

171911

171911

P. 210

Advanced Transport Operating System (ATOPS) Flight Management/Flight Controls (FM/FC) Software Description

David A. Wolverton
Richard W. Dickson
Winston C. Clinedinst
Christopher J. Slominski

*Computer Sciences Corporation
Hampton, Virginia*

Prepared For
Langley Research Center
under Contract NAS1-19038
April 1993

(NASA-CR-191457) ADVANCED
TRANSPORT OPERATING SYSTEM (ATOPS)
FLIGHT MANAGEMENT/FLIGHT CONTROLS
(FM/FC) SOFTWARE DESCRIPTION
Report, Jan. 1989 - Feb. 1991
(Computer Sciences Corp.) 370 p

N93-28621

Unclass

G3/06 0171911



National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-0001



TABLE OF CONTENTS

1.0	INTRODUCTION	1
2.0	SYSTEM OVERVIEW	2
2.1	PROCESSES AND EXECUTABLE IMAGES	3
	HDL (I/O Handler)	4
	FMFAST (Flight Management FAST loop executive)	5
	FCFAST (Flight Controls FAST loop executive)	6
	SLOW (Flight Management SLOW loop executive)	7
	DSTAR (DAS/SNAP Table Access Routine)	8
	VIEW (Global Section Data View Utility)	9
2.2	GLOBAL SECTIONS	10
2.3	INSPECTING GLOBAL VARIABLES WITH VIEW	12
2.4	STARTING AND STOPPING VAX FM/FC SOFTWARE	14
2.5	CONDITION HANDLING	15
2.5.1	EXCEPTION LOG FILES	17
3.0	I/O COMMUNICATIONS	18
	HDL	19
	HDL_MESG	25
	IOFLL	26
	IN10M	27
	GPSPKT	28
	OUT10M	29
4.0	EXECUTIVE SOFTWARE	31
	FMFAST	32
	FCFAST	33
	SLOW	34
5.0	FLIGHT MANAGEMENT	35
	OVERVIEW	35
5.1	FAST LOOP PROCESSING:	36
	CDUFST (CDU Fast Loop Processing)	36
	ACCPRC (Accelerometer Processing)	38
	GPSPRC (Global Positioning Satellite Processing)	39
	MLSEX (Microwave Landing System (MLS) Executive)	41
	RSCON (Reset Constants)	43
	CNTRM (Counter Module)	44
	CTLBLK (Control Block)	46
	PFILT (Pre-filter)	47
	XFORM (Coordinate Transformation)	49
	XYZIN (R-Az-El to X-Y-Z)	50
	CFILT (Complementary Filter)	52

PRINV (Inverse Computations)	54
HNAVFS (Horizontal/Vertical Navigation, Fast)	55
HNAVML (Horizontal/Vertical Navigation, MLS)	60
HNAVB (Horizontal/Vertical Navigation, plus Baro)	62
HNSWIT (Horizontal/Vertical Navigation Switching)	65
HVGUID (Horizontal/Vertical Guidance)	69
LEGSW (Leg Switching)	73
GD3D (3D Guidance Initialization)	74
AAA (Turn Vector Processing)	75
DTG1 (Distance-to-go (Turn) Calculations)	76
HVG2 (Distance-to-go (Straight Leg) Calculations)	77
TRALCBA (Primary Lateral Guidance Computations)	78
HVG6 (Primary Vertical Guidance Computations)	81
TGUID (Time Guidance)	83
AAT (Turn Computations)	86
CDG (Primary Time Guidance Computations)	88
NAVIG (Simulated Airplane)	90
FLYIC (Initialization)	92
ENGAGE_CAS (Operate Simulator)	93
5.2 SLOW LOOP PROCESSING:	95
CDUEXC (CDU Executive)	95
BLOW (Wind Computations)	96
EPRLMT (Engine Pressure Ratio Limit)	98
ERAD (Earth Radii Computations)	101
HNAVSL (Horizontal Navigation Slow Loop)	103
RADCAL (Earth Radius Calculations)	109
CRBSC (Range and Bearing Calculations)	110
TUNPTH (Path Defined Station Tuning)	112
NXTPS (Next Path Station)	114
TUNEPS (Tune Path Station)	115
T1CHEX (Tuning Checks)	116
TUNDM2 (Autotune DME #2)	117
NXTSTA2 (Select Next Station)	118
SEARCH_STA2 (Find Next Station)	119
BMPSTA2 (Pick Next-in-strip)	120
CHOOSE_STR2 (Select Next Longitudinal Strip)	121
EXT_RGE2 (Extend Search Range)	122
LOOKL2 (Select Next Strip to West)	123
TUNXTK (Cross Track Station Tuning)	124
XTK_MAN (Manual Tuning)	125
XTK_AUTO (Auto Tuning)	126
GEOM_CK (Geometry Checks)	127
NXTSTA (Select Next Station)	128
SEARCH_INIT (Initialize Station Search)	129
SEARCH_STA (Find Next Station)	130
CMP_FREQ (Check Station Tuning)	131
BUMP_STA (Pick Next-strip)	132
CHOOSE_STRIP (Select Next Longitudinal Strip)	133
LOOK_L (Select Next Strip to West)	134
EXT_RANGE (Extend Search Range)	135
TUNCK (Verify Station Geometry)	136
GMSG (Generate Message)	137

	MESG (Error Message Tables)	138
	SNAPOUT (Snap Output Processing)	139
6.0	FLIGHT CONTROLS	140
	OVERVIEW	140
	DATSEL (Flight Controls Data Selection)	141
	MSPLGC (Mode Select Panel Logic)	143
	KNOBER (MSP Knob Processing)	152
	MSPRO (Mode Select Panel Readout)	153
	MLOG (Mode Logic)	155
	DETNT (Logical Function)	164
	LATCMD (Lateral Path Command)	165
	CMPF (Localizer Complementary Filter)	171
	LATRL (Lateral Axis Control Laws)	173
	FRCWS (Forward Flight Deck Roll CWS)	177
	RCOM (Roll Computer - CWS)	178
	RBASC (Roll Basic - CWS Modes)	182
	RCOMA (Roll Computer - Auto Modes)	184
	RBASCA (Roll Basics - Auto Modes)	186
	DCRAB (De-crab (Autoland) Maneuver)	187
	VERCMD (Vertical Path Command)	188
	PAL (Pitch Auto Land)	194
	ELEVP (Elevator Control Processing)	196
	PFFD (Pitch Forward Flight Deck)	199
	PAFD (Pitch Aft Flight Deck)	200
	PVPC (Pitch Vertical Path Command)	203
	VTFCL (Variable Tau Flare Control Law)	204
	STABT (Stabilizer Trim Logic)	205
	SPDCMD (Speed Command)	206
	ATHCL (Autothrottle Control Law)	210
	DSPOT (Displays Output)	212
	OUTIO (I/O Output Processing)	214
6.1	SIGNAL FAILURE DETECTION	215
	DINUSE (Discrete In-Use Computations)	215
	SINUSE (Sensor In-Use Computations)	216
	DISFD (Discrete Select and Failure Detect)	217
	F2CMP (2nd Failure Computations)	219
	F2CMP2	220
	EXITF1	222
	EXITF2	223
	FDSTR (Failure Data Storage)	224
	TEST_SENSORS	226
	FMTMG (Format Message)	227
	ICO	229
	DSTOR (Data Storage for STP)	230
	PANEL (System Test Panel Interface)	231
6.2	DATA RECORDING	232
	DASOT (DAS Recording Output)	233
	SNAP (Snap Data Storage)	236
	DUMPS	237

6.3	PREFLIGHT OVERVIEW	238
	PRFLT (Pre-flight Test Executive)	239
	CLBIS (Control Surface Bias Outputs)	241
	CTLCK (Controls Check)	242
	ILSRC (ILS Receiver Checks)	243
	RDALT (Radar Altimeter Checks)	244
	RGYRO (Rate Gyro Checks)	245
	SRVCK (Servo Checks)	246
7.0	DAS/SNAP TABLE ACCESS	248
	DSTAR (DAS/Snap Table Access Routine)	248
	NSNAP (New Snap Processing)	251
	DASDUMP (Dump Recording Tables)	252
	DASPRC (DAS Table Processing)	253
	GITEM (Get Item Definition)	254
	CHECK (Check for Multi-item Entry)	255
	SNAPDEL (Snap Delete)	256
	SYM_SEARCH (Search Symbol Table)	257
	SNAPMOD (Snap Modification)	259
	SNPRC (Snap Processing)	260
	SNAPDUMP (Dump Snap Tables)	261
	OLDSNAP (R/W Snap Tables to Disc)	262
	UCASE (Convert to Upper Case)	263
	APPENDICES	
	A - Digital Systems Diagrams	264
	B - Horizontal/Vertical Guidance Computations	265
	C - Microwave Landing System Processing	313
	D - VIEW Command Entries	337
	F - System Global Variables	343
		351

Section 1.0 INTRODUCTION

This document describes the software created for the Flight Management/Flight Controls (FM/FC) 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.

The FM/FC host computer works in tandem with another MicroVAX computer, referred to as the Displays computer. The document

Advanced Transport Operating System
Color Displays Software Description
MicroVAX System

should be referenced for information about Displays software.

Throughout this document, descriptions of software modules are presented in a standardized format. The basic template is shown below. At the top of the form is a header block containing miscellaneous information about the module, including 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 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.

MODULE NAME:
FILE NAME:
PROCESS:

PURPOSE:
.....

CALLED BY:
CALLING SEQUENCE:
CALLS TO:

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

GLOBAL INPUTS:

GLOBAL OUTPUTS:

Section 2.0 SYSTEM OVERVIEW

The various sub sections of the system overview briefly describe the overall configuration of the FM/FC 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.

- Digital 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 seven executable images associated with the FM/FC software. Three of them are utility programs and four are FM/FC applications programs. Their names are given below with a brief description of their purposes.

	(utilities)
DSTAR	manipulate data recording tables
SECTION	install and remove global sections
VIEW	monitor global variables
	(FM/FC applications)
HDL	perform system functions (timing, interrupts, I/O)
FMAFAST	perform Flight Management real-time calculations
FCFAST	perform Flight Controls real-time calculations
SLOW	perform Flight Management background processing

The environment created for the FM/FC executable images consists of five VAX processes. They are the initial process created from the user login and four 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 four images remain active continuously under the context of their own sub process. Since the FM/FC applications images are always active, the VMS priority system determines how often the images actually execute. HDL, FMAFAST, and FCFAST are assigned priorities within the VMS real-time range: 19, 18 and 17, respectively. SLOW uses the default round-robin priority of 4.

The four FM/FC applications images each have a well defined set of responsibilities. The remaining pages of this section list the computations performed by FMAFAST, FCFAST, SLOW, and HDL. The names of modules which make up each image are also included.

PROCESS NAME: HDL (I/O Handler)

PURPOSE: HDL initializes system resources to allow external I/O, to schedule this external I/O, and to control the subprocesses FMFAST, FCFAST, and SLOW. HDL also formats I/O data to/from these processes.

EXECUTION PRIORITY: 19

INVOKED BY: DCL SPAWN

GLOBAL SECTIONS USED [READ (R)/WRITE (W)]:
AADCOM[R], BCKCOM[R], CDUCOM[R/W], CIOCOM[R/W],
DISNAV[R/W], DLNCOM[R/W], DSTDAT[R], DTCCOM[R/W],
FCCOM[R], INPCOM[R/W], IPLCOM[R/W], NAVCOM[R/W],
OUTCOM[R/W], RECCOM[R]

DESCRIPTION:

The executable image HDL.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 HDL raises its priority into the real-time region at level 19 which disables quantum expiration context switching. HDL will then use system context to configure I/O channels for DMA with external devices (DATAAC, CVIU, Displays MicroVAX, etc). Upon initialization completion, HDL will spawn the subprocess DSTAR (if necessary) for a single pass through. The subprocesses FMFAST and FCFAST will be synchronized into a 50 millisecond frame by HDL using an interrupt from the DATAAC. The DATAAC will also supply a 10 millisecond clock for synchronization of DMA I/O. This I/O data is formatted for use by the flight application software by HDL.

PROCESS NAME: FMAFAST (Flight Management FAST loop executive)

PURPOSE: To perform initialization functions for the Flight Management (FM) foreground software and serve as the executive for FM supporting modules.

EXECUTION PRIORITY: 18

INVOKED BY: DCL SPAWN

GLOBAL SECTIONS USED [READ (R)/WRITE (W)]:
AADCOM[R], BCKCOM[R/W], CDUCOM[R/W], CIOCOM[R/W],
DISNAV[R/W], DLNCOM[R/W], FCCOM[R/W], INPCOM[R/W],
NAVCOM[R/W], OUTCOM[R/W], RECCOM[R/W]

DESCRIPTION:

The FM process consists of those routines that provide the capability to create and interact with the aircraft flight plan. The ability to inspect and enter information that affects the flight plan is provided through software that controls the CDU (Control and Display Unit - see Contractor Report 189606). Navigation is provided for by software control of the two Distance Measuring Equipment (DME) tuning heads on the aircraft. The ability exists to provide for automatic tuning or manual tuning as selected by the CDU. Instrument Landing System (ILS), Microwave Landing System (MLS) and/or Global Positioning Satellite signals may be used to augment the navigation depending upon selection via the CDU.

Aircraft guidance commands for the various automatic modes of flight are also computed by the FM software. These may include horizontal, vertical, and time guidance commands depending upon the pilot selected flight mode (see description of MSPLGC).

PROCESS NAME: FCFEAST (Flight Controls FAST loop executive)

PURPOSE: To perform initialization functions for the Flight Controls (FC) software and serve as the executive for supporting modules.

EXECUTION PRIORITY: 17

INVOKED BY: DCL SPAWN

GLOBAL SECTIONS USED [READ (R)/WRITE (W)]:

AADCOM[R], BCKCOM[R/W], CDUCOM[R], CIOCOM[R],
DLNCOM[R/W], DISNAV[R/W], DSTDAT[R], DTCCOM[R/W],
FCCOM[R/W], INPCOM[R], IPLCOM[R], NAVCOM[R/W],
OUTCOM[R/W], RECCOM[R/W]

DESCRIPTION:

The FC software receives sensor inputs from the various interface units, calculates according to the mode control logic and control laws selected, and issues commands to direct the aircraft in flight. The primary commands issued to the control surfaces include aileron command (AILCMD), elevator command (DECMD), rudder command (RUDCMD), autothrottle position command (APCDG), and stabilizer trim discrettes (TRIMR, TRIMD). The computation of these commands, as well as the accompanying logic and error reporting are covered in detail in the Flight Controls section of this document.

PROCESS NAME: SLOW (Flight Management SLOW loop executive)

PURPOSE: To provide background processing for FM functions such as navigation, CDU operations, printing snapshot data, and error messages.

EXECUTION PRIORITY: 4

INVOKED BY: DCL SPAWN

GLOBAL SECTIONS USED [READ (R) /WRITE (W)]:
AADCOM[R], BCKCOM[R/W], CDUCOM[R/W], CIOCOM[R/W],
DISNAV[R/W], DLNCOM[R/W], DTCCOM[R], FCCOM[R/W],
INPCOM[R], IPLCOM[R/W], NAVCOM[R/W], OUTCOM[R/W],
RECCOM[R/W]

DESCRIPTION:

The executable image SLOW.EXE is activated by the HDL process upon system start up. This process executes continually in the background mode supporting those non-time-critical functions such as CDU processing, tuning of ranging stations such as DME (distance measuring equipment) or TACAN (Tactical Air Navigation) stations, and printing of snapshot data and error messages.

PROCESS NAME: DSTAR (DAS/SNAP Table Access Routine)

PURPOSE: To provide an interactive method of selecting data recording variables and snapshot criteria and variables.

EXECUTION PRIORITY: 4 (20 when invoked by HDL process)

INVOKED BY: RUN DSTAR (for interactive use)
Spawned by HDL at priority 20 on startup

GLOBAL SECTIONS USED [READ (R)/WRITE (W)]:

AADCOM[R], BCKCOM[R], CDUCOM[R], CIOCOM[R], DISNAV[R],
DLNCOM[R], DSTDAT[R/W], DTCCOM[R], FCCOM[R], INPCOM[R],
IPLCOM[R], NAVCOM[R], OUTCOM[R], RECCOM[R/W]

DESCRIPTION:

DSTAR is an interactive process used to select data recording variables, snapshot criteria and variables, and set up alternate tables for data recording. It permits an experimenter to modify an existing data list, snapshot data specification, or alternate table. Data recording is not actually performed by this process but a table is created (DASPAR) which is used by data recording software (OUTIO) running in a foreground process (FMFAST) to select data, format it according to scale factors specified and output it to the DATAC bus. For a detailed description of DSTAR capabilities, see the Data Recording Section of this document.

PROCESS NAME: VIEW (Global Section Data View Utility)

PURPOSE: To examine and modify variables in the ATOPS global sections.

EXECUTION PRIORITY: 5

INVOKED BY: RUN VIEW

GLOBAL SECTIONS USED [READ (R)/WRITE (W)]:
AADCOM[R], BCKCOM[R/W], CDUCOM[R/W], CIOCOM[R/W],
DISNAV[R/W], DLNCOM[R/W], DSTDAT[R/W], DTCCOM[R/W],
FCCOM[R/W], INPCOM[R/W], IPLCOM[R/W], NAVCOM[R/W],
OUTCOM[R/W], RECCOM[R/W]

DESCRIPTION:

This is an interactive process used to examine and modify variables in the global sections used by the flight software. Upon process startup, the user is prompted for a password to determine the read/write privileges associated with that password. A default is obtained by entering a carriage return. Passwords and privileges are established during the system build by the system administrator. A detailed description of VIEW commands is contained in appendix D.

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 FM/FC software each global section consists of one relocatable program section following the standard definition of the VAX Fortran common block. The FM/FC software uses 14 global sections. All but two, AADCOM and DSTDAT, 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 FM/FC 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 database
BCKCOM	Flight Management background data
CDUCOM	Control Display Unit data
CIOCOM	CDU input/output data
DSTDAT	Default recording list
DISNAV	Output data to Displays MicroVAX
DLNCOM	Data link information
DTCCOM	I/O memory for aircraft DATAC bus
FCCOM	Flight Controls data
INPCOM	Formatted DATAC Input variables
IPLCOM	Data link information
NAVCOM	Navigation data
OUTCOM	Formatted DATAC Output variables
RECCOM	Data recording 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 FM/FC 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 startup. 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 the variable that 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 D for information on the VIEW commands.

VIEW [V5.1]: FM/FC BASELINE 12-FEB-1991

Page 1

1	I.2	0	MAXF
2	H.2	0020	FCFLGS
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 FM/FC SOFTWARE

There are 12 files needed for a complete FM/FC software system. These include the seven 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 FM/FC software. First it checks to see if an old set of log files is 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.

GO.COM

This command procedure opens the process exception log file and starts the executable image. It is called once for each of the four FM/FC software processes.

HALT.COM

This command procedure is used to properly terminate the VAX FM/FC 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 E 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 FM/FC 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. Finally the utility program DSTAR 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 FM/FC 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 FM/FC 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 (HDLCNT, FMCNT, FSTCNT, SLWCNT). 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 (FMFAST.LOG, FCFAST.LOG, HDL.LOG, SLOW.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.

Following is a list of the exceptional conditions handled by the FM/FC condition handler. Any other exceptions signaled to the FM/FC condition handler will simply be resignaled to 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 FM/FC 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 restarted.

(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 Fortran compiler switch /NOCHECK.

Section 2.5.1 EXCEPTION LOG FILES

Exception messages are saved in log files defined for each process (HDL.LOG, FMFAST.LOG, FCFEAST.LOG, SLOW.LOG). Inactive log files may be viewed with DCL commands such as TYPE, COPY, or PRINT. When the FM/FC software is executing, the active set of log files are accessed with the SHOW_LOG.COM command procedure. There are three forms available for 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 in 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 sequence.

Section 3.0 I/O COMMUNICATIONS

In order for FM/FC application software (FMFAST, FCFEAST, SLOW, DSTAR, 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 HDL.

HDL initializes system resources to allow external I/O, to schedule this external I/O, and to control the subprocesses FMFAST, FCFEAST and SLOW. HDL also formats I/O data to/from these processes. The executable image HDL.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 HDL raises its priority into the real-time region at level 19, which disables quantum expiration context switching. HDL will then use system context to configure I/O channels for DMA with external devices (DATAC, Displays MicroVAX, inter-processor link, etc). The subprocesses FMFAST and FCFEAST will be synchronized into a 50 millisecond frame by HDL 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 FM/FC application software by HDL.

MODULE NAME: HDL
 FILE NAME: HDL.MAR
 PROCESS: HDL

PURPOSE: Initializing system resources to allow external I/O, to schedule this external I/O, and to control the subprocesses FMFAST, FCFAST, and SLOW.

CALLED BY: VMS
 CALLING SEQUENCE: RUN HDL
 CALLS TO: IN10M, IOFLL, LOCK, MAPCOM, OUT10M, PCK_IO

DESCRIPTION:

This module is very intimately tied with, 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 VMS Version 5, Volume 8 - Device Support, paying particular attention to 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 HDL contains four functional parts. These include:

- I.) Initialization code
- II.) Main loop processing
- III.) Kernel mode routines used in initialization
- IV.) Connect to interrupt routines

The following describes each:

- I.) Initialization code - This code performs the following:
 - A.) 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.
 - B.) Declare an exit handler which will remove all created subprocesses and set process priority back to level 4.
 - C.) Set process priority to level 19. This puts the process HDL into the real-time range and will disable any quantum expiration context switching.

- D.) Lock P0 process space into the working set by calling LOCK. This will reduce the possibility of page faulting during main loop execution.
- E.) Assign a channel number to each of the following devices:
 - 1.) XAA0 - DATAC DRV11
 - 2.) XAB0 - IPL DRV11
 - 3.) XAD0 - CVIU DRV11
 - 4.) KWA0 - K WV11
- F.) Call MAPCOM to map to the required global sections.
- G.) Use \$CRMPSC to map to the physical addresses of the I/O registers for the DRV11 and K WV11 devices. This will allow the process to reference specific physical locations via virtual addresses.
- H.) 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.
- I.) Associate to common event flag cluster. These event flags are used for synchronization of the flight application processes.
- J.) If the flag STARTUP is set, spawn the process DSTAR at priority level 20 (one above the handler). The handler will wait at this point for DSTAR to terminate.
- K.) Connect to interrupt vectors. A connect to interrupt \$QIO is executed for each of the four DRV11 and the K WV11 devices. This establishes connect to interrupt init, start, interrupt service, and cancel routines as discussed in detail in section 4.
- L.) Initialize output variables for walking bit, sawtooth ramp, and control mode panel outputs.
- M.) Wait for first 50 millisecond interrupt, then enable 10 millisecond clock and start main

loop software.

II.) Main loop processing. Main loop processing begins by waiting for either a 10 or 50 millisecond interrupt. Upon determining which interrupt occurred, HDL 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. HDL will, at the beginning of a major frame:
- 1.) Read the data necessary for 10 millisecond processing.
 - 2.) Format 10 millisecond input data by calling IN10M.
 - 3.) Format 10 millisecond output data by calling OUT10M.
 - 4.) Output 262 words (112 words for effector data plus 150 words for data recording) in OUTCOM to the DATAC SIR. This includes the 10 millisecond output data. The corresponding hexadecimal SIR addresses are from 390 to 496. This data was loaded in the preceding major frame by the process FCFAST using the subroutine OUTIO.
 - 5.) Read 587 words of raw data from the DATAC SIR via DRV11 into DTCCOM. This includes hexadecimal SIR addresses 1B through 266.
 - 6.) Format input data for use by flight application software, using IOFLL, into INPCOM.

Upon completion of I/O, HDL will set event flags 64 and 65, enabling FMAFAST and FCFAST 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. HDL 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 - This frame consists of three functions:
 - a.) 10 millisecond input data is read from the DATAC SIR.
 - b.) 10 millisecond input data is formatted by IN10M.
 - c.) Data for 10 millisecond output is formatted by OUT10M.
 - b.) 10 millisecond output data is written to the DATAC SIR.
- 3.) Minor frame 2 - Identical to minor frame 1.
- 4.) Minor frame 3 - This frame is the same as minor frame 1, with the addition of the display VAX interprocessor link as follows:
 - a.) The word count for the interprocessor transfer is loaded into the DRV11 data register for reading by the display VAX. 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 sent is loaded into the DRV11 buffer's address register.
 - c.) The transfer is initiated.
- 5.) Minor frame 4 - This frame is the same as minor frame 1, with the addition of the CVIU I/O as follows:
 - a.) Thirteen words of keyboard input are read from the CVIU.
 - b.) The number of words of screen information to be sent to the CVIU is loaded into the DRV11 data register for

reading by the CVIU (zero words is a possibility).

c.) The transfer is initiated.

d.) Keyboard input data is unpacked by the routine KEYBRD.

III.) 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\$POBR (P0 base register) and PR\$POLR (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.

IV.) 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. Generally, the only function this has 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 KVV11 (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 INPUTS: MODCNT

GLOBAL OUTPUTS: PACKET_RD_LST (in module GPSPKT.MAR), HDLCNT,
MS10ML, MS50ML, MFRAME, DTC_TO, IPL_NZ, ACT_CNT, ACT_BUF,
SEND_BUF, IOWAIT, CDUCNT

MODULE NAME: HDL_MESG
FILE NAME: HDL_MESG.MAR
PROCESS: HDL

PURPOSE: Contains error messages used by HDL.

CALLED BY: Non executable
CALLING SEQUENCE: Not called
CALLS TO: None

DESCRIPTION:

HDL_MESG contains ASCII error messages used by HDL. In the event of an error in initialization, these messages would be output to the default terminal of the process HDL.

GLOBAL INPUTS: N/A

GLOBAL OUTPUTS: N/A

MODULE NAME: IOFLL
FILE NAME: IOFLL.MAR
PROCESS: HDL

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

CALLED BY: HDL
CALLING SEQUENCE: JSB IOFLL
CALLS TO: None

DESCRIPTION:

IOFLL 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 flight application software. IOFLL stores formatted input data into INPCOM.

IOFLL 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.) DSMPLX - Takes as input MLS data, checks for valid, shifts, scales, and stores the resultant floating point value at the address specified by R0 and the signal validity boolean at the address R0 plus the offset specified in the macro call.
- 4.) 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.
- 5.) GSMPLX - Similar to ASMPLX but tailored for use in formatting GPS raw input data.
- 6.) SBOOL1 - Tests the specified bit in R2 and, if set, sets the boolean pointed to by R0. Otherwise the boolean is cleared.
- 7.) SBOOL2 - Tests the specified bit in R2 and, if clear, will set the boolean pointed to by R0. Otherwise the boolean is cleared.

GLOBAL INPUTS: DTC_IN, MAGVAR

GLOBAL OUTPUTS: All variables in INPCOM plus ALTATT (in DISNAV)

MODULE NAME: IN10M
FILE NAME: IN10M.MAR
PROCESS: HDL

PURPOSE: To format 10 millisecond DATAC SIR DMA input data into INPCOM.

CALLED BY: HDL
CALLING SEQUENCE: JSB IN10M
CALLS TO: None

DESCRIPTION:

IN10M is called at a 10 millisecond rate to format 10 millisecond raw SIR input data into floating point format for use by flight application software. This formatted data is written into INPCOM.

IN10M 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.) ASMPX - Similar to SMPLXF with the addition of a validity bit test preceding the conversion. Result is stored at the address specified in R0.
- 3.) FILTF - In addition to converting and scaling, the input value is filtered.

GLOBAL INPUTS: DTC_IN

GLOBAL OUTPUTS: BMACIN, HDD, P, Q, R

28

MODULE NAME: GPSPKT
FILE NAME: GPSPKT.MAR
PROCESS: HDL

PURPOSE: To format differential GPS data radio uplinked from the ground.

CALLED BY: HDL
CALLING SEQUENCE: JSB PCK_IO
CALLS TO: NONE

DESCRIPTION:

This module receives radio uplink data containing differential correction values needed by the GPSSU for its differential GPS mode. These values are received serially, checked for errors, formatted into the form needed for output, and loaded for output to the DIFF. The DIFF then sends this information to the GPSSU via an ARINC 409 channel. In addition, GPSPKT contains code to output an initialization stream to the Packet Terminal Node Controller (TNC). This sequence will be serially output to the TNC if the bottom bit of the variable PKT_CMD is set.

This module declares the global symbol PACKET_RD_LST which is modified by the module HDL.

GLOBAL INPUTS: PKT_CMD, RETRY, PKT_IN

GLOBAL OUTPUTS: GPS_OUT, HDL_DATA

MODULE NAME: OUT10M
FILE NAME: OUT10M.MAR
PROCESS: HDL

PURPOSE: To format 10 millisecond output data from OUTCOM for DMA to DATAC SIR.

CALLED BY: HDL
CALLING SEQUENCE: JSB OUT10M
CALLS TO: None

DESCRIPTION:

OUT10M is called at a 100 Hertz rate (10 millisecond interval) before each output of 10 millisecond SIR data. It is used to format floating point data from flight application software into fixed point data and store that data in DTC_OUT for output. Primary outputs are the aileron, elevator, rudder and throttle commands (AILCMD, DECMD, RUDCMD and APCDG).

OUT10M uses the programmer defined macro FLOAT which will scale a floating point value pointed to by R0, convert it to 16-bit integer and store it at the destination specified in the macro call. Each of the primary surface commands (AILCMD, DECMD and RUDCMD) are output to two separate SIR memory addresses (separated by a fixed interval) to drive the A and B servo channels. External hardware compares the two outputs and completes the link to the servo system iff the two commands are equal. The FLOAT macro performs both outputs from one call if a final 'D' parameter is coded. The above describes all processing performed for the rudder and throttle commands (RUDCMD and APCDG, respectively).

The aileron and elevator commands are more complex in that the rate damping signals (P and Q, respectively) are summed with the basic command at the 100 Hertz rate by OUT10M. Additionally, each of these commands is provided with a test mode which allows an operator to command the output of a specified fixed surface command for a specified period of time. The desired output command (KDECP) is specified in integer form at the output scaling, and the time (KDECT) is specified as the (integer) number of 10 millisecond iterations. To treat the simpler surface first, the vertical control law inner loop creates a nominal elevator command without pitch rate damping ($DECMQ = DECMD - Q$) at the 20 Hertz rate. It also computes the 16 second lagged Q output QX. QX approximates the pitch rate gyro bias and $Q - QX$ the true pitch rate. In the present software, DECMD is only output to the B channel. The A channel has been fitted with hardware pitch rate complementation and receives DECMQ and KQ, the gain to be used for Q. (Q is not applied to either channel if a fixed value is to be output). OUT10M performs the following computations:

```

if (KDECT > 0) then
  DTC_OUT(31) = KDECP
  DTC_OUT(43) = KDECP
  DTC_OUT(37) = 0
  KDECT = KDECT - 1
else
  R1 = Q - QX
  DECMD = DECMQ + KQ*R1
  DTC_OUT(31) = INT(102.4*DECMD)
  DTC_OUT(43) = INT(102.4*DECMQ)
  DTC_OUT(37) = INT(204.8*KQ)
endif

```

Roll rate (P) complementation of the nominal aileron command (AILCMP) to produce the actual aileron command (AILCMD) is also performed in OUT10M. Roll rate is not debiased as pitch rate is, but rather passed through a 50 msec lag filter to produce PF1. Depending on control mode and the bits specified in LATSEL, AILCMD is either passed through a rate limited lag or simply complemented with the lagged roll rate. The following computations are performed.

```

PF1 = KTPFL*(PF1 - P) + P
if ((mode = CWS) and ((LATSEL and '100'X) ≠ 0)) then
  R2 = AILCMP + KP*P - AILCMD
  R1 = R2*(1. - KTAUPF)*KAILG
  (Where KAILG = 1.0)
  If (R1 > ALRTL) then
    R1 = ALRTL
  elseif (R1 < -ALRTL) then
    R1 = -ALRTL
  endif
  R1 = limit((R1 + AILCMD), 12.)
  AILCMD = R1
else
  AILCMD = AILCMP + KP*PF1
endif

if (KAILT > 0) then
  DTC_OUT(29) = KDECP
  DTC_OUT(41) = KDECP
  KAILT = KAILT - 1
else
  DTC_OUT(29) = 102.4*AILCMD
  DTC_OUT(41) = 102.4*AILCMD
endif

```

GLOBAL INPUTS: AILCMD, AILCMP, ALRTL, APCDG, DECMQ, KAILP, KAILG, KAILT, KDECP, KDECT, KQ, KTAUPF, KTPFL, LATSEL, MODEX, P, Q, QX, RUDCMD

GLOBAL OUTPUTS: AILCMD, DECMD, DTC_OUT, KAILT, KDECT, PF1

Section 4.0 EXECUTIVE SOFTWARE

The four FM/FC VAX applications processes each have main modules which are entered directly from VMS when their respective executable images are started. The executable image HDL is covered in section 3.0 and will not be mentioned again in this section. The processes FMFAST, FCFAST and SLOW each have main modules which are described in the following pages.

Main modules contain operations to set up and initialize items which effect the entire process in which they reside. They also serve as a caller of subroutines which perform the actual computations for Flight Management, Navigation, Guidance and Flight Controls.

MODULE NAME: FMFAST
FILE NAME: FMFAST.FOR
PROCESS: FMFAST

PURPOSE: To serve as the executive routine for the Flight Management fast loop processing.

CALLED BY: DCL SPAWN
CALLING SEQUENCE: N/A
CALLS TO: ACCPRC, ASSIGN, CDUFST, GPSPRC, HNAVFS, HVGUID, MAPCOM, MLSEX, NAVIG, TGUID

DESCRIPTION:

This module serves as the executive for the FM fast loop processing. It invokes the VMS system service SYSSCMEXEC to call the LOCK utility to lock the P0 working set in memory. If any errors occur while attempting this operation, a message is displayed on the system console and the process stops. The next initialization activity is to associate to cluster #2 event flag 64 for communication with the I/O executive. Once again, if any errors occur during this process, execution is terminated. MAPCOM is then called to link the FMFAST process to the appropriate global sections. The address of the current DAS time is placed in a variable (TM_ADR) to be used by an exception handler for the ATOPS flight software processes. This handler is activated via the LIB\$ESTABLISH VMS system service. Next, SYS\$ERROR and SYS\$OUTPUT are assigned to the appropriate unit for the display of run-time error and informative messages. The final initialization activities are setting the cold start (COLDST) variable true and clearing the timer overflow indicator (FMOVER).

At this point the VMS system service SYSS\$WAITFR is invoked to wait for the setting of event flag 64 from the I/O Handler which indicates that data has arrived from the DATAC and foreground processing can begin. Once received, this flag is immediately cleared and the FM foreground modules are called in the following order: CDUFST, ACCPRC, MLSEX (if RUNM is true), GPSPRC, NAVIG, HNAVFS, HVGUID, and TGUID.

The final activity of FMFAST is to read event flag 64 again. If it is set at this point it indicates that a timer overflow has occurred (since data must have arrived from the DATAC prior to foreground processing completion) and an overflow flag is incremented accordingly. In any event, processing continues by returning to the activities described in the previous paragraph.

GLOBAL INPUTS: HRSS, RUNM

GLOBAL OUTPUTS: COLDST, FMCNT, FMOVER

MODULE NAME: FCFAST
 FILE NAME: FCFAST.FOR
 PROCESS: FCFAST

PURPOSE: To serve as the executive routine for the Flight Controls fast loop processing.

CALLED BY: DCL SPAWN
 CALLING SEQUENCE: N/A
 CALLS TO: ASSIGN, ATHCL, DASOT, DATSEL, DINUSE, DISFD,
 DSPOT, ELEVP, FDSTR, F2CMP, LATCMD, LATRL,
 MAPCOM, MLOG, MSPLGC, MSPRO, PANEL, PRFLT,
 SINUSE, SNAP, SPDCMD, OUTIO, VERCMD

DESCRIPTION:

This module serves as the executive for the FC fast loop processing. It invokes the VMS system service SYS\$CMEXEC to call the LOCK utility to lock the P0 working set in memory. If any errors occur while attempting this operation, a message is displayed on the system console and the process stops. The next initialization activity is to associate to cluster #2 event flag 65 for communication with the I/O executive. Once again, if any errors occur during this process, execution is terminated. MAPCOM is then called to link the FCFAST process to the appropriate global sections. The address of the current DAS time is placed in a variable (TM_ADR) to be used by an exception handler for the ATOPS flight software processes. This handler is activated via the LIB\$ESTABLISH VMS system service. Next, SYS\$ERROR and SYS\$OUTPUT are assigned to the appropriate unit for the display of run-time error and informative messages. The final initialization activities are setting the cold start (COLDST) variable true and clearing the timer overflow indicator (FMOVER).

At this point the VMS system service SYS\$WAITFR is invoked to wait for the setting of event flag 65 from the I/O Handler, which indicates that data has arrived from the DATAC and foreground processing can begin. Once received, this flag is immediately cleared and the FC foreground modules are called (conditionally) in the following order: DINUSE, SINUSE, DISFD, DATSEL, MSPLGC, MLOG, LATCMD, VERCMD, SPDCMD, LATRL, ELEVP, ATHCL, PRFLT, FDSTR, F2CMP, PANEL, MSPRO, DSPOT, DASOT, SNAP and OUTIO.

The final activity of FCFAST is to read event flag 65 again. If it is set at this point it indicates that a timer overflow has occurred (since data must have arrived from the DATAC prior to foreground processing completion) and an overflow flag is incremented accordingly. In any event, processing continues by returning to the activities described in the previous paragraph.

GLOBAL INPUTS: FCOVER, FLYFLG, FSTCNT, HOLDM, MAXF, MFRAME,
 MSWIT, RUNM, TIME, TOG100, WDTV

GLOBAL OUTPUTS: COLDST, FCCNT, FCOVER, FSTCNT, MAXF, TIME,
 TOG100, WDTV

MODULE NAME: SLOW
 FILE NAME: SLOW.FOR

PURPOSE: To serve as the executive routine for the Flight Management slow loop processing.

CALLED BY: DCL SPAWN
 CALLING SEQUENCE: N/A
 CALLS TO: ASSIGN, BLOW, CDUEXC, EPRLMT, ERAD, GMSG, HNAVSL, LOCK, MAPCOM, SNAPOUT

DESCRIPTION:

This module serves as the executive for Flight Management slow loop processing. Upon start up it calls LOCK to lock the P0 working set into memory to preclude being "swapped out". It then calls MAPCOM to link the SLOW process to the appropriate global sections for communication with the foreground processes (FMFAST and FCFAST). The address of the current DAS time is placed in a variable (TM_ADR) to be used by an exception handler for the ATOPS flight software processes. This handler is activated via the LIB\$ESTABLISH VMS system service. Next SYS\$ERROR and SYS\$OUTPUT are assigned to the appropriate unit for the display of run-time error and informative messages. The final initialization activities are the setting of the CDU initialization flag (CDU_INIT) and the assigning of the onboard line printer and system test panel to the appropriate logical units for the display of flight software failure messages, the printing of data recording tables and snapshot data.

Following the initialization activities is a sequence of code that is executed continuously until the system is halted. This code consists of a series of calls to background software routines in the following order: CDUEXC, ERAD, BLOW, EPRLMT, HNAVSL, SNAPOUT (if I/O not active and data is available for printing) and GMSG (if I/O not active and data is waiting for display).

The final activity of SLOW is to clear the cold start flag (COLDST) and update the exception counter (SLWCNT). As stated previously, processing continues by returning to the activities described in the preceding paragraph.

GLOBAL INPUTS: HRSS, IOACT, RPTR, SPTR, WRDCNT

GLOBAL OUTPUTS: CDU_INIT, COLDST, SLWCNT

Section 5.0 FLIGHT MANAGEMENT

FLIGHT MANAGEMENT OVERVIEW

The Flight Management routines provide for navigation, guidance and the ability to create and interact with the aircraft flight plan. The ability to inspect and enter information that affects the flight plan is provided through software that controls the CDU (Control Display Unit) and MSP (Mode Select Panel). The look-up and usage of the system data base (Bulk Data) is done via the CDU. This software is described in the Advanced Transport Operating System (ATOPS) Control Display Unit Software Description, NASA Contractor Report 189606.

The MSP controls the selection of the various modes of automatic guidance and -- depending on the selected mode -- can control the position of the aircraft directly using the MSP knobs for airspeed, altitude, flight path angle and track angle. Navigation and guidance software is described in this section. The MSP interface software resides in the Flight Controls process and is described in that section of this document.

5.1 FAST LOOP PROCESSING:

MODULE NAME: CDUFST (CDU Fast Loop Processing)
FILE NAME: CDUFST.FOR
PROCESS: FMFAST

PURPOSE: Serve as executive for the fast loop CDU processing.

CALLED BY: FMFAST
CALLING SEQUENCE: CALL CDUFST
CALLS TO: GET_REAL

DESCRIPTION:

This procedure is the interface between the background CDU software in the process SLOW, and the flight systems real-time processing. There are four distinct sections of code in the module corresponding to independent CDU functions.

The first section controls Navigation Display update requests. The variable GDTIME contains the time of the last update request made by the flight management computer. This variable is compared to the current system time, and if they differ by at least six seconds, an update request is made. Some CDU modules force a display update by storing a zero in GDTIME. An update request consists of setting the flag SEND_BUF to cause the I/O handler process to transmit the guidance buffers to the display computer. The actual update request flag MAPUPD is set on two frames after the I/O handler request to assure the guidance buffer transmissions are complete.

The map background of the Navigation Display is centered on particular reference points when it is placed in 'Plan mode'. CDUFST stores the latitude and longitude of a reference point which is determined by current CDU modes. If there is an active flight plan, the map center will be the current destination waypoint. If there is only a provisional flight plan available, the reference point becomes the last waypoint on the flight plan. If neither of these situations are satisfied, the map center becomes the origin airfield (if selected) or the current position of the aircraft.

When a new flight plan has been enabled, the sequence control variable SETGD is set to '2' to start the determination of the available guidance modes of the new flight plan. The code waits one complete real-time frame before setting the mode flags to assure the demode/re-mode operations occurring while modifying a flight plan do not occur in one real-time frame. After the wait, the guidance flags are set as follows. '2D' guidance will be allowed as long as the new flight plan contains two or more waypoints. If '2D' guidance is possible and each waypoint has an associated altitude constraint, '3D' guidance will be allowed. '4D' guidance is allowed when '3D' guidance has been verified, a speed constraint exists for each waypoint, and Reference Time of Arrival (RTA) has been assigned to one of the flight plan waypoints.

The last section of CDUFST determines when a TOPMS run has completed. Once a TOPMS run has been initiated, the CDU portion will be forced to reset if the aircraft has accelerated to 64 knots and subsequently decelerated to below 20 knots. The run is also canceled if the aircraft wheel squat switch becomes false or the displays computer signals termination through the discrete word DISPST.

GLOBAL INPUTS: ACTCNT, AIRPTS, CTRF, DISPST, GDTIME, GS, GUID2D, GUID3D, GUID4D, LAT, LATCEN, LON, LONCEN, MAPUPD, MODCNT, NAV64K, PMODE, RTA_PTR, SEND_BUF, SETGD, SOAT, SQUAT, TIME, TKFLEN, TOINDX, TOWD, TOWPT, TOWS, TST3D, TST4D, WPT_ACT, WPT_MOD

GLOBAL OUTPUTS: GDTIME, GUID4D, LATCEN, LONCEN, MAPUPD, SEND_BUF, SETGD, SOAT, TKFLEN, TOINDX, TOWD, TOWS

MODULE NAME: ACCPRC (Accelerometer Processing)
 FILE NAME: ACCPRC.FOR
 PROCESS: FMFAST

PURPOSE: To compute and debias the body mounted accelerometer inputs.

CALLED BY: FMFAST
 CALLING SEQUENCE: CALL ACCPRC
 CALLS TO: SCOSD, MXV

DESCRIPTION:

This module computes the body to flight path vector rotation matrix (LMB) as follows:

LMB(1,1) = CTHET*CDTK
 LMB(1,2) = SROLL*STHET*CDTK - CROLL*SDTK
 LMB(1,3) = CROLL*STHET*CDTK + SROLL*SDTK

LMB(2,1) = CTHET*SDTK
 LMB(2,2) = SROLL*STHET*SDTK + CROLL*CDTK
 LMB(2,3) = CROLL*STHET*SDTK - SROLL*CDTK

LMB(3,1) = STHET
 LMB(3,2) = -SROLL*CTHET
 LMB(3,3) = -CROLL*CTHET

Where: SDTK = sin(DFTANG), STHET = sin(PITCH),
 SROLL = sin(ROLL), etc.

It then sums the bias terms (BIASBA) computed by MLSEX with the accelerometer input vector (BMACIN) to produce the debiased accelerations in the body axis (BMACC), and rotates BMACC into the inertial vector (via a call to MXV) to produce ACCB. This vector is conditionally unpacked into XTKACC, ATKACC and HDD by other procedures (ELEVP, HNAVFS).

GLOBAL INPUTS: BIASBA, BMACIN, DFTANG, PITCH, ROLL

GLOBAL OUTPUTS: ACCB, BMACC, CROLL, CTHET, SROLL, STHET

MODULE NAME: GPSPRC (Global Positioning Satellite Processing)
 FILE NAME: GPSPRC.FOR
 PROCESS: FMFAST

PURPOSE: To compute GPS derived X, Y and Z position, and analogs of the glideslope and localizer deviation variables to be used for display and for the land control laws.

CALLED BY: FMFAST
 CALLING SEQUENCE: CALL GPSPRC
 CALLS TO: MXV, SCOSD, SQRT

DESCRIPTION:

GPSPRC begins by examining the GPS status words (GRSSTx), the GPSVLD discrete (set by IOFLL based on the input data validity bits) and the Horizontal Dilution of Precision figure (GPHDOP - input from the GPSSU by IOFLL). This information is used to set the GPS Navigation Valid (GPNAVV) and GPS select index (GPSSEL: 0-bad, 1-poor, 2-good), used by the navigation algorithms, as well as the Satellites in view (SATIN VW) and GPS receiver mode (GPSMOD) words, displayed for operator information. If GPNAVV is true and other criteria are met, the GPS Land Valid (GPLNDV) is also set. If GPLNDV is true and GPS land is selected (GPLND true), a bit is set in FCFLGS to cause 'GPS ON' to be displayed on the Nav Display unit.

If GPNAVV is false, the IRS North and East velocities are integrated into Hybrid Latitude and Longitude to keep these data current. Differences are then computed between the hybrid LAT, LON and ALT and the equivalent IDD and MLS quantities. Deltas are also computed between absolute GPS and IDD Lat and Lon. This data is for monitoring purposes only.

If GPLND is selected, GPLNDV is true, LAND mode requirements are met and MLSMOD is false, then Hybrid LAT, LON and ALT are also differenced with MLS Azimuth or ILS Localizer antenna site LAT, LON and ALT. The resultant deltas are converted to runway coordinate X, Y, Z and passed through a second order filter to produce equivalents of the MLS XHAT, YHAT and ZHAT. This vector is then used to produce GPS derived equivalent values of the ILS glideslope (GPSBTA) and localizer (GPSETA). Hybrid VN, VE, and HDOT are also rotated into the runway axes to produce equivalents of the MLS velocities, XDH, YDH and ZDH. Either an ILS or MLS type approach and landing may then be made, depending on the setting of the MLS configuration word (MCONF).

GLOBAL INPUTS: AIRPTS, ALTCOR, ANTLAT, ANTLON, COLDST, COSRH, CROLL, CTNET, DIFMOD, DLATFT, DLONFT, FLYFLG, GPHDOP, GPLND, GPNAV, GPSLAT, GPSLON, GPVDOP, GPSVLD, GRSST1, GRSST2, GSA, GUID2D, HDGTRU, HYBALT, HYBHDT, HYBLAT, HYBLON, ICM, IDDLAT, IDDLON, ILSZON, LANDR, LANDE, LMB, MCONF, MINSATH, MLSALT, MLSLAT, MLSLON, MLSMOD, MXHDOP, RUN, RWYHDG, RYELEV, SINRH, SROLL, STNET, TANGSA, VEINS, VNINS, WGSMSL, WPT_ACT,

GLOBAL OUTPUTS: DLALTM, DLLATA, DLLATM, DLLONA, DLLONM, DLTALT,
DLTLAT, DLTLOH, FCFLGS, GPBTAV, GPINIT, GPLNDV, GPNAVV,
GPSBTA, GPSETA, GPSMOD, GPSSEL, HGPIP, HYBLAT, HYBLON,
LMB, SATINVW, XGPIP, XDH, XHAT, YDH, YHAT, YPROF, ZDH,
ZHAT

MODULE NAME: MLSEX (Microwave Landing System (MLS) Executive)
 FILE NAME: MLSEX.FOR
 PROCESS: FMFAST

PURPOSE: To calculate the aircraft position and velocities in the
 MLS coordinate system.

CALLED BY: FMFAST
 CALLING SEQUENCE: CALL MLSEX
 CALLS TO: CFILT, CNTRM, CTLBLK, PFILT, PRINV, RSCON, UNPK,
 XFORM, XYZIN

DESCRIPTION:

Module MLSEX contains the computational routines which provide the MLS derived inputs to the Navigation, Display and Flight Control systems. It consists of a short executive portion and a series of subroutines called conditionally from the executive. All processing is under control of the MLS Configuration word (MCONF), which is normally set via the VIEW utility. Among other things, this word determines whether MLS calculations are to be made and, if so, whether the real or simulated input data is to be used. The configuration control parameters are identified and described in Appendix C, as well as in NAVCOM.INC.

MLSEX first checks for initialization conditions. If the MLS Compute discrete (MLSC) is false, or the Flight Controls IC button has been pressed or the MLS configuration word has been changed (MCONF not equal PMCONF), The first pass flag (FPF) is set to force re-initialization when computations are next begun. The MLS valid discrete (MLSVL) is then cleared and UNPK is called to unpack the upper six bits of the configuration word into the associated booleans. (The next two bits of MCONF affect only the usage of MLS parameters by other modules, and are unpacked in procedure MLOG when MLS mode is selected). The MSB of the configuration word is then checked. If clear, or if LABFLG is true and ILSZON is false (simulation in the EASILY lab generates erroneous signals when outside of normal coverage), MLSC and MLSVAL are cleared and processing ends. Otherwise, processing continues by checking the FPF flag.

If FPF is true, subroutine RSCON is called to set up the airport and receiving antenna parameters according to the selection words RWYSEL and ANTSEL. MLSEX next calls each of the remaining subroutines in the order described below:

- CNTRM - Operate the validity counters;
- CTLBLK - Compute the Complementary Filter Run counter (CFRUN) and the solution validity (MLSVL);
- PFILT - Operate the MLS signal prefilters;
- XFORM - Compute the rotation matrix LMB and rotate the receive antenna vector into the runway axis;

- XYZIN - Compute XYZ position from R, Az & E11;
- CFILT - Operate the XYZ Complementary Filter;
- PRINV - Compute the predicted R, Az & E11 inputs for the next pass.

CFILT and PRINV are only called when CFRUN is non-zero.

GLOBAL INPUTS: CFRUN, IC, ILSZON, LABFLG, MCONF, MLSC

GLOBAL OUTPUTS: BMAFLG, EL2F, MLSC, MLSVAL, RLMLS, SIMILS, VGSFLG

MODULE NAME: RSCON (Reset Constants)
FILE NAME: MLSEX.FOR
PROCESS: FMFAST

PURPOSE: To set up the airport and receiving antenna parameters according to the selection words RWYSEL and ANTSEL.

CALLED BY: MLSEX, HNAVFS
CALLING SEQUENCE: CALL RSCON
CALLS TO: SCOSD

DESCRIPTION:

RSCON is called by MLSEX on the first pass after MLSC becomes true, and by HNAVFS when MLSMOD is set true. This latter call is to take care of the case when Global Positioning Satellite navigation is selected into MLS coverage and some GPS autoland parameters might have been set up prior to selection of MLS mode. RSCON loads an entry of the RWYDEF array into the RWY_DEF vector according to the RWYSEL index, and loads an entry of the ANT_OFF array into the ANT_POS vector according to the ANTSEL index. RWYSEL values of 1 and 2 select definitions for runways at Wallops Island and Atlantic City, respectively, with a third entry reserved. ANTSEL values of 0, 1 and 2 select the tail, roof and chin antennas, respectively. The ANT_POS vector describes the position of the selected aircraft receiving antenna relative to the aircraft center of gravity, and the RWY_DEF vector describes the location of the Range and EL1 transmitter antennas and the glidepath intercept point (GPIP) relative to the MLS Azimuth antenna (MLS origin), and the Lat, Lon, elevation and phase center bearing (AZ_BRG) of the Azimuth antenna.

RSCON also calls SCOSD to compute the sine and cosine of AZ_BRG (SINAZB, COSAZB).

GLOBAL INPUTS: ANTSEL, RWYDEF, RWYSEL

GLOBAL OUTPUTS: COSAZB, MCONF, RWY_DEF, SINAZB

MODULE NAME: CNTRM (Counter Module)
 FILE NAME: MLSEX.FOR
 PROCESS: FMFAST

PURPOSE: To maintain a history of the MLS data validities and the complementary filter position limit exceedances.

CALLED BY: MLSEX
 CALLING SEQUENCE: CALL CNTRM
 CALLS TO: None

DESCRIPTION:

This module maintains a 128-cycle history of the valid flags associated with the MLS data (MLSSV = RV, AZV, EL1V), and the complementary filter position limit exceedants (CF_LIM_XC), and sets the IC_PF, PF_IC, and PF_CVAL Boolean arrays. Each history consists of a 128-bit bit string (4 long words). The associated counter (MLSSVC, CFXCC) is updated only when the state of the validity bit differs from the state of the history bit for that cycle. The counter thus indicates the number of valid cycles out the previous 128.

If the FPF or ICMLS flag is set, all flags, counters and histories are reset to their initial values and the ICMLS flag is cleared. (FPF is used again in subroutine CTLBLK). Otherwise, processing continues by monitoring the signal validity and complementary filter validity in a three pass 'DO' loop (for R, Az, EL1 and X, Y, Z). Initially, the local signal valid (MLSLSV) is set equal to the raw signal valid. Then, once the complementary filter has run at least 6 seconds (CFRUN > 120), outlier errors (OTLERR) are computed as (MLSRW - MLSS_PRED). OTLIR_V is set true if the corresponding OTLERR is less than OUTLIR_LIM, and MLSLSV is set to the 'AND' of MLSSV and OUTLIR_V.

Next the validity history (MLSS_HIST) and validity counter (MLSSVC) is updated. The counter value is then checked to set the operate discrettes. If MLSLSV is true, the following is performed:

- o The appropriate bit in MLSS_HIST is set and, if previously clear, MLSSVC is incremented;
- o If MLSSVC is equal to 1 or 58, the corresponding IC_PF flag is set, indicating that some prefilter initialization must be performed. If equal to 58, the PF_IC flag is also set, indicating that the prefilter for this signal has been fully initialized;
- o If the validity count is ≥ 115 , the PF_CVAL flag is set to indicate that the corresponding signal is usable for position computation (see XYZIN).

If `MLSLSV` is false the appropriate bit in `MLSS_HIST` is cleared and, if appropriate, `MLSSVC` is decremented.

If `MLS` is valid (`MLSVAL` is set by `CTLBLK` when `CFRUN` reaches 200), the following is performed:

- o The prefilter validity (`PF_CVAL`) and `PF_IC` flags are cleared when the associated counter becomes less than 18.
- o The complementary filter exceedance counters and histories are updated.
- o If `CF_LIM_XC` is true (a limit has been exceeded), the appropriate history is updated. If `CF_XC_C` then exceeds 70, `CFXCV` is set false, which will subsequently cause `MLSVAL` to be lost.
- o If `CF_LIM_XC` is false, the counter and history are updated. No further action is necessary, as `CFXCV` is set true during initialization, and can only be set false when `MLSVAL` is true and `CF_LIM_XC` is also true.

If `MLSVAL` is false, the `PF_CVAL` flag is cleared when its associated counter becomes less than 100, the `PF_IC` flag is cleared at 58 and the `IC_PF` flag is set at a count of zero.

Finally, the history pointers (indices) are updated in preparation for the next iteration.

GLOBAL INPUTS: `CFRUN`, `MLSRAW`, `MLSSV`, `MLSVAL`

GLOBAL OUTPUTS: `CFRUN`, `EX`, `OTLERR`

MODULE NAME: CTLBLK (Control Block)
 FILE NAME: MLSEX.FOR
 PROCESS: FMFAST

PURPOSE: To compute the CFILT Run counter (CFRUN) and the MLS Valid flag (MLSVAL).

CALLED BY: MLSEX
 CALLING SEQUENCE: CALL CTLBLK
 CALLS TO: VMG

DESCRIPTION:

This module controls selection of the altitude (Z) reference, initialization of the MLS Complementary Filter, operation of the CF Run counter (CFRUN), and setting of the MLS valid discrete (MLSVAL).

Initially, the first pass flag (FPF) is checked. If set, it is cleared, the ICMLS flag is set, the MLS second fail flag (FAIL2(9), set by F2CMP if any signal required by MLS is invalid) is cleared, and the procedure is exited. If FPF is clear, processing continues by setting the CF Count Valid (CF_CVAL) discrete initially equal to the 'AND' of the three exceedance count valid flags (CFXCV), the PF count valid flags for Range and Azimuth (PF_CVAL) and the 'NOT' of FAIL2(9). The altitude reference index (ALTREF) is initialized to zero. If CF_CVAL is true and GRD is false (the aircraft is in the air), ALTREF is selected as follows: If the MLS 'X' position is greater than X_HRSW (part of RWY_DEF), EL1 is primary and only; else HRAD is primary and only. HRAD is the default reference if GRD is true or CF_CVAL false.

If the primary altitude reference is invalid, CF_CVAL is cleared. Otherwise, the switch ALTREF is set to the selected reference. A check is then made of the Inertial Reference System (IRS) validity flags. If the IRS Attitude Valid (IATTV) is false, or either IRS Nav Valid (INAVV) or IRS Serial Bus Valid (ISBV) is false when the Body Mounted Accelerometer select flag (BMAFLG) is false, a counter is incremented. If this counter reaches 5, CF_CVAL is cleared.

If, after all checks, CF_CVAL is true and CFRUN is less than 200, the CFRUN counter is checked. If CFRUN = 0, ICCF is set true (causing the complementary filter to be initialized). If CFRUN is less than 200, CFRUN is incremented if the magnitude of the error vector (EX) is less than 210 feet. Once CFRUN becomes equal to 200, MLSVAL is set true.

If CF_CVAL is false, MLSVAL is set false and CFRUN is cleared to 0. If CFRUN was initially greater than zero, ICMLS is also set to force a complete reinitialization of the solution.

GLOBAL INPUTS: BMAFLG, CFRUN, EX, FAIL2, GRD, HRV, IATTV, INAVV, ISBV, MLSRAW, MLSSV, MLSVAL, POSHAT, RADALT, X_HRSW

GLOBAL OUTPUTS: CFRUN, FAIL2, MLSVAL

MODULE NAME: PFILT (Pre-filter)
 FILE NAME: MLSEX.FOR
 PROCESS: FMFAST

PURPOSE: To prefilter the MLS input parameters.

CALLED BY: MLSEX
 CALLING SEQUENCE: CALL PFILT
 CALLS TO: None

DESCRIPTION:

The Prefilter module uses alpha-beta filters to prefilter the MLS input parameters (Range, Azimuth, EL1). The IC_PF flags are OR'ed with the ICMLS flag to determine initialization processing. If the result is true, the respective prefilter is IC'd by setting both the filtered output value (S_HAT) and the next predicted value (MLSS_P) equal to the measured value (MLSRW) and the respective IC_PF flag is cleared. Note that IC_PF will be true on the first and 58th cycles of initialization. If both IC_PF and ICMLS (first cycle only) are true, the filter velocity estimate (MLSS_DHAT) is also zeroed.

If no initialization is to be performed, processing continues as follows:

If MLSLSV is true, filter input (TEMP) is the difference between the measured and predicted (MLSS_P) values. If MLSLSV is false and MLSVAL is true, filter input is the difference between MLSS_P and the prediction from PRINV (MLSS_PRED). I.e., the measured values are selectively replaced by predictions derived from the complementary filter outputs.

If both MLSLSV and MLSVAL are false, filter input is the difference between the measured and predicted values (limited to the maximum value specified by OUTLIR_LIM) when the input signal valid (MLSSV) is true, and zero otherwise.

If PF_IC is false, (indicating that the particular filter has operated less than 58 cycles since initialization), the value of TEMP is doubled, effectively cutting the filter time constant in half. This is to cause the filter to converge more rapidly during the initialization phase. The following equations are then evaluated:

$$\begin{aligned} S_HAT &= MLSS_P + ALPHA \ TEMP \\ MLSS_DHAT &= MLSS_DHAT + BETA \ TEMP \\ MLSS_P &= S_HAT + MLSS_DHAT \ DELTAT \end{aligned}$$

Where: ALPHA and BETA are functions of the filter time constants.

Finally, if MLSVAL is true and MLSLSV is also true, the filter output for that signal (S_HAT) is set equal to the measured value. This prevents the prefilter lags from affecting the complementary filter once that filter is initialized and stable, but retains the availability of stable prefiltered values for CF initialization and for those periods when the input signal may be erratic.

48

GLOBAL INPUTS: MLSRAW, MLSSV, MLSVAL

GLOBAL OUTPUTS: None

MODULE NAME: XFORM (Coordinate Transformation)
 FILE NAME: MLSEX.FOR
 PROCESS: FMFAST

PURPOSE: To compute the MLS rotation matrix LMB.

CALLED BY: MLSEX
 CALLING SEQUENCE: CALL XFORM
 CALLS TO: MXV, SCOSD

DESCRIPTION:

This module computes the 3 X 3 transformation matrix necessary to expand an aircraft body axis vector on the MLS coordinate frame. It also computes the vector ANT_VEC used by XYZIN and PRINV, which translates the X, Y, Z position from the aircraft antenna position to the aircraft center of gravity. The following equations are evaluated:

$$[LMB] = \begin{bmatrix} C\theta Ca & S\phi S\theta Ca - C\phi Ca & C\phi S\theta Ca + S\phi Sa \\ -C\theta Sa & -S\phi S\theta Sa - C\phi Ca & -C\phi S\theta Sa + S\phi Ca \\ S\theta & -S\phi C\theta & -C\phi C\theta \end{bmatrix}$$

Where: $C\theta$ = cosine(PITCH), $S\phi$ = sine(ROLL), etc;
 a = True Heading - Azimuth bearing + 180

$$\{ANT_VEC\} = [LMB] \{ANT_POS\}$$

Where: $\{ANT_POS\}$ is the X,Y,Z position of the MLS receiving antenna relative to the aircraft center of gravity (C.G.).

GLOBAL INPUTS: AZ_BRG, CROLL, CTHET, HDGTRU, SROLL, STHET

GLOBAL OUTPUTS: LMB

MODULE NAME: XYZIN (R-Az-E1 to X-Y-Z)
 FILE NAME: MLSEX.FOR
 PROCESS: FMFAST

PURPOSE: To compute the position of the aircraft in the MLS coordinate frame.

CALLED BY: MLSEX
 CALLING SEQUENCE: CALL XYZIN
 CALLS TO: SCOSD, SQRT, TAN

DESCRIPTION:

This module calculates the position of the aircraft center of gravity (C.G.) in the MLS coordinate system. It derives position information from one of the following set of sources depending on the switch ALTREF.

ALTREF = 0) range, azimuth and radio altitude data
 1) range, azimuth and elevation 1 data

XYZIN begins calculations when both range and azimuth prefilters have run at least 58 iterations (PF_IC true), even though the solution is not used until 115 iterations (PF_CVAL true). This permits the use of a semi-iterative method.

Position is first calculated at the receiving antenna, then translated to the aircraft C.G. The following equations are evaluated:

$$R_{az} = \hat{R} + \hat{X}_{dme} \cos(\hat{Az}) - \hat{Y}_{dme} \sin(\hat{Az})$$

$$Y_a = -R_{az} \sin(\hat{Az})$$

$$X_a = \sqrt{R_{az}^2 - Y_a^2 - Z_a^2}.$$

Where: \hat{R} and \hat{Az} are the prefiltered range and azimuth signals;

\hat{X}_{dme} and \hat{Y}_{dme} are the position of the DME relative to the Az antenna;

R_{az} is computed range to the MLS azimuth antenna;

X_a and Y_a are the X and Y position of the receiving antenna in the MLS coordinate system.

If ALTREF is equal to zero, the following equation is calculated:

$$Z_a = H_{rad} - H_{tdc} - ANT_VEC(3)$$

Where: H_{rad} is pitch corrected radio altitude;

H_{tdc} is the correction to H_{rad} to produce C.G. altitude above the MLS plane;

$ANT_VEC(3)$ is the displacement of the receiving antenna above the C.G.;

Z_a is the Z coordinate of the receiving antenna.

Otherwise,

$$Z_a = \tan(\overset{\wedge}{EL1}) \sqrt{(X_a - X_{el1})^2 + (Y_a - Y_{el1})^2} + Z_{el1g}$$

Where: $EL1$ is the prefiltered $EL1$ signal;

X_{el1} , Y_{el1} , Z_{el1g} are the XYZ coordinates of the $EL1$ antenna relative to the Azimuth antenna;

Z_a is the Z coordinate of the receiving antenna.

(all cases)

$$\{ POS_CG \} = \{ ANT_VEC \} + \begin{array}{c} - \quad - \\ | X_a | \\ | Y_a | \\ | Z_a | \\ - \quad - \end{array}$$

GLOBAL INPUTS: $EL1_DEP$, H_TDC , $PITCH$, $RADALT$, $ROMLS$, X_DME ,
 Y_DME , X_EL1 , Y_EL1 , Z_EL1G

GLOBAL OUTPUTS: None

MODULE NAME: CFILT (Complementary Filter)
 FILE NAME: MLSEX.FOR
 PROCESS: FMFAST

PURPOSE: To operate the MLS complementary filter.

CALLED BY: MLSEX
 CALLING SEQUENCE: CALL CFILT
 CALLS TO: MXV, VXM

DESCRIPTION:

The CFILT subroutine initializes and operates the MLS third order Complementary Filter, with operator selectable options as specified below. The MLS CF integrates accelerations into velocities and positions, with a complementary correction term (EX) derived from the position output of XYZIN (POS_CG).

The filter is operated whenever CFRUN is non-zero. Input accelerations in the MLS coordinate frame (vector ACC) may be derived from the IRS or from the Body Mounted Accelerometers, depending on BMAFLG. (BMA accelerations are also selected by default when the aircraft is moving too slowly for the IRS accelerations to be valid). BMA accelerations (vector BMACC computed in module ACCIN) are rotated into the MLS coordinate frame using the Lambda (LMB) matrix developed by module XFORM. Acceleration due to gravity is then subtracted from the Z axis. IRS cross track and along track accelerations are rotated into the MLS frame using the sine and cosine of runway heading error. (Vertical acceleration from the IRS is already in the proper coordinate frame).

Filter initialization occurs when ICCF is set true by CTLBLK. During initialization, position output is set to POS_CG, acceleration biases are zeroed, and vertical velocity (VELHAT(3)) is set to HDOT. XDH (VELHAT(1)) and YDH (VELHAT(2)) may be initialized from MLS prefilter velocity terms or from IRS VN and VE, depending on the setting of VGS_FLG.

Filter operation begins by updating the previous position estimate (POSHAT) by the distance moved in one iteration, and computing the residual term, EX:

$$\{ Xtp \} = \{ POSHAT \} + dt \left(\{ VELHAT \} + \{ ACCHAT \} dt / 2. \right)$$

$$\{ EX \} = \{ POS_CG \} - \{ Xtp \}$$

EX is then limited, and a flag (CF_LIM_XC) set for each element which exceeded the limit. Module CNTRM keeps a record of exceedances and causes loss of MLSVAL if any counter reaches 70 out of the previous 128 iterations.

If IRS accelerations are in use, the acceleration estimate (ACCHAT) is computed by summing ACC with the bias vector BIASIR. The bias vector is then updated by adding proportions of the present and previous values of EX:

$$\{ EXBA \} = DK(3) \{ EX \}$$

$$\{ \text{BIASIR} \} = \{ \text{BIASIR} \} - C1 \{ \text{EXBA} \} + C2 \{ \text{EXBAL} \}$$

$$\{ \text{EXBAL} \} = \{ \text{EXBA} \}$$

If BM accelerations are in use, the bias has already been applied by ACCIN, so ACCHAT is simply set equal to ACC. The bias vector (BIASBA) is then updated by first rotating EX into the body axis, then summing as above:

$$\{ \text{EXBA} \} = \text{DK}(3) \{ \text{EX} \} [\text{LMB}]$$

$$\{ \text{BIASBA} \} = \{ \text{BIASBA} \} - C1 \{ \text{EXBA} \} + C2 \{ \text{EXBAL} \}$$

$$\{ \text{EXBAL} \} = \{ \text{EXBA} \}$$

Processing concludes by computing the velocity (VELHAT) and position (POSHAT) estimates:

$$\{ \overset{\wedge}{\text{XDDtp}} \} = \{ \text{ACCHAT} \} + \text{DK}(2) \{ \overset{\wedge}{\text{EX}} \}$$

$$\{ \text{VELHAT} \} = \{ \overset{\wedge}{\text{VELHAT}} \} + C1 \{ \overset{\wedge}{\text{XDDtp}} \} - C2 \{ \text{XDDLST} \}$$

$$\{ \overset{\wedge}{\text{XDDLST}} \} = \{ \overset{\wedge}{\text{XDDtp}} \}$$

$$\{ \overset{\wedge}{\text{XDtp}} \} = \{ \text{VELHAT} \} + \text{DK}(1) \{ \overset{\wedge}{\text{EX}} \}$$

$$\{ \text{POSHAT} \} = \{ \overset{\wedge}{\text{POSHAT}} \} + C1 \{ \overset{\wedge}{\text{XDtp}} \} - C2 \{ \text{XDLST} \}$$

$$\{ \text{XDLST} \} = \{ \overset{\wedge}{\text{XDtp}} \}$$

GLOBAL INPUTS: ACCHAT, ATKINS, BIASBA, BMAFLG, BMACC, CFRUN,
 COSAZB, GSINS, HDDINS, HDOT, LMB, NAV64K, POSHAT, SINAZB,
 VE, VEINS, VELHAT, VGS_FLG, VN, VNINS, XTKINS

GLOBAL OUTPUTS: ACCHAT, BIASBA, EX, POSHAT, VELHAT

MODULE NAME PRINV (Inverse Computations)
 FILE NAME: MLSEX.FOR
 PROCESS: FMFAST

PURPOSE: To compute predicted values of the next MLS signal inputs.

CALLED BY: MLSEX
 CALLING SEQUENCE: CALL PRINV
 CALLS TO: MXV, VXM

DESCRIPTION:

PRINV computes predicted values for the next set of MLS measurements based on the present position estimate (POSHAT), velocity estimate (VELHAT) and the antenna configuration of the selected runway. These predictions are used in outlier computations, and to selectively edit the signal inputs to PFILT whenever the measured signals are determined to be invalid.

First POSHAT is updated by the distance traveled during the next iteration, then transformed to the receiving antenna location (Xa, Ya, Za) by summing with the ANT_VEC output of module XFORM. Slant range to the DME antenna is next computed using the Pythagorean theorem.

$$R_p = \text{sqrt}((X_a - X_{dme})^2 + (Y_a - Y_{dme})^2 + Z_a^2) - R_{OMLS}$$

Note that the difference in elevation between the DME and Azimuth antennas is assumed to be insignificant.

The angle transmitters (Az, EL1) define a conical angle. Thus Azimuth is computed as follows, using the four quadrant arctan.

$$Az_p = \text{atan2}(-Y_{AH}, \text{sqrt}(X_a^2 + Y_a^2))$$

EL1 is computed using the small angle arctan:

$$EL1_p = \text{atan} \frac{Z_a - Z_{EL1}}{\text{sqrt}((X_a - X_{EL1})^2 + (Y_a - Y_{EL1})^2)} - EL1_DEP$$

GLOBAL INPUTS: EL1_DEP, POSHAT, R_OMLS, VELHAT, X_DME, X_EL1,
 Y_DME, Y_EL1, Z_EL1G

GLOBAL OUTPUTS: None

MODULE NAME: HNAVFS (Horizontal/Vertical Navigation, Fast)
 FILE NAME: HNAVFS.FOR
 PROCESS: FMFAST

PURPOSE: To provide fast loop signal selection and integrations for navigation.

CALLED BY: FMFAST
 CALLING SEQUENCE: CALL HNAVFS
 CALLS TO: ANGL, ATAN2D, COSD, HNAVB, HNAVML, HNSWIT, RSCON, SCOSD, SIND, SQRT

DESCRIPTION:

The horizontal/vertical navigation routine (HNAVFS), together with its sub-procedures HNAVB, HNAVML and HNSWIT, computes and/or selects the position, velocity and acceleration data used by the guidance and display procedures. A simplified overview of the navigation system is depicted in figure 5-1. HNAVFS also maintains flags and mode words to indicate which source is in use for navigation.

Processing begins by performing initialization as required. The Radio Mode flag (FLRM) and local IRS valid flag (INSVAL) are initialized OFF and, if COLDST is true, NAVFLG, NAV64K, LLINIT, HDWAIT, TDWAIT and PASS1 are also initialized. TASFPS is then computed and, if NAV64K is true, the vertical component is subtracted to form TASGS.

$$\text{TASGS} = \begin{matrix} \text{TAS,} & (\text{GS} < 64 \text{ kts}) \\ \text{sqrt}(\text{TASFPS}^2 - \text{HDOT}^2) / \text{KTOFPS,} & (\text{GS} \geq 64 \text{ kts}) \end{matrix}$$

TASGS is used for the Air Data mode and also by the BLOW subroutine. Next, the IRS and Air Data valids, the simulated airplane flag (FLYFLG), and certain bits in the MLS configuration word (MCONF) are examined to determine the appropriate source for velocity and acceleration data.

Air Data or Radio Mode must be engaged if neither FLYFLG nor INAVV (Inertial Navigation unit Valid) is true. The acceleration inputs, normally input from the IRS, are derived from the body-mounted accelerometers by external code (ACCB vector) and assigned to IDDATK and IDDXTK if bit 5 of MCONF is set. Otherwise, no acceleration input is available; IDDATK and IDDXTK are zeroed and NCUVAL is set false. If bit 4 of MCONF is set, HDD is taken from ACCB, otherwise it continues to be input from HDDINS. In either case, true heading (HDGTRU) is taken as magnetic heading (COMPASS) corrected for local variation (MAGVAR), and external procedure SCOSD is called to compute the sine and cosine of true heading (SINTH, COSTH).

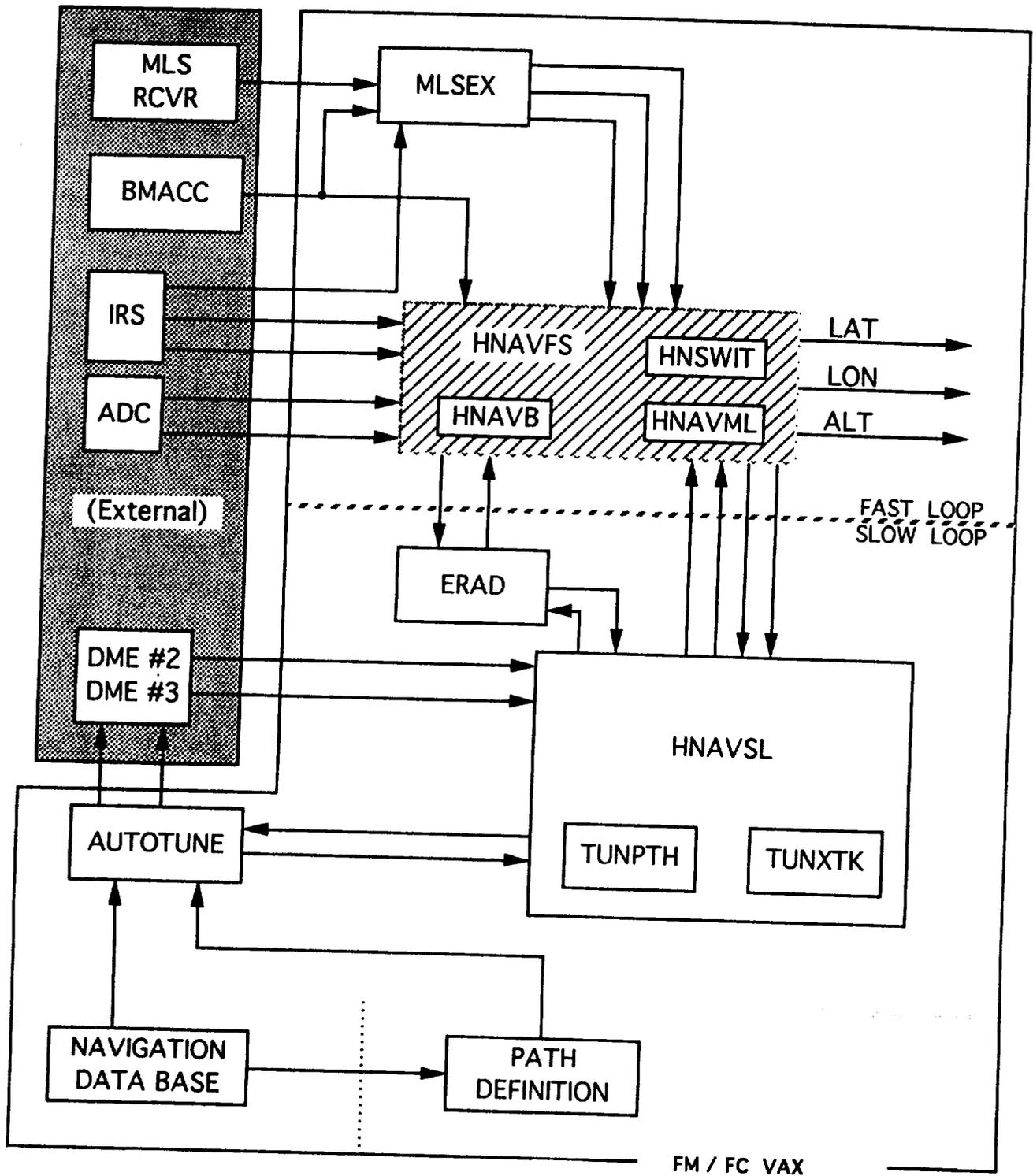


FIGURE 5-1: RELATIONSHIP BETWEEN HNAVFS AND THE REST OF THE SYSTEM

Air Data Mode will be selected if INAVV is false and the Air Data input is valid (ADVAL = true). The directional velocities are then calculated as:

EWVAVE = TASGS SINTKA
NSVAVE = TASGS COSTKA.

On the first pass for Air Data Mode, DVN and DVE are initialized as the difference between the directional velocities and the previous estimated IDD velocities:

DVN = IDDVN - NSVAVE
DVE = IDDVE - EWVAVE.

Radio Mode is selected by default if there is no valid airspeed input. No initialization is performed except for the mode flags FLRM (ON) and FLADM (OFF).

The Simulated Airplane is engaged if FLYFLG is true. The Simulator (procedure NAVIG) provides its own true heading (SMUHDG), true airspeed and cross track acceleration (IDDXTK), but no along track acceleration.

Inertial Mode, naturally, uses IRS derived velocity and acceleration data. Input variable initialization for some modes is summarized below:

<u>VARIABLE</u>	<u>SIMULATOR (FLYFLG)</u>	<u>SOURCE</u> <u>INERTIAL (INAVV)</u>	<u>AIR DATA (ADVAL)</u>
HDGTRU	SMUHDG	THDG	MAGHDG + MAGVAR
NSVAVE	TASGS COSTH	VNINS	TASGS COSTH
EWVAVE	TASGS SINTH	VEINS	TASGS SINTH
IDDATK	ZERO	ATKINS	ZERO (or ACCB(1))
IDDXTK	(set by NAVIG)	XTKINS	ZERO (or ACCB(2))
HDD	(set by NAVIG)	HDDINS	HDDINS (or ACCB(3))

For both simulated airplane and INS modes, INSVAL is set true and FLADM is set false. For all modes, the sine and cosine of HDGTRU (SINTH, COSTH) is computed once the source of HDGTRU has been determined. For INS mode, additional code exists to extract VN/VE from GSINS and COSTH/SINTH if the velocity data valid (VELVLD - set by IOFLL) is false (it was originally thought it might be at low ground speeds) and bit 6 of MCONF is set, or to extract IDDATK/IDDXTK from ACCB if the acceleration data valid (ACCVLD) is false (it is, at low ground speeds), but these options have not been required and probably should be removed. Also, on the first pass for INS mode, or following a change in the selected IRS unit (indicated by the unit field in IRSST1), DVN and DVE are initialized as the difference between the directional velocities and the previous estimated IDD velocities:

DVN = IDDVN - NSVAVE
DVE = IDDVE - EWVAVE.

Next, the conditions for the Microwave Landing System (MLS) mode are evaluated. The MLS data valid flag, MLSVLD, is initialized as MLSVAL * RUNM. (MLS computations are not performed if not in RUN mode.) MLS mode is selected by depressing the MLS bezel button on the navigation display unit. Code in the Displays MicroVAX sets the sign bit of the DISPST word for one iteration when the bezel button is depressed. This word is transmitted from the Displays to the FM/FC MicroVAX via the DATAC, and is used to set the MLS_ENABL Boolean if MLSVLD is true and MLSMOD is false, or clear it if MLSMOD is true. If the simulator is engaged or if MLS has not been selected, the mode flag (MLSMOD) is set false. Otherwise, MLSMOD is set true if MLSVLD and MLS_ENABL are both true. If MLSVLD is false, MLSMOD and MLS_ENABL are both set false.

Sub-procedure HNAVB is called next to calculate the Radio Nav (IDD) position and velocity data. These are based on the previously selected raw velocities (NSVAVE, EWVAVE), Earth radii of curvature (RMP, RNP - computed by procedure ERAD in SLOW), and either radio navigation data computed by HNAVSL in the SLOW process or the MLS data computed in procedure HNAVML. (See HNAVB and HNAVML discussion below.)

If MLSVLD is true, sub-procedure HNAVML is called to calculate position and velocity data based on the MLS solution. Yet a third set of data is available from the Global Positioning Satellite (GPS) system, with validity checks and selection logic performed by the external procedure GPSRPC. A count (NAVCTR) is kept of iterations where good radio navigation (R-nav) data is available. ("Good" R-nav data is here defined as a position computation performed with a circular error of less than 0.3 NM). When NAVCTR reaches 5000, the navigation valid flag (NAVVLD) is set. (NAVCTR continues to increment to a maximum value of 6000). If R-nav data is absent, NAVCTR is decremented and NAVVLD is cleared when NAVCTR becomes less than 2000. These switch points were chosen as a first approximation and may not be optimum. NAVVLD is also forced true if either MLS or GPS is valid and selected (bad idea!).

Finally, sub-procedure HNSWIT is called to evaluate the operator selection flags and the system validity flags and select one of the potential data sources for navigation. HNSWIT performs separate selection logic and maintains separate mode words for lateral and vertical navigation. In addition, an easy-on is performed to minimize transients when a mode switch occurs. See the description of HNSWIT for details.

Once all the required data have been selected, the sines and cosines of latitude and longitude are derived through calls to the system SIN and COS functions and groundspeed in knots (GS) and in feet per second (GSFPS) are derived from the North (VN) and East (VE) velocities. Then, if the groundspeed equals or exceeds 64 knots, or if both MLSMOD and NAVFLG (GS > 4) are set, the track angle (TK), the sine and cosine of track angle (SINTKA, COSTKA) and the flight path angle (GAMMA) are calculated from HDCF, VN, VE and GS. Functionally,

$$SINTKA = VE / GS$$

$COSTKA = VN / GS$
 $TK = \text{atan2}(VE, VN)$
 $GAMMA = (HDCF / GSFPS) (180 / \pi)$

Otherwise, the track angle, and the sine and cosine of the track angle, are set equal to the true heading, and sine and cosine of true heading (as they were set during initial mode determination) and GAMMA is set to zero.

Unconditionally, the drift angle and magnetic track angle are calculated through calls to external procedure ANGL. Then, if the groundspeed is below 4 knots, both the navigation mode flag (NAVFLG) and the 64 knot flag (NAV64K) are set false, and the ground speed references (GS and GSFPS) are limited to a minimum of one knot and the corresponding value in feet per second, respectively. This limit protects against subsequent division by zero. If the groundspeed exceeds four knots, NAVFLG is set true. At 64 knots, NAV64K is set true and the Lat/Lon initialization flag, LLINIT, is also set true to indicate that the latitude and longitude variables have been initialized.

GLOBAL INPUTS: ACCB, ACCVLD, ADVAL, ATKINS, COLDST, COMPASS, DISPST, DPE, DPN, EWVAVE, FLADM, FLYFLG, GPLND, GPNAV, GPNAV, GSINS, HDCF, HDOT, HDDINS, IC, INAVV, IRSST1, LAT, LON, MAGVAR, MCONF, MLSMOD, MLSSLI, MLSVAL, NAV64K, NAVCTR, RUNM, SMUHDG, TAS, THDG, VE, VEINS, VELVLD, VN, VNINS, XTKINS

GLOBAL OUTPUTS: CLAT, CLON, COSTH, DFTANG, DVE, DVN, FLADM, FLRM, GAMMA, GS, GSFPS, GSFPS2, HDD, HDGTRU, IDDATK, IDDXTK, LLINIT, MLSMOD, MLSVLD, NAV64K, NAVCTR, NAVVLD, NAVFLG, NCUVAL, SINTH, SLAT, SLON, TASFPS, TASGS, TK, TKMAG

MODULE NAME: HNAVML (Horizontal/Vertical Navigation, MLS)
 FILE NAME: HNAVFS.FOR
 PROCESS: FMFAST

PURPOSE: To provide Navigation position and velocities from the Microwave Landing System (MLS) solution.

CALLED BY: HNAVFS
 CALLING SEQUENCE: CALL HNAVML
 CALLS TO: COSD, SIND

DESCRIPTION:

This routine calculates navigation velocities and position based on MLS data inputs. It is called only when the MLS solution is valid. Processing begins by computing local copies of the lengths of a degree of latitude (DLTFT) and longitude (DLNFT) and the radii of curvature, LRM and LRL. It evaluates the same equations as are used by procedure ERAD, but bases them on average latitude (LATA) between the aircraft (LAT) and the MLS azimuth site (LAT_MLS) and the WGS-84 altitude of the MLS plane at the EL-1 site (ZOMLS + WGSMSL). It then rotates the MLS X and Y velocities (in fps) to North and East velocities (in knots), as follows:

$$\begin{aligned} \text{MLSVN} &= -(\text{VELHAT}_1 \cos(\text{AZ_BRG}) + \text{VELHAT}_2 \sin(\text{AZ_BRG})) / \text{KTOFPS} \\ \text{MLSVE} &= -(\text{VELHAT}_1 \sin(\text{AZ_BRG}) - \text{VELHAT}_2 \cos(\text{AZ_BRG})) / \text{KTOFPS} \end{aligned}$$

Delta latitude and longitude are then computed based on MLS X and Y position and DLTFT / DLNFT:

$$\begin{aligned} \text{DLAT} &= -(\text{POSHAT}_1 \cos(\text{AZ_BRG}) + \text{POSHAT}_2 \sin(\text{AZ_BRG})) / \text{DLTFT} \\ \text{DLON} &= -(\text{POSHAT}_1 \sin(\text{AZ_BRG}) - \text{POSHAT}_2 \cos(\text{AZ_BRG})) / \text{DLNFT} \end{aligned}$$

The MLS latitude and longitude estimate are computed by summing DLAT and DLON with the location of the MLS Azimuth antenna.

$$\begin{aligned} \text{MLSLAT} &= \text{LAT_MLS} + \text{DLAT} \\ \text{MLSLON} &= \text{LON_MLS} + \text{DLON} \end{aligned}$$

Finally, the Earth curvature (ZDIF) between the aircraft and the MLS EL1 site is computed and used to convert MLS Z and Z-dot (relative to the MLS plane) to altitude and altitude rate relative to the Earth.

$$\begin{aligned} \text{T1} &= (\text{POSHAT}(1) - \text{X_EL1})^2 \\ \text{T2} &= (\text{POSHAT}(2) - \text{Y_EL1})^2 \\ \text{TEMP} &= (\text{T1} + \text{T2}) / (\text{LRM} + \text{LRN}) * \text{NMTFT} \end{aligned}$$

"TEMP" is filtered into ZDIF using a one second time constant, producing both a smoothed altitude correction and an estimate of the altitude rate error, ZDIFD.

$$\begin{aligned} \text{ZDIFD} &= \text{TEMP} - \text{ZDIF} \\ \text{ZDIF} &= \text{TEMP} - \text{KT1SEC} * \text{ZDIFD} \end{aligned}$$

MLSALT and MLSHDT are then computed as follows:

$$\begin{aligned}\text{MLSALT} &= \text{POSHAT}_3 + \text{ZOMLS} + \text{ZDIF} \\ \text{MLSHDT} &= \text{VELHAT}_3 + \text{ZDIFD}.\end{aligned}$$

These MLS derived outputs are used for comparison purposes, and are conditionally output for navigation purposes by procedure HNSWIT.

GLOBAL INPUTS: COSAZB, LAT, LAT_MLS, LON_MLS, POSHAT, SINAZB,
 VELHAT, WGSMSL, X_EL1, Y_EL1, ZHAT, ZDIF, ZOMLS

GLOBAL OUTPUTS: MLSALT, MLSLAT, MLSLON, ZDIF

MODULE NAME: HNAVB (Horizontal/Vertical Navigation, plus Baro)
 FILE NAME: HNAVFS.FOR
 PROCESS: FMFAST

PURPOSE: To provide Radio Navigation positions and velocities and the inertially smoothed, Baro-corrected altitude and altitude rate.

CALLED BY: HNAVFS
 CALLING SEQUENCE: CALL HNAVB
 CALLS TO: SQRT

DESCRIPTION:

This procedure inputs the navigation position errors (DPN, DPE) computed by slow loop navigation (HNAVSL) or by the MLS navigation module (HNAVML) within this compilation unit, the Earth radii (RMP, RNP) calculated by procedure ERAD in the SLOW process and computes the corrected positions and velocities using a second order filter. Although the names (IDDxxx) imply Inertial Dual-Dme (inertial velocities integrated into dual DME corrected positions), the North and East velocity references may be either inertial, air data or none (Radio-only mode) as selected by HNAVFS. Similarly, the position reference may be based on DME's or on ILS or MLS landing aids as selected by HNAVSL or HNAVML. The filter gains (K1P, K2P) are set by HNAVSL, and adjust the filter time constant (LAT/LON, VN/VE) to 20, 30 or 50 seconds according to the selected velocity and position references. See the documentation for HNAVSL for details. The altitude filter is a third order filter driven by vertical acceleration (HDD) and baro altitude (HBARO). The time constant is fixed at 20 seconds. It has nothing to do with DME's.

If PASS1 is true (indicating that position information is available from HNAVML) and if MLSMOD is true or MLSSLI and MLSVLD are both true, the radio navigation gains and error terms are replaced by terms calculated from the MLS position estimates computed by procedure HNAVML.

$$\text{DPN} = (\text{MLSLAT} - \text{IDDLAT}) * \text{DTOR} * \text{RMP}$$

$$\text{DPE} = (\text{MLSLON} - \text{IDDLON}) * \text{DTOR} * \text{RNP}$$

Where: MSLAT, MSLON is the position calculated by HNAVML;
 RMP, RNP are the North and East radii of curvature (in NM) calculated by ERAD; and
 DTOR is the degree to radian conversion factor.

If MLSVLD is false, PASS1 is cleared. If PASS1 was true on the previous pass, DPN and DPE are zeroed to prevent them from being integrated further.

If FLYFLG is false and either NAVFLG is true (indicating that groundspeed is greater than four knots) or MLSMOD and PASS1 are both true, DVN and DVE are integrated from DPN and DPE to correct for position errors determined in the slow loop. They are then summed with the velocities NSVAVE and EWVAVE (computed in mainline

code during initial mode determination) to produce the North and East velocity estimates, IDDVN and IDDVE. These velocities are then converted to earth coordinates and integrated to form the inertially derived latitude (IDDLAT) and longitude (IDDLON). The following equations are evaluated:

$$\begin{aligned} \text{DVN} &= \text{DVN} + \text{K2P DPN} \\ \text{IDDVN} &= \text{NSVAVE} + \text{DVN} \\ \text{IDDLAT} &= \text{IDDLAT} + (\text{IDDVN DTS} + \text{K1P DPN}) / \text{RMP} \\ \\ \text{DVE} &= \text{DVE} + \text{K2P DPE} \\ \text{IDDVE} &= \text{EWVAVE} + \text{DVE} \\ \text{IDDLON} &= \text{IDDLON} + (\text{IDDVE DTS} + \text{K1P DPE}) / \text{RNP} \end{aligned}$$

Where: K1P, K2P are the gains selected by HNAVSL,
DTS is a conversion factor from knots to degrees
per 50 msec iteration and
RMP, RNP are radii as described above.

If the JUMP flag (used for test purposes) is true, an error is introduced into the navigation output equal in magnitude and opposite in sign to the present guidance cross track error (XTK). The JUMP flag is then cleared.

$$\begin{aligned} \text{IDDLAT} &= \text{IDDLAT} + \sin(\text{TK}) \text{XTK} / \text{DLATFT} \\ \text{IDDLON} &= \text{IDDLON} + \sin(\text{TK}) \text{XTK} / \text{DLONFT} \end{aligned}$$

In the event that the above conditions of NAVFLG, MLSMOD and FLYFLG do not hold, the velocity estimates are set equal to NSVAVE and EWVAVE, DVN and DVE are zeroed and the integration of velocities into position is bypassed. (This is an error in that FLYFLG should only bypass the integration of position error into delta velocity, not the integration of velocity into position). Then, if the IRS signals are valid and LLINIT is false (indicating Lat/Lon not yet initialized by navigation or pilot input), the raw IRS latitude and longitude are set into IDDLAT and IDDLON. Also in this section, the altitude error integrator (DVH) is zeroed and, if FLYFLG is false, ALT and HDOT are initialized to HBARO and HDOTB, respectively.

The final section of HNAVB computes the inertially smoothed, baro-corrected altitude (IDDALT). If the simulated airplane is engaged (FLYFLG true), HDOT is set equal to HDCF (set by NAVIG), and operation of the altitude filter is bypassed, as HDDOT and ALT are also set directly by NAVIG. Otherwise, the vertical acceleration, velocity, and altitude are computed through a series of integrations. The vertical acceleration bias, DVH, is formed by integrating DELH, the difference between HBARO and ALT. Note that DELH is limited to 50 feet to prevent data dropouts from disturbing the filter. If a dropout occurs, the error counter (HBECTR) is also incremented. DVH is gained by XKA and summed with HDD (vertical acceleration) to form HDDOT, the vertical acceleration estimate in ft/sec/sec. Then, HDOT, the vertical velocity estimate, is formed by integrating the sum of HDDOT and XKV times

DELH. Finally, the inertially smoothed altitude (ALT) is computed by integrating the sum of HDOT and KD times DELH.

The final calculations produce IDDALT, the inertially smoothed altitude estimate, by applying a filtered, rate limited, barometric correction (FBARC) to ALT. The baro altitude correction in feet (BARSFT) is computed in the SLOW process whenever the pilot enters a new baroset via the CDU. FBARC is derived by taking the difference between BARSFT and the current value of FBARC, limiting that delta to a 10 foot maximum, and integrating. The rate limit is only applied when airborne (SQUAT is false), and serves to limit the rate of change of altitude due to baroset entries to 10 feet per second.

GLOBAL INPUTS: ALT, BARSFT, DISPST, DLATFT, DLONFT, DPE, DPN, DVE, DVN, FLYFLG, HBARO, HBECTR, HDCE, HDD, HDOT, HDOTB, IDDLAT, IDDLON, JUMP, K1P, K2P, LATINS, LLINIT, LONINS, MCONF, MLSLAT, MLSLON, MLSMOD, MLSSLI, MLSVLD, NAVFLG, RMP, RNP, SQUAT, XTK

GLOBAL OUTPUTS: ALT, DPE, DPN, DVE, DVN, HBECTR, HDDOT, HDOT, IDDALT, IDDLAT, IDDLON, MLSC, MLSMOD

MODULE NAME: HNSWIT (Horizontal/Vertical Navigation Switching)
 FILE NAME: HNAVFS.FOR
 PROCESS: FMFAST

PURPOSE: To select and easy-on the navigation position, velocity and acceleration outputs.

CALLED BY: HNAVFS
 CALLING SEQUENCE: CALL HNSWIT
 CALLS TO: SQRT

DESCRIPTION:

This procedure evaluates the selected navigation mode (IXX, IDD, GPS, or MLS) and selects the appropriate source data for output as VN, VE, HDCF, LAT, LON and ALTCOR. The selected mode is indicated by NAVMOD for LAT/LON and NVVMOD for altitude. Values are: 0 = IXX, 1 = IDD (NAVMOD only), 2 = GPS, 3 = MLS. An easy-on is provided for the velocity switch and also for the altitude switch on NVVMOD. No easy-on is provided for the LAT/LON switch, as this is handled by the lateral control law.

Processing begins by setting the PVM_ENABL switch, which determines whether MLS or GPS altitude is to be used. Basically, IDDALT and HDOT will be the selected output except on final approach, when MLS or GPS derived altitude may be selected to ensure that 3D guidance will guide the aircraft onto the glideslope.

$$PVM_ENABL = LOCE \cdot (GSENG + (VERPTH \cdot (PFPA \leq -2.)))$$

IDDALT and HDOT are then preselected for output and processing continues by checking the modes in priority sequence. If MLSMOD is true, MLS derived position and velocity outputs will be used. A check is made for first pass conditions. If NAVMOD is not equal to 3, some other mode was active on the previous pass. DELVN and DELVE, used in the velocity easy-on, are set to the difference between the previous output velocities and the MLS derived velocities and NAVMOD is set to 3.

$$\begin{aligned} DELVN &= VN - MLSVN \\ DELVE &= VE - MLSVE \end{aligned}$$

The outputs of the MLS complementary filter (POSHAT, VELHAT and ACCHAT) are then loaded into the specific item names (XHAT... ZDDH) using the overlay vector PVHAT, where PVHAT(1) is equivalenced to XHAT. The effect is:

$$\begin{array}{ccc} [POSHAT] & [VELHAT] & [ACCHAT] \\ | & | & | \\ v & v & v \\ [XHAT, YHAT, ZHAT] & [XDH, YDH, ZDH] & [XDDH, YDDH, ZDDH] \end{array}$$

The along-track (VGSDOT) and across-track (XTACC) acceleration

outputs are then computed as follows:

$$\begin{aligned} \text{GSFPS} &= \text{sqrt}(\text{XDH}^2 + \text{YDH}^2) \\ \text{VGSDOT} &= (\text{XDH XDDH} + \text{YDH YDDH}) / \text{GSFPS} \\ \text{XTACC} &= (\text{YDH XDDH} - \text{XDH YDDH}) / \text{GSFPS} \end{aligned}$$

MLSVN and MLSVE are then loaded into SELVN and SELVE, respectively. These intermediate variables are used in the easy-on logic, ultimately appearing as VN and VE. MLSLAT and MLSLON are next set into LAT and LON and the vertical guidance mode logic is executed. If MLSSV(4) and PVM_ENABL are both true, MLSALT and MLSHDT are loaded into the intermediate variables SELPH and SELVH, respectively, to ultimately appear as ALTCOR and HDCF. MLSSV(4) indicates that the MLS DME is in precision mode (which guarantees best altitude accuracy) and PVM_ENABL is discussed above. If NVVMOD is not equal to 3, DELVH and DELPH are initialized and NVVMOD is set to 3.

$$\begin{aligned} \text{DELVH} &= \text{HDCF} - \text{MLSHDT} \\ \text{DELPH} &= \text{ALTCOR} - \text{MLSALT} \end{aligned}$$

If the conditions for a NVVMOD of 3 are not satisfied, the correct outputs are HDOT and IDDALT, for a NVVMOD of 0. If NVVMOD is not 0, a reversion from either MLS or GPS is indicated. DELVH and DELPH are initialized and NVVMOD is set to 0.

$$\begin{aligned} \text{DELVH} &= \text{HDCF} - \text{HDOT} \\ \text{DELPH} &= \text{ALTCOR} - \text{IDDALT} \end{aligned}$$

If MLSMOD is false, then the mode is GPS or IDD. In either case, VGSDOT and XTACC are loaded from IDDATK and IDDXTK, respectively. If GPNAV is true, indicating that GPS navigation has been selected on the CDU, HYBLAT and HYBLON will be used for LAT and LON. Use of GPS velocities depends on the validity checks performed by GPSPRC. If GPSSEL is 0 (indicating bad status), the IRS velocities, VNINS and VEINS, are loaded into SELVN and SELVE. Otherwise, HYBVN and HYBVE are used. If NAVMOD was not previously 2, or if there is a change in the GPS validity as indicated by GPSSEL, then DELVN and DELVE are initialized:

$$\begin{aligned} \text{If GPSSEL} &= 0, \\ &\text{DELVN} = \text{VN} - \text{VNINS} \\ &\text{DELVE} = \text{VE} - \text{VEINS} \\ \text{Else} \\ &\text{DELVN} = \text{VN} - \text{HYBVN} \\ &\text{DELVE} = \text{VE} - \text{HYBVE} \end{aligned}$$

In either case, NAVMOD is set to 2 and a check is made for use of GPS vertical guidance. If GPLNDV (indicating that GPS vertical accuracy is sufficient for autoland) and PVM_ENABL are both true, HYBALT is set into SELPH. If NVVMOD was not previously 2, DELPH is set to ALTCOR minus HYBALT, and NVVMOD is set to 2. HDOT continues to be the selected altitude rate reference.

If neither MLS nor GPS guidance is selected, the position reference is IDDLAT, IDDLON and IDDALT, but a check still needs to be made for the velocity reference. If radio navigation updates are not available for a period of time, a large position error can accrue. The process of correcting this position error can introduce significant velocity error. The NAVVLD flag maintained by HNAVFS indicates that R-nav updates are available and the solution has converged sufficiently that the IDD velocities are accurate. Also, if a change in NAVMOD occurred on this pass, the delta velocities are initialized:

```

if NAVVLD then
  SELVN = IDDVN
  SELVE = IDDVE
  if NAVMOD ≠ 1 then
    DELVN = VN - IDDVN
    DELVE = VE - IDDVE
    NAVMOD = 1
  endif
else
  SELVN = NSVAVE
  SELVE = EWVAVE
  if NAVMOD ≠ 0 then
    DELVN = VN - NSVAVE
    DELVE = VE - EWVAVE
    NAVMOD = 0
  endif
endif

```

NVVMOD is also checked to see if a reversion to mode 0 occurred on this pass, and the delta altitude and altitude rate is initialized if necessary:

```

if NVVMOD ≠ 0 then
  DELVH = HDCF - HDOT
  DELPH = ALTCOR - IDDALT
  NVVMOD = 0
endif

```

Finally, a check is made to see if a change of NAVMOD occurred on this iteration ($\text{NAVMOD} \neq \text{PNVMOD}$), and the lateral velocity easy-on timer (TDWAIT) is initialized to 1.0 if so. Then if TDWAIT is greater than zero it is decremented by .005 (1 / 10 seconds times 20 Hertz), and TDWAIT times DELVN and DELVE is summed with SELVN and SELVE to produce the output velocities VN and VE. Otherwise SELVN and SELVE are simply loaded into VN and VE and TDWAIT is zeroed.

Identical processing is then performed for NVVMOD, with HDWAIT, DELVH and DELPH controlling the easy-on of SELVH and SELPH.

GLOBAL INPUTS: ALTCOR, GPLNDV, GPNAV, GPSSEL, GSENG, HDCF, HDD,
 HDOT, HYBALT, HYBLAT, HYBLON, HYBVE, HYBVN, IDDALT,

IDDATK, IDDLAT, IDDLON, IDDXTK, INSVAL, JUMP, K1P, K2P,
LATINS, LLINIT, LOCE, LONINS, MCONF, MLSALT, MLSLAT,
MLSLON, MLSMOD, MLSSLI, MLSSV, MLSVLD, NAVMOD, NVVMOD,
PFPA, POSHAT, RUNM, VE, VEINS, VERPTH, VN, VNINS

GLOBAL OUTPUTS: ALTCOR, GSEPS, HDCF, LAT, LON, PVHAT, VE, VGSDOT,
VN, XDDH, XDH, XHAT, XTACC, YDDH, YDH, YHAT, ZDDH, ZDH,
ZHAT

MODULE NAME: HVGUID (Horizontal/Vertical Guidance)
 FILE NAME: HVGUID.FOR
 PROCESS: FMFAST

PURPOSE: To provide the horizontal/vertical path errors and the speed commands for automatic aircraft guidance in 2D/3D/4D modes.

CALLED BY: FMFAST
 CALLING SEQUENCE: CALL HVGUID
 CALLS TO: AAA, DTG1, GD3D, HVG2, HVG6, LEGSW, TRALCBA, UVC, VDP

DESCRIPTION:

The Horizontal/Vertical guidance equations, implemented in procedure HVGUID and its attendant subprocedures, compute the deviations between the 2, 3, or 4D path defined by the path definition equations and the present aircraft speed and position computed by the navigation equations. Its primary outputs are: crosstrack error (XTK), desired track (DSRTK) and track error (TKE) used by the lateral steering equations, and altitude error (HER), desired altitude rate (HDTC) and flight path angle (PFPA) used by the vertical steering equations. The desired speed on the path (SDC) and the required along track acceleration (SDD) used by TIMPTH (4D) and speed modes are also computed here, although it would be more logical to compute them in TGUID. A detailed theoretical discussion of the algorithms implemented will be found in Appendix B, Horizontal / Vertical Guidance Computations. The discussion here will concentrate on implementation methodology.

Processing begins by checking for initialization conditions. If COLDST is set, indicating the first few iterations after power up, a few statements are executed which are, in fact, superfluous. Subsequently, the '2D guidance possible' flag (GUID2D) is checked. If it is false, control passes to the bottom of the HVGUID subroutine where certain lateral axis flags and variables are zeroed, and subroutine GD3D is called to initialize variables for the vertical axis. In every case, the last step is to set the present value of GUID2D into GD2DP.

If GUID2D is true, a check is made to see if ground speed is less than 64 knots (NAV64K = false) or if GUID2D was false on the previous pass (GD2DP = false). If either case prevails, further initialization is performed. The 2D guidance pointer (PTR2D) is limited to a minimum of two (any waypoint but the first may be selected as the 'to' waypoint), and the 4D guidance pointer (PTR4D) and the 'to waypoint' pointer (TOWPT) are set equal to PTR2D. The HVG first pass flag (HVGPl) is then cleared, which will cause some additional initialization to occur on the next iteration. Next, the unit normal vector (U12C) is loaded from the guidance buffer (WPT_ACT.NMV). This needs to be done here as it will not otherwise be done on a straight leg segment until PTR2D is updated. The lateral guidance variables are then initialized and GD3D is called to initialize the vertical guidance outputs. (Note: LATSTR should not be set here as it is no longer computed in this module).

As soon as NAV64K and GD2DP are both true (this will be the next iteration, assuming the path was entered or modified in flight), normal guidance processing begins by computing the aircraft position unit vector (P0).

$$\hat{P}_0 = \begin{bmatrix} \text{SLAT} \\ -\text{SLON} * \text{CLAT} \\ \text{CLON} * \text{CLAT} \end{bmatrix}$$

Where: SLAT = sin(Latitude), CLON = cos(Longitude), etc.

The abeam point unit vector is then computed by subtracting the component of P0 which is perpendicular to the path segment (i.e., parallel to the path normal vector) from P0 as follows:

$$\text{TMPVEC} = \hat{P}_0 - (\hat{P}_0 \cdot \text{NMV}) * \text{NMV}$$

TMPVEC will be loaded into POP by subroutine HVG2 when on a straight leg. POP locates the point on the great circle path abeam of present aircraft position. It is equal to P0 if the A/C is on the path.

When in a turn, both POP and U12C will be recomputed in subroutine AAA for turn guidance, but they are needed here in every event to compute the westward pointing vector, MHAT.

$$\hat{M} = \begin{bmatrix} 0 \\ \text{TMPVEC}(3) \\ -\text{TMPVEC}(2) \end{bmatrix}$$

MHAT is used in subroutine TRALCBA to compute the desired track.

Next the HVG first pass flag (HVGP1) is checked. If set, the 'in a DME-arc turn' flag (DMA) is cleared (LEGSW will set the PDMA flag equal to it) and LEGSW is called to do the normal path segment initialization. The leg distance to go (DTOGO) is then set equal to DTG (computed by LEGSW) and HVGP1 is set true, indicating initialization complete. HVGUID then exits, leaving the calculation of the actual guidance variables to the next pass. Note that this sequence requires three iterations after GUID2D is set to produce guidance inputs to the steering command modules, which causes problems there. With reorganization it should be possible to produce usable outputs on the very first iteration, thus allowing the removal of 'fix-it' code in LATCMD and VERCMD.

The remaining processing varies according to whether the path segment is a curved (TURN = true) or straight leg. Note that the 'in turn' (TURN) and '2nd half of turn' (TEND) flags serve a dual purpose. When transitioning from one straight path segment to another, TURN and TEND are set on consecutive iterations, then cleared on the third to mechanize the 2/3/4 -D pointer update logic. When actually guiding around a turn, TURN is set when entering the turn, TEND is set and PTR4D is updated when halfway

through the turn, and both are cleared and PTR2D is updated when the end of the turn is reached. In the special case of exiting from a DME-arc turn (which has a waypoint on each end as opposed to one in the middle), TURN is set again on the very next iteration to mechanize the transition to the next straight leg. If GUID3D is true, PTR3D is updated at least 300 feet before entering the turn (by subroutine HVG6). The actual distance depends on the degree (if any) of flight path angle change between the path segments. If GUID3D is false, PTR3D is synchronized to PTR4D by subroutine GD3D.

If TURN is false the following processing occurs:

Distance to go on a straight leg (DTG) is checked against the distance from the next waypoint back to the tangent point (DTT, the beginning of a (non-DME) turn), or simply to zero if approaching a DME turn. If DTG is less than DTT or if DTG is less than the distance that will be traveled in the next iteration (GSFPS * DELTAT) and if PTR2D is less than the total number of waypoints on the path (ACTCNT), TURN is set. This will initiate a path segment update by causing the 'TURN is true' logic to be exercised below. However if PTR2D equals ACTCNT, we have reached the end of the path. GUID- 2D, 3D and 4D are cleared and PTR2D is reset to 1 (a hint to other modules that path end has been reached). If it is not time to set TURN, HVG2 is called to compute DTG and DTOGO and TRALCBA is called to compute the lateral guidance errors and the speed commands.

If TURN is true the following processing occurs:

If PDMA is false, subroutine AAA is called to calculate the vectors required for turn guidance and the 'turn angle made good' (AMG). (Note: PDMA is the value of DMA on the previous leg, set by LEGSW during path leg initialization. If it is true with TURN true, we are in the process of doing the second pointer update at the end of a DME turn, which has nothing to do with 'turn' guidance. MAGTA and AMG will both be zero in this case). If TEND is false, AMG is checked to see if it is time to set it. If AMG is greater than or equal to 1/2 the magnitude of the turn angle (MAGTA), TEND is set, distance made good (DMG, used by time (4D) guidance) is increased by the center-to-center distance from the previous waypoint to the center of this turn (WPT_ACT.CCD) and PTR4D is incremented. If PTR4D then equals HLD_PTR, we are at the last waypoint of a holding pattern. If EXHOLD is false, we want to repeat it, so PTR4D is decremented by four and PTR3D is set equal to PTR4D. As a holding pattern is flown with a constant altitude and speed, but the initial inbound leg to the holding pattern may have varied both, the ground speed in the 'from' waypoint (WPT_ACT(PTR4D-1).GS) is set equal to that in the 'to' waypoint and the flight path angle in the 'to' waypoint (WPT_ACT(PTR3D).FPA) is set to zero. Regardless of the state of TEND, DTG1 is next called to compute distance to go (DTOGO - see the discussion under DTG1) and TRALCBA is called to compute the lateral guidance and speed commands.

If TEND was initially true (already in the second half of the turn or second step of a leg update sequence), then AMG is checked again. If AMG is less than MAGTA, we are still in the turn, and DTG1 and TRALCBA are called as above. Otherwise, PTR2D is incremented. If PTR2D now equals HLD_PTR and EXHOLD is false, PTR2D is decremented by four as was done above for PTR4D. U12C is then loaded from the guidance buffer. (Note: the above sequence causes problems when repeating a holding pattern which is not parallel to the inbound leg. U12C should be loaded before backing up PTR2D). LEGSW is then called to initialize the invariant parameters for the next leg, and the flag DOHVG6 is cleared to block HVG6 for one iteration (see below).

HVGUID processing completes by calling HVG6 if 3D processing is possible (GUID3D and DOHVG6 true with PDMA false). (The DOHVG6 flag was added to prevent a spurious altitude error on exiting a DMA turn.) HVG6 computes the vertical guidance parameters, HER, HDTC and PFPA, and updates PTR3D. DOHVG6 is set on return. If GUID3D is false, GD3D is called to maintain the vertical guidance variables in the IC state and keep PTR3D synchronized with PTR4D.

GLOBAL INPUTS: ACTCNT, AMG, CLAT, CLON, COLDST, DMG, DTG, EXHOLD, GSFPS, GUID2D, GUID3D, NAV64K, PTR2D, PTR4D, SLAT, SLON, TEND, TURN, WPT_ACT

GLOBAL OUTPUTS: ALCBA, ALCFLG, DMG, DTOGO, GUID2D, GUID3D, GUID4D, LATSTR, PTR2D, PTR3D, PTR4D, TEND, TKE, TOWPT, TURN, WPT_ACT, XTK

MODULE NAME: LEGSW (Leg Switching)
FILE NAME: HVGUID.FOR
PROCESS: FMFAST

PURPOSE: Set up invariant parameters for the next leg of the path.

CALLED BY: HVGUID
CALLING SEQUENCE: CALL LEGSW
CALLS TO: ANGL

DESCRIPTION:

The TURN and TEND flags are cleared (completing the leg update sequence), ALCFLG and ALCBA are cleared (if required on the next leg, they will be recomputed by TRALCBA) and the angle-made-good (AMG) is cleared. Parameters are then loaded from the guidance buffer entry pointed to by PTR2D as described below:

If DMA is false, turn angle (TA) is set to $WPT_ACT(i).ANGLE$ and DTG is set to the distance from the end of the previous turn to the next waypoint ($WPT_ACT(i).PPD - WPT_ACT(i - 1).DTT$). If DMA is true, these parameters refer to the DME turn being exiting, not an upcoming turn, and TA and DTG are set to zero. The sign (SIGNTA) and magnitude (MAGTA) of TA are then computed, PDMA is set to DMA and the DMA flag is loaded from $WPT_ACT(i).DMA$. The radius of turn (RTN) is then loaded from $WPT_ACT(i).RAD$ and the half-arc distance (AO2) is loaded from $WPT_ACT(i).ARC2$.

GLOBAL INPUTS: PTR2D, WPT_ACT

GLOBAL OUTPUTS: ALCBA, ALCFLG, AMG, DTG, RTN, TEND, TURN

MODULE NAME: GD3D (3D Guidance Initialization)
FILE NAME: HVGUID.FOR
PROCESS: FMFAST

PURPOSE: Initialize 3D guidance variables.

CALLED BY: HVGUID
CALLING SEQUENCE: CALL GD3D
CALLS TO: None

DESCRIPTION:

GD3D is called when GUID3D, GD2DP, or HVGP1 is false. It sets VERSTR, PFPA, HER, and DMG to zero, sets HDTC to present vertical speed (HDCF), synchronizes PTR3D to PTR4D and sets the nominal leg flight path angle (FPA) to WPT_ACT.FPA (this is only significant on the last call before setting HVGP1). FPA and HDTC will be required by HVG6 to ensure a smooth transition in vertical guidance should GUID3D become true, and PFPA is referenced by external modules without reference to the state of GUID3D. VERSTR should not be set here as it is no longer computed by HVGUID.

GLOBAL INPUTS: HDCF, PTR4D, WPT_ACT

GLOBAL OUTPUTS: DMG, FPA, HDTC, HER, PFPA, PTR3D, VERSTR

MODULE NAME: AAA (Turn Vector Processing)
 FILE NAME: HVGUID.FOR
 PROCESS: FMFAST

PURPOSE: Compute the unit normal and abeam point vectors used for guidance when in a turn.

CALLED BY: HVGUID
 CALLING SEQUENCE: CALL AAA
 CALLS TO: ATAN2D, UVC, VCP, VDP

DESCRIPTION:

The 'abeam point' when in a turn is simply the center of the turn, so P0P is loaded from WPT_ACT.TCV. (This must be done every iteration as HVGUID mainline processing continues to compute P0P as for straight segment processing). U12C, the unit vector normal to the plane defined by the airplane position vector (P0) and P0P, is then computed by subtracting from the difference between P0 and P0P, the component of that vector in the turn center vector direction:

$$U12C = ((P0 - P0P) - ((P0 - P0P) \cdot P0P) * P0P) * SIGNTA$$

Where: SIGNTA is positive for a right turn.

UVC is then called to ensure that the result is a unit vector.

The angle made good (AMG) is calculated as follows:

$$AMG = \text{atan}(\sin(A), \cos(A))$$

$$\text{Where: } \sin(A) = (U12C \times N\hat{M}V) \cdot P0P$$

$$\cos(A) = U12C \cdot N\hat{M}V$$

N $\hat{M}V$ is the unit vector normal to the inbound leg.

Finally, AMG is corrected according to its magnitude and the turn direction to make sure that it remains \geq zero and \leq 330 degrees (the maximum permissible turn magnitude). Also, if the aircraft is sufficiently off course as to cross on the wrong side of the turn center (indicated by XTK times the sign of the turn angle being greater than the turn radius), AMG is set equal to MAGTA, causing immediate update to the next path leg.

GLOBAL INPUTS: RTN, WPT_ACT, XTK

GLOBAL OUTPUTS: AMG

MODULE NAME: DTG1 (Distance-to-go (Turn) Calculations)
FILE NAME: HVGUID.FOR
PROCESS: FMFAST

PURPOSE: Calculate the abeam distance, DTOGO, according to position in and type of turn.

CALLED BY: HVGUID
CALLING SEQUENCE: CALL DTG1
CALLS TO: None

DESCRIPTION:

DTG1 calculates the distance-to-go (DTOGO) when in a turn. When in a DME-arc turn, it is simply the distance around the turn (CCD) minus the arc distance made good (AMG). For non-DME turns, it is the distance from the beginning to the center of the turn (AO2) minus the arc distance made good when in the first half of the turn, and the distance to the center of the next turn (CCD to the next waypoint is added) when in the second half of a turn.

GLOBAL INPUTS: AMG, PTR2D, RTN, TEND, WPT_ACT

GLOBAL OUTPUTS: DTOGO

MODULE NAME: HVG2 (Distance-to-go (Straight Leg) Calculations)
 FILE NAME: HVGUID.FOR
 PROCESS: FMFAST

PURPOSE: Compute DTG and DTOGO when on a straight leg.

CALLED BY: HVGUID
 CALLING SEQUENCE: CALL HVG2
 CALLS TO: ATAN2, VCP, VDP

DESCRIPTION:

HVG2 is called when on a straight leg. It loads the abeam point vector (POP) from TMPVEC (computed by HVGUID mainline code) and computes two measures of 'distance to go'. DTG is the distance from the abeam point to the next waypoint (WPV), computed as:

$$DTG = Re * \text{atan}(\sin(A), \cos(A))$$

Where: Re is the local Earth radius in feet (RADFT)

$$\begin{aligned} \sin(A) &= (POP \times WPV) \cdot U12C \\ \cos(A) &= POP \cdot WPV \end{aligned}$$

and

POP is the abeam point vector

WPV is waypoint vector of the to waypoint

U12C is the unit normal vector (= WPT_ACT.NMV

when on a straight leg)

If the 'to' waypoint is the inbound waypoint of a DME-arc turn, or if there is no turn at the next waypoint, then DTOGO = DTG. Otherwise, DTOGO is the distance to the center of the next turn, computed as:

$$DTOGO = DTG - DTT + AO2$$

Where: DTT is the distance from the next waypoint back to the tangent point of the turn;
 AO2 is one half the distance around the turn.

AO2 is loaded from the guidance buffers by LEGSW.

GLOBAL INPUTS: RADFT, WPT_ACT

GLOBAL OUTPUTS: DTG, DTOGO

MODULE NAME: TRALCBA (Primary Lateral Guidance Computations)
 FILE NAME: HVGUID.FOR
 PROCESS: FMFAST

PURPOSE: Calculate NOMBA, XTK, DSRTK and TKE. Ground speed mode
 SDC and SDD are also calculated here.

CALLED BY: HVGUID
 CALLING SEQUENCE: CALL TRALCBA
 CALLS TO: ANGL, ATAN2, ATAN2D, SCOSD, SIND, VCP, VDP

DESCRIPTION:

The nominal bank angle (NOMBA) required to compensate for lateral acceleration around a turn is computed as:

$$\text{NOMBA} = \arctan (\text{GSFPS}^2 / (g * \text{RTN}))$$

Where: g is the nominal acceleration due to gravity

RTN is the radius of the turn in feet

and If $g * \text{RTN}$ is $< \text{GSFPS}^2$, NOMBA is set to 45 degrees.

Then RALC, the distance before the turn to apply the nominal bank angle in order to minimize tracking error, is calculated as

$$\text{RALC} = \text{GSFPS} * \text{NOMBA} / 4$$

Where: 4 degrees per second is the lateral control law roll rate limit in auto mode.

If ALCFLG (the flag controlling application of NOMBA to the control law) is true, the exit angle (ALCXA), or number of degrees before the end of turn to remove NOMBA is calculated as

$$\text{ALCXA} = \text{RTOD} * \text{RALC} / \text{RTN}$$

Where: RTOD is the radian to degree conversion constant

Then, if the angle made good (AMG) is greater than or equal to the magnitude of the turn angle minus the exit angle, ALCFLG is set false and ALCBA (the signed bank angle command applied) is zeroed. Otherwise, ALCBA is set to NOMBA times the sign of the turn angle.

If ALCFLG is false and the half-arc distance of the turn (AO2) is greater than RALC (if the turn angle is large enough to require ALCBA), then RALC is compared to the distance to the start of the turn. If the turn is a DME-arc turn, this is simply distance to go (DTG). Otherwise it is DTG minus the distance to the tangent point (DTT). If this distance is less than RALC, ALCFLG is set true.

The remaining code is bypassed during the second update at the

end of a DME-arc turn (if PDMA is true). Otherwise, processing continues by calculating cross track distance to the planned flight path (XTK) as follows:

$$XTK = R_e * \text{atan}(\sin(A), \cos(A))$$

Where: R_e is the local radius of the Earth (RADFT)

$$\sin(A) = \hat{P}_0 \cdot \hat{U}_{12C}$$

$$\cos(A) = \hat{P}_0 \cdot \hat{P}_{OP}$$

\hat{P}_0 is the airplane position vector

\hat{P}_{OP} is the abeam point vector (points to the center of turn if in a turn)

\hat{U}_{12C} is the unit normal vector. XTK is measured parallel to \hat{U}_{12C}

XTK is positive when to the right of the path.

If in a turn (TURN is true), XTK as calculated above is the distance to the center of the turn, and must be corrected by adding the radius of turn times the sign of the turn angle to produce the distance to the path:

$$XTK = XTK + RTN * \text{SIGNTA}$$

The desired track (DSRTK) is then calculated as follows:

$$DSRTK = \text{atan}(\sin(A), \cos(A))$$

$$\text{Where: } \sin(A) = (\hat{U}_{12C} \times \hat{M}) \cdot \hat{P}_{OP}$$

$$\cos(A) = \hat{U}_{12C} \cdot \hat{M}$$

\hat{M} points West from \hat{P}_{OP}

The track angle error (TKE) is computed as the actual minus the desired track angle (TK - DSRTK) and SCOSD is called to calculate the sine and cosine of TKE (STKE, CTKE).

The remaining code computes DTOTL (used by time guidance) and the nominal ground speed (SDC) and acceleration commands (SDD) used for time guidance (4D) and ground speed modes, as well as the ground speed error (GSE) that may be displayed on line 10 of the CDU. The 'to waypoint' pointer (TOWPT) which always points to the next waypoint ahead of the aircraft is also computed. Initially, TOWPT is set to PTR4D and DTOTL is computed as the distance made good (DMG) plus the center-to-center distance (CCD) to the waypoint pointed to by TOWPT. If in the first half of a DME-arc turn, PTR4D will still be pointing to the inbound waypoint of the turn, so

TOWPT is incremented and DTOTL is increased by the distance around the turn (CCD to the outbound waypoint). If the 'speed mode possible' flag (SPDF) in the waypoint 'TOWPT' is false or if DTOTL is less than zero (not yet beyond the first waypoint on the path), SDC and SDD are zeroed. Otherwise, SDD is computed as the desired speed at the 'to' waypoint (.GS) minus the speed at the previous waypoint all over the center-to-center distance between them. SDC is then computed as the speed at the 'to' waypoint minus SDD times the distance to go (DTOGO). GSE (in knots) is then equal to GSFPS minus SDC, divided by the knots-to-feet/second conversion factor (KTOFPS).

GLOBAL INPUTS: ALCFLG, AMG, DMG, DTG, DTOGO, GSFPS, GSFPS2,
PTR4D, RADFT, RALC, RTN, TEND, TURN, WPT_ACT

GLOBAL OUTPUTS: ALCBA, ALCFLG, CTKE, GSE, DSRTK, DTOTL, SDC, SDD,
STKE, TKE, TOWPT, XTK

MODULE NAME: HVG6 (Primary Vertical Guidance Computations)
 FILE NAME: HVGUID.FOR
 PROCESS: FMFAST

PURPOSE: Calculate the vertical guidance parameters.

CALLED BY: HVGUID
 CALLING SEQUENCE: CALL HVG6
 CALLS TO: None

DESCRIPTION:

The local index (I) is set to PTR3D + 1. If I is then equal to HLD_PTR and EXHOLD is false (indicating that holding pattern repeat is selected) I is reset to PTR4D (which was adjusted in HVGUID mainline code). Then, if PTR3D is less than the total number of waypoints in the path (ACTCNT), the nominal flight path angle for the next segment on the path (FPAN) is set from WPT_ACT(I).FPA. If PTR3D is equal to ACTCNT, FPAN is set to zero, as end-of-path will cause a reversion to altitude hold mode.

The distance before the waypoint to initiate the segment update (HDIS) is then calculated as the difference between the present and next segment flight path angles times the ground speed squared all over 2.

$$HDIS = \text{abs}(FPA - FPAN) * GSFPS^2 / 2.$$

This is the distance required to make an asymptotic capture of the next vertical path segment at an average vertical acceleration of 2 fps per second. A nominal minimum advance distance of 300 feet is imposed. If the distance to go to the end of the path leg (DTCGO) is less than HDIS, PTR3D is set to 'I' and FPA is set to FPAN. This update also occurs if PTR3D is less than PTR4D, (which should never occur, but has been known to happen during a path update).

Next, the commanded altitude at the abeam point (HC) is calculated as the altitude at the 'to' waypoint minus the distance to the end of the path segment times the flight path angle of the segment in radians.

$$HC = \text{WPT_ACT}(TOWPT).ALT - DTCGO * \text{WPT_ACT}(TOWPT).FPA * DTOR$$

The pointer TOWPT is used here as it always points to the upcoming waypoint. Altitude error (HER) is then the difference between HC and the present altitude (ALTCOR). The nominal vertical speed (HDCR) is calculated as the ground speed in feet per second times FPA times the cosine of the track angle error (CTKE). This value is passed through a rate limited lag to produce the vertical speed command HDTC. The rate limit imposed is 2.0 feet per second per second with a lag of .125 sec. HER and HDTC are the inputs to the VERCMD procedure used to compute the vertical steering command.

The planned flight path angle (PFPA) is computed as HDTC divided by ground speed in feet per second (GSFPS) times the radian to degree conversion factor (RTOD). PFPA is (may be) used to drive the GAMMA wedges on the Primary Flight Display and is also used to compute the flight path angle error for display on the CDU.

GLOBAL INPUTS: ACTCNT, ALTCOR, CTKE, DTOGO, EXHOLD, FPA, GSFPS,
GSFPS2, HDTC, HLD_PTR, PTR3D, PTR4D, WPT_ACT

GLOBAL OUTPUTS: FPA, HDTC, HER, PFPA, PTR3D

MODULE NAME: TGUID (Time Guidance)
FILE NAME: TGUID.FOR
PROCESS: FMFAST

PURPOSE: To compute the speed command for the time path (4D) mode.

CALLED BY: FMFAST
CALLING SEQUENCE: CALL TGUID
CALLS TO: AAT, CDG

DESCRIPTION:

TGUID and its subroutines AAT and CDG, together with some code in the HVGUID procedure, compute the longitudinal acceleration command (SCMD) for the time guidance mode. HVGUID computes the nominal ground speed (SDC) for the aircraft at its present position, the average acceleration along the present path leg (SDD), the distance made good (DMG) along the path and DTOTL, which is DMG plus the length of the present leg minus the distance to go to the next waypoint. TGUID computes the position of the timebox on the path and differences that with the aircraft position to find the separation distance (SEPR) which is used (together with SDC, SDD and present ground speed) to compute the acceleration required to drive SEPR to zero.

Processing begins by checking for time guidance possible. If GUID4D is false or if NAV64K is false (the airplane must still be on the ground), SEPR is set to zero, the time guidance first pass flag (TGP1) is cleared and processing terminates. Otherwise, HER and XTK are checked for non-zero values to ensure that HVGUID has completed its initialization. If this check passes, TGP1 is checked and, if false, initialization is performed.

During initialization the 4D turn flags are cleared, the 'distance made good' for the time box (DMG1) and for the airplane (DMG) are zeroed and the time box 4D reference pointer (PTR4D1) is initialized to 2. The guidance buffer is then searched for an entry having an arrival time (ETA) greater than the present time. PTR4D1, PWVADR and TBOXPTR are initialized to point to this entry and TOWTPTR is set equal to the aircraft 4D pointer set by HVGUID (PTR4D). (Note: the 'velocity pointer' (PWVADR) serves no useful purpose in this implementation, but is a carry over from the original code where it was useful. TBOXPTR and TOWTPTR are only used during the initialization process). If the airplane is ahead of the timebox (TOWTPTR is greater than TBOXPTR), DMG is increased by the center to center distance (CCD) of the leg indicated by TBOXPTR and TBOXPTR is incremented. If the timebox is ahead of the airplane, DMG1 is increased by the CCD indicated by TOWTPTR and TOWTPTR is incremented. This is continued until TBOXPTR equals TOWTPTR. TGP1 is then set and the average path acceleration in fps/sec (SDDC) is calculated by differencing the speed in the waypoints at the beginning and end of the leg the timebox is presently on (as indicated by PWVADR), divided by the nominal time for the leg.

$$SDDC = \frac{KTOFPS * (WPT_ACT(PWVADR).GS - WPT_ACT(PWVADR-1).GS)}{WPT_ACT(PTR4D1).TIME}$$

The initial timebox velocity in fps (SDCC) is then computed as the speed at the 'to' waypoint minus SDDC times the desired time to this waypoint.

$$TOTIME = WPT_ACT(TBOXPTR).ETA$$

$$SDCC = KTOFPS * WPT_ACT(PWVADR).GS - SDDC * (TOTIME - TIME)$$

The distance from the timebox to the 'to' waypoint is then calculated in one of two ways, depending on whether the initial velocity is positive or negative:

for SDCC < zero:

$$SC = (SDCC/SDDC + TOTIME - TIME) * KTOFPS * WPT_ACT(PWVADR).GS / 2.$$

I.e., the time for SDCC to become positive plus the time to the waypoint, all times the average speed on the path. The timebox remains frozen at the beginning of the path until SDCC becomes positive.

for SDCC >= zero:

$$SC = (TOTIME - TIME) * (KTOFPS * WPT_ACT(PWVADR).GS + SDCC) / 2.$$

If the 'to' waypoint is not the start of a DME arc turn, SC is adjusted by adding the distance to the tangent point and subtracting the half arc distance to the midpoint of the turn (if any). (This is apparently done to facilitate calculation of timebox position in a turn. Since the drawing of the timebox itself is handled entirely by the displays computer in the present implementation, this code could be simplified considerably).

$$SC = SC + WPT_ACT(PTR4D1).DTT - WPT_ACT(PTR4D1).ARC2$$

If the 'from' waypoint is the inbound waypoint of a DME turn, the box is in the second half of the turn. TURN1 and TEND1 are set, the magnitude of the turn angle (MAGTA1) is set from WPT_ACT(PTR4D1 - 1).ANGLE, the turn radius (RT) is set from WPT_ACT(PTR4D1).RAD and the arc-distance made good (ADMG) is set to WPT_ACT(PTR4D1).CCD - SC. Subroutine AAT is then called to complete turn processing, CDG is called to compute the speed command and processing terminates until the next pass. In all other cases, MAGTA1 is set from WPT_ACT(PTR4D1).ANGLE. Then, if SC is less than the distance to the tangent point, TURN1 is set and ADMG and RT are set as above. (Two lines, depending on the state of TEND1, are coded but since TEND1 was cleared at the beginning and has not been set, only the line that sets RT from WPT_ACT(PTR4D1).RAD is meaningful). AAT is then called to perform the turn processing and CDG is called (at

the end of this procedure) to compute the speed command.

On subsequent passes, processing begins by integrating SDDC into SDCC. If SDCC is then positive, DELTAT * SDCC is subtracted from SC. If the timebox is not presently in a turn (TURN1 is false), the end of the straight leg is checked for. This is indicated when SC becomes less than zero or less than the distance to the tangent point. If, when this occurs, PTR4D1 is equal to the number of waypoints on the path (ACTCNT), the end of the path has been reached, so GUID4D is cleared, PTR4D1 is set to 1 and processing terminates. Otherwise, if this is the start of a DME arc turn (the DMA flag is set in the 'to' waypoint buffer), SDDC, SDCC and SC are reset as given:

$$SDDC = \frac{KTOFPS * (WPT_ACT(PWVADR).GS - WPT_ACT(PWVADR-1).GS)}{WPT_ACT(PTR4D1).TIME}$$

$$SDCC = KTOFPS * WPT_ACT(PWVADR).GS - SDDC * (TOTIME - TIME)$$

$$SC = WPT_ACT(PTR4D1+1).CCD$$

Where: PWVADR is set to PTR4D1 + 1.

If the DMA flag is set in the 'from' waypoint, this must be the second update at the end of a DME turn, so MAGTA1 is set to zero. Otherwise, MAGTA1 is set to WPT_ACT(PTR4D1).ANGLE. RT is set to WPT_ACT(PTR4D1).RAD, the arc-distance made good is zeroed, TURN1 is set and AAT and CDG are called.

GLOBAL INPUTS: ACTCNT, DMG, GUID4D, HER, NAV64K, PTR4D1, SC, SDCC, TEND1, TIME, TURN1, WPT_ACT, XTK

GLOBAL OUTPUTS: ADMG, DMG, GUID4D, PTR4D1, SC, SDCC, SEPR, TEND1, TOTIME, TURN1

MODULE NAME: AAT (Turn Computations)
 FILE NAME: TGUID.FOR
 PROCESS: FMFAST

PURPOSE: To perform timebox calculations when in a turn.

CALLED BY: TGUID
 CALLING SEQUENCE: CALL AAT
 CALLS TO: None

DESCRIPTION:

Unconditional processing consists of integrating the time box velocity (SDCC) into the arc-distance made good (ADMG), then computing the angle made good (AMG1) as ADMG divided by the turn radius (RT) times the radian to degree conversion factor. If the timebox is already in the second half of the turn (TEND1 is true), a check is made for end of turn: if AMG1 is equal to or greater than the magnitude of the turn (MAGTA1), TURN1 and TEND1 are set false and processing terminates. Otherwise, a check is made for the halfway point: if $2 * AMG1$ is equal to or greater than MAGTA1, TEND1 is set true and the timebox distance-made-good (DMG1) is updated by the leg center to center distance (CCD). If DMG1 is less than the aircraft distance made good (DMG), DMG is reduced by DMG1 and DMG1 is set to zero. Otherwise DMG1 is reduced by DMG and DMG is set to zero. (This is done to maintain DMG and DMG1 at the minimum values required to correctly compute the separation distance, thus improving computational accuracy.) PTR4D1 is then incremented. If PTR4D1 is then equal to HLD_PTR and EXHOLD is false, the time box has just completed a circuit of a holding pattern that is still in effect. To repeat the pattern, PTR4D1 is set to PTR4D1 - 4. (This needs to be checked. The timebox should probably skip a holding pattern altogether, or at best repeat it only once.) The 'velocity pointer' (PWVADR) is then set equal to PTR4D1.

If not in a DME-arc turn, the timebox acceleration, velocity and distance to go (SDDC, SDCC, SC) are then reset as given (this is done in mainline code for a DME turn):

$$SDDC = \frac{KTOFPS * (WPT_ACT(PWVADR).GS - WPT_ACT(PWVADR-1).GS)}{WPT_ACT(PTR4D1).TIME}$$

$$SDCC = KTOFPS * WPT_ACT(PWVADR-1).GS$$

$$SC = WPT_ACT(PTR4D1).CCD$$

If the new 'to' waypoint is not the entry to a DME-arc turn, SC is corrected by adding the distance to the tangent point (DTT) and subtracting the half arc distance at the next turn (ARC2).

GLOBAL INPUTS: ADMG, DMG, EXHOLD, HLD_PTR, PTR4D1, SDCC, TEND1,
WPT_ACT

GLOBAL OUTPUTS: ADMG, DMG, PTR4D1, SC, SDCC, TEND1, TURN1

MODULE NAME: CDG (Primary, Time Guidance Computations)
 FILE NAME: TGUID.FOR
 PROCESS: FMFAST

PURPOSE: To compute the acceleration command (SCMD) for time guidance.

CALLED BY: TGUID
 CALLING SEQUENCE: CALL CDG
 CALLS TO: None

DESCRIPTION:

The timebox distance to go (SC) is set into DTOG01. If the 'to' waypoint is not a DME-arc turn, DTOG01 is then corrected by subtracting the distance to the tangent point and adding the half-arc distance to the midpoint of the next turn (if any). The timebox equivalent of the aircraft total distance made good (DTOTL) is then computed in the variable SEPR by first setting SEPR to the time box distance made good (DMG1) plus the center-to-center distance (CCD) to the guidance buffer entry pointed to by PTR4D1. If this waypoint is the entry to a DME-arc turn and TURN1 is true with TEND1 false, the timebox is in the first half of that turn, and this entry is behind us. In that case SEPR is increased by the arc-distance made good in the turn (ADMG). In all other cases, this entry is still ahead of the timebox and SEPR is decreased by DTOG01. The separation distance between the aircraft and the timebox (SEPR) is then computed as 'SEPR' minus DTOTL. Obviously some names should be changed here to avoid confusing the innocent. The time error is then computed as the negative of SEPR divided by the average of the timebox and the aircraft speed, with a negative timebox velocity ignored:

$$\text{TIMERR} = -\text{SEPR} / ((\text{SDC} + \max(0., \text{SDCC})) / 2)$$

Where: SDC is the nominal speed on the path;
 SDCC is the time box velocity (limited to positive values);

The time path acceleration command (SCMD) is then computed as the sum of the nominal path acceleration (SDD) + .0025 * SEPR, limited to one percent of the nominal velocity, plus 0.1 times the difference between SDC and the present groundspeed:

$$\text{SCMD} = \text{SLIM}((\text{SDD} + .0025 \text{ SEPR}), .01 \text{ SDC}) + 0.1 (\text{SDC} - \text{GSFPS})$$

SCMD is then attenuated as the maximum CAS or MACH is approached, or set to -.25 if either maximum is exceeded. (This code is not necessary, as overspeed protection is applied elsewhere, nor is it completely correct as coded). Finally, the resultant SCMD is limited to a maximum of +/-1.0 fps/sec. Note: The above equation is erroneous in that the nominal timebox acceleration (SDDC) should be used vice SDD.

GLOBAL INPUTS: ADMG, CAS, DTOTL, GSFPS, MACH, SC, SDC, SDCC,
SDD, TURN1, TEND1, WPT_ACT

GLOBAL OUTPUTS: SCMD, SEPR, TIMERR

MODULE NAME: NAVIG (Simulated Airplane)
 FILE NAME: NAVIG.FOR
 PROCESS: FMFAST

PURPOSE: To permit ground checkout of several dynamic functions of
 the Flight Management/Flight Controls software.

CALLED BY: FMFAST
 CALLING SEQUENCE: CALL NAVIG
 CALLS TO: ENGAGE_CAS, FLYIC

DESCRIPTION:

NAVIG is the executive program for the point mass model simulated airplane. It's operation is controlled via the utility program VIEW, by setting bits in the simulation control word SIMFLG. The actions initiated by setting different bits in SIMFLG are enumerated below:

- o Bit 0 - Initialize airplane
- o Bit 1 - Fly airplane @ 200 knots
- o Bit 2 - Fly Calibrated Air Speed or 4D path
- o Bit 3 - Not used
- o Bit 4 - Hold airplane
- o Bit 5 - Not used
- o Bit 6 - Not used
- o Bit 7 - Force VOR2 = CVOR2 for navigation testing (HNAVSL)
- o Bit 8 - Force DME2 = CDME2 for navigation testing (HNAVSL)
- o Bit 9 - Force DME3 = CDME3 for navigation testing (HNAVSL)
- o Bit 10-12 - Not used
- o Bit 13 - Index to APDAT (Sim initialization data)
- o Bit 14 - Index to APDAT (Sim initialization data)
- o Bit 15 - Set up Sim initialization values per bits 13, 14

Upon entry, the IC and RUN discrettes are checked to see if a change in Flight Controls operate mode was selected. If IC and LABFLG are both true, SIMFLG is set to HEX 0011 (reinitialization). If RUN and RUNM are both true, SIMFLG is set to 0. SIMFLG is then checked for a negative value (Bit 15 set). If so, bits 13 and 14 are used as an index into the APDAT matrix to set up the IC values for the airplane, and SIMFLG is set to HEX 0011.

If bit 0 of SIMFLG is set, which signifies initialization of the airplane, an initialization counter (FLYICF) is set to -32 if it was zero. For the next 32 iterations, the counter is incremented, subroutine FLYIC is called to perform initialization, calibrated air speed (CAS) is set to 2 knots, the Ground-speed-greater-than-four-knots flag (NAVFLG) is cleared and subroutine ENGAGE_CAS is called to set up the remaining variables required for operation of the flight software. (This is a no-op, since ENGAGE_CAS does nothing until the hold and ic bits are cleared from FLYFLG). On the 32nd iteration, bit 0 of SIMFLG is cleared

(allowing further processing to proceed) and the guidance possible flags GUID2D and GUID3D are forced true.

If any one of the bits (other than bit 0), that control the simulated airplane is set the following occurs:

- o FLYFLG and AUTOE are set true signifying that the simulated airplane is now in operation.
- o If the hold bit is set, CAS is set to 2 knots and NAVFLG is turned off.
- o If the hold bit is off, a check is made to determine if the CAS bit (bit 2) is set. If so, CAS is computed from the longitudinal acceleration command (ATCMD) and the ground speed in feet per second variable (GSFPS). Otherwise, CAS is set to a constant 200 knots, or the value stored in SIMCAS.

Procedure ENGAGE_CAS is then called to set up all other required variables.

GLOBAL INPUTS: ATCMD, CAS, GSFPS, IC, LABFLG, RUN, RUNM, SIMCAS,
SIMFLG

GLOBAL OUTPUTS: AUTOE, CAS, FLYFLG, GUID2D, GUID3D, NAVFLG,
SIMALT, SIMCAS, SIMFLG, SIMHDG, SIMLAT, SIMLON, SMMAGV

MODULE NAME: FLYIC (Initialization)
FILE NAME: NAVIG.FOR
PROCESS: FMFAST

PURPOSE: To perform basic initialization for the simulated airplane.

CALLED BY: NAVIG
CALLING SEQUENCE: CALL FLYIC
CALLS TO: None

DESCRIPTION:

This procedure initializes the value of the fast loop navigation (HNAVFS) position integrators, IDDLAT, IDDLON and ALT, to the values found in SIMLAT, SIMLON and SIMALT, respectively. The Control Display Unit latitude/longitude initialization flag LLINIT is then set to prevent these variables from being initialized by HNAVFS. HDOT complementary filter (HDCF) is set to zero and magnetic variation (MAGVAR) is set to the value stored in SMMAGV. (FLYFLG set prevents these from being recomputed by HNAVFS or ERAD, respectively, but TAS also needs to be set here to work properly with the present version of HNAVFS). The integrator for simulated airplane heading (SMUHDG) is initialized to the value found in SIMHDG and the simulation roll command (PHISYM) is set to zero. The 2D, 3D and 4D guidance flags (GUID2D, GUID3D, GUID4D) are turned off, auto engage (AUTOE) is turned off and the 2D guidance pointer (PTR2D) is set to 1.

GLOBAL INPUTS: SIMALT, SIMHDG, SIMLAT, SIMLON, SMMAGV

GLOBAL OUTPUTS: ALT, AUTOE, GUID2D, GUID3D, GUID4D, HDCF, IDDLAT, IDDLON, LLINIT, MAGVAR, PHISYM, PTR2D, SMUHDG

MODULE NAME: ENGAGE_CAS (Operate Simulator)
 FILE NAME: NAVIG.FOR
 PROCESS: FMFAST

PURPOSE: To set the simulated state variables for the simulated airplane into the global locations for the flight software.

CALLED BY: NAVIG
 CALLING SEQUENCE: CALL ENGAGE_CAS
 CALLS TO: ANGL, TAN

DESCRIPTION:

If the hold or initialize bits are set in SIMFLG, an immediate return is made to NAVIG. Otherwise the following variables are computed:

- o The simulated roll command (PHISYM) is computed as

$$\text{PHISYM} = .0625 * (\text{BACMD} - \text{PHISYM}) + \text{PHISYM} \quad (.77 \text{ sec lag})$$

PHISYM is then stored into the bank angle (ROLL);

- o TAS is set equal to CAS and MACH is forced to 0.2;

- o The heading for the simulated airplane (SMUHDG) is computed as

$$\text{SMUHDG} = \text{ANGL}(\text{SMUHDG} + (0.5 * \text{TMP1} / \text{TMP2}))$$

Where: $\text{TMP1} = 1.91 * \text{PHISYM}$

$$\text{TMP2} = \text{MAX}(\text{CAS}, 100)$$

SMUHDG is set into HDGTRU by HNAVFS;

- o The cross track acceleration (IDDXTK) is computed as

$$\text{IDDXTK} = \text{IDDXTK} + \text{DELTAT} * (\text{GRAV0} * \tan(\text{PHISYM}) - \text{IDDXTK});$$

- o The vertical acceleration (HDDOT) is set to the vertical acceleration command (VACMD);

- o The HDOT complementary filter output (HDCF) is computed as

$$\text{HDCF} = \text{HDCF} + \text{DELTAT} * \text{VACMD};$$

- o The inertial altitude (ALT) is computed as

$$\text{ALT} = \text{ALT} + \text{DELTAT} * \text{HDCF}.$$

NAVIG specific code in HNAVSL performs the additional adjustments necessary to use these outputs in lieu of data input by IOFLL.

GLOBAL INPUTS: ALT, BACMD, CAS, HDCE, IDDXTK, PHISYM, SIMFLG,
SMUHDG, VACMD

GLOBAL OUTPUTS: ALT, HDCE, HDDOT, IDDXTK, MACH, PHISYM, ROLL,
SMUHDG, TAS

5.2 SLOW LOOP PROCESSING:

MODULE NAME: CDUEXC (CDU Executive)
FILE NAME: CDUEXC.FOR
PROCESS: SLOW

PURPOSE: Executive for the Control Display Unit Slow loop procedures.

CALLED BY: SLOW
CALLING SEQUENCE: CALL CDUEXC

DESCRIPTION:

A detailed description of CDUEXC and the attendant subroutines involved in CDU processing is found in the document

NASA Contractor Report 189606

Advanced Transport Operating System (ATOPS)
Control Display Unit Software Description

MODULE NAME: BLOW (Wind Computations)
 FILE NAME: BLOW.FOR
 PROCESS: SLOW

PURPOSE: Calculate wind speed and direction.

CALLED BY: SLOW
 CALLING SEQUENCE: CALL BLOW
 CALLS TO: SQRT, ATAN2, SCOSD

DESCRIPTION:

If the radio altitude (HRAD) is less than five feet, the wind direction is not computed and wind speed is set to zero. Otherwise, two separate wind speed models are computed. The equations are as follows:

EQUATION 1:

OLDWD = atan2 (X, Y)
 OLDWS = sqrt (X² + Y²)

Where: X = SINTH * TASGS - VE
 Y = COSTH * TASGS - VN
 VE = velocity east in knots
 VN = velocity north in knots
 SINTH = sin of true heading
 COSTH = cos of true heading
 TASGS = true airspeed corrected for gamma

Equation 1 is from the original C-4000 software. It appears accurate when flying straight and level, but indicates erroneous winds when the aircraft is maneuvering.

EQUATION 2:

DFVWD = atan2 (X, Y)
 DFVWS = sqrt (X² + Y²)

Where: X = S * ASPD - VE
 Y = C * ASPD - VN
 ASPD = TAS * CTHET
 S = sin (HDGTRU-TMPA*SROLL)
 C = cos (HDGTRU-TMPA*SROLL)
 TMPA = ALFAV or ALW - 1, depending on
 the setting of Bit 0 of MCONF

The above equations were derived from the DFVLR 4D Guidance experiment. In the lab it appears free of the erroneous indications during maneuvering, but has never been adequately tested in the air.

Setting WNDMOD to 1, 2 or 3 (via VIEW) allows the operator to select one of the 2 wind models or just accept the IRS computations. The selected values (1: OLD, 2: DFV, 3: IRS) are set

into WS and WD.

GLOBAL INPUTS: ALFAV, ALW, COSTH, CTHET, HDGTRU, HRAD, IRSWD,
IRSW, MCONF, NAV64K, SINTH, SROLL, TAS, TASGS, VE, VN,
WNDMOD

GLOBAL OUTPUTS: DFVWD, DFVWS, OLDWD, OLDWS, WD, WS

MODULE NAME: EPRLMT (Engine Pressure Ratio Limit)
 FILE NAME: EPRLMT.FOR
 PROCESS: SLOW

PURPOSE: To calculate the maximum engine pressure ratio (EPR) for the Boeing 737 aircraft jet engine.

CALLED BY: SLOW
 CALLING SEQUENCE: CALL EPRLMT
 CALLS TO: None

DESCRIPTION:

EPRLMT produces maximum Engine Pressure Ratio (EPR) limits for climb (MCLEPR), cruise (MCREPR), and continuous thrust (MCTEPR). Depending upon the setting of EPRFLG (pilot selected via the CDU), one of these is selected as the maximum EPR (MXEPR) for display and for use by the auto throttle control law (ATHCL).

The program is organized in two logical sections, for engine bleed-air on, and for engine bleed-air off. If either engine has bleed-air on, the equations for bleed-air on are used (resulting in lower EPR limits). The calculations consider the effect of static air pressure (STAT_PRES), total air temperature (TAT), and altitude (ALT). A first order approximation is used to calculate the maximum EPR for each possible case. Constants for the approximations were supplied by Boeing for the JT8D-7 engine and are documented in the equations below.

Static pressure is first calculated as a function of barometric altitude:

$$AF = ALT / 145499.6$$

$$STAT_PRES = 29.92 (1.0 - AF (5.258 - AF (11.9 - 12.16 AF)))$$

If airbleed is on (the path taken if airbleed is on for either engine), the following equations are evaluated:

$$MXEPRT = 1.94 - .00628 TAT$$

if (ALT > 30000) then

$$AF = (ALT - 30000.) / 5000.$$

$$MXEPRT = MXEPRT - AF (.025 + .00039 TAT)$$

endif

$$MXEPRP = 3.514 - .0535 STAT_PRES$$

Where: MXEPRT is max EPR as a function of temperature;
 MXEPRP is max EPR as a function of pressure.

$$MCLEPR = MCTEPR = MCREPR = \min(MXEPRP, MXEPRT, 2.35)$$

```

if (1500. < ALT < 30000.) then
  if (ALT > 18500.) then
    MCTEPR = MCLEPR
  else
    if (TAT < 17.5) then
      MXEPRT = 2.03 - .00545 TAT
    else
      MXEPRT = 2.08 - .008375 TAT
    endif
    MXEPRP = 3.56 - .0547 STAT_PRES
    MCTEPR = min(MXEPRP, MXEPRT, 2.35)
  endif

  if (ALT > 10000.) then
    if (TAT < -20.) MCREPR = 1.8568 - .00591 TAT
  elseif (TAT > 20.)
    MCREPR = 1.764 - .00571 TAT
  else
    MCREPR = 1.8125 - .008125 TAT
  endif

endif

```

If airbleed is off, the following equations are evaluated:

```

MXEPRT = 1.97 - .0068 TAT
MXEPRP = 3.589 - .0549 STAT_PRES

MCLEPR = MCTEPR = min(MXEPRP, MXEPRT)

if (TAT < -20.) then
  MXEPRT = 1.8568 - .00591 TAT
elseif (TAT > +20.) then
  MXEPRT = 1.843 - .00714 TAT
else
  MXEPRT = 1.855 - .00775 TAT
endif

MCREPR = min(MXEPRP, MXEPRT)

if (1500. < ALT < 18500.) then
  if (TAT >= 17.5) then
    MXEPRT = 2.053 - .00538 TAT
  else
    MXEPRT = 2.11 - .00853 TAT
  endif
endif

GAEPR = MCTEPR

if (EPRFLG = 1) then

```

100

```
    MXEPR = MCLEPR
elseif (EPRFLG = -1) then
    MXEPR = MCREPR
else
    MXEPR = MCTEPR
endif
```

GLOBAL INPUTS: ABLOFF, ABROFF, ALT, EPRFLG, TAT.

GLOBAL OUTPUTS: MCLEPR, MCREPR, MCTEPR, MXEPR.

MODULE NAME: ERAD (Earth Radii Computations)
 FILE NAME: ERAD.FOR
 PROCESS: SLOW

PURPOSE: To compute the radii of curvature in the East-West and North-South planes.

CALLED BY: SLOW
 CALLING SEQUENCE: CALL ERAD
 CALLS TO: ANGL, MAG_VAR

DESCRIPTION:

If the simulated airplane is engaged, then magnetic variation (MAGVAR) has been set from the simulator data tables (SMMAGV). If not, then MAGVAR is set by ERAD as follows:

If IRS navigation is valid (INAVV true), MAGVAR is computed as the difference between IRS true heading (THDG) and magnetic heading (MAGHDG). Otherwise, the function MAG_VAR is invoked to compute the local MAGVAR based on present latitude (LAT) and longitude (LON).

The local North (RM) and East (RN) radii of curvature (using the WGS84 Earth model) and several related variables used by the navigation and guidance procedures are then set by evaluating the following equations:

$$\begin{aligned} RN &= \text{RADIUS} (1. + \text{ELLIP} * \sin^2(\text{LAT})) \\ RNP &= \text{CLAT} (\text{RN} + \text{WGSALT FTONM}) \\ RM &= \text{RADIUS} (1. - 2. \text{ELLIP} + 3. \text{ELLIP} \sin^2(\text{LAT})) \\ RMP &= \text{RM} + \text{WGSALT FTONM} \end{aligned}$$

Where: RADIUS is the nominal Earth radius in nautical miles (taken as 3443.9186 NM).

ELLIP is the eccentricity (3.3528107E-3)
 SLAT is the sine of the present latitude.
 CLAT is the cosine of the present latitude.
 WGSALT is the present altitude corrected for the height of the local mean sea level above the reference geoid.

$$\begin{aligned} \text{RADFT} &= \text{NMTFT} (\text{RM} + \text{RN}) / 2. + \text{WGSALT} \\ \text{DLATFT} &= \text{RMP NMTFT DTOR} \\ \text{DLONFT} &= \text{RNP NMTFT DTOR} \end{aligned}$$

Where: RADFT is the best-fit local Earth radius in NM.
 DLATFT is the number of feet per degree of LAT.
 DLONFT is the number of feet per degree of LON.
 DTOR is the conversion from degrees to radians.

102

GLOBAL INPUTS: ALTCOR, CLAT, FLYFLG, INAVV, LAT, LON, MAGHDG,
 SLAT, THDG

GLOBAL OUTPUTS: DLATFT, DLONFT, MAGVAR, RADFT, RM, RMP, RN, RNP

MODULE NAME: HNAVSL (Horizontal Navigation Slow Loop)
 FILE NAME: HNAVSL.FOR
 PROCESS: SLOW

PURPOSE: To manually or automatically tune and monitor the path defined and cross track stations and use information from these stations and/or an ILS installation to create aircraft position updates for use by fast loop navigation.

CALLED BY: SLOW
 CALLING SEQUENCE: CALL HNAVSL
 CALLS TO: ANGL, CRBSC, GET_REAL, RADCAL, SIN, SIND, SQRT, TAN, TUNPTH, TUNXTK

DESCRIPTION:

The radio update navigation routine (HNAVSL) acts as the executive for navigation updates. It calls the routines that autotune (or manually tune) the VORTAC's as path-defined or cross-path stations. Once suitable stations have been selected, HNAVSL performs validity checks and calls routines to compute range and bearing from the stations. The difference between the measured position and the estimated position is weighted according to the navigation mode (Inertial with radios, Air Data mode with radios, or radios only). These weighted differences are output for use in updating the fast loop position and velocity estimates. In addition to VORTAC's, ILS may be utilized if the airplane is in the correct zone. The ILS derived bearing of the airplane from the runway center line may be used in conjunction with a DME distance to provide position error estimates. ILS may also be used as the sole navaid in the runway area.

The acronym for the CDU display (IDD, IDX, etc.) is created according to the navigation mode being used and the combination of stations tuned. This navigation mode acronym is also displayed on the Navigation display. MLS and GPS inputs may also be used for navigation, but this is accomplished entirely in foreground processing. HNAVSL sets up the appropriate navigation mode display, but continues to concern itself with selection and tuning of standard navaids.

HNAVSL is logically partitioned into the following sections:

Initialization code:

Packed error flag words F0, F2, F3, DMEER, and VORER to be used in the current HNAVSL pass are cleared. Local variables (SLLAT, SLLON) are assigned the most recent position estimate from fast loop navigation and local copies (DME3, DME2, VOR2, SLOCDV, SGSDEV) are made of the most recent navaid input data. The Boolean flag, MLSFLG, is set if MLSMOD is true or if MLS navigation updates have been selected via the nav pallet switch (MLSSLI) and MLSVLD is true. Local subroutine RADCAL is then called to compute the airplane position vector, PSVECT.

Station tuning:

The local subroutine TUNPTH and the external subroutine TUNXTK are next called to select and tune the path (DME2) and cross-path (DME3) stations, respectively. Failures are recorded in FALST2/FALST3. On return, additional checks are performed to ensure that valid signals are being received from the selected stations, with errors recorded in DMEER and VORER as appropriate. Ultimately, the error summary is logged in F0, F2 and F3 (for DME3, DME2 and VOR2, respectively).

Position error computations:

Processing begins with the cross station, DME #3. If bit 9 of the simulated airplane control word (SIMFLG) is set, DME3 is set equal to the computed range (CDME3) and the station validity flag (DME3VD) is ignored. Otherwise, bit 13 is set in DMEER if DME3VD is false. Bit 14 of DMEER is set if any bits were set in FLSTA3 by TUNXTK. Next, the basic filter gain (DELGN) is set to 0.5 if either the Air Data (FLADM) or Radio (FLRM) mode flag is set and otherwise set to 0.25. The magnitude of the measured range is then checked. If DME3 is not greater than zero (indicating invalid data), bit 11 is set in DMEER, and if it is greater than 200 nautical miles (nm), bit 12 is set. Next, if the selected station address (NVAD3A) is zero (no station selected), CDME3 is zeroed. Otherwise, local subroutine CRBSC is called to compute the range and bearing of the station. (Note: the bearing is not used for DME #3). CRBSC also computes the cosine of the depression angle (COSDA) and the sine and cosine of the computed bearing (SINCB, COSCB). If the computed range is greater than 200 nm, bit 10 is set in DMEER, or if CDME3 was forced to zero, bit 7 is set. Otherwise, DRANGE is computed as (CDME3 - DME3) times COSDA. If this delta ground range is greater than 5 nm, bit 9 of DMEER is set. Otherwise, the North and East position errors (elements 1 and 5 of the H matrix) are computed as follows:

$$\begin{aligned} H(1) &= DRANGE * COSCB * DELGN \\ H(5) &= DRANGE * SINCB * DELGN \end{aligned}$$

Additionally, bit 8 of F0 is set if the aircraft altitude is greater than the ground range to the station. This 'H > R' indication is a 'not broke but don't use' flag which inhibits navigation using this station, but does not cause retuning. (Note: this check as implemented works fine on the East coast, but would not be reliable in Denver. Some potential fixes would be: Subtract CRBEL from IDDALT before comparison; Check for COSDA > 0.707; Use TRANSV(3) (once fixed) for the comparison).

Finally, DMEER is OR'd into F0. If DMEER was zero, the station timer (TIMS1) is cleared and the 'data timeout' bit in FLSTA3 should be cleared. (Bit 12 ('no path station') is being cleared).

If DMEER was non-zero, TIMS1 is checked. If it was previously zero, it is set to the present time. Otherwise, a check is made to see if the error has existed for more than 15 seconds ((TIME - TIMS1) > 15). If so, TIMS1 is cleared and, if the bank angle

(ROLL) is less than 15 degrees, bit 14 is set in FLSTA3. Subroutine TUNXTK will check FLSTA3 on the next iteration and retune if necessary. The 'timeout' error is ignored when ROLL is greater than 15 degrees, as the data is not used under these conditions.

The path station (DME2) is processed next. Processing is the same as for the cross station except as noted below. The PTHSTA flag is set to enable additional processing in CRBSC. Provision is made for calculating position errors from measured versus calculated VOR2 bearing as well as from DME range, although the present hardware configuration does not permit input of VOR bearing data. The VORVLD bit is hardwired false in low level code (DISFD) to prevent its inadvertent use for navigation. Nevertheless, the simulator is set up to test the code. Bit 8 of SIMFLG is used for DME2 and bit 7 is used for VOR2. Errors detected by TUNPTH are logged in FLSTA2 and the error summary is stored in F2 (for the DME) and F3 (for the VOR). If no station has been selected (NVAD2A = 0), both calculated range (CDME2) and calculated bearing (CVOR2) are zeroed. Any DME errors are logged in both DMEER and VORER, as the VOR data is necessarily scaled by range to the station. Additional errors are logged in VORER for CDME2 > 150 nm (bit 12), delta bearing (DELBR) > 30 degrees (bit 9), and DELBR * ground range (GRMAG) > 5 nm (bit 11). If the DME data passes all checks, North and East position errors are computed as follows:

$$\begin{aligned} H(3) &= DRANGE * COSCB * DELGN \\ H(7) &= DRANGE * SINCB * DELGN \end{aligned}$$

If it were possible for the VOR data to pass all checks, DELBR would be converted to radians and the North and East position errors computed as follows:

$$\begin{aligned} H(4) &= TEMP1 * DELBR * TRANSV(1) * DELGN \\ H(8) &= TEMP1 * DELBR * TRANSV(2) * DELGN \end{aligned}$$

Where: TEMP1 is a VOR weighting gain computed as
 $.125 * (DMEMAX - DME2) / DMEMAX$,
 TRANSV is the transpose vector computed by
 CRBSC (measured in the local horizontal
 plane),
 DMEMAX is DME max range = 200 nm.

Finally, DMEER and VORER are OR'd into F2 and F3 and 'data time-out' is checked, using TIMS2 and the same error in clearing FLSTA2 as noted above for FLSTA3.

ILS processing:

Checks are made to see if the aircraft is within the appropriate range and bearing of a selected ILS station. If GPS navigation mode has been selected (NAVMOD = 2), RADCAL is first called to recalculate the airplane position vector using GPS position data. This is to preclude a possibly erroneous ILSZON calculation based on radio navigation data. If an airport has been

selected and it is within the delta lat/lon limits, CRBSC is called to calculate the range and bearing (RNGLS, BRGLS) from the localizer to the aircraft, and BRGLS is rotated into runway coordinates. If the range and bearing are within limits, the ILSZON flag (used by autoland control laws) is set, and if further criteria are met, North and East position errors are computed from the difference between the measured and estimated ILS bearing and (possibly) glideslope.

Position corrections perpendicular to runway heading are computed if neither MLS nor GPS navigation has been selected, the localizer is valid and the localizer deviation and aircraft delta track are within specified limits. The equations evaluated are:

$$\begin{aligned} \text{DLPP} &= \text{DTOR} * (\text{BRGLS} - \text{LOCDEV}) \\ \text{DLNPP} &= \text{DLPP} * \text{TRANSV}(1) \\ \text{DLEPP} &= \text{DLPP} * \text{TRANSV}(2) \end{aligned}$$

Where: TRANSV is the vector computed by CRBSC giving the location of the localizer in locally South, East and up coordinates in nautical miles (nm).

If no field DME is available and the aircraft is on the last leg of a 3D path coincident with the glideslope, position corrections along the runway centerline may be made using glideslope data. Two algorithms are coded. The second of these (which was the first coded and the only one which has been used in flight) computes both along runway and across runway corrections as follows:

$$\begin{aligned} \text{temp} &= \text{RWYLEN} + \text{HTDZ} / \tan(\text{GSA} + \text{GSDEV}) \\ \text{DLPP} &= (\text{XTK} + \text{temp} * \sin(\text{LOCDEV})) * \text{FTONM} \\ \text{DLNPP} &= \text{DLPP} * \text{SINRH} \\ \text{DLEPP} &= -\text{DLPP} * \text{COSRH} \\ \\ \text{temp} &= \text{HTDZ} * \text{GSDEV} / (\text{GSA} + \text{GSDEV}) \\ \text{DLPP} &= (\text{HER} + \text{temp}) * \text{FTONM} / \tan(\text{GSA}) \\ \text{DLNPP} &= 0.5 * (\text{DLNPP} + \text{DLPP} * \text{COSRH}) \\ \text{DLEPP} &= 0.5 * (\text{DLEPP} + \text{DLPP} * \text{SINRH}) \end{aligned}$$

Where: XTK is the 3D path crosstrack error (in feet)
HER is the 3D path altitude error (in feet)
HTDZ is height above ground level (in feet)

This algorithm works reasonably well, but the range estimate used to scale the beam errors is derived from radar altitude divided by nominal glide slope angle. This is subject to errors caused by terrain, altimeter scaling errors and the maximum altimeter range (1326 ft). In Autoland, this is masked by the fact that the beam errors are essentially zero. If the plane is being manually flown, however, these scaling errors can cause navigation errors. The other algorithm attempts to avoid this problem by using the range estimate (RNGLS) derived from CRBSC. The equations

are:

$$\begin{aligned} \text{DLPP} &= \text{XTK} * \text{FTONM} + \text{RNGLS} * \sin(\text{LOCDEV}) \\ \text{DLNPP} &= \text{DLPP} * \text{SINRH} \\ \text{DLEPP} &= -\text{DLPP} * \text{COSRH} \end{aligned}$$

$$\begin{aligned} \text{DLPP} &= (\text{HER} + \text{DTOGO} * \sin(\text{GSDEV})) * \text{FTONM} / \text{TANGSA} \\ \text{DLNPP} &= 0.5 * (\text{DLNPP} + \text{DLPP} * \text{COSRH}) \\ \text{DLEPP} &= 0.5 * (\text{DLEPP} + \text{DLPP} * \text{SINRH}) \end{aligned}$$

Where: RNGLS is range to the localizer antenna (in nm)
DTOGO is distance to last waypoint (GPIP) (in ft)

Position correction computations:

The accumulated status bits are checked and the best combination of available navigation aids is selected to produce the position corrections (DPN and DPE) to fast loop navigation (HNAVFS).

Note: the following errors exist. If DME3 is good and DME2 is bad, an attempt is made to use DME_VOR navigation. The correct VOR to use would be VOR3, but F3 is checked, which is the status word for VOR #2. (Confusing, isn't it?). As F3 always shows bad status, this mode is never entered. In perfect consistency, F1 (the status word for VOR #3) is checked when DME2 is good and DME3 is bad, although VOR2 would be the correct VOR to use. This CAN cause a problem because F1 is never set to anything. It therefore shows good status (0) for the non-existent VOR3. Should this condition occur, the data selected would be H4 and H8, which is the data for the (correct) VOR #2. This data is always zero (since VOR2 is forced invalid), so the effective navigation mode is IXD or single DME, which does not yield reliable updates. Current plans are to correct these problems in a future release of the software.

The circular error is then computed and, if greater than 0.5 nm, the error flag, DPERRF, is set. If greater than 2.0 nm, it is limited to 2.0 to reduce perturbation of the filter. The filter gains (K1P, K2P) are then selected based on the available velocity reference (INS, Air data or Radio) and the nav aids in use (DME, ILS or MLS).

Navigation mode indicators:

An ASCII character string (NAV TYP) and a coded mode word (MNAV TY) are set up to drive indications on the CDU and the Navigation Display, which advise the pilot as to the navigation data source, and thus (by implication) the probable navigation accuracy. Potential modes (arranged in hierarchical order) are as follows:

AMX: MLS mode (MLS data used directly for navigation)
IMX: MLS nav source (MLS drives radio nav filter, which is then used for navigation)
DGP: Differential GPS with good status
AGP: Absolute GPS with good status

GPn: Degraded GPS tracking 'n' satellites
 GXX: GPS selected but not valid

The following codes may begin with 'I' (Inertial velocity reference), 'A' (Air data velocity reference) or 'X' (Radio mode only). Ixx is the only code likely to be seen.

ILD: ILS localizer and DME #2
 ILG: ILS localizer and glideslope
 ILX; ILS localizer only (No along track correction)
 IDD: DME #3 and DME #2 (Inertial-Dual-DME)
 IVD: VOR #2 and DME #2 (Presently not possible)
 IDV: DME #3 and VOR #3 (Also not possible)
 IDX/IXD: Single DME is tuned and valid. No navigation updates are computed (except as noted above).
 IXX: No valid DME, or bank angle greater than 15 degrees. No navigation updates are computed.

HNAVSL completes its processing by copying the local failure flags, F0, F2 and F3, into the global flags, F0G, F2G and F3G.

GLOBAL INPUTS: ACTCNT, AIRPTS, ANTLAT, ANTLON, CDME2, CDME3, COSRH, CVOR2, DIFMOD, DTOGO, DME2VD, DME3VD, DVE, DVN, FLADM, FLRM, FLSTA2, FLSTA3, GPNVAV, GPNVAVV, GPSSEL, GSA, GSDEV, GSVLD, HER, HRAD, HTDZ, IDDALT, IDDLAT, IDDLON, ILDPREF, ILDHIG, ILGRNG, ILSZON, LAT, LON, LOCDEV, LOCVLD, MAGVAR, MDME2, MDME3, MLSMOD, MLSSLI, MLSVLD, MVOR2, NAV64K, NAVMOD, NVAD2A, NVAD3A, PTR2D, ROLL, RWYHDG, RWYLEN, RYELEV, SATINVW, SIMFLG, SINRH, TANGSA, TIME, TK, VORVLD, WPT_ACT, XTK

GLOBAL OUTPUTS: BRGLS, CDME2, CDME3, CVOR2, DPE, DPERRE, DPN, DVE, DVN, F0G, F2G, F3G, ILSZON, K1P, K2P, MNAVTY, NAVCTR, NAVTYP, RNLGS

MODULE NAME: RADCAL (Earth Radius Calculations)
 FILE NAME: HNAVSL.FOR
 PROCESS: SLOW

PURPOSE: Calculate the aircraft position vector, PSVECT.

CALLED BY: SLOW
 CALLING SEQUENCE: CALL RADCAL
 CALLS TO: COSD, SIND

DESCRIPTION:

RADCAL uses SLLAT and SLLON (the local copies of the aircraft position estimate calculated by HNAVFS), aircraft altitude (ALTCOR) and the Earth model parameters (RADIUS, ELLIP and ESQ) to compute PSVECT, the ellipsoidal Earth vector from the center of the Earth to the aircraft position. The equations evaluated are as follows (simplified form).

$$\text{temp} = \text{RADIUS} * (1. + \sin^2(\text{LAT}) * \text{ELLIP}) + \text{ALTCOR} * \text{FTONM}$$

$$\begin{aligned} \text{PTVECT} = & \text{temp} * \sin(\text{LAT}) * (1. - \text{ESQ}) \\ & - \text{temp} * \sin(\text{LON}) * \cos(\text{LAT}) \\ & \text{temp} * \cos(\text{LON}) * \cos(\text{LAT}) \end{aligned}$$

Where: RADIUS is the nominal earth radius (3443.9186 nm),
 ELLIP is the ellipticity (3.3528107 E-3), and
 ESQ is the eccentricity squared (6.6943801 E-3).
 All quantities are per the WGS-84 Earth model.

GLOBAL INPUTS: ALTCOR, SLLAT, SLLON

GLOBAL OUTPUTS: None

MODULE NAME: CRBSC (Range and Bearing Calculations)
 FILE NAME: HNAVSL.FOR
 PROCESS: SLOW

PURPOSE: Compute the slant range, ground range and bearing from the aircraft to a VOR, ILS or DME.

CALLED BY: SLOW
 CALLING SEQUENCE: CALL CRBSC(RANGE, BRG)
 CALLS TO: ATAN2D, COSD, MXV, SIND, SQRT, VMG

DESCRIPTION:

If the path station flag (PTHSTA) is true, DPH11 and DLAM1 (the delta latitude and longitude between the aircraft and the path station) are calculated and the PTHSTA flag is cleared. DPH11 and DLAM1 are used by TUNXTK to calculate the wedge angle between the path station and the cross station.

CRBSC then uses CRBLT, CRBLG and CRBEL (the station latitude, longitude and elevation set up by HNAVSL), and the Earth model parameters (RADIUS, ELLIP and ESQ) to compute PTVECT, the ellipsoidal Earth vector from the center of the Earth to the navaid. DVECT is computed by subtracting PTVECT from PSVECT (the vector from the center of the Earth to the aircraft, computed by RADCAL). SRMAG (slant range to the navaid) is then computed as the length of this vector. The equations evaluated are as follows (simplified form).

$$\text{temp} = \text{RADIUS} * (1. + \sin^2(\text{CRBLT}) * \text{ELLIP}) + \text{CRBEL} * \text{FTONM}$$

$$\begin{aligned} \text{PTVECT} = & \text{temp} * \sin(\text{CRBLT}) * (1. - \text{ESQ}) \\ & - \text{temp} * \sin(\text{CRBLG}) * \cos(\text{CRBLT}) \\ & \text{temp} * \cos(\text{CRBLG}) * \cos(\text{CRBLT}) \end{aligned}$$

Where: RADIUS is the nominal earth radius (3443.9186 nm),
 ELLIP is the ellipticity (3.3528107 E-3), and
 ESQ is the eccentricity squared (6.6943801 E-3).
 All quantities are per the WGS-84 Earth model.

$$\text{DVECT} = \text{PSVECT} - \text{PTVECT}$$

$$\text{SRMAG} = \text{VMG}(\text{DVECT}, 3)$$

Where: VMG is an external function which returns the magnitude of a vector of a specified length. I.e, the square root of the sum of the squares of the elements of the vector.

The transpose matrix (TVECT) is then computed and multiplied times DVECT to produce the transposed difference vector, TRANSV. The elements of TRANSV are the North, East and up coordinates of

the aircraft relative to the navaid in a locally horizontal plane. That is, a plane with its origin on the geoidal approximation of the Earth and directly under the aircraft. The equations evaluated are as follows:

$$\text{TVECT} = \begin{matrix} 0.0 & \cos(\text{LON}) & \sin(\text{LON}) \\ -\cos(\text{LAT}) & -\sin(\text{LAT}) \sin(\text{LON}) & \sin(\text{LAT}) \cos(\text{LON}) \\ \sin(\text{LAT}) & \cos(\text{LAT}) \sin(\text{LON}) & \cos(\text{LAT}) \cos(\text{LON}) \end{matrix}$$

$$\text{TRANSV} = [\text{TVECT}] [\text{DVECT}]$$

Note: The Z coordinate of TRANSV (TRANSV(3)) is incorrect because the sign of TVECT(3,2) is incorrect. The quantity $(-\cos(\text{LAT}) \sin(\text{LON}))$ yields the correct value. TRANSV(3) is not used in any calculation.

Finally, the ground range and bearing (magnetic) of the aircraft from the station (GRMAG, MAGBEAR), the sine and cosine of the (true) bearing (SINCB, COSCB) and the cosine of the depression angle (COSDA) are computed.

$$\text{GRMAG} = \text{sqrt} (\text{TRANSV}_1^2 + \text{TRANSV}_2^2)$$

$$\text{MAGBEAR} = \text{atan} (\text{TRANSV}_1, \text{TRANSV}_2)$$

$$\text{SINCB} = \text{TRANSV}_1 / \text{GRMAG}$$

$$\text{COSCB} = \text{TRANSV}_2 / \text{GRMAG}$$

$$\text{COSDA} = \text{GRMAG} / \text{SRMAG}$$

TRANSV₁ is then negated for use by HNAVSL in computing position errors from the calculated VOR or ILS angular errors (making this element positive south vice positive north).

GLOBAL INPUTS: SLLAT, SLLON

GLOBAL OUTPUTS: None

MODULE NAME: TUNPTH (Path Defined Station Tuning)
FILE NAME: HNAVSL.FOR
PROCESS: SLOW

PURPOSE: To select and tune a station for DME #2.

CALLED BY: HNAVSL
CALLING SEQUENCE: CALL TUNPTH
CALLS TO: NXTPS, T1CHEX, TUNDM2, TUNEPS

DESCRIPTION:

TUNPTH, together with its internal subroutines NXTPS, T1CHEX and TUNEPS, and the external subroutine TUNDM2, selects and tunes a navaid for DME #2 (the path station). It first checks to see if the station was manually tuned (indicated by the auto-tune flag, ATNAV2, being false and the station address, NVAD2A, being non-zero). If so, it checks the status word (FLSTA2). The check should be for 'station-not-tuned (bit 13) and timer expired' or 'station failed' (any other bit set in FLSTA2). The way the parentheses are grouped, the timer would be checked only if the 'station-not-tuned' bit is NOT set. If status is bad and ground speed is greater than 140 kts, the RETUN2 flag is set. (In the original NCDU code, this caused a message to be displayed on the CDU, but this feature was not implemented in the new CDU code. Had it been, we might have noticed that the logic used to set it is erroneous.) Next TUNPTH calls TUNEPS and T1CHEX to output the tuning code and verify that the station has properly tuned to the selected frequency.

If auto-tune is enabled or if no station has been selected, an attempt is made to automatically select a path station. If ground speed is less than 64 knots (NAV64K false), only the path defined station may be used. PTRSTA is set to the 'to waypoint' pointer (TOWPT) and NXTPS is called to fetch the path station address. If PTRPS is returned as a zero (indicating 'no path station'), bit 14 of F2 is set. (This is an error, but the effect is only to lose the record of the original failure, since any bit set in FLSTA2 causes bit 14 of F2 to be set in HNAVSL mainline code. Bit 12 of FLSTA2 should be set.) Otherwise, NVAD2A is set to PTRPS, TUNEPS is called to output the tuning code and T1CHEX is called to check the tuning, setting bit 13 of FLSTA2 if unsuccessful for more than four seconds. Next, FLSTA2 is set unconditionally to '1000' hex (bit 12 only). This is also an error. The effect is to disable radio navigation below 64 knots, while wiping out any real errors that may have been logged. Bit 12 should only be set as indicated above.

If NAV64K is true, full station search logic is enabled. The 'no path station' flag is cleared and the PSFAIL flag is set if any other errors were logged in FLSTA2 and the error is timed out. Next, a check is made to see if it's time to update the path defined station (done when halfway to the next waypoint). If the update has already been made (station pointer ahead of waypoint pointer), PSFAIL is tested and if true, TUNDM2 is called to find another station. TUNEPS and T1CHEX are then called as above.

If the station pointer is behind the waypoint pointer, it is set equal to the waypoint pointer and NXTPS is called to find the next path station. If the pointers are equal and the aircraft is not yet at the halfway point and PSFAIL is false, T1CHEX is called to verify tuning. Otherwise, PTRPS is checked. If it is non-zero and not equal to NVAD2A, NVAD2A is set to PTRPS and TUNEPS is called to output the new tuning code. If PTRPS is zero or equal to NVAD2A, T1CHEX is called if PSFAIL is false, otherwise the station pointer is updated and NXTPS is called. Then, if no next station is found (PTRPS = 0), TUNDM2 is called to search for any usable station. Otherwise, NVAD2A is set to PTRPS and TUNEPS is called to output the tuning code.

GLOBAL INPUTS: ATNAV2, DTOGO, FLSTA2, GS, NAV64K, NVAD2A,
 RETUN2, TOWPT

GLOBAL OUTPUTS: FLSTA2, NVAD2A

114

MODULE NAME: NXTPS (Next Path Station)
FILE NAME: HNAVSL.FOR
PROCESS: SLOW

PURPOSE: To get station defined by next waypoint.

CALLED BY: TUNPTH
CALLING SEQUENCE: CALL NXTPS
CALLS TO: None

DESCRIPTION:

NXTPS first checks GUID2D. If it is false, there is no path and thus no path station. PTRPS is returned as a zero. Otherwise, PTRPS is set equal to the station address found in the guidance buffer entry indicated by PTRSTA, and COP is computed as one half the center-to-center distance for the path segment pointed to by the waypoint pointer, TOWPT.

GLOBAL INPUTS: GUID2D, TOWPT, WPT_ACT

GLOBAL OUTPUTS: None

MODULE NAME: TUNEPS (Tune Path Station)
FILE NAME: HNAVSL.FOR
PROCESS: SLOW

PURPOSE: Tune the path station (DME #2).

CALLED BY: TUNPTH
CALLING SEQUENCE: CALL TUNEPS
CALLS TO: GET_WORD

DESCRIPTION:

If the path station address (NVAD2A) is non-zero, the tuning code for the navaid pointed to by NVAD2A is fetched from the navigation data base and placed in the tune output word (ATUNE2). The status word (FLSTA2) is then zeroed.

GLOBAL INPUTS: NVAD2A

GLOBAL OUTPUTS: ATUNE2, FLSTA2

116

MODULE NAME: T1CHEX (Tuning Checks)
FILE NAME: HNAVSL.FOR
PROCESS: SLOW

PURPOSE: Check path station tuning

CALLED BY: TUNPTH
CALLING SEQUENCE: CALL T1CHEX
CALLS TO: None

DESCRIPTION:

If the input frequency code (DME2FQ) equals the output code (ATUNE2), or if bit 9 of SIMFLG is set (It should check bit 8 - indicating that the DME #2 inputs are to be simulated for NAVIG), the 'out-neq-in' bit (bit 13) in FLSTA2 is cleared and the station fail timer (PSTMR) is zeroed. Otherwise, the 'out-neq-in' bit is set in FLSTA2 and PSTMR is checked. If PSTMR is zero (indicating first pass for this condition), it is set to the present time in seconds (TIME). Otherwise, PSTMR is subtracted from TIME and, if the result is greater than 4 seconds, PSTMR is zeroed. TUNPTH mainline code checks for PSTMR equal zero with FLSTA2 not zero.

GLOBAL INPUTS: ATUNE2, DME2FQ, FLSTA2, SIMFLG, TIME

GLOBAL OUTPUTS: FLSTA2

MODULE NAME: TUNDM2 (Autotune DME #2)
FILE NAME: TUNDM2.FOR
PROCESS: SLOW

PURPOSE: Find a suitable 'path station' when no path has been entered or the path has no stations coded.

CALLED BY: TUNPTH
CALLING SEQUENCE: CALL TUNDM2
CALLS TO: GET_LONG, GET_REAL, NXT_STA2, SEARCH_STA2

DESCRIPTION:

TUNDM2 begins by checking the 'search-in-progress' flag (RINPFL). If RINPFL is true, the high altitude flag (HIALT2) is set according to present aircraft altitude, and NXTSTA2 is called to update to the next station in the longitudinal strip.

If RINPFL is false, the permissible search range (ZONELM) is set to the minimum (ZONERGE = 40 nm), the zone search counter (RZNCTR) is reset to zero, the local strip pointer (X2PTR) is reset to the beginning of the longitudinal strip pointers and a search is initiated for that two degree wide longitudinal strip which includes the aircraft present position. This search continues until the strip is found or until the end of the longitudinal strips in the navigation data base is found. (The present data base contains data for all of the continental United States: 66 to 124 degrees west longitude).

When the proper strip is found, the zone pointer (RZNPTR) is set to point to the address of the first station in the strip, NAV_ADD2 is loaded with the address of this station, RINPFL is set true and SEARCH_STA2 is called to check if this is a usable station and search further if not.

GLOBAL INPUTS: IBPTR, IDDALT, SLLON,

GLOBAL OUTPUTS: RZNCTR, RZNPTR, ZONELM

118

MODULE NAME: NXTSTA2 (Select Next Station)
FILE NAME: TUNDM2.FOR
PROCESS: SLOW

PURPOSE: Select the next station in a search sequence.

CALLED BY: TUNDM2
CALLING SEQUENCE: CALL NXTSTA2
CALLS TO: CHOOSE_STR2, EXT_RGE2, GET_LONG, SEARCH_STA2

DESCRIPTION:

The station address (NAV_ADD2) is incremented to point to the next station in the longitudinal strip and the word at that address is checked. If the word is non-zero, SEARCH_STA2 is called to see if this is a valid station. If the address contains a zero (indicating the end of the strip), the zone counter (RZNCTR) is incremented. If it is then equal to five, EXT_RGE2 is called to extend the search range. Otherwise, CHOOSE_STR2 is called to choose the next longitudinal strip.

GLOBAL INPUTS: RZNCTR

GLOBAL OUTPUTS: RZNCTR

MODULE NAME: SEARCH_STA2 (Find Next Station)
FILE NAME: TUNDM2.FOR
PROCESS: SLOW

PURPOSE: Search a longitudinal strip for a suitable station.

CALLED BY: TUNDM2, NXTSTA2
CALLING SEQUENCE: CALL SEARCH_STA2
CALLS TO BMPSTA2, GET_BYTE, GET_REAL, SQRT

DESCRIPTION:

Initially the search-done flag (DONE) is cleared. Each station in the longitudinal strip is checked to verify that it is a VORTAC, that it is not one of the stations presently tuned and (if the high altitude flag (HIALT2) is set) that it is a high altitude navaid. If any of these checks fail, BMPSTA2 is called to update to the next station or set the DONE flag.

If the station is of the correct type, the distance to the station (DIST) is approximated and the effective range of the station (RANGE) is computed as a function of the elevation of the aircraft above the station. If the station is out of range (DIST > RANGE) or is beyond the allowable search range (DIST > ZONE LM), BMPSTA2 is called as above.

If all tests have passed, NVAD2A is set to the selected station address. Note that exit from this subroutine does not guarantee that a station has been found. At most, one full strip will be searched. If no suitable station is found, this will be detected by TUNDM2 on a subsequent iteration of the slow loop and result in another call to SEARCH_STA2.

GLOBAL INPUTS: ALTCOR, NVAD2A, NVAD3A, SLLAT, SLLON, ZONE LM

GLOBAL OUTPUTS: NVAD2A

120

MODULE NAME: BMPSTA2 (Pick Next-in-strip)
FILE NAME: TUNDM2.FOR
PROCESS: SLOW

PURPOSE: Increment to the next station in a search sequence.

CALLED BY: SEARCH_STA2
CALLING SEQUENCE: CALL BMPSTA2 (DONE)
CALLS TO: CHOOSE_STR2, EXT_RGE2, GET_LONG

DESCRIPTION:

The station address (NAV_ADD2) is incremented to point to the next station in the longitudinal strip and the word at that address is checked. If this word is zero, (indicating the end of the strip), the zone counter (RZNCTR) is incremented. If RZNCTR is then equal to five, EXT_RGE2 is called to extend the search range. Otherwise, CHOOSE_STR2 is called to choose the next longitudinal strip. In either event, the DONE flag is then set to cause SEARCH_STA2 to exit its search sequence.

GLOBAL INPUTS: RZNCTR

GLOBAL OUTPUTS: RZNCTR

MODULE NAME: CHOOSE_STR2 (Select Next Longitudinal Strip)
FILE NAME: TUNDM2.FOR
PROCESS: SLOW

PURPOSE: Choose a new longitudinal strip to search.

CALLED BY: BMPSTA2, NXTSTA2
CALLING SEQUENCE: CALL CHOOSE_STR2
CALLS TO: EXT_RGE2, GET_LONG, LOOKL2

DESCRIPTION:

This subroutine is entered when the previously selected longitudinal strip has been searched unsuccessfully for a suitable station. It begins by setting INDST, used by LOOKL2. (Note: INDST is actually a constant pointing to the westernmost strip in the navigation data base, but it cannot be computed at compile time). One of five sections of code is then executed depending on the value of the zone counter, RZNCTR. (hereinafter referred to as 'the value').

If the value is 1, we are in that 2 degree strip which includes the aircraft position. If another strip exists to the east, NAV_ADD2 is set to point to the first navaid in that strip. If not (the location at RZNPTR + 8 is zero), RZNCTR is set to 3 (to insure that a subsequent entry doesn't repeat the attempt to move east) and LOOKL2 is called to attempt to move to the west.

If the value is 2, processing proceeds identically, except that the initial attempt is to move to the second strip to the east of present position.

If the value is 3, LOOKL2 is called to move to the first strip to the west of present position.

If the value is 4, the second strip to the west is chosen if one exists. Otherwise EXT_RGE2 is called to extend the search range and reset the zone counter.

GLOBAL INPUTS: IBPTR, RZNCTR, RZNPTR

GLOBAL OUTPUTS: None

122

MODULE NAME: EXT_RGE2 (Extend Search Range)
FILE NAME: TUNDM2.FOR
PROCESS: SLOW

PURPOSE: Extend station search range by 40 nm.

CALLED BY: BMPSTA2, CHOOSE_STR2, NXTSTA2, LOOKL2
CALLING SEQUENCE: CALL EXT_RGE2
CALLS TO: GET_LONG

DESCRIPTION:

EXT_RGE2 is called when all permissible longitudinal strips have been searched out to the present search limit without finding a suitable navaid. The zone counter (RZNCTR) is zeroed, resetting the zone to aircraft present longitude, NAV_ADD2 is set to point to the first navaid in that strip (as indicated by RZNPTR) and the search range limit (ZONELM) is increased by 40 nm. If this increase results in ZONELM being more than 200 nm, the search in progress flag (RINPFL) is cleared -- thus reinitializing the search -- and ZONELM is reset to 40 nm.

GLOBAL INPUTS: RZNPTR

GLOBAL OUTPUTS: RNZCTR, ZONELM

MODULE NAME: LOOKL2 (Select Next Strip to West)
FILE NAME: TUNDM2.FOR
PROCESS: SLOW

PURPOSE: Choose a new longitudinal strip to the west (Look Left).

CALLED BY: CHOOSE_STR2
CALLING SEQUENCE: CALL LOOKL2(INDST)
CALLS TO: GET_LONG

DESCRIPTION:

The value of (RZNPTR - 24) is compared to INDST and, if greater, NAV_ADD2 is set equal to the longitudinal strip address found at the indicated address in the navigation data base. If (RZNPTR - 24) is less than INDST, there are no more longitudinal strips to the west, and EXT_RGE2 is called to extend the search range and reset the zone counter.

GLOBAL INPUTS: RZNPTR

GLOBAL OUTPUTS: None

MODULE NAME: TUNXTK (Cross Track Station Tuning)
FILE NAME: TUNXTK.FOR
PROCESS: SLOW

PURPOSE: To tune station #3 manually or by auto-tune to a suitable cross track station.

CALLED BY: HNAVSL
CALLING SEQUENCE: CALL TUNXTK
CALLS TO: GET_WORD, SEARCH_INIT, XTK_AUTO, XTK_MAN

DESCRIPTION:

A cross-station can be either manually or automatically selected and tuned. To manually tune a cross-station, one is entered via the PROGRESS #2 page on the CDU. The routine handling this CDU input finds the navaid in the navigation database and sets its address in the navaid address variable, NVAD3A. It also clears the cross-track mode flag, ATNAV3, indicating manual tuning mode. When not manual tuning, ATNAV3 will be set and auto tuning will be the active mode. In auto tuning mode, TUNXTK (with its subroutines) searches the navigation data base for a suitable navaid and tunes it. When error conditions are detected, new stations may be sought.

TUNXTK first checks to see if a cross path station is presently selected (NVAD3A not equal 0). If not, SEARCH_INIT is called to initiate the search for a station. Otherwise, F2 is next checked for failures. (This has no effect since F2 is cleared immediately before calling this procedure). The NAV64K flag is then checked. If it is false (GS less than 64 knots), the airplane must be on the ground, so no search would be successful. The output tuning code (ATUNE3) is set equal to that of the navaid pointed to by NVAD3A. FLSTA3 is then set to '1000' hex (which will inhibit use of the cross station) and processing is terminated.

If NAV64K is true and ATNAV3 is true (indicating that auto-tuning is permitted for DME3), subroutine XTK_AUTO is called to select a station. If ATNAV3 is false (indicating that the pilot has manually selected a station), XTK_MAN is called to tune the station and check its validity.

GLOBAL INPUTS: ATNAV3, FLSTA3, NAV64K, NVAD3A

GLOBAL OUTPUTS: ATUNE3, FLSTA3

MODULE NAME: XTK MAN (Manual Tuning)
FILE NAME: TUNXTK.FOR
PROCESS: SLOW

PURPOSE: To tune station #3 manually.

CALLED BY: TUNXTK
CALLING SEQUENCE: CALL XTK MAN
CALLS TO: CMP_FREQ, GET_REAL, GET_WORD, TUNCK

DESCRIPTION:

The output tuning code (ATUNE3) is set equal to that of the navaid pointed to by NVAD3A, the station status word (FLSTA3) is cleared and TUNCK is called to verify station geometry. If bad geometry status is returned (BADG = true), the bad geometry bit (bit 11) is set in FLSTA2 if ATNAV2 is true (forcing selection of a new path station) and otherwise is set in FLSTA3 (which will cause a DME fail indication on the CDU). If geometry is good, subroutine CMP_FREQ is called to verify that the selected station has been tuned.

Note: the line of code which sets FLSTA2 sets it equal to the 'OR' of FLSTA3 (vice FLSTA2) and the 'bad geometry' bit. This has no functional significance, but could make it confusing to analyze the status words.

GLOBAL INPUTS: ATNAV2, FLSTA3, NVAD3A

GLOBAL OUTPUTS: ATUNE3, FLSTA2, FLSTA3

MODULE NAME: XTK_AUTO (Auto Tuning)
FILE NAME: TUNXTK.FOR
PROCESS: SLOW

PURPOSE: To auto-tune station #3 to a suitable cross path station.

CALLED BY: TUNXTK
CALLING SEQUENCE: CALL XTK_AUTO
CALLS TO: CMP_FREQ, GEOM_CHK, NXTSTA, SEARCH_INIT,
SEARCH_STA

DESCRIPTION:

XTK_AUTO begins by checking that the cross station status word (FLSTA3) equals 0 and bit 8 of F0 (set by HNAVSL to indicate that the station is within the H > R cone) is not 0. This check will fail, as F0 was cleared immediately before calling TUNXTK. It should be checking for the PATH station within the H > R cone, which at this point would be logged in bit 8 of F2G. If within the H > R cone and no errors have been logged for the cross station (FLSTA3 = 0), there is no point in doing a station search as the geometry checks would be unreliable. CMP_FREQ would then be called to verify station tuning and processing terminated.

Processing continues by checking the search-in-progress flag (CINPFL). If it is set, SEARCH_STA is called to continue the search. Otherwise, FLSTA3 is checked. If it is zero, or if the only error is a 'station-not-tuned' error (bit 13) which has not timed out, GEOM_CHK is called to check the station geometry.

If FLSTA3 showed a station failure on entry or a 'station-not-tuned' has timed out, the cross station timer (XSTMR) is cleared and XST_INIT is checked. If it is false, SEARCH_INIT is called to initiate a station search. If it is true (indicating a search has already been initiated), NXTSTA is called to select the next station.

GLOBAL INPUTS: ATUNE3, FLSTA3, NAV64K, NVAD3A, TIME

GLOBAL OUTPUTS: ATUNE3, FLSTA2, FLSTA3

MODULE NAME: GEOM_CK (Geometry Checks)
FILE NAME: TUNXTK.FOR
PROCESS: SLOW

PURPOSE: Verify the cross station / path station geometry.

CALLED BY: XTK_AUTO
CALLING SEQUENCE: CALL GEOM_CK
CALLS TO: TUNCK, CMP_FREQ

DESCRIPTION:

GEOM_CK calls TUNCK to actually perform the geometry check. If bad geometry status is returned (BADG = true), the bad geometry bit (bit 11) is set in FLSTA3 (which will cause a new station to be selected). If geometry is good, the tuning code is fetched from the navigation data base and set in ATUNE3, FLSTA3 is zeroed and subroutine CMP_FREQ is called to verify that the selected station has been tuned.

GLOBAL INPUTS: FLSTA3, NVAD3A

GLOBAL OUTPUTS: ATUNE3, FLSTA3

128

MODULE NAME: NXTSTA (Select Next Station)
FILE NAME: TUNXTK.FOR
PROCESS: SLOW

PURPOSE: Select the next station in a search sequence.

CALLED BY: XTK_AUTO
CALLING SEQUENCE: CALL NXTSTA
CALLS TO: CHOOSE_STRIP, EXT_RANGE, GET_LONG, SEARCH_STA

DESCRIPTION:

The station address (NAV_ADDR) is incremented to point to the next station in the longitudinal strip and the word at that address is checked. If the word is non-zero, SEARCH_STA is called to see if this is a valid station. If the word is zero (indicating the end of the longitudinal strip), the zone counter (ZONCTR) is incremented. If it is then equal to five, EXT_RANGE is called to extend the search range. Otherwise, CHOOSE_STRIP is called to select the next longitudinal strip.

GLOBAL INPUTS: None

GLOBAL OUTPUTS: None

MODULE NAME: SEARCH_INIT (Initialize Station Search)
FILE NAME: TUNXTK.FOR
PROCESS: SLOW

PURPOSE: Initialize the search for station #3.

CALLED BY: TUNXTK, XTK_AUTO
CALLING SEQUENCE: CALL SEARCH_INIT
CALLS TO: GET_LONG, GET_REAL, SEARCH_STA

DESCRIPTION:

Processing begins by setting the station high altitude flag (HIALTF) if the aircraft is above 18000 feet. The permissible search range (ZONLIM) is set to the minimum (ZONRGE1 = 40 nm), the zone search counter (ZONCTR) is reset to zero, the local strip pointer (XPTR) is reset to the beginning of the longitudinal strips and a search is initiated for a strip which includes the aircraft present position. This search continues until the strip is found or the end of the longitudinal strips in the navigation data base is found. (The present data base contains data for all of the continental United States: 66 to 124 degrees West longitude).

Once the proper strip is found, ZONPTR is set to point to the this strip, NAV_ADDR is set to the address of the first navaid in the strip, the search-in-progress flag (CINPFL) is set and SEARCH_STA is called to select a usable station.

GLOBAL INPUTS: IBPTR, IDDALT, SLLON

GLOBAL OUTPUTS: ATUNE3, FLSTA2, FLSTA3

130

MODULE NAME: SEARCH_STA (Find Next Station)
FILE NAME: TUNXTK.FOR
PROCESS: SLOW

PURPOSE: To select and tune a suitable cross track station.

CALLED BY: SEARCH_INIT, XTK_AUTO
CALLING SEQUENCE: CALL SEARCH_STA
CALLS TO: BUMP_STA, CMP_FREQ, GET_BYTE, GET_REAL,
GET_WORD, TUNCK

DESCRIPTION:

Initially the search-done flag (DONE) is cleared. Each station in the longitudinal strip is checked to see if it is a VORTAC and (if the high altitude flag (HIALTF) is set) if it is a high altitude navaid. If either of these checks fail, BUMP_STA is called to select the next station and set the DONE flag (if this involved selection of a new longitudinal strip) to cause an exit from the subroutine.

If the station is of the correct type, TUNCK is called to check station geometry. If a bad geometry status is returned, or if either DME2 or DME3 is already tuned to this station, BUMP_STA is called as above. Otherwise, FLSTA3 is cleared, the 'search-in-progress' flag (CINPFL) is cleared, NVAD3A is set to the selected station address, ATUNE3 is set to the tuning code found in the navigation database for this station and CMP_FREQ is called to verify the tuning. Finally, the DONE flag is set to cause an exit.

GLOBAL INPUTS: ATUNE3, NVAD2A, NVAD3A

GLOBAL OUTPUTS: ATUNE3, FLSTA3, NVAD3A

MODULE NAME: CMP_FREQ (Check Station Tuning)
FILE NAME: TUNXTK.FOR
PROCESS: SLOW

PURPOSE: Verify tuning of DME3.

CALLED BY: GEOM_CK, SEARCH_STA, XTK_AUTO, XTK_MAN
CALLING SEQUENCE: CALL CMP_FREQ
CALLS TO: None

DESCRIPTION:

If the input frequency code (DME3FQ) equals the output code (ATUNE3), or if bit 9 of SIMFLG is set (indicating that the DME #3 inputs are to be simulated for NAVIG), the station fail timer (XSTMR) is zeroed. Otherwise, the 'out-neq-in' bit is set in FLSTA3 and XSTMR is checked. If it is zero (indicating first pass for this condition), it is set to the present time in seconds (TIME).

GLOBAL INPUTS: ATUNE3, DME3FQ, FLSTA3, SIMFLG, TIME

GLOBAL OUTPUTS: FLSTA3

132

MODULE NAME: BUMP_STA (Pick Next-strip)
FILE NAME: TUNXTK.FOR
PROCESS: SLOW

PURPOSE: Increment to the next station in a search sequence.

CALLED BY: SEARCH_STA
CALLING SEQUENCE: CALL BUMP_STA
CALLS TO: CHOOSE_STRIP, EXT_RANGE, GET_LONG

DESCRIPTION:

The station address (NAV_ADDR) is incremented to point to the next station in the longitudinal strip and the word at that address is checked. If the word is non-zero, Processing is terminated. If the word is zero (indicating the end of the strip), the zone counter (ZONCTR) is incremented. If it is then equal to five, EXT_RANGE is called to extend the search range. Otherwise, CHOOSE_STRIP is called to select the next longitudinal strip. In either case, a DONE status is returned.

GLOBAL INPUTS: None

GLOBAL OUTPUTS: None

MODULE NAME: CHOOSE_STRIP (Select Next Longitudinal Strip)
FILE NAME: TUNXTK.FOR
PROCESS: SLOW

PURPOSE: Select a two degree strip to search for a station.

CALLED BY: BUMP_STA, NXTSTA
CALLING SEQUENCE: CALL CHOOSE_STRIP
CALLS TO: EXT_RANGE, GET_LONG, LOOK_L

DESCRIPTION:

This subroutine is entered when the previously selected longitudinal strip has been searched unsuccessfully for a suitable station. It begins by setting INDST, used by LOOK_L. (Note: INDST is actually a constant pointing to the westernmost strip in bulk data, but cannot be computed at compile time). One of five sections of code is then executed depending on the value of ZONCTR. (ZONCTR has an entry value of 0 to 4 -- hereinafter referred to as 'the value').

If the value is 1, we are in that 2 degree strip which includes the aircraft position. If another strip exists to the east, NAV_ADDR is set to point to the first navaid in that strip. If not (the location at ZONPTR + 8 is zero), ZONCTR is set to 3 (to east) and LOOK_L is called to attempt to move to the first strip to the west.

If the value is 2, processing proceeds identically, except that the initial attempt is to move to the second strip to the east of present position.

If the value is 3, LOOK_L is called to attempt to move to the first strip to the west.

If the value is 4, the second strip to the west is chosen if one exists. Otherwise EXT_RANGE is called to extend the search range.

GLOBAL INPUTS: IBPTR

GLOBAL OUTPUTS: None

134

MODULE NAME: LOOK_L (Select Next Strip to West)
FILE NAME: TUNXTK.FOR
PROCESS: SLOW

PURPOSE: To extend the station search to the first longitudinal strip to the West, if one exists.

CALLED BY: CHOOSE_STRIP
CALLING SEQUENCE: CALL LOOK_L(INDST)
CALLS TO: EXT_RANGE, GET_LONG

DESCRIPTION:

LOOK_L sets NAV_ADDR to point to the first navaid in the next longitudinal strip to the west of aircraft present position if one exists. If there is no next strip to the west (ZONPTR - 24 is less than INDST), EXT_RANGE is called to extend the search range by 40 nm.

GLOBAL INPUTS: None

GLOBAL OUTPUTS: None

MODULE NAME: EXT_RANGE (Extend Search Range)
FILE NAME: TUNXTK.FOR
PROCESS: SLOW

PURPOSE: Extend search range for a cross-track station.

CALLED BY: BUMP_STA, CHOOSE_STRIP, LOOK_L, NXTSTA
CALLING SEQUENCE: CALL EXT_RANGE
CALLS TO: GET_LONG

DESCRIPTION:

EXT_RANGE is called when all permissible longitudinal strips have been searched unsuccessfully for a suitable navaid out to the present search limit. The zone counter (ZONCTR) is zeroed, resetting the zone to aircraft present longitude, NAV_ADDR is set to point to the first navaid in that strip (as indicated by ZONPTR) and the search range limit (ZONLIM) is increased by 40 nm. If this increase results in ZONLIM being more than 200 nm, the search in progress flag (CINPFL) is cleared, the search initialized flag (XST_INIT) is cleared and ZONLIM is reset to 40 nm.

GLOBAL INPUTS: None

GLOBAL OUTPUTS: None

MODULE NAME: TUNCK (Verify Station Geometry)
 FILE NAME: TUNCK.FOR
 PROCESS: SLOW

PURPOSE: To verify the geometry of a cross-track station

CALLED BY: GEOM_CK, SEARCH_STA, XTK_MAN
 CALLING SEQUENCE: CALL TUNCK(STA_LAT, STA_LON, STA_ALT, RANGE, TB)
 CALLS TO: SQRT

DESCRIPTION:

Processing begins by presetting 'good geometry' status (TB = false) and computing the delta lat (DTLAT) and delta lon (DTLON) between the aircraft and the selected cross station. DTLON is immediately multiplied by cos(LAT) to account for the varying width of a degree of longitude. Analogues of the sine and cosine of the angle between the stations (SINW, COSW) are then computed by cross multiplying DTLAT and DTLON by the delta lat (DPHI1) and lon (DLAM1) between the aircraft and the path station computed by CRBSC in HNAVSL. The following relations are implicit: The sine and cosine of the bearing to the cross station are proportional to DTLON and DTLAT, respectively. Similarly, the sine and cosine of the bearing to the path station are proportional to DLAM1 * cos(LAT) and DPHI1. Calling the bearing to the path station P, the bearing to the cross station X and the angle between them (the wedge angle) W, then

$$\begin{aligned}\sin(W) &= \sin(P - X) = \sin(P) \cos(X) - \cos(P) \sin(X) \\ \cos(W) &= \cos(P - X) = \cos(P) \cos(X) + \sin(P) \sin(X) \\ \tan(W) &= \sin(W) / \cos(W)\end{aligned}$$

The minimum (primary) wedge angle between stations for a good position estimate is 30 degrees. I.e, between 30 and 150 degrees left or right. This requirement is met if the tangent of the primary angle (abs(sin(W) / abs(cos(W))) is greater than the tangent of 30 degrees (0.57735). If this criteria is not met, TB is set true.

If the above test passes, processing continues by computing the approximate distance to the cross station (DIST), and the approximate effective range of the station (RANGE).

$$\begin{aligned}\text{DIST} &= 60.0 * \text{sqrt} (\text{DTLAT}^2 + \text{DTLON}^2) \\ \text{RANGE} &= 1.23 * \text{sqrt} (\text{ALTCOR} - \text{STA_ALT})\end{aligned}$$

If the distance is greater than the specified search range (MAXRG) or greater than the effective range (RANGE), TB is set true.

GLOBAL INPUTS: ALTCOR, SLLAT, SLLON

GLOBAL OUTPUTS: None

MODULE NAME: MSGG (Generate Message)
FILE NAME: MSGG.FOR
PROCESS: SLOW

PURPOSE: To output messages to the system test panel and onboard
 line printer.

CALLED BY: SLOW
CALLING SEQUENCE: CALL MSGG
CALLS TO: LIB\$SIGNAL

DESCRIPTION:

MSGG drives the system test panel display and indicators with data stored in a message buffer. This message buffer is also used to echo each message that goes to the system test panel on the printer as well. The length of the buffer (in bytes) is stored in WRDCNT. The message length determines whether there is a text message to output, or only lights and switches to turn on or off. A value of four in WRDCNT denotes the latter, otherwise a message is output to the system test panel and the onboard printer. These output operations are designed to occur during separate iterations of MSGG to minimize the occurrence of I/O interrupts which might affect the foreground timing. The status of these I/O operations is tested during each iteration and, if an exception is detected, an error message is displayed on the system console device.

GLOBAL INPUTS: IOACT, MSBUF, MSGST, WRDCNT

GLOBAL OUTPUTS: IOACT, WRDCNT

138

MODULE NAME: MESH (Error Message Tables)
FILE NAME: MESH.MAR
PROCESS: SLOW

PURPOSE: Repository for ASCII error messages.

CALLED BY: Non executable
CALLING SEQUENCE: Not called
CALLS TO: None

DESCRIPTION:

MESH contains a pool of ASCII error messages that are displayed on the system test panel.

GLOBAL INPUTS: N/A

GLOBAL OUTPUTS: N/A

MODULE NAME: SNAPOUT (Snap Output Processing)
FILE NAME: SNAPOUT.FOR
PROCESS: SLOW

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

CALLED BY: SLOW
CALLING SEQUENCE: CALL SNAPOUT
CALLS TO: None

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 SLOW 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 IOACT is used to signal that I/O is in progress and the output is then initiated through a call to SY\$\$QIO. Subroutine SNAST, specified in the QIO statement, clears IOACT when the I/O is complete.

GLOBAL INPUTS: RPTR, SNAPBUF

GLOBAL OUTPUTS: IOACT, RPTR

Section 6.0 FLIGHT CONTROLS

FLIGHT CONTROLS OVERVIEW

The Flight Controls routines provide for control of the ATOPS aircraft in all coupled flight modes. This includes an Attitude Control Wheel Steering mode (pitch and roll only) for the Forward Flight Deck, a Manual Electric mode (pitch, roll and yaw) for the Research (Aft) Flight Deck (RFD) and a number of computer aided modes offering pitch, roll, yaw and speed control from the Research Flight Deck. These include Attitude Control Wheel Steering (ACWS), Velocity Control Steering (VCWS) and a variety of autopilot modes ranging from track and flight path angle hold to 4D autopilot, through full autoland. Interface with the Mode Select Panel (MSP) for flight mode selection, direct control of selected track, flight path angle (or altitude held) and airspeed, or selection of 2-, 3- or 4D guidance is provided by these routines. The computation of aircraft state commands based on the Flight Management computation of errors from a predefined path (which may include speed and/or time constraints) is also computed here.

Flight Controls (FC) logic checks for validity of outputs from the Flight Management process, and provides for minimum operational capability from the RFD even in the absence of Flight Management outputs. Some of the signals required to drive the Navigation Display and Primary Flight Display are either computed primarily by the FC software or are conditionally computed when validity of the Flight Management outputs cannot be verified.

MODULE NAME: DATSEL (Flight Controls Data Selection)
 FILE NAME: DATSEL.FOR
 PROCESS: FCFAST

PURPOSE: To select input data for use by the Flight Controls Process (FCFAST).

CALLED BY: FCFAST
 CALLING SEQUENCE: CALL DATSEL
 CALLS TO: ANGL, ATAN2, SQRT

DESCRIPTION:

This module selectively overwrites those signals computed by HNAVFS in the Flight Management process, based upon the discretes INAVV (IRS Navigation Valid) and NCUVAL (Navigation Computations Valid).

If INAVV is true and NCUVAL is false, then LAT, LON, ALTCOR, HDCE, HDGTRU, HDD, VN, VE, GS, VGSDOT and XTACC are overwritten with the equivalent raw IRS inputs: LATINS, LONINS, IRSALT, IRSHDOT, THDG, HDDINS, VNINS, VEINS, GSINS, ATKINS, and XTKINS, respectively. GUID2D and MLSMOD are forced false as they cannot legitimately be true if NCUVAL is false. The following computed values are then overwritten with values computed from the raw inputs:

```
NAV64K = (GS >= 64)
TASFPS = TAS KTOFPS
GSFPS = GS KTOFPS
GSFPS2 = GSFPS GSFPS
DFTANG = ANGL(TK - THDG)
TKMAG = ANGL(TK - MAGVAR)
```

```
If NAV64K then
  TK = atan(VE, VN)
  GAMMA = RTOD HDCE / GSFPS
  TASGS = sqrt(TASFPS2 - HDOT2)
else
  TK = THDG
  GAMMA = 0.0
  TASGS = TAS
endif
```

Finally, HRAD is calculated by compensating the radar altimeter input (RADALT) for pitch angle,

$$\text{HRAD} = \text{RADALT} - 0.38 (\text{PITCH} - 2.0)$$

and runway heading error (DLPSI) and cross runway velocity (XTVEL) are calculated if a destination runway has been selected.

```

If (AIRPTS(2,3) ≠ 0) then
    DLPSI = ANGL(HDGTRU - RWYHDG)
    XTVEL = KTOFPS (cos(RWYHDG) VEINS - sin(RWYHDG) VNINS)
else
    DLPSI = -180.
    XTVEL = 0.0
endif

```

In the above, DLPSI is set to -180 when no runway has been selected to prevent LAND mode from engaging erroneously. Also, in the above computation of TASGS, HDCF should be used instead of HDOT.

GLOBAL INPUTS: AIRPTS, ATKINS, COSRH, GSINS, HDDINS, HDGTRU, INAVV, IRSALT, IRSHDOT, LATINS, LONINS, MAGVAR, NCUVAL, PITCH, RADALT, RWYHDG, SINRH, TAS, THDG, VEINS, VNINS, XTKINS

GLOBAL OUTPUTS: ALTCOR, DFTANG, DLPSI, GAMMA, GS, GSEPS, GSEPS2, GUID2D, HDCF, HDD, HDGTRU, HRAD, LAT, LON, MLSMOD, NAV64K, TASFPS, TASGS TK, TKMAG, VE, VGSDOT, VN, XTACC, XTVEL

MODULE NAME: MSPLGC (Mode Select Panel Logic)
 FILE NAME: MSPLGC.FOR
 PROCESS: FCFEAST

PURPOSE: To process inputs from the Control Mode Panel (CMP - previously known as the MSP) and to perform the logic associated with selection of guidance mode.

CALLED BY: FCFEAST
 CALLING SEQUENCE: CALL MSPLGC
 CALLS TO: ANGL, KNOBER, UNPK

DESCRIPTION:

MSPLGC consists of two fundamental parts. The first part handles inputs from the CMP knobs (using the subroutine KNOBER) and buttons (less the bottom-left four which are handled by MLOG). The second section contains the logic used to calculate which guidance modes are required. These modes are then output to the CMP and the rest of the system via the set of Booleans outlined in the description of MSPRO. This logic is outlined below.

The buttons on the CMP each contain lamps which may take a maximum of four possible states: off (unlit), or illuminated blue, amber or green. If unlit, the mode is not active; if green, the mode is providing guidance (engaged) and, if amber, the mode is armed. Blue indicates a preselect condition for those buttons which have an associated knob (CAS, ALT, FPA, TKA) or a 'not available' situation for the remaining guidance select buttons (LAND, HOR PATH, VERT PATH, TIME PATH). In the control wheel steering (CWS) modes, a maximum of one guidance mode in each axis may be engaged and one may be armed. (LAND provides guidance in both axes). In AUTO mode, exactly one guidance mode must be engaged in each axis and one may be armed. If the engaged mode is deselected or fails, the system will revert to VCWS. If both an engaged and an armed condition exists, the following priorities prevail (descending order):

Horizontal guidance - LAND, HOR PATH, TKA SEL;
 Vertical guidance - LAND, VERT PATH, ALT ENG, FPA SEL;
 Speed guidance - TIME PATH, CAS ENG.

The four knob/button pairs also have an associated numeric readout. If the lamp is unlit, the readout indicates the current value of airspeed, altitude, flight path angle or track angle, respectively. If lit, the selected value is displayed. Turning a knob adjusts the displayed value for the associated parameter and, if the button was previously unlit, causes the blue lamp to be lit for eight seconds. If the button is then pressed within this time, the mode is armed or engaged (as appropriate) with the preselected value retained. If not, the blue light goes off and the display returns to current value. CAS is unique in that the preselect state results in an amber lamp (which does not auto-

extinguish). Successive pushes of the CAS ENG button causes the mode to toggle between preselect and engage (green). To extinguish the lamp / return to disengaged state, the autothrottle disconnect button on the throttle handles (ATDC) must be pressed.

Of the remaining buttons (the buttons on the bottom row), the first three (VEL CWS, ATT CWS, AUTO) are handled entirely by procedure MLOG and are mutually exclusive. I.e, at most one will be illuminated green and the remainder will be off. The next (LAND) is also handled by MLOG, but interacts with and shares lamp color logic with the remaining three mode select buttons (HOR PATH, VERT PATH, TIME PATH), which are handled by MSPLGC. If the LAND button is depressed when unlit, the blue light will be illuminated for eight seconds if the minimum requirements for LAND mode are not met. Otherwise, the amber (armed) light will be engaged until the engage criteria are met, at which time the green light will be lit and the amber light extinguished. If depressed when amber or green, it will return to the unlit state. The remaining three buttons are serially dependent: i.e, HOR PATH engaged is a prerequisite for VERT PATH, which is itself a prerequisite for TIME path. However, pressing TIME PATH button when HOR PATH is neither armed nor engaged will cause all three modes to become armed or engaged (as appropriate) if the criteria are met. If not, TIME PATH will be lit blue. Similarly, VERT PATH will either arm/engage both HOR PATH and VERT PATH or turn blue trying. Additional logical requirements exist which should become clear as the mechanizing logic is explained.

The first two lines of code are remnants of the data link experiment. In that experiment, it was possible to transmit new altitudes and headings (among other things) from the ground, and code in process HDL would place the values directly in the appropriate summer (ALTSUM and TKASUM, for the above), then set the preselect Boolean (PSTALT or PSTTKA) to retain the value. The lines in question check for PSTALT or PSTTKA true with the matching 'blue light' counter (RAKNOB, RTKNOB) zero. If this situation exists, the appropriate counter is set to 161 to cause the blue light to stay on for eight seconds to give the pilot time to accept the input by pressing ALTSEL or TKSEL. The next DO loop decrements all counters that are non-zero (RCKNOB, RAKNOB, RFKNOB, RTKNOB, D2CTR, D3CTR, D4CTR, ALTIN). Next the transfer Booleans in the array BUTNS (DACWS, D2D, D3D, D4D, DIASEL, DALSEL, DFPSEL, DTKSEL) are cleared preparatory to reading the new button data, and the local copy of the mode reversion flag (MODREV) is cleared.

If COLDST is true, indicating system initialization in progress, OLDTOG is set equal to the present value of TOGGLE (the CMP word which indicates which button or knob to check) and processing ends. Otherwise processing continues by taking the exclusive or (saved in ITMP) of TOGGLE and OLDTOG. ITMP is then used to determine what to do next.

If ITMP equals 1, the altitude knob was turned. Procedure KNOBER is called to process the knob input. The formal parameters specify the input word to process, the high and low speed scale factors, the location of the return value and of the counter (timer) for the knob. KNOBER returns the selected delta value and

sets RAKNOB to 160 if a valid input was found. (Otherwise zeros are returned. See KNOBER documentation.) Subsequent inline code conditionally adds the delta value to the previous value of ALTSUM. If the previous value was not an even multiple of 100 feet, ALTSUM is set to the next nearest 100 foot value in the direction the knob was turned. Otherwise it is set to the sum of ALTSUM and the delta value, limited to a minimum of zero and a maximum of 35000 feet (MAXALT). Additionally, the altitude attained flag (ALTATT) is cleared and the variable ALTIN is set to 30. When ALTIN is non-zero, logic in procedure VERCMD applies an easy-on to the vertical acceleration command.

If ITMP equals 2, the flight path angle knob was turned. KNOBER is called and inline code adds the delta to FPASUM, limiting the result to +/- GAMLIM (the maximum allowable flight path angle, which is flight mode dependent). ALTIN is also set to 30 when this knob is turned.

If ITMP equals 4, the track angle knob was turned. KNOBER is called and TKASUM is updated using the ANGL function, which returns an angle with a maximum value of +/- 180 degrees. The variable TRKIN is also set to 60, which causes procedure LATCMD to provide an easy-on to the bank angle command. It also causes procedure LATRL to clear the TKSUNC discrete, which re-activates the curved trend vector on the PFD. Procedure MSPRO sets a bit in the FCFLGS word which forces the selected track display to be enabled on the PFD. MSPRO decrements TRKIN.

If ITMP equals 8, the air speed knob was turned. KNOBER is called and IASSUM is updated and limited to a minimum of the reference airspeed (IASREF) and a maximum of 350 knots (MAXIAS). IASREF is pilot selected via the CDU INIT page.

If ITMP equals 16, a button was pushed. Procedure UNPK is called to unpack the CMP select discrete word (CMPMSD) into the temporary Boolean array, BUTNS. The following DO loop sets the appropriate counter to 160 if HOR PATH (D2D), VERT PATH (D3D) or TIME PATH (D4D) was pressed. If the selected mode cannot be armed or engaged, this will result in the eight second blue light mentioned above. The following logic forces D3D if D4D is selected and VERARM is false (VERT PATH neither armed nor engaged), then forces D2D if D3D was selected (or forced) and HORARM and LOCE are both false.

The next section implements the actual guidance selection logic. But first a few more preliminary equations. LOFF is set whenever an engaged lateral guidance mode is deselected; i.e, if TKA SEL is pressed when track select is engaged, HOR PATH is pressed when horizontal path guidance is engaged or LAND is pressed when land mode is engaged. VOFF is set whenever an engaged vertical guidance mode is deselected: i.e, if ALT ENG is pressed when altitude hold is engaged, FPA SEL is pressed when flight path angle is engaged or VERT PATH is pressed when vertical path is engaged. Auto Select (AUTOS) is then forced when AUTOE is true and either LOFF or VOFF becomes true. This causes logic in MLOG to demode from AUTO to the next highest available mode (usually VCWS). Also, Land Select (LANDS) is forced if Land Ready (LANDR) is true and either HOR PATH or TKA SEL is selected. This causes MLOG to

turn off LANDR and all other land mode flags. If land mode was previously engaged, this also causes a demode from AUTO mode (if engaged).

The next section of logic ensures retention of path guidance engage during modification of a path. When a path is modified or a new path is entered, the horizontal and vertical path guidance available Booleans (GUID2D and GUID3D) are temporarily set false. This is necessary to ensure proper reinitialization of the horizontal/vertical guidance equations in procedure HVGUID, but would normally cause loss of both horizontal and vertical path arm (which is neither necessary nor desirable). SETGD is set non-zero during a path update sequence by procedure EXECUTE. The local Boolean NEWPLN is set true when SETGD is non-zero. The local Booleans GD2D and GD3D are then set to the OR of NEWPLN and GUID2D or GUID3D, respectively.

The default logic causes HORPTH to engage immediately when horizontal path is selected, regardless of distance from the path. However if the pilot prefers to set his own approach angle, he may do so by turning the track select knob after pressing HOR PATH iff the distance from the path is greater than the mandatory engage distance (BCFLAG = XTK > XTKLIM). Alternatively, the logic can be configured to never set HORPTH when BCFLAG is true by setting the global Boolean SPOB1. This is controlled by the local Boolean B, defined as follows:

$$B = (B \cdot \text{HORARM} \cdot \overline{\text{AUTOS}}) + (\text{RTKNOB} = 160) + \text{SPOB1}$$

The following equations are then evaluated:

Horizontal Path Select:

$$\text{IF } (\text{HORARM}) \quad \text{D2CTR} = 0$$

$$\text{HORARM} = \text{GD2D} \cdot ((\text{D2D} \cdot \overline{\text{HORARM}}) + (\text{HORARM} \cdot \overline{\text{NOT}(\text{DTKSEL} + \text{D2D} + \text{LOCE} + (\text{LANDS} \cdot \overline{\text{HORPTH}}))}))$$

Where: DTKSEL is a momentary Boolean set when the TKA SEL button is pressed;
D2D is a momentary Boolean set when the HOR PATH button is pressed.

$$\text{HORBLU} = (\text{D2CTR} > 0) \cdot \overline{\text{HORARM}}$$

$$\text{HORPTH} = \text{HORARM} \cdot \text{GUID2D} \cdot \overline{\text{BCFLAG}}$$

When the end of a 3D path is reached, or when land mode is selected and the localizer is engaged but the glideslope is not engaged (vertical path guidance is still armed or engaged) when the aircraft descends below 500 feet AGL, altitude hold is forced by the following logic:

$$\text{if } (\text{VERARM} \cdot (\text{GD3D} + (\text{PTR2D} = 1) + \underline{\hspace{2cm}}))$$

```

                (((ALTCOR - RYELEV) < 500) • LOCE)))
VERARM = .FALSE.
ALTARM = .TRUE.
MODREV = .TRUE.

```

endif

Vertical Path Select:

```

if (VERARM) D3CTR = 0
VERARM = GD3D • (HORARM + LOCE) • ((D3D •  $\overline{\text{VERARM}}$ ) +
    (VERARM • NOT(D3D + DALSEL + DFPSEL + GSENG +
    (LANDS •  $\overline{\text{VERPTH}}$ ))))

```

Where: DALSEL is a momentary Boolean set when the
ALT ENG button is pressed;
DFPSEL is a momentary Boolean set when the
FPA SEL button is pressed;
D3D is a momentary Boolean set when the
VER PATH button is pressed.

```
VERBLU = (D3CTR > 0) •  $\overline{\text{VERARM}}$ 
```

```
VERPTH = VERARM • (HORPTH + LOCE) • GUID3D •  $\overline{\text{VBCFLG}}$ 
```

Where: VBCFLG is the vertical path 'be careful' flag.

Time Path Select:

```
IF (TIMARM) D4CTR = 0
```

```

TIMARM = VERARM • (SDC > IASREF) • (CAS > IASREF) •
    ((D4D •  $\overline{\text{TIMARM}}$ ) + (TIMARM •  $\overline{(\text{D4D} + \text{DIASEL} + \text{ATDC})})$ )

```

Where: D4D is a momentary Boolean set when the
TIME PATH button is pressed;
DIASEL is a momentary Boolean set when the
CAS ENG button is pressed;
ATDC is a Boolean set when the autothrottle
disconnect button is pressed;
SDC is the speed command calculated in
HVGUID or TGUID.

$$\text{TIMBLU} = (\text{D4CTR} > 0) \cdot \overline{\text{TIMARM}}$$

$$\text{TIMPTH} = \text{TIMARM} \cdot \text{VERPTH} \cdot \overline{\text{SPDLMT}}$$

Where: SPDLMT is a Boolean set when the present CAS is either above or below the limits for the present flap setting.

$$\text{if } (\text{TIMPTH} \cdot \text{PTIMPT}) \quad \text{TIMARM} = \text{ATE}$$

Where: PTIMPT is the previous interation's value of TIMPTH;

ATE is the auto-throttle engage discrete set by procedure ATHCL.

Airspeed hold Select:

$$\text{IF } (\text{ATDC}) \quad \text{RCKNOB} = 0$$

$$\text{IASARM} = \overline{\text{ATDC}} \cdot ((\text{RCKNOB} = 160) + \text{IASARM}) \cdot \overline{\text{IASSEL}} + (\text{IASSEL} \cdot \text{DIASEL})$$

Where: RCKNOB is set when the CAS knob is turned;
 D4D is a momentary Boolean set when the TIME PATH button is pressed;
 DIASEL is a momentary Boolean set when the CAS ENG button is pressed.

$$\text{IASSEL} = \overline{\text{ATDC}} \cdot ((\text{IASSEL} \cdot \text{ATE} \cdot (\overline{\text{DIASEL} + \text{TIMPTH}})) + (\text{DIASEL} \cdot \overline{\text{IASSEL}}) + (\text{PTIMPT} \cdot (\overline{\text{DIASEL} + \text{TIMPTH} + \text{IASSEL}})))$$

Where: ATDC is the autothrottle disconnect flag;
 PTIMPT is TIMPTH from the previous frame (this logic causes reversion from TIMPTH to IASSEL).

Altitude Select:

$$\text{PSTALT} = (\text{RAKNOB} > 0) \cdot \overline{\text{ALTARM}}$$

Where: RAKNOB is set to 160 when the altitude knob is turned.

$$\text{ALTARM} = (\text{DALSEL} \cdot \overline{\text{ALTARM}}) + (\text{ALTARM} \cdot \text{.NOT.}(\text{DALSEL} + \text{DFPSEL} + \text{GSENG} + \text{VERPTH} + ((\text{LANDS} + \text{D3D}) \cdot \overline{\text{ALTSEL}})))$$

```
IF (ALTARM) RAKNOB = 0
```

The following logic is required

- 1) to ensure ALTSEL is cleared when demoding from AGCSE (MODEX < 4),
- 2) to ensure ALTSEL is cleared when a new altitude is selected which is beyond the engage limits and
- 3) to invert the priority of ALTSEL and VERPTH when a new altitude select input is made.

```
if (MODEX < 4) ALTSEL = .FALSE.
```

```
if (ALTSEL * (|DELALT| > 1200)) THEN
```

```
  DALSEL = .TRUE.
```

```
  IF (VERPTH) VERARM = .FALSE.
```

```
endif
```

```
ALTSEL = ALTARM * ((ALTSEL * (|DELALT| < 1200)) +
  (|DELALT| < 52) +
  ((DELALT >= 0) * (DELALT < 14*HDCF)) +
  ((DELALT < 0) * (DELALT > 14*HDCF)))
```

Where DELALT is altitude select/hold error and
HDCF is filtered vertical velocity.

The following logic sets the approach speed to either a vertical path or a selected altitude at the time that ALT ENG or VERT PATH is selected.

```
if ((DALSEL * ALTARM * ALTSEL) + (D3D * VERARM * VERPTH)) then
  if (D3D) then
    TMP1 = HER
    TMP2 = PFPA
```

```
  else
    TMP1 = DELALT
    TMP2 = ZERO
  endif
```

```
  if (HDCF sign(1, TMP1) <= 8.333)
    TEMP = 8.333
```

```
  else
    TEMP = |HDCF|
  endif
```

```
  FPASUM = TMP2 + DTOR (sign(TEMP, TMP1) / max(GSFPS, 170))
  PFPSEL = .TRUE.
  ALTIN = 30
```

```
endif
```

Where: HER is the vertical path error in feet;
 PFPA is the vertical path flight path angle in degrees;
 DELALT is the altitude error in feet;
 ZERO is the altitude hold nominal flight path angle;
 8.333 is the desired approach velocity (500 fpm).
 PFPSEL is set to prevent a mode reversion alarm when reverting from ALTSEL to FPA SEL and VER ARM. (As the code is written, this demode will never occur, but it should.)

Note: in the above, the segment 3D flight path angle (FPA) should be used vice the programmed flight path angle (PFPA).

Flight Path Angle Select:

$$PSTFPA = (RFKNOB > 0) \cdot \overline{(FPASEL + DFPSEL)}$$

Where: RFKNOB is set to 160 when the FPA knob is turned;
 DFPSEL is a momentary Boolean set when the FPA SEL button is pushed.

$$FPASEL = (DFPSEL \cdot \overline{FPASEL}) + ((AUTOE \cdot \overline{VOFF}) + ((VERARM + ALTARM + (FPASEL \cdot \overline{DFPSEL})) \cdot \overline{AUTOE}) \cdot \overline{(VERPTH + ALTSEL + GSENG)})$$

Track Angle Select:

$$PSTTKA = (RTKNOB > 0) \cdot \overline{(TKSEL + DTKSEL)}$$

Where: RTKNOB is set to 160 when track knob is turned;
 DTKSEL is a momentary Boolean set when the TKA SEL button is pressed.

$$TKSEL = (DTKSEL \cdot \overline{TKSEL}) + (((AUTOE \cdot \overline{LOFF}) + (((D2D \cdot HORARM) + LANDS + (TKSEL \cdot \overline{DTKSEL})) \cdot \overline{AUTOE})) \cdot \overline{(LOCE + HORPTH)})$$

Where: D2D is a momentary Boolean set when the HOR PATH button is pressed.

Final processing for this procedure sets the mode reversion flag (MDWARN) whenever a mode reversion occurs not deliberately initiated by the pilot. MDWARN is set for one iteration to cause an audible alarm to sound in the research flight deck. The present

value of the mode select discrettes are then loaded into the 'past value' discrettes, used primarily to mechanize the MDWARN logic. This routine shares common local data with procedures MSPRO, LATCMD and VERCMD.

GLOBAL INPUTS: ACWSS, ALTARM, ALTCOR, ALTSEL, ALTSUM, ATDC, ATE, AUTOS, AUTOE, BCFLAG, CAS, CMPALT, CMPFPA, CMPIAS, CMPMSD, CMPTKA, COLDST, DELALT, FPASEL, FPASUM, GAMLIM, GAS, GSENG, GSFPS, GUID2D, GUID3D, HDCF, HER, HORARM, HORPTH, IASREF, IASSEL, IASSUM, LANDR, LANDS, LOCE, PFPA, PSTALT, PSTTKA, PTR2D, RYELEV, SDC, SETGD, SPDLMT, SPOB1, TIMARM, TKASUM, TKSEL, TOGGLE, VBCFLG, VCWSS, VERARM, VERPTH

GLOBAL OUTPUTS: ALTARM, ALTATT, ALTSEL, ALTSUM, D2D, D3D, DALSEL, DFPSEL, DIASEL, DTKSEL, FPASEL, FPASUM, HORARM, HORBLU, HORPTH, IASARM, IASSEL, IASSUM, LANDS, MDWARN, PSTALT, PSTFPA, PSTTKA, TIMARM, TIMBLU, TIMPTH, TKASUM, TKSEL, TRKIN, VERARM, VERBLU, VERPTH

MODULE NAME: KNOBER (MSP Knob Processing)
FILE NAME: KNOBER.MAR
PROCESS: FCFAST

PURPOSE: To convert the bit pattern in a Control Mode Panel (CMP) input word to a floating point value.

CALLED BY: MSPLGC
CALLING SEQUENCE: CALL KNOBER(P1, P2, P3, P4, P5)

Where: P1 - CMP input
P2 - high increment value
P3 - low increment value
P4 - floating point output
P5 - knob count variable

CALLS TO: None

DESCRIPTION:

This subroutine is called by MSPLGC to convert a control mode panel knob input word into a floating point delta value. Prior to calling this routine, a parity check is performed (by procedure IOFLL in the HDL process) on each knob input value. If an error is detected, the input value is cleared (set to zero). KNOBER checks this input value and if it is zero, control is immediately returned to the caller.

The knob input word is examined to determine if the high-speed or low-speed gain should be used to compute the delta value. Then a test is made to determine if the input value is positive or negative. If it is negative, the sign bit is extended through the entire word for ease of computation and to guard against extraneous inputs. If the knob input word is positive, all bits are cleared except those in the actual count field. The input value is then multiplied by the appropriate gain and the knob count variable is set to 160 decimal, which serves as the eight-second timer for the preselect (blue light) condition when a knob is turned.

GLOBAL INPUTS: NONE

GLOBAL OUTPUTS: NONE

MODULE NAME: MSPRO (Mode Select Panel Readout)
 FILE NAME: MSPRO.MAR
 PROCESS: FCFAST

PURPOSE: Commands the Control Mode Panel (CMP) to light the appropriate mode lights and to display air speed, altitude, flight path angle, and track angle.

CALLED BY: FCFAST
 CALLING SEQUENCE: CALL MSPRO
 CALLS TO: NONE

DESCRIPTION:

Each time this routine is called, all lamps are initially turned off. A test is then made to determine if a new airplane track has been selected. If so (TRKIN not equal 0), TRKIN is decremented and the track pointer display flag is set (bit 12 of LFCFLG). In the first half of MSPRO the following booleans are checked and, if true, the corresponding light is lit. These booleans are packed into two words (DSTOMP, DSTOMP+2) for output. A low bit turns the light on and a high bit turns it off (active low).

DSTOMP:

<u>BIT</u>	<u>BOOLEAN</u>	<u>MEANING</u>	<u>LAMP COLOR</u>
8	PSTTKA	preselect Track Angle mode	blue
9	HORBLU	Horizontal Path unavailable	blue
10	TKSEL	Track Angle mode engaged	green
11	PSTFPA	Preselect Flight Path Angle	blue
12	TIMPTH	Speed mode/Time Path(4D) engaged	green
13	FPASEL	Flight Path Angle mode engaged	green
14	PSTALT	Preselect Altitude hold	blue
15	ALTARM	Altitude hold mode Armed	amber

DSTOMP+2:

0	ALTSEL	altitude hold mode engaged	green
1	IASARM	preselect calibrated air speed	amber
2	VERBLU	vertical path unavailable	blue
3	IASSEL	calibrated air speed engaged	green
4	TIMBLU	time path mode (4D) unavailable	blue
5	TIMARM	time path mode (4D) armed	amber
6	VERARM	vertical path mode (3D) armed	amber
7	VERPTH	vertical path mode (3D) engaged	green
8	HORARM	horizontal path mode (2D) armed	amber
9	HORPTH	horizontal path mode (2D) engaged	green

The second half of MSPRO calculates the correct values to be output to the four display windows on the CMP. These windows display one of three possible values for CAS, ALTITUDE, FPA, and TKA. The value displayed is either the preselected value (if in

preselect mode), the selected value (if mode is engaged), or the actual value (if mode is neither preselected or engaged).

GLOBAL INPUTS: ALTARM, ALTSEL, ALTSUM, FPASEL, FPASUM, GUID4D,
 HER, HORARM, HORPTH, IASSEL, IASSUM, MAGVAR, PSTALT,
 PSTFPA, PSTIAS, PSTTKA, TIMARM, TIMPTH, TKASUM, TKE,
 TKASEL, TRKIN, VERARM, VERPTH, XTK

GLOBAL OUTPUTS: ALTOMP, ASTOMP, DSTOMP, FCFLGS, FPTOMP, TKTOMP,
 TRKIN

MODULE NAME: MLOG (Mode Logic)
FILE NAME: MLOG.FOR
PROCESS: FCFAST

PURPOSE: To determine and control the flight mode for operation of the Flight Controls programs.

CALLED BY: FCFAST
CALLING SEQUENCE: CALL MLOG
CALLS TO: ANGL, DETNT

DESCRIPTION:

Mode Logic (MLOG) determines and selects a flight control mode under the Advanced Guidance and Control System (AGCS). It first checks the system run status and terminates if in HOLD Mode, otherwise, processing continues. If on initial entry neither RUN nor HOLD mode is selected, IC mode is forced. The first of the two packed discrete words sent to displays (FCFLGS) is then loaded from the local copy (LFCFLG) built during the previous iteration. (Only one bit is used in the second word, so no intermediate word is required). All but the Displays Lateral Beam Sensed (DLBS) and Vertical Beam Sensed (DVBS) bits are then cleared from LFCFLG and the MLSVLD and MLSMOD bits are set as appropriate. As the flight mode and submode is selected, the appropriate bit is set in LFCFLG. The FCFLGS words are sent to the Displays computer where they are used to display the flight mode on the Primary Flight Display. (For a complete description of the bit usage in FCFLGS see DISNAV.INC).

The next section of code determines the source of land guidance signals. Possibilities are ILS, ILS simulated from MLS, ILS simulated from GPS, full MLS, and MLS simulated from GPS. The discrete MLSM indicates that MLS-type signals are used for land guidance, whether derived from MLS or GPS. MLSMOD (MLS in use for navigation) indicates that MLS signals are also to be used for Autoland, assuming the MLS Azimuth phase center bearing matches runway heading for the selected runway. MLS, once selected, takes precedence over GPS (GPNAV/GPLND) when GPS is selected for navigation into the MLS coverage area. If MLSM is true, MSW6 and MSW1 (indicating certain additional MLS or GPS derived signals are to be used for approach and flare, respectively) are set from bits in the MLS configuration word (MCONF), and height above the touchdown zone (HTDZ) is set from MLS altitude (ZHAT). Otherwise, MSW1 and MSW6 are cleared and HTDZ is set to radar altitude (HRAD).

If MLSM is false, ILS-type capture and guidance are in effect, but the signal source must still be selected. The default is standard ILS, and LOCDEV and GSDEV are loaded from RLOCDV and RGSDEV, respectively (the raw ILS inputs). If the simulated ILS bit in MCONF ('0800'X) is set, LOCDEV and GSDEV are loaded from GPSETA and GPSBTA if GPLND is true, and otherwise from MLS AZ and EL1. (Note: the exact implementation of this logic, together with the way GPLND is set by the CDU logic, results in improper operation if both GPS and MLS are selected). Next, if LANDR or LANDS is true, the selected ILS parameters are processed. If no

glideslope angle has been selected (this could only be true if no runway is selected), a default value of 2.75 degrees is forced. Prior to localizer ONCRS, localizer deviation is limited to 2.0 degrees of beam error. At ONCRS, localizer deviation has a variable limit applied to it which is a function of altitude (HTDZ). The limiter is used to reduce localizer deviation due to external disturbances. The localizer variable limit function is:

$$\begin{aligned} \text{LOCVL} &= .2 \text{ degrees,} && \text{for } H < 500 \text{ feet.} \\ &.0006 H - .1 \text{ degree,} && \text{for } H \Rightarrow 500 \text{ feet.} \end{aligned}$$

Since the localizer beam converges as the transmitter is approached, the localizer error signal needs to be attenuated as distance from the localizer transmitter decreases in order to maintain constant beam error per foot of deviation from beam center. Since this distance is not known directly, altitude is used as an analog of distance. The gain does not decrease below .20 because localizer beam error is used for rollout guidance. After gain programming, localizer deviation (ETAFT) is 'per foot', rather than 'per degree'.

The next section determines flight mode, beginning by saving the previous value of MODEX in MODE2. For Advanced Guidance and Control System (AGCS) engagement, either the FFD CWS or the AFD Flight Controls paddle must be raised on the Control and Command Panel in the forward flight deck. Successful engagement occurs when several conditions are satisfied and the AGCS Engage Enable flag (AEE) becomes true. To allow time for this logic the pilot must hold the paddle up for at least 0.15 seconds. If AEE remains false, the paddle will spring down to the off position. AEE true is a precondition for all flight modes except PRENG.

If the DELAY variable is non-zero, it indicates that there was a mode logic "dropout" and mode determination is delayed for four iterations (200 Msec) to allow time for the engage paddle to drop. If DELAY is zero, the next step is to select and scale the pilot inputs (DCOL and DWHL): if Forward Flight Deck Select (FFDS) is true, the forward flight deck inputs FCOL and FWHL are used; if not, the Research Flight Deck (RFD) SAC inputs (SDCOL/SDWHL) are selected if the sidearm controller is selected (SSTICK is true) and the RFD broolley inputs (BDCOL/BDWHL - presently only available in the EASILY lab) otherwise. Appropriate deadzone values for wheel (WDZNE) and column (DZNE) are set at the same time. If FFDS is false, subsequent logic increases these deadzone values by a factor of 2.0 if LABFLG is true, as the 'joy sticks' in the EASILY lab are not very precise. If LABFLG is false, DZNE is still increased by a factor of 1.5 if ALTHLD is true and WDZNE by a factor of 1.5 if TRKHLD is true. (This applies to VCWS only).

The logic for initial AEE determination is:

$$\text{AFCSS} = \text{FFDS} + \text{AFDS} \quad (\text{performed in hardware})$$

$$\text{AEE} = \text{AFCSS} \cdot \text{AFCSV} \cdot \text{DETNT}(1.5, 4.5) \cdot \overline{\text{FAIL2}(\text{AFCS})}$$

Where: AFCSV = Advanced Flight Control System Valid
 DETNT is a function which returns true if both DCOL & DWHL, respectively, are within parameter limits.

And to remain valid:

$$AEE = AFCSS \cdot AFCSV \cdot \overline{FAIL2(AFCS)}$$

At the time AEE becomes true, Computational Reset (CRSET) is also set, which causes the second fail (Fail2) array to be cleared, and DSPLF(AFCS) is cleared to ensure that any subsequent errors will result in display of the 'Mode Failure' message. The appropriate DSPLF entry is also cleared at engagement of each of the modes discussed below.

MODE OVERVIEW:

There are ten flight modes available. Pre-engage (PRENG) is the default when no other mode can be sustained (AEE is false). Forward Flight Deck Engage (FFDE) permits control wheel steering (CWS) from the forward flight deck. All other modes pertain only to the Research (aft) Flight Deck. Manual-Electric mode (MANEL) permits manual control with minimum computer aid. The RFD CWS modes permit control wheel steering with reference to attitude (ACWS) or to the velocity vector (VCWS). The AUTO modes provide automatic flight control for 3D guidance and the landing phase through decrab (DECRB), FLARE, and rollout (RLOUT). In each case where a mode is said to be set, the mode index word (MODEX) is set to the appropriate value and a Boolean true is subsequently stored in FLAGS(MODEX) to set the appropriate mode discrete.

FFDE (Forward Flight Deck Engage Mode):

FFDE is the only computer aided flight mode available to the FFD. If AEE is true or becomes true according to the logic above, then, if FFD is selected (FFDS is true) and the FFD 2nd fail flag is clear, the FFD mode is engaged and further mode determination is bypassed. The pilots then have CWS in the forward cockpit. The logic for FFDE is:

$$FFDE = AEE \cdot FFDS \cdot \overline{FAIL2(FFD)}$$

Once true FFDE remains true as long as AEE remains true, FAIL2(FFD) remains false and no other mode is selected by pilot action.

AFT FLIGHT DECK MODES:

ACWSE (Attitude CWS Engaged Mode):

Attitude control wheel steering maintains whatever pitch and roll attitude exists when the pilot returns the column and wheel to detent. It is the default mode for the RFD when no other mode is selected. ACWS Engage criteria is:

$$ACWSE = AEE \cdot (ACWSS + ACWSE + (MODEX = 0)) \cdot AGCSS \cdot \\ \frac{FFDS + (ACWSE \cdot ACWSS) + FAIL2(ACWS)}{\cdot DETNT(DZNE, WZNE)}$$

Where: The flag AGCSS corresponds with the position of the guarded MANEL/AGCSS toggle switch on the flight controls pallet. If all conditions except AGCSS are met, control goes to MANEL.

Once true, ACWSE remains true so long as AEE and AGCSS remains true, FAIL2(ACWS) remains false and no other mode is selected.

VCWSE (Velocity CWS Engaged Mode):

This mode provides aided CWS with respect to the velocity vector of the aircraft. The bank, track and flight path angles are the autopilot hold references. VCWSE mode may be selected by depressing the VEL CWS button on the Mode Control Panel (MCP). It is also the primary default mode when AUTOE is lost, Autoland modes are lost or Go-Around (GAS) is selected. The VCWSE mode is engaged when:

- 1) VEL CWS is depressed (VCWSS true) when VCWSE is false and AEE is true;
- 2) ATT CWS is depressed (ACWSS true) when ACWSE is true;
- 3) AUTO is depressed (AUTOS true) while in AUTO mode;
- 4) Go Around is commanded (GAS is true) with AEE true;
- 5) LANDE goes from true to false while in AUTO mode;
- 6) AUTO or LAND second fail occurs with AUTOE true.

Disengagement occurs when:

- 1) AEE is lost, causing reversion to PRENG;
- 2) ATT CWS or VEL CWS is depressed causing reversion to ACWSE;
- 3) AUTO is depressed while in VCWSE Mode (AUTOE becomes true);
- 4) VCWS second fail occurs, causing reversion to ACWSE.

AUTOE (AUTO Engage Mode):

The AUTO mode is a precondition for the higher modes: AUTOLAND, DECRB, FLARE and RLOUT. It can be set true only by pressing the AUTO button on the MCP. The initial criteria are:

$$\text{AUTOE} = \text{AEE} \cdot \text{AUTOS} \cdot \overline{\text{AUTOE}} \cdot \text{DETNT}(\text{DZNE}, \text{WDZNE}) \cdot \overline{\text{FFDS} + \text{FAIL2}(\text{AUTO})}$$

Where: DZNE = selected column dead zone;
WDZNE = selected wheel dead zone.

Subsequently, AUTOE mode will be disengaged if:

- 1) Another mode is selected;
- 2) GAE becomes true, causing reversion to VCWSE;
- 3) AEE becomes false, causing reversion to PRENG;
- 4) DETNT(0.78, 8.0) becomes false, causing reversion to VCWSE;
- 5) AUTO is selected on the MCP when in AUTOE mode, causing reversion to VCWSE;
- 6) An engaged guidance mode is deselected on the MCP when in AUTO mode, causing reversion to VCWSE (AUTOS is set by MSPLGC);
- 7) FAIL2(AUTO) becomes true, causing reversion to VCWSE;
- 6) LANDE becomes false, having been true, causing reversion to VCWSE.

LANDE (Land Engage Mode):

There are presently two variants of LANDE: Autoland (AUTOE is true) and LAND guidance for RFD CWS modes. The criteria for engagement and retention of LANDE in CWS and AUTO differ. Common processing is discussed first.

LANDR (LAND Ready) is set by:

$$\text{LANDR} = \text{LANDS} \cdot \overline{\text{LANDR}} \cdot (\text{LOCVLD} + (\text{MLSM} \cdot \text{MLSMOD}) + \text{GPLNDV}) \cdot (\text{ILSZON} + \overline{\text{AUTOE}}) \cdot \overline{\text{FAIL2}(\text{LAND})}$$

If LANDS does not result in LANDR, LNDCTR is set to 160, which causes the blue light in the LAND select button to be lit for 8 seconds.

Once set, LANDR is cleared by:

- 1) Selecting LANDS when LANDR is true;
- 2) Selecting HORPTH or TKSEL when LANDR is true (LANDS is set by MSPLGC);
- 3) MODEX becoming less than 4 (ACWS);
- 4) ILSZON becoming false with AUTOE true;
- 5) FAIL2(LAND) becoming true;
- 6) GAE becoming true.

Criteria for engagement of the lateral and vertical beams are as follows:

- 2) LOCA (Localizer Armed):

$$\text{LOCA} = \text{LANDR} \cdot (\text{MLSM} + \text{LOCVLD}) \cdot \overline{\text{LOCE}}$$

Where: LOCE is set by LATCMD once LOCA is true and other criteria are met.

- 3) GSARM (Glide Slope Armed):

$$\text{GSARM} = \text{LANDR} \cdot \text{GSEENABL} \cdot (\text{MLSM} + \text{GSVLD}) \cdot \overline{\text{GSENG}}$$

Where: GSENG is set by VERCMD once GSARM is true and other criteria are met.

If AUTOE is false with LANDR true, LANDE is set as soon as LOCE and GSENG are both true. Neither ONCRS nor GSTRK are allowed to become true, so the higher land modes (DECRB, FLARE, RLOUT) are never enabled. Also, glideslope guidance can be turned off (GSEENABL is set false) without loss of LANDE by selecting any other vertical guidance mode (FPA, 3D or ALTSEL) or by loss of GSVLD. Once GSEENABL is cleared, it can only be reset by deselecting, then re-selecting LAND. If AUTO is selected with LANDE true, LANDE, LOCE and GSENG are cleared to permit a smooth automatic capture of the path.

THE AUTOLAND MODES:

When AUTO mode is established, MLOG evaluates the criteria for the Autoland mode. There are several submodes or conditions required for Autoland control. These are outlined below:

4) LANDE (Land Engage Mode):

$$\text{LANDE} = \text{LANDR} \cdot \text{LOCE} \cdot \text{ONCRS} \cdot \text{GSENG} \cdot \text{GSTRK}$$

Where: LOCE and ONCRS are set by LATCMD, and GSENG and GSTRK are set by VERCMD.

When LAND mode is established, the autoland modes are entered in sequence as their associated logic is satisfied. At 150 feet altitude, DECRB mode is activated to align the aircraft with the runway heading. At 42 feet (HTDZ) FLARE mode raises the nose of the aircraft to establish a landing attitude. Finally, the Rollout (RLOUT) mode maintains a course down the runway centerline as the aircraft rolls to a stop. Each of these submodes is latched once set until loss of LANDE. If Go-Around (GAS) is selected, the autoland progression terminates and control reverts to VCWSE with a 2 degree fly-up bias. The logic for each mode follows:

1) DECRB (Decrab Mode):

$$\text{DECRB} = \text{LANDE} \cdot \text{AUTOE} \cdot ((\text{H} < 150.) + \text{DECRB})$$

2) FLARE (Flare Mode):

FLARE criteria are evaluated in ELEVP once DECRB is true. FLARE is set when (HRAD or HTDZ) becomes less than 42 ft.

3) RLOUT (Rollout Mode):

$$\text{RLOUT} = \text{DECRB} \cdot ((\text{WSPIN} \cdot \text{SQUAT}) + \text{RLOUT})$$

DEFAULT OR SPECIAL MODES:

MANEL (Manual-Electric Mode):

This is not a normal flight mode. Rather, it is a 'utility' mode used for development and checkout. MANEL permits the RFD pilot to hand-fly the aircraft with computer assistance limited to 'translation' of the wheel/column inputs. It is entered by raising the MANEL toggle switch on the flight controls pallet. It will then engage only if the AFD paddle is raised on the FFD Control & Command panel and no other mode is selected. The toggle switch enables MANEL in lieu of ACWS. If another mode is selected while in MANEL mode, MANEL will terminate. It will also terminate if the toggle is lowered when no other mode is selected. Control then defaults to ACWS.

The MANEL enabling logic is:

$$\text{MANEL} = \text{AEE} \cdot (\text{ACWSS} + (\text{MODEX} = 0)) \cdot \overline{\text{FFDS} + \text{AGCSS}} \cdot \\ ((\text{MANEL} \cdot \overline{\text{FAIL2}(\text{MANEL})}) + \text{DETNT}(\text{DZNE}, \text{WDZNE}))$$

PRENG (Pre-Engage Mode):

This is the ultimate default when all else fails. It is entered whenever conditions for "higher" modes are not satisfied or when the engage paddles will not remain in the raised position.

In mode logic final processing, if no mode has been set, AEE is cleared, MODEX is set to 1 (PRENG), and the variable DELAY is set to four to allow the 200 Msec intermission for the engage paddle to drop. Some or all of the mode discrettes (FLAGS) are cleared as appropriate and, if VCWS mode was not selected, the Go-Around flag (GAE) is set false. Next, the FLAGS array is set to reflect the selected mode. The basic flight mode flags (PRENG to AUTOE) are mutually exclusive, so all are cleared, then one is set per MODEX. The flags associated with LAND mode (LANDE thru ONCRS) are cumulative and once set are not cleared until the conditions required for LAND mode are lost. Last, the LBS and VBS flags are cleared if no longer valid, and the Computational Reset (CRSET) and Error Reset (ERSET) discrettes are set upon exiting or entering PRENG, respectively. CRSET clears the FAIL2 array and causes a "RESET" message to be stored in the message buffer, and ERSET clears the signal failure flags and the 'Failure Read' and 'Status Alert' lamps on the System Test Panel.

BIT/ARRAY/INDEX ALIGNMENTS:

VALUE/INDEX	MODEX	FAIL2/DSPLF	FLAGS
0	UNDEF	---	---
1	PRENG	AFCS	PRENG
2	FFDE	FFD	FFDE
3	MANEL	MANEL	MANEL
4	ACWS	ACWS	ACWSE
5	VCWS	VCWS	VCWSE
6	AUTO	AUTO	AUTOE
7	LAND	LAND	LANDE
8	DECRB	AUTOTHROTTLE*	DECRB
9	FLARE	MLS*	FLARE
10	RLOUT	---	RLOUT
11	---	---	LANDA
12	---	---	LOCA
13	---	---	LOCE
14	---	---	GSARM
15	---	---	GSENG
16	---	---	LANDR
17	---	---	GSTRK
18	---	---	ONCRS

* Entries 8 & 9 of FAIL2/DSPLF are used to record autothrottle and MLS failures, but this has no relationship to MODEX or the flight modes.

GLOBAL INPUTS: ACWSS, ACWSE, AEE, AFCSS, AFCV, AGCSS, ALTARM,

ALTHLD, AUTOE, AUTOS, AZ, AZ BRG, BDCOL, BDWHL, DECRB,
DFPSEL, DLPSI, DSTAT, D3D, EL1, FAIL2, FCOL, FFDS, FLAGS,
FLARE, FPASEL, FWHL, GAE, GAS, GPINIT, GPLND, GPLNDV,
GPSBTA, GPSETA, GSA, GSENG, GSTRK, GSVLD, HGPIP, HOLD,
HOLDM, HRAD, IC, ILSZON, KDCOL, KDWHL, LABFLG, LANDE,
LANDR, LANDS, LFCFLG, LNDCTR, LOCE, LOCVLD, MANEL, MCONE,
MLSM, MLSMOD, MLSVLD, MODEX, MSWIT, ONCRS, PRENG, GSDEV,
RLMLS, RLOCDV, RLOUT, RUN, RUNM, RWYHDG, SDCOL, SDWHL,
SIMILS, SQUAT, SSTICK, TANGSA, TRKHLD, VCWSE, VCWSS,
VERARM, WSPIN, ZHAT

GLOBAL OUTPUTS: ACWSS, AEE, ALTARM, AUTOE, CRSET, DCOL, DSPLF,
DWHL, DZNE, ERSET, ETAFT, ETAVL, FCFLGS, FLAGS, FLAGWD,
GAE, GPGSDV, GPLOCD, GSARM, GSDEV, GSENG, HOLDM, HTDZ,
ICM, LANDA, LANDE, LANDR, LBS, LFCFLG, LNDBLU, LNDCTR,
LOCA, LOCDEV, LOCE, LOCVL, MLSM, MODE2, MODEX, MSW1,
MSW6, RUNM, TANGSA, VBS, VCWSE, VCWSS, WZNE, YPROF

164

MODULE NAME: DETNT (Logical Function)
FILE NAME: MLOG.FOR
PROCESS: FCFAST

PURPOSE: To verify wheel and column in detent and to output an advisory message if not.

CALLED BY: MLOG
CALLING SEQUENCE: IF DETNT(Col_lmt, Whl_lmt) ...
CALLS TO: FMTMG

DESCRIPTION:

The DETNT function compares DCOL to the formal parameter COLMT and DWHL to WHLMT. If abs(DCOL) is greater than COLMT, the Boolean DETNT is returned false and FMTMG is called to output the message "COL OUT OF DETNT". If abs(DWHL) is greater than WHLMT, the Boolean DETNT is returned false and FMTMG is called to output the message "WHL OUT OF DETNT". If both are within limits, DETNT is returned true.

GLOBAL INPUTS: DCOL, DWHL, TEST3A

GLOBAL OUTPUTS: None

MODULE NAME: LATCMD (Lateral Path Command)
 FILE NAME: LATCMD.FOR
 PROCESS: FCFAST

PURPOSE: To compute the Lateral Steering Command for AUTO flight modes and VCWS Track Hold mode.

CALLED BY: FCFAST
 CALLING SEQUENCE: CALL LATCMD
 CALLS TO: ANGL, ATAN, CMPF, SCOSD

DESCRIPTION:

This module computes the steering command (BACMD) for VCWS Track Hold mode and AUTO mode Track select, 2D (HORPTH), and localizer guidance. Horizontal path guidance is addressed first. Processing begins by calculating the variable path tracking gain (KALT - a misnomer) which may be used for both horizontal and vertical path tracking once the path has been acquired. KALT varies from 0.7 at 215 knots CAS to a maximum of 2.0 (in MLS mode) at 140 knots. If GUID2D is false (2D guidance is not available), KALT is set to 1.0, the Be Careful Flag (BCFLAG) is set, the FWDPTH integrator and the 2D steering command (LATSTR) are zeroed and control passes to the Autoland computations. Otherwise processing continues based on the condition of the OLDLAT flag.

HORPTH COMPUTATIONS

Two steering algorithms are coded, differing primarily in the path capture algorithm. Both make use of the relationship between ground speed (GSFPS), bank angle (PHI) and the radius of turn (R):

$$R = \text{GSFPS}^2 / g \tan(\text{PHI})$$

where: g is the acceleration due to gravity (32.1739 fps/s).

The first coded (selected with OLDLAT true) is a modification of the original Boeing algorithm (specified in the "Terminal Configured Vehicle B-737 Navigation Computer Software Description", D6-32686), with the modification intended to reduce the time required to acquire the track. Although it uses the higher nominal bank angle (20 degrees), it's capture is more gradual since it makes the simplifying assumption that delta track angle is a linear function of cross-track distance (XTK). The mandatory capture point (XTKLIM) is computed as:

$$\text{XTKLIM} = (\text{abs}(\text{TKE}) / 60.) * \text{GSFPS}^2 / 11.71$$

Where: TKE is limited to the range 10 to 135 degrees;
 TKE / 60 is the approx radian value of TKE;
 11.71 is g tan(20).

If abs(XTK) is less than XTKLIM, BCFLAG is cleared, which forces the transition from Hor Path Armed to Hor Path Engaged (done in MSPLGC). This is significant only when the pilot has specified the

approach angle via the track knob, as HORPTH is otherwise set immediately.

The limiting course cut angle (TMP1) is then computed as:

$$\text{TMP1} = 702.6 * \text{abs}(\text{XTK}) / \text{GSFPS}^2$$

$$\text{Where: } 702.6 = 60 * g \tan(20),$$

with the result limited to a min of 10 and a max of 90 degrees.

The steering command (LATSTR) is the solution of a linear second order differential equation,

$$\text{LATSTR} = \text{KY} \text{ dY} + \text{KYD} \dot{\text{Y}} + \ddot{\text{Y}}$$

implemented as follows. The commanded Y dot (KY dY, labeled TMP2 in the code) is computed as XTK * KY and limited by a Y dot limit computed from the course cut limit (TMP1) as follows:

$$\dot{\text{Y}}_c = \text{LIM}(\text{KY} * \text{XTK}, \text{TMP1} * \text{DTOR} * \text{GSFPS} * \text{KYD})$$

This is then summed with the approximation of Y dot, where

$$\dot{\text{Y}} = \text{TKE} * \text{DTOR} * \text{GSFPS} * \text{KYD}$$

and limited again (see below) before adding the Y acceleration term (in the form of the nominal bank angle required to fly a curved path (ALCBA) computed by HVGUID).

$$\text{LATSTR} = \text{LIM}(-(\dot{\text{Y}} + \dot{\text{Y}}_c), 50) + \text{ALCBA} + \text{FWDPTH}$$

where FWDPTH is the limited integral term computed when AUTOE and HORPTH are true, XTK is less than 1000 feet and TKE is less than five degrees. FWDPTH is limited to five degrees bank angle contribution.

$$\text{FWDPTH} = \text{LIM}(\text{FWDPTH} - \text{KI} \text{ dT} \dot{\text{Y}}_c, 5.)$$

The purpose of the Forward Path integrator is to improve the tracking accuracy of Horizontal Path guidance by nulling the cross path stand-off error. The time constant (1 / KI) is ~80 sec. Note: the 50 degree limit in the computation of LATSTR above was a scaling limit designed to prevent overflow in the old fixed point code. It isn't really necessary in the present implementation.

The second algorithm coded is also a modification of the Boeing algorithm, designed to correct several of its deficiencies: the slow capture sequence (partially corrected by the above), the unpleasantly abrupt control response when HORPTH is selected or when a switch in position source information occurs (e.g, the RNAV/MLS transition), and a complaint that the control law is overly aggressive at cruise speeds, while being marginally 'tight'

at approach speeds.

The capture sequence describes a circle of radius R computed (as above) using a nominal bank angle of 15 degrees. This value was chosen since 15 degrees is the value used for curved paths by the path definition equations, and it was desired that the path flown be indistinguishable from that that would be flown if the CDU were used to define the course cut to the desired path. Given a course cut of angle TKE and a circle (of radius R) tangent to both the defined path and the approach course, then the distance of the tangent point of the approach course from the path (XTK) is defined by the equation

$$XTK_{\text{tanpt}} = (1 - \cos(\text{TKE})) * R$$

$$\text{Where: } R = \text{GSFPS}^2 / g \tan(15)$$

As with any transition to a curved segment, the turn initiation must precede the tangent point by the time required to roll up to the nominal bank angle. Unlike the calculation for ALCBA in HVGUID (which assumes the initial bank angle is zero), the initial bank angle here is unknown (as the aircraft might be turning towards the path when the tangent point is reached) and must be accounted for. The equation used is

$$XTKLIM = XTK_{\text{tanpt}} - Y \text{ sign}(XTK) * \text{Advance_time}$$

$$\text{Where: } \text{Advance_time} = (15 - \text{PHICMD} * \text{sign}(XTK)) / 3.5$$

A minimum value of 250 ft is imposed on XTKLIM.

This doesn't completely handle the problem. In order to ensure a smooth ogee during a capture from a parallel path less than two radii from the desired path, the 'TKE' used in the above equations is advanced (prior to initiation of the turn onto course) from that calculated by HVGUID by Track dot times a function of bank angle:

$$\text{TKE}(\text{local}) = \text{TKE} + \text{TKDT} * \text{abs}(\text{PHICMD}) / 6$$

When XTK becomes less than XTKLIM, the LCAP flag is set, which starts the circular capture sequence and switches off the Track advance calculations. On a directed approach to a straight path, BCFLAG is also cleared at this time. The above was a later addition to the initial design. It now appears that these equations could be cost reduced slightly by removing the initial bank angle adjustment from the XTKLIM equation and adjusting the TKE advance time to compensate.

During the circular portion of the capture of a straight path, a separate control law is used to generate the steering command. It computes the bank angle required to remain on a circle tangent to the desired path by using the inverse of the equation used to compute XTKLIM:

$$\text{LATSTR} = \arctan (\text{CIRCLD} / \text{XTK})$$

$$\text{Where: } \text{CIRCLD} = \text{GSFPS}^2 * (1 - \cos(\text{TKE})) / g$$

The actual equation replaces XTK with $(\text{XTK} - 25 \text{ sign}(\text{XTK}))$, to avoid overflying the path during roll out. (This is, of course, gilding the lily, since if a defined path is flown, the aircraft does diverge to the outside of the path during the roll out at the end of the turn.) This equation is used only during capture of a straight path segment. When TKE is reduced to 8 degrees (the point at which the output from the two equations is approximately equal), control is relinquished. At all other times (including capture of a curved path) the standard tracking algorithm is used. When approaching the path from 'far away', a fixed 90 degree course cut limit is used with the standard tracking algorithm until LCAP is set.

Neither algorithm does a creditable job of capturing a curved path without pilot assistance, but the second algorithm will make a smooth capture in the directed capture (track angle selected) mode by delaying the clear of BCFLAG until the tangent point of the turn is reached.

The only other modification to the tracking algorithm involves the use of KALT to vary the gain. KKYD , the gain on KYD , is set to zero when HORPTH is false or when a navigation mode switch occurs ($\text{PNMOD} \neq \text{NAVMOD}$). It is then brought back up to KALT through a 4 second easy-on. KKY , the gain on KY , is then computed as the square of KKYD . An increase in gain (if any) is not permitted until the path is acquired, as KALT is limited to 1.0 until PTHTRK is true. (PTHTRK is set when XTK becomes less than 500 feet and TKE is less than 2 degrees). Maximum gain-up (2.0) is permitted only if MLS mode is true. This is because (a), we're less likely to know where we really are in RNAV mode and (b), the MLS autoland equations have no capture algorithm and thus are dependent on a precise delivery to the runway centerline by the 2D guidance equations for a stable 'loc' capture. The PTHTRK integrator is gated by PTHTRK and a closing track angle of less than three degrees.

Should a mode switch or path update occur (HORPTH is lost), or the path be overflowed due to a zero radius turn or excessive ground speed, the control flags and integrators are cleared to force a recapture sequence (approach, circular capture and track).

AUTOLAND COMPUTATIONS

This section begins by computing the beam capture limit (LOCVL) for MLS mode. (LOCVL for ILS mode is still computed in MLOG , but probably should be pulled into here for consistency). For MLS mode, the beam capture window is funnel shaped: a constant ~30 feet at less than 1.5 nm (XHAT , measured from the Az site), 0.2 degrees from 1.5 to 3.0 nm, then increasing exponentially to 0.8 degrees at 6.2 nm. The offset from runway centerline (DELTY , the MLS equivalent of ILS ETAFT) is then computed as $-(\text{YHAT})$ plus the offset of the Az antenna from the runway centerline (YPROF , nominally zero) and limited to LOCVL . If Localizer Armed (LOCA) is true (Land mode has been selected), a check is made to see if the Lateral Beam Sensed (LBS) flag can be set. This is set when DLPSI

(A/C heading - RWY heading) is less than 90 degrees and DELTY is less than (LOCVL - 2.67). Localizer Engaged (LOCE) is not set until the aircraft is rolled out on track. I.e, when bank angle is less than 3 degrees and beam rate is less than .027 dps.

If MLSM is false with LOCA true (ILS mode), LBS and LOCE are set simultaneously when DLPSI is less than 90 and LOCDEV is less than LOCVL (computed in MLOG as a function of HRAD). For both ILS and MLS mode, if AUTOE and LOCE are both true the localizer Complementary Filter subroutine (CMPF) will be called. If MLSM and ONCRS are both false, checks will be made first to see if it is time to set the ONCRS flag. This will be done 5 seconds after the criteria (LOC dot < .027 dps) and (LOCDEV < LOCVL) and (ROLL < 3 deg) are met, where LOCVL is recomputed for ONCRS requirements and LOC dot (RF4) is computed by passing LOCDEV through a .12 second lead-lag filter (RF3), limiting the output to 1.0 dps and passing that through a 1.24 sec lag filter. If either LOCA or AUTOS is true (indicating first pass of AUTOE or LOCE), RF3 and RF4 are initialized and the timer is set for 5 seconds. Once the criteria for On Course are met, the timer is counted down and ONCRS set when it reaches zero. If for any reason the On Course requirements are lost before ONCRS is set, the timer is reset to 5 seconds. For MLS mode, ONCRS is set by CMPF approximately 3 seconds after LOCE.

TRACK ANGLE COMPUTATIONS

This section begins by passing TK through a one second lead-lag filter to form Track dot (TKDT), used in the track angle gain-up logic and also above in the 2D path capture look-ahead logic. The basic track angle gain (KTA) is then computed as a linear function of ground speed (.00475 GSFPS) with a lower limit of 1.0. The next section of logic permits the pilot to select a track change of up to 300 degrees in the same direction when in the Auto/Track select mode. Since angles are normally carried as 0 +/- 180 degrees, the aircraft would reverse direction when the delta track was dialed through 180 degrees without this logic. Bank angle is monitored in this logic so that it is not active in track preselect or VCWS/Track select mode. Thus if a track change of more than 180 degrees is first selected and then Auto/Track select engaged, the aircraft will take the shortest path around. DTKF is the 'adjusted' filtered track angle which is ultimately sent to the Displays computer as TKREL. Its equivalent in VCWS Track Hold mode is TRKBG. In interim processing, TKREL is used to contain the unfiltered 'adjusted' delta track (DELTKA in Auto mode) used in the track mode steering algorithm. (This code should be rewritten to make it less confusing).

If TRKHLD is true (VCWS Track hold mode), TKREL is set to VCWS delta track (HLDTRK - TK) and TRKBG is computed by passing TKREL through a 0.2 sec lag. Otherwise, HLDTRK is synchronized to TK, TRKBG is zeroed (which causes the track bug to disappear from the PFD) and, if AUTOE, TKREL is set to DELTKA. If neither AUTO/TKSEL nor TRKHLD, TKREL is zeroed. Otherwise, both a speed (CAS) dependent gainup (KTGA) and an error integrator (DTKI) are operated once the track has been acquired. KTGA is computed in either AUTO/TKSEL or VCWS/TRKHLD once delta track is less than four

degrees and track dot is less than 0.5 dps. Once track dot is less than 0.2 dps, DTKI is computed as the integral of delta track, with a 10 second time constant and a four degree limit. DTKI is washed out whenever TKSEL and TRKHLD are both false or delta track is greater than three degrees. (DTKI is also cleared when TKSEL and PSTTKA are both false, but this is an error). KTGA is reset to 1.0 if the track knob is turned (to prevent an abrupt response to a large delta track input) and washed out to 1.0 if a track change command is made via the lateral trim switch (VCWS). The time constants were chosen to make the gain up / gain down transparent to the pilot.

The steering command for track angle modes is computed as

$$\text{TEMP} = \text{KTA} * \text{KTGA} * (\text{TKREL} + \text{DTKI}).$$

A check is then made to determine the active guidance mode. If in VCWS, only track angle (TRKHLD) is available. If in AUTO mode then, if LOCE is true, TEMP is set to LOCCMD from the CMPF subroutine or, if HORPTH is true, TEMP is set to LATSTR from the 2D steering equations. TEMP is then limited to 30 degrees and set into BACMD, which is the input to the LATRL procedure.

GLOBAL INPUTS: ALCBA, AUTOE, AUTOS, BCFLAG, CAS, CTKE, DELTKA, DLPSI, GSFPS, GSFPS2, GUID2D, HLDTRK, HORARM, HORPTH, HTDZ, ICM, KTKREL, LATSTR, LOCA, LOCCMD, LOCDEV, LOCE, LOCVL, MLSM, NAVMOD, OLDLAT, ONCRS, PHICMD, PNAV64, PSTTKA, ROLL, RTKNOB, RTN, STKE, TK, TKASUM, TKE, TKSEL, TRKHLD, TRKIN, XHAT, XTK, YDH, YPROF

GLOBAL OUTPUTS: BACMD, BCFLAG, CTKE, DELTKA, DELTY, HLDTRK, KALT, LATSTR, LBS, LOCE, LOCVL, ONCRS, STKE, TKASUM, TKREL, TRKBG, XTKLIM

MODULE NAME: CMPF (Localizer Complementary Filter)
 FILE NAME: LATCMD.FOR
 PROCESS: FCFAST

PURPOSE: To compute the Lateral Steering Command for AUTOLand mode.

CALLED BY: LATCMD
 CALLING SEQUENCE: CALL CMPF
 CALLS TO: None

DESCRIPTION:

Since this routine is called only if AUTOE and LOCE are both true, first pass processing is indicated if either AUTOS or LOCA is true. (LOCA is cleared on the iteration after LOCE is set, and AUTOS is true for one iteration when the AUTO button is pushed and released, with AUTOE being set in MLOG prior to execution of this module. As implemented, full initialization is only performed if LOCA is true. This is an error). If first pass, the filters and integrators are initialized. Otherwise, processing continues according to the state of the MLS mode flag (MLSM).

In MLS mode, no filtering is done on the position error signal (DELTY) since it was derived from complementary filtered YHAT. The proportional signal (LOCCF) is set to .05556 DELTY until RLOUT, when it is gained up as a function of XHAT to improve roll out tracking. Since LOCCF is not filtered, and since MLS Autoland demands 2D guidance to the runway centerline, the transition from 2D to Autoland is smoothed by a three second easy-on between BACMD and LOCCMD. (This was correct when CMPF was a subprocedure of LATRL. With CMPF contained in LATCMD, this easy-on would work better if LATSTR were used instead of BACMD).

LOCINT, the integral of LOCCF, is computed once ONCRS is set (when LOCGAIN reaches .98 -- approximately three seconds after LOCE), with a limit of 20 fps. LOCINT is used to balance out any bias in XTVEL and to improve beam tracking. The damping term (SEL_XTV) is computed as .98634 times the Y dot output of the MLS filter (YDH).

In ILS mode, ETA is calculated as .05556 ETAFT (calculated by MLOG as 720. GPLOCD, thus $ETA = 40. GPLOCD$). Prior to ONCRS, 0.5 ETA is passed through a two second lag filter to produce LOCCF, and SEL_XTV is set to .7052 XTVEL. Once ONCRS is set, ETA is complemented by 1.094 XTVEL (Close. $.05556 * 20 = 1.111$) before being passed through a 20 second lag filter. SEL_XTV is then set to .98634 XTVEL and the LOCINT integrator is operated.

In common code, the XTKDMP filter is IC'd to SEL_XTV if LOCA is true, and computed as a 0.4 second lag on SEL_XTV otherwise. LOCCMD is then computed as

$$LOCCMD = 2. * LOCCF + LOCINT - SEL_XTV.$$

Note that in ILS mode, given the initialization value of LOCCF

$$(.7052 * XTVEL + PHICMD) / 2 ,$$

the first pass output of LOCCMD will be essentially equal to the previous iteration's value of PHICMD. In MLS mode, it might differ markedly, thus LOCCMD is passed through a three second easy-on.

GLOBAL INPUTS: AUTOE, AUTOS, BACMD, DELTY, ETAFT, LOCA, MLSM,
 ONCRS, PHICMD, RLOUT, XHAT, XTVEL, YDH

GLOBAL OUTPUTS: LOCCMD, ONCRS

MODULE NAME: LATRL (Lateral Axis Control Laws)
FILE NAME: LATRL.FOR
PROCESS: FCFAST

PURPOSE: To calculate aileron and rudder commands for the flight control system.

CALLED BY: FCFAST
CALLING SEQUENCE: CALL LATRL
CALLS TO: FRCWS, RCOM, RCOMA, RBASC, RBASCA

DESCRIPTION:

LATRL is the executive which performs common processing and calls the other procedures which make up the lateral axis control laws. Various combinations and levels of automatic flight control assistance are available to the Research flight deck (RFD) and Forward flight deck (FFD) pilots. The lateral control laws for each mode are depicted in the DSDs found in Appendix A. Possible modes are listed below, with a brief summary of their capabilities and the subroutines involved.

FLIGHT MODES (See also procedure MLOG).

Pre-engage (PRENG): The autopilot is disconnected. Processing is limited to system initialization which is performed by LATRL mainline code.

Forward Flight Deck Control Wheel Steering (FFDE): Aileron command processing is handled entirely within subroutine FRCWS. Rudder control is direct from the FFD pedals. Submodes are Attitude Sync and Attitude Hold.

Manual Electric (MANEL): RFD wheel input is applied directly to the aileron command by LATRL main line code. RFD rudder pedal and aileron trim inputs are processed by subroutine RBASC.

Attitude control wheel steering (ACWSE): This is a rate command / attitude hold system with automatic turn coordination. Roll attitude command is computed by subroutine RCOM with the aileron command computed in mainline code. Rudder pedal, aileron trim and turn coordination are processed by subroutine RBASC. Submodes are Attitude Sync and Attitude Hold, with an automatic wings leveler applied if the commanded bank angle is near zero.

Velocity control wheel steering (VCWSE): See notes for ACWSE. This is a rate command / velocity vector hold system. Submodes are Attitude Sync and turn radius hold, with track hold entered if the commanded bank angle is near zero. Track hold mode is essentially identical to AUTO/Track Select mode and uses the bank angle command (BACMD) generated by procedure LATCMD.

Automatic (AUTOE): This is a rate limited bank angle command system with automatic turn coordination. Small wheel and pedal

inputs are also permitted but a large (> 25%) wheel input causes reversion to VCWSE. The bank angle command (BACMD) is input from procedure LATCMD and entirely processed within subroutine RCOMA. Aileron trim, rudder inputs and turn coordination are processed in procedure RBASCA. Submodes are Track Select, Horizontal Path and Autoland (LANDE), with LANDE being further subdivided into acquisition, track (ONCRS), Decrab (DECRB), FLARE and roll-out (RLOUT). The rudder and aileron contributions for the Decrab maneuver and roll out guidance are computed by subroutine DCRAB.

Note: all flight modes except PRENG and MANEL use filtered roll rate (PF1) feedback for roll stabilization. PF1 is computed in subroutine OUT10M of the HDL process by passing body roll rate (P) through a .05 second lag filter. In ACWS and VCWS, PF1 is also summed with AILCMP to produce AILCMD in OUT10M, using the feedback gain (KP) calculated by LATRL. In FFD and AUTO modes, roll rate feedback is applied within subroutines FRCWS and RCOMA, respectively.

The sub-procedures are:

FRCWS Forward flight deck Control Wheel Steering
 RCOM Roll command Computer (CWS)
 RCOMA Roll command Computer (Auto)
 RBASC Basic trim and Rudder command computations (CWS)
 RBASCA Basic trim and Rudder command computations (Auto)
 DCRAB Decrab flight mode calculations (called by RCOMA)

LATRL begins by checking for initialization conditions. If the Flight Controls process is in IC Mode or if the flight mode is Pre-engage (PRENG), the aileron command (AILCMP) is synchronized to aileron servo position (ALVDT), rudder command (RUDCMD) is synchronized to rudder position (DRPOS), roll rate feedback gain (KP) is zeroed and the aileron trim command (SYNCL) is initialized to the gained and limited research flight deck (RFD) aileron trim knob input (ATRIM) plus ALVDT as given:

$$\text{SYNCL} = \text{SLIM}(1.52 \text{ ATRIM}, 10.) + \text{ALVDT}$$

Note that the RFD cab does not have an aileron trim mechanism, therefore the aileron trim knob command is brought into the FM/FC computer separately and is summed with the aileron command. The aileron trim command is initialized to zero at auto-pilot engagement regardless of aileron trim knob position. This is accomplished by including the aileron trim command in SYNCL during PRENG, then subtracting SYNCL from the aileron command in all engaged modes. If in IC Mode, all first pass flags are cleared

(which will cause all filters to be IC'd) and the roll rate filter (PF1) is zeroed. Otherwise (not IC but PRENG), the rudder/aileron easy-on (RASWIT) is set to 0.5 (1.0 would be better) to smooth the transient when the RFD is engaged.

Next, the raw control wheel input (DWHL) is gained and dead-zoned to produce WHLINP and the wheel-out-of-detent flag (RCWOD) is set according to whether the deadzone (WDZNE) has been exceeded. If MODEX is less than 4, i.e, flight mode is PRENG, FFDE or MANEL, the displayed roll signal (DROLL) is set equal to actual roll angle (ROLL). Otherwise, if the side stick is selected or LABFLG is true, WHLINP is exponentially shaped (conditionally) and limited to a maximum of 12.5 degrees. The spoiler feedback inhibit (SPFINH) and radius hold (RADHLD) flags are then cleared.

The next section handles mode specific processing. If FFDE is selected, FRCWS is called to compute the aileron command. (The rudder is actuated directly by the FFD rudder pedals without computer aiding). If PRENG, no further processing is required. If AUTOE, RCOMA and (if RUN Mode is true) RBASCA are called to compute the aileron and rudder commands, respectively.

For the CWS modes, the aileron command is computed in line: If MANEL, KP is set to zero, RBASC is called to compute the RFD aileron trim knob contribution (AILTRM) and the rudder command (RUDCMD), and AILCMP is computed as the negative of WHLINP plus AILTRM minus SYNCL. The negative is used to accomodate aircraft wiring polarity which is positive left wing down. For ACWS and VCWS modes, RCOM is called to compute the bank angle command (PHICMD) and bank angle error (PHIERR) and (if RUN Mode) RBASC is called to compute the rudder command and aileron trim contribution to the aileron command. Next, if the aircraft is in the air (GRD is false) the basic inner loop gain (KP1) is computed as a function of flap setting, with a minimum value of 1.41 and maximum value (KPLIM) of 1.73. Otherwise, KP1 is fixed at 1.41 to minimize aileron activity on the ground. The roll rate feedback gain (KP) is then calculated based on bit selections in the LATSEL word. (Note: during program development in the real time simulator KP = KP1 worked best, but this caused problems in the air. We have been using $KP = KP1 * KVCAS$ but there have been some indications that this is still too high. We should test the option $KP = KP1 * KV$ (which is what was used by the original control law) sometime soon so that the selection logic can be removed.)

The intermediate aileron command (AILCD1) is next computed as $KP * KP1 * PHIERR$. If in Velocity CWS mode and HTDZ is less than 300 feet and FLAP is greater than 20 (i.e, landing configuration), the aileron cross-feed term (AILXFD) is added to AILCD1. The final aileron command is first loaded in TEMP as

$$-(AILCD1 + KWHLF4 * WHLINP + AILTRM - SYNCL)$$

TEMP is then switched into AILCMP by RASWIT, which mechanizes a one second easy-on to minimize transients when switching modes. This same variable is also used to switch between sources for RUDCMD (in RBASC and RBASCA).

Final processing for the higher flight modes computes the

displayed roll signal (DROLL) and the nominal aileron command gain (KAILG - applied by OUT10M). If GRD is true or ACWSE, VCWSE and TKSEL are all false, DROLL is set equal to ROLL and KAILG is set to 1.0. Otherwise, DROLL is computed as a 0.3 second lag on DRIN, which is set to commanded bank angle (PHICMD) when turning and is washed out to zero when in a track hold mode and the track has been acquired. This wash out is to prevent turbulence from appearing in the curved trend vector when in a track mode. KAILG is still normally set to 1.0, but logic is implemented to compute a lower gain at a flap setting of 30 or 40 degrees. This is test code intended to address a problem of aileron oscillation at high command rates at the higher flap settings.

LATRL processing completes by updating the first pass flags. These flags are set to 2 by each procedure which uses them when those procedures are called. This code subtracts 1 from each flag which is non-zero. If the pertinent flag is found zero by a controlled procedure, it means that that procedure did not execute on the previous iteration and initialization is required.

GLOBAL INPUTS: ACWSE, AILCMP, ALVDT, ATRIM, AUTOE, DRPOS, DWHL, FFDE, FLAP, FLPPF, GRD, HTDZ, ICM, KPLIM, KTDROL, KV, KVCAS, KWHLF4, LABFLG, LATSEL, MANEL, MODEX, PHICMD, PHIERR, PRENG, ROLL, RUNM, SSTICK, SYNCL, TKSEL, VCWSE, WZNE, WHLSHAP

GLOBAL OUTPUTS: AILCMP, DROLL, KAILG, KP, PF1, RADHLD, RUDCMD, SPFINH, SYNCL, WHLINP

MODULE NAME: FRCWS (Forward Flight Deck Roll CWS)
 FILE NAME: LATRL.FOR
 PROCESS: FCFAST

PURPOSE: Compute the aileron command for FFD CWS mode.

CALLED BY: LATRL
 CALLING SEQUENCE: CALL FRCWS
 CALLS TO: None

DESCRIPTION:

FRCWS implements an aileron command/attitude capture control law. If the first pass flag is set, the roll command integrator (FPHCMD) is initialized to ROLL, KP is set to zero (preventing roll rate stabilization in OUT10M) and the roll rate filter time constant (KTPFL) is set for 50 msec. If the pilot's control wheel is out of detent, the FFD wheel force input (FWHL) is passed through a dead zone, gained, limited and stored as FCOM.

There are two submodes within FRCWS: attitude-hold and attitude-synchronization (ATT_SYNC). The difference is that ATT_SYNC tracks the existing bank angle when the control wheel is out of detent, whereas the attitude-hold mode maintains whatever bank angle is established when the wheel is returned to detent. ATT_SYNC and the wheel-out-of-detent flag (FRWOD) are set when FWHL is greater than the deadzone (3 lbs). FRWOD is cleared when FWHL becomes less than 3, and ATT_SYNC is cleared (implying attitude hold) when roll error (FPHIER) becomes less than .05. In the ATT_SYNC submode, ROLL input is filtered, limited and output as FPHCMD. Otherwise FPHCMD retains its last value. FPHIER is calculated as FPHCMD - ROLL.

AIL1 is calculated as 2.0 FPHIER when FRWOD is true. Otherwise, roll rate damping is added, yielding

$$AIL1 = 2.0 FPHIER - 1.5 FRF1.$$

SPFINH is also set when FRWOD is true, which inhibits hardware spoiler feedback. In every case the aileron command is calculated as:

$$AILCMP = -(KV * AIL1 + FCOM).$$

The result is negated to match aircraft wiring polarity requirements. Note: since $KP = 0$, AILCMD will be identical to AILCMP.

GLOBAL INPUTS: ATT_SYNC, FWHL, KV, PF1, ROLL

GLOBAL OUTPUTS: AILCMP, ATT_SYNC, KP, KTPFL, SPFINH

MODULE NAME: RCOM (Roll Computer - CWS)
 FILE NAME: LATRL.FOR
 PROCESS: FCFAST

PURPOSE: To calculate the bank angle command (PHICMD) and command error term (PHIERR) for the Aft Flight Deck CWS modes.

CALLED BY: LATRL
 CALLING SEQUENCE: CALL RCOM
 CALLS TO: ANGL, ATAN, TAN

DESCRIPTION:

This module provides bank angle computations for both Attitude (ACWS) and Velocity (VCWS) Control Wheel Steering modes. Both flight modes are rate command / state hold systems, with the difference being in the state held. Both modes have three submodes: Attitude Sync (ATT_SYNC), Attitude Hold (ATT_HOLD) and a special mode entered when wings are commanded level. ATT_SYNC is identical for both modes. The pilot is commanding a roll rate proportional to a side stick controller input (RCWOD is true and WHLINP is non-zero) and both the commanded (PHICMD) and actual (ROLL) bank angle are progressing in the commanded direction, with PHICMD leading by an amount proportional to roll rate. So long as RCWOD is true, a counter (DELCTR) is loaded with a value proportional to PHIERR (PHICMD - ROLL). When the side stick is centered (RCWOD becomes false) PHICMD is frozen and DELCTR is decremented, giving time for ROLL to catch up with PHICMD. Once DELCTR becomes zero, ATT_SYNC is cleared and (assuming the commanded bank angle differs materially from zero but is less than 30 degrees) ATT_HOLD is set. For ACWS, this effectively completes processing. The commanded bank angle is maintained until a further sidestick input. PHICMD and PHIERR are the outputs to the roll inner loop processing. For VCWS, a computation is made of the effective radius of the turn that can be flown at the commanded bank angle and ground speed prevailing at the time DELCTR is counted down to 1. On subsequent iterations, the nominal cross track acceleration (XTKNOM) and bank angle (RCLIM) required to maintain this turn radius at the prevailing ground speed is computed and used to adjust the bank angle as necessary. XTK1, the integral of (XTKNOM - XTACC), is computed and summed with PHIERR to improve the tracking accuracy in the face of gusts or roll mistrim.

If the initial commanded bank angle is greater than 30 degrees, the bank angle is returned to 30 degrees as WHLINP becomes zero. DELCTR is not allowed to decrease below 2 with PHICMD greater than 30, so ATT_HOLD never becomes true. This is a direct bank angle command mode, with the bank angle commanded being 30 plus $f(\text{WHLINP})$, to a maximum of 45 degrees.

While RCWOD is zero and DELCTR is non-zero, PHICMD and average commanded roll rate (PDTF) are checked to see if ultimate roll attitude approximates zero. If either PHICMD or (PHICMD + PDTF) is less than TKHDZN (1.5 degrees), PHICWO is set. This causes PHICMD to be washed out to zero. For ACWS, this completes processing. The mode is effectively identical to ATT_HOLD except that the

attitude held is identically zero degrees bank angle. For VCWS, TRKHLD is set and PHICWO is cleared once PHICMD decays to less than 0.25 degrees. When TRKHLD is set, logic within procedure LATCMD freezes HLDTRK at the prevailing track angle and subsequently computes a bank angle command (BACMD) based on the difference between TK and HLDTRK. This mode is essentially identical to AUTO/TKSEL, except that the selected track is adjusted via the lateral trim switch on the sidestick rather than via the Control Mode Panel. When the trim switch is depressed and held (VATRR or VATRL true), HLDTRK is slewed at a rate that begins at 0.25 dps and gradually increases to a maximum of 1.5 dps after 5 seconds. In Track Hold mode, the difference between TK and HLDTRK is contained in TRKBG, which is sent to displays. TRKBG non-zero in VCWS mode causes the "track bananas" to be displayed, both indicating that Track Hold mode is active and displaying the selected track.

For both ACWS and VCWS, processing begins by checking the first pass flag and initializing the filters and mode discretetes if true. Next, if TRKHLD is true (VCWS only), the left (VATRL) and right (VATRR) trim switches are checked. If neither is true, CLKV is set to zero. Otherwise, if CLKV was previously zero, it is set to plus or minus 0.5 for right or left trim, respectively, and the variable gain (KG1) is initialized to 0.5. On subsequent iterations without an intervening release of the trim switch, KG1 is incremented at a rate of 0.5/sec to a maximum of 3.0. In every case, $KG1 * CLKV$ is then integrated into HLDTRK, DELCTR (used in this module to control TKSIN) is set to 5. KG1 and TRKIN (used in procedure LATCMD to control track error gain and by MSPRO to force displays to show the track line on the Nav display) is set to 20.

The next section of code determines which submode to engage. Engage criteria (effective) for the CWS submodes are as follows:

ATT_SYNC: Set by $RCWOD * \overline{GRD}$
Cleared by $DELCTR = 0$

ATT_HOLD: Set by $(DELCTR = 0) * \overline{PHICWO + TRKHLD + GRD}$
Cleared by $RCWOD$

RADHLD: = $VCWSE * ATT_HOLD$

PHICWO: Set by $GRD + ((DELCTR > 0) * (|PHICMD| --TKHDZN->) * \overline{TRKHLD})$
Cleared by $RCWOD + TRKHLD$

TRKHLD: Set by $VCWSE * PHICWO * (|PHICMD| < 0.25)$
Cleared by $RCWOD$
[Note: It should also be cleared by $(GRD + \overline{VCWSE})$]

TKSYNC Set by $TRKHLD * (DELCTR = 0) * (|TRKBG| < 1.5)$
Cleared by $DELCTR * PDCMD * PHICMD > 0.$

Input to the PHICMD integrator (PDCMD) is computed based on the selected submode, with the following hierarchical order:

Direct bank angle command ($|\text{PHICMD}| \geq 30$):

$\text{PDTCMD} = \text{SLIM}(*, 10.)$

Where: $*$ = $1.25 * \text{WHLINP} - \text{sign}(|\text{PHICMD}| - 29.5), \text{PHICMD}$
i.e, PHICMD is increased 1.25 deg beyond 29.5 per each unit of WHLINP.

TRKHLD or RADHLD (mutually exclusive):

$\text{PDTCMD} = \text{SLIM}((5. * \text{ROLERR}), 4.)$

Where: $\text{ROLERR} = \text{RCLIM} - \text{PHICMD}$

PHICWO:

$\text{PDTCMD} = \text{SLIM}(-2. * \text{PHICMD}), 10.)$

ATT_SYNC:

$\text{PDTCMD} = \text{KPDTC} * \text{WHLINP}$

Where; $\text{KPDTC} = 0.8$

Else:

$\text{PDTCMD} = 0.$

For TRKHLD mode, $\text{RCLIM} = \text{SLIM}(\text{BACMD}, \text{LMTR})$, where BACMD is the command input from procedure LATCMD and LMTR is set to the number of degrees of bank required to fly a standard rate (2 min) turn, up to a maximum of 29.5 degrees.

For RADHLD mode, $\text{RCLIM} = \text{SLIM}(\text{atan}(\text{XTKNOM} / g), \text{LMTR})$, where $\text{XTKNOM} = \text{GSFPS}^2 / \text{RADTRN}$, RADTRN is the commanded turn radius, g is the nominal acceleration due to gravity (32.174 fps/sec) and $\text{LMTR} = 30$.

Finally, PHICMD is computed as the integral of PDTCMD, with a limit of 45 degrees, and PHIERR is computed as the difference between PHICMD and ROLL plus XTK1. XTK1 is integrated whenever RADHLD is true and washed out whenever the conditions for RADHLD do not prevail. The equation for clearing TKSUNC is intended to delay activation of the curved trend vector until the airplane is banked in the direction trimmed, thus avoiding a false 'moving in the wrong direction' cue during turbulence. The clear of DRIN is no longer necessary, but was part of an earlier attempt to solve the same problem.

GLOBAL INPUTS: ACWSE, ACWSS, ATT_SYNC, BACMD, GRD, GS, GSFPS2, HLDTRK, ICM, KPDTC, KTAÜPF, LATSEL, PHICMD, PHIERR, RADHLD, ROLL, TKHDZN, TRKBG, TRKHLD, VATRR, VATRL, VCWSE, VCWSS, WHLINP, XTACC, XTK1

GLOBAL OUTPUTS: ATT_HOLD, ATT_SYNC, HLDTRK, KTPFL, PDCMD,
PHICMD, PHIERR, RADHLD, SPFINH, TRKHLD, TRKIN, XTK1

MODULE NAME: RBASC (Roll Basic - CWS Modes)
 FILE NAME: LATRL.FOR
 PROCESS: FCFAST

PURPOSE: To calculate the rudder command, aileron trim and aileron crossfeed term (VCWS).

CALLED BY: LATRL
 CALLING SEQUENCE: CALL RBASC
 CALLS TO: None

DESCRIPTION:

This routine is called in MANEL, ACWS and VCWS modes to process the aileron trim knob (ATRIM) and rudder-pedal-plus-trim (PEDAL) inputs. In ACWS and VCWS modes it also computes the turn coordinator, aileron cross-feed (AILXFD) for VCWS decrab and auto trim for aileron and rudder.

On first pass all filter outputs are zeroed and the CLOSED switch is cleared (to prevent the possibility of creating an erroneous turn coordinator output if control is received with the airplane in a bank).

Subsequently, a check is made to see if trim computation is permissible. This requires a mode of ACWS or VCWS, no pilot pedal input (note: since the inputs are pedal-plus-trim and trim, these two must be differenced to make the determination) and a 'wings level' submode (PHICWO or TKSUNC).

If these conditions prevail, the aileron trim limit (SYNCLM) is first initialized to the greater of SYNCL or 5 degrees. This is to account for the fact that the present initialization algorithm for SYNCL can produce spuriously large values if RFD control is received with the aircraft in a bank. As trim occurs, SYNCLM will gradually be reduced to 5 degrees. SYNCL is then computed as the integral of KATRM * PHIERR with a limit of SYNCLM. This trims for minimum bank angle error when near-zero bank angle is being commanded. KATRM is selected to give a time constant of 80 seconds. If Height AGL is greater than 300 feet and the option is enabled via LATSEL, RUDTRM is computed as the integral of KRTRM * AY limited to 2 degrees. KRTRM is selected to give a time constant of 80 seconds.

Next, aileron trim input (ATRIM) is multiplied by 1.52 and stored in AILTRM. The turn coordinator input (DELRF) is calculated in a washout filter as follows:

$$\begin{aligned} \text{DELRF} &= \text{PHICMD} - \text{RF2} \\ \text{RF2} &= \text{PHICMD} - \exp(-0.3 \text{ dT}) * \text{DELRF} \\ \text{DELRF} &= .235 \text{ DELRF} \end{aligned}$$

I.e, $\text{DELRF} = (.785 \text{ S} / 3.33 \text{ S} + 1) * \text{PHICMD}$.

The PEDAL input is multiplied by 6.55 and stored in RUDCML, the local rudder command. PEDALF is then computed as a one second lag on RUDCML. If VCWS, AILXFD is calculated as -2 PEDALF with a limit of 10 degrees for HTDZ > 15 feet, decreasing to zero at HTDZ

≤ 0 . ALW is then passed through a 2 second lag for use in the turn coordinator.

If ACWS or VCWS mode and PHICMD is less than 2 degrees, then the turn coordinator is enabled. Once enabled, turn coordination remains latched until MODEX becomes less than 4. The gain on DELRF (TCGAIN) is first computed as a function of filtered flap position (FLPPF): increasing linearly from 0 to 0.4 as FLPPF varies from 0 to 25 degrees, and linearly from 0.4 to 1.0 as FLPPF increases from 25 to 40. This gain is then increased by the quantity $0.1 * (1. - \sin^2(\text{ROLL}) * \text{ALFAF}$. Finally, RUDCML is recomputed as:

$$\text{RUDCML} = \text{RUDCML} + \text{TCGAIN} * \text{DELRF} + \text{RUDTRM}.$$

Processing concludes by moving RUDCML into RUDCMD using the easy-on RASWIT.

GLOBAL INPUTS: ALW, ATRIM, BMACC, FLPPF, HRAD, HTDZ, KATRM,
KDCRB, KRTRM, KTDCRB, LATSEL, MODEX, PEDAL, PHICMD,
PHIERR, RTRIM, RUDCMD, RUDTRM, SROLL, SYNCL, VCWSE

GLOBAL OUTPUTS: RUDCMD, RUDTRM, SYNCL

MODULE NAME: RCOMA (Roll Computer - Auto Modes)
 FILE NAME: LATRL.FOR
 PROCESS: FCFAST

PURPOSE: To calculate the aileron command (AILCMP) for the Aft Flight Deck Auto modes.

CALLED BY: LATRL
 CALLING SEQUENCE: CALL RCOMA
 CALLS TO: ATAN, DCRAB

DESCRIPTION:

RCOMA is called in all AUTO modes to process the horizontal path command (BACMD). Processing begins by forcing KTPFL to the standard value (50 msec) and clearing TRKHLD (just in case). It then computes the maximum value for BACMD (LMTR) as: 5 degrees if in FLARE mode, 10 degrees if ONCRS is true, the value appropriate to a standard rate (2 minute) turn if TKSEL (to a maximum of 25 degrees), or 25 degrees otherwise. PHICMD is then subtracted from the limited BACMD to produce ROLERR.

Next, LMTR is recomputed as a rate command limit: 7 dps if ONCRS, and 4 dps otherwise. The roll rate command (PDTCMD) is computed as 5. ROLERR, limited as indicated. This permits quicker response once ONCRS is set, but reduces the maximum allowable excursion. (Note: once set, ONCRS remains true through DECRAB, FLARE and RLOUT). PHICMD is then computed as the integral of PDTCMD. (The limit on this integrator is meaningless, since the input was already limited to a lesser value). PHICMD is thus a 0.2 second rate limited lag on BACMD.

If RLOUT is true, PHIERR is set to the negative of ROLL, effectively commanding wings level. Otherwise, PHIERR is computed as indicated:

$$\text{PHIERR} = \text{PHICMD} - \text{ROLL} + 0.5 * \text{XTK1}$$

XTK1 is washed out by the next statement, so its inclusion here is simply an easy-on when switching from VCWS to AUTO.

The TKSUNC discrete is set in track select mode when the track is essentially acquired and cleared when a track change is commanded, exactly as is done in VCWS TRKHLD mode. Again, the clearing of DRIN is superfluous.

If in AUTOLAND/DECRB mode, subroutine DCRAB is called next to compute the rudder and aileron contributions for the decrab maneuver. The intermediate aileron command (AILCD1) is initially computed as 2. PHIERR. If RCWOD is false (as it usually will be), the roll damping term (-1.4 PF1) is then added and, if DECRB, the aileron crossfeed term (AILXFD) is added. AILCD1 is then multiplied by the airspeed gain (KV) and (once ONCRS becomes true) is gained up by 1.25.

The final aileron command is first loaded in TEMP as

$$-(\text{AILCD1} + \text{WHLINP} + \text{AILTRM} - \text{SYNCL}).$$

TEMP is then switched into AILCMP by RASWIT, which mechanizes a 1 second easy-on when switching into AUTO mode. Note that RASWIT progresses from 0 to 1 when switching from CWS to AUTO, and from 1 to 0 when switching into CWS. Since KP is set to zero, AILCMD will be equal to AILCMP.

GLOBAL INPUTS: AILCMP, BACMD, DECRB, FLARE, GS, KV, ONCRS, PF1
PHICMD, ROLL, SYNCL, TKREL, TKSEL, TRKIN, WHLINP, XTK1

GLOBAL OUTPUTS: AILCMP, KP, KTPFL, PDTCMD, PHICMD, PHIERR,
TRKHLD, XTK1

MODULE NAME: RBASCA (Roll Basics - Auto Modes)
 FILE NAME: LATRL.FOR
 PROCESS: FCFAST

PURPOSE: To calculate the rudder command, aileron trim and aileron crossfeed term (Auto).

CALLED BY: LATRL
 CALLING SEQUENCE: CALL RBASCA
 CALLS TO: None

DESCRIPTION:

The roll basic (RBASCA) procedure is called in AUTO mode to process the pilot's ATRIM and PEDAL inputs and compute rudder commands for turn coordination. This procedure is a recent modification of an old procedure and contains several errors, the first of which is the absence of initialization code. Since AUTO mode is typically entered from VCWS, this has caused no apparent problems.

Aileron trim (AILTRM) is computed as $1.52 * ATRIM$, and the rudder trim value (RUDTRM) that may have been computed by RBASC in VCWS mode is washed out. The input to the turn coordinator (DELRF) is then computed by passing PHICMD through a washout filter as indicated:

$$DELRF = (.785 S / 3.33 S + 1) * PHICMD.$$

RUDCML, the local rudder command, is first computed as $6.55 * PEDAL$. The turn coordinator gain (TCGAIN) is next computed. TCGAIN varies linearly from 0.0 to 0.5 as filtered flap position (FLPPF) varies from 0 to 20, remains constant at 0.5 until FLPPF reaches 30, then varies from 0.5 to 1.0 as FLPPF increases to 40. Unfortunately, a leftover line of code blocks turn coordination at $FLPPF < 20$. This needs to be removed. Turn coordination is legitimately blocked until PHICMD is less than 2 degrees to prevent a transient that might be caused by enabling it with the plane in a bank. Once enabled, RUDCML is recomputed as follows:

$$RUDCML = RUDCML + TCGAIN * DELRF.$$

Finally, if DECRB is true, the decrab maneuver rudder contribution (DECRAB - computed by subroutine DCRAB) is subtracted. RUDCML is then switched into RUDCMD using the easy-on RASWIT.

GLOBAL INPUTS: ATRIM, DECRB, FLPPF, PEDAL, PHICMD, RUDCMD,
 RUDTRM

GLOBAL OUTPUTS: RUDCMD, RUDTRM

MODULE NAME: DCRAB (De-crab (Autoland) Maneuver)
 FILE NAME: LATRL.FOR
 PROCESS: FCFAST

PURPOSE: To calculate the rudder and aileron command contribution
 for Auto / Decrab mode (Autoland).

CALLED BY: RCOMA
 CALLING SEQUENCE: CALL DCRAB
 CALLS TO: None

DESCRIPTION:

This routine generates decrab and aileron crossfeed signals to align the aircraft with the runway centerline just prior to touchdown.

The first-pass flag is checked and if set, the yaw filter FLTR_1 (the lag portion of the washout filter YF1) and the yaw integrator YINT1 are initialized to zero, and the runway heading error (DLPSI) is passed through a 5 degree deadzone and placed in PSIDZ. Before RLOUT, PSIDZ is subtracted from DLPSI and the difference is stored in PSILIM. Subsequently, PSILIM = DLPSI. PSILIM is passed through a 6 second easy-on to produce YTEMP.

Next, DLPSI is differentiated (passed through a 0.4 second washout) and stored as YF1. Prior to RLOUT the yaw damping term YAWDMP is zero and YTEMP is integrated, limited, and stored in YINT1. Subsequent to RLOUT, YINT1 should be held constant (due to a coding error it continues to be integrated) and YAWDMP is computed as indicated:

$$YAWDMP = 12.55 * YF1 - 2. * PHICMD$$

$$\text{Where: } 12.55 = 5.0 / 0.4$$

Finally, both before and after RLOUT, aileron cross-feed (AILXFD) and decrab rudder command (DECRAB) are generated as follows:

$$YAWCMD = YTEMP + .756 YINT1$$

$$DECRAB = YAWCMD + YAWDMP$$

$$AILXFD = 1.4 YAWCMD$$

GLOBAL INPUTS: DLPSI, PHICMD, RLOUT

GLOBAL OUTPUTS: None

MODULE NAME: VERCMD (Vertical Path Command)
 FILE NAME: VERCMD.FOR
 PROCESS: FCFAST

PURPOSE: To calculate the vertical acceleration command for the automatic flight modes plus VCWS altitude hold.

CALLED BY: FCFAST
 CALLING SEQUENCE: CALL VERCMD
 CALLS TO: LNITRP, PAL

DESCRIPTION:

This module begins by computing the filtered flap position (FLPPF), used here and in LATRL, as a two second lag on FLPPPOS. If on the ground (SQUAT is true), wing alpha (ALW) is set to pitch attitude plus one degree, and the basic overspeed gain (KDCALF) is set to 2.0. Otherwise, ALW is computed as the sum of alpha vane input (ALFAV) plus a flap position dependent bias plus one degree. The alpha margin (ALWMARG) is then computed by passing the difference of ALW and the flap dependent alpha limit through a lead/lag filter as given:

$$ALWMARG = \frac{FLPTAB(FLPNDX) - ALW}{(2S + 1)(2S + 1)}$$

Where: FLPTAB is the table of alpha limits
 FLPNDX is an index based on the flap handle setting, equal to 1 for flaps 0, and 9 for flaps 40.

Finally, if ALWMARG is less than zero, the Research Flight Deck (RFD) is engaged (MODEX > 2) and SQUAT is false, a bit is set in the displays flag word (LFCFLG) to cause the AOA message to appear, and the stick shaker is activated by setting the output discretized FADERS and SHAKER true, then setting the stick force output command (EXPFRC) to plus or minus 1.0 (SACVIB) on alternate iterations.

The next section (if enabled) computes a dynamic alpha and overspeed protection algorithm. This was the original algorithm coded. The alpha protection portion suffers from the fact that when the stick shaker algorithm was added, the alpha limit numbers provided for that algorithm were simply coded into the original flap table. These numbers are lower for most flap settings (this is legitimate since the stick shaker algorithm does nothing until the limit is reached, whereas the original algorithm starts to dynamically reduce alpha at 80% of the limit). The result is that this algorithm limits alpha (thus also gamma) prematurely. Two separate flap tables should have been coded.

The alpha upper limit (AULIM) is computed via a call to the linear interpolation routine (LNITRP) with FLPPF and FLPTAB as inputs. The lower limit (ALLIM) is then set to 0.6 AULIM. If bit 15 of the vertical control option word (VSTSEL) is set, then for AUTO mode the gamma limit (GAMLIM) is set to 10 degrees and otherwise to 20 degrees. If bit 15 is clear, GAMLIM is set to 8.5

degrees for negative and 1.4 AULIM for positive gamma. GAMLIM then would vary from 8.5 to 14.8 degrees depending on flap setting and gamma. The vertical speed limit (LHDC) is then computed as GAMLIM times the degree to radian conversion factor (DTOR) times true air speed in fps (TASFPS). The vertical acceleration limit (LHDD) is set to 5 fps/sec unless the maximum airspeed for the existing flap setting is approached (KHCAS < 1.), in which case it is set to 10 fps/sec. The vertical path command limit in deg/sec (LVPC) is then computed as LHDD times the radian to degree conversion factor (RTOD) divided by LIMGS, which is the lesser of ground speed in fps (GSFPS) or 170 fps (100 knots).

The alpha control variable (KALFA) is computed as follows. If ALW is less than ALLIM, TEMP is set to 1.0. Otherwise it is set as follows:

$$\text{TEMP} = 1. - (\text{ALW} - \text{ALLIM}) / (\text{AULIM} - \text{ALLIM})$$

This is a number that varies from 0.0 to 1.0 as ALW varies from ALLIM to AULIM. Two times TEMP is then passed through a four second lag to produce KDCALF. KALFA is KDCALF limited to the range 0.0 to 1.0. If enabled, KDCALF is used in the SPDCMD module to begin increasing thrust when alpha exceeds 0.6 AULIM, and KALFA is used in this module (or in ELEVP) to begin reducing the gamma or pitch command when alpha exceeds 0.8 AULIM. If bit 15 of VSTSEL is set or if AUTOE is false or the autothrottle is not engaged (ATE is false), KALFA is forced to 1.0. The vertical guidance modes are calculated next.

FLIGHT PATH ANGLE MODE

If flight path angle mode is selected or preselected (FPASEL or PSTFPA true), FPASUM is limited to GAMLIM and the flight path angle gain (KFPA) is computed as .01 GSFPS. The intermediate TEMP is then set to KALFA times FPASUM if FPASUM is positive, and KHCAS times FPASUM otherwise. The FPA mode vertical acceleration command (DFPAHM) is then computed as $\text{KFPA} * (\text{TEMP} - \text{GAMMA})$. If FPASEL and PSTFPA are both false, TEMP is set to the VCWS gamma command (PGAMC) if in VCWS mode, and to GAMMA otherwise. TEMP is then rounded to the nearest tenth of a degree and set into FPASUM.

ALTITUDE MODE

If altitude mode is selected, preselected or armed (ALTHLD, ALTARM or PSTALT true), the altitude error (DELALT) is computed as $\text{ALTSUM} - \text{ALTCOR}$. If ALTHLD is true (indicating VCWS altitude hold), TEMP is set to $\text{HLDALT} - \text{ALTCOR}$ and the altitude mode vertical speed command (ASC) is computed as follows:

$$\text{ASC} = .075 * \text{SLIM}(\text{TEMP}, (8.333 / .075))$$

This limits the maximum vertical speed command to 500 fpm (8.333 fps). Otherwise, TEMP is set to DELALT and ASC is computed as follows:

$$\text{ASC} = .075 * \text{SLIM}(\text{TEMP}, (\text{ALHDC} / .075))$$

Where: ALHDC is the greater of 500 fpm or the vertical speed which prevailed at the time altitude mode was selected.

Note: it is possible to be in VCWS altitude hold at one altitude, with an altitude armed or preselected at (ALTSUM equal to) a different altitude. DELALT always refers to ALTSUM.

If in VCWS mode or if AUTOE and ALTSEL, GAMCMD is computed as RTOD times ASC divided by LIMGS. In AUTO/ALTSEL mode, GAMCMD will be set into NCL2 (the throttle feed forward command) and also into GAMC, which is sent to the displays computer to drive the gamma wedges on the PFD. In VCWSE (with ALTHLD true), GAMCMD will be integrated into PGAMC, the gamma command integrator in the PAFD subroutine of ELEVP, which is subsequently set into GAMC. NCL2 is set by logic in ELEVP.

If PSTALT, ALTARM and ALTSEL are all false, ALTSUM is set to ALTCOR (rounded to the nearest 10 foot increment) and DELALT is set to zero. If ALTHLD is false, HLDALT is set in the same manner. Finally, ALHDC is computed as follows: If ALTSEL is true, the Boolean ALTHOLD is set when HDCF becomes less than 8.334, remaining set (once true) until ALTSEL becomes false. Then, if ALTHOLD is true, ALHDC is set to 8.333 fps (500 fpm). Otherwise it is set to the lesser of its previous value or 0.5 fps more than the present magnitude of HDCF. If ALTSEL is false, ALTHOLD is set false and ALHDC is set to the present magnitude of HDCF.

VERTICAL PATH (3D) MODE

If 3D guidance is possible (GUID3D is true), the vertical path guidance algorithms are evaluated. An easy-on is provided on initial engagement of VERPTH, or when a switch occurs in the altitude reference source (NVVMOD changes value) or when switching from CWS to AUTO with VERPTH engaged. The Boolean MODSW is set when NVVMOD differs from its previous value. The absolute value of 3D altitude error (HER - set by HVGUID) is computed, and the vertical speed error is computed as the difference between actual complementary filtered hdot and path nominal hdot ($HDE = HDCF - HDTC$). If $|HER|$ is not identically zero and either AUTOS or MODSW is true or VERPTH is false, then the following occurs: The vertical path acquired flag (VPTHK) is set false. If VERPTH is true and either AUTOS or MODSW becomes true, the variable vertical gain (KKH) is set to 0.1 to mechanize the easy-on. Else, if $|HER|$ is greater than 500 feet, or if $|HER|$ is greater than 100 feet and the aircraft is diverging from the path (HDE times HER is greater than zero), the vertical 'be careful' flag (VBCFLG) is set true. (This prevents MSPLGC from setting VERPTH true). Else if $|HER|$ is less than 25 feet, or if HDE times HER is greater than zero (we're flying toward the path) and the commanded minus the measured components of the vertical steering command ($VSTRA - VSTRB$) times HDE is less than zero (it's time to begin reducing the rate of approach), then VBCFLG is set false and KKH is set to 0.1. The check for HER non-zero in the above is intended to avoid false

captures caused by the fact that HER is not computed by HVGUID until several iterations after GUID3D becomes true. Once VERPTH becomes true, |HER| is less than 25 feet and |HDE| is less than one foot per second, VPTHK is set true if AUTOE is true.

The variable tracking gain (KKH) is computed next. If IC mode is true or VPTHK is false, TEMP is set to 1.0. If IC mode, KKH is also set to 1.0. Otherwise, TEMP is set to KALT (the airspeed gain computed by LATCMD) divided by 1.6. TEMP is then passed through a four second lag to produce KKH. The vertical error and vertical speed error gains are then computed as follows:

$$\begin{aligned} KH &= .09 * KKH^2 \\ KHD &= 0.6 * KKH \end{aligned}$$

The (pre-limiting) commanded hdot (VSTRA) is a function of altitude error and nominal path climb/descent rate as given:

$$VSTRA = HER * KH + HDTC * KHD$$

TMP1 is then set to VSTRA limited to KHD times LHDC. (The original value of VSTRA is retained to set the capture point). TMP1 is used to compute commanded gamma as indicated:

$$\begin{aligned} TMP1 &= SLIM(VSTRA, (LHDC * KHD)) \\ TMP2 &= RTOD * TMP1 / (LIMGS * KHD) \end{aligned}$$

TMP1 is then multiplied by either KALFA or KHCAS depending on whether the command is 'fly up' or 'fly down' respectively. The vertical acceleration command (VERSTR) is computed as:

$$VERSTR = TMP1 - VSTRB$$

$$\text{Where: } VSTRB = HDCF * KHD.$$

Finally, if AUTOE and VERPTH are both true, GAMCMD is set to TMP2. Further, if bit 0 of VSTSEL is set, the path segment flight path angle times the along track acceleration (FPA * VGSDOT) is added to VERSTR. If bit 1 of VSTSEL is set, GAMCMD is also set into NCL2. Otherwise, the Programmed Flight Path Angle (PFPA) is set into NCL2. If GUID3D is false, VBCFLG is set true and VERSTR is zeroed.

GLIDE SLOPE ENGAGE / TRACK MODE

The Glideslope Engage logic is next evaluated. Initially the local glideslope enable flag (VTMP) is set false. If a MLS type capture is indicated (MLSM is true), and land mode is requested (LANDR true), the MLS glideslope profile is computed as follows:

$XDIST = XHAT - XGPIP$
 $ZPROF = \tan(GSA) * XDIST - HGPIP$
 $DELTH = ZHAT - HGPIP$

Where: XHAT is the X distance from the A/C to the Azimuth antenna;
 XGPIP is the X distance from the GPIIP to the Az antenna;
 ZHAT is the aircraft height above the MLS plane;
 HGPIP is the height of the GPIIP above the MLS plane.

The height error limit for capture (DHLIM) is then calculated as:

$DHLIM = 15$ for $XDIST < 4300$;
 $DHLIM = .00349 * XDIST$ for $XDIST \geq 4300$.

If $|DELTH|$ is less than $.54 DHLIM$, VTMP is set true if LOCE is true. (Glideslope engage is not permitted until the localizer is engaged). If MLSM is false, VTMP is set when $|GSDEV|$ is less than $.1085$ degrees and LOCE is true.

Finally, if GSARM and VTMP are both true and $|DLPSI|$ is less than 90 degrees, The vertical beam sensed (VBS) and glideslope engaged (GSENG) flags are both set true. With GSENG and AUTOE both true, a ten second countdown is initiated to set the glideslope track flag (GSTRK), and subroutine PAL is called to compute the vertical acceleration command for glideslope capture and track (GSCMD). Note that GSTRK is never set in the CWS modes since it affects only the calculation of GSCMD, which is used only in AUTO mode. With GSENG or AUTOE false, the counter is reset to 10 seconds.

The next section selects the vertical acceleration (VACMD), commanded gamma for display (GAMC) and throttle feed forward (NCL2) terms, to the extent that these haven't already been set.

ALTSEL:

$VACMD = ASC$
 $VACMD = VACMD + SLIM((ASC - VACMD), 0.25)$ (ALTIN = 0)
 (ALTIN > 0)

Where: ASC is the altitude mode acceleration command;
 ALTIN is a counter set by MSPLGC when the altitude or flight path angle knob is turned.

FPASEL:

$VACMD = DFPAHM$ (ALTIN = 0)
 $VACMD = VACMD + SLIM((DFPAHM - VACMD), .25)$ (ALTIN > 0)
 $SELFPA = FPASUM$
 $NCL2 = FPASUM$ (AUTO mode)
 $GAMCMD = FPASUM$ (AUTO mode)

Where: DFPAHM is the flight path angle mode accel command;
 SELFPA drives the gamma reference bar on the PFD;
 NCL2 is the throttle feed forward command;

GAMCMD drives the gamma wedges on the PFD (AUTO mode).

GSENG:

VACMD = GSCMD
 SELFPA = -GSA
 NCL2 = -3.0
 GAMCMD = GAMMA (FLARE mode)
 GAMCMD = RTOD*(GSCMD + 1.56*HDCF)/(1.56*LIMGS) (not FLARE)

Where: GSCMD is the land mode vertical acceleration command;
 GSA is the nominal glide slope for the selected runway.

VERPTH:

VACMD = VERSTR
 SELFPA = PFPA

Where: VERSTR is the 3D mode acceleration command;
 PFPA is the programmed flight path angle for the path.

In final processing, VACMD is limited to 5 fps/second (if not GSENG), GAMCMD is conditionally passed through a rate limited filter (if AUTOE and not FLARE and bit 2 of VSTSEL is set) before setting into GAMC, and GAMCMD is converted to a rate limited gamma dot command (for use in VCWS altitude hold).

GLOBAL INPUTS: ALFAV, ALTARM, ALTCOR, ALTHLD, ALTIM, ALTSEL, ALTSUM, ATE, AUTOS, AUTOE, DLPSI, FLARE, FLPNDX, FLPPPOS, FPA, FPASEL, FPASUM, GAMC, GAMMA, GSARM, GSDEV, GSENG, GSFPS, GSTRK, GUID3D, HDCF, HDTC, HER, HGPIP, ICM, KALT, KHCAS, LANDR, LFCFLG, LOCE, MLSM, MODEX, NVVMOD, PFPA, PGAMC, PITCH, PSTALT, PSTFPA, SACVIB, SQUAT, TANGSA, TOG100, VACMD, VCWSE, VERPTH, VSTSEL, XGPIP, XHAT, ZHAT

GLOBAL OUTPUTS: ALB, ALTSUM, ALW, ALWMARG, DELALT, DELTH, EXPFRC, FPASUM, FADERS, FLPPF, FPASUM, GAMC, GAMCMD, GAMLIM, GSENG, GSTRK, KHCAS, KALFA, LFCFLG, NCL2, SELFPA, SHAKER, VACMD, VBCFLG, VBS, VERSTR, VSTRA, VSTRB

MODULE NAME: PAL (Pitch Auto Land)
 FILE NAME: VERCMD.FOR
 PROCESS: FCFAST

PURPOSE: Compute the vertical acceleration command for the
 Glideslope Engaged mode.

CALLED BY: VERCMD
 CALLING SEQUENCE: CALL PAL(GSCMD)
 CALLS TO: None

DESCRIPTION:

This procedure provides the vertical steering command (GSCMD) for both ILS and MLS autoland. No filtering is done if MLSM is true, since the input (DELTH) is derived from filtered quantities. In ILS mode, a lower time constant filter is used during glideslope acquisition than is used once glideslope track (GSTRK) is set by VERCMD. Filter initialization is performed when the glideslope arm discrete (GSARM) is true. Since this routine is called only when glideslope engage (GSENG) is true, and GSARM is cleared by procedure MLOG when GSENG is true, GSARM can be true only on the first pass after selecting land mode. Initialization is also performed when AUTOS is true to smooth the transition when the pilot has hand flown the aircraft into GSENG in VCWS mode, then selected AUTO mode.

The following equations are evaluated:

If MLSM is true

$$\begin{aligned} \text{GSGPA} &= 0.0666 * \text{DELTH} \\ \text{GSPF} &= 1.06 * \text{GSGPA} \end{aligned}$$

Where: DELTH is the MLS beam error in feet;
 GSPF is the vertical speed command in K*fps.

Else

$$\text{GSGPA} = 0.0666 * \text{GPGSDV}$$

If GSARM is true

$$\text{GSPF} = 1.06 * \text{GSGPA}$$

Else if GSTRK is true

$$\text{GSPF} = \frac{1.06}{15 S + 1} (\text{GSGPA} + \tan(\text{GSA}) * \text{GS}_{\text{tps}} + \text{HDCF})$$

Else

$$\text{GSPF} = \frac{1.06}{1.5 S + 1} \text{GSGPA}$$

Where: GPGSDV is the gain programmed localizer deviation in feet;
 GSA is the glide slope angle in degrees;
 HDCF is complementary filtered h dot in fps.

End if

Unconditionally:

$$\text{ALC} = 4. (\text{GSPF} + 0.39 \text{HDCF})$$

Where: ALC is the proportional path command output in K*fps.

If GSARM is true

$$\text{GSI} = \text{DECMD} - \text{ALC}$$

Else

$$\text{GSI} = \frac{.28}{S} \text{GSGPA}$$

End if

Where: DECMD is the previous iteration's elevator command output;
 GSI is the proportional path command output in k*fps;
 GSI is limited to a max value of 80.

$$\text{Finally, GSCMD} = -(\text{ALC} + \text{GSI})$$

Where: GSCMD is the vertical acceleration command in fps/sec.

GLOBAL INPUTS: DECMD, DELTH, GPGSDV, GSARM, GSINS, GSTRK, HDCF,
 MLSM, TANGSA

GLOBAL OUTPUTS: None

MODULE NAME: ELEV (Elevator Control Processing)
 FILE NAME: ELEV.FOR
 PROCESS: FCFAST

PURPOSE: To calculate the elevator command and the stabilizer trim discretetes.

CALLED BY: FCFAST
 CALLING SEQUENCE: CALL ELEV
 CALLS TO: PAFD, PFFD, PVPC, STABT, VTFCL

DESCRIPTION:

ELEV is the main program for the longitudinal axis control laws. It computes certain global variables (HDDF, HDILS, KQ, PDCOL, PHIVS, QFB1), calls mode specific processing to compute the local elevator command (DECGL) and the trim discretetes (TRIMT, TRIMD), then outputs DECMQ (Delta Elevator Command Minus Q. Pitch rate (Q) complementation is accomplished externally (at the 10 msec rate) through an easy-on, based on DECGL. Calculations for manual electric mode (MANEL) and the rollout (RLOUT) submode of Land Engage (LANDE) are accomplished entirely within ELEV.

Only two modes are available from the forward flight deck for elevator commands: pre-engage (PRENG) and forward flight deck attitude control wheel steering (FFDE). During the PRENG mode, the autopilot is disengaged. The computational requirements for the elevator command for the FFD mode are described in sub-procedure PFFD.

There are five distinct modes that can be flown from the research flight deck (RFD): MANUAL Electric (MANEL), Attitude Control Wheel Steering (ACWS), Velocity Control Wheel Steering (VCWS), AUTO, and LAND, with LAND having separate computational requirements for glideslope tracking (GSENG), flare (FLARE) and roll-out (RLOUT). The overall DSD showing the computational requirements for the elevator command for these RFD modes is given in Appendix A of this document. The overall static gains are included in these figures.

Procedure ELEV is the driver for all the sub-procedures which comprise the longitudinal axis control law. It consists of initialization routines, signal preprocessing and procedure calls.

ELEV begins with a check for a change in flight mode since the last iteration. If there was such a change, the easy-on/off switch is initialized (INIT = TRUE) to gradually phase in the elevator command for the new mode. Also, the high rate Q feedback gain (KQ) is set to zero. This gain is reset to the appropriate value for those modes which use it by mode dependent logic. If the current mode is PRENG or if the IC Mode (ICM) flag is set, the filters and command outputs are initialized. Then, for all modes, pitch rate (Q) is filtered and stored in QFB1 (a 16 second washout) and QX (the 16 second lagged output used by OUT10M). Versine of bank angle is approximated and stored in PHIVS. Filtered vertical acceleration (HDDF) is then computed, the ILS vertical velocity (HDILS) is initialized or computed as required, and the Flare first pass flag, FLARE_1, is updated. Next, the column input (DCOL) is

passed through a variable deadzone to produce PDCOL, which is used locally and also sent to the displays computer, where it is used to actuate the out-of-detent indicators on the Primary Flight Display. If PDCOL is non-zero, the out-of-detent flag (PCWOD) is set. Subsequently, ELEVP processes the active mode as indicated by the value of MODEX. The output of the mode specific processing is DECGL, which is passed through the 1/2 second easy-on/off switch initialized at ICM or mode change. Except for LANDE, Decrab (DECRB), and FLARE, the final elevator command is phased in as DECMQ (Delta Elevator Command Minus Q. Note that Q complementation is performed in OUT10M for all modes except FLARE and MANEL.) For LAND submodes, the switch in control law source (and setting of INIT) occurs at Glide Slope Engage (GSENG). The next switch occurs at FLARE, but the control law itself takes care of the easy-on.

For PRENG mode, DECGL is set equal to the elevator servo position (DEPOS). This synchronization is to minimize the transient at autopilot engage.

For MANEL mode, the deadzoned column position input (PDCOL) is multiplied by -6.96, and the result stored in DECGL.

For FFDE mode, the sub-procedure PFFD is called to compute DECGL.

For ACWSE and VCWSE modes, sub-procedure PAFD is called.

For AUTOE mode, sub-procedure PVPC is called.

For LANDE mode, PVPC is called and the easy-on/off switch is disabled. This permits the intermediate elevator command DECGL to pass directly through to the output DECMQ.

For DECRB and FLARE modes, the variable-tau flare control law, sub-procedure VTFCL, is called. If the FLARE flag is false, sub-procedure PVPC is also called. In this mode also, the easy-on/off switch is disabled.

For RLOUT mode, the elevator is driven to zero by a one second lag filter for nose wheel letdown. For all high speed modes (FFD, ACWS, VCWS, AUTO), the summation of pitch rate feedback is done externally at the 10 Msec rate, using the ELEVP outputs KQ, QX, DECMQ, and the 10 Msec pitch rate (Q) input.

To conclude ELEVP processing, the stabilizer trim sub-procedure, STABT is called unconditionally.

198

GLOBAL INPUTS: ACCB, DCOL, DECRB, DEPOS, DZNE, FLARE, HDCF, HDD,
HRAD, ICM, MODE2, MODEX, MSW1, MSW6, PRENG, Q, ROLL, ZDH,
ZDDH

GLOBAL OUTPUTS: DECMQ, HDILS, KQ, MODEX, PDCOL, QFB1, QX, TRIMD,
TRIMT

MODULE NAME: PFFD (Pitch Forward Flight Deck)
FILE NAME: ELEVP.FOR
PROCESS: FCFAST

PURPOSE: To calculate the elevator command for Forward Flight Deck
ACWS mode.

CALLED BY: ELEVP
CALLING SEQUENCE: CALL PFFD
CALLS TO: None

DESCRIPTION:

This routine computes the delta elevator command in the forward flight deck CWS mode. The column force (FCOL) is processed through a +/- 5 lb. deadzone to yield FCOM. If FCOM is non-zero, the Pitch Control Wheel Out of Detent flag (PCWOD) is set. If IC mode or first pass in FFD mode, all filters are initialized and the pitch command integrator (PINT1) is set equal to pitch attitude. Otherwise, FCOM is gained and filtered to give FFD1. If the elevator authority limit flag (DEAL) is false, FCOM is multiplied by .1935 and integrated to give PINT1, with a limit of +/- 40 degrees pitch command. The error signal (PFFDE) is calculated by subtracting PINT1 from PITCH. This error is integrated only if PCWOD is false and the bank angle command (PHICMD) is less than 5 degrees. The basic elevator command, FCWSE is computed as the gained sum of PINT2, PFFDE and FFD1. This signal is compensated by PHIVS to produce FDED, which is then used to set the elevator authority limit flag (DEAL) if the effective elevator command is greater than 15 degrees. Finally, FDED is attenuated as a function of KV and output as DECGL. KQ is set to 2.16 times KV for use by OUT10M.

GLOBAL INPUTS: FCOL, KV, PHICMD, PITCH

GLOBAL OUTPUTS: KQ

MODULE NAME: PAFD (Pitch Aft Flight Deck)
FILE NAME: ELEVP.FOR
PROCESS: FCFAST

PURPOSE: To calculate the elevator command for the Aft (Research) Flight Deck ACWS and VCWS modes.

CALLED BY: ELEVP
CALLING SEQUENCE: CALL PAFD
CALLS TO: None

DESCRIPTION:

This procedure computes the delta elevator command (DECGL) and pitch/gamma command (GAMC - used to display the commanded angle on the Primary Flight Display) for both the velocity and attitude CWS modes. It provides the DECGL command by direct pilot control over the commanded flight path angle (GAMMA) for VCWSE, or the pitch attitude (PITCH) for ACWSE. This may be accomplished by column input (DCOL), which provides a rate command proportional to column deflection, modified by ground speed to provide a nearly constant vertical acceleration per inch of column deflection (0.2 g / in). Alternatively, pilot input may be made through the vertical trim switch (VATRM, VATRD) which results in a constant 0.5 deg/sec rate command. When the pilot input is removed, the command integrator (PGAMC) remains constant. Additional command options are as follows. If the Go-Around Switch (GAS) is depressed, VCWS mode is forced (assuming the research (aft) flight deck was engaged), and PGAMC is driven to 2.0 degrees fly up. Also, when in VCWS mode, setting commanded gamma to approximately zero causes altitude hold mode to be entered (ALTHLD true). In this mode, the actual altitude at the time ALTHLD is engaged is maintained. Logic in procedure VERCMD monitors the altitude deviation and provides a gamma rate command (GAMCMD) which is integrated into PGAMC. In this mode, the displayed gamma (GAMC) is driven to zero and a bit is set in FCFLGS which causes the center diamond in the gamma wedges on the PFD to be raster filled as a mode engaged indicator.

For the following discussion, it would be helpful to refer to either the source code or the Digital System Diagram (DSD). Processing begins by selecting the reference angle (PAFB) according to flight mode. If in ACWS, the reference angle is PITCH and the command angle limit computed by VERCMD (GAMLIM) is increased to produce an appropriate pitch angle limit. PDCOL, the deadzoned DCOL input is then programmed with ground speed such that the commanded normal acceleration per inch of column input remains constant over the speed range. The mode first pass flag (INIT) or any control input causes ALTHLD to be cleared. Also, INIT or SQUAT causes PGAMC to be synchronized to the selected angle reference (PAFB). If first pass in the mode, further initialization processing is performed. Otherwise, processing continues by evaluating the submodes.

If GAE (Go-Around Engaged) is set, then a 2 degree bias (GAC) is integrated into GAMC until the 2 degree fly-up is achieved. At this point GAE is reset. GAE is also reset if a column input is

made. If GAE is false and PCWOD is true, GAMDC, used in the ALTHLD checks, is set to the previous value of filtered column input, DCLFI. If PCWOD and GAE are both false, the Vertical trim switches are checked. If VATRM (trim up) or VATRD (trim down) is true, +/- 0.5 deg/sec is integrated into PGAMC and GAMDC is set appropriately. If none of the above, the possibility of entering altitude hold mode is checked: if VCWS mode and altitude above ground level is greater than 300 feet and PGAMC is less than one degree, the previous values of PGAMC and GAMDC are summed. If the result is less than .25 deg or passes through zero, the ALTHLD discrete is set. This causes VERCMD to freeze its altitude reference signal and begin computing a gamma dot command (GAMCMD) based on altitude error and altitude rate.

Finally, if ALTHLD, DCLFI is set equal to GAMCMD and otherwise computed as a 0.3 sec lag on gained PDCOL. PGAMC is computed as the integral of DCLFI, limited to +/- GAMLIM. PDCOL is then degained if negative and an overspeed condition exists (KHCAS less than 1.0) and passed through a .093 second lag to produce DCOLF. DCOLF is used in a parallel proportional path to improve response to column inputs at low speeds.

In the integral path, GAMC is subtracted from either GAMMA or PITCH to produce the error signal, GAMER. Since DCOLF is degained by KV and GAMER is not, GAMER is the primary controlling term at high speeds (CAS > 235 kts). (Note: in the present implementation, GAMER is used for reference only. The analogous term used in the control law is 'TEMP', computed later.) Next, NCL2, the throttle feed forward term, is set equal to PGAMC unless GAE is true, in which case it is set to GAC (2.0 deg). PAFTMP in the proportional path is computed as KV times the negative of 9.4 DCOLF plus PHIVS, and KQ is set to 4.32 times KV.

In VCWS the damping term GAMD is set to computed gamma dot (with a 100 knot floor imposed on GSINS) unless SQUAT is true, in which case it is set to zero to keep the elevator quiet during taxiing. GAMD is then summed with QFB1 and 1.85 times DCOLF and passed through a one second lag to produce GAMDF. Note that QFB1 is also used directly for damping, but this is added to the elevator command in OUT10M. In ACWS, GAMD is set to QFB1 and GDFP is set to zero.

TEMP is now computed. This term will be equal to GAMER unless commanded gamma is negative and an overspeed condition exists, in which case the effective commanded gamma is driven towards zero to pull the nose up.

The gained sum of TEMP and GAMD is then integrated to produce ERINT, which is used to improve tracking of small angular errors. ERINT is IC'd on first pass and held constant when the difference between the elevator command and the elevator position (DEPOS) is greater than one degree. Finally, the intermediate terms ERINT, TEMP, GDFP, and PAFTMP are gained and summed to produce DECGL, which is returned to ELEVP.

202

GLOBAL INPUTS: ACWSE, ALTHLD, DECMD, GAE, GAMCMD, GAMLIM, GSINS,
HDCF, HTDZ, KHCAS, KV, NAV64K, PDCOL, PITCH, QFB1,
VATRD, VATRM, VCWSE

GLOBAL OUTPUTS: ALTHLD, DCOLF, ERINT, GAE, GAMC, GAMD, GAMER,
GAMLIM, KQ, LFCFLG, NCL2, PDCOL, PGAMC

MODULE NAME: PVPC (Pitch Vertical Path Command)
FILE NAME: ELEVP.FOR
PROCESS: FCFAST

PURPOSE: To calculate the elevator command for the Aft (Research)
Flight Deck AUTO modes.

CALLED BY: ELEVP
CALLING SEQUENCE: CALL PVPC
CALLS TO: None

DESCRIPTION:

This routine computes the elevator command DECGL for all AUTO submodes except FLARE and RLOUT.

A vertical path command (EVPC) is derived as the difference between the vertical acceleration command (VACMD) from procedure VERCMD and the filtered vertical acceleration signal (HDDF). If first pass in AUTO mode, INT2 is initialized so as to produce no net change in DECGL this pass, and the first pass flag (PVPCI) and the VCWS altitude hold flag (ALTHLD) are cleared. Otherwise, EVPC is integrated whenever TRIMT is false (indicating stabilizer trim is not active) or when the sign of EVPC differs from the sign of the integrator output (INT2). INT2 is limited to +/- 20 degrees, gained by .25, and summed with PHIVS and the original estimated path command (EVPC) to produce EVPCS. This is gained by KV and output as DECGL. KQ is then output as 2.16 times KV for use in OUT10M.

GLOBAL INPUTS: GSENG, KV, TRIMT, VACMD

GLOBAL OUTPUTS: ALTHLD, KQ

MODULE NAME: VTFCL (Variable Tau Flare Control Law)
 FILE NAME: ELEVP.FOR
 PROCESS: FCFAST

PURPOSE: To calculate the elevator command in FLARE mode.

CALLED BY: ELEVP
 CALLING SEQUENCE: CALL VTFCL
 CALLS TO: None

DESCRIPTION:

VTFCL is called when DECRB is true. This routine initializes itself and sets the FLARE flag at descent through 42 feet of altitude (HTDZ for MLSM, HRAD for ILS). The variable-tau law then functions to drive the commanded rate of descent exponentially to approximately 2 fps at touchdown.

First, values of altitude and vertical acceleration are selected depending on MLS Switch 1 (normally false). HT is set to the selected altitude plus 10 feet (which sets the nominal touchdown sink rate: $10. * .6616 / 3.25 = 2.04$), and the damping term HD2 is set to the selected vertical acceleration plus 7.6 times filtered pitch rate (QFB1). HT is then monitored to set the FLARE discrete.

Once FLARE is true, the subroutine proceeds by computing the ground speed compensation term ($XD = 1.0 @ 120 \text{ kts}$), and the flare error term (DEFL1). Note that if the aircraft is on a nominal 3 degree glideslope, DEFL1 will equal zero at flare initiation.

On the first pass of FLARE, the filters and integrators are initialized such that the first calculation of DECGL will result in zero transient error. Errors in the sink rate are then corrected by use of a complementary filter using HD2 and filtered pitch rate (QFB1). The result is stored in HDDQ. The output of the FTAUF3 filter, which is initially set to zero and ramps to -4 degrees to set the elevator command bias, is added to DEFL1 and HDDQ. This, multiplied by KV, yields the output DECGL. NCL2, the nominal gamma command to ATHCL, is set to zero, and KQ remains zero, having been cleared in ELEVP mainline code.

GLOBAL INPUTS: FLARE, GSINS, HDD, HDILS, HRAD, HTDZ, KV, MSW1,
 QFB1, ZDDH

GLOBAL OUTPUTS: FLARE, NCL2

MODULE NAME: STABT (Stabilizer Trim Logic)
FILE NAME: ELEVP.FOR
PROCESS: FCFAST

PURPOSE: To calculate the stab trim discretetes

CALLED BY: ELEVP
CALLING SEQUENCE: CALL STABT
CALLS TO: None

DESCRIPTION:

This routine determines the stabilizer trim command and the on-ground discrete, GRD. GRD is set when SQUAT becomes true, and remains set as long as the radar altitude remains less than 10 feet.

The trim discrete, TRIMT, is cleared if GRD, MANEL, or PRENG are set. If the magnitude of the auto stabilizer trim pot input (ASTP) is less than one degree, then TRIMT is also cleared. If its magnitude is greater than one degree and less than 1.33 degrees, TRIMT is not changed. If ABS(ASTP) remains greater than 1.33 for 24 cycles, then TRIMT is set true. If ASTP is negative and TRIMT is set, the trim down discrete, TRIMD, is set, otherwise, TRIMD is cleared. The stabilizer is trimmed nose up if TRIMT is true and TRIMD is false, and trimmed nose down if TRIMT is true and TRIMD is also true.

GLOBAL INPUTS: ASTP, HRAD, MANEL, PRENG, SQUAT

GLOBAL OUTPUTS: GRD, TRIMD, TRIMT

MODULE NAME: SPDCMD (Speed Command)
FILE NAME: SPDCMD.FOR
PROCESS: FCFAST

PURPOSE: To calculate the along track acceleration command for the autothrottle modes.

CALLED BY: FCFAST
CALLING SEQUENCE: CALL SPDCMD
CALLS TO: LNITRP

DESCRIPTION:

This module computes terms associated with overspeed protection and computes the autothrottle acceleration command (ATCMD) for air speed (CAS) and ground speed (GS) hold modes as well as the time path (4D) mode. In the event that maximum allowable mach is reached, it in effect operates in mach hold mode by computing the equivalent CAS which will prevent the maximum mach number from being exceeded.

Initial processing computes the CAS to TAS ratio (KCTOT), the CAS to MACH ratio (CMRAT) and the filtered wind speed (WSF). CMRAT is computed using an alpha/beta filter to smooth the term without introducing a lag. If PNAV64 is false (groundspeed below 64 knots), all filtered terms are held in the IC state and KCTOT is set to 1.0.

The next section of code packs the 'nose gear down' discrete (GEAR) in the FCFLGS word for transmission to displays, and sets lower maximum values for MACH and CAS when the gear is down. MXMACH is set to .82 vice .84, and CASTAB(1) (flap placard limit for flaps 0) is set to 310 vice 340 knots. The maximum permissible CAS (CASHI) for the existing flap position is then computed via a call to the linear interpolation routine, LNITRP.

The operator selectable CAS and mach limits (MAXSPD) are processed next. These were originally intended for a controlled profile descent and were settable from the CDU, but this capability was not included in the new CDU code. Setting the MAXSPD elements via the VIEW utility is still useful for test purposes. Neither element is used unless MAXSPD(2) is non-zero. If so, element 2 is limited to a min of 0.4 and a max of MXMACH, then set into MXMACH. MAXSPD(1) is then limited to a min of $1.2 * IASREF$ and a max of CASHI and the result is set into CASHI. If the CAS equivalent of MXMACH is then less than CASHI, CASHI is set to $CMRAT * MXMACH$.

Provision is made to use either raw CAS or filtered CAS (FMCASF) for input to the speed control equations (SELCAS). In practice, only CAS has been used. The intent of the filter was to reduce throttle activity by rejecting turbulence without introducing significant lag. The purpose of the wind speed/direction term is to reduce the standoff that can occur when turning into or away from the wind. This term is zeroed when DPERRF (set by HNAVSL when navigation errors are sufficiently large to cause bogus wind estimates) is true. Although the filter works reasonably well in normal turbulence, further testing indicates it does not respond well to wind shear (though it errs in the

direction of safety), so it should probably be removed or replaced.

Overspeed control is effected through the use of two terms: KHCAS, used in either VERCMD (auto modes) or ELEVP (CWS) to control gamma, and KDCKH, used to control the throttle. Once SELCAS comes within 15 knots of CASHI, the difference between CASHI and SELCAS is converted to the ft/sec equivalent (times 0.1) as given:

$$TMP1 = KDCAS * (CASHI - SELCAS) / KCTOT$$

$$\text{Where: } KDCAS = 0.16878 = KTOFPS / 10.$$

This term is then concatenated with along path acceleration, gained and limited:

$$TMP2 = SLIM(0.125 * (TMP1 - ABACC), 0.2)$$

Where: ABACC is the body mounted accelerometer 'x' acceleration, rotated and biased in the CAS filter.

TMP2 becomes negative when the present speed and acceleration would cause the limit speed to be reached within 10 seconds, and is otherwise positive. If not within 15 knots of CASHI, TMP2 is set to 0.1. If the option is enabled (bit 15 of ATHSEL is zero), TMP2 is integrated into KDCKH which is initialized to 2.0 and limited to +/- 2.0. KHCAS is KDCKH - 0.5, limited to the range 0.0 to +1.0. If the option is not enabled, or if the autothrottle is not engaged, KDCKH is forced to 2.0, thus forcing KHCAS to 1.0.

If SELCAS is less than 120 knots, KVCAS is set to 1.0. Otherwise, KVCAS is computed as $1.2378 - 1.98165E-3 * SELCAS$. The basic airspeed gain (KV) is then computed as the square of KVCAS. KV varies from 1.0 at 120 knots to 0.32 at 340 knots (maximum speed for the Research Flight Deck). KV is used in procedure ELEVP for the elevator command (DECMD) and in procedure LATRL for the aileron command. KVCAS may optionally be used in CWS modes for the aileron command.

Ground speed / Time path (4D) options are computed next. The nominal ground speed command in feet per second (SDC) and the average acceleration along a path segment (SDD) are computed by procedure HVGUID when speeds have been entered in all waypoints of a 3D path. If SDC is zero (indicating ground speed guidance not possible), the autothrottle command derived from ground speed mode (SCMD) is set to zero. Otherwise, if GUID4D is false (indicating not time mode), SCMD is computed as follows:

$$SCMD = 0.1 * (SDC - GSFPS) + SDD$$

Where: GSFPS is the present ground speed in feet / second.

If GUID4D is true, SCMD is computed by procedure TGUID using the nominal ground speed plus a function of the time error. In either case, SCMD is used for the calculation of ATCMD when time path mode is selected on the Control Mode Panel (TIMPTH is true). No over- or under- speed protection is provided in these modes, so a check

is made here to see if the ground speed command is safe. If SELCAS is less than the reference airspeed (IASREF), or if ground speed is more than 10 knots less than IASREF, or if either KALFA or KHCAS is less than 0.5, the SPDLMT flag is set and otherwise cleared. When SPDLMT is true, logic in procedure MSPLGC causes a demode to CAS engaged with TIMPTH armed.

Airspeed (CAS) hold mode calculations begin by selecting the reference airspeed and storing it in TEMP. This will be IASSUM unless it is greater than CASHI, in which case CASHI is used. For negative commanded flight path angles (as indicated by NCL2), CASHI is first decreased by the magnitude of NCL2 in degrees. This is to give a bit of additional cushion when throttle reduction might not have much effect on airspeed. If airspeed hold mode is neither armed nor engaged (IASSEL and IASARM both false), IASSUM is set to SELCAS rounded to the nearest knot and DELCAS is set to zero. Otherwise (IASSEL or IASARM is true), the airspeed error (DELCAS) is computed as $TEMP - SELCAS$. TMP1, the actual input to the throttle command calculation, is first set equal to $IASREF - 10. - GSINS$ (the groundspeed floor calculation). If DELCAS is greater than (more likely to produce a positive acceleration than) TMP1, TMP1 is set to DELCAS.

Next, if alpha protection via throttle control is enabled (bit 0 of ATHSEL is set) and AUTO mode is true and KDCALF is less than 2.0, some throttle command adjustment will be done as follows. The temporary TMP2 is set to $KDCALF - 1.0$. If this quantity is negative (implying KALFA is less than 1.0), TMP2 is set to $10. * TMP2^2$, i.e., a positive, exponentially increasing value. If TMP2 is then greater than TMP1, TMP1 is set equal to TMP2. If $(KDCALF - 1.0)$ is positive (thus between 0.0 and 1.0) and TMP1 is negative (a deceleration command), TMP1 is set to $TMP1 * TMP2^2$, exponentially reducing TMP1 as TMP2 approaches zero.

If KDCKH is less than 2.0 (possible only if overspeed protection is enabled), TMP2 is set to $KDCKH - .5$. (Note: this is a coding error. It should be set to $KDCKH - 1.5$). If the result is negative, TMP2 is reset to $-10. * TMP2^2$. If TMP2 is then less than TMP1, TMP1 is set equal to TMP2. If the result is positive (which should imply a value between 0.0 and 0.5) and TMP1 is greater than zero (an acceleration command), TMP2 is set to $TMP1 * 2. * TMP2$, reducing TMP1 as TMP2 approaches zero. (The actual code will produce a window where TMP1 is increased when it should be decreased).

Finally, if TIMPTH is true, ATCMD is set to SCMD, and otherwise set to $KDCAS * TMP1$, where $KDCAS = 0.16878$. In either event, ATCMD is limited to ± 5.0 fps/sec.

GLOBAL INPUTS: ABACC, ALT, AUTOE, ATE, ATHSEL, CAS, DPERRF,
FLPNDX, GEAR, GSFPS, GSINS, GUID4D, IASREF, IASARM,
IASSEL, IASSUM, KALFA, KDCALF, LFCFLG, MACH, MAXSPD,
NAV64K, NCL2, PNAV64, SDC, SDD, SCMD, TIMPTH, TK, WD, WS

GLOBAL OUTPUTS: ATCMD, DELCAS, FMCASF, IASSUM, KCTOT, KHCAS, KV,
KVCAS, LFCFLG, MAXSPD, PNAV64, SELCAS, SCMD,
SPDLMT

210

MODULE NAME: ATHCL (Autothrottle Control Law)
FILE NAME: ATHCL.FOR
PROCESS: FCFAST

PURPOSE: To calculate the throttle command.

CALLED BY: FCFAST
CALLING SEQUENCE: CALL ATHCL
CALLS TO: None

DESCRIPTION:

ATHCL is called unconditionally on each cycle of FCFAST. There are three operating modes which depend on autothrottle engagement (ATE) and on the state of the FLARE flag. The routine first evaluates the indicators for autothrottle engagement and sets the ATE flag accordingly. Then the throttle position aft limit, AFTLIM, is set to zero during flare or whenever the airspeed is greater than 250 knots. Otherwise, it is increased as a function of airspeed in order to reduce "spool-up" time of the engines.

AFTLIM = 10 degrees, for CAS < 200 knots.
= .2 (250. - CAS), for 200<=CAS<=250 knots.
= 0, for CAS > 250 knots.

The complementary filtered longitudinal acceleration signal (NCI01) is derived by washing out the inertial longitudinal acceleration (VGSDOT) with the true airspeed signal (TASFPS).

TEMP = 5. (TASFPS - NCI01)
NCI01 = NCI01 + DELTAT (TEMP + VGSDOT)

This signal is then passed through a turbulence filter to limit high frequency components:

TEMP = TEMP - NCI02, limit = + 1.
NCI02 = NCI02 + (.2 DELTAT) TEMP

NCL1, the longitudinal acceleration damping signal, is computed as the sum of VGSDOT and NCI02, and limited to +/-16.0 feet per second per second (fps2).

If the autothrottle is engaged (ATE) and the FLARE flag is set, then the raw commanded autothrottle position rate, APCPRM, is set to -2.8. This will cause the position command integrator, NCI03, to be ramped back to zero at a 2.8 degrees per second rate.

If not ATE, APCPRM is set to 10.0 if the autothrottle feedback flag (ATFDBK) is set, otherwise, it is set to -10.0. This causes the autothrottle command to be synchronized to the throttle handle position.

If (ATE * .NOT. FLARE), then a series of actions occur. NCL3F, the vertical acceleration command to throttle feed forward signal is computed as:

$$NCL3F = NCL2 * \frac{\text{-----}}{(S + .25) (.5S + 1)}$$

Where: NCL2 is the nominal gamma command (in AUTO or VCWS), or the nominal pitch command (in ACWS). This filter is reinitialized whenever a mode change request occurs which could change the reference angle.

The raw throttle position rate command, APCPRM, is calculated as:

$$APCPRM = 3.0 (VSPHI - LAG16S + ATCMD - NCL1 + ATHFF * NCL3F)$$

Where: VSPHI = $5 - (5 \cos(\text{ROLL}))$
 LAG16S = last cycle filter output
 ATHFF = the gain on NCL3F (nominally 0.35)

VSPHI is then passed through a 16 second washout filter to generate a new roll compensation signal:

$$LAG16S = e^{*(-.05/16.0)} (LAG16S - VSPHI) + VSPHI.$$

The engine pressure ratio (EPR) limit calculation then controls the throttle position command so as not to exceed the limits for the engine with the highest current EPR. If the throttle command is positive, the gain (DELEPR) is calculated as the difference between the maximum EPR (MXEPR) and EPR, gained by 3.3333, and limited to 1.0. APCPRM is then gained by DELEPR. This attenuates the rate command as the engine approaches maximum output. However, if one EPR already exceeds MXEPR, (DELEPR less than zero), then the raw throttle command is immediately set to 10 times DELEPR.

For all modes, APCPRM is integrated to form NCI03, then limited to a value between 60 degrees and AFTLIM. Finally, the intermediate command is conditionally summed with the damping term to produce the final autothrottle position command, APCDG.

$$APCDG = 2.4 NCL1 - NCI03$$

If FLARE is set or CAS is greater than 250 knots,

$$APCDG = -NCI03.$$

GLOBAL INPUTS: ACWSS, ATCMD, ATE, ATFDBK, AUTOS, CAS, CROLL, EPR1, EPR2, FAIL2, FLARE, GRD, IASSEL, ICM, MANEL, MXEPR, NCL2, TASFPS, TIMPTH, VCWSS, VGSDOT

GLOBAL OUTPUTS: APCDG, ATE, CRSET, DSPLF

MODULE NAME: DSPOT (Displays Output)
 FILENAME: DSPOT.FOR
 PROCESS: FCFAST

PURPOSE: To compute analogs of the glideslope and localizer deviation variables to be used by the Displays task.

CALLED BY: FCFAST
 CALLING SEQUENCE: CALL DSPOT
 CALLS TO: GET_CHAR

DESCRIPTION:

Prior to glideslope beam sensed, the vertical path deviation (BETAH) is set to the negative of 3D guidance altitude error (HER). Prior to localizer beam sensed, the localizer error (ETAH) is set to the negative of 2D crosstrack error (XTK). BETAH and ETAH are negated to make them compatible with the analogous ILS signals.

Once within .7 degree of the glideslope, BETAH is defined as:

For MLS or GPS mode:

$$\text{BETAH} = \text{GSESCL} * \text{RTOD} * \begin{array}{l} \text{ZHAT} + \text{HGPIP} \\ \text{-----} \\ \text{XHAT} - \text{XGPIP} \end{array} - \text{TANGSA}$$

For ILS mode:

$$\text{BETAH} = \text{GSESCL} * \text{GSDEV}$$

Where: GSESCL is a selected gain to produce a standard course width at the threshold.

until proximity to the antenna makes calculation impossible. Then, BETAH = 0.

Once within 2.5 degrees of the runway centerline, ETAH is defined as:

For MLS or GPS mode:

$$\text{ETAH} = \text{LOCSCL} * \text{RTOD} * (\text{YPROF} - \text{YHAT}) / \text{XHAT}$$

For ILS mode:

$$\text{ETAH} = \text{LOCSCL} * \text{LOCDEV}$$

Where: LOCSCL is a selected gain to produce a standard course width at the threshold.

In any case, both BETAH and ETAH are limited to +/- 1000.

GLOBAL INPUTS: AIRPTS, DSTAT, GPLND, GSARM, GSDEV, GSENG, HER,
HGPIP, HRAD, LOCDEV, LFCFLG, LOCDEV, LOCVLD, MLSM, RLMLS,
SIMILS, TANGSA, XGPIP, XHAT, XTK, YHAT, YPROF, ZHAT

GLOBAL OUTPUTS: BETAH, ETAH, LFCFLG (DLBS, DVBS)

MODULE NAME: OUTIO (I/O Output Processing)
FILE NAME: OUTIO.MAR
PROCESS: FCFAST

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

CALLED BY: FCFAST
CALLING SEQUENCE: CALL OUTIO
CALLS TO: None

DESCRIPTION:

OUTIO is called once per major frame at the end of FCFAST major frame processing. It is responsible for formatting and packing output data into a form readable by external aircraft systems. Output data to be formatted is read from OUTCOM. OUTIO also gathers data recording data for output.

OUTIO uses the following programmer defined macros:

- 1.) FLOAT - Scales and converts a floating point value pointed to by R0 into a 16 bit integer and stores it at the destination specified as an input parameter.
- 2.) BOOL1 - Tests the byte pointed to by R0, and if set, will set the specified bit in the packed discrete as indicated by the input parameters.
- 3.) BOOL2 - Same as BOOL1 with the addition that the packed bit will also be set if the boolean LAMP is set.

GLOBAL INPUTS: All variables in OUTCOM.

GLOBAL OUTPUTS: DTC_OUT

6.1 SIGNAL FAILURE DETECTION

MODULE NAME: DINUSE (Discrete In-Use Computations)
 FILE NAME: DINUSE.MAR
 PROCESS: FCFAST

PURPOSE: To tell DISFD which discrete sensors it is required to check at any given time.

CALLED BY: FCFAST
 CALLING SEQUENCE: CALL DINUSE
 CALLS TO: None

DESCRIPTION:

Based on the current flight mode and condition of the aircraft, DINUSE sets the sensor-in-use bit (400 hexadecimal) of the appropriate sensor status word. Each discrete signal is allocated one status word. This bit is then detected by DISFD, which checks the signal validity if the in use bit is set.

The DSTAT array consists of 18 words in the following order:

1. HRV H-radar valid
2. LANDS Land select
3. AUTOS Auto select
4. VCWSS Velocity control wheel steering select
5. ACWSS Attitude control wheel steering select
6. GSVLD Glideslope valid
7. LOCVLD Localizer valid
8. LAMPS Lamp select for FC functions on MCP
9. IATTV IRS attitude valid
10. ADVAL Air Data computer valid
11. LOCFS Localizer frequency select
12. INAVV IRS navigation data valid
13. ASBV Air Data serial bus valid
14. ISBV IRS serial bus valid
15. TSBV TDS serial bus valid (used by MLS)
16. unused
17. unused
18. unused

GLOBAL INPUTS: DISPST, DLPSI, FCFLGS, FLARE, GSVLD, ILSZON,
 LANDR, LOCFS, LOCVLD, MLSM, MODEX

GLOBAL OUTPUTS: DSTAT

MODULE NAME: SINUSE (Sensor In-Use Computations)
 FILE NAME: SINUSE.MAR
 PROCESS: FCFEAST

PURPOSE: To tell FDSTR which digital sensor signals it is required to check at any given time.

CALLED BY: FCFEAST
 CALLING SEQUENCE: CALL SINUSE
 CALLS TO: None

DESCRIPTION:

Based on the current flight mode and state of the aircraft, SINUSE outputs four packed discrete words called SINUS0, SINUS1, SINUS2, and SINUS3, which contain information on which sensors are in use. This bit string is then unpacked sequentially into the sign bit of the elements of the STFAIL array (bit 15 of SINUS0 to bit 15 of STFAIL(1), bit 14 to bit 15 of STFAIL(2), etc.). Subroutines FDSTR and F2CMP reference the STFAIL array either directly or via the byte overlay STFBYT. The SINUS(i) words are not otherwise used, but may be recorded or referenced via the VIEW utility to see which sensors are marked 'in use' at any time. The sensor-to-bit assignments (and thus the STFAIL index assignments) are shown below. For a description of the bit usage in an STFAIL entry, see FAILCP.INC.

<u>BIT</u>	<u>SINUS0</u>	<u>SINUS1</u>	<u>SINUS2</u>	<u>SINUS3</u>
15	IR Q	IR VE	ADR HBARO	ADR B ALT 1
14	IR P	IR W SPD	ADR TAS	ADR B ALT 2
13	IR R	IR W DIR	ADR HDOT	
12	IR AX	HYB LATC	ADR CAS	
11	IR AY	HYB LONC	ADR MACH	
10	IR AN	IR HDG MAG	ADR TAT	
9	IR XDD	HYB LATF	ADR IND AOA	
8	IR XTK	HYB LONF	ADR COR AOA	
7	IR HDD	HYB ALT	ADR L ST PR	
6	IR LAT	HYB VN	ADR R ST PR	
5	IR LON	HYB VE	ADR U AV ST	
4	IR VGS	HYB STAT	ADR C AV ST	
3	IR TACHD	HYB HDOT	ADR TOT PRE	
2	IR THETA	IR VER SPD	ADR IMP PRE	
1	IR PHI	IR P VER S	ADR ST TEMP	
0	IR VN	IR ALT INI	ADR COR SLIP	

GLOBAL INPUTS: ATE, DSTAT

GLOBAL OUTPUTS: SINUS0, SINUS1, SINUS2, SINUS3, STFAIL

MODULE NAME: DISFD (Discrete Select and Failure Detect)
 FILE NAME: DISFD.MAR
 PROCESS: FCFAST

PURPOSE: To detect momentary discrepancies and failed sensor or serial bus errors for the packed discretetes.

CALLED BY: FCFAST
 CALLING SEQUENCE: CALL DISFD
 CALLS TO: None

DESCRIPTION:

Raw discretetes arrive from the triplicate sensors, packed in one word per sensor, via the DATAC bus. The three words are "debounced" over a period of 5 input cycles to ensure that any bit change was not a transient. The output is a single packed word with bits being set with slightly different logic depending on whether it is a select, a valid, or the duplex radar altimeter valid, HRV. The output is stored in VDISC for transmission to the Displays computer. In addition, the VDISC word is unpacked to set the respective booleans as shown below.

PACKED DISCRETES WORD FORMAT (VDISC)

<u>BIT</u>	<u>SIGNIFICANCE</u>
0	HRV (Duplex)
1	ISBV
2	LANDS
3	AUTOS
4	VCWSS
5	ACWSS
6	GSVLD
7	LOCVLD
8	LAMPS
9	IATTV
10	ADVAL
11	UNUSED
12	LOCFS
13	INAVV
14	UNUSED
15	ASBV

For selects, the individual debounced bits are compared with the majority logic vote from the debounce routine. If any select bit differs from the majority three times within 512 cycles, then a fail flag for that channel is set in the status word. A second channel failure causes the second fail flag to be set.

For the valids, if any bit is false 3 times within the 512 pass cycle, then the fail bit is set in the status word for that channel. A second fail results in the second fail bit being set in the status word.

Special processing is performed for the glideslope and

localizer valid (GSVLD, LOCVLD) discretizes if MLS or GPS land guidance has been selected (specified by the MLS configuration word, MCONF). If ILS type land capture algorithms under MLS or GPS guidance is specified (MCONF=x8xx), LOCFS is forced true and both LOCVLD and GSVLD are set according to the MLS azimuth and elevation valids, respectively (if MLSMOD is true) or according to the GPS land (GPLNDV) and glideslope (GPBTAV) valids if MLSMOD is false. If GPLND (GPS land selected) is true and MLSMOD is false, GSVLD is set according to GPBTAV regardless of capture mode.

If FLYFLG is true, special processing forces all required valids true.

DEBOUNCE ALGORITHM

```
A3 = A2
A2 = A1
A1 = A ⊕ MLO
A2 = A2 • A1
AF = (AF • .NOT. A3) + (A • A3)
```

(Repeat for channels B & C)

```
MLV = (AF • BF) + (AF • CF) + (BF • CF)
```

```
MLO = (MLO • PMLV) + (MLV • PMLV)
```

```
WHERE:  A    = Latest input
        A1   = Input N-1
        A2   = Input N-2
        A3   = Input N-3
        AF   = Debounced value
        MLO  = Majority Logic Output
        MLV  = Majority Logic Voted value
        PMLV = mlv from last pass
```

```
GLOBAL INPUTS:  ASBV, CRESET, CRSET, DISCW1, DISCW2, DISCW3,
                ERESET, FLYFLG, GPBTAV, GPLND, GPLNDV, IRSST1, IRSST2,
                ISBV, MCONF, MLSMOD, MLSSV
```

```
GLOBAL OUTPUTS: ACWSS, ADVAL, AUTOS, CRSET, DSTAT, ERSET, GSVLD,
                HRV, IATTV, INAVV, LAMP, LANDS, LOCFS, LOCVLD, MLO,
                VCWSS, VDISC, VORVLD
```

MODULE NAME: F2CMP (2nd Failure Computations)
 FILE NAME: F2CMP.FOR
 PROCESS: FCFAST

PURPOSE: Checks individual sensor second fails to determine if a mode failure is required.

CALLED BY: FCFAST
 CALLING SEQUENCE: CALL F2CMP
 CALLS TO: EXITF1, F2CMP2

DESCRIPTION:

Given the existing flight mode, this routine, together with its sub-procedures EXITF1, EXITF2 and F2CMP2, determines which sensors are critical for that mode and checks them for second failures. If a second failure is found, a failure of that mode is flagged and the appropriate mode failure message is displayed on the system test panel. This set of routines also clears all mode failure flags in the FAIL2 array upon CRSET and clears the error message tables FSTBL, TEST3, and STFAIL upon ERSET.

F2CMP contains three subroutines:

- F2CMP2 - Checks for second failures.
- EXITF1 - Tests for autothrottle Fail2 and displays error messages if required.
- EXITF2 - Clears all mode failure flags if CRSET is true, and/or clears all error messages if ERSET is true.

FAIL2 ARRAY

<u>ARRAY POSITION</u>	<u>SIGNIFICANCE</u>
1	AFCS
2	FFD
3	MANEL
4	ACWS
5	VCWS
6	AUTO
7	LAND
8	AUTO THROTTLE
9	MLS

GLOBAL INPUTS: None

GLOBAL OUTPUTS: None

MODULE NAME: F2CMP2
FILE NAME: F2CMP.FOR
PROCESS: FCFAST

PURPOSE: Checks individual sensor second fails to determine if a mode failure is required.

CALLED BY: F2CMP
CALLING SEQUENCE: CALL F2CMP2
CALLS TO: EXITF2

DESCRIPTION:

If the simulated airplane is active (FLYFLG is true) or MODEX is less than 1 (indicating a disengage situation), F2CMP2 merely sets the CLEARED flag for F2CMP and calls EXITF2. Otherwise it sets the local MODX variable equal to MODEX (limited to the range 1 - 7) and proceeds to check failures based on MODX and other indicators.

If MLSVAL is true, heading, pitch and roll inputs are checked for validity. If the body mounted accelerometers are not in use (BMAFLG false), IRS navigation valid (INAVV), attitude valid (IATTV), along track acceleration, cross track acceleration, vertical acceleration, ground speed, VN and VE are checked. If any of the above are failed, FAIL2(MLS) is set, which causes a loss of MLSVAL.

Next, the SEIU status, the Air Data computer status (ADVAL and ASBV, the bus status), and (if not MLS mode) the baro altimeter status are checked. Failure of any of these causes ALTNV (altitude not valid) to be set and the active mode (FAIL2(MODX)) to be failed. If the active mode is PRENG, processing concludes at this point.

Next AFCSV is checked and, if false, the active mode is failed. If the mode is MANEL, processing concludes here. Otherwise, processing continues by checking the status of CAS, Q, P, ROLL, ADVAL, ASBV and IATTV. If MODEX is less than five (VCWS), PITCH status is also checked. Again, failure of any of the above will cause mode failure. If the active mode is Forward Flight Deck (FFDE), processing terminates.

With MODEX \geq 4 (ACWS), processing continues by checking the status of the mode select discrettes (ACWSS, VCWSS, AUTOS, LANDS), and IRS ground speed. Failure of any of these causes mode failure. If ACWSE is true, processing is complete. Otherwise, vertical acceleration status is checked. If the mode is VCWS, IRS HDOT status is checked and, if not MLS mode and NCUVAL is false (indicating that valid velocity signals from the IRS are mandatory), cross track acceleration, VN and VE status are checked.

At this point, if VCWSE is false, the mode must be AUTO. Processing continues by checking for Horizontal path guidance selected (HORARM) and NCUVAL false. This condition will cause mode failure. (Track angle select or LAND select might still be legitimate.) Next, MODX is forced to 7 if LAND has been selected (LANDR true). Otherwise, return is made to F2CMP.

LAND processing continues by checking Radar altimeter valid (HRV) and, if full MLS has been selected (RLMLS and not SIMILS), either MLSMOD and MLSVAL or GPLND and GPLNDV must be true to maintain mode. If GPS LAND has been initialized (GPINIT true) then either GPS glide slope guidance (GPBTAV) or FLARE must be true if in AUTO mode. (LAND guidance can be maintained in VCWS with only valid localizer guidance.)

If Flight Controls is using MLS or GPS guidance (MLSM true) and either MLSMOD or GPLND is true, processing is complete. Otherwise, processing continues by checking PITCH, INAVV, THDG, VN and VE status. (Note that the above signals were already checked to maintain MLS valid. GPS valid may need to be investigated further.) If FLARE is false, glideslope, localizer and localizer frequency select statuses are checked. (Once FLARE is engaged, we are committed to touch down.) This completes F2CMP2 processing.

GLOBAL INPUTS: ACWSE, AFCSV, AUTOE, BMAFLG, DSTAT, FFDE, FLARE, FLYFLG, GPINIT, GPBTAV, GPLND, GPLNDV, HORARM, LANDR, MANEL, MLSM, MLSMOD, MLSVAL, MODEX, NCUVAL, PRENG, RLMLS, SIMILS, STFAIL, VCWSE

GLOBAL OUTPUTS: FAIL2

MODULE NAME: EXITF1
FILE NAME: F2CMP.FOR
PROCESS: FCFAST

PURPOSE: Performs auto throttle mode failure checks and outputs mode failure (demand) messages if indicated.

CALLED BY: F2CMP
CALLING SEQUENCE: CALL EXITF1
CALLS TO: EXITF2, FMTMG

DESCRIPTION:

EXITF1 begins by clearing the CALLD flag. This flag is used to ensure that only one mode failure message is output on a given entry. Next, if the autothrottle is engaged (ATE is true), the status of TAS, CAS, TAT, ADVAL, ASBV and the ALTNV flag (set by F2CMP2) is checked, and the failure of any signal used to fail the autothrottle mode. If NCUVAL is false, IRS nav valid (INAVV) and the status of the IRS along track acceleration signal is also checked.

Next, FRCNT is checked to see if a message has been output within the last 20 iterations. If FRCNT is non-zero, it is decremented, EXITF2 is called and a return is made to F2CMP. Otherwise, the FAIL2 array is scanned for any failure logged with the corresponding entry in the DSPLF array false (indicating that this failure has yet to be displayed). If none are found, exit is made as above. Otherwise, the mode fail and status alert bits are set in the LIGHTS word, DSPLF is set, FRCNT is set to 20 and FMTMG is called to display the failure message and transmit the LIGHTS code. EXITF2 is then called and processing ends.

GLOBAL INPUTS: ATE, DSPLF, DSTAT, FAIL2, NCUVAL, STFAIL

GLOBAL OUTPUTS: DSPLF, FAIL2, FSIDX, LIGHTS

MODULE NAME: EXITF2
FILE NAME: F2CMP.FOR
PROCESS: FCFAST

PURPOSE: Clear the appropriate error tables when CRSET or ERSET is set.

CALLED BY: EXITF1, F2CMP2
CALLING SEQUENCE: CALL EXITF2
CALLS TO: None

DESCRIPTION:

If CRSET or FLYFLG is true, the FAIL2 array is cleared. The clear on FLYFLG is to positively block mode failures when the simulated airplane is active. If ERSET is true, the STFAIL array and the signal failure counters in FAILBLK are cleared, and the failure read and status alert lamp bits in the LIGHTS word are cleared.

GLOBAL INPUTS: CRSET, ERSET, FLYFLG

GLOBAL OUTPUTS: FAIL2, LIGHTS, STFAIL

MODULE NAME: FDSTR (Failure Data Storage)
 FILE NAME: FDSTR.FOR
 PROCESS: FCFAST

PURPOSE: Examines sensor status to detect failure and records failure information in a table of failure data.

CALLED BY: FCFAST
 CALLING SEQUENCE: CALL FDSTR
 CALLS TO: DSTOR, TEST_SENSORS

DESCRIPTION:

This routine has four major parts. The first section examines aircraft state and the System Test Panel switch word (FALST) to determine whether flight or preflight tests are being performed, and sets the STORE pointer to the appropriate failure table. The second part checks the digital ADIRS and GPIRS sensors. Inertial and/or air data sensors may be checked depending on the state of the validity flags, INAVV, IATTV and ADVAL. These checks are performed by the sub-procedure TEST_SENSORS. The third part tests the discrete signals. Lastly five separate tests are run on the Sensor-Effector Interface Unit (SEIU), Research Flight Deck Interface Unit (RFDIU) System and a test on the DATAC link between the FM/FC and DS/DF VAX computers.

All sensors are checked for their failure status using a similar procedure. Each sensor is scanned initially for second failures and then for first fails. For the present system, virtually all sensor inputs are simplex, therefore first and second failures are synonymous. The exceptions are the radar altimeter (duplex) and certain Control Mode Panel switches (triplex). If a failure is indicated and has not already been recorded, the needed failure data is formatted into 8 1/2 words and stored in a failure data table by procedure DSTOR for later retrieval by FMTMG when a failure message display is requested.

A series of tests is performed on the SEIU/RFDIU system. The first test checks communication between the VAX and the SEIU. Second is a check for power failure on the SEIU. Third, a test is run on the six SEIU and two RFDIU analog to digital converters. Fourth is a test of VAX to RFDIU communication. Fifth, a test for RFDIU power fail is made. Finally, a test of the DATAC link from the DS/DF to the FM/FC VAX is made.

STFAIL Format:

Array Position	Signal	Array Position	Signal
1	Q	34	TAS
2	P	35	HDOT
3	R	36	CAS
4	AX	37	MACH
5	AY	38	TAT
6	AN	39	IND AOA
7	ATKINS	40	COR AOA

8	XTKINS	41	L ST PR
9	HDDINS	42	R ST PR
10	LATINS	43	U AV ST
11	LONINS	44	C AV ST
12	GSINS	45	TOT PRE
13	THDG	46	IMP PRE
14	PITCH	47	ST TEMP
15	ROLL	48	COR SLIP
16	VNINS	49	B ALT 1
17	VEINS	50	B ALT 2
18	IRSW	51	HRV
19	IRSWD	52	ISBV
20	HYBLAT	53	AUTOS
21	HYBLON	54	VCWSS
22	MAGHDG (ADIRS)	55	ACWSS
23	HYBLAT_FIN	56	GSVLD
24	HYBLON_FIN	57	LOCVLD
25	HYBALT	58	LAMPS
26	HYBVN	59	IATTV
27	HYBVE	60	ADVAL
28	GRSST	61	LOCFS
29	HYBHDT	62	INAVV
30	IRSHDOT (ADIRS)	63	ASBV
31	P VER S (N/U)	64	ISBV
32	IRSALT	65	TSBV
33	HBARO	66-68	N/U

This routine shares common local data with DSTOR, PANEL, FMTMG, and F2CMP.

GLOBAL INPUTS: ADVAL, CRSET, DSTAT, DTC_IN, FALST, IATTV, INAVV, INSST, MSWIT, PRENG, SQUAT, STFAIL, WSPIN

GLOBAL OUTPUTS: DSTAT, FIDENT, LIGHTS, MSWIT, PMSWIT, STFAIL

226

MODULE NAME: TEST_SENSORS
FILE NAME: FDSTR.FOR
PROCESS: FCFAST

PURPOSE: Performs error checks on ADIRS and GPIRS input data.

CALLED BY: FDSTR
CALLING SEQUENCE: CALL TEST_SENSORS
CALLS TO: DSTOR

DESCRIPTION:

Each sensor within the range indicated by NDX and COUNTER (set by FDSTR) is initially checked to see if it is in use (indicated by the sign bit of STFALL) and if a parity error has already been stored. If either of the above is false, the pointer NDX is incremented and the next sensor is checked. Otherwise, the parity and validity of the sensor is checked. If a failure is detected, a counter is incremented, otherwise it is cleared. A separate count of parity and validity errors is maintained, each occupying one byte of the STFALL entry pointed to by NDX. If either counter reaches three, the failure read bit in LIGHTS is set, DSTOR is called to log the error and the 'failure stored' bit (0080x for parity, 4000x for validity) is set.

GLOBAL INPUTS: DTC_IN, DTC_OUT, LIGHTS, STFALL

GLOBAL OUTPUTS: LIGHTS, STFALL

MODULE NAME: FMTMG (Format Message)
 FILE NAME: FMTMG.FOR
 PROCESS: FCFAST

PURPOSE: Formats failure messages from the form stored in the failure data tables into an ASCII form which can be output to the system test panel by GMSG.

CALLED BY: CTLCK, DETNT (MLOG), ILSRC, PANEL, PRFLT

CALLING SEQUENCE: CALL FMTMG (P1, P2)

Where: P1 = address of the message to be formatted.

P2 = the index into the failure data table (FDSTR) which contains data to be added to the basic message. P2 is required only if the message is of the type requiring additional data.

CALLS TO: ICO

DESCRIPTION:

Five different types of messages are formatted by FMTMG. The first byte of the message to be formatted contains a control character that denotes its type. The four control characters and their meanings are:

<space> - Demand message. Format only the message text. This type of failure is not stored in the failure data tables. Contains up to 32 characters of text.

< 1 > - Analog sensor failure. Format time of failure, nine characters of message text, channel failed, flight mode when failure occurred, analog failed value, and analog voted value.

< 2 > - Discrete sensor failure. Format time of failure, seventeen characters of message text, channel failed, flight mode when failure occurred, discrete failed value, and discrete voted value.

< 3 > - General purpose failure. Format time of failure, seventeen characters of message text, failed channel and flight mode when failure occurred.

< 4 > - Digital sensor failure (ADIRS/GPIRS). Format time of failure, seventeen characters of message text and either "PAR" or "VLD" depending on whether a parity or validity

error occurred.

This routine uses a local subroutine called ICO to convert binary values into decimal ASCII. It shares common local data with FDSTR, DSTOR, PANEL, and F2CMP.

GLOBAL INPUTS: LIGHTS, MSWIT

GLOBAL OUTPUTS: MSBUF, MSGST, WRDCNT

MODULE NAME: ICO
FILE NAME: FMTMG.FOR
PROCESS: FCFAST

PURPOSE: Convert input integer data (I*2) to 5 ASCII digits.

CALLED BY: FDSTR
CALLING SEQUENCE: CALL ICO(NUMBER, INDX)

Where: NUMBER is the number of digits to be converted.
INDX is the pointer to the first digit in MSBUF.

CALLS TO: None

DESCRIPTION:

This subroutine converts the failed and selected values to their ASCII representation. Since the number can be up to 5 digits, the routine puts the sign in MSBBYT, then starts inserting the converted number from the rear. E.g, the first time through the index is 23. The sign is put in MSBBYT(23). Then, the index is incremented to 28 and a digit is converted and stored. The index is decremented and the process is repeated 4 times.

GLOBAL INPUTS: None

GLOBAL OUTPUTS: MSBUF, MSBBYT

Where: MSBBYT is the byte equivalent of MSBUF, the output message buffer.

MODULE NAME: DSTOR (Data Storage for STP)
 FILE NAME: DSTOR.FOR
 PROCESS: FCFAST

PURPOSE: Stores data for error messages into the failure data table (FDTBL).

CALLED BY: FDSTR, PREFLT(ILSRC, SRVCK)
 CALLING SEQUENCE: CALL DSTOR(P1, P2, P3, P4)

Where: P1 - Address of the failure message
 P2 - Channel (for formats 2 and 3)
 P3 - Failed value
 P4 - Voted value

CALLS TO: None

DESCRIPTION:

DSTOR packs 17 of 19 bytes of information needed to create an error message into a data area for retrieval later by FMTMG. The bytes are stored in records which are defined as:

Field 1	Failure ID (address of failure message in MESSG)	4 bytes
Field 2	Sensor channel failed / mode at fail time (MODEX)	2 bytes
Field 3	Hour of failure	1 byte
Field 4	Minute of failure	1 byte
Field 5	Second of failure	1 byte
Field 6	Failed data value	2 bytes
Field 7	Voted data value	2 bytes
Field 8	Reset **	2 bytes

** The information for field 1 is stored by FDSTR when the reset button is pressed.

This routine shares common local data with FDSTR, PANEL, FMTMG, and F2CMP.

GLOBAL INPUTS: HRSS, MINS, MODE2, MSWIT

GLOBAL OUTPUTS: FDTBL

MODULE NAME: PANEL (System Test Panel Interface)
 FILE NAME: PANEL.FOR
 PROCESS: FCFAST

PURPOSE: Service the System Test Panel requests.

CALLED BY: FCFAST
 CALLING SEQUENCE: CALL PANEL
 CALLS TO: FMTMG

DESCRIPTION:

PANEL sets bits in 'SWITCH' according to the hardware status word (FALST) from the System Test Panel to determine which button was depressed on the panel. The bits are defined as:

<u>Bit</u>	<u>Button pushed</u>	<u>Description</u>
15	Panel Test	Produces a momentary illumination of all lights and display segments for test purposes.
14	Panel Reset	Results in a hardware reset of the System Test Panel. If this button is depressed, PANEL is exited with with no further button checks.
13	Mode Fail	Turns off the Mode Fail lamp.
12	Status Alert	Displays Status Alert (mode-failure) messages on the System Test Panel.
10	Data Clear	Clears Failure Data table (FDSTR) and blanks the display.
8	Failure Read	Displays recorded sensor failure messages on the System Test Panel.

This routine shares common local data (FAILBLK) with FDSTR, DSTOR, FMTMG, and F2CMP.

GLOBAL INPUTS: FALST, FSIDX, LIGHTS, MSWIT, SWITCH, WRDCNT

GLOBAL OUTPUTS: LIGHTS, MSBUF, MSGST, SWITCH, WRDCNT

Section 6.2 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 DSTDAT.MAR, linked with the process DSTAR, 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. DSTAR 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. DSTAR processes the recording list information and stores addresses and scale factors for the DAS. The module SNAP works with tables generated through DSTAR. When a user specified condition is encountered, SNAP saves the associated set of data items. Subsequently, the background module SNAPOUT prints the data values to the line printer.

The subroutine DASOT takes the data specified in the DAS lists, formats it, and stores it in the 150 words of DAS output memory. The strip chart data are also included in the DASOT output, which is routed to the onboard strip charts by the DAS.

The output from SNAP 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: DASOT (DAS Recording Output)
 FILE NAME: DASOT.MAR
 PROCESS: FCFAST

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

CALLED BY: FCFAST
 CALLING SEQUENCE: CALL DASOT
 CALLS TO: None

DESCRIPTION:

DASOT first checks the boolean NODAS to see if the recording tables are being modified by DSTAR. If NODAS is true, no further processing is performed. Otherwise, the globals RECWD, RECWD1, RECWD2, and RSWADR are checked to determine which set of alternate tables should be stored in the global DASPARG 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 DASPARG. This will happen when the alternate tables have been changed through DSTAR and bit 15 of RECWD is set. The values in RECWD1, RECWD2, and RSWADR are user specified through task VIEW as follows:

RSWADR: USAGE

CLEAR = The primary set of alternate tables (RECWD1) will be written to the DASLST strip chart blocks.
 ADDRESS = The address of some discrete (e.g, MLSVAL) which will, when true, cause the secondary set of alternate tables (RECWD2) to be used. (NOTE: the VIEW and FCFAST processes must be identically linked for this to work.)

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 Alt 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 0154 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.

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 was loaded by DSTAR from DSTDAT and/or from user input. ALTPAR 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    Set 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 24 entries (8 x 3) in DASPARE, 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 DISOUT are listed locally at label DISLST. The sign bit of each boolean is shifted left into a register which finally is shifted to place the bits at 0:9, and moved into DISOUT. Next, the position errors are computed as follows:

```
MLS-VALID:
LATDIF = IDDLAT - MLSLAT
LONDIF = IDDLON - MLSLON
ALTDIF = IDDALT - MLSALT
```

```
ELSE:
LATDIF = IDDLAT - LATINS
LONDIF = IDDLON - LONINS
ALTDIF = IDDALT - IRSALT.
```

Next, IDDLAT and IDDLON are converted to 32 bit integer data and output as IDDLTF and IDDLNF. Finally, XHAT and YHAT are scaled (x 4) and stored off as the longwords XHTFIN and YHTFIN.

GLOBAL INPUTS: ALTPAR, IDDALT, IDDLAT, IDDLON, IRSALT, LATINS,
LONINS, MLSALT, MLSLAT, MLSLON, RECWD, RECWD1, RECWD2,
RSWADR, XHAT, YHAT

GLOBAL OUTPUTS: ALTDIF, DASPARG, DISOUT, IDDLTF, IDDLNF, LATDIF,
LONDIF, RECWD, XHTFIN, YHTFIN

MODULE NAME: SNAP (Snap Data Storage)
FILE NAME: SNAP.FOR
PROCESS: FCFAST

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

CALLED BY: FCFAST
CALLING SEQUENCE: CALL SNAP
CALLS TO: DUMPS, GET_BYTE, GET_REAL, GET_WORD

DESCRIPTION:

The DSNAP routine records single-event values, called snapshots, for selected variables and stores them in SNAPBUF, in RECCOM, 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 RECCOM listing.) These tables are set up by the user through the DSTAR process. DSNAP checks as many snap tables as have been defined in DSTAR (indicated by SMENT). 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.

GLOBAL INPUTS: NOSNAP, SCRIT, SMENT, SPTR, SRST

GLOBAL OUTPUTS: RPTR, SCRIT, SPTR, SRST

MODULE NAME: DUMPS
FILE NAME: SNAP.FOR
PROCESS: FCFAST

PURPOSE: To load the snapshot data specified by the SCRIT table entry 'index' into the SNAPBUF table for subsequent output by the SLOW loop procedure SNAPOUT.

CALLED BY: SNAP
CALLING SEQUENCE: CALL DUMPS(index)
CALLS TO: GET_BYTE, GET_REAL, GET_WORD

DESCRIPTION:

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 INPUTS: SCRIT, SPTR

GLOBAL OUTPUTS: SPTR, SNAPBUF

6.3 PREFLIGHT OVERVIEW

The Preflight software performs an automatic operational test of various flight control systems. Upon initiating this procedure several tests are made on the system in parallel. These include stimulating and checking sensors and servos, checking sensor valids, and testing pilot control inputs. After concluding, preflight will list all failures found on the system test panel.

Subroutine PRFLT contains the control logic for the automated preflight tests. There are seven routines associated with pre-flight, all of which are called from FCFEAST when MSWIT (set in FDSTR) = 1. Note that when pre-flight is active the calls to MLSEX, LATRL, ELEVP, DINUSE and SINUSE are bypassed, and MLOG processing is minimal. The other routines and their functions are as follows.

- CTLCK - checks the AFD wheel column, rudder pedal and trim inputs.
- CLBIS - sets values in AILCMD, RUDCMD and DECMD.
- RDALT - tests the radar altimeters.
- ILSRC - tests the ILS localizer and glideslope receivers.
- RGYRO - tests the rate gyros (yaw, pitch, roll).
- SRVCK - tests the aileron, spoiler panel, rudder, elevator and stabilizer position inputs.

Procedure FDSTR is active during pre-flight and logs errors resulting from sensor disagreement. Errors of magnitude are logged by the preflight routines directly.

MODULE NAME: PRFLT (Pre-flight Test Executive)
 FILE NAME: PRFLT.MAR
 PROCESS: FCFAST

PURPOSE: To initiate and control the system preflight tests.

CALLED BY: FCFAST
 CALLING SEQUENCE: CALL PRFLT
 CALLS TO: CLBIS, CTLCK, FMTMG, ILSRC, RDALT, RGYRO, SRVCK

DESCRIPTION:

The preflight system is composed of six subroutines that are called serially from PRFLT (if in preflight mode) in the following order:

JSB	CTLCK	; Control check test.
JSB	CLBIS	; Control bias procedure.
JSB	RDALT	; Radio altimeter test.
JSB	ILSRC	; ILS self test.
JSB	RGYRO	; Rate gyro test.
JSB	SRVCK	; Servo displacement test.

Each of the test modules performs a certain system check by stimulating individual sensors and then checking for the proper response. The tests and procedures are controlled by a packed test inhibit word called INHIBT. Each routine has its own bit which, if set, suspends that test from running. There is also a master inhibit bit which, if set, inhibits all preflight tests from running.

The bits in the INHIBT word are packed as follows:

<u>BIT</u>	<u>USAGE</u>
0	PRFLT
1	CLBIS
2	RDALT
3	ILSRC
4	RGYRO
5	CTLCK
6	SRVCK
7-14	N/U
15	MASTER INHIBIT

The module PRFLT is called once the system test panel mode switch is moved from flight to preflight and the routine FDSTR has acknowledged that the preflight mode is possible (by checking for the presence of SQUAT and not WSPIN). The module PRFLT on its first pass performs initialization of preflight variables and flags including setting the master inhibit bit to prevent the remainder of the modules from running. PRFLT itself is immune to the master inhibit bit. PRFLT then checks whether the Aft Flight Deck (AFD) is engaged. If not, a message is displayed to prompt the operator to engage the AFD paddle. Once engaged, a message is displayed to

prompt the user to push the start test button. PRFLT then waits for this response, then clears the INHIBT word so that all tests begin running. After all the tests are completed (each test sets its own INHIBT bit when finished) PRFLT tests if any failures were detected. If so a message is displayed to alert the operator.

This routine shares common local data with all other preflight modules.

GLOBAL INPUTS: AEE, AFCSV, FAIL2, LIGHTS, MMSG (pool of messages), MSWIT, PMSWIT, SWITCH, WRDCNT

GLOBAL OUTPUTS: AEE, FLAGS, LIGHTS, MODEX, MODE2, MSGST, MSBUF, STRUA, STRUB, STRUC, SWITCH, WRDCNT

MODULE NAME: CLBIS (Control Surface Bias Outputs)
FILE NAME: PREFLT.MAR
PROCESS: FCFAST

PURPOSE: This preflight procedure sets the airplane control surfaces for testing by SRVCK.

CALLED BY: PREFLT
CALLING SEQUENCE: JSB CLBIS
CALLS TO: None

DESCRIPTION:

CLBIS is one of seven preflight modules that are all associated. For an overview of the whole preflight system see the description of PREFLT. If its INHIBT bit is set this program is exited. If not, CLBIS will either reset the control surfaces (aileron, rudder, and elevator) or set them to test values. This is determined by the flag RST (local variable to the preflight routines) which is controlled by the SRVCK test. The test values which the control surfaces are set to are the following:

AILERON : 5 deg left wing down
RUDDER : 10 deg left yaw
ELEVATOR : 13.84 deg fly up

This routine shares common local data with all other preflight modules.

GLOBAL INPUTS: None

GLOBAL OUTPUTS: AILCMD, DECMD, RUDCMD

MODULE NAME: CTLCK (Controls Check)
 FILE NAME: PRFLT.MAR
 PROCESS: FCFAST

PURPOSE: This preflight test checks the various controls for proper displacement by the operator.

CALLED BY: PRFLT
 CALLING SEQUENCE: JSB CTLCK
 CALLS TO: FMTMG

DESCRIPTION:

CTLCK is one of seven preflight modules that are all associated. For an overview of the whole preflight system see the description of PRFLT. If this test's inhibit bit is cleared the following prompts will be issued one at a time on the system test panel, waiting each time for the requested stimulus.

TURN RUDDR TRM RT-LT	(Turn rudder trim knob right and left. Input must exceed 7.4 PU.)
PUSH RUDDR PEDALS RT-LT	(Push rudder pedals right and left. Input must exceed 2.64 deg.)
TURN AILRN TRM RT-LT	(Turn aileron trim knob right and left. Input must exceed 5.05 PU.)
ROLL PMC RWD-LWD	(Turn wheel right and left. Input must exceed 27.7 deg.)
PUSH-PULL PMC ND-NU	(Push-pull column. Input must exceed 1.7 in.)
RE-CTR AILRN TRM	(Re-center aileron trim: < 1.01 PU.)
RE-CTR RUDDR TRM	(Re-center rudder trim: < 0.83 PU.)

This program contains two local subroutines called DISP and NULL. DISP performs the first five displacement tests above. NULL performs the remaining two re-centering tests.

This routine shares common local data with all other preflight modules.

GLOBAL INPUTS: ATRIM, PEDAL, DCOL, DCOL, MMSG (pool of messages), PEDAL, RTRIM, WRDCNT

GLOBAL OUTPUTS: FLAGS

MODULE NAME: ILSRC (ILS Receiver Checks)
FILE NAME: PRFLT.MAR
PROCESS: FCFEAST

PURPOSE: To test the ILS receivers during preflight testing.

CALLED BY: PRFLT
CALLING SEQUENCE: ILSRC
CALLS TO: FMTMG, DSTOR

DESCRIPTION:

ILSRC is one of seven preflight modules that are all associated. For an overview of the whole preflight system see the description of PRFLT. If this test's INHIBT bit is clear, this module first checks for the presence of LOCALizer Frequency Selected (LOCFS). If not present, a message is displayed to alert the operator and the test is aborted. Otherwise, the signal valids for the localizer and glide-slope are checked. Provided these are found to be true, the localizer and glide-slope receivers are both tested for the proper response to the stimulus provided by PRFLT immediately after pushing the start-test button (LOC between 0.8911 and 1.074 deg and GS between 0.3274 and 0.4126 deg). If the response is not within expected limits an error is flagged.

This routine shares common local data with all other preflight modules.

GLOBAL INPUTS: GSDEV, GSVLD, LOCDEV, LOCFS, LOCVLD, MMSG (pool of messages)

GLOBAL OUTPUTS: FIDENT, FLAGS, STRUA, STRUB, STRUC

244

MODULE NAME: RDALT (Radar Altimeter Checks)
FILE NAME: PRFLT.MAR
PROCESS: FCFAST

PURPOSE: To test the radio altimeters during preflight.

CALLED BY: PRFLT
CALLING SEQUENCE: JSB RDALT
CALLS TO: None

DESCRIPTION:

RDALT is one of seven preflight modules that are all associated. For an overview of the whole preflight system see the description of PRFLT. If this test's INHIBT bit is not set this test checks both radio altimeters for proper response to an artificial 100 foot stimulus (failing if not between 95 and 105 feet) and ground level (failing if greater than 2 feet).

This routine shares common local data with all other preflight modules.

GLOBAL INPUTS: DTC_IN

GLOBAL OUTPUTS: DSTAT, DTC_OUT, STRUA, STRUB

MODULE NAME: RGYRO (Rate Gyro Checks)
FILE NAME: PRFLT.MAR
PROCESS: FCFAST

PURPOSE: To test the pitch, roll, and yaw rate gyros during preflight testing.

CALLED BY: PRFLT
CALLING SEQUENCE: JSB RGYRO
CALLS TO: None

DESCRIPTION:

RGYRO is one of seven preflight modules that are all associated. For an overview of the whole preflight system see the description of PRFLT. If this test's INHIBT bit is set, the program is exited. If not, each of the three rate gyros are checked against values that are expected (between 0.85 and 1.43 deg/sec) as a result of stimuli supplied by PRFLT immediately after pushing the start-test button. This is done by the local subroutine RGTST. After stimulation, the gyros are checked for null values (below 0.7 deg/sec). This is performed by the local subroutine NORM. Any errors are recorded for display by the system test panel.

This routine shares common local data with all other preflight modules.

GLOBAL INPUTS: P, Q, R

GLOBAL OUTPUTS: DTC_OUT, STRUA, STRUB, STRUC

MODULE NAME: SRVCK (Servo Checks)
 FILE NAME: PRFLT.MAR
 PROCESS: FCFAST

PURPOSE: Tests the aileron, elevator, and rudder servos during preflight testing.

CALLED BY: PRFLT
 CALLING SEQUENCE: JSB SRVCK
 CALLS TO: DSTOR

DESCRIPTION:

SRVCK is one of seven preflight modules that are all associated. For an overview of the whole preflight system see the description of PRFLT. If this test's INHIBT bit is not set, one of three possible paths is chosen. If it has been less than three seconds since the start-test button was pushed the test is exited. At exactly three seconds the boolean RST is cleared and the INHIBT bit for CLBIS is toggled so that it may clear AILCMD, RUDCMD, and DECMD, which had previously been set to static test values at start-test time. After three seconds from start-test, four local subroutines are called serially to test the servos. These four tests are listed below.

- 1.) LATA - Tests aileron position and rate of change of spoiler position. Commands 5 deg aileron and fails if response after three seconds is not between 4.0 and 5.5 deg, or if spoiler change is not between 3.0 and 12.0 deg.
- 2.) ELEVT - Tests elevator position. Commands 5 deg elevator and fails if response after three seconds is not between 4.5 and 5.5 deg.
- 3.) HZSTB - Tests the stabilizer trim. Sets trim true and checks change in stabilizer after twenty seconds. Failure is indicated if change is not between 0.5 and 0.7 deg for flaps < 2 deg, or 2.9 and 4.3 deg for flaps >= 2 deg.
- 4.) AFRUD - Tests rudder position. Commands 5 deg rudder and fails if response is not between 4.0 and 6.0 deg after three seconds.

If any discrepancies are noticed between actual and expected values, an error is recorded for later display on the System Test Panel.

This routine shares common local data with all other preflight modules.

GLOBAL INPUTS: ALVDT, DEPOS, DRPOS, FLAP, MESH (pool of
messages), SPR7, STABP

GLOBAL OUTPUTS: AILCMD, DECMD, FIDENT, LIGHTS, RUDCMD, TRIMD,
TRIMT

7.0 DAS/SNAP TABLE ACCESS

MODULE NAME: DSTAR (DAS/Snap Table Access Routine)
 FILE NAME: DSTAR.FOR
 PROCESS: DSTAR

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

CALLED BY: A: HDL (on cold start)
 B: The User (manually)

CALLING SEQUENCE:

A. VAXHDL:
 DS_PROC_NAME: .ASCID /DSTAR/

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 DSTAR

CALLS TO: DASDUMP, DASPRC, FOR\$CLOSE, FOR\$DATE_T_DS,
 FOR\$OPEN, MAPCOM, SNAPDUMP, SNPRC

DESCRIPTION:

The primary function of DSTAR is to load the tables (DASPAR & ALTPAR) used by DASOT 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 DSTAR is called by the I/O handler (HDL). In this case, DAS processing is enabled for whatever data is defined in the default DAS list, nominally DSTDAT. This is an external file included in the DSTAR process which is documented separately in this volume. (DSTAR defines DSTDAT 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 DSTDAT.)

The secondary function of DSTAR 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 DSTAR. DSTAR 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 150 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 24 are used for the strip chart parameters and are organized in three 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 DASOT. These alternate tables are maintained in the lower section of DSTDAT, 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 DASPARG and ALTPARG 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 DSTDAT 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, DSTAR 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 DSTAR. 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 DSTAR module includes 11 subroutines and calls one external subroutine (SYM_SEARCH).

250

GLOBAL INPUTS: ALTDMP, CHCNT, COLDST, DASDMP, DDATE, PRINTOUT,
SNAPDMP, SNENT, TERM, TUTOR

GLOBAL OUTPUTS: GETNAME, NOSNAP, RECWD1, SNAPACT

MODULE NAME: NSNAP (New Snap Processing)
FILE NAME: DSTAR.FOR
PROCESS: DSTAR

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

CALLED BY: SNPRC
CALLING SEQUENCE: CALL NSNAP

Alternate entry point: CALL GETKEY

CALLS TO: CHECK, GITEM, SYM_SEARCH, UCASE

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 DSTAR 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 INPUTS: ADR, CBUF, CHCNT, LGTH, LPAREN, NNAME, NNENT,
RPAREN, SDONE, TUTOR, TYP

GLOBAL OUTPUTS: ERROR, ITNUM, MAX, SNENT, TABLE, TERM

252

MODULE NAME: DASDUMP (Dump Recording Tables)
FILE NAME: DSTAR.FOR
PROCESS: DSTAR

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

CALLED BY: DSTAR
CALLING SEQUENCE: CALL DASDUMP
CALLS TO: FMPTIM, FOR\$IMVBITS

DESCRIPTION:

If the DASDMP flag is true, the DAS recording parameters are transferred to the DASPARG buffer from the temporary area DDATA. Since this affects the data recording process, the NODAS flag is set to inhibit data recording while DASPARG 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 INPUTS: ALTDMP, ATABL, DASDMP, DDATA, DDATE, DNENT,
NNAME, PRINTOUT, TIME

GLOBAL OUTPUTS: ALTPAR, DASPARG, NODAS, RECWD

MODULE NAME: DASPRC (DAS Table Processing)
FILE NAME: DSTAR.FOR
PROCESS: DSTAR

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

CALLED BY: DDSTAR
CALLING SEQUENCE: CALL DASPRC
CALLS TO: FOR\$CLOSE, FOR\$OPEN, GITEM, UCASE

DESCRIPTION:

This module serves as the user interface to DSTAR 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 INPUTS: CBUF, CHCNT, DDATE, DNENT, ITNUM, TUTOR

GLOBAL OUTPUTS: ACNT, ALFLG, DASDMP, DASFLG, DCNT, ERROR,
GETDAS, GETOLD, MAX, TABLE, TERM

MODULE NAME: GITEM (Get Item Definition)
FILE NAME: DSTAR.FOR
PROCESS: DSTAR

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

CALLED BY: DASPRC, NSNAP, SNAPMOD
CALLING SEQUENCE: CALL GITEM
CALLS TO: CHECK, SYM_SEARCH, UCASE

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 DSTAR 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 INPUTS: ACNT, ADR, ALTFLG, ASTER, CBUF, CHCNT, DASFLG,
DCNT, GETNAME, LGTH, LPAREN, MAX, REPEAT, RPAREN, SDONE,
SMOD, SNAPACT, TABLE, TYP

GLOBAL OUTPUTS: DNENT, ERROR, GETDAS, GETOLD, ITNUM, SNENT, TERM

MODULE NAME: CHECK (Check for Multi-item Entry)
FILE NAME: DSTAR.FOR
PROCESS: DSTAR

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

CALLED BY: GITEM, NSNAP, OLDSNAP
CALLING SEQUENCE: CALL CHECK
CALLS TO: None

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 INPUTS: CBUF, LPAREN, RPAREN

GLOBAL OUTPUTS: ASTER, CHCNT, ERROR

256

MODULE NAME: SNAPDEL (Snap Delete)
FILE NAME: DSTAR.FOR
PROCESS: DSTAR

PURPOSE: To delete a previously entered snap table.

CALLED BY: SNPRC
CALLING SEQUENCE: CALL SNAPDEL
CALLS TO: FOR\$BITEST

DESCRIPTION:

This routine is used to delete a snap table from the current working set of DSTAR 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 INPUTS: None

GLOBAL OUTPUTS: DEL, SCRIT, SNAME, SNENT

MODULE NAME: SYM_SEARCH (Search Symbol Table)
 FILE NAME: SYM_SEARCH.FOR
 PROCESS: DSTAR

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

CALLED BY: DSTAR
 CALLING SEQUENCE: CALL SYM_SEARCH(SYMBOL, ADDRESS, FORM, SIZE)
 CALLS TO: GET_BYTE, GET_LONG, STR\$COMPARE

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 hexadecimal
- 5 ASCII

The symbol search starts at the beginning 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.

258

GLOBAL INPUTS: SYMNUM, SYMTAB

GLOBAL OUTPUTS: None

MODULE NAME: SNAPMOD (Snap Modification)
FILE NAME: DSTAR.FOR
PROCESS: DSTAR

PURPOSE: To modify an existing snap table.

CALLED BY: SNPRC
CALLING SEQUENCE: CALL SNAPMOD
CALLS TO: GETKEY, GITEM, SNAPDUMP, UCASE

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 INPUTS: CBUF, CHCNT, MOD

GLOBAL OUTPUTS: ERROR, GETNAME, ITNUM, MAX, NNENT, SCRIT,
SMOD, SNAME, TABLE, TERM

260

MODULE NAME: SNPRC (Snap Processing)
FILE NAME: DSTAR.FOR
PROCESS: DSTAR

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

CALLED BY: DDSTAR
CALLING SEQUENCE: CALL SNPRC
CALLS TO: NSNAP, OLDSNAP, SNAPDEL, SNAPMOD

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 INPUTS: CHCNT, SNENT

GLOBAL OUTPUTS: DEL, MOD, NNENT, SNAPSAV, TERM

MODULE NAME: SNAPDUMP (Dump Snap Tables)
FILE NAME: DSTAR.FOR
PROCESS: DSTAR

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

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)

CALLS TO: FMTTIM, FOR\$BITEST

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 INPUTS: DDATE, SCRIT, SNAME, SMENT, TIME

GLOBAL OUTPUTS: None

MODULE NAME: OLDSNAP (R/W Snap Tables to Disc)
FILE NAME: DSTAR.FOR
PROCESS: DSTAR

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

CALLED BY: SNPRC
CALLING SEQUENCE: CALL OLDSNAP
CALLS TO: CHECK, FOR\$CLOSE, FOR\$OPEN, SYM_SEARCH

DESCRIPTION:

If a file is to be 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 INPUTS: ADR, CBUF, CHCNT, ERROR, LGTH, LPAREN, RPAREN,
SNENT, TYP

GLOBAL OUTPUTS: SNAPDMP, SNPSAV, SRST

MODULE NAME: UCASE (Convert to Upper Case)
FILE NAME: DSTAR.FOR
PROCESS: DSTAR

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

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

CALLS TO: None

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 INPUTS: None

GLOBAL OUTPUTS: None

264

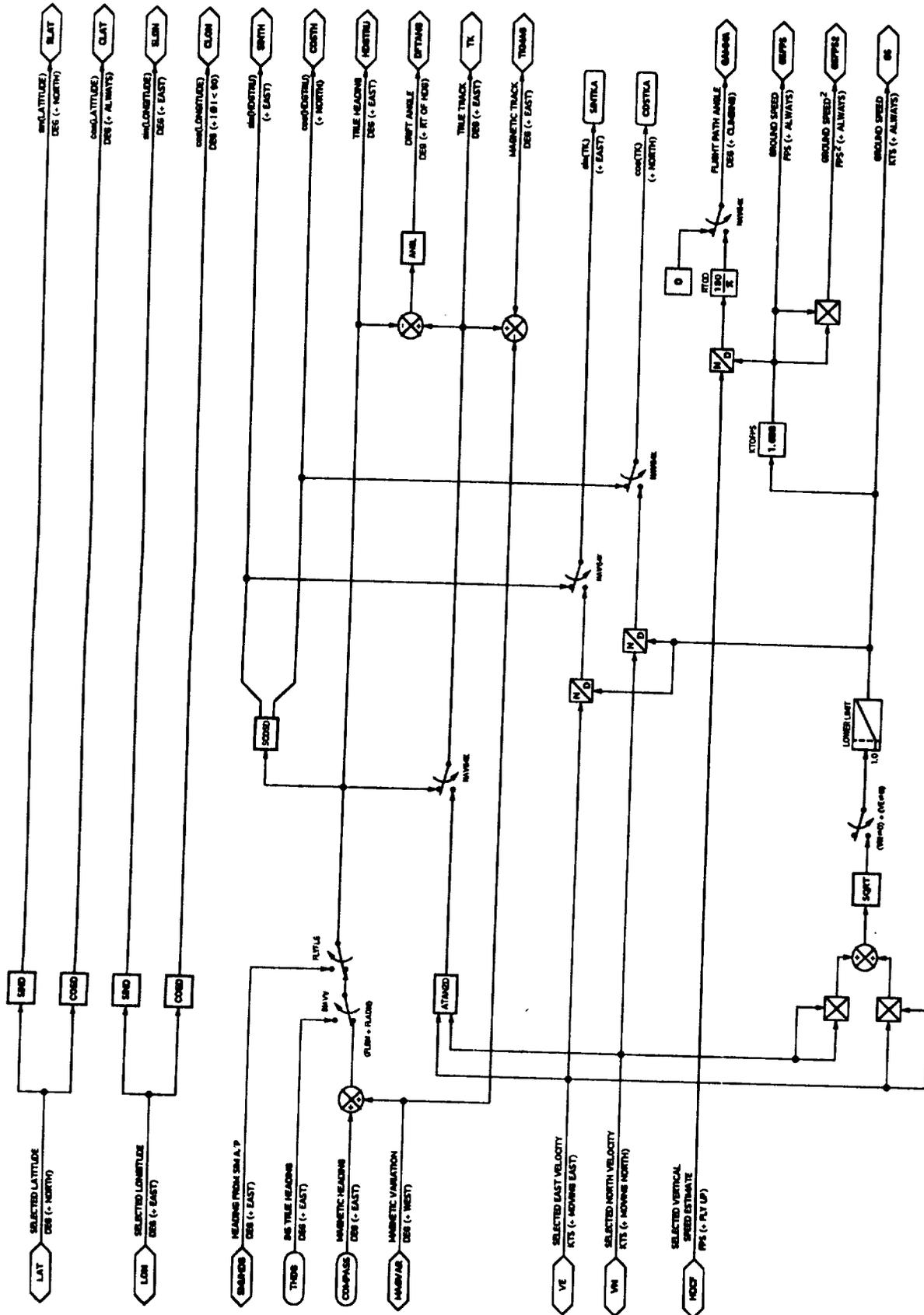
APPENDICES

Appendix A - Digital Systems Diagrams

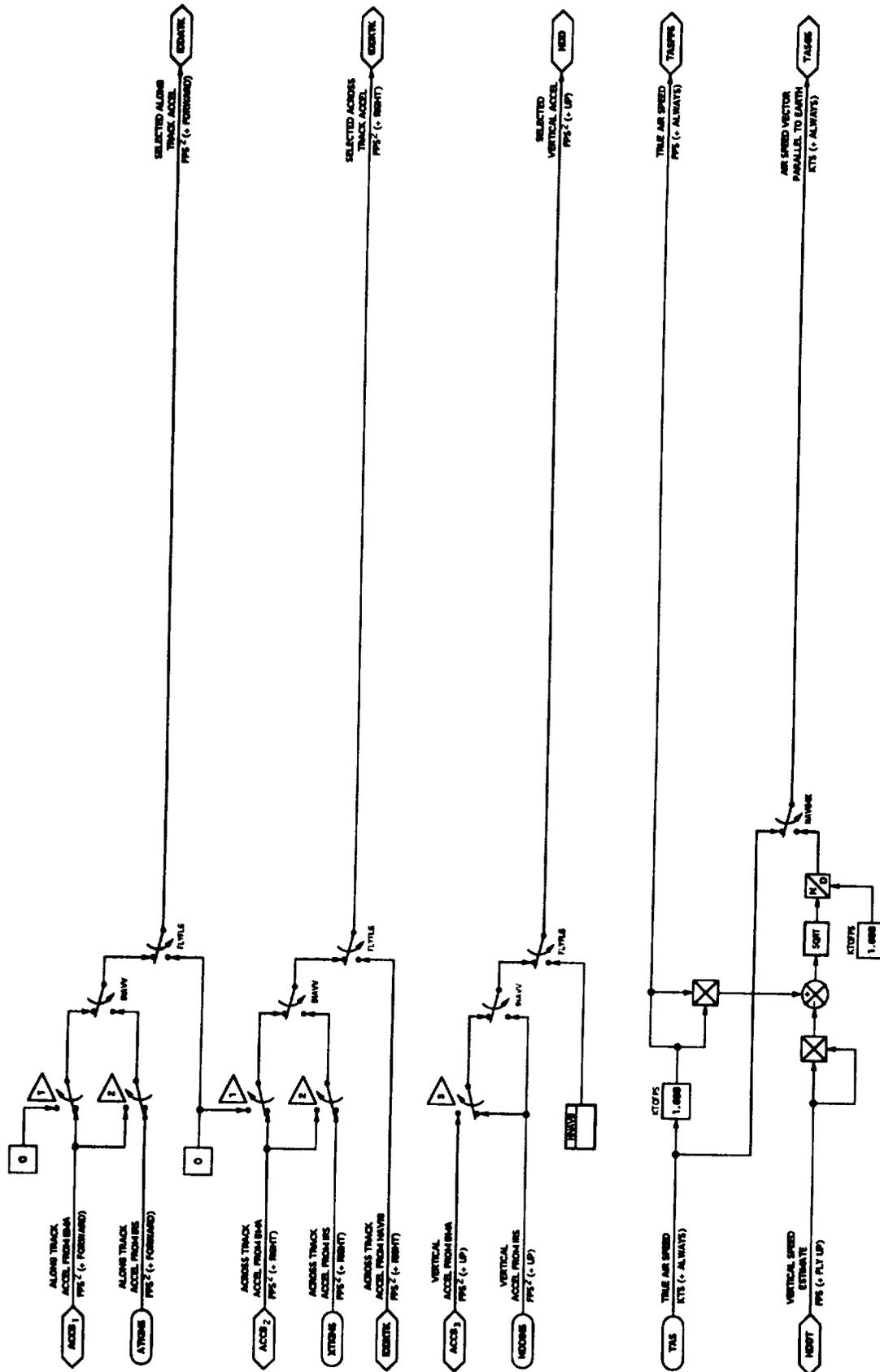
HNAVFS Procedure	266
HNAVML Procedure	270
HNAVB Procedure	272
HNSWIT Procedure	275
MLSEX Overview	277
TRALCBA Procedure (HVGUID)	278
LATCMD Procedure	280
CMPF Procedure	284
LATRL Procedure	285
FRCWS Procedure	287
RCOM Procedure	288
RBASC Procedure	289
RCOMA Procedure	290
RBASCA Procedure	291
DCRAB Procedure	292
HVG6 Procedure (HVGUID)	293
VERCMD Procedure	294
PAL Procedure	298
ELEVP Procedure	299
PFFD Procedure	301
PAFD Procedure	302
PVPC Procedure	304
VTFCL Procedure	305
STABP Procedure	306
CDG Procedure (TGUID)	307
SPDCMD Procedure	308
ATHCL Procedure	310
MLOG Procedure	311

Note: The above drawings are produced in single page format for ease of reproduction. They are available on electronic media (Mac Draw II) should you wish to make your own copy in a more readable size. To receive a copy please contact David Wolverton at the following address:

Computer Sciences Corporation
3217 N. Armistead
Hampton, VA 23666
Phone (804) 865-1725.

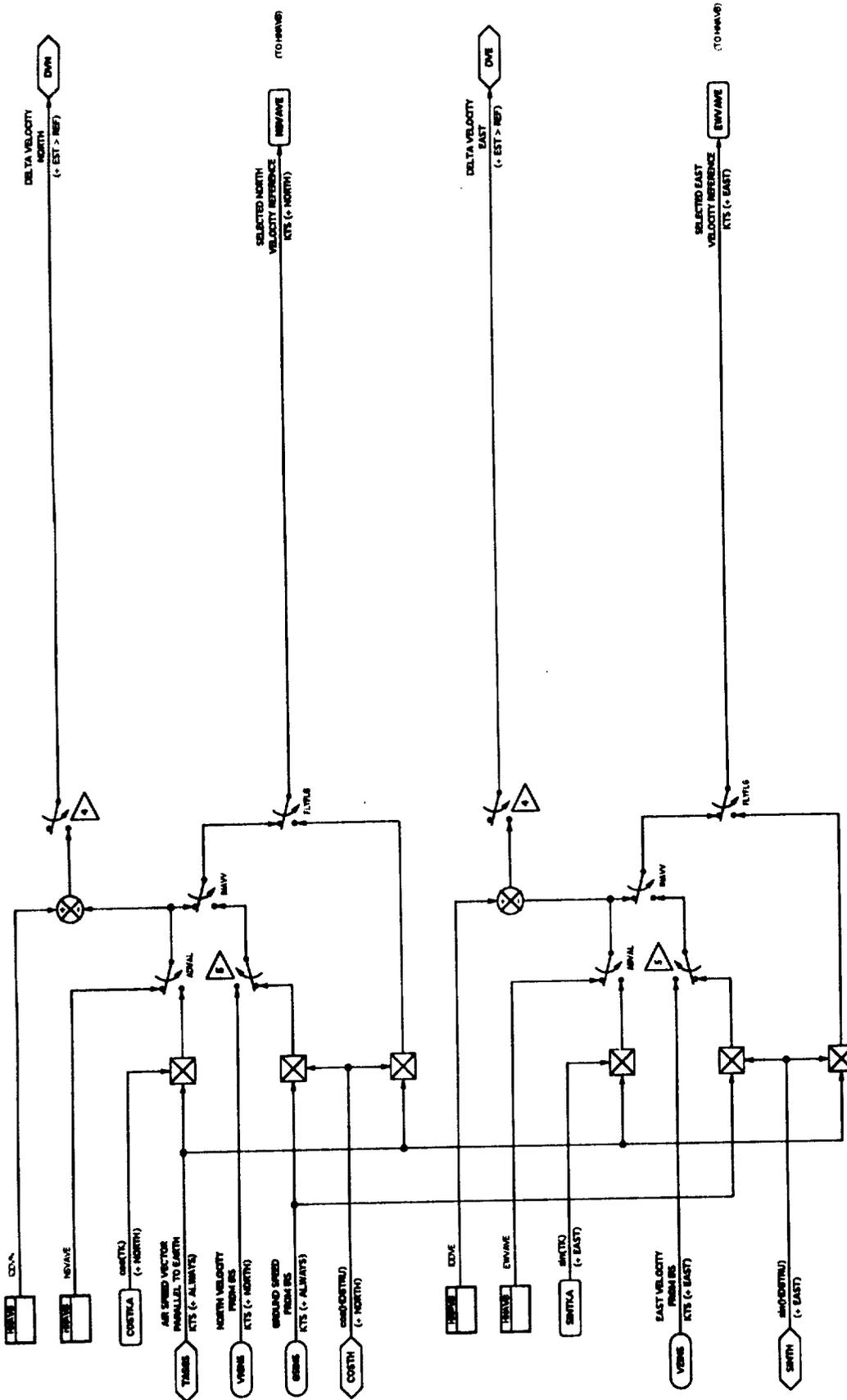


HNAVFS PROCEDURE



- 1 (KDFW - 200) = 0
- 2 (KDFW - 10) = 0 - ACCALD
- 3 (KDFW - 100) = 0

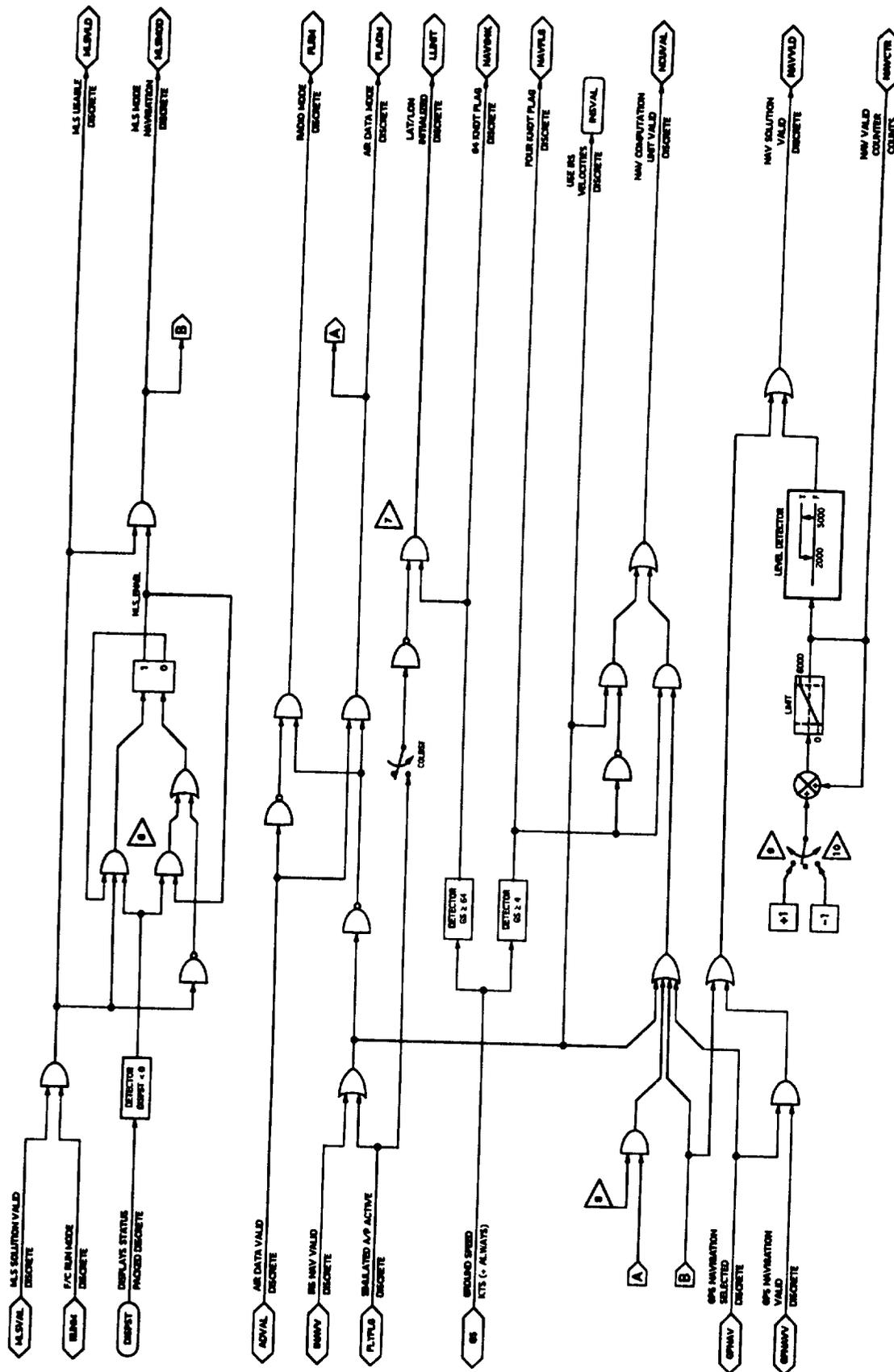
HNAVFS PROCEDURE



Δ DELTA VELOCITY (MAG ESTIMATE ± 0.01)
 Δ FIRST PASS FROM PLANA TO NORTHAL (HNAV)
 Δ SECOND PASS FROM PLANA TO NORTHAL (HNAV)
 Δ OR WHEN IS UNIT SELECTION CHANGES
 (RS SEL STATUS = PAST VALUE)

Δ VELVD + (MCOFF * V0V) = 0

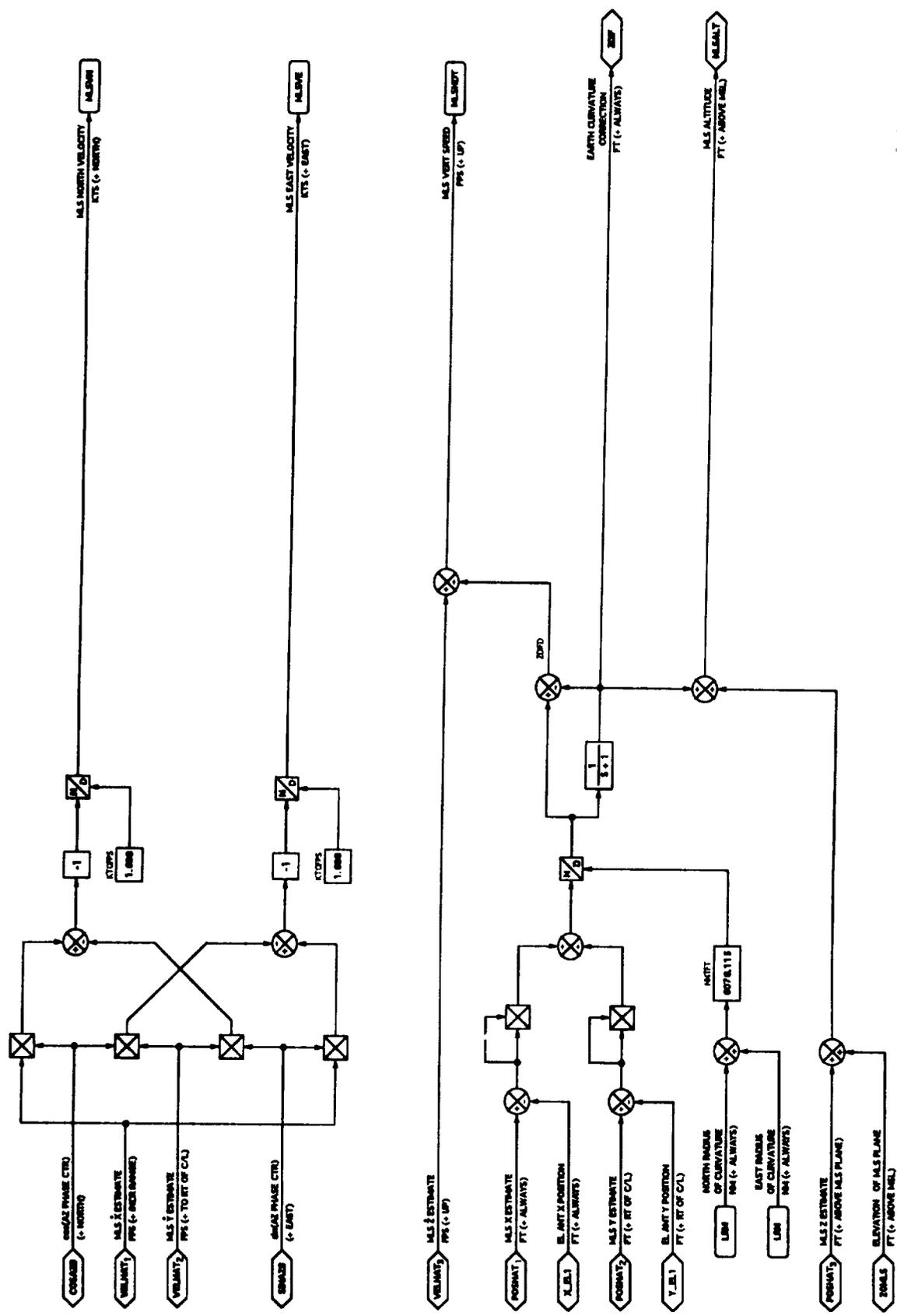
HNAVFS PROCEDURE



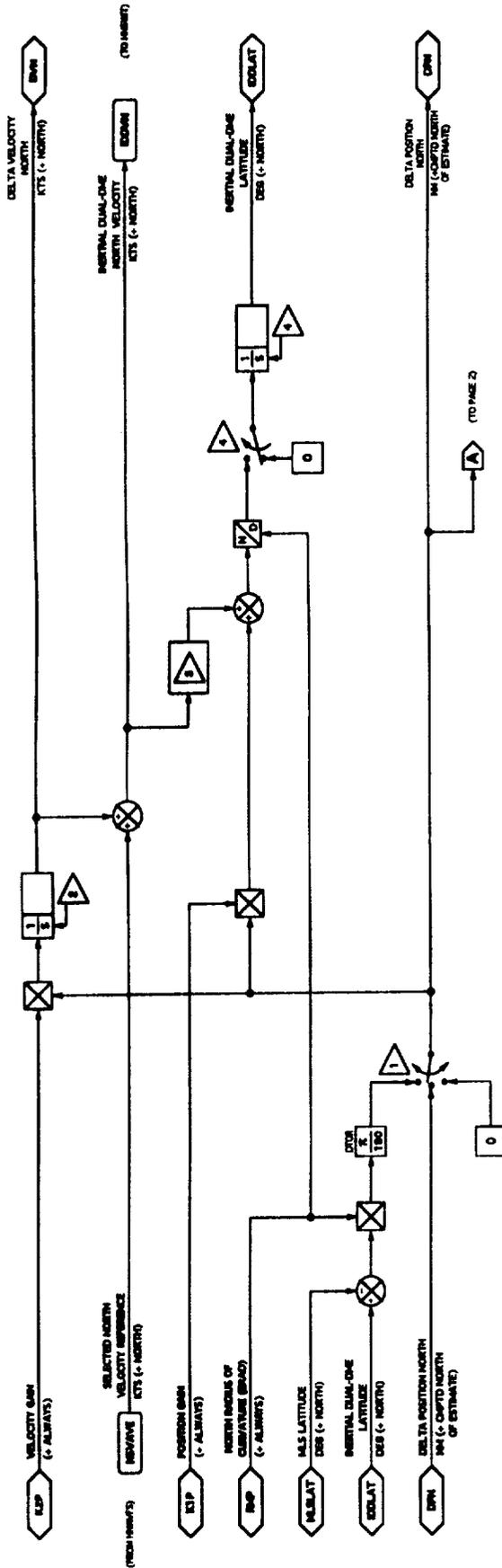
(MCDM + 20N) = 0
 0.0 + SQRT(MN² + DR²) - 0.3
 (OPN + DR) = 0

BIT BODY OF COUNT IS ONE SHIPPED BY DISPLAY PROCESSING. THIS PERMITS THE M/S SELECT SWITCH TO FUNCTION AS "PUSH COMMAND OFF".
 LIMIT IS LATCHED ONCE SET BY M/VMS OR M/TMS, AND IS CLEARED ONLY BY COUNT = 10.

HNAVFS PROCEDURE

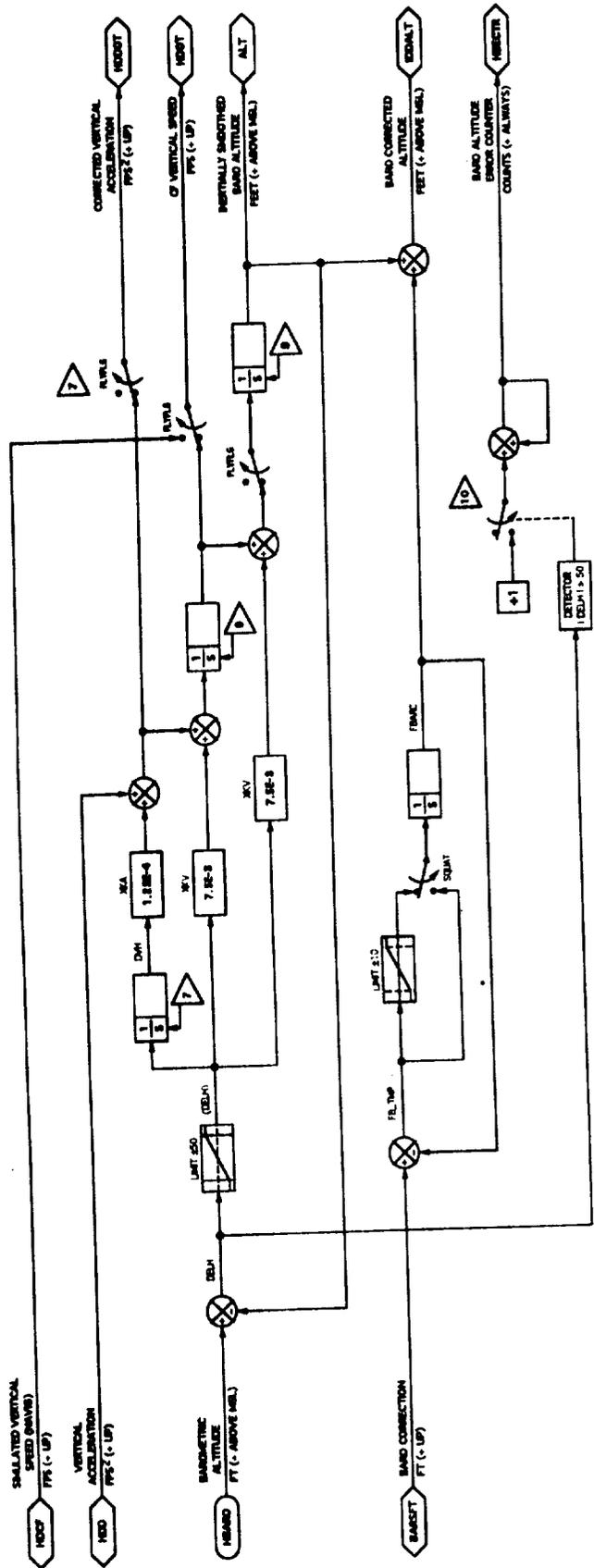


HNAVMI PROCEDURE
 SUBROUTINE OF HNAVTS
 PAGE 2 OF 2



- 1 IF PASS 1 IS TRUE, DIVIDOR ARE COMPUTED FROM DELTA VELOCITY NORTH (ALSO CALLED "DELTA V") AND DELTA POSITION NORTH (ALSO CALLED "DELTA P"). DIVIDOR ARE COMPUTED BY DIVIDING DELTA VELOCITY NORTH BY DELTA POSITION NORTH. DIVIDOR ARE SET BY MANUAL WHEN DELTA IS TRUE AND IS CLEARED BY MANUAL WHEN DELTA IS FALSE.
- 2 INTEGRATED B (GAINS) = (DELTA V) * (PASS 1) + FILTERED DELTA VELOCITY NORTH. (NOTE: A1 IS INCLUDED IN DELTA V. SEE HNAV'S PAGE 3 FOR A1 CRITERIA.)
- 3 DT5 = 180.81 / 3600.0 = 7.5277 E-4.
- 4 INTEGRATED B (GAINS) = (DELTA V) * (PASS 1) + FILTERED DELTA VELOCITY NORTH. (NOTE: A1 IS INCLUDED IN DELTA V. SEE HNAV'S PAGE 3 FOR A1 CRITERIA.)

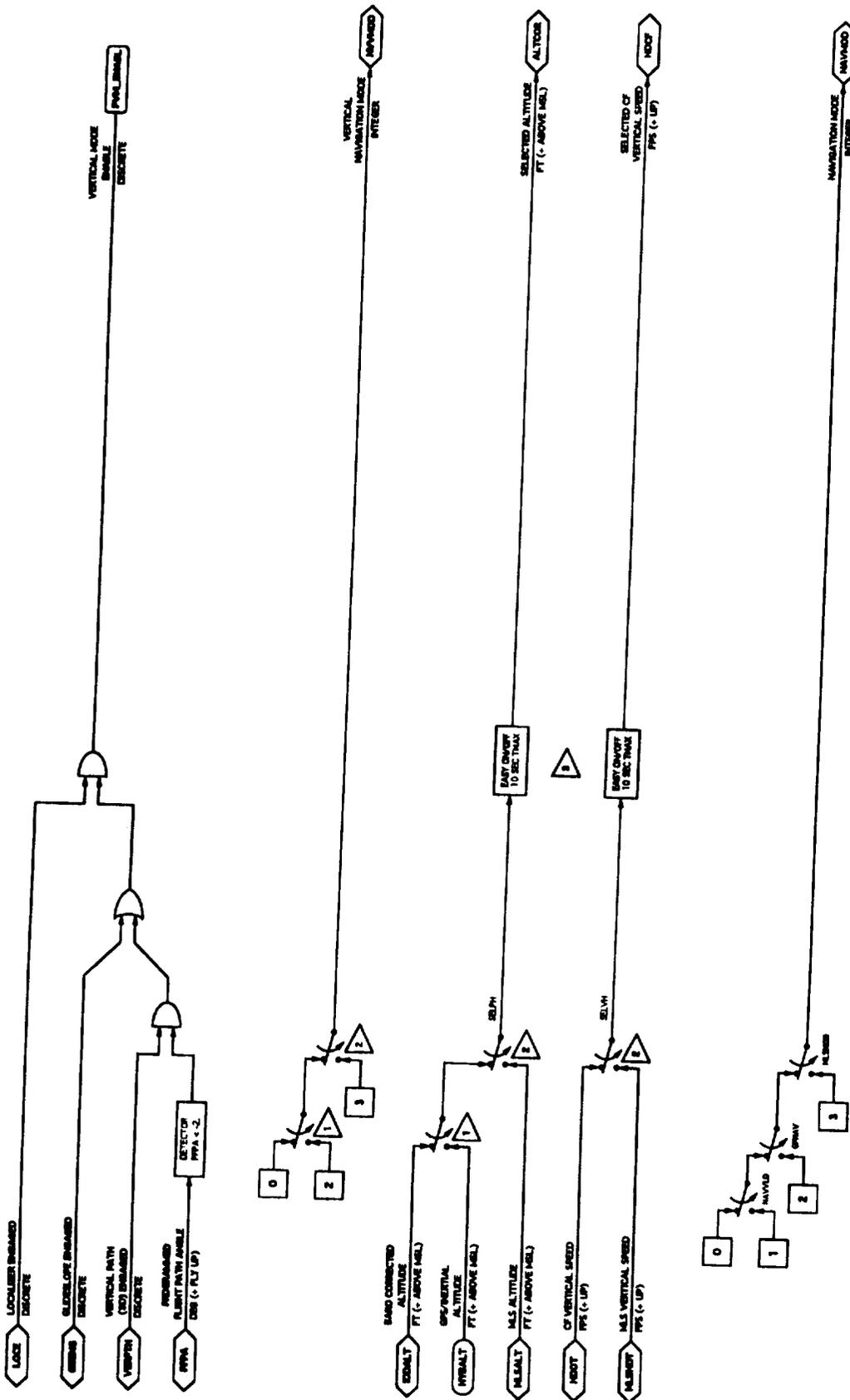
HNAV PROCEDURE
 SUBROUTINE OF HNAV'S
 PAGE 1 OF 3



- 7 INTEGRATE \odot (INAVLG + INLNOO + PASS1) \cdot F(VLG)
 B.S. L = 0.
 F (VLG) HOBT IS COMPUTED BY HAVG.
- 9 INTEGRATE \odot (HAWLS + INLNOO + PASS1) \cdot F(VLS)
 B.S. IF (VLVLO) HOBT IS IC TO HOBT.
 B.S. HOBT IS IC TO HOBTB.
- 9 INTEGRATE \odot (HAWLS + INLNOO + PASS1) \cdot F(VLS)
 B.S. IF (VLVLO) ALT IS COMPUTED BY HAVG.
 B.S. ALT IS IC TO HBAWT.
- 10 A LARGE DELTA IS INDICATOR OF HAVG ERROR.
 A COUNT OF THESE EXCEEDED IS MAINTAINED FOR HBSCTR.

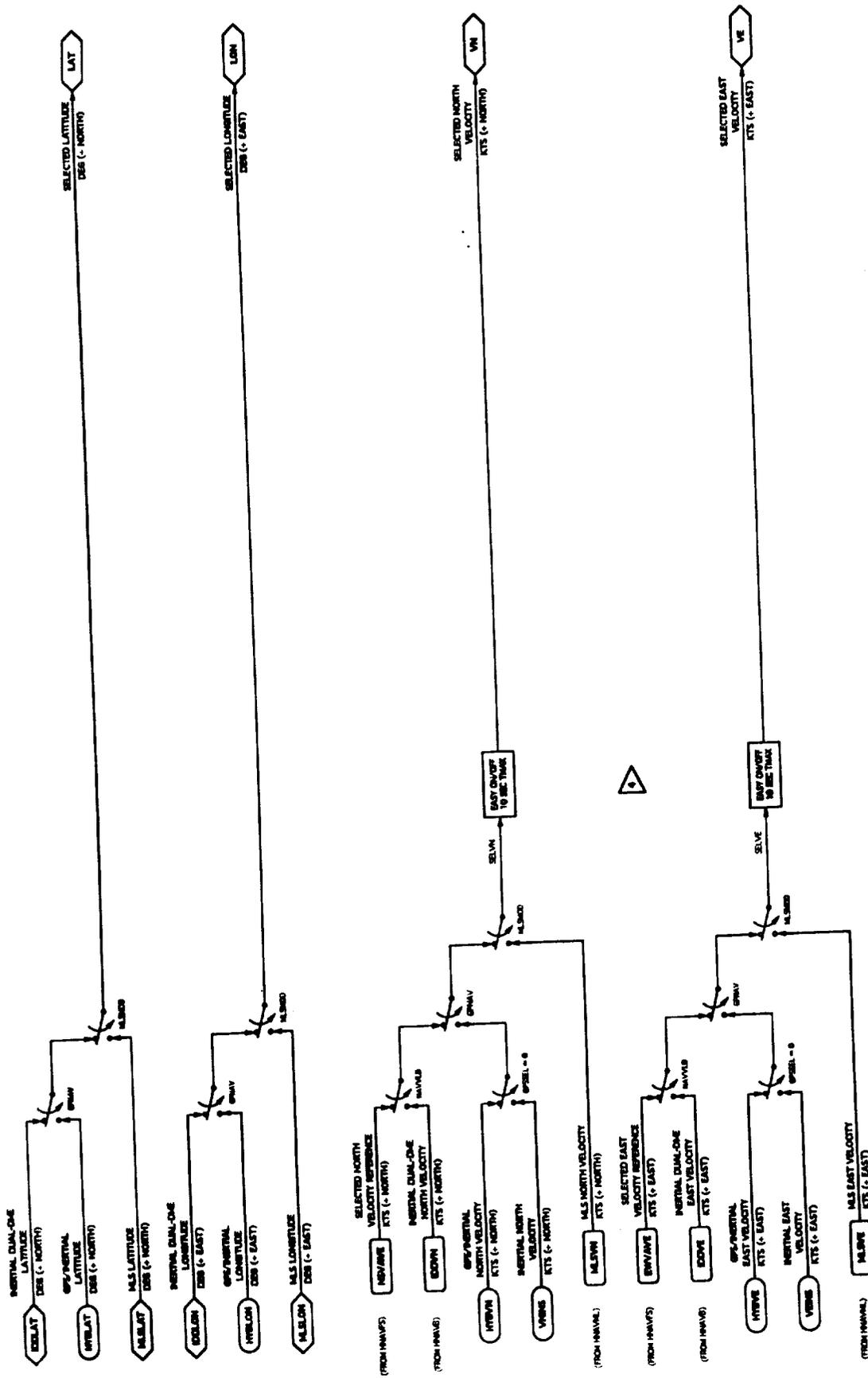
HNAVb PROCEDURE

SUBROUTINE OF HNAVFS



- 1 0PNVY • 0PNVY • PNL ENBA
- 2 MLSND • MLSN(4) • PNL ENBA
WHERE : MLSN(4) INDICATES THE PRESSION MODE.
- 3 EAST-ON INITIATED WBSHWARD OWNES STATE.

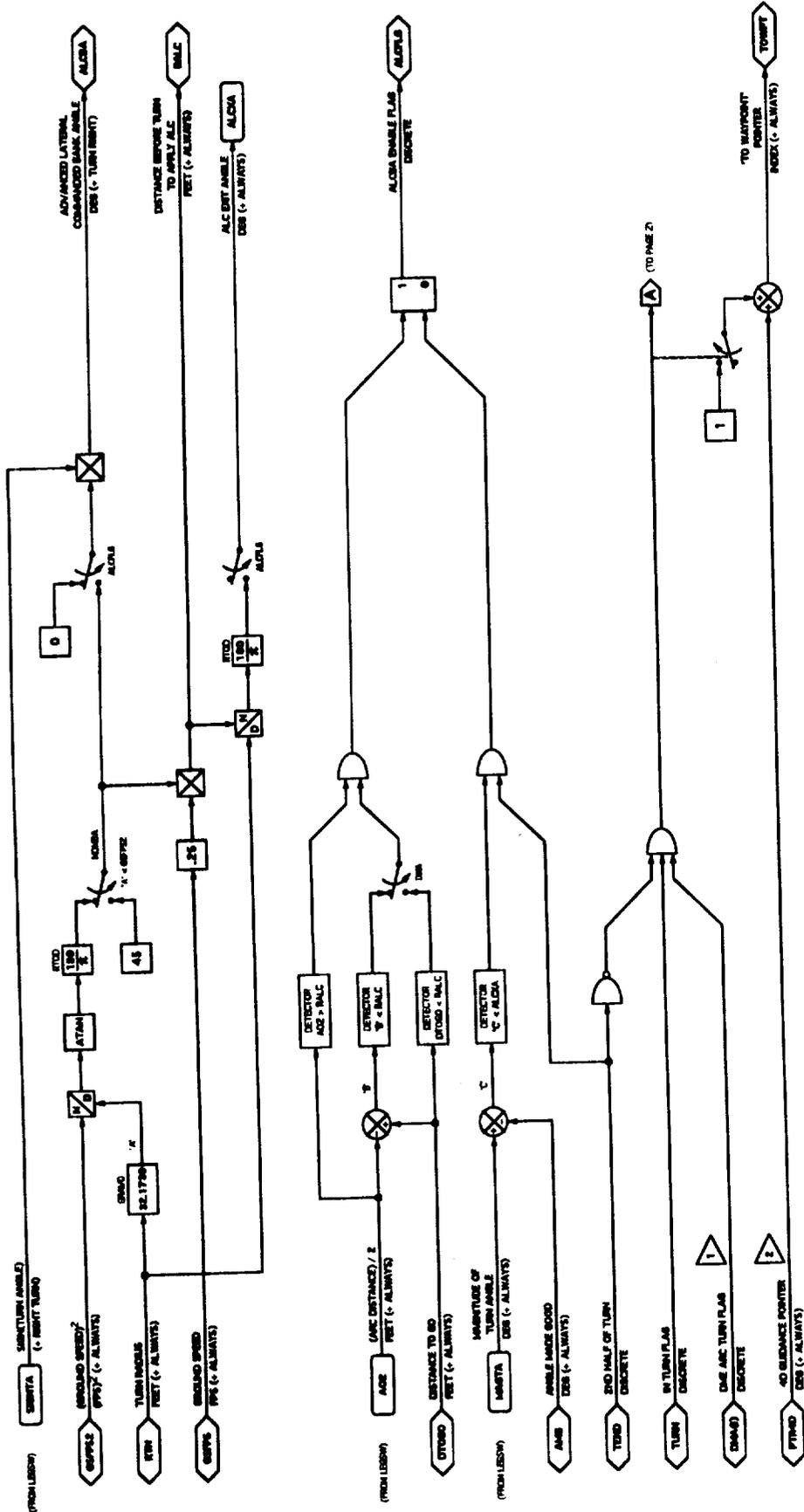
HNSWIT PROCEDURE
 SUBROUTINE OF HNAVPS
 PAGE 1 OF 2



HNSWIT PROCEDURE

SUBROUTINE OF HNAVPS
PAGE 2 OF 2

⚠️ EARLY-CONTINUED UNRECORDED
(SEE PAGE 1) CHANGED STATE.



TRALCBA PROCEDURE

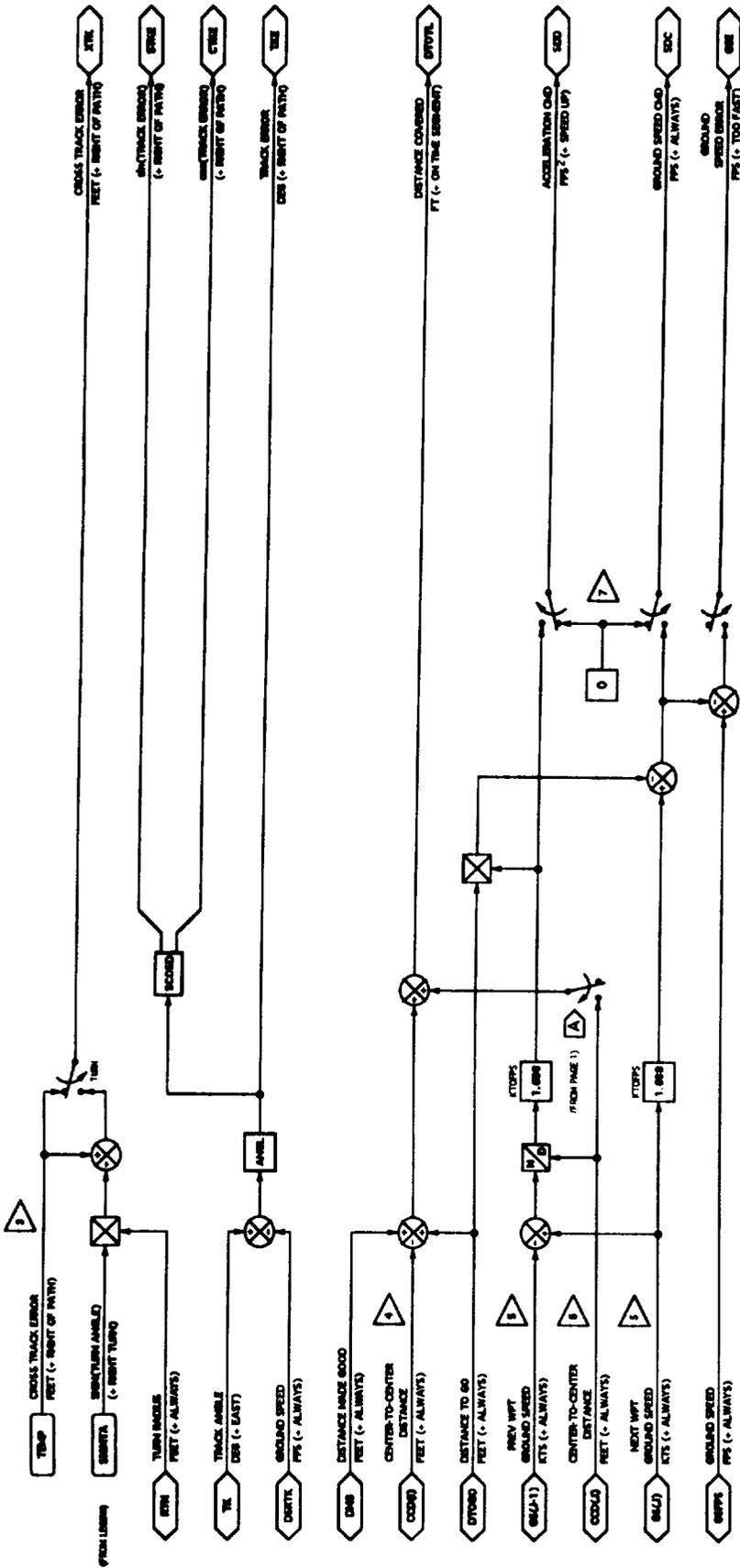
SUBROUTINE OF HYBRID

PAGE 1 OF 2

GLANCE MESSAGES (WT_ACT.DMA).
 WISE: 1 = PRIO.

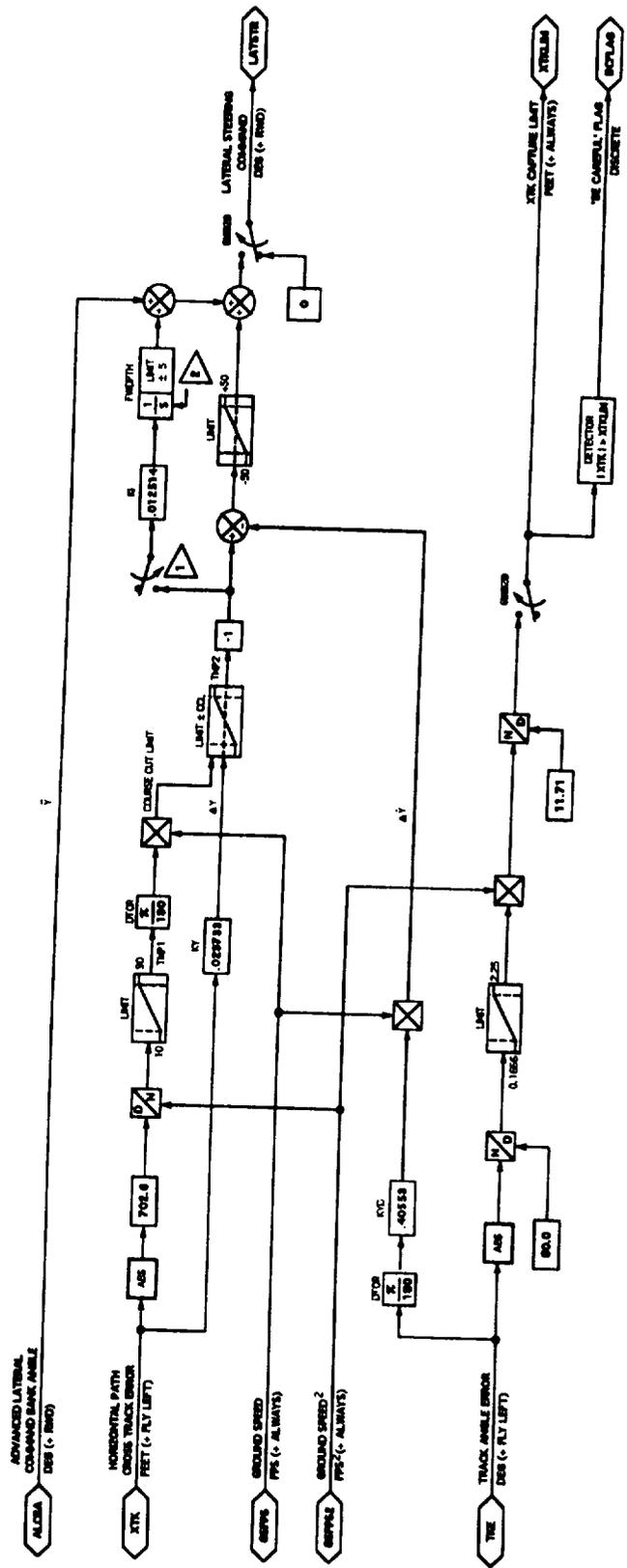
PRIO IS INITIALIZED AND UPDATED IN HFIELD HANDLE
 CODE. MESSAGE VALUES 2.





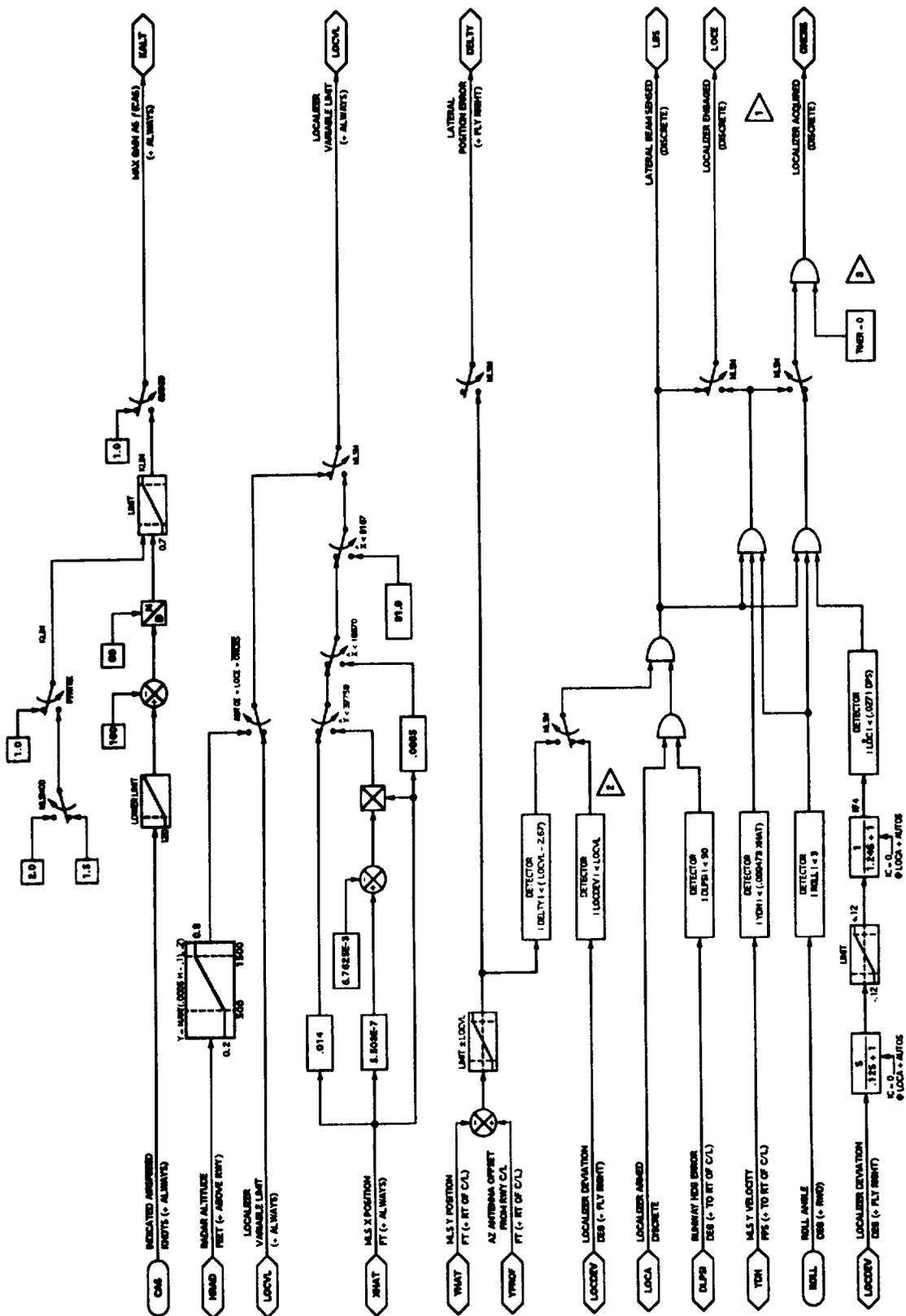
- △ WHEN IN A TURN, XTRK IS FIRST CALCULATED TO THE CENTER OF THE TURN, THEN CORRECTED BY THE TURN INCLINE.
- △ GROUND SPEED ENTRY WPT_ACT(COOR), WHERE : : = PTHNO.
- △ GROUND SPEED ENTRY WPT_ACT(SRD), WHERE : : = TURNPT.
- △ GROUND SPEED ENTRY WPT_ACT(COOR), WHERE : : = TURNPT.
- △ OUTPUT DISABLED IF (WPT_ACT(TURNPT).SRD = 0) + (TURNPT = 0)

TRALCBA PROCEDURE
 SUBROUTINE OF INVOLUD
 PAGE 2 OF 2



- 1 INTEGRATE @ AUTOE = YESE + 0 XTR (15 1000) + 0 TIE (155)
- 2 K - 0 @ INTEGRATE
- 3 EQUATIONS ON THIS PAGE ENCLOSED WITH OUTLAF 6 TIE.

LATCMD PROCEDURE



1 LIM, LOCK, AND GATES ARE INITIATED ONCE SET.
 2 THE LOCAL, CALCULATED BY PROCEDURE ALSO IS USED TO SET LOCK,
 WITH THE LOCAL CALCULATION USED TO SET GATES.

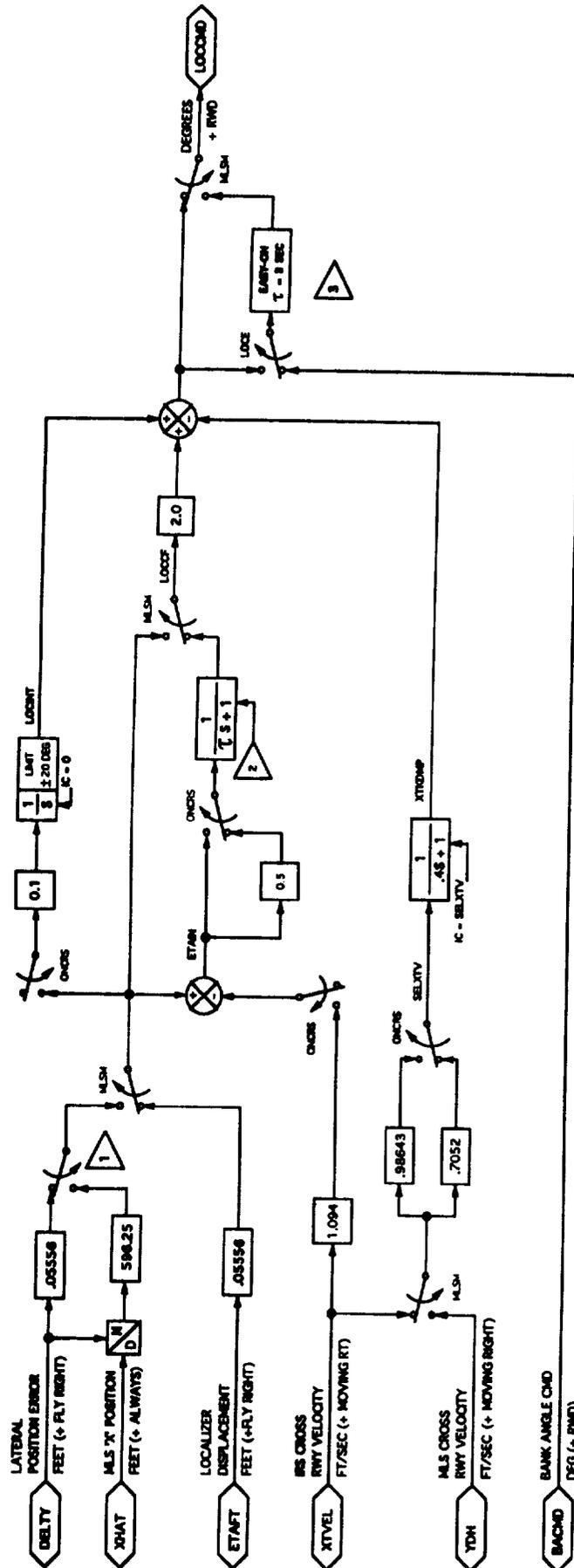
3 IN MODE A, 10 SECOND TIMER IS INITIATED WHEN LOCK IS SET.
 IN MODE A, 3 SECOND TIMER IS INITIATED WHEN THE CONDITIONS
 FOR GATES ARE FULFILLED.

1

2

3

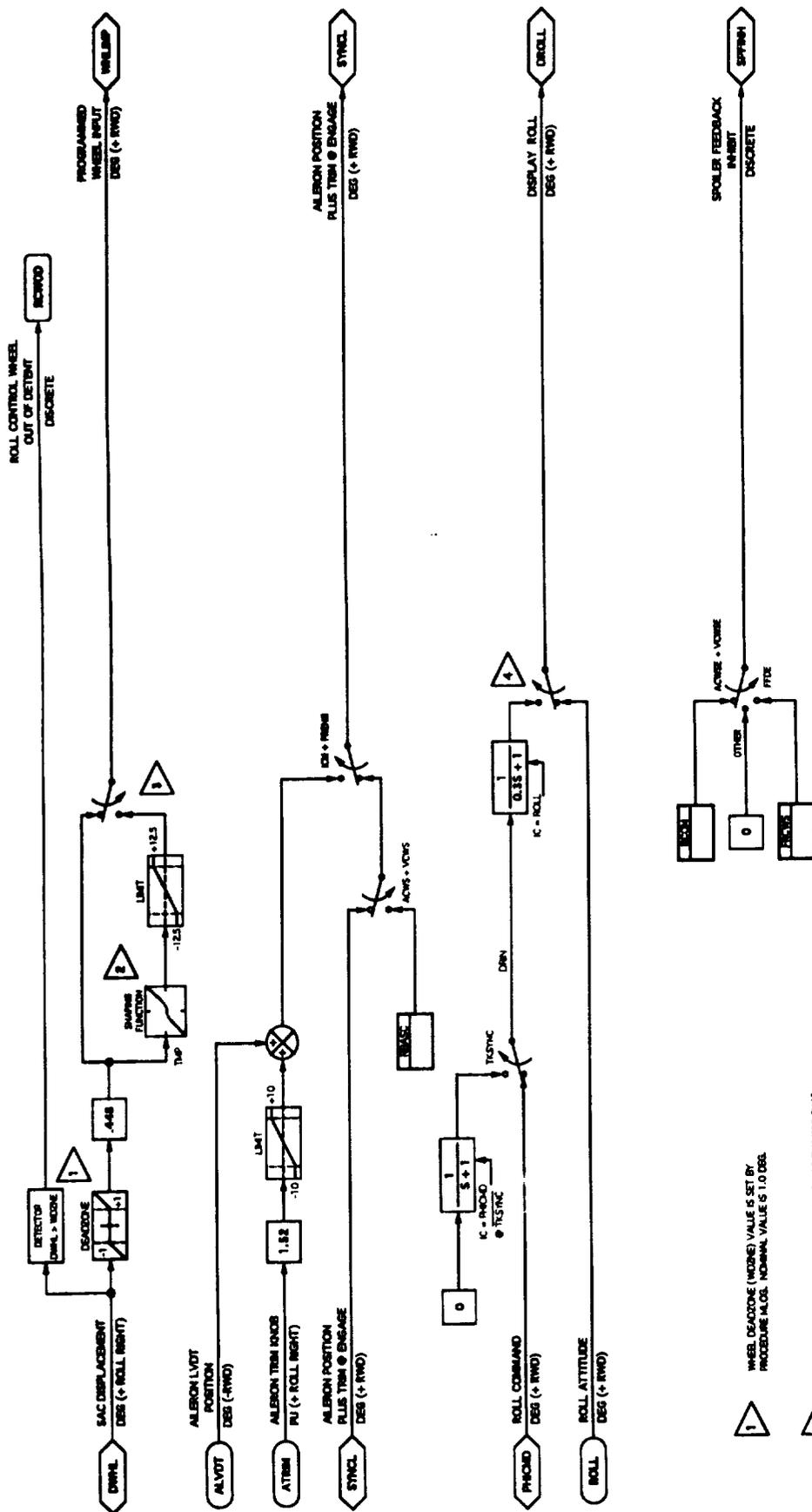
LATCMD PROCEDURE



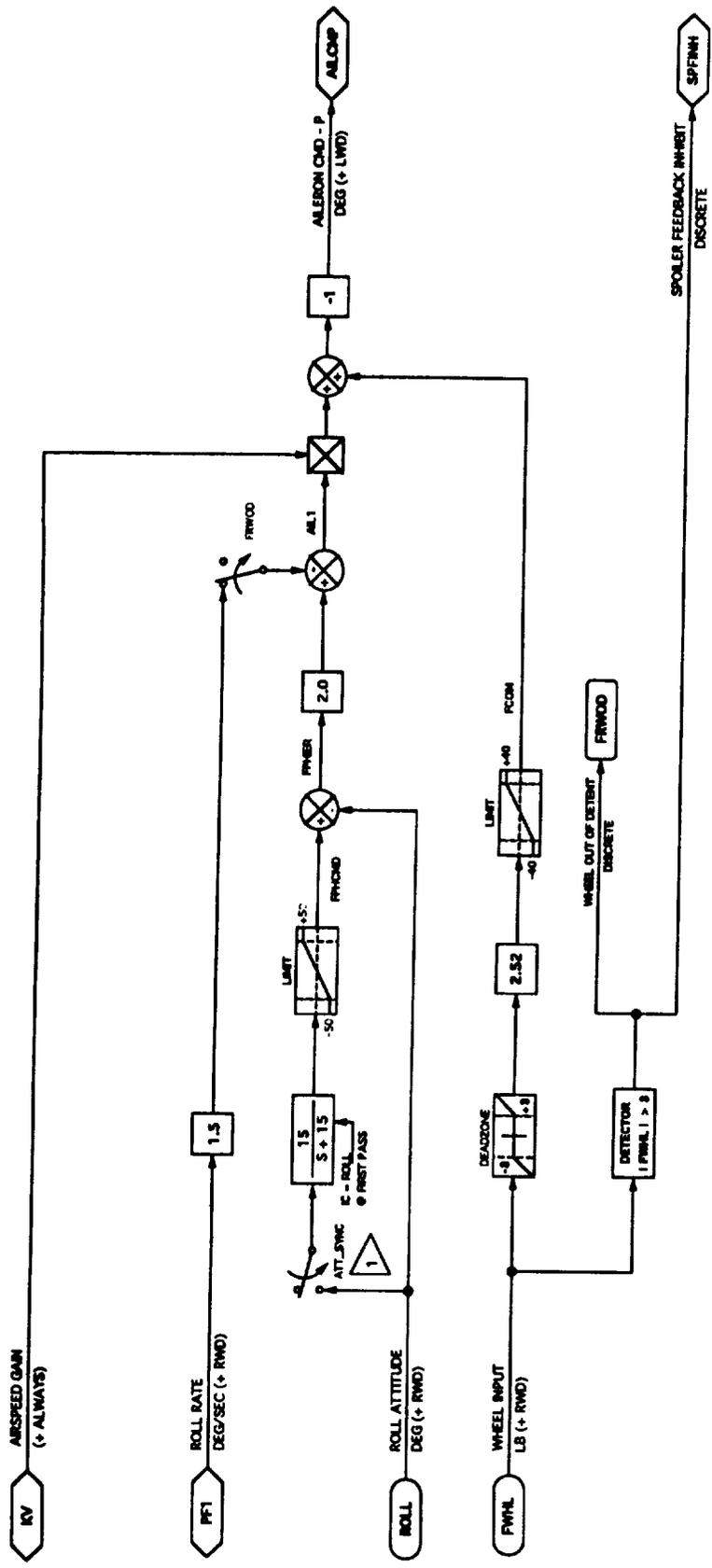
CMPE PROCEDURE

SUBROUTINE OF LATCHMD

- 1 CLOSE @ SQUAT • (HAT - 10743)
- 2 IC - (.7052 XTVEL • $\frac{d}{dt}$) / 2 @ LOCE
T - 2.0 @ LOCE SWITCHING TO 20. @ ONCRS
- 3 IN MLS MODE ONCRS IS SET WHEN THE EARLY ON EXPIRES @ 10 SEC



LATRL PROCEDURE

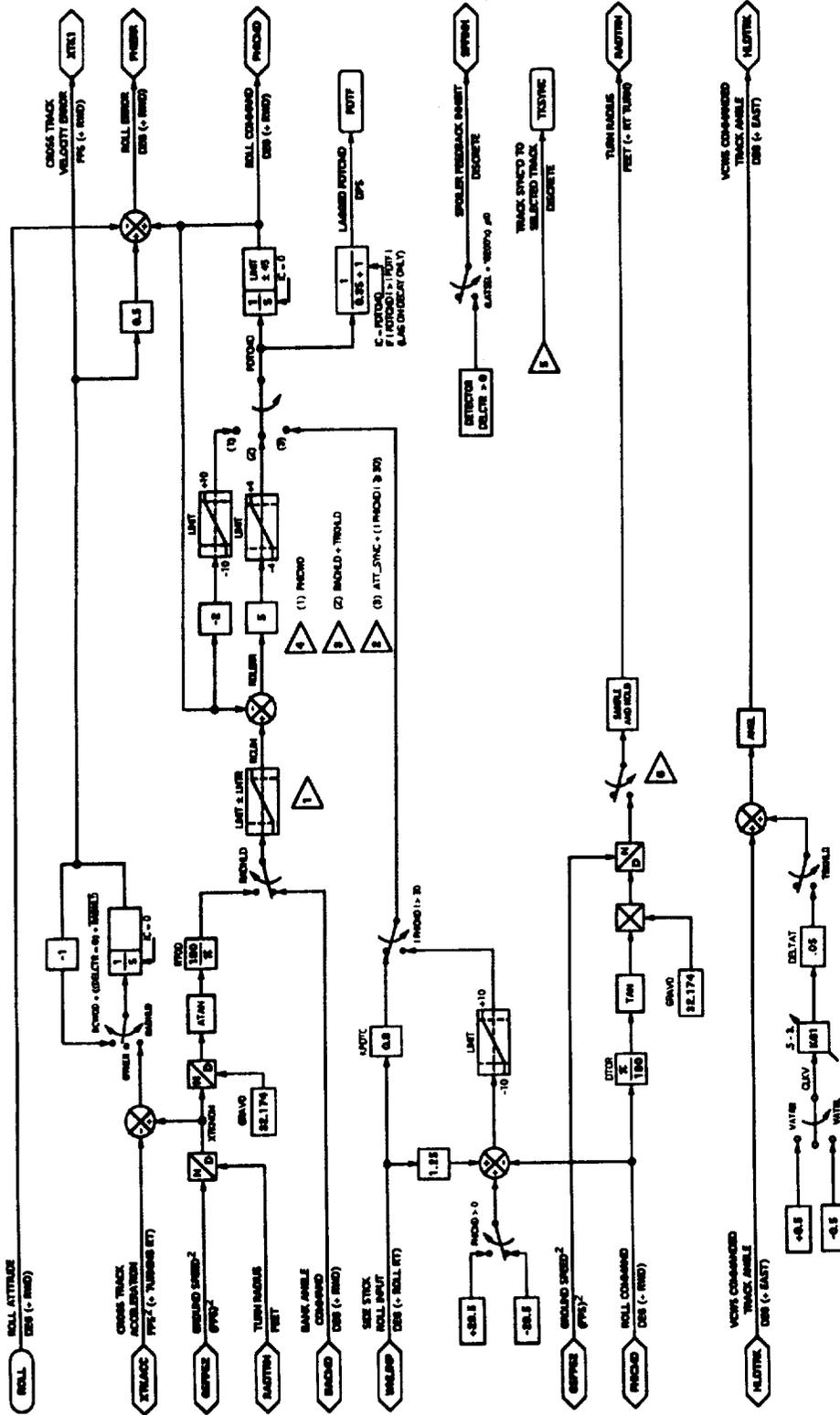


FRCWS PROCEDURE

SUBROUTINE OF LATRL

ATT_SYNC SET @ FRWOD,
CLEARED @ 0 (FRWERT1 < .05)

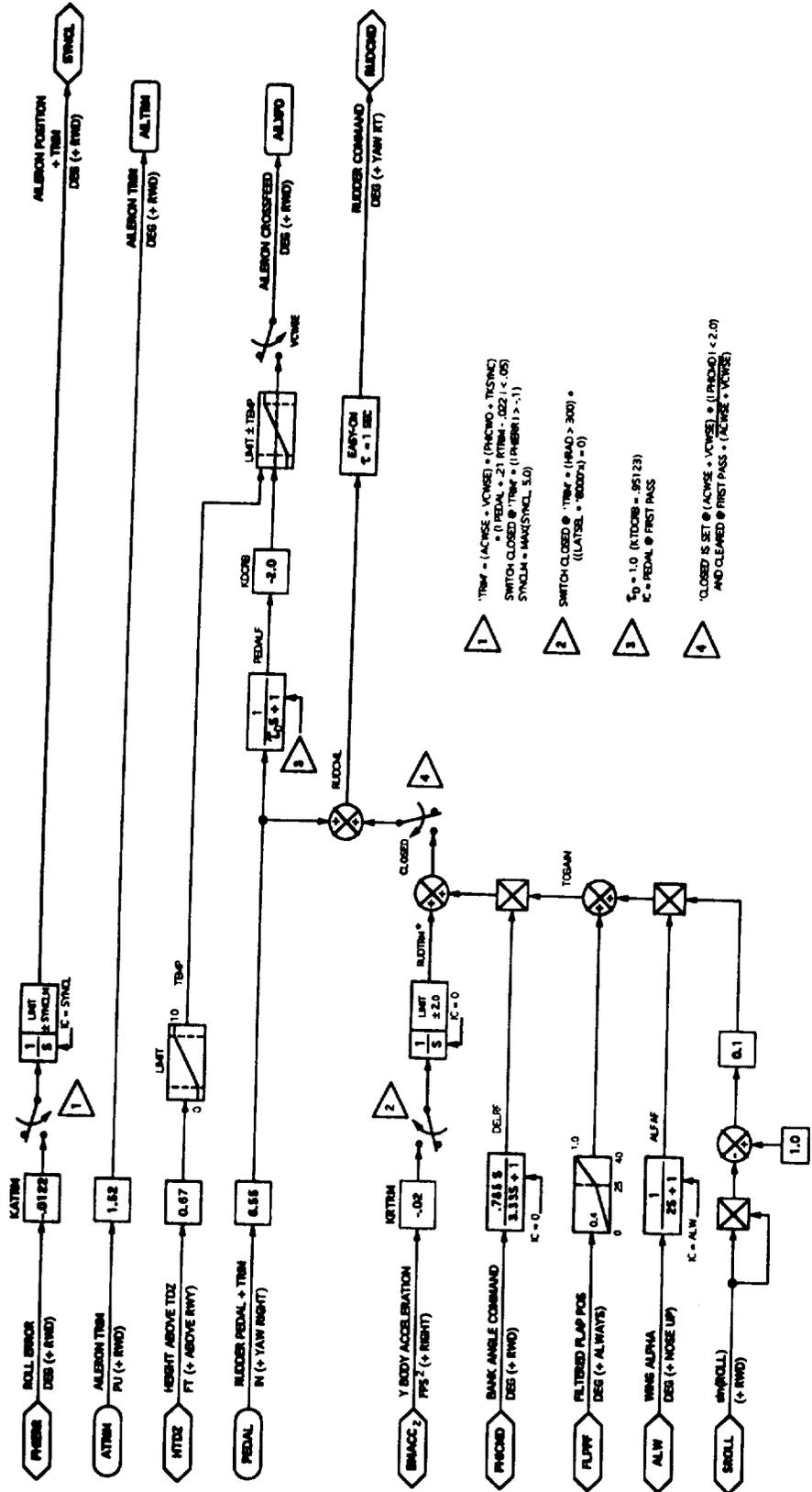




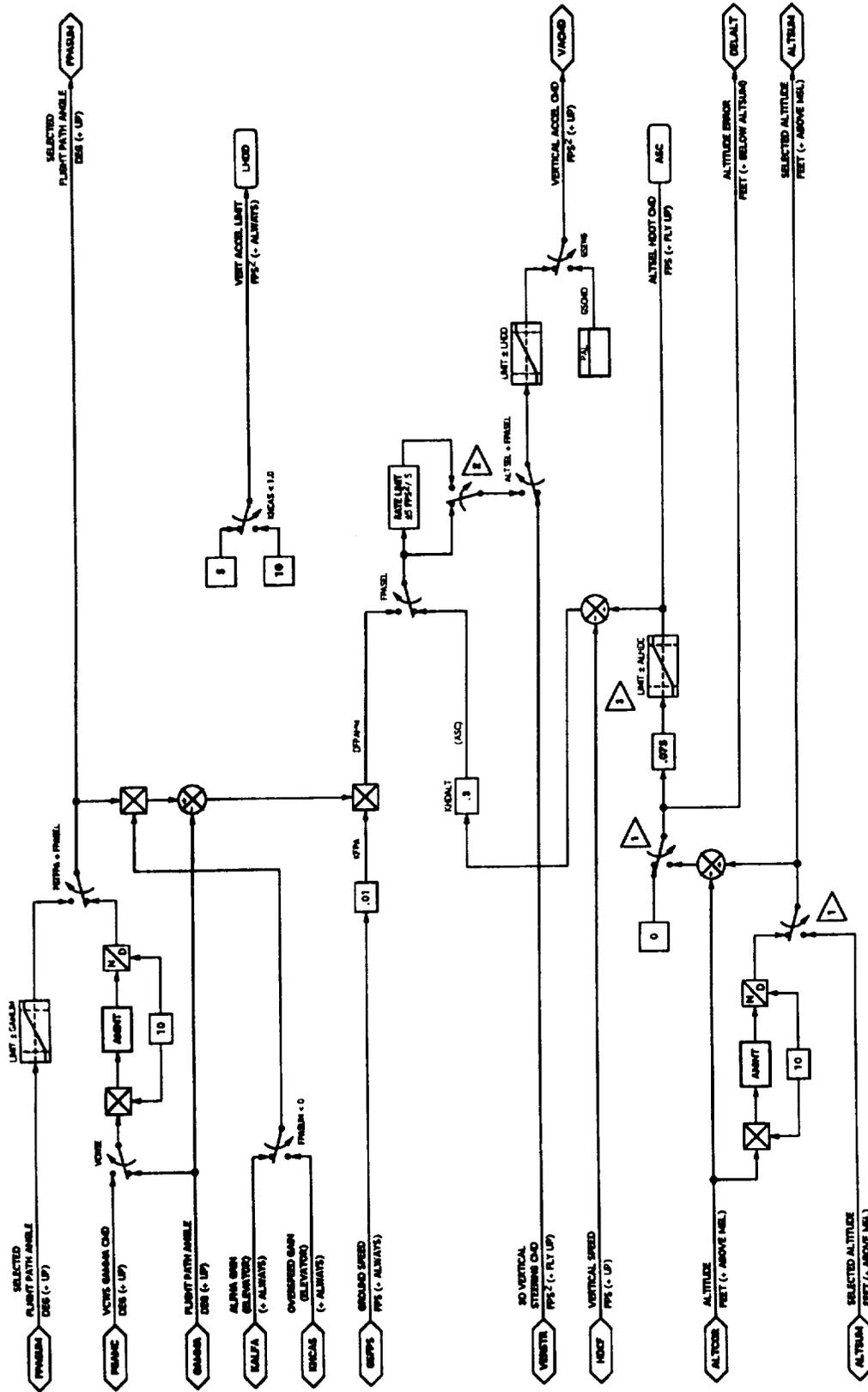
RCOM PROCEDURE

SUBROUTINE OF LATFL

- 1 ▷ IF THROTL, LIMIT, -NEGATIVES / 256.027, 30
ELSE
LIMIT = 30
- 2 ▷ ATL_SYNC IS SET @ ROLLO AND CLEARED
WHEN COLCTR IS DECREMENTED TO ZERO.
ROLDTR IS SET @ ROLLO AND CLEARED @ ROLLO.
TO 2 IF COLCTR > 0 AND COLCTR < 0.
TO 5 - LAST IF (WATER + VALTR) = THROTL,
OR TO (2 + 5 - FIRST) IF ROLLO.
- 3 ▷ ROLLO IS SET @ VCMES = ATT_HOLD = (THROTL * THROTL).
THROTL IS SET @ VCMES = COLCTR = 0 - (1/8 * 1/8 * 0.25 DEG)
AND CLEARED @ ROLLO.
- 4 ▷ PRCMD = (ACTIVE * VCMES * (ROLD - 1/8) * 1.0) /
(COLCTR = 0 - (1/8 * 1/8 * 1.0))
IF VCMES, PRCMD TRANSITIONS TO THROTL (SEE NOTE 1)
- 5 ▷ TRKVE IS SET @ THROTL = (THROTL * 1.5) * (COLCTR = 0
AND CLEARED @ FIRST PLUS = COLCTR * (COLCTR * PRCMD * PRCMD * 0)
- 6 ▷ WATERS COMPUTED @ VCMES = (COLCTR = 0 *
(PRCMD * THROTL * PRCMD)
- 7 ▷ C = 0.5 @ COLV = 0. IMPROV TO A MAX
OF 3.0 @ (WATER + VALTR) * 5 SECONDS



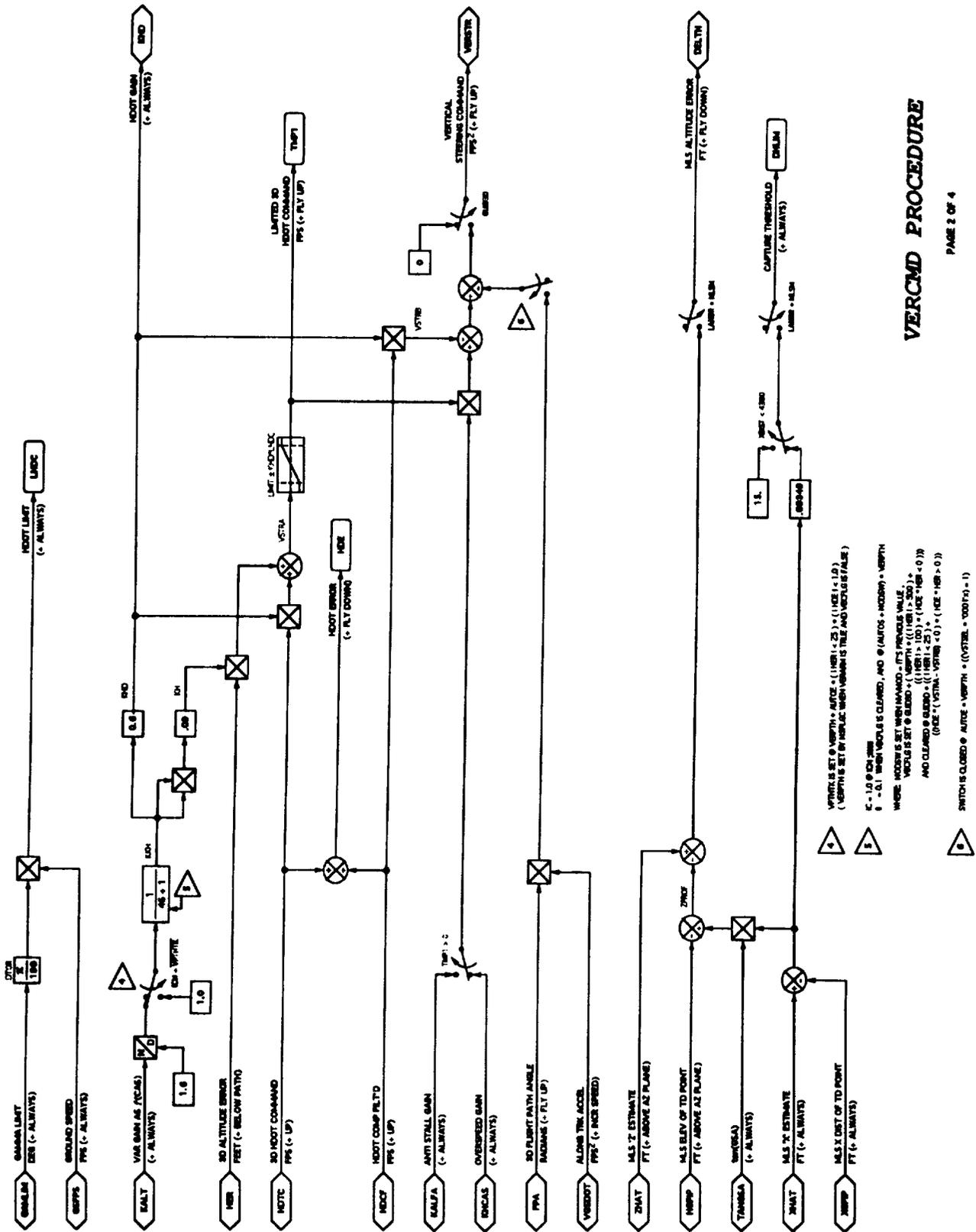
RBASC PROCEDURE
 SUBROUTINE OF LATRL



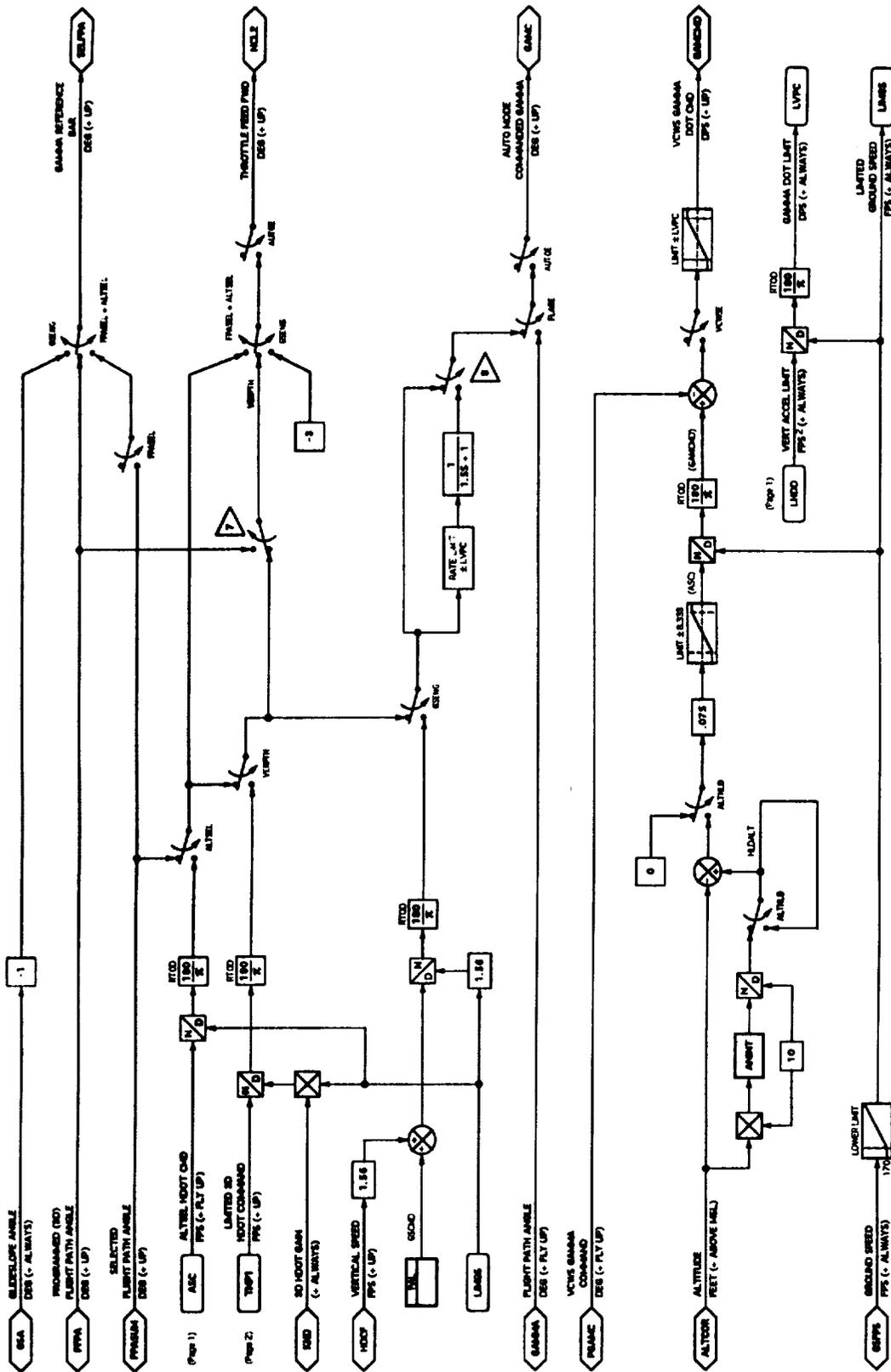
VERCMD PROCEDURE

PAGE 1 OF 4

- 1 ALTSEL + ALTARM + PSTALT
- 2 (ALTSEL + FRSEL) + (ALTIN + 0), WHERE ALTIN IS SET TO 30 (1.5 SEC) BY HOLDING WHEN THE PPA OR ALT. HOLD IS TURNED, OR ALTITUDE OR HEIGHT IS SELECTED AND THE RESULT IS ALTARM OR VERARM.
- 3 THE CONTROL LOGIC IN ALT THOLD IS SET WHEN ALTSEL IS TRUE AND INDC1 = 6.394 AND CLEARED WHEN ALTSEL IS FALSE. ALTDC IS SET TO 1001 WHEN ALTSEL IS FALSE TO 0.983 WHEN A THOLD IS TRUE, AND TO MIN(10001 + 0.5, ALTDC) OTHERWISE.

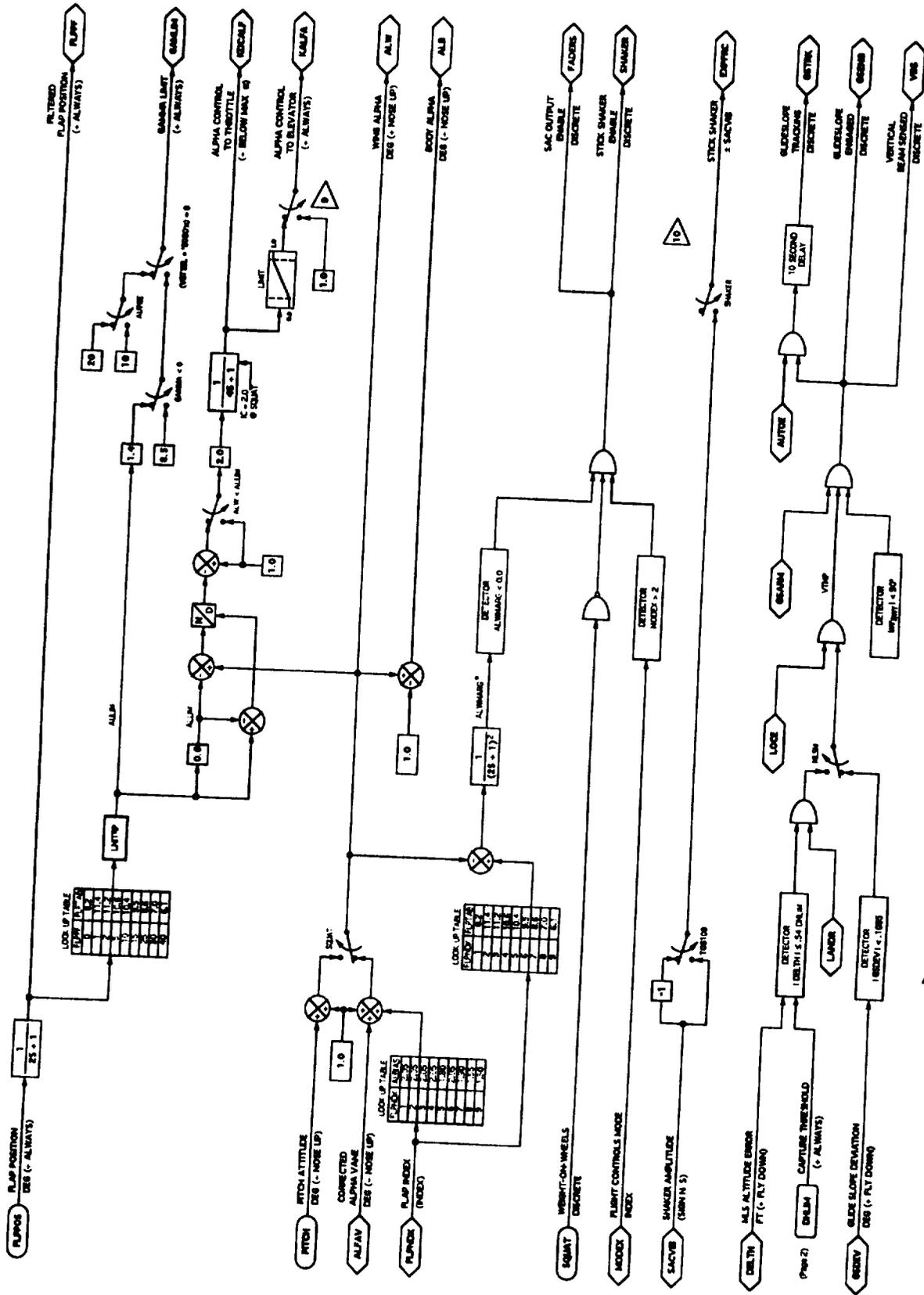


VERCMD PROCEDURE



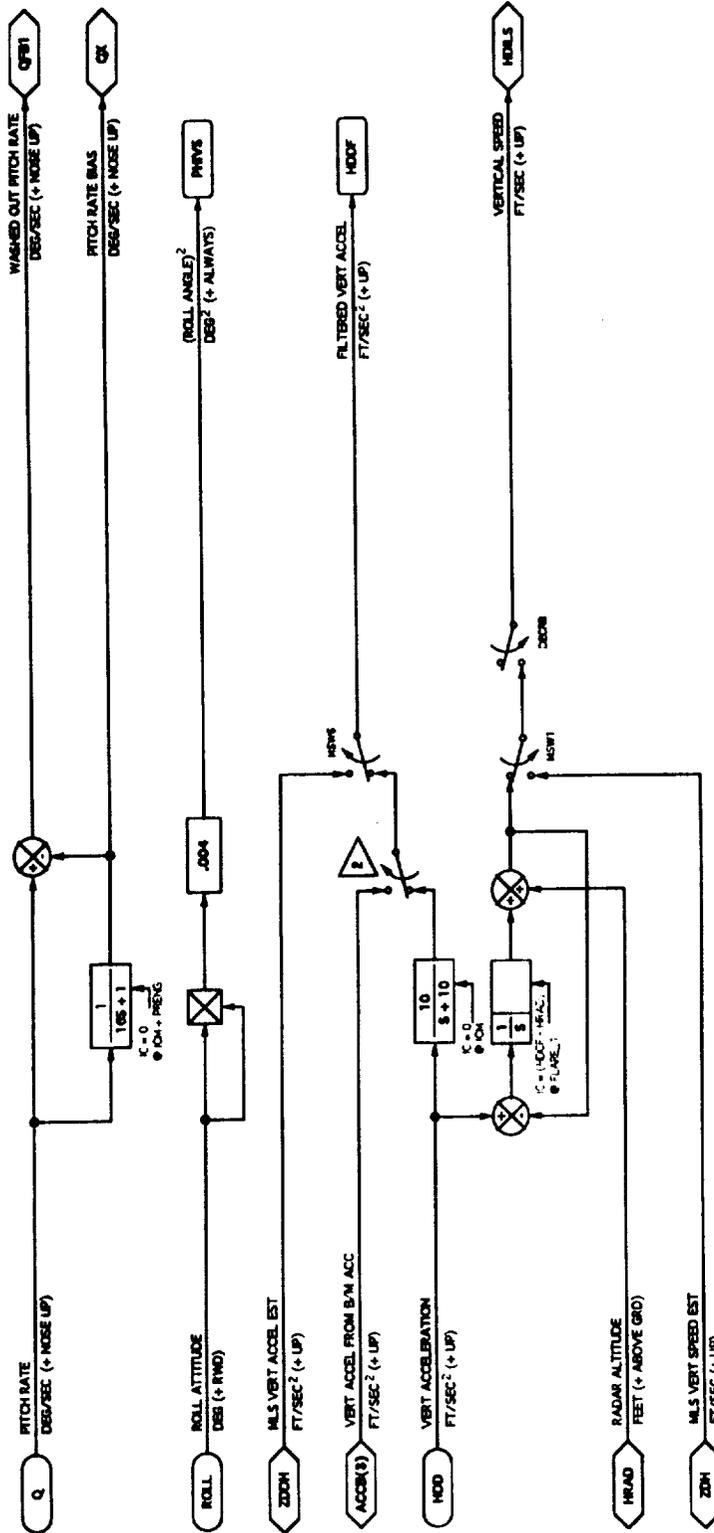
7 ((VSTRL * 1000) = 0)
 B ((VSTRL * 1000) = 0)

VERCMD PROCEDURE



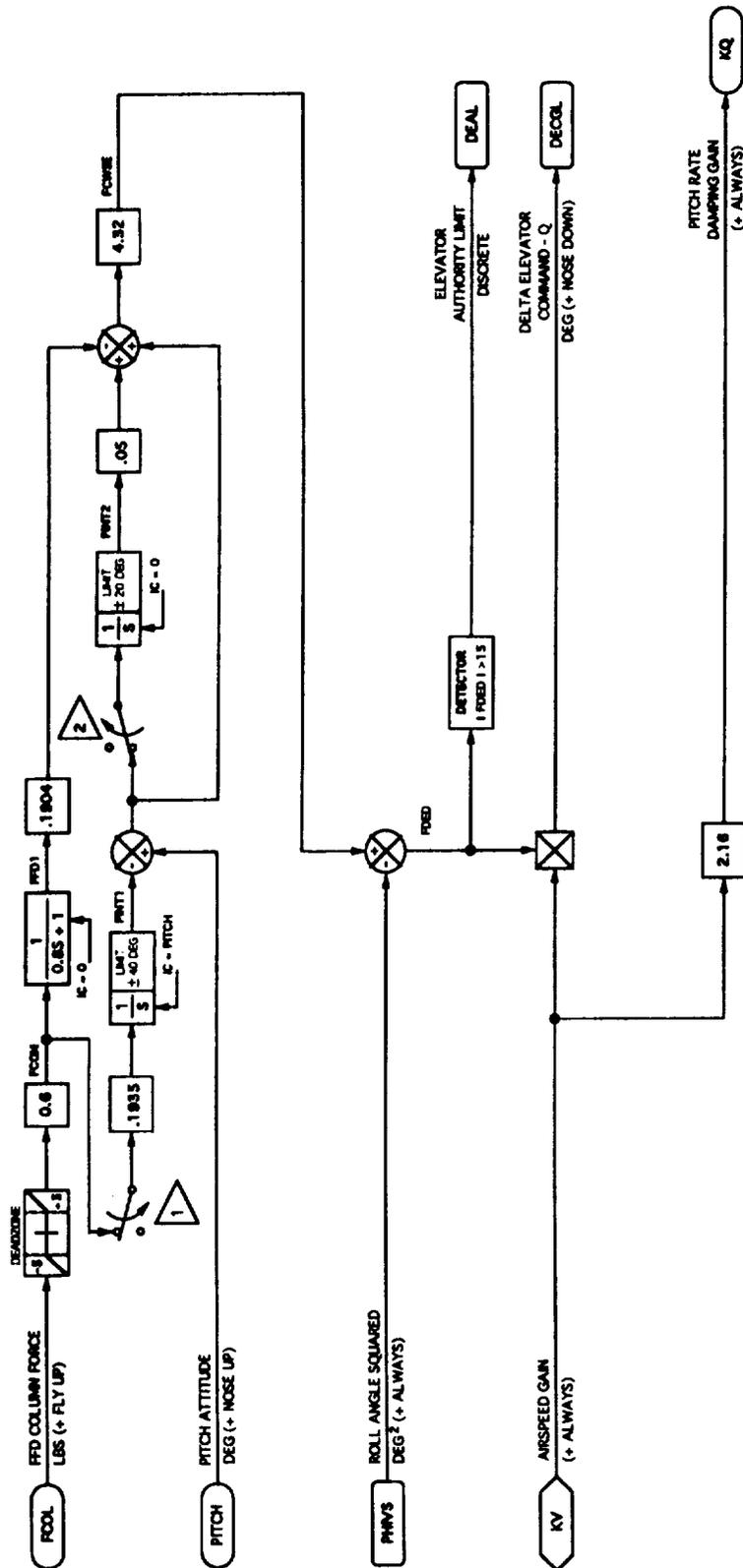
VERCMD PROCEDURE

(VTSBL * 10000) - 0) * (ATE * AUTO)
 BIT 20000 OF FOTUS (ADA WARNING TO DISPLAY) IS SET WHEN 'SHAKER' IS TRUE.



ELEV PROCEDURE

$\frac{1}{8}$ (MODIF * 00107) = 0

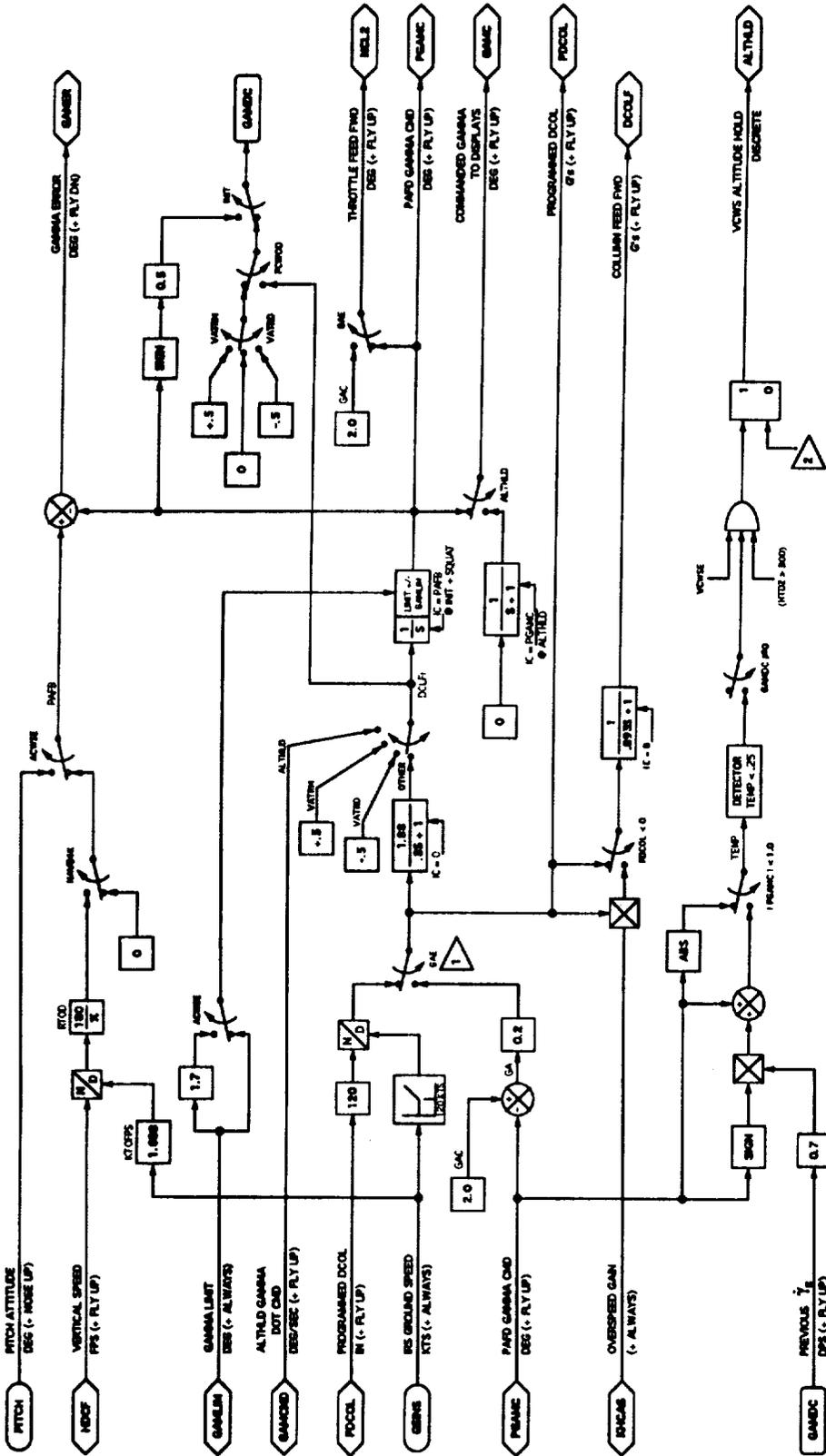


- 1 OPEN @ (DEAL • FIRST PASS)
- 2 OPEN @ (PCOIN • 0.05 • |FCOL| > 15) • FIRSTPASS

OUTPUT/INPUT STATIC GAINS	
VARIABLE	MAX GAIN
DEC/FOOL	0.5 DEG/LB
DEC/PITCH	4.32 DEG/DEG

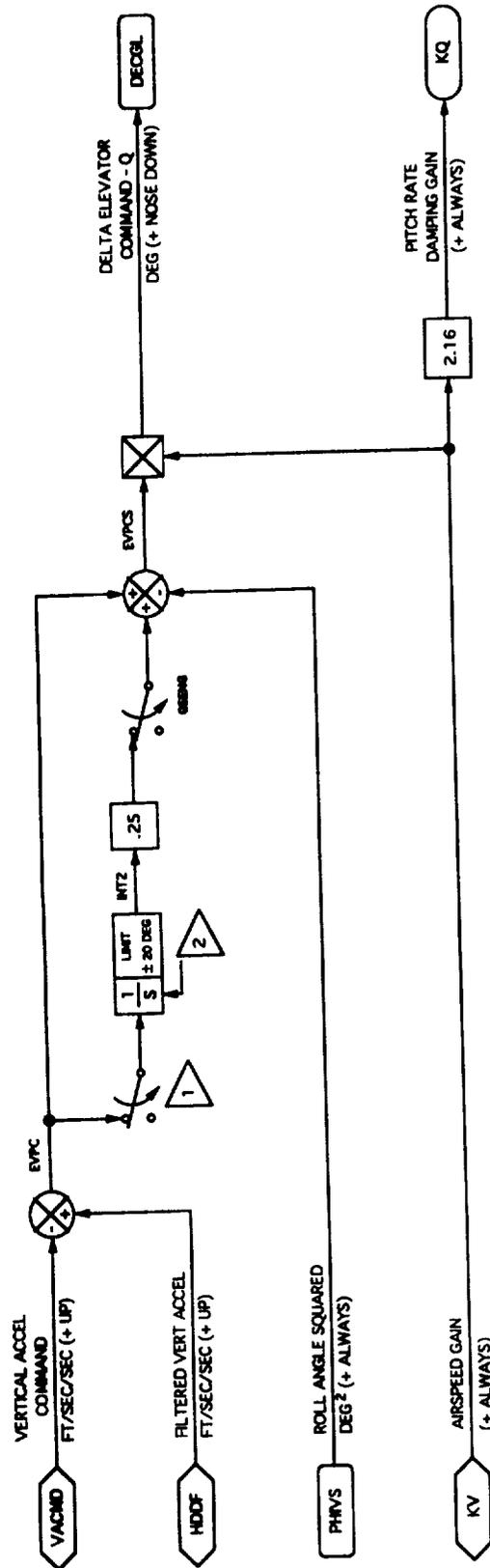
PFFD PROCEDURE

SUBROUTINE OF ELEVP



PAFD PROCEDURE
 SUBROUTINE OF ELEVP
 PAGE 1 OF 2

- 1 GAC IS RESET @ PWRD + (GA + DE)
- 2 RESET @ INT + PWRD + GAC + VAIRO + VAIRO

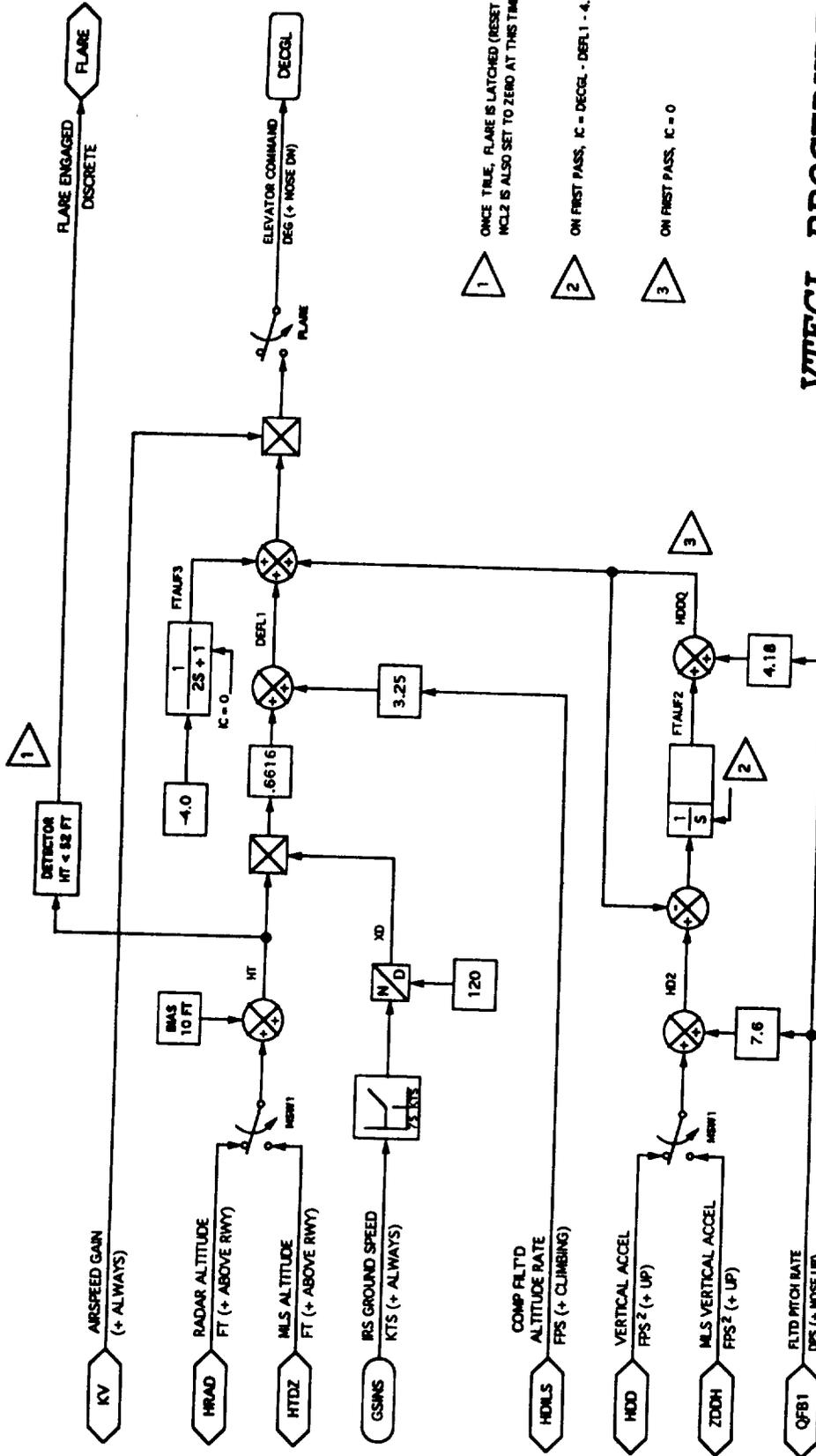


OUTPUT/INPUT STATIC GAINS	
VARIABLE	MAX GAIN
DEC/EVPC	1.0 DEG/PPS 2
DEC/PHVS	1.0 DEG/DEG 2

- 1 OPEN ● TRIMT • (MTZ • EIPC \geq 0)
- 2 IC = 4 (DECEL / KV + PHVS - EIPC)
VCWS ALTHLD IS CLEARED ON FIRST PASS OF PVPC

PVPC PROCEDURE

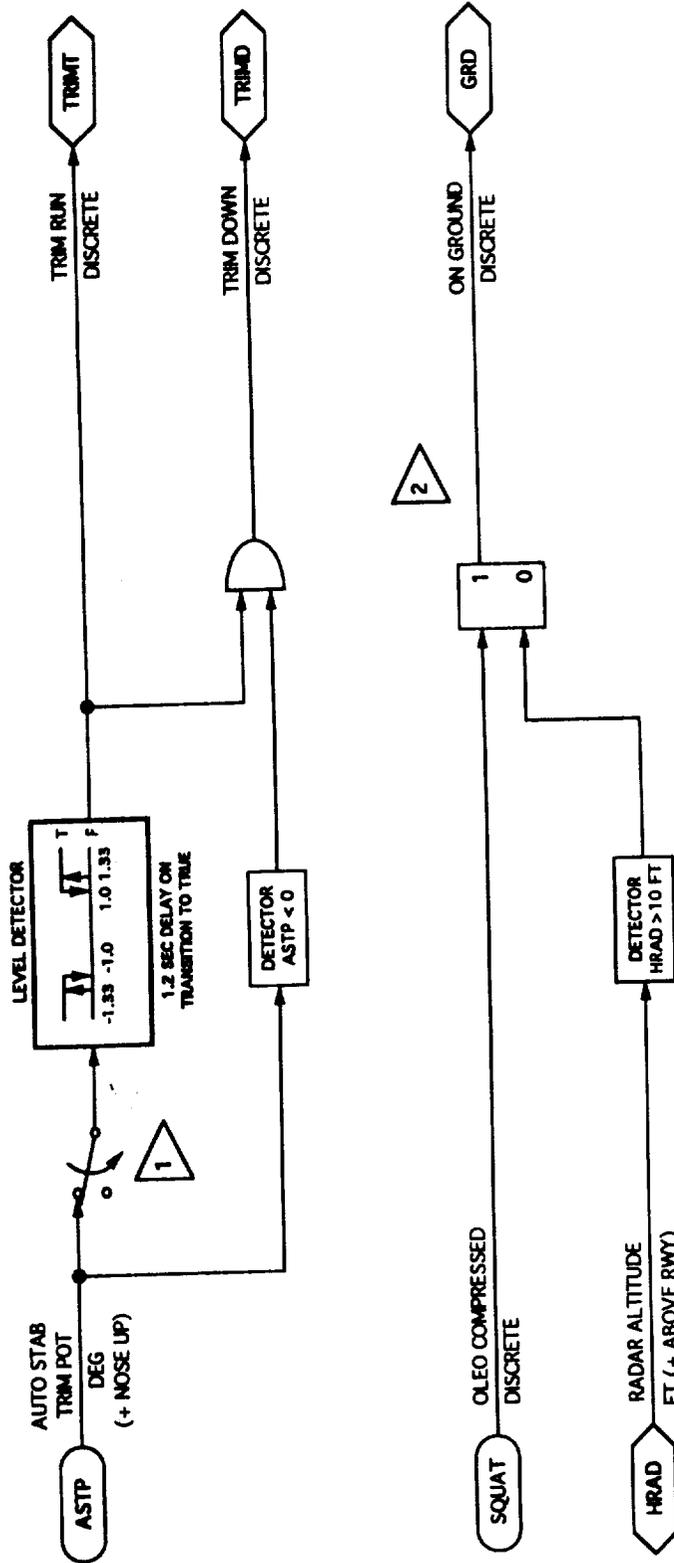
SUBROUTINE OF ELEVP



- 1 ONCE TIME, FLARE IS LATCHED (RESET @ LANDE)
MCL2 IS ALSO SET TO ZERO AT THIS TIME
- 2 ON FIRST PASS, IC = DECEL - DEFL1 - 4.18 QFB1
- 3 ON FIRST PASS, IC = 0

VTFCL PROCEDURE

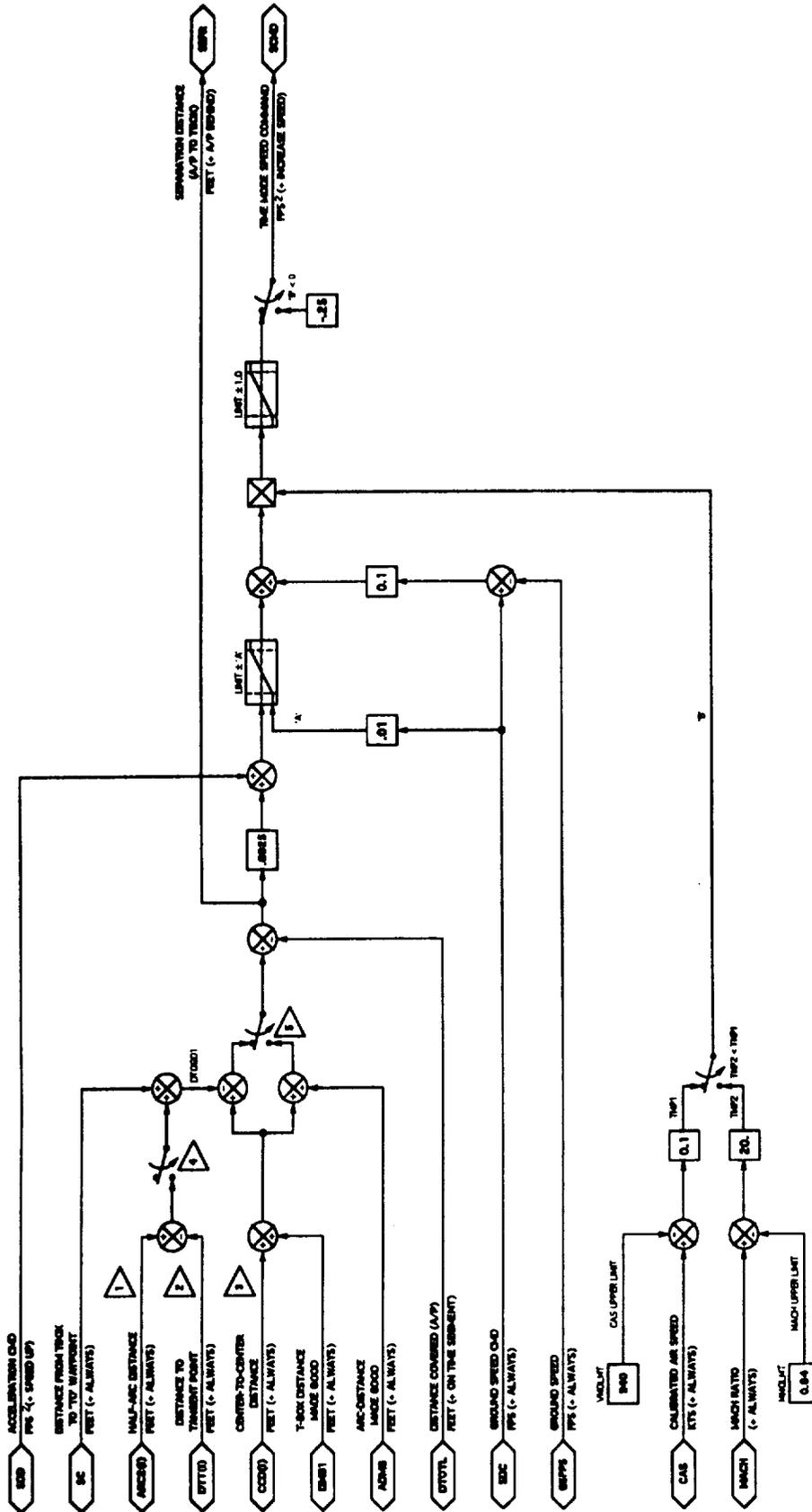
SUBROUTINE OF ELEVP



STAB PROCEDURE

SUBROUTINE OF ELEVP

- 1 GRD + MANEL + PRENG
- 2 GRD IS SET @ SQUAT AND CLEARED ON HRAD > 10 FT

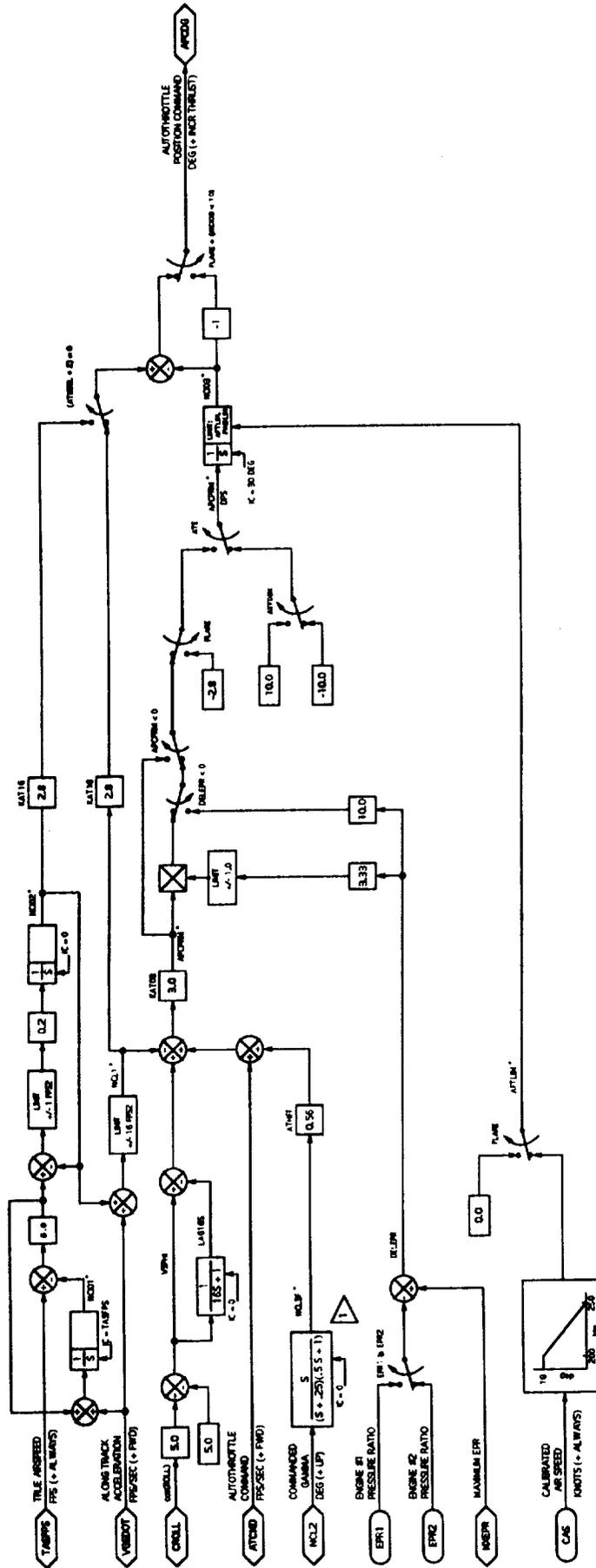


- 1 △ GUIDANCE BUFFER ENTRY
- 2 △ GUIDANCE BUFFER ENTRY WITH ACT(PTNRD) LIMIT
- 3 △ GUIDANCE BUFFER ENTRY WITH ACT(PTNRD) LIMIT AND OPENED IN SUBROUTINE LIST OF TOLND
- 4 △ SWITCH CLOSED WITH GUIDANCE BUFFER ENTRY WITH ACT(PTNRD) LIMIT IS FALSE
- 5 △ WITH ACT(PTNRD) LIMIT IS TRUE AND PREST(NU) OF ONE AND TURNS

CDG PROCEDURE

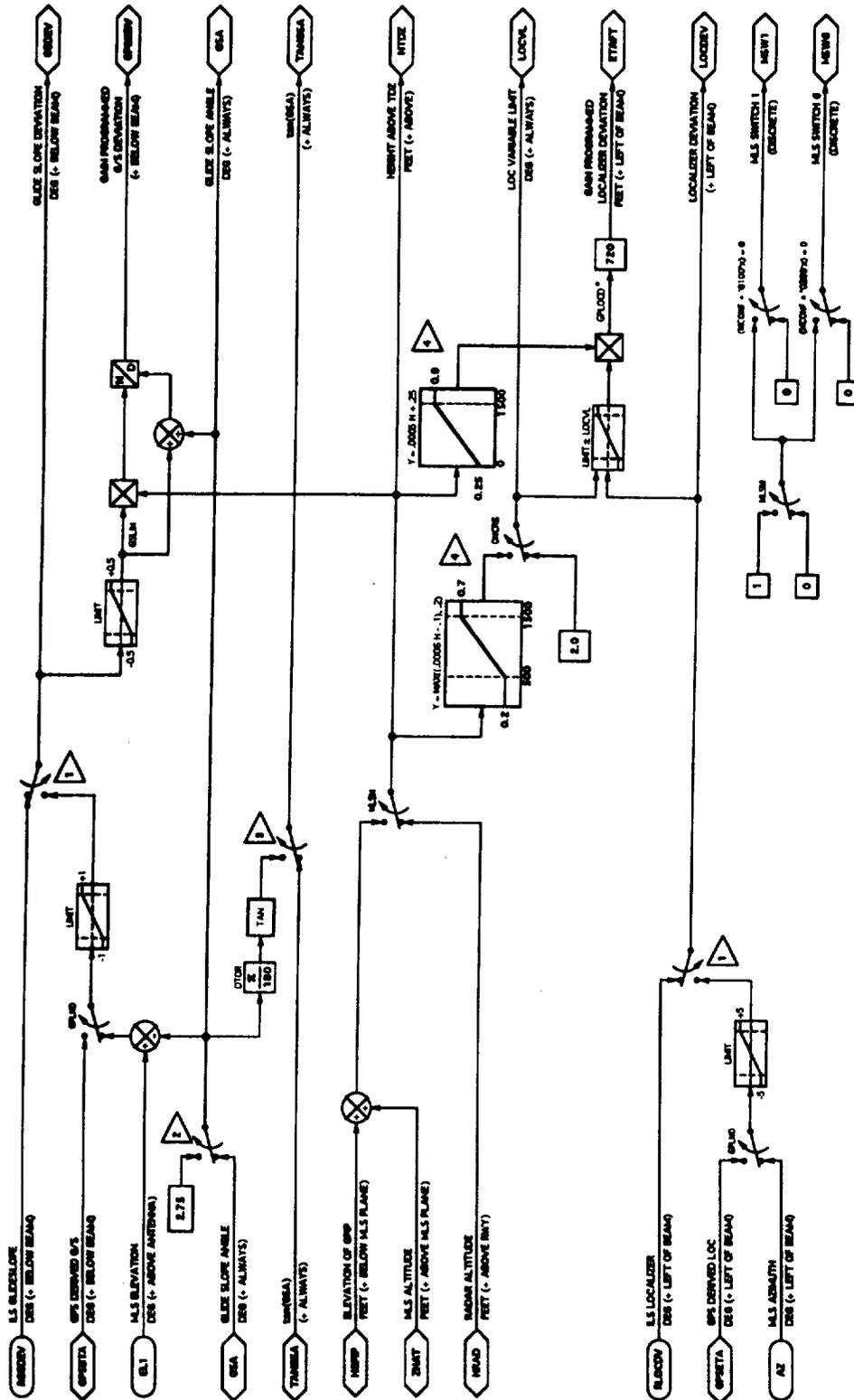
SUBROUTINE OF TOLND

PAGE 1 OF 1



ATHCL PROCEDURE

△ EPR = 3.1 MPZ
 ▲ ALWAYS + AUTO
 ▲ ALWAYS + AUTO



MLOG PROCEDURE

LAND COMPUTATIONS

- 1 (HICOF - 1000) = 0 = INDEX
- 2 GBA IS COMPUTED IF (LANDR + LANDS) (SMLS + INDEX) = (GBA - 0)
- 3 TANGBA IS COMPUTED IF (LANDR + LANDS) (SMLS + INDEX) = (TANGBA - 0)
- 4 MAX VALUE IS LIMITED BY HIRAD INRIT (MAX OF 1325 FEET)

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B

HORIZONTAL/VERTICAL GUIDANCE COMPUTATIONS:

The HVGUID software receives information about the present position of the airplane from the navigation software (HNAVFS) via data stored in global sections. Planned flight path information is obtained from the guidance buffer which is prepared by the path definition (PATHDF) software. These two sets of data are used in the computation of the major outputs of this module: cross track error (XTK), track error (TKE), advanced lateral control bank angle (ALCBA), altitude error (HER), commanded altitude rate (HDTC), planned flight path angle (PFPA), nominal ground speed (SDC) and acceleration command (SDD).

The outputs from HVGUID may be separated into horizontal (XTK, TKE and ALCBA) and vertical (HER, HDTC and PFPA) axes, with a short section devoted to the ground speed commands (SDD, SDC). The horizontal and vertical axes will be discussed separately.

Horizontal Guidance:

Processing begins in the HVGUID procedure and any of four different paths can be taken from there -- Initialization, Straight Segment processing, Turn processing, or Path Termination. Once the horizontal path calculations have been completed, control passes to the vertical guidance equations or to the executive if path end has been reached.

I. Initialization:

The initialization path is taken if either of the flags GUID2D, and NAV64K are not set. GUID2D is set when there are at least two waypoints in the guidance buffer and NAV64K is set when the ground speed exceeds 64 knots.

During initialization, the 'to' waypoint pointer (TOWPT) is limited to two or more ('2' indicating the second waypoint on the path) with the assumption that the airplane is initially in the area of the first waypoint. The pilot may select any waypoint on the path (except the first) as the 'to' waypoint via the CDU select page.

Once the proper 'to' waypoint has been determined, the 2D pointer (PTR2D) and the 4D pointer (PTR4D) for the airplane are set and the HVGP1 flag is cleared. The HVGP1 flag is used to indicate that one pass has been made through the HVGUID routine. After the HVGP1 flag is cleared, the program clears certain variables, sets up the waypoint normal vector (U12C) and transfers to the Vertical Guidance Off procedure (GD3D).

II. Straight Segment Calculations:

At the beginning of the straight segment the following flags and variables are cleared in the LEGSW procedure: TURN (Airplane in Turn Flag), TEND (Airplane halfway through Turn Flag), ALCFLG (Advanced Lateral Control Flag), ALCBA (Advanced Lateral Control Bank Angle) and AMG (Angle Made Good). In addition, DTOGO (Abeam Point to Next Waypoint Distance) is initialized to the initial value of DTG (Distance To Go). HVGP1 is then set and DTOGO is calculated from then on by the regular guidance equations.

The following quantities are computed in the straight segment portion of the routine:

1.) Airplane Position Unit Vector (P0):

The airplane position vector is one of the primary references in the Guidance Equations. The Airplane Position Unit Vector (P0) is calculated from the airplane latitude (LAT) and longitude (LON) supplied by the navigation algorithms. This value is computed in procedure HVGUID as follows:

$$\hat{P}_0 = \begin{bmatrix} \sin(\text{LAT}) \\ -\cos(\text{LAT}) \sin(\text{LON}) \\ \cos(\text{LAT}) \cos(\text{LON}) \end{bmatrix} = \begin{bmatrix} \text{SLAT} \\ -\text{SLON} * \text{CLAT} \\ \text{CLON} * \text{CLAT} \end{bmatrix}$$

The values CLAT, SLAT, CLON and SLON are the cosine and sine of latitude and longitude as computed by the Navigation equations. The following figure describes P0 in relationship to the position of the airplane and the path.

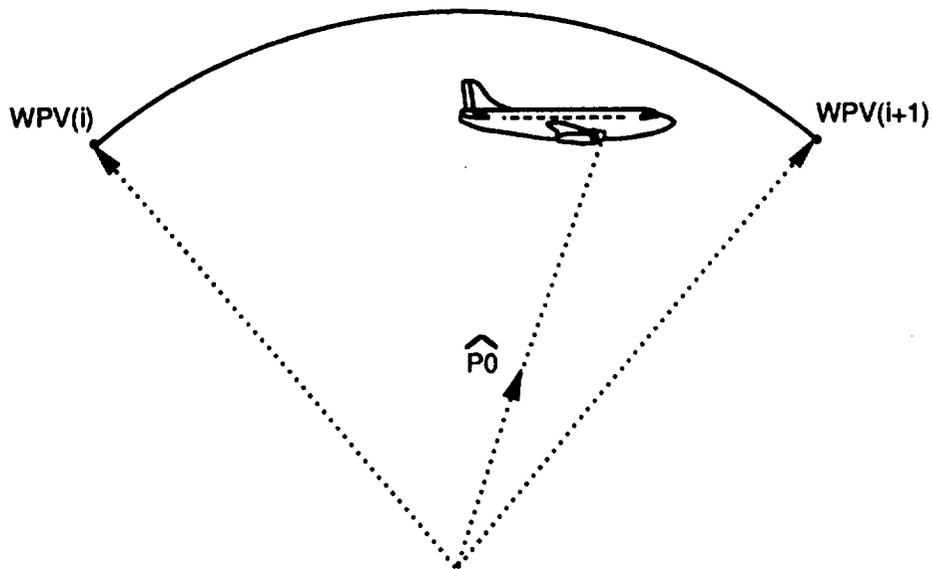


Figure B-1. Position Unit Vector (P0)

2.) Abeam Point Unit Vector (POP):

The Abeam Point Unit Vector (POP) points to the location on the great circle path to which the airplane has progressed at any time. A line from the airplane perpendicular to the plane of the great circle intersects the plane at point B, and the extension of a straight line from the earth's center through B intersects the great circle at the abeam point POP. The components of the corresponding unit vector POP are found by subtracting the component of P0 which is perpendicular to the plane of the great circle (i.e., parallel to U12C) from the vector P0 as follows:

$$\widehat{POP} = \widehat{P0} - (\widehat{P0} \cdot \widehat{U12C}) * \widehat{U12C}$$

where: U12C is the Flight Path Normal Unit Vector (WPT_ACT(i).NMV), calculated in the Path Definition routines.

POP locates the point on the great circle path to which the airplane has progressed at any particular time; therefore, the point POP is sometimes referred to as the airplane's 'progress point' as illustrated in figure B-2.

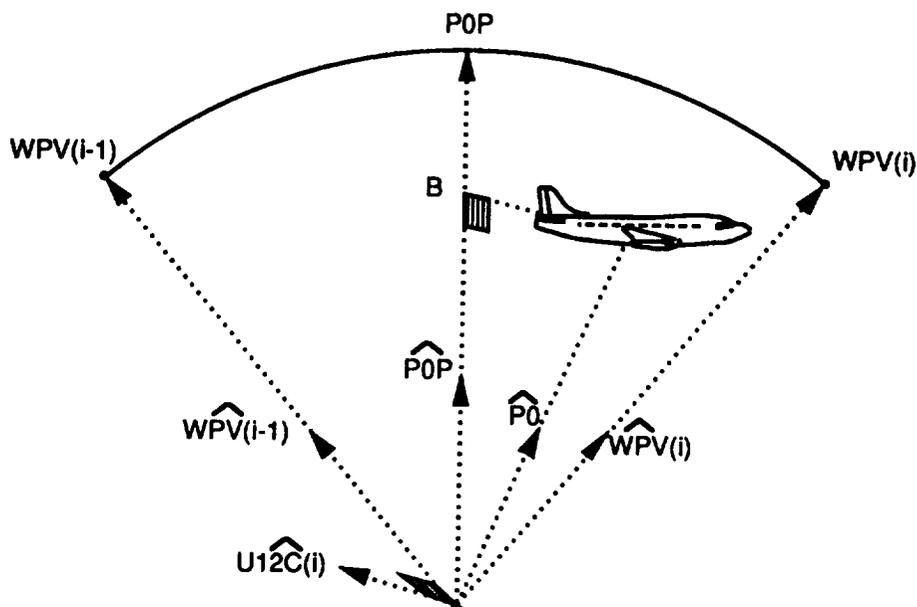


Figure B-2. Geometry of the Abeam Vector

3.) Abeam Point to Next Waypoint Distance (DTG):

The great circle distance from POP to WPV(i) is found by computing the angle subtended at the earth center by the unit vectors POP and WPV(i), and multiplying this angle in radians by the earth radius. The length of this cross product could be found by squaring each component of the product, adding and taking the square root. However, this lengthy computation can be avoided by dotting the cross product into the normal unit vector U12C. Since POP and WPV(i) both lie in the plane of the great circle, their cross is perpendicular to that plane and therefore parallel to U12C as given by:

$$\sin(\beta) = (\widehat{POP} \times \widehat{WPV}(i)) \cdot \widehat{U12C}$$

where: U12C is set to WPT_ACT(i).NMV in mainline code.

In addition to reducing computational time, the general technique of using the cross and dot products together makes it possible to maintain the sense (positive or negative) of the computed quantities. The cosine of the angle is found from the dot product as follows:

$$\cos(\beta) = \widehat{POP} \cdot \widehat{WPV}(i)$$

The angle is found from a double argument arctangent subroutine, hence the distance is given by:

$$DTG = Re * \arctangent(\sin(\beta), \cos(\beta))$$

where: Re is the local radius of the earth.

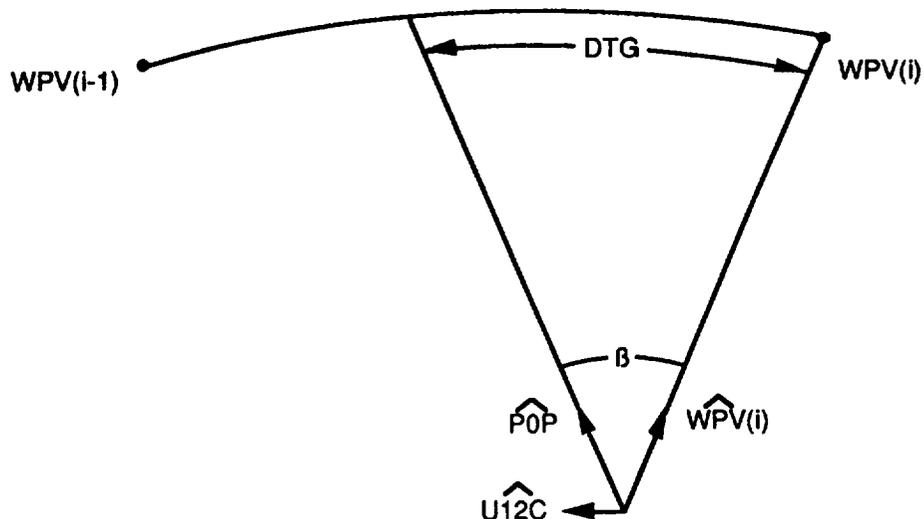


Figure B-3. Abeam Point to Waypoint Distance

4.) Abeam Point to Middle of Arc at the Next Waypoint Distance (DTOGO):

The purpose of the Abeam Point to the Middle of the Arc Distance (DTOGO) is to aid in determining the time to the next waypoint. The DTOGO value is determined by subtracting the Tangent Point to Waypoint distance, DTT(i) from the Abeam Point to Waypoint distance (DTG), and then adding the half Arc Distance (AO2) as given:

$$D\text{TOGO} = D\text{TG} - \text{WPT_ACT}(i) \cdot D\text{TT} + A\text{O2}$$

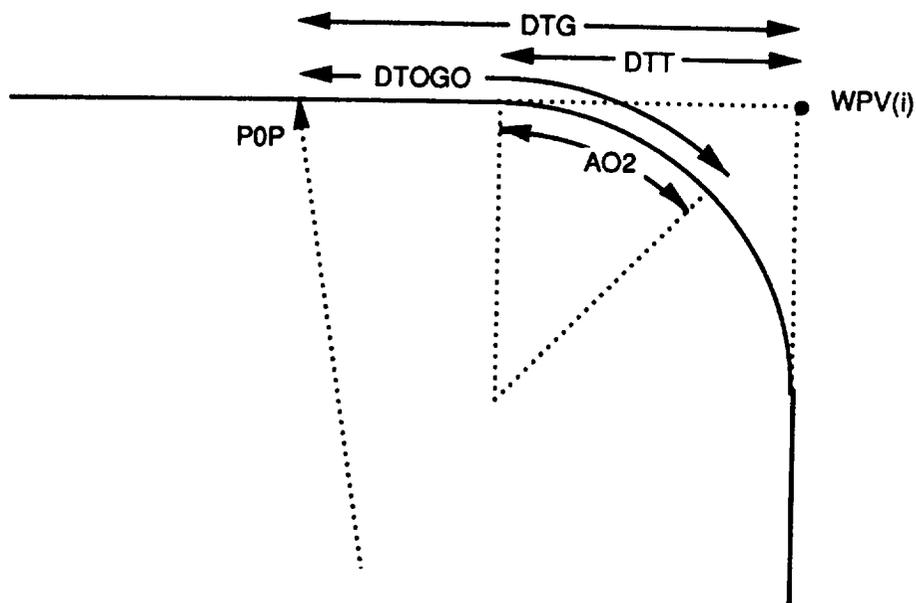


Figure B-4. Abeam Point to Middle of Arc

If the upcoming waypoint is the inbound waypoint of a DME-Arc, then the equation for the Abeam Point to the Middle of the Next Waypoint is simply the Abeam Point to Waypoint Distance (DTG) as given:

$$D\text{TOGO} = D\text{TG}$$

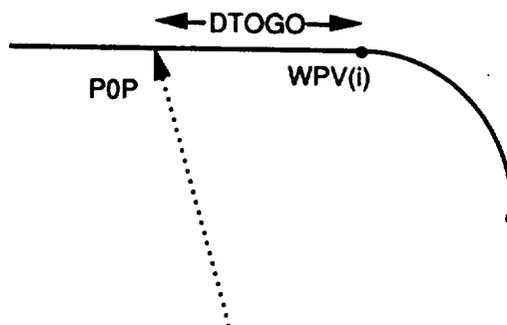


Figure B-5. DME-Arc Distance

5.) Crosstrack Distance to the Planned Flight Path (XTK):

The crosstrack distance to the planned flight path (XTK) is used to compute the Lateral Steering Command (LATSTR) as well as determining the status of the Be Careful Flag (BCFLAG). The crosstrack distance is the distance from the airplane to the abeam point. It is positive when the airplane is to the right of the path. The distance is computed by multiplying the angle between the airplane position unit vector (\hat{P}_0) and the abeam point unit vector (\hat{P}_{0P}) in radians, by the radius of the earth as given:

$$XTK = R_e * \arctangent(\sin(\alpha), \cos(\alpha))$$

Where: R_e = local radius of the earth

$$\sin(\alpha) = -\hat{P}_0 \cdot \hat{U}_{12C}$$

$$\cos(\alpha) = \hat{P}_0 \cdot \hat{P}_{0P}$$

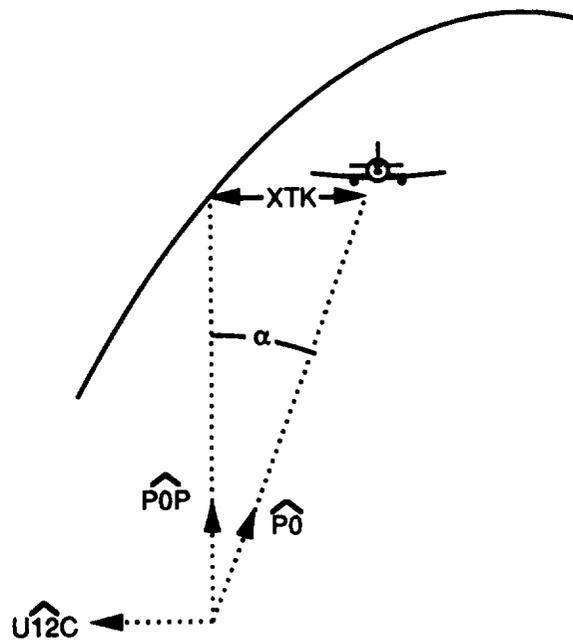


Figure B-6. Crosstrack Distance

6.) Desired Track Along a Great Circle Path (DSRTK):

By definition, the desired track is the track angle relative to true north of a vector tangent to the great circle path at the abeam point POP, pointing in the forward direction along the path. The computation of the desired track involves the unit normal vector (U12C) and a unit vector M which points westward at POP. A westward pointing vector was chosen because, unlike a north pointing vector, one component is always zero and the other two components depend only on longitude, thus simplifying the calculations. The vector M is given by the equations:

$$\hat{M} = \begin{bmatrix} 0 \\ \text{POP}_3 \\ -\text{POP}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \cos(\text{LON}) \\ \sin(\text{LON}) \end{bmatrix}$$

where LON is the longitude at the abeam point POP. The orientation of the unit vectors M and U12C to the path are shown below.

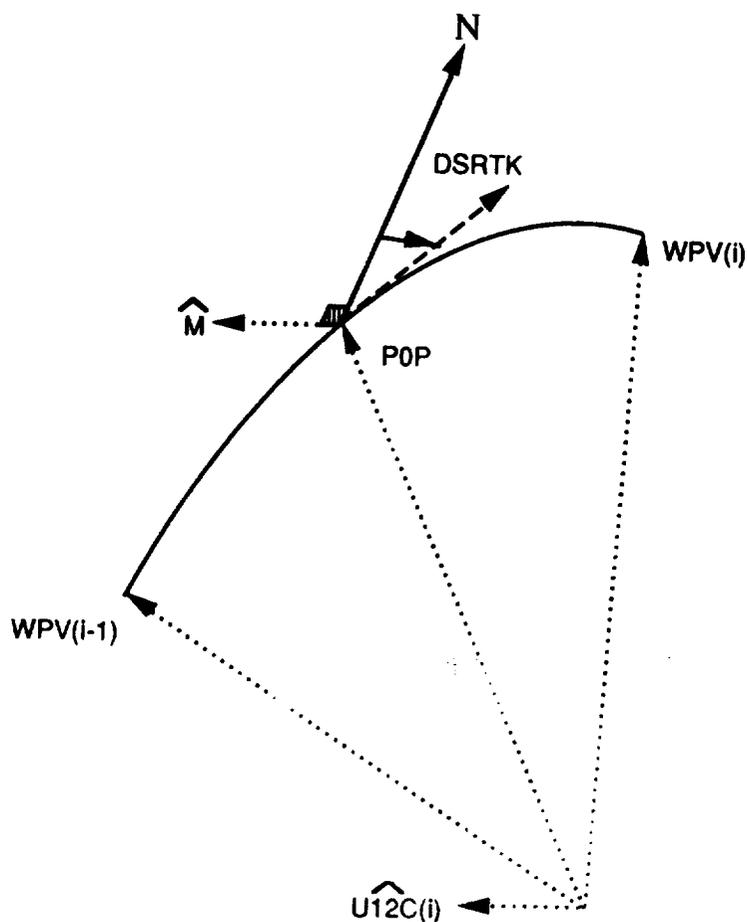


Figure B-7. Desired Track

The desired track DSRTK is found from the equation:

$$\text{DSRTK} = \arctangent(\sin(A), (\cos(A)))$$

$$\text{Where: } \sin(A) = (\widehat{U12C} \times \widehat{M}) \cdot \widehat{POP}$$

$$\cos(A) = \widehat{U12C} \cdot \widehat{M}$$

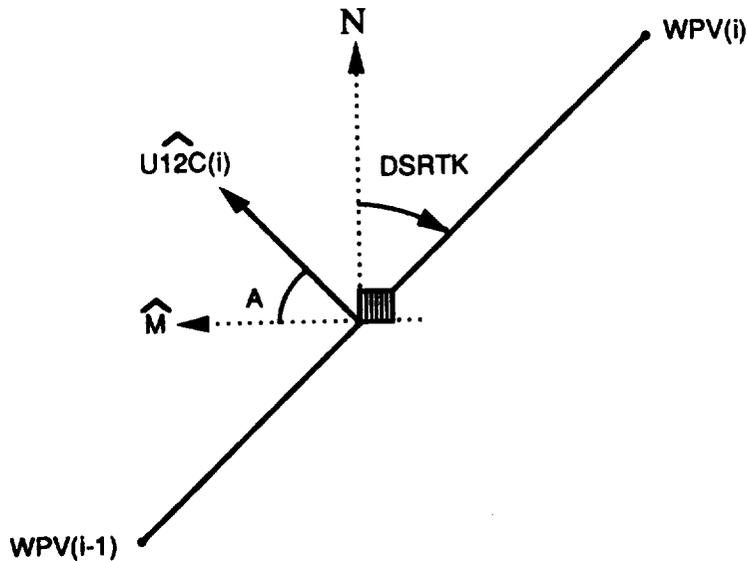


Figure B-8. End View of \widehat{POP}

The computation of $\sin A$ utilizes a dot product with \widehat{POP} to convert the vector $\widehat{M} \times \widehat{U12C}$ to a scalar quantity. This is legitimate because \widehat{POP} is a unit vector parallel to $\widehat{M} \times \widehat{U12C}$.

7.) Track Angle Error (TKE):

The Track Angle Error is used as one of the parameters in the Lateral Steering Command (LATSTR) calculations. This value is computed from the airplane's track (TK) and the Desired Track Angle (DSRTK).

$$\text{TKE} = \text{TK} - \text{DSRTK}$$

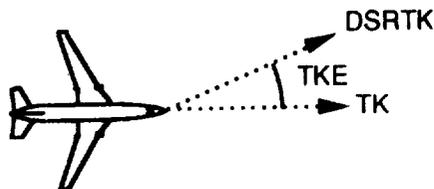


Figure B-9. Track Angle Error

III. Turn Computations:

The turn calculations start when the Abeam Point of the airplane gets to the Tangent Point of the turn as shown below.

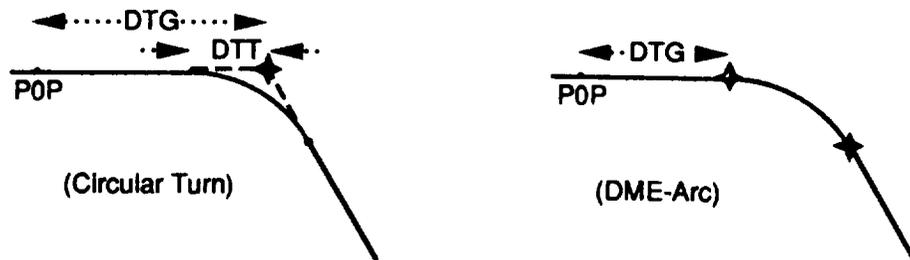


Figure B-10. Distances to the Turn

There are three criteria for determining that the guidance routine should be making turn calculations:

1. TURN = TRUE
2. $DTG < (VGS * DELTAT)$
3. $DTG < WPT_ACT(I).DTT$

The first check is for the turn discrete (TURN). If the guidance equations have previously decided that the airplane is in the turn, TURN is set. The second test determines if the distance the airplane will travel before the next time through the loop ($VGS * DELTAT$) is greater than or equal to the Abeam Point to Next Waypoint Distance (DTG). This test is used in the DME-Arc case where the inbound waypoint WP(IN) is at the Tangent Point. So when the airplane passes the waypoint, the guidance routine should switch to the turn calculations. The third test is for the circular turns.

The following quantities are computed in the Turn portion of the routine:

1. The Magnitude of the Track Angle Change (MAGTA)
2. The Turn Normal Unit Vector (U12C)
3. The Abeam Point to Middle of Arc at Next Waypoint Distance (DTCGO)
4. The Crosstrack Distance_to_Planned_Flight_Path (XTK) during turns
5. The Desired Track (DSRTK) during turns
6. The Turn Angle Made Good (AMG)
8. The Nominal Bank Angle (NOMBA)
9. The Distance Before the Turn to Apply Advanced Lateral Control (RALC)
10. The Advanced Lateral Control Exit Angle (ALCXA)
11. The Advanced Lateral Control Bank Angle (ALCBA)

1.) Magnitude of the Track Angle Change (MAGTA):

The Magnitude of the Track Angle Change (MAGTA) is used to determine the midpoint and end of a turn. It is calculated by taking the absolute value of the Track Angle Change (TA) defined for the next waypoint, as follows:

$$\text{MAGTA} = | \text{WPT_ACT}(i).\text{TA} |$$

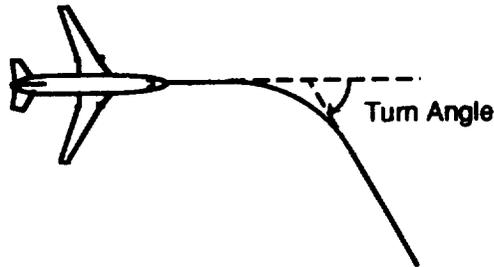


Figure B-11. Turn Magnitude

2.) Turn Normal Unit Vector (U12C):

The Turn Normal Unit Vector is used to calculate airplane position for Horizontal Guidance during turns. It is derived by subtracting from the difference between the Airplane Position Vector (\hat{P}_0) and \hat{POP} (set equal to the Turn Center Unit Vector (TCV) in subroutine AAA), the component of that vector in the Turn Center Unit Vector direction $((\hat{P}_0 - \hat{POP}) \cdot \hat{POP}) * \hat{POP}$, as given:

$$\hat{U}_{12C} = \hat{TMPV} - \text{sign}(TA) * ((\hat{TMPV} \cdot \hat{POP}) * \hat{POP})$$

$$\text{Where: } \hat{TMPV} = \hat{P}_0 - \hat{POP}$$

TA is positive for right turn, negative for left turns

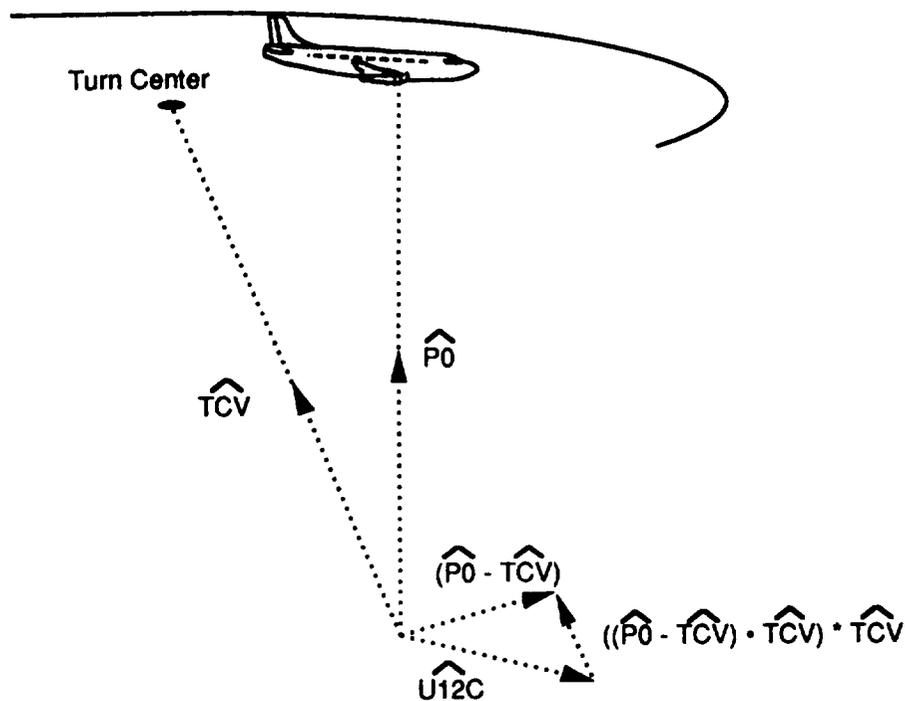


Figure B-12. Turn Normal Unit Vector

3.) The Abeam Point to Middle of Arc at Next Waypoint Distance (DTOGO)

DTOGO is adjusted during turns according to the position in the turn. During the first half of a turn, DTOGO is the half-arc distance minus the angle made good times the turn radius:

$$\text{DTOGO} = \text{AO2} - \text{RTN} * \text{AMG} * \text{DTOR}$$

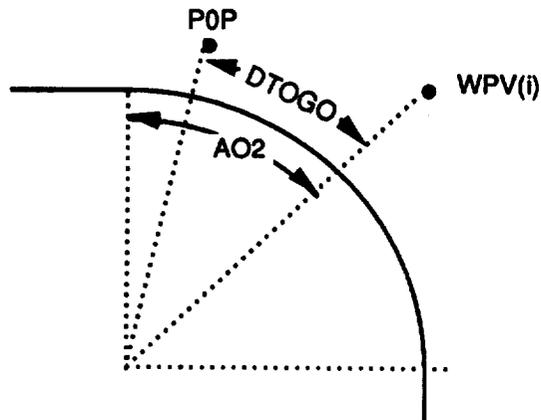


Figure B-13. Abeam Point to Middle of Arc during first half of turn

During the second half of the turn, the center-to-center distance (WP_ACT.CCD) between the previous and next waypoints is added as follows:

$$\text{DTOGO} = \text{DTOGO} + \text{WP_ACT}(\text{PTR2D}+1).\text{CCD}$$

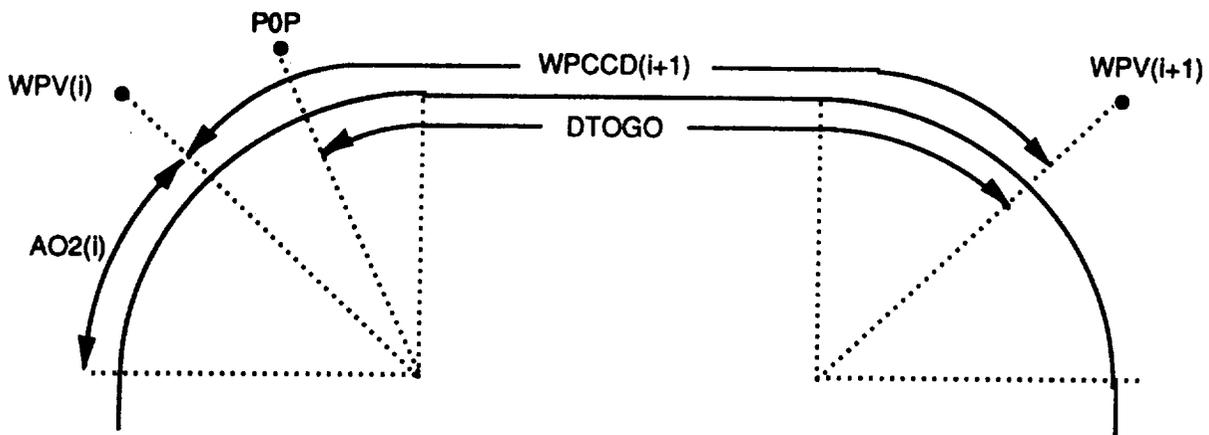


Figure B-14. Abeam Point to Middle of Arc during second half of turn

4.) Crosstrack Distance to Planned Flight Path during turns (XTK):

The purpose of Crosstrack Distance (XTK) is to compute the Lateral steering signal LATSTR and to determine the status of the Be Careful Flag, BCFLAG. The Crosstrack distance is calculated as the distance to the turn center, then adjusted by subtracting the turn radius to reflect the distance from the airplane to the Abeam Point as given:

$$XTK = R_e \cdot \arctangent(-(\hat{P}_0 \cdot \hat{U}_{12C}), (\hat{P}_0 \cdot \hat{P}_{OP})) + RTN \cdot \text{sign}(TA)$$

Where: R_e is the local radius of the earth
 P_{OP} is set equal to $WPT_ACT(i).TCV$

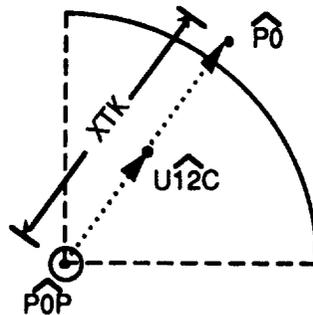


Figure B-15. Crosstrack Distance to POP

5.) Desired Track Angle During Turns (DSRTK):

The Desired Track Along the Great Circle Path is used to compute the Track Angle Error (TKE), an important element of the Lateral Steering Signal (LATSTR). The Desired Track calculation is modified during turns by replacing the path normal unit vector (NMV) with U_{12C} , and P_{OP} with TCV (refer to DSRTK for straight segments), as follows:

$$DSRTK = \arctangent(((\hat{U}_{12C} \times \hat{M}) \cdot \hat{P}_{OP}), (\hat{U}_{12C} \cdot \hat{M}))$$

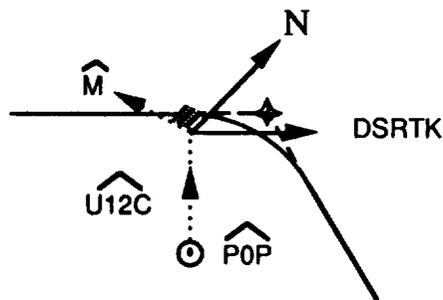


Figure B-16. Desired Track During Turns

6.) Turn Angle Made Good (AMG):

The purpose of computing the Turn Angle Made Good (AMG) is to continually determine the progress of the airplane around the turn. The Angle Made Good is the angle between the Path Normal Unit Vector (NMV) and the Turn Normal Unit Vector (U12C), as given:

$$\sin(\text{AMG}) = (\widehat{U12C} \times \widehat{NMV}(i)) \cdot \widehat{TCV}(i)$$

$$\cos(\text{AMG}) = \widehat{U12C} \cdot \widehat{NMV}(i)$$

$$\text{AMG} = \text{arctangent} (\sin(\text{AMG}), \cos(\text{AMG}))$$

If the turn angle is negative, ($\text{.TA}(i) < 0$), then:

$$\text{AMG} = 360 - \text{AMG}$$

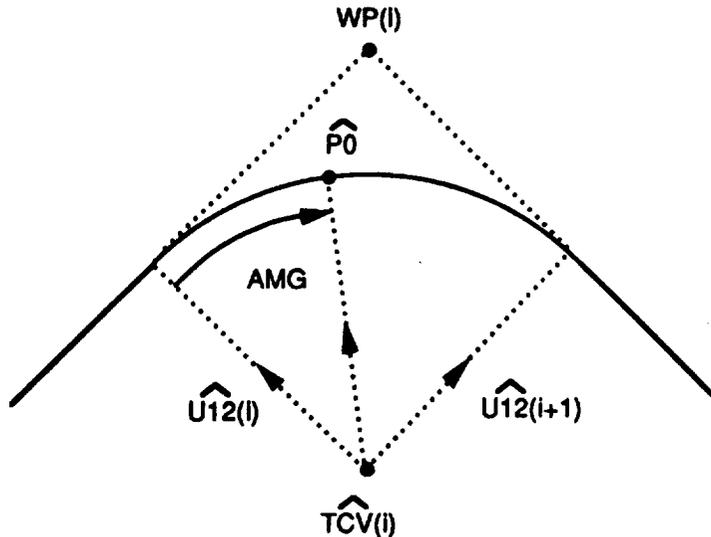


Figure B-17. Turn Angle Made Good

7.) Nominal Bank Angle (NOMBA):

The Nominal Bank Angle is the bank angle necessary for the airplane to turn at the specified turn radius. It is used for the Advanced Lateral Control equations and as an element of the Lateral Steering Command. It is based on the turn radius and speed of the airplane, as given:

$$\text{NOMBA} = \text{arctangent} (\text{GSFPS}^2 / (\text{GRAVO} * \text{RTN}))$$

where: GRAVO = Nominal acceleration of gravity
RTN = Radius of turn in feet.

8.) Distance to Apply Advanced Lateral Control (RALC):

The roll command must be given prior to the point of tangency (tp) so that the airplane has enough time to be at the proper bank angle when it begins the turn. The time necessary for the roll is approximately equal to the nominal bank angle divided by the roll rate limit of the control system:

$$t = \text{NOMBA} / \text{ROLL RATE LIMIT}$$

Then the distance before the tangent point is:

$$\text{RALC} = \text{NOMBA} * 0.25 * \text{GSFPS}$$

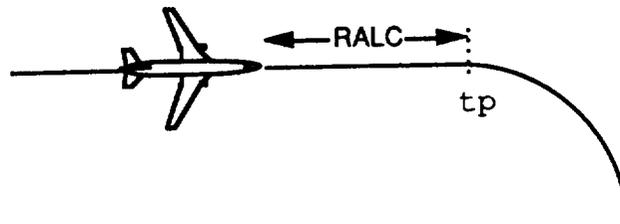


Figure B-18. ALC Advance Distance

9.) Distance to Remove Bank Angle Command (ALCXA):

In order to smoothly meet the next leg of a flight path, it is necessary to begin the roll out after a turn before the end of the turn. The roll out to wings level is begun when the arc length remaining is equal to the time necessary for roll out times the angular speed of the airplane (GSFPS/WPRTN), as given:

$$\frac{\text{NOMBA}}{\text{Roll Rate Limit}} = \frac{\text{GSFPS} * \text{RTOD}}{\text{RTN}} = \frac{\text{RTOD} * \text{RALC}}{\text{RTN}}$$

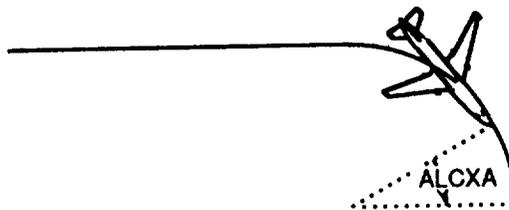


Figure B-19. ALC Exit Angle

10.) Advanced Lateral Control Bank Angle (ALCBA):

The Advanced Lateral Control Bank Angle (ALCBA) is the element of the Lateral Steering Signal (LATSTR) that represents the lateral acceleration necessary to stay on the curved path during turns. ALCBA is only used when the airplane is beyond the distance before the turn to apply Advanced Lateral Control (RALC) and has not yet reached the distance before the end of turn to remove the Advanced Lateral Control (ALCXA). The bank angle is calculated from the Nominal Bank Angle (NOMBA) and the sign of the Waypoint Turn Angle (WPTA) as follows:

$$\text{ALCBA} = \text{NOMBA} * \text{sign}(\text{TA})$$

The Bank Angle is computed under the following conditions:

1. when $\text{DTOGO} \geq \text{RALC} + \text{AO2}$
then: the airplane has not reached the roll-up point
(RALC) and $\text{ALCBA} = 0$
2. when $\text{DTOGO} < \text{RALC} + \text{AO2}$
then: past the roll-up point but not yet in the turn
and $\text{ALCBA} = \text{Sign}(\text{TA}) * \text{NOMBA}$
3. when $\text{MAGTA} - \text{AMG} \geq \text{ALCXA}$
then: in the turn, not to roll out point and
 $\text{ALCBA} = \text{Sign}(\text{TA}) * \text{NOMBA}$
4. when $\text{MAGTA} - \text{AMG} < \text{ALCXA}$
then: beyond the roll out point and $\text{ALCBA} = 0$

IV. Vertical Guidance Computations

The following quantities are computed in the Vertical Guidance portion of the routine:

1. The Abeam Point Commanded Altitude (HC)
2. The Vertical Path Error (HER)
3. The Vertical Guidance Switch Point Distance (HDIS)
4. The Raw Vertical Speed Command (HDTCR)
5. The Vertical Speed Command (HDTC)
6. The Planned Flight Path Angle (PFPA)

1.) Abeam Point Commanded Altitude (HC):

The change from $WPALT(i-1)$ and $WPALT(i)$ is a linear transition with distance beginning at the midturn of $WP(i-1)$ and ending at the midturn of $WP(i)$. The Waypoint to Waypoint Gradient ($WPT_ALT.FPA$) describes this transition. To determine the abeam point planned altitude it is only necessary to subtract from the waypoint height the product of the Waypoint to Waypoint Path Gradient and the distance to the midpoint (DTORG), as given:

$$HC = WPT_ACT(i).ALT - DTORG * WPT_ACT(i).FPA * DTOR$$

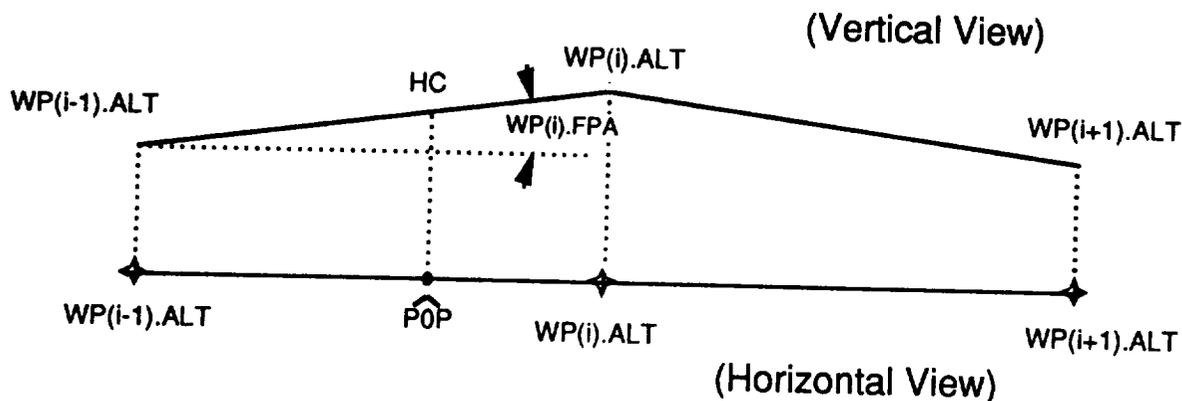


Figure B-20. Commanded Altitude

Because the DME-Arc parameters are defined and computed differently, the computation of HC during the first half of the DME-Arc becomes:

$$HC = WP_ACT(i+1).ALT - DTORG * WP_ACT(i+1).FPA * DTOR$$

2.) Vertical Path Error (HER):

The Vertical Path Error (HER) is one of the major elements used in the calculation of the Altitude Rate Command (VSTRA). HER is the difference between the Abeam Point Commanded Altitude (HC) and the actual altitude (ALTCOR), as given:

$$HER = HC - ALTCOR$$

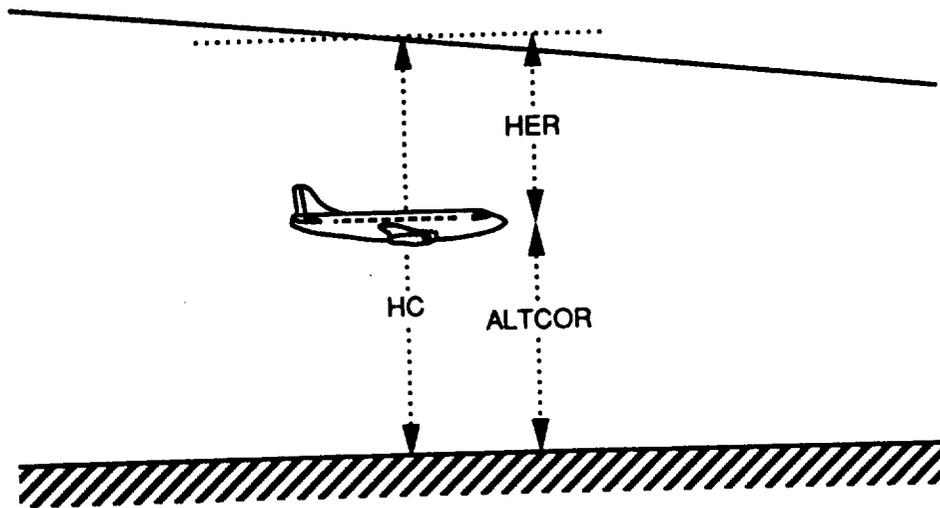


Figure B-21. Altitude Error

3.) Vertical Guidance Switch Point Distance (HDIS):

The 3D pointers for the Waypoint to Waypoint Path Gradient (WP_ACT.FPA) are switched (or updated) prior to reaching the waypoint to anticipate vertical maneuvers. This is the equivalent in the vertical axis of the Advanced Lateral Control in the lateral axis. The switch point is calculated according to ground speed, change in flight path angle at the waypoint and the nominal vertical acceleration rate (2 fps/sec):

$$\text{HDIS} = (\text{FPA} - \text{FPAN}) * \text{DTOR} * \text{GSFPS}^2 / 2$$

Where: FPA is the path gradient of the present segment (in radians)

FPAN is the path gradient of the next segment (in radians).

If HDIS < 300. HDIS is set to 300.

PTR3D is updated when DTOGO becomes less than HDIS.

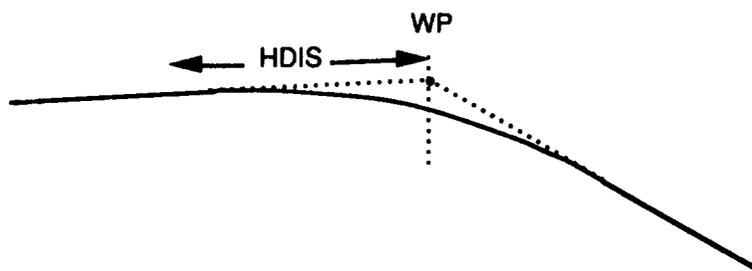


Figure B-22. Switch Point Distance

4.) Raw Vertical Speed Command (HDTCR):

The raw vertical speed command (HDTCR) is the input to the rate limited vertical speed command (HDTC). HDTCR is derived from the Waypoint to Waypoint Path Gradient (FPA) and the current ground speed of the airplane (GSFPS), as given:

$$\text{HDTCR} = \text{GSFPS} * \text{FPA} * \cos(\text{TKE})$$

Where: FPA is the current segment path gradient (in radians)

5.) Vertical Speed Command (HDTC):

The vertical speed command is one of the three major elements which make up the Altitude Rate Command (VSTRA). HDTC is developed by passing the raw vertical speed command (HDTCR) through a rate limited lag. As indicated, the lag is 6.25 seconds, with a maximum vertical acceleration of 2.0 fps².

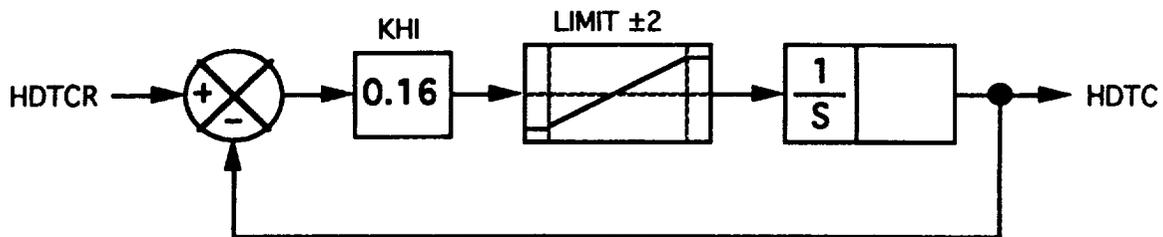


Figure B-23. Vertical Speed Command

6.) Planned Flight Path Angle (PFPA):

The Planned Flight Path Angle (or commanded gamma) is used in the calculation of the Flight Path Angle Error displayed on the CDU. It was previously used to drive the dashed gamma wedges on an earlier version of the PFD, and may optionally be used to drive the gamma wedges on the present Primary Flight Display (PFD). The PFPA is determined by dividing the vertical speed command (HDTC) by the present Ground Speed of the Airplane (GSFPS).

$$\text{PFPA} = \text{RTOD} * \text{HDTC} / \text{GSFPS}$$

V. Speed Guidance Computations

The following quantities are computed for use in the ground speed and/or time path (4D) guidance computations:

1. The Average Path Acceleration (SDD)
2. The Abeam Point Commanded Ground Speed (SDC)
3. The Airplane Distance Made Good (DMG)
4. The Progress Distance of the Airplane (DTOTL)

1.) The Average Path Acceleration (SDD) is calculated in fps per foot as given:

$$SDD = \frac{KTOFPS * (WPT_ACT(TOWPT).GS - WPT_ACT(TOWPT-1).GS)}{WPT_ACT(TOWPT).CCD}$$

where: TOWPT is set equal to PTR4D except during the first half of a DME_arc turn, when it is set equal to PTR4D+1.

2.) Abeam Point Commanded Ground Speed (SDC):

The Abeam Point Commanded Ground Speed is calculated from the defined waypoint groundspeeds (WPT_ACT.GS), the waypoint center to center distance (WPT_ACT.CCD), the abeam point to middle of arc at the next waypoint distance (DTOGO) and the average path acceleration (SDD). SDC is used for Ground Speed mode and (by TGUID) time path (4D) guidance.

$$SDC = KTOFPS * WPT_ACT(TOWPT).GS - SDD * DTOGO$$

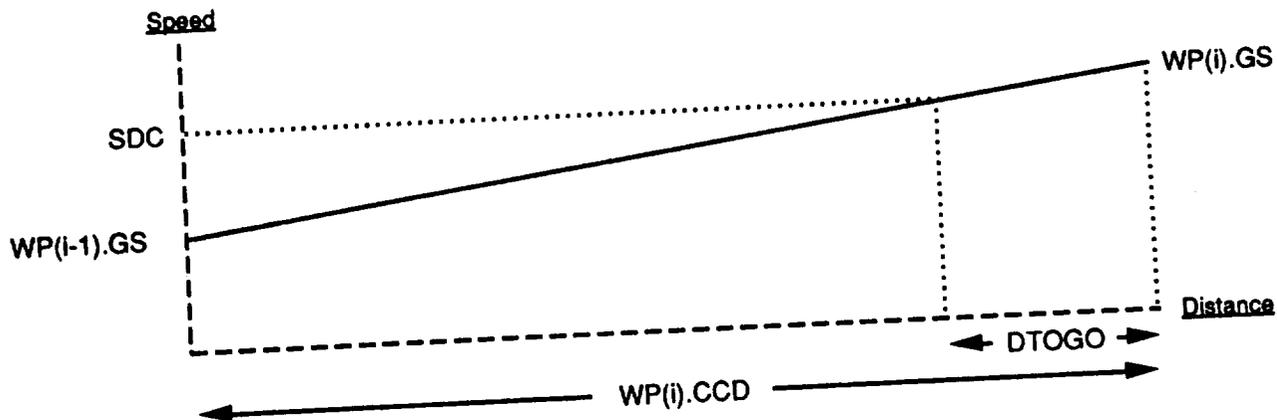


Figure B-24 Velocity Profile

3.) Airplane Distance Made Good (DMG):

The Airplane Distance Made Good (DMG) is the length of the path segments between both the reference waypoint and the airplane. The reference waypoint is the waypoint closest to but behind the airplane and the time box. When the airplane reaches the end of a path segment, the length of that segment (WPCCD) is added to DMG as given below. In the TGUID routine the DMG is compared to the Distance Made Good of the time box (DMG1) and they are used to update each other.

when: $2 * AMG > MAGTA$
 then: $DMG = DMG + WPCCD$

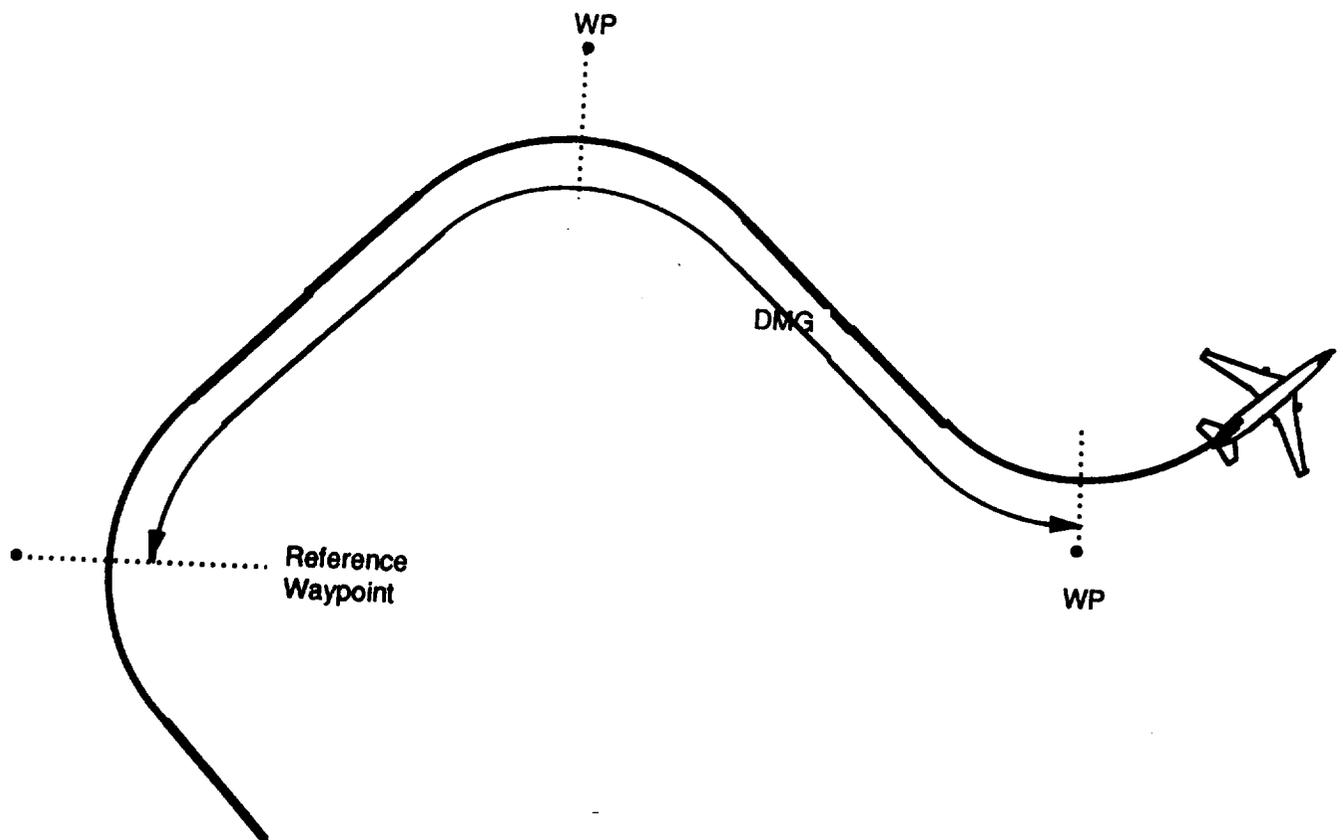


Figure B-25. Distance Made Good

4.) Progress Distance of the Airplane (DTOTL):

The Progress Distance (DTOTL) is used in the calculation of Separation Distance (SEPR) in the Time Path routine. The Progress Distance is calculated from the Distance Made Good (DMG) plus the Turn Center to Turn Center Distance (WPCCD) of the 'to' Waypoint, minus the Abeam Point to the Next Waypoint Center of Turn Distance (DTOGO), as given:

$$DTOTL = DMG + WPCCD(i) - DTOGO$$

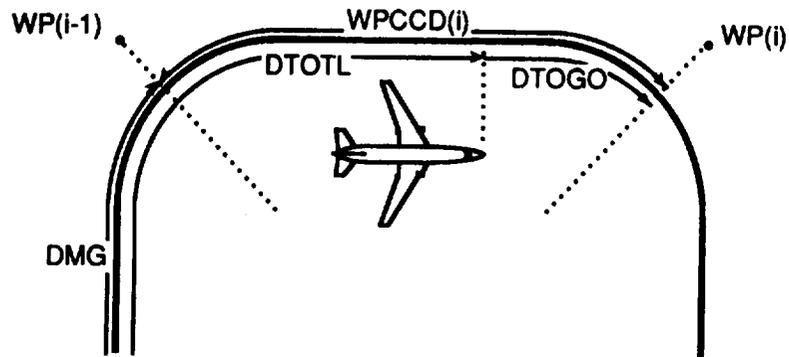


Figure B-26. Progress Distance

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C

MICROWAVE LANDING SYSTEM (MLS) OVERVIEW

The conventional MLS ground equipment transmits two cone-shaped time-reference scanning radio beams from the runway area (Figure C-3). One beam scans 60 degrees from side to side of the runway center at a rate of 13 1/2 times per second to provide azimuth (AZ) referencing. The second beam scans up 20 degrees and down to some minimum angle above the runway (which varies from airport to airport) at a rate of 40 times per second to provide basic glideslope guidance (EL1). A third non-scanning beam transmitted from a distance measuring equipment (DME) site provides ranging information. This DME beam transmits on interrogation and has an angular coverage of 120 degrees in azimuth and 20 degrees in elevation. Time reference means that the receiving equipment on the aircraft will measure the time difference between successive 'to' and 'fro' sweeps of the scanning beams to determine aircraft position relative to the runway centerline and to a preselected glide path.

The following discussion is based on the Wallops Island MLS facility but applies equally well to any MLS installation, except for specific antenna locations, which would vary from airport to airport.

The MLS coordinate system (Figure C-1) has as its origin a point 1283 feet from the stop end of runway 22 at the Wallops facility. It is located on runway centerline at a height of 41 feet above MSL. The DME is located 6 feet in front of and 62 feet to the right of the azimuth antenna, which is at the MLS origin. The EL1 site is offset to the left of the runway centerline by 400 feet (YEL1 = -400 ft in the MLS coordinate frame) with an X coordinate of 9218 feet.

Using the computation of the aircraft center of gravity (X, Y, Z) in the MLS coordinate system, the MLS / Flight Controls software will track the defined glideslope shown in Figure C-2.

Figure C-3 illustrates the MLS coordinate system with its origin established at the azimuth transmitter site. The positive X-axis is defined by the line of zero azimuth. It is assumed that the azimuth antenna system has been aligned with the physical runway centerline, i.e., bore sighted, although provision has been made to operate with an MLS system which has the AZ beam parallel to the runway but displaced to one side of the center-line. The positive Z direction is defined vertically up and the positive Y direction is defined as shown to complete the right-hand Cartesian coordinate system. Figure C-3 has represented the aircraft at three hypothetical positions, A, B and C. At position 'A' the aircraft is to the left of runway centerline, where the viewer is assumed to be facing in the direction of magnetic runway heading. The aircraft is at some altitude (Z) above the MLS X-Y plane. The sense of various quantities are as follows:

- o positive EL1, Azimuth and 'delta Y';
- o positive Rdme X and Z displacement;
- o negative Y displacement.

Note that the 'delta Y' (DELTY) signal derived from the MLS processing is equivalent to the gain programmed localizer deviation signal ETAFT, derived from the ILS localizer deviation angle (LOCDEV). As the sense of ETAFT is 'positive to cause a fly-right command', it is also required that DELTY be defined with positive sense for fly-right command, i.e, when the aircraft is to the left of runway centerline.

At position 'B', the aircraft is assumed to be to the right of runway centerline, therefore DELTY is negative to cause a fly left command from the lateral control law. Note also that azimuth is negative at position 'B', while Y(mls) is positive.

Although not obvious from Figure C-3, Z may be either positive or negative. Likewise, X could be either positive or negative (assuming ground equipment with a 'back azimuth' capability) but the present software does not support this.

The planform view of the aircraft, with aircraft heading equal to runway heading, is shown at position 'C'. This illustration is presented to clarify velocities and accelerations developed relative to aircraft body axes, versus velocities and accelerations expressed in MLS coordinates. Two factors are responsible for some confusion over these items:

- o In the normal landing attitude the X-body axis lies in a direction approximately 180 degrees to the X-axis of the inertial MLS reference frame.
- o In the normal landing attitude the Z-body axis lies in a direction approximately 180 degrees to the Z-axis of the MLS reference frame.

These factors result in the following:

$$\begin{aligned} X(\text{mls}) &= -X(\text{body}) \\ Y(\text{mls}) &= Y(\text{body}) \\ Z(\text{mls}) &= -Z(\text{body}) = h \end{aligned}$$

The parameters XDME, YDME and ZE1G are simply defined with respect to the MLS coordinate system: if the DME transmitter is located to the left/right of the azimuth transmitter, the YDME parameter will have negative/positive sense; if the EL1 antenna is located above/below the MLS X-Y plane, the ZE1G parameter will have positive/negative sense. Table C-1 lists several of the frequently used parameters in the MLS software and their respective polarities.

<u>ITEM</u>	<u>LATITUDE</u>	<u>LONGITUDE</u>	<u>h_{MSL}</u>
Az	37.923962	-75.473675	41.1
DME	37.924066	-75.473845	49.7
EL1	37.944799	-75.455473	34.7
TP6	37.926944	-75.471304	34.2
TP12	37.929268	-75.469457	34.8
TP74	37.946930	-75.455418	32.6
TP75	37.947274	-75.455145	32.2

True Heading Runway 22 = 212.193°

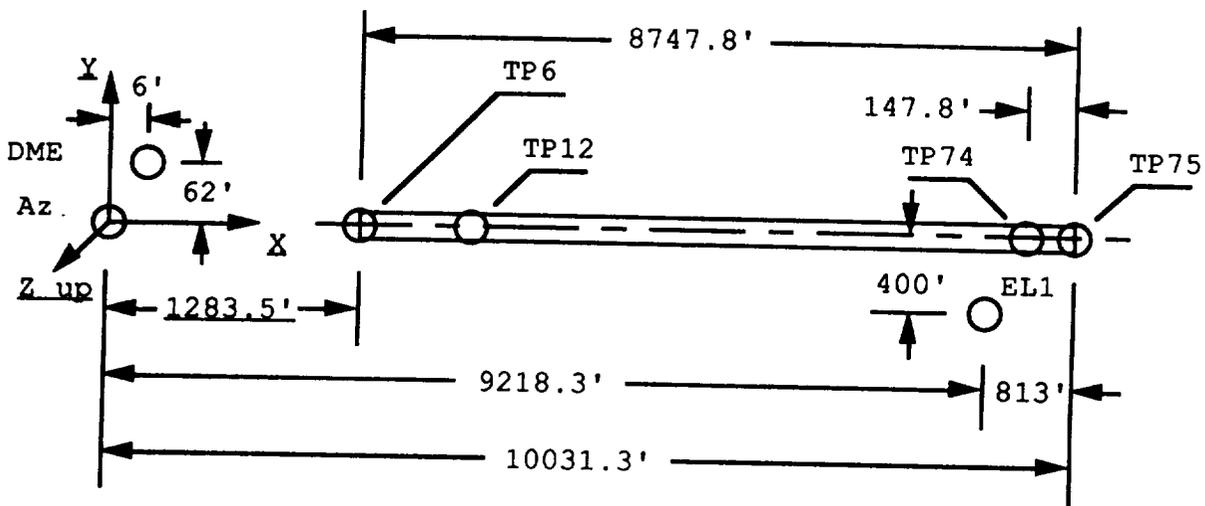


FIGURE C-1. WALLOPS FLIGHT CENTER MLS Antenna and Test Point Locations on Runway 22

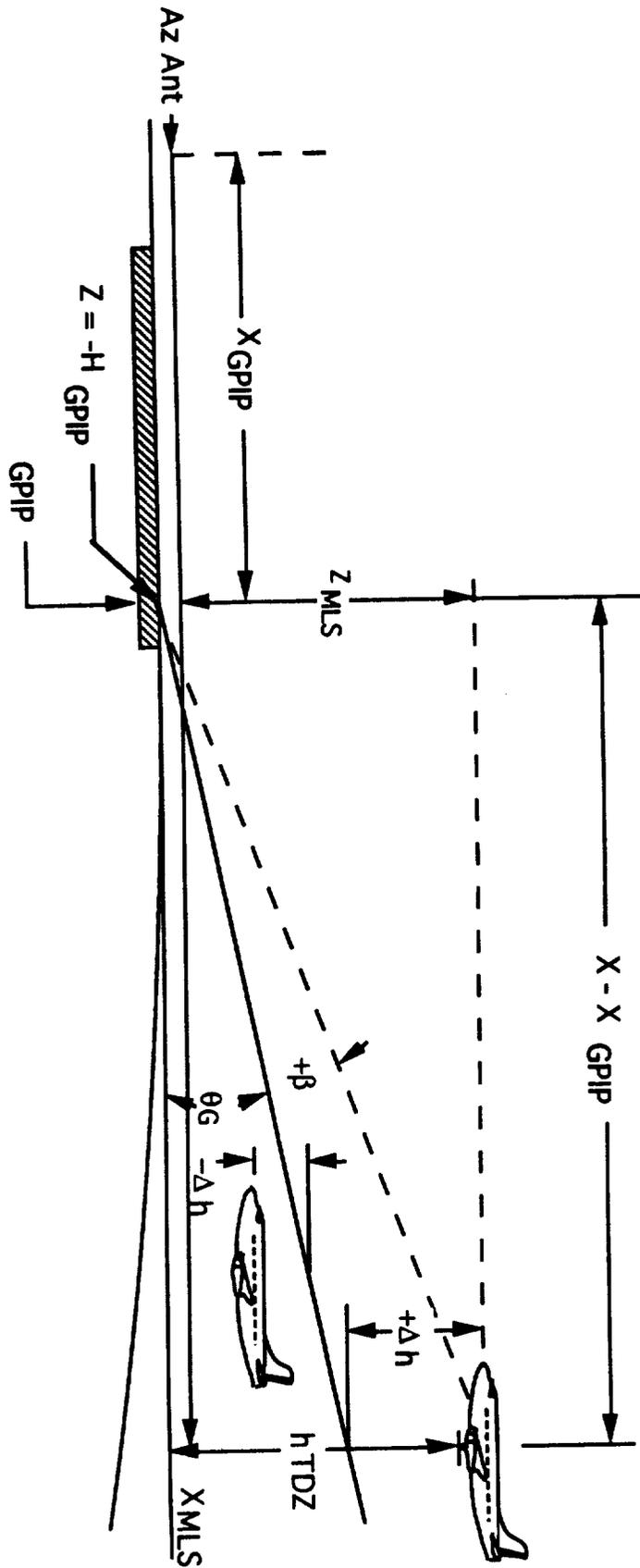


Figure C-2 Glideslope Deviation Calculations

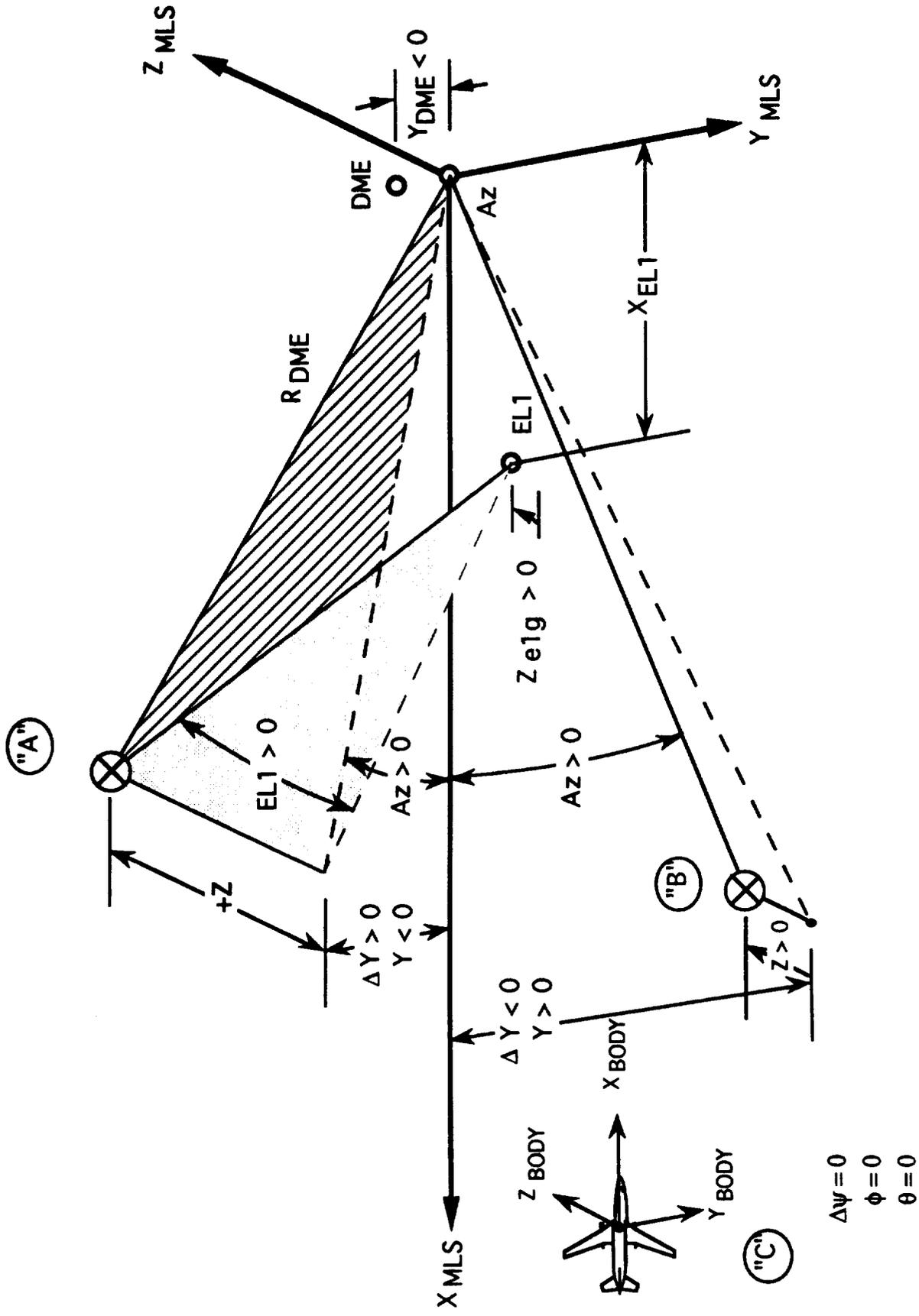


Figure C-3 Localizer Deviation Calculations

TABLE C-1 MLS PARAMETER INFORMATION

VECTORS:

<u>Vector name</u>	<u>Computed by</u>	<u>Contents</u>
ACC	CFILT	XDD, YDD, ZDD
ACCHAT	CFILT	XDDH, YDDH, ZDDH
BIASBA	CFILT	(est of bias on BMAcc)
BIASIR	CFILT	(est of bias on IRS Acc)
BMAcc	IOFLL	Ax, Ay, Fn
EX	CFILT	{POS_CG} - {XHtp}
MLSRW	IOFLL	R, AZ, EL1
MLSS_P	PFILT	Rhat, Azhat, EL1hat
MLSS_PRED	PRINV	Rp, AZp, EL1p
OTLERR	CNTRM	{MLSRW} - {MLSS_PRED}
POS_CG	XYZIN	X, Y, Z (Compt'd)
POSHAT	CFILT	Xhat, Yhat, Zhat
VELHAT	CFILT	XDH, YDH, ZDH
XHTP	CFILT	(Updated POSHAT)

POLARITY INFORMATION:

<u>Parameter</u>	<u>Polarity</u>
R, Rhat, Rp	always positive
Az, Azhat, Azp	+ left of Rwy C.L.
El1, El1hat, El1hat	+ above phase center
X, Xhat, XGPIP	+ from Az antenna to threshold
Y, Yhat, YPROF	+ right of Rwy centerline
Z, Zhat,	+ above MLS X-Y Plane
XDH, YDH, ZDH	+ on increasing X, Y, Z
XDDH, YDDH, ZDDH	+ on increasing XDH, YDH, ZDH
HGPIP	+ when GPIP below MLS X-Y plane
ETAH	+ above glideslope
BETAH	+ left of Rwy centerline

MCONF BIT CONFIGURATION:

<u>BIT</u>	<u>BOOLEAN SET</u>	<u>USE</u>
15	MLSC	Enable MLS computations
14	RLMLS	Use MLS data in Flight Controls
13	(EL2F)	Update IDD on first pass of MLSMOD
12	BMAFLG	Use body mtd vs. IRS accel's in CFILT
11	SIMILS	Simulate ILS signals from raw MLS
10	VGS_FLG	Use VN and VE vs. prefilter velocities velocities to initialize XDH, YDH
9	MSW6	Use MLS ZDDH vs. Hddf in VCWS, Auto, and glidescope tracking
8	MSW1	Use MLS derived values in flare control law
7-0	--	Unused by MLS

Appendix D - 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 preceded 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 off 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.

<u>FORMAT</u>	<u>LENGTH</u>	<u>DESCRIPTION</u>
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

<u>COMMAND</u>	<u>RESULT</u>
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 E - 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
```

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix F - System Global Variables

<u>ITEM</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>SET BY</u>	<u>RESCOM</u>	<u>TYP</u>
A429SW	ARINC 429 Xmit Switch	p.d.	GPSPKT/OUTIO	OUTCOM	I*2
ABACC	Along Body Accel	fps2	SPDCMD	FCCOM	R*4
ABLOFF	Left Air Bleed OFF discrete	Bool	IOFLL	INPCOM	L*1
ABROFF	Right Air Bleed OFF discrete	Bool	IOFLL	INPCOM	L*1
ACC	MLS Accelerations (X/Y/Z)	fps2	MLSEX	NAVCOM	R*4
ACCB	Accel vector from BM Accel's	fps2	ACCPRC	NAVCOM	R*4
ACCHAT	MLS accel'n estimate vector	fps2	CFILT	NAVCOM	R*4
ACCVLD	ATK/XTK Accelerations valid	Bool	IOFLL	INPCOM	L*1
ACTCNT	# of wpts in Active buffer	cnt	EXECUTE	DISNAV	I*2
ACWSS	ACWS Select	Bool	DISFD	FCCOM	L*1
ACWSE	Attitude CWS Engage: FLAGS(4)	Bool	MLOG	OUTCOM	L*1
ADCER	ADC Serial bus error counter	cnt	IOFLL	INPCOM	I*2
ADMG	Arc Dist Made Good in turn	ft	TGUID	NAVCOM	R*4
ADVAL	Air Data VALid	Bool	DISFD	FCCOM	L*1
AEE	AFCS Engaged	Bool	MLOG	OUTCOM	L*1
AFCSS	AFCS engage Select	Bool	IOFLL	INPCOM	L*1
AFCSV	Advanced F/C System valid	Bool	IOFLL	INPCOM	L*1
AFDTH	AFD Throtl hdl pos	deg	IOFLL	INPCOM	R*4
AFTLIM	A/T Position Cmd lower limit	deg	ATHCL	FCCOM	R*4
AGCSS	AGCS Select	Bool	IOFLL	INPCOM	L*1
AILCMD	Aileron Command	deg	OUT10M/PREFLT	OUTCOM	R*4
AILCMP	AILCMD - P contribution	deg	LATRL	OUTCOM	R*4
AIRPTS	Selected Airpt def array(2,3) adrs	CDUEXC/DEPARR/ +++ EXECUTE/LINK/ROUTE/RTE_INTC/XLAT RTE		DISNAV	I*4
		<u>Origin</u>	<u>Prov</u>	<u>Dest</u>	<u>Dest.</u>
	Airfield:	(1,1)	(1,2)	(1,3)	
	Runway:	(2,1)	(2,2)	(2,3)	
ALB	Body Alpha (angle of attack)	deg	VERCMD	DISNAV	R*4
ALCBA	2D (Adv Lat Cmd) Bank Angle	deg	HVGUID	NAVCOM	R*4
ALCFLG	Advanced Lat Cmd bank FLaG	Bool	HVGUID	NAVCOM	L*1
ALFAV	alpha vane	deg	IOFLL	INPCOM	R*4
ALRTL	Aileron rate limit	deg/10ms	COMMON	OUTCOM	R*4
ALT	Inert. smoothed baro ALTitude	ft	HNAVB	INPCOM	R*4
ALTARM	Altitude mode Armed discrete	Bool	MLOG/MSPLGC	DISNAV	L*1
ALTATT	Altitude Attained discrete	Bool	MSPLGC/IOFLL	DISNAV	L*1
ALTCOR	Baro Corrected Altitude (ALT)	ft	HNSWIT/DATSEL	DISNAV	R*4
ALTDIF	(IDDALT - MLSALT)	ft	MLSEX	NAVCOM	R*4
ALTHLD	VCWS Altitude Hold discrete	Bool	PAFD	FCCOM	L*1
ALTOMP	Altitude to Mode Panel	p.d.	MSPRO	OUTCOM	I*4
ALTSEL	Alt Hold mode discrete	Bool	MSPLGC	DISNAV	L*1
ALTSUM	MSP ALT Summer value	ft	MSPLGC/VERCMD	DISNAV	R*4
ALV1	Left Alfa Vane	deg	IOFLL	INPCOM	R*4
ALV2	Right Alfa Vane	deg	IOFLL	INPCOM	R*4
ALVDT	Aileron pos	deg	IOFLL	INPCOM	R*4
ALW	Computed wing alpha	deg	VERCMD	FCCOM	R*4
ALWMARG	Wing ALFA Margin	deg	VERCMD	FCCOM	R*4
AMG	Angle Made Good	deg	HVGUID	NAVCOM	R*4
AMG1	Angle Made Good - Timebox	ft	TGUID	NAVCOM	R*4
ANTLAT	Latitude of LOC/MLS_AZ Antenna	deg	EXECUTE	NAVCOM	R*4

<u>ITEM</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>SET BY</u>	<u>RESCOM</u>	<u>TYP</u>
ANTLON	Longitud of LOC/MLS AZ Antenna	deg	EXECUTE	NAVCOM	R*4
ANTSEL	MLS Antenna select Index	ndx	(VIEW)	NAVCOM	R*4
APCDG	Auto Throtl Pos Cmd	deg	ATHCL	OUTCOM	R*4
APCPRM	Auto throttle rate command	dps	ATHCL	FCCOM	R*4
ASBV	ADIRS ADC Bus Valid	Bool	IOFLL	INPCOM	L*1
ASTOMP	Air Speed to Mode Panel	p.d.	MSPRO	OUTCOM	I*4
ASTP	Auto Stab Trim Pot	deg	IOFLL	INPCOM	R*4
ATCMD	Auto Throttle Command	fps2	SPDCMD	FCCOM	R*4
ATDC	AutoThrottle Disengage	Bool	IOFLL	INPCOM	L*1
ATE	Auto Throttle Engage	Bool	ATHCL	OUTCOM	L*1
ATFDBK	Auto Throttle FeedBack	Bool	IOFLL	INPCOM	L*1
ATHFF	Gamma cmd to Throt Fwd gain	()	COMMON	FCCOM	R*4
ATHSEL	AUTO Throtl options	p.d.	VIEW	FCCOM	I*2
ATKINS	IRS Along track accel'n	fps2	IOFLL	INPCOM	R*4
ATNAV2	Auto Tune NAV2	Bool	PROGRESS	NAVCOM	L*1
ATNAV3	Auto Tune NAV3	Bool	PROGRESS	NAVCOM	L*1
ATRIM	AFD Aileron Trim pot	deg	IOFLL	INPCOM	R*4
ATT_HOLD	ACWS/VCWS Att./Radius Hold	Bool	LATRL	FCCOM	L*1
ATT_SYNC	ACWS/VCWS Roll commanded	Bool	LATRL	FCCOM	L*1
ATUNE2	DME #2 Freq Tune (2X5)	p.d.	PROGRESS/TUNPTH	OUTCOM	I*2
ATUNE3	DME #3 Freq Tune (2X5)	p.d.	PROGRESS/TUNXTK	OUTCOM	I*2
AUTOE	Auto Engage: FLAGS (6)	Bool	MLOG	OUTCOM	L*1
AUTOS	AUTO Select	Bool	DISFD	FCCOM	L*1
AZ	MLS Azimuth beam input	deg	IOFLL	INPCOM	R*4
AZ_BRG	MLS Az antenna phase ctr brg	deg	RSCON	NAVCOM	R*4
BACMD	Bank Angle Command	deg	LATCMD	FCCOM	R*4
BARSET	Barometric ref Setting	in/hg	CDUEXC	DISNAV	R*4
BARSFT	Baro set in feet	ft	CDUEXC	NAVCOM	R*4
BCFLAG	Be Careful guid alrt Flg (2D)	Bool	LATCMD	FCCOM	L*1
BDCOL	AFD Brolley Col pos	in	IOFLL	INPCOM	R*4
BDWHL	AFD Brolley Whl pos	deg	IOFLL	INPCOM	R*4
BETAH	MLS Beta Hat	deg	DSPOT	DISNAV	R*4
BETA V	Sideslip [beta] Vane	deg	IOFLL	INPCOM	R*4
BIASBA	B/M Accel's bias vector	fps2	MLSEX	NAVCOM	R*4
BMACC	Debiased B/M accel'n vector	fps2	ACCPRC	NAVCOM	R*4
BMACIN	B/M Accel'meter input vector	fps2	IN10M	INPCOM	R*4
BMAFLG	B/M Accel'meter (in use) Flag	Bool	MLSEX	NAVCOM	L*1
BRGLS	Brg to LOC shack	deg	HNAVSL	BCKCOM	R*4
BSET0	Std Atmosphere press. (29.92)in_hg		(constant)	CON	R*4
BUTNS	Overlay of MSP buttons	Bool(8)	MSPLGC	FCCOM	I*4
CAS	Calibrated Airspeed	kts	IOFLL	INPCOM	R*4
CDME2	Computed DME2 range	nm	HNAVSL	NAVCOM	R*4
CDME3	Computed DME3 range	nm	HNAVSL	NAVCOM	R*4
CDUCNT	Current byte cnt in CDU o/p buffer		CDUEXC	CDUCOM	I*2
CDU_INIT	Initialize CDU on start up	Bool	SLOW/CDUEXC	CDUCOM	L*1
CDU_MODE	Stages of msg composition mode ()		CDUEXC/LINK	CDUCOM	I*2
CFRÜN	MLS CFILT Run counter	cnt	MLSEX	NAVCOM	I*2
CFXCC	MLS Exceednce (Ex) error cnts	cnt	MLSEX	NAVCOM	I*2
CG	A/C Center of Gravity	%MAC	PFINIT	DISNAV	R*4

<u>ITEM</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>SET BY</u>	<u>RESCOM</u>	<u>TYP</u>
CINDEX	PGA Cost Index	()	PFINIT	CDUCOM	R*4
CLAT	Cos(latitude)	()	HNAVFS	NAVCOM	R*4
CLON	Cos(longitude)	()	HNAVFS	NAVCOM	R*4
CMPALT	ALT knob input	()	IOFLL	INPCOM	I*2
CMPFPA	FPA knob input	()	IOFLL	INPCOM	I*2
CMPIAS	IAS knob input	()	IOFLL	INPCOM	I*2
CMPSMD	CMP Discrete inputs	()	IOFLL	INPCOM	I*2
CMPTKA	TKA knob input	()	IOFLL	INPCOM	I*2
COLDST	System startup (Cold start)	Bool	FMFAST/FCFAST/ SLOW	NAVCOM	L*1
	+++			INPCOM	R*4
COMPASS	Ship's Mag Heading	deg	IOFLL	NAVCOM	R*4
COSAZB	Cos(AZ BRG)	()	RSCON	NAVCOM	R*4
COSRH	Cos(RWYHDG)	()	EXECUTE	DISNAV	R*4
COSTH	Cos(HDGTRU)	()	HNAVFS	NAVCOM	R*4
CRESET	Comp Reset select	Bool	IOFLL	INPCOM	L*1
CROLL	Cos(ROLL)	()	ACCPRC	NAVCOM	R*4
CRSET	Comp ReSET discrete	Bool	DISFD/MLOG/ATHCL	FCCOM	L*1
CRZALT	Selected Cruise Altitude	ft	EXECUTE/RTE	CDUCOM	R*4
CTHET	Cos(PITCH) (theta)	()	ACCPRC	NAVCOM	R*4
CTKE	cos(TKE)	()	HVGUID/LATCMD	NAVCOM	R*4
CTRF	Plan mode map center defined	Bool	CDUEXC	CDUCOM	L*1
CVIU TO	CVIU TimeOut count	cnt	IOFLL	INPCOM	I*2
CVOR $\bar{2}$	Computed VOR $\bar{2}$ bearing	deg	HNAVSL	NAVCOM	R*4
D2D	MSP HORPTH Select pressed	Bool	MSPLGC	FCCOM	L*1
D3D	MSP VERPTH Select pressed	Bool	MSPLGC	FCCOM	L*1
D4D	MSP TIMPTH Select pressed	Bool	MSPLGC	FCCOM	L*1
DACWS	MSP ACWS/VCWS/AUTO/LAND prs'd	Bool	MSPLGC	FCCOM	L*1
DALSEL	MSP Alt Select pressed	Bool	MSPLGC	FCCOM	L*1
DASCAL	DAS Calibration flag	Bool	(VIEW)	RECCOM	L*2
DECMD	Elevator Command	deg	OUT10M/PRFLT	OUTCOM	R*4
DECRB	Decrab Engage: FLAGS(8)	Bool	MLOG	OUTCOM	L*1
DELAY	Mode engage Delay counter	()	MLOG	FCCOM	I*2
DELCAS	AutoThrottle CAS error	kts	SPDCMD	FCCOM	R*4
DCOL	Column position input	in	IOFLL	INPCOM	R*4
DCOLF	Gained & Filtrd DCOL	in	ELEVP	FCCOM	R*4
DECMD	Delta elevator command (total)	deg	OUT10M	OUTCOM	R*4
DECMQ	DECMD - Q contribution	deg	ELEVP	OUTCOM	R*4
DECRB	Decrab mode discrete	Bool	MLOG	OUTCOM	L*1
DELALT	Altitude Select error	ft	VERCMD	FCCOM	R*4
DELTAT	Major frame cycle time (.05)	sec	(constant)	CON	R*4
DELTH	MLS Altitude Deviation (LAND)	ft	VERCMD	FCCOM	R*4
DELTKA	Track Angle Error select	deg	LATCMD	FCCOM	R*4
DELTY	Delta Y (MLS)	ft	LATCMD	FCCOM	R*4
DEPOS	Elevator servo position	deg	IOFLL	INPCOM	R*4
DFPSEL	MSP FPA Select pressed	Bool	MSPLGC	FCCOM	L*1
DFTANG	Selected drift angle	deg	HNAVFS	DISNAV	R*4
DFVWS	DFVLR wind model (speed)	kts	BLOW	BCKCOM	R*4
DFVWD	DFVLR wind model (dir)	deg	BLOW	BCKCOM	R*4
DIASEL	MSP IAS Select pressed	Bool	MSPLGC	FCCOM	L*1
DIFMOD	GPS Differential Mode	Bool	IOFLL	INPCOM	L*1

<u>ITEM</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>SET BY</u>	<u>RESCOM</u>	<u>TYP</u>	
DISCW1	Simplex/triplex discrete inpt	p.d.	IOFLL	INPCOM	I*2	
DISCW2	Simplex/triplex discrete inpt	p.d.	IOFLL	INPCOM	I*2	
DISCW3	Simplex/triplex discrete inpt	p.d.	IOFLL	INPCOM	I*2	
DISOUT	PACKED DISCRETE for output	p.d.	DASOT	RECCOM	I*2	
DISPST	Displays Status	word	IOFLL	INPCOM	I*2	
DSPST2	Displays Status	word 2	()	IOFLL	INPCOM	I*2
DSTOMP	Lamp Outputs to Mode Panel	p.d.	IOFLL	INPCOM	I*2	
DLALTM	HYBALT - MLSALT	ft	MSPRO	OUTCOM	I*4	
DLATFT	Feet per degree of latitude	fpd	GPSPRC	RECCOM	R*4	
DLLATA	GPSLAT - IDDLAT	ft	ERAD	DISNAV	R*4	
DLLATM	HYBLAT - MLSLAT	ft	GPSPRC	RECCOM	R*4	
DLLONA	GPSLON - IDDLON	ft	GPSPRC	RECCOM	R*4	
DLLONM	HYBLON - MLSLON	ft	GPSPRC	RECCOM	R*4	
DLONFT	Feet per degree of longitude	fpd	GPSPRC	RECCOM	R*4	
DLPSI	Yaw relative to runway	deg	ERAD	DISNAV	R*4	
DLTALT	HYBALT - IDDLAT	ft	DATSEL/NAVIG	FCCOM	R*4	
DLTLAT	HYBLAT - IDDLAT	ft	GPSPRC	RECCOM	R*4	
DLTLON	HYBLON - IDDLON	ft	GPSPRC	RECCOM	R*4	
DME2FQ	DME #2 Freq selected	ft	GPSPRC	RECCOM	R*4	
DME2VD	DME #2 Valid discrete	(2X5) Bool	IOFLL	INPCOM	I*2	
DME3FQ	DME #3 Freq selected	(2X5) Bool	IOFLL	INPCOM	L*1	
DME3VD	DME #3 Valid discrete	Bool	IOFLL	INPCOM	I*2	
DMG	Separation reference Distance	ft	OFL	INPCOM	L*1	
DPE	Delta Position East	nm	HVGUID/TGUID	NAVCOM	R*4	
DPERRF	DPN/DPE too large flag	Bool	HNAVSL/HNAVB	NAVCOM	R*4	
DPN	Delta Position North	nm	HNAVSL	BCKCOM	L*1	
DROLL	ROLL for sky pointer	deg	HNAVSL/HNAVB	NAVCOM	R*4	
DRPOS	Rudder servo Pos	deg	LATRL	DISNAV	R*4	
DSBV	DTU Serial Bus Valid (MLS)	Bool	IOFLL	INPCOM	R*4	
DSPLF	Failure DISPLAYed array(9)	Bool	IOFLL	INPCOM	L*1	
DSRTK	Desired Track (2D)	deg	MLOG/ATHCL	FCCOM	L*1	
DSTAT	Discrete Status array(18)	p.d.	HVGUID	DISNAV	R*4	
DTC_IN	Raw data input bfr (frm DATAC)	--	DISFD	FCCOM	I*2	
DTC_NRDY	DATAC Not Ready count	cnt	HDL	DTCCOM	I*2	
DTC_OUT	Raw data output bfr (to DATAC)	--	IOFLL	INPCOM	I*2	
DTC_TO	DATAC TimeOut count	cnt	HDL/OUT10M/OUTIO	DTCCOM	I*2	
DTG	Abeam pt Dist to 'TO' wpt	ft	IOFLL	INPCOM	I*2	
DTKSEL	MSP TKA Select pressed	Bool	HVGUID	NAVCOM	R*4	
DTOGO	Distance to mid-turn	ft	MSPLGC	FCCOM	L*1	
DTOR	Conversion factor (pi/180)	rpd	HVGUID	DISNAV	R*4	
DTOTL	Progress Distance of A/P	ft	(constant)	CON	R*4	
DVE	Delta Vel. East (IDD filter)	kts	HVGUID	NAVCOM	R*4	
DVN	Delta Vel. North (IDD filter)	kts	HNAVFS/HNAVB	NAVCOM	R*4	
DWHL	Selected Whl input	lbs/deg	HNAVFS/HNAVB	NAVCOM	R*4	
DZNE	DCOL Dead zone value	in	MLOG	FCCOM	R*4	
			MLOG	FCCOM	R*4	
EL1	MLS Elevation beam input	deg	IOFLL	INPCOM	R*4	
EL1_DEP	MLS Elevation antenna bias	deg	RSCON (MLSEX)	NAVCOM	R*4	
EL2F	Update IDD L/L on MLS select	Bool	MLSEX	NAVCOM	L*1	
ELLIP	Ellipticity of Earth (3.352811E-3)	(constant)	(constant)	CON	R*4	
EPR1	Engine #1 Pressure Ratio	()	IOFLL	INPCOM	R*4	

<u>ITEM</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>SET BY</u>	<u>RESCOM</u>	<u>TYP</u>
EPR2	Engine #2 Pressure Ratio	()	IOFLL	INPCOM	R*4
EPRFLG	EPR limit selection flag	ndx	EPRLIM	CDUCOM	I*2
ERESET	Error Reset select	Bool	IOFLL	INPCOM	L*1
ERINT	integral(K GAMD + K GAMER)	deg	ELEVP	FCCOM	R*4
ERSET	Error ReSET discrete	Bool	DISFD/MLOG	FCCOM	L*1
ESQ	Eccentricity Squared(6.6943801E-3)	(constant)		CON	R*4
ETAFT	Gain Progrmd LOCDEV	ft	MLOG	FCCOM	R*4
ETAH	MLS Eta Hat	deg	DSPOT	DISNAV	R*4
ETAVL	Limited ETA (LOCDEV)	deg	MLOG	FCCOM	R*4
EXHOLD	Arm Exit Hold pattern	Bool	HOLD	CDUCOM	L*1
EX	CFILT Error terms - EX,EY,EZ	ft	MLSEX	NAVCOM	R*4
F0G	DME #3 Fail flags	p.d.	HNAVSL	BCKCOM	I*2
F2G	DME #2 Fail flags	p.d.	HNAVSL	BCKCOM	I*2
F3G	VOR #2 Fail flags	p.d.	HNAVSL	BCKCOM	I*2
FADERS	FADER Switch	Bool	VERCMD	OUTCOM	L*1
FAIL2	Second FAIL array	Bool	F2CMP/MLSEX	FCCOM	L*1
FALST	System Test Panel switch input	ndx	IOFLL	INPCOM	I*2
FCCNT	Exception Count for FCFAST	cnt	FCFAST	OUTCOM	I*4
FCOL	FFD Column force input	lbs	IOFLL	INPCOM	R*4
FCOVER	Overflow Count for FCFAST	cnt	FCFAST	OUTCOM	I*4
FCFLGS	Flight Ctrls Flags to DSP	p.d.	GPSPRC/MLOG/	IOFLL	DISNAV
	+++				I*2
FEXP16	KTAU for VS(PHI) Filter	()	COMMON	FCCOM	R*4
FFDE	Fwd Flt Deck Engage: FLAGS(2)	Bool	MLOG	OUTCOM	L*1
FFDNAV2	FFD has Nav2 tuning	Bool	IOFLL	INPCOM	L*1
FFDS	Fwd Flight Deck Select	Bool	IOFLL	INPCOM	L*1
FIDENT	Failure ID Table (array of 7)	ndx	FDSTR	FCCOM	I*2
FLADM	Air Data Mode Flag	Bool	HNAVFS	NAVCOM	L*1
FLAGS	O'lay (18 bytes) of F/C modes	Bool	MLOG	OUTCOM	L*1
FLAGWD	Overlay (9 words) of 'FLAGS'	()	MLOG	OUTCOM	I*2
FLAP	FFD Flap handle pos	deg	IOFLL	INPCOM	R*4
FLARE	Flare Engage: FLAGS(9)	Bool	MLOG/ELEVP	OUTCOM	L*1
FLPNDX	Index to Flap table	ndx	IOFLL	INPCOM	I*2
FLPPF	Filtered FLPPOS	deg	VERCMD	FCCOM	R*4
FLPPOS	Flap Position	deg	IOFLL	INPCOM	R*4
FLRM	Radio Mode Navigation flag	Bool	HNAVFS	NAVCOM	L*1
FMCNT	Exception Count for FMFAST	cnt	FMFAST	OUTCOM	I*4
FMOVER	Overflow Count for FMFAST	cnt	FMFAST	OUTCOM	I*4
FPA	3D segment Flight Path Angle	deg	HVGUID	NAVCOM	R*4
FPASEL	FPA Hold mode discrete	Bool	MSPLGC	DISNAV	L*1
FPASUM	MSP FPA Summer value	deg	MSPLGC/VERCMD	DISNAV	R*4
FPTOMP	Flight Path Angle to Mode Panel	p.d.	MSPRO	OUTCOM	I*4
FRAME	Running count of 10 msec	int cnt	IOFLL	INPCOM	I*4
FRAMES	Running count of 50 msec	int cnt	IOFLL	INPCOM	I*4
FSBV	RFDIU (Fuel flow) Ser Bus Vld	Bool	IOFLL	INPCOM	L*1
FSIDX	Fail Status Index pointer	ndx	F2CMP/PANEL	FCCOM	I*2
FSTCNT	Fast loop time estimate	msec	FCFAST	FCCOM	R*4
FTFQ	Filtered Tot Fuel Qty	lbs	IOFLL	INPCOM	R*4
FTONM	Conversion factor (1/6076.1155)	(constant)		CON	R*4
FWDLIM	A/T Position Cmd Upper Limit	deg	COMMON	FCCOM	R*4

<u>ITEM</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>SET BY</u>	<u>RESCOM</u>	<u>TYP</u>
FWHL	FFD Wheel Force	lbs	IOFLL	INPCOM	R*4
GAE	Go Around Enable	Bool	MLOG	OUTCOM	L*1
GAEPR	Go Around EPR	()	EPRLMT	CDUCOM	R*4
GAMC	Commanded Gamma value	deg	ELEVP/VERCMD	DISNAV	R*4
GAMCMD	Gamma Rate command (Althold)	dps	VERCMD	FCCOM	R*4
GAMD	Gamma Dot	dps	ELEVP	FCCOM	R*4
GAMER	(PGAMC - (THETA or GAMMA))	deg	ELEVP	FCCOM	R*4
GAMLIM	Gamma/Pitch command limit	deg	VERCMD	FCCOM	R*4
GAMMA	Flight path angle	deg	HNAVFS/DATSEL	DISNAV	R*4
GAS	Go Around Select (NOT)	Bool	IOFLL	INPCOM	L*1
GDTIME	Buffer send timer	sec	AIRWAY/CDUFST/ +++ EXECUTE/LEGS	BCKCOM	I*4
GEAR	Nose Gear down	Bool	IOFLL	INPCOM	L*1
GPBTAV	GPS G/S (beta) Valid	Bool	GPSPRC	NAVCOM	L*1
GPGSDV	Gain Program'd G/S Deviation	ft	MLOG	FCCOM	R*4
GPHDOP	GPS Hor Dilution Of Precision	()	IOFLL	INPCOM	R*4
GPINIT	GPS land comp. INITIALized	BOOL	GPSPRC	NAVCOM	L*1
GPLND	GPS Land select	Bool	GPSPG	NAVCOM	L*1
GPLNDV	GPS Land valid	Bool	GPSPRC	NAVCOM	L*1
GPLOCD	Gain Program'd LOC Deviation	deg	MLOG	FCCOM	R*4
GPNAV	GPS Navigation select	Bool	GPSPG	NAVCOM	L*1
GPNAVV	GPS NAVigation Valid	Bool	GPSPRC	NAVCOM	L*1
GPSBTA	GPS derived BETAH	deg	GPSPRC	NAVCOM	R*4
GPSETA	GPS derived ETAH	deg	GPSPRC	NAVCOM	R*4
GPSHDT	GPS Hdot	fps	IOFLL	INPCOM	R*4
GPSLAT	Absolute GPS Latitude	deg	IOFLL	INPCOM	R*4
GPSLON	Absolute GPS Longitude	deg	IOFLL	INPCOM	R*4
GPSMOD	GPS sensor Mode	ndx	GPSPRC	NAVCOM	I*2
GPSSEL	GPS Select word	ndx	GPSPRC	DISNAV	I*2
GPSST1	GPS Status word 1	p.d.	IOFLL	INPCOM	I*2
GPSST2	GPS Status word 2	p.d.	IOFLL	INPCOM	I*2
GPSVE	GPS VE	kts	IOFLL	INPCOM	R*4
GPSVLD	Hybrid GPS signals valid	Bool	IOFLL	INPCOM	L*1
GPSVN	GPS VN	kts	IOFLL	INPCOM	R*4
GPS_OUT	Raw GPS output data (to DATAC)	--	OUTIO	DTCCOM	I*2
GPVDOP	GPS Vert Dilution of Precision	()	IOFLL	INPCOM	R*4
GRAV0	Nom acc of gravity (32.1739)	fps2	(constant)	CON	R*4
GRD	on GROUND	Bool	ELEVP	FCCOM	L*1
GRSST1	GP_IRS Status word 1	bits	IOFLL	INPCOM	I*2
GRSST5	GP_IRS Status word 2	bits	IOFLL	INPCOM	I*2
GRWGT	A/C initial Gross Weight	lbs	PFINIT	DISNAV	R*4
GS	Ground Speed	kts	HNAVFS/DATSEL	DISNAV	R*4
GSA	Glide Slope Angle	deg	EXECUTE/MLOG	DISNAV	R*4
GSARM	Glide Slope Armed: FLAGS(14)	Bool	MLOG	OUTCOM	L*1
GSDEV	Glide slope deviation	~deg	MLOG	FCCOM	R*4
GSE	4-D Guidance Ground Speed Error	fps	HVGUID	CDUCOM	R*4
GSENG	Glide Slope Engaged: FLAGS(15)	Bool	MLOG/ELEVP	OUTCOM	L*1
GSFPS	Ground Speed	fps	HNAVFS/DATSEL	DISNAV	R*4
GSFPS2	Ground Speed**2	f2/s2	HNAVFS/DATSEL	NAVCOM	R*4
GSINS	IRS ground speed	kts	IOFLL	INPCOM	R*4

<u>ITEM</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>SET BY</u>	<u>RESCOM</u>	<u>TYP</u>
GSTRK	Glide Slope Track: FLAGS(17)	Bool	MLOG/ELEVP	OUTCOM	L*1
GSVLD	ILS G/S Rcvr Valid	Bool	DISFD	FCCOM	L*1
GUID2D	2D Guidance possible flag +++	Bool	CDUFST/DATSEL/ EXECUTE/HVGUID/NAVIG/ROUTE	DISNAV	L*1
GUID3D	3D Guidance possible flag +++	Bool	CDUFST/EXECUTE/ HVGUID/NAVIG/ROUTE	DISNAV	L*1
GUID4D	4D guidance possible flag +++	Bool	CDUFST/EXECUTE/ HVGUID/NAVIG/ROUTE/TGUID	DISNAV	L*1
HBARO	Baro altitude	ft	IOFLL	INPCOM	R*4
HBECTR	Counts of ALT-HBARO > 50 ft	cts	HNAVB	NAVCOM	I*2
HDCF	HDOT Complementary filt'd +++	fps	HNSWIT/DATSEL/ NAVIG	DISNAV	R*4
HDD	Selected vertical accel'n	fps2	HNAVFS/DATSEL	NAVCOM	R*4
HDDFEXP	KTAU for HDDF Filter	()	COMMON	FCCOM	R*4
HDDINS	IRS vertical acceleration	fps2	IN10M	INPCOM	R*4
HDDOT	Debiased HDD from ALT filter	fps2	HNAVB/NAVIG	NAVCOM	R*4
HDGTRU	Selected true A/C heading	deg	HNAVFS/DATSEL	DISNAV	R*4
HDILS	Vert velocity from HRAD & HDD	fps	ELEVP	FCCOM	R*4
HDLCNT	Exception Count for HDL	cnt	HDL	OUTCOM	I*4
HDOT	HDOT Complementary filt'd	fps	HNAVB	NAVCOM	R*4
HDOTB	Altitude rate	fps	IOFLL	INPCOM	R*4
HDTC	Altitude rate command (3D)	fps	HVGUID	DISNAV	R*4
HER	Vertical Path error	ft	HVGUID	DISNAV	R*4
HGPIP	GPIP Height above MLS plane	ft	MLSEX	NAVCOM	R*4
HLD2D	New PTR2D after hold ptrn rmvd	ndx	EXECUTE/HOLD	CDUCOM	I*2
HLDTRK	TKASUM for TRKHLD mode	deg	LATCMD/LATRL	FCCOM	R*4
HLD_PTR	Active Hold Waypoint Pointer	ndx	EXECUTE/ROUTE	CDUCOM	I*2
HLD_WPT	Name of Holding ptrn fix Wpt +++	char	EXECUTE/HOLD/ LEGS/ROUTE	CDUCOM	C*5
HOLD	HOLD Mode select discrete	Bool	IOFLL	INPCOM	L*1
HOLDM	HOLD Mode engaged discrete	Bool	MLOG	OUTCOM	L*1
HORARM	Hor Path mode Armed discrete	Bool	MSPLGC	DISNAV	L*1
HORBLU	HOR Path blue light	Bool	MSPLGC	DISNAV	L*1
HORPTH	HOR Path mode engaged dscrete	Bool	MSPLGC	DISNAV	L*1
HRAD	Corrected radar altitude	ft	DATSEL	NAVCOM	R*4
HRV	Radio Altitude Valid	Bool	DISFD	FCCOM	L*1
HRSS	System time in hours/mins	bcd	IOFLL	INPCOM	I*2
HTDZ	Selected (MLS/HRAD) alt (AGL)	ft	MLOG	FCCOM	R*4
HYBALT	Hybrid GPS/IRS Altitude	ft	IOFLL	INPCOM	R*4
HYBHDT	Hybrid GPS/IRS Altitude rate	fps	IOFLL	INPCOM	R*4
HYBLAT	Hybrid GPS/IRS Latitude	deg	IOFLL	INPCOM	R*8
HYBLON	Hybrid GPS/IRS Longitude	deg	IOFLL	INPCOM	R*8
HYBVE	Hybrid GPS/IRS East velocity	kts	IOFLL	INPCOM	R*4
HYBVN	Hybrid GPS/IRS North velocity	kts	IOFLL	INPCOM	R*4
H_TDC	Ht of Az plane above rwy @ GPIP	ft	RSCON (MLSEX)	NAVCOM	R*4
IASARM	IAS Armed mode discrete	Bool	MSPLGC	DISNAV	L*1
IASREF	IAS Ref (min) airspeed	kts	PFINIT	NAVCOM	R*4
IASSEL	IAS SElect	Bool	MSPLGC	DISNAV	L*1
IASSUM	MSP IAS Summer value	kts	MSPLGC/SPDCMD	DISNAV	R*4

<u>ITEM</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>SET BY</u>	<u>RESCOM</u>	<u>TYP</u>
IATTV	IRS ATTitude Valid	Bool	DISFD	FCCOM	L*1
IC	IC Mode select discrete	Bool	IOFLL	INPCOM	L*1
ICM	IC Mode engaged discrete	Bool	MLOG	OUTCOM	L*1
IDDALT	Comp Filtered (HBARO/HDD) Alt	ft	HNAVB	NAVCOM	R*4
IDDATK	Selected Along Track Accel	fps2	HNAVSL/NAVIG	NAVCOM	R*4
IDDGS	GS from INS data	kts	HNAVFS	NAVCOM	R*4
IDDLAT	Inertial-Dual-DME Latitude	deg	HNAVB	NAVCOM	R*8
IDDLNC	MSW of Integer IDDLON	deg	DASOT	RECCOM	I*2
IDDLNF	LSW (LSB Approx 4 ft)	deg	DASOT	RECCOM	I*2
IDDLON	Inertial-Dual-DME Longitude	deg	HNAVB	NAVCOM	R*8
IDDLTC	MSW of Integer IDDLAT	deg	DASOT	RECCOM	I*2
IDDLTF	LSW (LSB = 180*2**24)	deg	DASOT	RECCOM	I*2
IDDXTK	Selected Across Track Accel	fps2	HNAVSL/NAVIG	NAVCOM	R*4
ILDHIG	Use hi gain for ILD/ILG	Bool	VIEW	BCKCOM	L*1
ILDPFR	Use ILD vs ILG if both valid	Bool	VIEW	NAVCOM	L*1
ILGRNG	Use cptd rng vs HRAD for ILG	Bool	VIEW	NAVCOM	L*1
ILSZON	Within ILS (guidance) Zone	Bool	HNAVSL	DISNAV	L*1
INAVV	IRS NAVigation Valid	Bool	DISFD	FCCOM	L*1
INSST	Ignore Display toggle fail	Bool	VIEW	FCCOM	I*2
IPL NR	I/P Link Not Rdy count	cnt	IOFLL	INPCOM	I*2
IRSALT	IRS corrected altitude	ft	IOFLL	INPCOM	R*4
IRSER	IRS Serial bus error counter	cnt	IOFLL	INPCOM	I*2
IRSHDOT	IRS vertical velocity	fps	IOFLL	INPCOM	R*4
IRSST1	IRS Status word 1	p.d.	IOFLL	INPCOM	I*2
IRSST2	IRS Status word 2	p.d.	IOFLL	INPCOM	I*2
IRSWD	WD from ADIRS	deg	IOFLL	INPCOM	R*4
IRSW	WS from ADIRS	kts	IOFLL	INPCOM	R*1
ISBV	ADIRS IRS Bus Valid	Bool	IOFLL	INPCOM	L*1
JUMP	Test flg for RNAV/MLS trnsxn	Bool	HNAVB/VIEW	NAVCOM	L*1
K1P	RNAV filter position gain	()	HNAVSL	NAVCOM	R*4
K2P	RNAV filter velocity gain	()	HNAVSL	NAVCOM	R*4
KAILG	Gain on AILCMD (nom=1.)	()	LATRL	OUTCOM	R*4
KAILP	Aileron pulse magnitude: 102.4*deg		(VIEW)	OUTCOM	R*4
KAILT	Aileron pulse duration : 20*sec		(VIEW)	OUTCOM	R*4
KALFA	Vert gain as f(ALFA)	()	VERCMD	FCCOM	R*4
KATRM	KTAU for Ail Trim (SYNCL)	()	COMMON	FCCOM	R*4
KAT08	Fwd loop gain for Autothrottle	()	COMMON	FCCOM	R*4
KAT16	Parallel loop gain for Autothr	()	COMMON	FCCOM	R*4
KCAS	Hor/Ver Path gain as f(CAS)	()	LATCMD	FCCOM	R*4
KCMR	TC for CAS/MACH Filter	()	COMMON	FCCOM	R*4
KCTOT	Ratio of CAS to TAS	()	SPDCMD	FCCOM	R*4
KDCALF	Speed gain as f(ALFA)	()	VERCMD	FCCOM	R*4
KDCOL	Gain on SAC Col input	()	COMMON	FCCOM	R*4
KDCRB	VCWS Aileron Crossfeed gain	()	COMMON	FCCOM	R*4
KDECP	Elevator pulse magnitude:102.4*deg		(VIEW)	OUTCOM	R*4
KDECT	Elevator pulse duration : 20*sec		(VIEW)	OUTCOM	R*4
KDWHL	Gain on SAC Whl input	()	COMMON	FCCOM	R*4
KHCAS	Overspeed degain for gamma cmd	()	VERCMD	FCCOM	R*4
KHD	Vert. Path Vel. error gain	()	VERCMD/COMMON	FCCOM	R*4

<u>ITEM</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>SET BY</u>	<u>RESCOM</u>	<u>TYP</u>
KLRG	SAC Left/Right gain	()	COMMON	FCCOM	R*4
KP	Gain on P	()	LATRL	OUTCOM	R*4
KPDTC	PDTCMD gain	()	COMMON	FCCOM	R*4
KPLIM	Upper Limit on KP	()	COMMON	FCCOM	R*4
KQ	Gain on Q	()	ELEVP	OUTCOM	R*4
KRTRM	KTAU for Rud Trim	()	COMMON	FCCOM	R*4
KT0P5S	e ^{**-} (dt/Tau) where Tau = .5	()	(constant)	CON	R*4
KT1SEC	e ^{**--} (dt/Tau) where Tau = 1.	()	(constant)	CON	R*4
KT1P5S	e ^{**--} (dt/Tau) where Tau = 1.5	()	(constant)	CON	R*4
KT2SEC	e ^{**--} (dt/Tau) where Tau = 2.	()	(constant)	CON	R*4
KT3SEC	e ^{**--} (dt/Tau) where Tau = 3.	()	(constant)	CON	R*4
KT4SEC	e ^{**--} (dt/Tau) where Tau = 4.	()	(constant)	CON	R*4
KT5SEC	e ^{**--} (dt/Tau) where Tau = 5.	()	(constant)	CON	R*4
KTAUPF	TC (KTAU) for PF1 (roll rate)	()	COMMON	FCCOM	R*4
KTDCRB	VCWS Aileron Crossfeed KTAU	()	COMMON	FCCOM	R*4
KTDROL	KTAU for DROLL	()	COMMON	FCCOM	R*4
KTKREL	TC (KTAU) for TKREL (0.2 sec)	()	(constant)	FCCOM	R*4
KTOFPS	Kts to fps Conv factor (1.6878099)		(constant)	CON	R*4
KTP25S	e ^{**--} (dt/Tau) where Tau = .25	()	(constant)	CON	R*4
KTPFL	TC for PF1 (0.05 sec)	()	LATRL	OUTCOM	R*4
KV	Airspeed gain	()	VERCMD	FCCOM	R*4
KVCAS	Speed gain on ATCMD	()	SPDCMD	FCCOM	R*4
KWHLF4	CWS Whl Feed fwd gain	()	COMMON	FCCOM	R*4
LABFLG	Operating in the Hot Bench	Bool	HNAVSL	DISNAV	L*1
LAMP	Lamp Test Select	Bool	DISFD	FCCOM	L*1
LANDS	LAND Select	Bool	DISFD	FCCOM	L*1
LANDA	Land Armed: FLAGS(11)	Bool	MLOG	OUTCOM	L*1
LANDE	Land Engage: FLAGS(7)	Bool	MLOG	OUTCOM	L*1
LANDR	Land Ready: FLAGS(16)	Bool	MLOG	OUTCOM	L*1
LAT	Selected Latitude	deg	HNSWIT	DISNAV	R*4
LATCEN	North up map Center - Latitude	deg	AIRWAY/CDUFST/ +++	LEGS/NAVPG	DISNAV R*4
LATDIF	(IDDLAT - MLSLAT)	deg	MLSEX	NAVCOM	R*4
LATINS	IRS latitude	deg	IOFLL	INPCOM	R*4
LAT_MLS	Latitude of MLS Az antenna	deg	RSCON	NAVCOM	R*4
LATSEL	LAT Steer options	p.d.	VIEW	FCCOM	I*2
LATSTR	Lateral Steering signal	deg	LATCMD	FCCOM	R*4
LBS	Lateral Beam Sensed flag	Bool	LATCMD/MLOG	FCCOM	L*1
LCFLG	Scratch copy of FCFLGS	p.d.	DSPOT/ELEVP/MLOG +++	SPDCMD/VERCMD	FCCOM I*2
LHDC	HDOT Command Limit	fps	VERCMD	FCCOM	R*4
LIGHTS	STP Lamp control word	bits	F2CMP/FDSTR/PANEL	FCCOM	I*2
LLINIT	Lat/Lon Initialized flag	Bool	HNAVES/INITUP/ +++	NAVIG	NAVCOM L*1
LMB	RWY-to-body rotation (3X3)	matrix	ACCPRC/GSPRC/ +++	NAVEEM	R*4
LNDBLU	LAND Mode blue light	Bool	MLOG	DISNAV	L*1
LNDCR	LANDS Delay Counter	()	MLOG	FCCOM	I*2
LOCA	Localizer Armed: FLAGS(12)	Bool	MLOG	OUTCOM	L*1
LOCCMD	Localizer Steering signal	deg	LATCMD	FCCOM	R*4

<u>ITEM</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>SET BY</u>	<u>RESCOM</u>	<u>TYP</u>
LOCDEV	Selected Loc Dev (ILS/MLS/GPS)	~deg	MLOG	FCCOM	R*4
LOCE	Localizer Engaged: FLAGS(13)	Bool	MLOG/LATCMD	OUTCOM	L*1
LOCVL	ETA/DELTY Variable Limit	()	MLOG/LATCMD	FCCOM	R*4
LOCFS	Localizer Freq Selected	Bool	DISFD	FCCOM	L*1
LOCVLD	Localizer Valid	Bool	DISFD	FCCOM	L*1
LON	Selected Longitude	deg	HNSWIT	DISNAV	R*4
LONCEN	North up map Center - LON	deg deg	AIRWAY/CDUFST/ +++ LEGS/NAVPG	DISNAV	R*4
LONDIF	(IDDLON - MLSLON)	deg	MLSEX	NAVCOM	R*4
LONINS	IRS longitude	deg	IOFLL	INPCOM	R*4
LON_MLS	Longitude of MLS Az antenna	deg	RSCON	NAVCOM	R*4
MACH	MACH number	()	IOFLL	INPCOM	R*4
MAGHDG	IRS Magnetic Heading	deg	IOFLL	INPCOM	R*4
MAGVAR	Magnetic variation	deg	ERAD/NAVIG	DISNAV	R*4
MANEL	Manual Electric: FLAGS(3)	Bool	MLOG	OUTCOM	L*1
MAPUPD	Map Update request	Bool	HNAVSL	DISNAV	L*1
MAXF	Max Frame count @ end of FCFAST	()	FCFAST	FCCOM	I*2
MAXSPD	Max CAS/MACH (2 items)	kts/mach	VIEW	FCCOM	R*4
MCLEPR	EPR Limit for max Climb Thrust	()	EPRLMT	CDUCOM	R*4
MCONF	MLS Configuration word	bits	VIEW	NAVCOM	I*2
MCREPR	EPR Limit for max Cruise Thrust	()	EPRLMT	CDUCOM	R*4
MCTEPR	EPR Limit for max Cont. Thrust	()	EPRLMT	CDUCOM	R*4
MDME2	Measured DME #2 distance	nm	IOFLL	INPCOM	R*4
MDME3	Measured DME #3 distance	nm	IOFLL	INPCOM	R*4
MDWARN	Mode Reversion Warning	Bool	MSPLGC	OUTCOM	L*1
MFRAME	Present Minor Frame (0 - 4)	cnt	IOFLL	INPCOM	I*2
MINS	System time in secs/tenths	bcd	IOFLL	INPCOM	I*2
MINSATH	Minimum Satellites for GPS nav	cnt	COMMON/VIEW	NAVCOM	I*2
MINSATV	Minimum Sat's for Vert guid	Bool	(VIEW)	NAVCOM	I*2
MLO	Maj Logic O/P from DISFD	p.d.	DISFD	FCCOM	I*2
MLSALT	MLS derived Altitude (MSL)	ft	HNAVML	DISNAV	R*4
MLSC	MLS Compute flag	Bool	MLSEX/HNAVB	NAVCOM	L*1
MLSER	MLS Serial bus error counter	cnt	IOFLL	INPCOM	I*2
MLSLAT	MLS derived Latitude	deg	HNAVML	DISNAV	R*4
MLSLON	MLS derived Longitude	deg	HNAVML	DISNAV	R*4
MLSM	MLS Mode selected (F/C)	Bool	MLOG	FCCOM	L*1
MLSMOD	MLS Mode selected (Nav)	Bool	HNAVFS/DATSEL	OUTCOM	L*1
MLSRW	MLS signal input ary: DME,AZ,EL --		IOFLL	INPCOM	R*4
MLSSLI	MLS Select switch (Nav pallet)	Bool	IOFLL	INPCOM	L*1
MLSSV	MLS Signal Valid array (4)	Bool	IOFLL/NAVIG	INPCOM	L*1
MLSSVC	MLS Signal Valid Ctrs(R,Az,El)	cnt	MLSEX	NAVCOM	I*2
MLSVAL	MLS solution Vali	Bool	MLSEX	NAVCOM	L*1
MLSVLD	MLS Valid (MLSVAL .and. RUNM)	Bool	HNAVFS	OUTCOM	L*1
MNAVTY	Nav type for mode word	p.d.	HNAVSL	DISNAV	I*2
MODCNT	# of wpts in Mod buf	cnt	EXECUTE/XLAT RTE	DISNAV	I*2
MODE2	Prev iteration value of MODEX	ndx	MLOG	FCCOM	I*2
MODEX	Flight MODE index (see FCCOM)	ndx	MLOG/ELEVP	FCCOM	I*2
MSBUF	Format'd Failure Msg (38 byte)	ASCII	FMTMG	FCCOM	I*2
MSGST	Status of next Message to STP	Bool	FMTMG	FCCOM	L*1
MSPER	MSP Serial bus error counter	cnt	IOFLL	INPCOM	I*2

<u>ITEM</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>SET BY</u>	<u>RESCOM</u>	<u>TYP</u>
MSW1	MLS Switch 1	Bool	MLOG	FCCOM	L*1
MSW6	MLS Switch 6	Bool	MLOG	FCCOM	L*1
MSWIT	STP Mode Switch position	ndx	FDSTR	FCCOM	I*2
MS10ML	# Missed 10 Msec int	cnt	IOFLL	INPCOM	I*2
MS50ML	# Missed 50 Msec int	cnt	IOFLL	INPCOM	I*2
MURWY	Origin Rwy friction coef	()	TKOFF	DISNAV	R*4
MVINS	Computed mag var from ADIRS	deg	BLOW	BCKCOM	R*4
MVOR2	Measured VOR #2 bearing	deg	IOFLL	INPCOM	R*4
MXEPR	MaX permissable EPR	()	EPRLMT	DISNAV	R*4
MXHDOP	Max allowable HDOP for GPS nav	Bool	COMMON/VIEW	NAVCOM	R*4
MXVDOP	Max VDOP value for guidance	()	(VIEW)	NAVCOM	R*4
NAV64K	GS > 64 knots flag	Bool	HNAVFS/DATSEL	DISNAV	L*1
NAVCTR	Iter's of valid nav data inpt	cnts	HNAVFS/HNAVSL	NAVCOM	I*2
NAVFLG	GS > 4 kts: Integrate Velocity	Bool	HNAVFS/NAVIG	NAVCOM	L*1
NAVMOD	Navigation mode GPS/MLS/IDD..	ndx	HNSWIT	NAVCOM	I*2
NAVTyp	Navigation mode Type (2 words)	ASCII	HNAVSL	DISNAV	I*2
NAVVL	Navigation solution valid	Bool	HNAVFS	NAVCOM	L*1
NCI01	ATHCL Windshear filter internal	fps	ATHCL	FCCOM	R*4
NCI02	ATHCL Windshear filter internal	fps2	ATHCL	FCCOM	R*4
NCI03	ATHCL APC Integrator	deg	ATHCL	FCCOM	R*4
NCL1	ATHCL Accel damping term	fps2	ATHCL	FCCOM	R*4
NCL3F	Filtered NCL2	dps	ATHCL	FCCOM	R*4
NCL2	Gamma cmd feed fwd to APCDG	deg	VERCMD/ELEVP	FCCOM	R*4
NCUVAL	Nav Computations Valid	Bool	HNAVFS	OUTCOM	L*1
NMTFT	Conversion factor (6076.1155)	(constant)	(constant)	CON	R*4
NODAS	DASOT Disable flag	Bool	DSTAR	RECCOM	L*2
NOSNAP	SNAP Disable flag	Bool	DSTAR	RECCOM	L*2
NVAD2A	Pointer to NAV2 Auto	adrs	PROGRESS/TUNPTH	DISNAV	I*4
NVAD2B	Pointer to next NAV2 Auto	adrs	PROGRESS	DISNAV	I*4
NVAD3A	Pointer to NAV3 Auto	adrs	PROGRESS/TUNXTK	DISNAV	I*4
NVVMOD	Vert Nav mode (GPS/MLS/IDD..)	ndx	HNSWIT	NAVCOM	I*2
OLDLAT	Select old Lat capture	Bool	VIEW	FCCOM	L*1
OLDWD	Old wind model (dir)	deg	BLOW	BCKCOM	R*4
OLDWS	Old wind model (speed)	kts	BLOW	BCKCOM	R*4
ONCRS	(Loc) On Course: FLAGS(18)	Bool	MLOG/LATCMD	OUTCOM	L*1
OPTALT	PGA Optimum cruise Altitude	ft	COMMON	CDUCOM	R*4
OTLERR	MLS Sgnl Outlier Errors (R/Az/El)		MLSEX	NAVCOM	R*4
P	Roll rate gyro input	dps	IN10M	INPCOM	R*4
PDCOL	Deadzoned Column position input	in	ELEVP	DISNAV	R*4
PDTCMD	Roll Rate Command	dps	LATRL	FCCOM	R*4
PEDAL	AFD Rud Ped + Trim	deg	IOFLL	INPCOM	R*4
PF1	Filtered P	dps	LATRL	OUTCOM	R*4
PFPA	3D Progammed Flight Path Angle	deg	HVGUID	NAVCOM	R*4
PGAMC	Local GAMC for PAFD	deg	ELEVP	FCCOM	R*4
PHICMD	Bank angle command	deg	LATRL	FCCOM	R*4
PHIERR	(PHICMD - ROLL)	deg	LATRL	FCCOM	F*4
PHISYM	Simulated A/P roll command	deg	NAVIG	NAVCOM	R*4
PITCH	Pitch attitude (theta)	deg	IOFLL	INPCOM	R*4

<u>ITEM</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>SET BY</u>	<u>RESCOM</u>	<u>TYP</u>
PMSWIT	Previous value of MSWIT	ndx	FDSTR	FCCOM	I*2
PNAV64	Prev value of NAV64K	Bool	SPDCMD	FCCOM	L*1
PMODE	Prov/Mod/Act flight plan mode	ndx	EXECUTE/ROUTE	DISNAV	I*2
POSHAT	MLS position estimate vector	ft	CFILT	NAVCOM	R*4
PRENG	Pre-engage: FLAGS(1)	Bool	MLOG	OUTCOM	L*1
PSTALT	Pre-selected Altitude discrete	Bool	MSPLGC	DISNAV	L*1
PSTFPA	FPA preselect discrete	Bool	MSPLGC	DISNAV	L*1
PSTTKA	Track angle preslct discrete	Bool	MSPLGC	DISNAV	L*1
PTR2D	2D Reference PTR	ndx	EXECUTE/HVGUID/ +++ NAVIG	NAVCOM	I*2
PTR3D	3D Reference PTR	ndx	HVGUID	NAVCOM	I*2
PTR4D	4D Reference PTR	ndx	HVGUID	NAVCOM	I*2
PTR4D1	Time-box ref Ptr (4D)	ndx	TGUID	NAVCOM	I*2
PVHAT	Array(9) overlayng XHAT...ZDDH	--	HNSWIT	NAVCOM	R*4
Q	Body pitch rate	dps	IOFLL	INPCOM	R*4
QFB1	Filtered (washed out) Q	deg	ELEVP	FCCOM	R*4
QX	Bias on Q	dps	ELEVP	OUTCOM	R*4
R	Yaw rate gyro input	dps	IN10M	INPCOM	R*4
R0MLS	MLS Range bias	ft	RSCON (MLSEX)	NAVCOM	R*4
RADALT	Raw radar altitude	ft	IOFLL	INPCOM	R*4
RADEF	Best fit Earth rad of curvature	ft	ERAD	NAVCOM	R*4
RADHLD	VCWS Radius Hold enabled	Bool	LATRL	FCCOM	L*1
RADIUS	Nom radius of Earth (3443.9186)	ft	(constant)	CON	R*4
RALC	Dist before turn to apply ALC	ft	HVGUID	NAVCOM	R*4
RDME	MLS Range input	ft	IOFLL	INPCOM	R*4
RECWD	Recording Select Word in-use	p.d.	DSTAR/DASOT	RECCOM	I*2
RECWD1	Normal Recording Select Word	p.d.	(VIEW)	RECCOM	I*2
RECWD2	Alternate Recording Slct Word	p.d.	(VIEW)	RECCOM	I*2
RGSDEV	ILS Glideslope Deviation	deg	IOFLL	INPCOM	R*4
RLMLS	MLSM Enable (vs MLS NAV only)	Bool	MLSEX	NAVCOM	L*1
RLOCDV	ILS Localizer Deviation	deg	IOFLL	INPCOM	R*4
RLOUT	Roll-out engage: FLAGS(10)	Bool	MLOG	OUTCOM	L*1
RM	Local North radius of curvature	nm	ERAD	NAVCOM	R*4
RMP	N R of C including A/C altitude	nm	ERAD	NAVCOM	R*4
RN	Local East radius of curvature	nm	ERAD	NAVCOM	R*4
RNGLS	Range to LOC shack	nm	HNAVSL	BCKCOM	R*4
RNP	E R of C including A/C altitude	nm	ERAD	NAVCOM	R*4
ROLL	Roll attitude (phi)	deg	IOFLL	INPCOM	R*4
RPTR	SNAP read pointer	ndx	SNAP/SNAPOUT	RECCOM	I*2
RSWADR	Adrs of Select Discrete	adrs	(VIEW)	RECCOM	I*4
RTA_IND	'Req Time of Arrival' waypnt	ndx	EXECUTE/LEG_TIME		
	+++		XLAT RTE	CDUCOM	I*2
RTA_PTR	Active cntrpart of 'RTA_IND'	ndx	EXECUTE	CDUCOM	I*2
RTA_TM	ETA at the RTA waypoint	sec	LEG_TIME	CDUCOM	I*4
RTA_WPT	Name of 'RTA' Waypoint	char	EXECUTE/LEG_TIME		
	+++		XLAT RTE	CDUCOM	C*4
RTN	Radius of Turn (2D)	ft	HVGUID	NAVCOM	R*4
RTOD	Conversion factor (180/pi)	dpr	(constant)	CON	R*4
RTRIM	AFD Rudder Trim pot	deg	IOFLL	INPCOM	R*4

<u>ITEM</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>SET BY</u>	<u>RESCOM</u>	<u>TYP</u>
RUDCMD	Rudder Command	deg	LATRL/PRELT	OUTCOM	R*4
RUDMOD	Rudder Model	deg	IOFLL	INPCOM	R*4
RUDPOS	Rudder surface Pos	deg	IOFLL	INPCOM	R*4
RUDTRM	Rudder Trim	deg	LATRL	FCCOM	R*4
RUN	Flight Controls Run select	Bool	IOFLL	INPCOM	L*1
RUNM	Flight Controls Run Mode disc	Bool	MLOG	OUTCOM	L*1
RWYHDG	Runway heading (true)	deg	EXECUTE	DISNAV	R*4
RWYLAT	LAT of Runway threshold	deg	EXECUTE	DISNAV	R*4
RWYLEN	Length of Runway	ft	EXECUTE	DISNAV	R*4
RWYLON	LON of Runway threshold	deg	EXECUTE	DISNAV	R*4
RWYSEL	MLS Runway Select index	ndx	(VIEW)	NAVCOM	I*2
RYELEV	Runway elevation (MSL)	ft	EXECUTE	DISNAV	R*4
SACVIB	SAC Vibration (shaker) amp.	lbs	COMMON	FCCOM	R*4
SATINVW	# (GPS) SATellites tracked	cnt	GPSPRC	NAVCOM	I*2
SC	Dist from T-box to wpt	ft	TGUID	NAVCOM	R*4
SCMD	Gnd Spd/4D Speed Cmd	fps2	SPDCMD/TGUID	FCCOM	R*4
SDC	Progress pt Vel command	fps	HVGUID	NAVCOM	R*4
SDCC	Time box velocity	fps	TGUID	NAVCOM	R*4
SDCOL	SAC pitch pos	in	IOFLL	INPCOM	L*1
SDD	Progress pt Acc command	fps/ft	HVGUID	NAVCOM	R*4
SDWHL	SAC roll pos	in	IOFLL	INPCOM	R*4
SELFPA	Selected FPA (FPA/GS/3D)	deg	MSPLGC/VERCMD	DISNAV	R*4
SEND_BUF	Signal HDL to send wpt bfrs	Bool	CDUFST/HDL	CDUCOM	L*1
SEPR	Dist between T-box and A/P	ft	TGUID	NAVCOM	R*4
SETGD	SET Guidance flags index	ndx	CDUFST/EXECUTE	CDUCOM	I*2
SFCOL	SAC pitch force	lbs	IOFLL	INPCOM	R*4
SFWHL	SAC roll force	lbs	IOFLL	INPCOM	R*4
SHAKER	Stick Shaker	Bool	VERCMD	OUTCOM	L*1
SIMALT	Altitude Sim. airplane (init)	ft	NAVIG	NAVCOM	R*4
SIMCAS	Air Speed Sim. airplane (init)	kts	NAVIG	NAVCOM	R*4
SIMFLG	Simulated airplane ctrl word	p.d.	VIEW/NAVIG	NAVCOM	I*2
SIMHDG	Sim. A/P heading (init)	deg	NAVIG	NAVCOM	R*4
SIMILS	Enable SIMulated ILS	Bool	MLSEX	NAVCOM	L*1
SIMLAT	Latitude Sim. airplane (init)	deg	NAVIG	NAVCOM	R*4
SIMLON	Longitude Sim. airplane (init)	deg	NAVIG	NAVCOM	R*4
SINAZB	Sin(AZ BRG)	()	RSCON	NAVCOM	R*4
SINRH	Sin(RWYHDG)	()	EXECUTE	DISNAV	R*4
SINTH	Sin(HDGTRU)	()	HNAVFS	NAVCOM	R*4
SINUS0	Packed Sensor_In_Use word (1)	p.d.	SINUSE	FCCOM	I*2
SINUS1	Packed Sensor_In_Use word (2)	p.d.	SINUSE	FCCOM	I*2
SINUS2	Packed Sensor_In_Use word (3)	p.d.	SINUSE	FCCOM	I*2
SINUS3	Packed Sensor_In_Use word (4)	p.d.	SINUSE	FCCOM	I*2
SLAT	Sin(latitude)	()	HNAVFS	NAVCOM	R*4
SLLAT	Slow Loop local IDDLAT	deg	HNAVB/HNAVSL	BCKCOM	R*4
SLLON	Slow Loop local IDDLON	deg	HNAVB/HNAVSL	BCKCOM	R*4
SLON	Sin(longitude)	()	HNAVFS	NAVCOM	R*4
SLWCNT	Exception Count for SLOW	cnt	SLOW	OUTCOM	I*4
SMMAGV	Mag Var for Sim A/P (init)	deg	NAVIG	NAVCOM	R*4
SMUHDG	Simulated A/P Heading	deg	NAVIG	NAVCOM	R*4
SNENT	SNAP table count	ndx	DSTAR	RECCOM	I*2

<u>ITEM</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>SET BY</u>	<u>RESCOM</u>	<u>TYP</u>
SOAT	Selected Outside Air Temp	degF	CDUFST/TKOFF	DISNAV	R*4
SPDLMT	4D Speed exceeds Limit	Bool	SPDCMD	FCCOM	L*1
SPFINH	Spoiler Feedback Inhibit	Bool	MLOG	OUTCOM	L*1
SPINTG	Spare Integer / pad	--	--	NAVCOM	I*2
SPL2	Spoiler pan #2 pos	deg	IOFLL	INPCOM	R*4
SPOB1	Old YAWDI / Pad	Bool	--	OUTCOM	L*1
SPOB2	Pad / Recording test	Bool	--	OUTCOM	L*1
SPR7	Spoiler pan #7 pos	deg	IOFLL	INPCOM	R*4
SPREAL	Dummies (12) for debug	()	--	FCCOM	R*4
SPTR	SNAP store pointer	ndx	SNAP	RECCOM	I*2
SQUAT	Squat (weight-on wheels) sw	Bool	IOFLL	INPCOM	L*1
SROLL	Sin(ROLL)	()	ACCPRC	NAVCOM	R*4
SRST	SNAP Reset flag	Bool	DSTAR/SNAP	RECCOM	L*2
SSTICK	SAC (vs. broolly) select	Bool	IOFLL	INPCOM	L*1
STABP	Stabilizer Pos (n/u)	PU	IOFLL	INPCOM	R*4
STFAIL	(signal) STatus FAIL arry(68)	bits	FDSTR/F2CMP	FCCOM	I*2
STHET	Sin(PITCH) (Theta)	()	ACCPRC	NAVCOM	R*4
STKE	sin(TKE)	()	HVGUID	NAVCOM	R*4
STRUA	Self test outputs	p.d.	PRFLT	OUTCOM	I*2
STRUB	Self test outputs	p.d.	PRFLT	OUTCOM	I*2
STRUC	Self test outputs	p.d.	PRFLT	OUTCOM	I*2
SWITCH	STP Switch status word	()	PANEL	FCCOM	I*2
SYNCL	Latrl (aileron) trim value	deg	LATRL	FCCOM	R*4
TANGSA	tan(GlideSlope Angle)	()	EXECUTE/MLOG	NAVCOM	R*4
TAS	True Air Speed	kts	IOFLL	INPCOM	R*4
TASFPS	True Air Speed	fps	HNAVFS/DATSEL	NAVCOM	R*4
TASGS	Airspeed (horizontal comp.)	kts	HNAVFS/DATSEL	NAVCOM	R*4
TAT	True Air Temperature	degC	IOFLL	INPCOM	R*4
TEND	2D 'Turn END' flag	Bool	HVGUID	NAVCOM	L*1
TEND1	Second half Turn flag (4D)	Bool	TGUID	NAVCOM	L*1
THDG	IRS True heading	deg	IOFLL	INPCOM	R*4
THRLIM	Throttle Limit placard	Bool	IOFLL	INPCOM	L*1
THROT	FFD Throtl hdl pos	deg	IOFLL	INPCOM	R*4
TIME	Greenwich Mean Time	sec	FCFAST/IDENT	DISNAV	I*4
TIMERR	4D Guidance Time Error	sec	TGUID	CDUCOM	R*4
TIME_VLD	Valid system GMT available	Bool	CDUEXC/IDENT	CDUCOM	L*1
TIMPTH	TIME PaTH / Grd Speed mode	Bool	MSPLGC	DISNAV	L*1
TK	Track angle (True)	deg	HNAVFS	DISNAV	R*4
TKASUM	Track Angle Summer value	deg	MSPLGC/LATCMD	DISNAV	R*4
TKE	2D Track angle Error	deg	HVGUID	NAVCOM	R*4
TKFLEN	Origin Rwy Length	ft	CDUFST/TKOFF	DISNAV	I*2
TKHDZN	Roll window for TRKHLd	deg	COMMON	FCCOM	R*4
TKMAG	Track angle (Magnetic)	deg	HNAVFS	NAVCOM	R*4
TKREL	Track Pointer on CTV	deg	LATCMD	DISNAV	R*4
TKSEL	Track hold mode discrete	Bool	MSPLGC	DISNAV	L*1
TKTOMP	Track Angle to Mode Panel	p.d.	MSPRO	OUTCOM	I*4
TOWPT	'To' waypoint pntr to WPT_ACT	ndx	HVGUID	DISNAV	I*2
TOEPR	Take-Off EPR	()	TKOFF	CDUCOM	R*4
TOFLPS	Takeoff Flaps	deg	TKOFF	DISNAV	R*4
TOG100	100 M. sec. Toggle flag	Bool	FCFAST	NAVCOM	L*1

<u>ITEM</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>SET BY</u>	<u>RESCOM</u>	<u>TYP</u>
TOINDEX	Takeoff Index	ndx	CDUFST/TKOFF	DISNAV	I*2
TOMPWD	Overlay (10) of Mode Panel Words--		MSPRO	OUTCOM	I*2
TOPOS	Takeoff Runway offset	ft	TKOFF	DISNAV	R*4
TOSTAB	Takeoff Stabilizer pos	P.U.	TKOFF	DISNAV	R*4
TOTIME	Clock time of the wpt	sec	TGUID	NAVCOM	R*4
TOWD	Takeoff Wind Direction	deg	CDUFST/TKOFF	DISNAV	R*4
TOWPT	2D 'To waypoint' Ptr	ndx	HVGUID	DISNAV	I*2
TOWS	Takeoff Wind Speed	kts	CDUFST/TKOFF	DISNAV	R*4
TRIMD	Stab trim down discrete	Bool	ELEVP/STABT	OUTCOM	L*1
TRIMT	Stab trim run discrete	Bool	ELEVP/STABT	OUTCOM	L*1
TRKBG	Track Bug delta	deg	LATCMD	DISNAV	R*4
TRKHL	VCWS Track Hold enabled	Bool	LATRL	FCCOM	L*1
TST3D	3D Guidance o.k. test	Bool	EXECUTE	CDUCOM	L*1
TST4D	4D Guidance o.k. test	Bool	EXECUTE	CDUCOM	L*1
TURN	2D 'In TURN' flag	Bool	HVGUID	NAVCOM	L*1
TURN1	In Turn flag (4D)	Bool	TGUID	NAVCOM	L*1
V1	V-Speed	kts	TKOFF	DISNAV	R*4
V2	V-Speed	kts	TKOFF	DISNAV	R*4
VR	V-Speed	kts	TKOFF	DISNAV	R*4
VACMD	Vertical Acceleration Command	fps2	VERCMD	FCCOM	R*4
VATRD	Vertical trim discrete (down)	Bool	IOFLL	INPCOM	L*1
VATRL	Trim Left Select	Bool	IOFLL	INPCOM	L*1
VATRM	Vertical trim discrete (up)	Bool	IOFLL	INPCOM	L*1
VATRR	Trim Right Select	Bool	IOFLL	INPCOM	L*1
VBCFLG	Vert (3D) Be Careful flag	Bool	VERCMD	FCCOM	L*1
VBS	Vertical Beam Sensed flag	Bool	VERCMD/MLOG	FCCOM	L*1
VCWSE	Velocity CWS Engage: FLAGS(5)	Bool	MLOG	OUTCOM	L*1
VCWSS	VCWS Select discrete	Bool	DISFD	FCCOM	L*1
VDISC	Debounced Discretes to DSP	p.d.	DISFD	DISNAV	I*2
VE	Selected East velocity	kts	HNSWIT/DATSEL	NAVCOM	R*4
VEINS	IRS East velocity	kts	IOFLL	INPCOM	R*4
VELHAT	MLS velocity estimate vector	fps	CFILT	NAVCOM	R*4
VELVLD	IRS Velocities Valid	Bool	IOFLL	INPCOM	L*1
VERPTH	Vert Path (3D) engaged disc.	Bool	MSPLGC	DISNAV	L*1
VERSTR	Vert. Steering signal	fps2	VERCMD	FCCOM	R*4
VGSDOT	Along track acceleration	fps2	HNSWIT/DATSEL	DISNAV	R*4
VGSFLG	VGS vs. PFILT vel for CF IC	BOOL	MLSEX	NAVCOM	L*1
VN	Selected North velocity	kts	HNSWIT/DATSEL	NAVCOM	R*4
VNINS	IRS North velocity	kts	IOFLL	INPCOM	R*4
VSTRA	Vert. path HDOT Cmd	K*fps	VERCMD	FCCOM	R*4
VSTRB	Vert. path gained HDOT	K*fps	VERCMD	FCCOM	R*4
VSTSEL	VERT Steer options	p.d.	VIEW	FCCOM	I*2
VORVLD	VOR #2 Rcvr Valid (always false)	Bool	DISFD	FCCOM	L*1
WD	Wind Direction	deg	BLOW	DISNAV	R*4
WDTV	Watchdog Timer	--	FCFAST/COMMON	OUTCOM	I*2
WDZNE	Var Wheel Dead Zone	deg	COMMON/MLOG	FCCOM	R*4
WEIGHT	A/C gross Weight	lbs	IOFLL	INPCOM	R*4
WGSMSL	WGS-84 reference height of MSL	ft	COMMON/VIEW	NAVCOM	R*4
WHLINP	Dead-zoned DWHL	deg	LATRL	DISNAV	R*4

<u>ITEM</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>SET BY</u>	<u>RESCOM</u>	<u>TYP</u>
WHLSHAP	Exp Shaping for Whl input	()	COMMON/VIEW	FCCOM	R*4
WDMOD	Wind model selection index	ndx	VIEW	BCKCOM	I*2
WPT_ACT	Active Waypoint buffer	--	EXECUTE/HVGUID	NAVCOM	---
WRDCNT	# of (bytes) to o/p to STP	cnt	FMTMG/GMSG/PANEL	FCCOM	I*2
WS	Wind Speed	kts	BLOW	DISNAV	R*4
WSPIN	Wheel SPIN	Bool	IOFLL	INPCOM	L*1
XDDH	MLS/GPS along rwy accel	fps2	HNSWIT/GSPRC	NAVCOM	R*4
XDH	MLS/GPS along runway speed	fps	HNSWIT/GSPRC	NAVCOM	R*4
XGPIP	GPIP X dist from Az/Loc ant	ft	MLSEX/GSPRC	NAVCOM	R*4
XHAT	MLS/GPS X pos estimate	ft	HNSWIT/GSPRC	NAVCOM	R*4
XHTCRS	MSW of Integer XHAT	ft	DASOT	RECCOM	I*2
XHTFIN	LSW (LSB = .25 FT)	ft	DASOT	RECCOM	I*2
XTACC	Selected Cross Track accel	fps2	HNSWIT/DATSEL	DISNAV	R*4
XTK	Horizontal Path error	ft	HVGUID	DISNAV	R*4
XTK1	Integral of (XTKNOM - XTACC)	fps	LATRL	FCCOM	R*4
XTKINS	IRS Cross track acceleration	fps	IOFLL	INPCOM	R*4
XTKLIM	HORPTH Capture limit	ft	LATCMD	FCCOM	R*4
XTVEL	Computed Cross RWY Velocity	fps	DATSEL/NAVIG	FCCOM	R*4
X_DME	X- coord of MLS DME antenna	ft	RSCON (MLSEX)	NAVCOM	R*4
X_EL1	X dist of MLS Ell ant from Az	ft	RSCON (MLSEX)	NAVCOM	R*4
X_HRSW	Radar altitude switch point	ft	RSCON (MLSEX)	NAVCOM	R*1
YDDH	MLS/GPS across rwy accel	fps2	HNSWIT/GSPRC	NAVCOM	R*4
YDH	MLS/GPS across runway speed	fps	HNSWIT/GSPRC	NAVCOM	R*4
YHAT	MLS/GPS Y pos estimate	ft	HNSWIT/GSPRC	NAVCOM	R*4
YHTCRS	MSW of Integer YHAT	ft	DASOT	RECCOM	I*2
YHTFIN	LSW (LSB = .25FT)	ft	DASOT	RECCOM	I*2
YPROF	Az C/L offset from Rwy C/L	ft	MLSEX/GSPRC	NAVCOM	R*4
Y_DME	Y- coord of MLS DME antenna	ft	RSCON (MLSEX)	NAVCOM	R*4
Y_EL1	Y dist of MLS Ell ant from Az	ft	RSCON	NAVCOM	R*4
Z0MLS	Ht of MLS plane above MSL @ Ell	ft	RSCON	NAVCOM	R*4
ZDDH	MLS/GPS vertical acceleration	fps2	HNSWIT/GSPRC	NAVCOM	R*4
ZDH	MLS/GPS vertical speed	fps	HNSWIT/GSPRC	NAVCOM	R*4
ZDIF	Z0 + comp due to Earth curve	ft	HNAVML	NAVCOM	R*4
ZERO	Constant 0.0	()	(constant)	CON	R*4
ZFW	Zero Fuel Weight of airplane	lbs	PFINIT	CDUCOM	R*4
ZHAT	MLS/GPS Z pos estimate	ft	HNAVFS/GSPRC	NAVCOM	R*4
Z_EL1G	Z- coordinate of MLS EL1 ant	ft	RSCON (MLSEX)	NAVCOM	R*4



REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE April 1993	3. REPORT TYPE AND DATES COVERED Contractor Report Jan. 89 - Feb. 91	
4. TITLE AND SUBTITLE Advanced Transport Operating System (ATOPS) Flight Management/Flight Controls (FM/FC) Software Description			5. FUNDING NUMBERS WU 505-64-13	
6. AUTHOR(S) David A. Wolverton, Richard W. Dickson, Winston C. Clinedinst, and Christopher J. Slominski				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Computer Sciences Corporation 3217 N. Armistead Avenue Hampton, Virginia 23666			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) NASA Langley Research Center Hampton, VA 23681-0001			10. SPONSORING/MONITORING AGENCY REPORT NUMBER NASA CR-191457	
11. SUPPLEMENTARY NOTES Langley Technical Monitor: Robert A. Kudlinski				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified - Unlimited Subject Category 06			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) This document describes the flight software developed for the Flight Management/Flight Controls (FM/FC) MicroVAX computer used on the Transport Systems Research Vehicle for Advanced Transport Operating Systems (ATOPS) research. The FM/FC software computes navigation position estimates, guidance commands, and those commands issued to the control surfaces to direct the aircraft in flight. Various modes of flight are provided for, ranging from computer assisted manual modes to fully automatic modes including automatic landing. This document contains a high-level system overview as well as a description of each software module comprising the system. Digital systems diagrams are included for each major flight control component and selected flight management functions.				
14. SUBJECT TERMS Flight Management, Flight Controls, digital avionics, glass-cockpit, autoland, navigation, guidance, computerized flight systems			15. NUMBER OF PAGES 371	
			16. PRICE CODE A16	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	