VIEW only one line in the update sequence is changed to reflect the most recent sample of the variable. When the line is updated a two digit hexadecimal sequence number is appended to the end of line. The sequence number is used to denote which line within the update sequence was updated last. On the next update cycle the next line within the sequence is changed. Past values of the variable remain on the screen on the other lines of the update sequence. Note that only one update sequence may be in effect at a time.

** Modifying Variables **

A value may be stored into a variable which is shown on the VIEW display line by specifying the line number and the desired value. The entered value must be appropriate for the format used to display the variable. The format of the command is as follows.

#<line>=<value> [/R=n]

Note that variables on several consecutive display lines may be modified by using the /R switch to supply the count.

** Deleting Display Lines **

This command is used to remove a variable from the VIEW display. The format of the command is as follows.

-<line> [/C] [/R=n]

The /R switch is used when several consecutive lines must be removed. If it is desired that the variables following the deleted lines should be moved up to fill the vacated space the compress switch (/C) is entered. To remove all the variables from the display use "-*".

** Changing Pages **

VIEW has four display pages consisting of twenty lines each. Simply enter the desired page number, no <CR> necessary, to get to the desired one. Note that entering the current page number is a convenient way to erase unwanted output showing on the CRT screen.

** Creating Command Sets **

A sequence of VIEW commands may be saved on a file for use at a later time. To enable command logging enter

\<file>

where <file> is the name of the disk file where the VIEW commands are to be stored. To disable logging the "\" is entered again with no file name appended. Note the standard VIEW prompt "->" is changed to ">>" when command logging is enabled. Erroneous VIEW entries, which cause the display of an error diagnostic, will not be added to the command log.

There is one VIEW command which is valid only while logging is enabled. The "." command places a pause into the command set file. Later when the command set is executed the stream of VIEW commands will be interrupted at the point where the "." was entered. Two options exist for continuing from a pause during command set execution. An <esc> entry terminates the command set, returning VIEW to standard update mode. Any other key stroke will cause VIEW to continue on with the remainder of the command set.

Built in command sets can be created from a command log file by using the program VIEW_SET. To use the program enter

RUN UTL:VIEW_SET

on the software development VAX. The SETUP.MAR file linked with VIEW can be modified by this menu driven program. After exiting VIEW_SET the SETUP.MAR file must be assembled and a new VIEW.EXE must be created using the linker.

** Predefined Command Sets **

An entire set of VIEW commands may be executed by using a predefined command set. The format of the command is as follows.

@n or @<file>

When the @n form is used one of the built-in command sets is executed. To get a directory of all the built-in command sets enter @0. To execute a built in command set type the "@" command followed by the number of the desired command set. A command set that exists on a file is executed by following the "@" command by the name of the file containing the set of VIEW commands. In either case the commands are executed as if they were entered manually in the order saved in the command set.

** Symbolic Name Directory **

The names of global variables which VIEW has stored in its symbol table may be displayed on the CRT screen with this command. The format of the command is as follows.

?<pattern>

All variables that match the entered pattern are shown, in alphabetic order. The wildcard characters "*" and "%" may be used in the pattern. The "*" means any characters may fit in the entered position, including none at all. The "%" symbol can represent exactly one character position.

** Exiting View **

Enter ^Z to stop the program and save the state of the display pages. "QUIT" exits VIEW freeing all display lines. Since a ^Z exit reserves a block of VAX memory for storage of page configuration, the QUIT should be used at the end of a session.

** Obtaining Help **

Enter "HELP" to produce a page of command reference text. Any key stroke will return the display to the standard VIEW page.

VIEW command examples

COMMAND	RESULT
PITCH	Places the variable PITCH on the next available display line using the format stored in the VIEW symbol table.
LIST(4)/L=6/R=3/F=H.2	Places three elements of the array LIST, starting with the fourth element, onto display lines 6 - 8. The data format is two byte hexadecimal representation.
#4=17.51	Changes the global memory associated with the variable on line #4 to 17.51. The format used on the display would need to be either "F" or "E".
-15	Remove from the display the variable shown on line #15.
-1/R=3/C	Remove the variables on lines 1 - 3, moving the rest of the displayed variables up to fill the empty lines.
@4	Execute the fourth built in command set.
@[-]COMMANDS.LOG	Execute the command set stored in the specified file.
\[-]COMMANDS.LOG	Log VIEW commands on the specified file.
\	Terminate command logging.
?S*1	Display all global variable names which start with "S" and end with "1".
?R%%%	Display all four letter global variable names which start with "R".

Appendix C CREATING THE EXECUTABLE IMAGES

The following files are provided with the source code files in a delivery set. These files are used in the generation of the display executable images.

BUILD.COM	Builds all images using following ".COM" files
DDSTAR.COM	Linker commands for building DDSTAR.EXE
DSPFST.COM	Linker commands for building DSPFST.EXE
DSPHDL.COM	Linker commands for building DSPHDL.EXE
DSPSLW.COM	Linker commands for building DSPSLW.EXE
SECTION.COM	Linker commands for building SECTION.EXE
VIEW.COM	Linker commands for building VIEW.EXE
DSPFST.OPT	Linker options for DSPFST.EXE
DSPHDL.OPT	Linker options for DSPHDL.EXE
DSPSLW.OPT	Linker options for DSPSLW.EXE
OPT.OPT	Linker options for all executable images
SECTION.OPT	Linker options for SECTION.EXE and VIEW.EXE
MAPTBL.MAR	Global section mapping table for DSPFST, DSPHDL, DSPSLW
PASS.MAR	Global section mapping associated with VIEW password entries.
COMMON.FOR	Fortran "Block Data" module for BLKMAC
GBLNAME.DAT	Contains names of all images and global sections for use by the utilities GLOBAL and SECTION.

Several command procedures and utility programs exist for maintenance of the display software. Users must have the following commands in their LOGIN.COM file.

```
DEFINE UTL DUB0:[CSC.CJS.CMS]/JOB
@UTL:SET_UP
```

The executable programs have been defined as DCL commands, therefore they are accessed by simply entering their names (CMS_SYSTEM, GLOBAL, BLKMAC). The command files are activated by prefixing "@UTL:" to the file name.

CMS_SYSTEM.EXE	Accesses source file delivery sets
GLOBAL.EXE	Interactive program for global section linkage
BLKMAC.EXE	Creates object modules for global data
MACALL.COM	Assembles macro source files
FORALL.COM	Compiles Fortran source files
FTN.COM	Compiles individual Fortran source file.

The first step in creating the executable images is the generation of VMS object modules from the source code files described in this document. The VMS Fortran compiler and Macro assembler are used to create object modules for ".FOR" and ".MAR" files respectively. Object files for the global data modules, ".INC" files with Fortran COMMON definitions, are generated with the utility program BLKMAC. One source file, COMMON.FOR, is provided to BLKMAC as input. The file is a Fortran "Block Data" module, containing INCLUDE statements for each of the ".INC" files containing common blocks. Also data initialization statements for the global variables appear in COMMON.FOR. BLKMAC creates one object module for each COMMON statement encountered in the input stream. The file name will have the same name as the Fortran common block. Two VMS command files were designed to facilitate the generation of object modules.

```
@UTL:MACALL
@UTL:FORALL
```

The first command assembles all VAX macro source files on an account. The second command both compiles all Fortran files on the account and automatically executes BLKMAC to compile global data specification files.

The next step is the creation of global section access files. These files are used by the VMS linker and by the executable images to determine the global section access allowed for the individual executable images. The VMS command GLOBAL_SECTIONS (GLOBAL for short) is executed to interactively select the global section usage for each of the applications images. This command gets the names of all the images and global sections from the file GBLNAME.DAT, which must exist on the current default directory. Information about read and write access to the various global sections must be provided for each executable image. This information is used to generate the ".OPT" files and the VAX macro file MAPTBL.MAR. Also the user is prompted for VIEW passwords. When all password entries are complete, information about the global section access privileges for each password must be provided. The file PASS.MAR is created from this information. When GLOBAL is finished the two ".MAR" files must be assembled as follows.

```
GLOBAL
<interactive session>
MAC PASS,MAPTBL
```

The last step for the creation of the executable images is Linking. All the required linking is performed by using the build command file provided.

```
@BUILD
```

Appendix D GENERAL UTILITIES

A group of general purpose utility procedures are used by display software. All but one reside in the utility library UTIL.OLB. The following is a list of the modules used from the library. The description of these modules is provided in the flight software utilities reference manual.

Condition Handling
C_HDL EXCEPTIONS REPORT REPORT_CHECK SHOW_TT

Data Formatting
BCDTIM FMTTIM OTS$FLOAT

Mathematics
ANGL MXV POLAR SCOS UVC VCP VDP VMG XYZ

Map projection
CLIP GRID POSBTS

Miscellaneous
ASSIGN GET GET_CHAR LOCK MAPCOM

The utility procedure PROJECT is not part of the utility library. Its description is provided on the following page.

MODULE NAME: PROJECT
FILE NAME: PROJECT.FOR
PROCESS: DSPFST, DSPSLW
CALLED BY: TURN, LINE, PASSBY
CALLING SEQUENCE: CALL PROJECT(WPTS,I,DIST,RLAT,RLON)

PURPOSE:

To compute a position between waypoints.

DESCRIPTION:

This utility procedure is called to find the latitude and longitude of a point on the "great circle" arc connecting two waypoints. The first three items in the parameter list are inputs to PROJECT, while the last two are outputs. The first input parameter is one of the waypoint buffers, either the active (ACT_WPTS) or the provisional (MOD_WPTS). The index into the waypoint buffer of the end waypoint of the leg being processed is passed next. The last input parameter is the distance, in feet, to the desired position from the end waypoint. The values returned through the remaining parameters in the calling list are the latitude and longitude of the desired position.

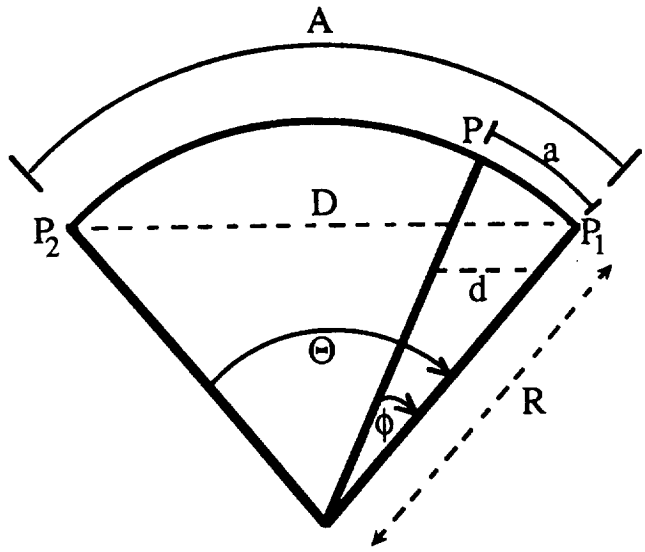
Figure D.1 on the following page shows how the position of the desired point is found. The values given are A, a, R, and the unit vectors for the arc end points.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
UVC POLAR

"GREAT CIRCLE" ARC

POSITION CALCULATION



$\Theta = A/R \qquad \phi = a/R$ {Subtended angles}

$$d/D = \beta = \frac{\sin(\phi)}{\sin(\Theta)\cos(\phi) + \sin(\phi) - \sin(\phi)\cos(\Theta)}$$ {Chord ratio }

$$\vec{P} = \beta \hat{P}_2 + (1 - \beta)\hat{P}_1$$ {direction vector}

$$\hat{P} = \frac{\vec{P}}{|\vec{P}|}$$ Unit Vector

$\text{LATITUDE} = \text{ARCSIN}(\hat{P}_x)$

$\text{LONGITUDE} = \text{ARCTANGENT}(-\hat{P}_y, \hat{P}_z)$

-figure D.1-

Appendix E DISPLAY FORMAT "FREEZE"

The global variable FREEZE is used to stop portions of the display software while the display processes are active. This technique is used for debugging display formats and customizing display screen appearance for photograph sessions. The utility VIEW is used to manually modify the variable FREEZE. By default the variable is set to zero for normal display software operation. Note that the variable will be automatically cleared if the flight management flag LABFLG is not set on. Shown below are the values that may be placed in FREEZE with VIEW, and their effect on the VAX display software.

FREEZE = 1

No DATAC or interprocessor link I/O is performed. This freezes the current sensor and FM/FC inputs at their current values. Display microprocessor I/O is performed as usual.

DISFIL and FFPRC are not called. This eliminates the intermediate processing of input variables.

Map background requests are generated regularly. This is normally performed by the FM/FC update request MAPUPD, which is no longer received. This is done since the navigation format require fresh map backgrounds at least every 15 seconds.

FREEZE = 3

Everything from FREEZE = 1.

Display applications software is not executed, except map background generation modules. This means the display microprocessor output buffer remains frozen with the last values sent to the formats.

When FREEZE is set to either value, the utility VIEW may be used to manually modify VAX display variables which effect the display formats in the microprocessors. With FREEZE = 1 the inputs to the applications modules are changed with VIEW. These modules will then perform their computations on the input variables and format the display microprocessor output buffer. When using FREEZE = 3, the display microprocessor output buffer must be modified directly with VIEW. The advantage of the first setting is the user can work in engineering units such as feet and

degrees and does not need to know the format of the display output buffer. The setting of "3" is used when it is more convenient to modify the output buffer directly, or the active applications software produces undesirable results. This happens in the case of display outputs that are the differential of input signals. Since inputs are frozen at their last value, the differential becomes zero, which will be stored by an applications module into the display output buffer when FREEZE = 1.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 1992	Contractor Report

4. TITLE AND SUBTITLE
Advanced Transport Operating System (ATOPS) Color Displays Software Description - MicroVAX System

5. FUNDING NUMBERS
C NAS1-19038
WU 505-64-13-11

6. AUTHOR(S)
Christopher J. Slominski
Valerie E. Plyler
Richard W. Dickson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Computer Sciences Corporation
3217 North Armistead Avenue
Hampton, Virginia 23666-1379

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665-5225

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA CR-189603

11. SUPPLEMENTARY NOTES
Langley Technical Monitor: Dr. James R. Schiess (COTR)
Robert A. Kudlinski

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited

Subject Category 06

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This document describes the software created for the Display MicroVAX computer used for the Advanced Transport Operating Systems (ATOPS) project on the Transport Systems Research Vehicle (TSRV). The software delivery of February 27, 1991, known as the "baseline display system", is the one described in this document. Throughout this publication, module descriptions are presented in a standardized format which contains module purpose, calling sequence, detailed description and global references. The global references section includes subroutines, functions and common variables referenced by a particular module.

The system described supports the Research Flight Deck (RFD) of the TSRV. The RFD contains eight Cathode Ray Tubes (CRTs) which depict a Primary Flight Display, Navigation Display, System Warning Display, Takeoff Performance Monitoring System Display, and Engine Display.

14. SUBJECT TERMS			15. NUMBER OF PAGES
Electronic Flight Instrumentation System	Glass Cockpit		293
Primary Flight Display	Multifunction Display		16. PRICE CODE
Navigation Display	Flight Display Software		A13

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL