ADVANCED TRANSPORT OPERATING SYSTEM SOFTWARE UPGRADE

FLIGHT MANAGEMENT / FLIGHT CONTROLS
SOFTWARE DESCRIPTION

Computer Sciences Corporation
Hampton, Virginia 23666
January 1988

Contract NAS1 - 17999
TAO 50277

Prepared by:
Winston C. Clinedinst
Kelly R. Robb
Richard W. Dickson
William J. Heaply
Mark A. Parks
Christopher J. Slominski
David A. Wolerton

Approved by:
Winston C. Clinedinst
Winston C. Clinedinst
Sr. Computer Scientist
Madel B. Reynolds
Operation Director
INTRODUCTION

This document describes the Flight Management/Flight Controls (FM/FC) software for the Norden #2 (PDP-11/70M) computer installed on the NASA 737 airplane. The software computes the navigation position estimates, guidance commands, and those commands to be issued to the control surfaces to direct the airplane in flight based on the modes selected on the Advanced Guidance Control System (AGCS) mode panel, and the flight path selected via the Navigation Control/Display Unit (NCDU).

Real-time data recording and data display for inflight analysis is also provided for. These functions are provided by several "stand-alone" utility tasks that are executed on an as required basis.

In addition to the FM/FC functions performed, a compool of data (DISNAV Compool) which contains the active and provisional guidance buffers as well as other variables needed to generate displays is shipped to the Norden #1 computer via a DRI1-R interprocessor link.

The software described in this document is delivery 4.1 (D4.1) of the baseline FM/FC system which was released in July 1987.
# TABLE OF CONTENTS

1) INTRODUCTION ................................................... i
   1.1) ALPHABETIC LISTING OF MODULE DESCRIPTIONS ................. v

2) EXECUTIVE
   2.1) SYSTEM OVERVIEW ............................................ 2
   2.2) NORDEN #2 EXECUTIVE DESCRIPTION (FLTHDL) ................... 9
   2.3) PRIMARY FOREGROUND SOFTWARE EXECUTIVE (FAST) .............. 12
       2.3A) FAST LOOP DRIVER PROGRAM (FAST) ..................... 19
   2.4) BACKGROUND SOFTWARE EXECUTIVE (SLOW) ...................... 21
       2.4A) BACKGROUND DRIVER PROGRAM (SLOW) .................... 24
   2.5) ERROR MONITOR (FLTEM) ..................................... 25
   2.6) CHANGE MAPPING REGISTERS (MAP) ............................ 26
   2.7) COMPOOL DESCRIPTION (BCKCOM,BULK,CONCOM,DISNAV,FCPOOL,
       FMBFCM,FSTCOM,IOPOOL,NAVCOM,RECOM,SLWCOM) .......... 28

3) INPUT/OUTPUT
   3.1) I/O OVERVIEW .............................................. 67
   3.2) INPUTS
       3.2A) SIU SIGNAL FORMATTING - 50MS (IOFLL) ................ 68
       3.2B) SIU SIGNAL FORMATTING - 10MS (IN1OM) ............... 70
   3.3) OUTPUTS
       3.3A) EIU SIGNAL FORMATTING - 50MS (OUTIO) ................ 75
       3.3B) EIU SIGNAL FORMATTING - 10MS (OUT1OM) ............... 76

4) FLIGHT CONTROLS
   FLIGHT CONTROLS OVERVIEW ....................................... 80
   SET SENSOR IN USE BITS FOR DISFD (DINUSE) ...................... 81
   DISCRETE SELECT AND FAILURE DETECT (DISFD) ..................... 82
   STORE ERROR MESSAGES IN FAILURE DATA TABLE (DSTOR) .......... 84
   ELEVATOR CONTROL LAWS - PITCH (ELEVP) ......................... 85
   ENGINE PRESSURE RATIO LIMIT (EPLMT) .......................... 93
   CHECK SENSOR STATUS FOR FAILURES (FDSTR) ..................... 94
   FORMAT SYSTEM TEST PANEL FAILURE MESSAGES (FMTMG) .......... 95
   CHECK SENSOR SECOND FAILS (F2CMP) ............................ 96
   OUTPUT TO SYSTEM TEST PANEL (GMSG) ............................ 97
   LATERAL AXIS CONTROL LAW (LATRL) ................................ 98
   LOCAL COMPOOL FOR SYSTEM PANEL ERROR MESSAGES (MESG) ....... 107
   MODE LOGIC (MLOG) ............................................. 108
   SERVICE SYSTEM TEST PANEL REQUESTS (PANEL) .................... 115
   INFORM SSFD WHICH SENSORS TO CHECK (SINUSE) ................... 116

5) PRE-FLIGHT
   PRE-FLIGHT OVERVIEW ........................................... 118
   SET AIRPLANE CONTROL SURFACES (CLBIS) ......................... 119
   TEST CONTROLS FOR PROPER DISPLACEMENT (CTLCR) .............. 120
   TEST ILS RECEIVERS (ILSRC) ................................... 121
   PRE-FLIGHT EXECUTIVE (PRFLT) ................................... 122
   TEST RADIO ALTIMETERS (RDALT) ................................ 124
   TEST PITCH, ROLL AND YAW GYROS (RGYRO) ...................... 125
   TEST AILERON, ELEVATOR AND RUDDER SERVOS (SRVCK) ........... 126
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6)</td>
<td>FLIGHT MANAGEMENT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FLIGHT MANAGEMENT OVERVIEW</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>DETERMINE POSSIBLE GUIDANCE MODES (ALTGSX)</td>
<td>129</td>
</tr>
<tr>
<td></td>
<td>FLOATING POINT TO BCD (BT2BCD)</td>
<td>130</td>
</tr>
<tr>
<td></td>
<td>COPY PROVISIONAL TO ACTIVE (BUFSW)</td>
<td>131</td>
</tr>
<tr>
<td></td>
<td>PROCESS CONTROL MODE PANEL INPUTS (MSPLGC)</td>
<td>132</td>
</tr>
<tr>
<td></td>
<td>COMMAND CONTROL MODE PANEL (MSPRO)</td>
<td>134</td>
</tr>
<tr>
<td></td>
<td>PERFORM PATH DEFINITION/GUIDANCE CALCULATIONS (PATHDF)</td>
<td>136</td>
</tr>
<tr>
<td></td>
<td>COMPUTE PLANNED TIME OF ARRIVAL FOR PATH (SRPTA)</td>
<td>154</td>
</tr>
<tr>
<td>6.1) NCDU</td>
<td>COPY ACTIVE TO PROVISIONAL (ACTOPV)</td>
<td>156</td>
</tr>
<tr>
<td></td>
<td>PROCESS ATC PAGE DISPLAY (ATCMOD)</td>
<td>157</td>
</tr>
<tr>
<td></td>
<td>PROCESS DEBUG PAGE (DEBUG)</td>
<td>160</td>
</tr>
<tr>
<td></td>
<td>PARSE NCDU DATA ENTRIES (DECODE)</td>
<td>162</td>
</tr>
<tr>
<td></td>
<td>PROCESS FLIGHT PLAN PAGE (FLTMOD)</td>
<td>166</td>
</tr>
<tr>
<td></td>
<td>PROCESS INIT PAGE (INIMOD)</td>
<td>168</td>
</tr>
<tr>
<td></td>
<td>INITIALIZATION OF SELECT PAGE #1 OPTIONS (INSELH)</td>
<td>170</td>
</tr>
<tr>
<td></td>
<td>PROCESS LOOK-UP PAGE (LKMOD)</td>
<td>171</td>
</tr>
<tr>
<td></td>
<td>INTERFACE OF NCDU AND PATH DEFINITION FUNCTIONS (NCDGUD)</td>
<td>174</td>
</tr>
<tr>
<td></td>
<td>NCDU FAST LOOP FUNCTIONS (NCDKEY)</td>
<td>175</td>
</tr>
<tr>
<td></td>
<td>EXECUTIVE FOR NCDU ROUTINES (NCDUEX)</td>
<td>178</td>
</tr>
<tr>
<td></td>
<td>PROCESS NCDU I/O FUNCTIONS (NCUIO)</td>
<td>179</td>
</tr>
<tr>
<td></td>
<td>PROCESS NAVIGATION DATA PAGES (NVDMOD)</td>
<td>181</td>
</tr>
<tr>
<td></td>
<td>RE-INITIALIZE NCDU VARIABLES (PAGRST/RESETM/RESET2)</td>
<td>184</td>
</tr>
<tr>
<td></td>
<td>SET PROVISIONAL GUIDANCE BUFFER POINTERS (PRVSIZ)</td>
<td>185</td>
</tr>
<tr>
<td></td>
<td>PROCESS SELECT PAGES (SELMOD)</td>
<td>186</td>
</tr>
<tr>
<td></td>
<td>PROCESS TEST PATTERN (TSTMOD)</td>
<td>189</td>
</tr>
<tr>
<td></td>
<td>CREATE A NEW WAYPOINT (WPINV)</td>
<td>190</td>
</tr>
<tr>
<td>7)</td>
<td>NAVIGATION</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NAVIGATION OVERVIEW</td>
<td>192</td>
</tr>
<tr>
<td></td>
<td>COMPUTE ALTERNATE ACCELERATIONS (ACCPRC)</td>
<td>193</td>
</tr>
<tr>
<td></td>
<td>CALCULATE WIND SPEED AND DIRECTION (BLOW)</td>
<td>194</td>
</tr>
<tr>
<td></td>
<td>COMPUTE RADII OF CURVATURE (ERAD)</td>
<td>195</td>
</tr>
<tr>
<td></td>
<td>FAST LOOP NAVIGATION (HNAVFS)</td>
<td>196</td>
</tr>
<tr>
<td></td>
<td>SLOW LOOP NAVIGATION (HNAVSL)</td>
<td>201</td>
</tr>
<tr>
<td></td>
<td>MLS PROCESSING (MLSEX)</td>
<td>210</td>
</tr>
<tr>
<td></td>
<td>SIMULATED AIRPLANE (NAVIG)</td>
<td>223</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE THE SIMULATED AIRPLANE (SMINIT)</td>
<td>225</td>
</tr>
<tr>
<td></td>
<td>CHECK GEOMETRY FOR CROSS TRACK TUNING (TUNCK)</td>
<td>226</td>
</tr>
<tr>
<td></td>
<td>SELECT AND TUNE DME #2 (TUNDM2)</td>
<td>229</td>
</tr>
<tr>
<td></td>
<td>SELECT AND TUNE CROSS TRACK STATION (TUNXTK)</td>
<td>231</td>
</tr>
<tr>
<td>8)</td>
<td>GUIDANCE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GUIDANCE OVERVIEW</td>
<td>235</td>
</tr>
<tr>
<td></td>
<td>COMPUTE GLIDESLOPE/LOCALIZER VARIABLES FOR DISPLAYS (DSPOT)</td>
<td>236</td>
</tr>
<tr>
<td></td>
<td>HORIZONTAL/VERTICAL PATH COMMANDS (HVGUID)</td>
<td>238</td>
</tr>
<tr>
<td></td>
<td>NON-CRITICAL FLIGHT MODES (NCFM)</td>
<td>290</td>
</tr>
<tr>
<td></td>
<td>TIME BOX AND FLIGHT DIRECTOR SPEED COMMANDS (TGUID)</td>
<td>292</td>
</tr>
</tbody>
</table>
9) DATA RECORDING/AQUISITION
DATA RECORDING OVERVIEW ........................................ 296
COLLECT AND FORMAT DATA (DASOT) ............................... 297
DAS/SNAP TABLE ACCESS ROUTINE (DSTAR) .......................... 299
DATA RECORDING LIST (DSTDAT) ..................................... 303
SNAPSHOT RECORDING ROUTINE (SNAP) ............................ 306
SNAPSHOT OUTPUT ROUTINE (SNPOUT) ............................. 308

APPENDICES
  A) FLOWCHARTS ................................................ A-1
  B) DIGITAL SYSTEMS DIAGRAMS ................................. B-1
### ALPHABETICAL LIST OF FM/FC MODULES

<table>
<thead>
<tr>
<th>Module</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCPRC</td>
<td>193</td>
</tr>
<tr>
<td>ACTOPV</td>
<td>156</td>
</tr>
<tr>
<td>ALTGSX</td>
<td>129</td>
</tr>
<tr>
<td>ATCMOD</td>
<td>157</td>
</tr>
<tr>
<td>BLOW</td>
<td>194</td>
</tr>
<tr>
<td>BTBOSD</td>
<td>130</td>
</tr>
<tr>
<td>BUFSWT</td>
<td>131</td>
</tr>
<tr>
<td>CLRIS</td>
<td>119</td>
</tr>
<tr>
<td>CTLCK</td>
<td>120</td>
</tr>
<tr>
<td>DASOT</td>
<td>297</td>
</tr>
<tr>
<td>DEBUG</td>
<td>160</td>
</tr>
<tr>
<td>DECODE</td>
<td>162</td>
</tr>
<tr>
<td>DINUSE</td>
<td>81</td>
</tr>
<tr>
<td>DISFD</td>
<td>82</td>
</tr>
<tr>
<td>DSPDET</td>
<td>236</td>
</tr>
<tr>
<td>DSTAR</td>
<td>299</td>
</tr>
<tr>
<td>DSTDAT</td>
<td>303</td>
</tr>
<tr>
<td>DSTOR</td>
<td>84</td>
</tr>
<tr>
<td>ELEVP</td>
<td>85</td>
</tr>
<tr>
<td>EPRLMT</td>
<td>93</td>
</tr>
<tr>
<td>ERAD</td>
<td>195</td>
</tr>
<tr>
<td>FAST</td>
<td>19</td>
</tr>
<tr>
<td>FDSTR</td>
<td>94</td>
</tr>
<tr>
<td>FLTEM</td>
<td>25</td>
</tr>
<tr>
<td>FLTHDL</td>
<td>9</td>
</tr>
<tr>
<td>FLTMOD</td>
<td>166</td>
</tr>
<tr>
<td>FMTMG</td>
<td>95</td>
</tr>
<tr>
<td>F2CMP</td>
<td>96</td>
</tr>
<tr>
<td>GMSG</td>
<td>97</td>
</tr>
<tr>
<td>HNAVFS</td>
<td>196</td>
</tr>
<tr>
<td>HNAVSL</td>
<td>201</td>
</tr>
<tr>
<td>HVGUID</td>
<td>238</td>
</tr>
<tr>
<td>ILSRC</td>
<td>121</td>
</tr>
<tr>
<td>INIOM</td>
<td>70</td>
</tr>
<tr>
<td>INIMOD</td>
<td>168</td>
</tr>
<tr>
<td>INSHEL</td>
<td>170</td>
</tr>
<tr>
<td>IOFLL</td>
<td>68</td>
</tr>
<tr>
<td>LATRL</td>
<td>98</td>
</tr>
<tr>
<td>LOKMOD</td>
<td>171</td>
</tr>
<tr>
<td>MAP</td>
<td>26</td>
</tr>
<tr>
<td>MSEG</td>
<td>107</td>
</tr>
<tr>
<td>MLOG</td>
<td>108</td>
</tr>
<tr>
<td>MLSEX</td>
<td>210</td>
</tr>
<tr>
<td>MSPLGC</td>
<td>132</td>
</tr>
<tr>
<td>MSPRO</td>
<td>134</td>
</tr>
<tr>
<td>NAG</td>
<td>223</td>
</tr>
<tr>
<td>NCDGUD</td>
<td>174</td>
</tr>
<tr>
<td>NCDKEY</td>
<td>175</td>
</tr>
<tr>
<td>Symbol</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------</td>
<td>------</td>
</tr>
<tr>
<td>NCDUX</td>
<td>178</td>
</tr>
<tr>
<td>NCDUIO</td>
<td>179</td>
</tr>
<tr>
<td>NCFM</td>
<td>180</td>
</tr>
<tr>
<td>NVDMOD</td>
<td>181</td>
</tr>
<tr>
<td>OUTIOM</td>
<td>182</td>
</tr>
<tr>
<td>OUTIO</td>
<td>183</td>
</tr>
<tr>
<td>PAGRST/RESETM/RESET2</td>
<td>184</td>
</tr>
<tr>
<td>PANEL</td>
<td>185</td>
</tr>
<tr>
<td>PATHDF</td>
<td>186</td>
</tr>
<tr>
<td>PRFLT</td>
<td>187</td>
</tr>
<tr>
<td>PRVSIZ</td>
<td>188</td>
</tr>
<tr>
<td>RDALT</td>
<td>189</td>
</tr>
<tr>
<td>RGYRO</td>
<td>190</td>
</tr>
<tr>
<td>SELMOD</td>
<td>191</td>
</tr>
<tr>
<td>SINUSE</td>
<td>192</td>
</tr>
<tr>
<td>SLOW</td>
<td>193</td>
</tr>
<tr>
<td>SMINIT</td>
<td>194</td>
</tr>
<tr>
<td>SNAP</td>
<td>195</td>
</tr>
<tr>
<td>SNMPUS</td>
<td>196</td>
</tr>
<tr>
<td>SRPTA</td>
<td>197</td>
</tr>
<tr>
<td>SRVCK</td>
<td>198</td>
</tr>
<tr>
<td>TGUID</td>
<td>199</td>
</tr>
<tr>
<td>TSTMOD</td>
<td>200</td>
</tr>
<tr>
<td>TUNCK</td>
<td>201</td>
</tr>
<tr>
<td>TUNDM2</td>
<td>202</td>
</tr>
<tr>
<td>TUNXTK</td>
<td>203</td>
</tr>
<tr>
<td>WPINVT</td>
<td>204</td>
</tr>
</tbody>
</table>
**SYSTEM OVERVIEW**

The Flight Management/Flight Controls (FM/FC) software resides in a PDP-11/7OM computer herein referred to as Norden #2. As the name FM/FC implies, this computer contains software that performs the flight management functions, guidance, navigation, and flight controls functions which generate commands that are issued to the control surfaces to direct the airplane in flight.

These functions are performed by three tasks which execute under the RSX-11S operating system. The version used at the time of this document is RSX-11S version 4.2 revision A. The tasks which perform the flight functions are FLTHDL, FAST, and SLOW. Since each of these tasks is described in detail later in this document, only a synopsis of each will be given here.

<table>
<thead>
<tr>
<th>TASK</th>
<th>PRIORITY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLTHDL</td>
<td>175</td>
<td>FM/FC system executive task. Performs all I/O functions with the exception of terminal advisory messages.</td>
</tr>
<tr>
<td>FAST</td>
<td>160</td>
<td>Application software task. Performs all time critical operations. Invokes SLOW task on first pass and aborts SLOW task if FAST task is terminated.</td>
</tr>
<tr>
<td>SLOW</td>
<td>40</td>
<td>Application software task. Performs all operations that are not time critical.</td>
</tr>
<tr>
<td>TERMX</td>
<td>55</td>
<td>Utility task. Allows up to three terminals to &quot;log on&quot; to data monitor program.</td>
</tr>
<tr>
<td>TERMAK</td>
<td>50</td>
<td>Utility task. Allows inspection and modification of resident common data areas.</td>
</tr>
<tr>
<td>DSTAR</td>
<td>50</td>
<td>Utility task. Allows operator interaction via a VT-100 compatible terminal with the Data Acquisition System (DAS) recording variables list.</td>
</tr>
</tbody>
</table>

Only one task may have control of the computer's CPU at any given time. The operating system has the responsibility of granting individual tasks control. The following definitions will be used to describe the five states that a task may be in. Note that the last three are actually sub-states of the second.
Dormant  
No instructions will be executed until the console operator or another task issues a "RUN" command.

Active  
Not dormant.

Waiting  
An active task that has relinquished control of the CPU pending the initiation or completion of some system event.

Competing  
An active task wanting control of the CPU.

Executing  
An active task currently using the CPU.

The operating system grants control of the CPU to the highest priority competing task.

In addition to these program tasks, there are resident common data areas through which data are shared among tasks. These areas are called BULK, NAVCOM, and INCOM. BULK contains the geographic and navigation data required for guidance and for the generation of the Navigation Display (ND), Primary Flight Display (PFD) and NCDU display functions. NAVCOM Rescom contains primarily compools that contain data with initial values. These compools are BCKCOM, FCPOOL, NAVCOM, IOPOOL, and RECOM. IOCOM Rescom contains those compools that do not have initial data values other than zero. These compools are DISNAV and FMBFCM. The IOCOM Rescom is the only Rescom accessed by FLTHDL.
FLIGHT MANAGEMENT/FLIGHT CONTROLS
SOFTWARE FOR ATOPS

RSX-11S
Priority: 250

FLTHDL
Priority: 175
(10 MSEC)

DSTAR
Priority: 50
(offline)

TERMX
Priority: 55
(setup only)

FAST
Priority: 160
(50 MSEC)

SLOW
Priority: 40
(>=1 SEC)

BULK
8K

NAVCOM
4K

I/O COM
4K

SYMTAB
3K

Resident Common Areas
PHYSICAL MEMORY LAYOUT
TASK NAME: FLTHDL

PURPOSE: Executive for Norden #2

TASK SIZE: 1280 words

TASK PRIORITY: 175

INVOKED BY: RSX-11S (RUN FLTHDL)

TASKS INVOKED: None

RESCOMS USED: IOCOM

DESCRIPTION:

This task serves as the overall ATOPS controller, operating in conjunction with the RSX-11S operating system. The entire system is driven by the occurrence of a 10 msec interrupt received via the KW11-P interrupt handler. This interrupt is generated by a 10 msec pulse received from the Digital Autonomous Terminal Access Communication (DATAC) system. The receipt of this interrupt indicates that 10 msec input data can be read and that the 10 msec output data should be written. This is accomplished in FLTHDL by calls to INIOM and OUTIOM respectively.

FLTHDL is organized such that it recognizes five distinct 10 msec interrupts. These 10 msec intervals are referred to as minor frames 0-4. In each 10 msec minor frame, the high speed data is output to the EIU/SIU followed immediately by the high speed data input. The high speed data output consists of the following signals:
- RUDCMD - rudder command
- AILCMD - aileron command
- DECMD - delta elevator command

The high speed input data consists of the following signals:
- Q - pitch rate
- P - roll rate
- R - yaw rate
- HDD - vertical acceleration
- BMACC - body mounted accelerometers.

Five of these 10 msec minor frames are combined to form one 50 msec major frame. During minor frame 0, which coincides with the 50 msec attention bit from the DR11-B DMA interface, denoting the start of a major frame, the large data input is done in addition to the high speed data input. This data block input consists of 362 words being read from the DATAC Shared Interface Ram (SIR) memory using the DR11-B DMA interface. The output of the Data Acquisition System (DAS) recording variables is also done in minor frame 0. A maximum of 150 words can be output to the DATAC SIR memory for subsequent data recording. The limit as far as data recording is concerned is 200 words but a decision has been made to allocate 150 to the FM/FC computer and 50 to the Displays/Data Formatting computer. This decision is not inflexible in that the I/O handler in both machines can be modified to accommodate any combination as long as the total does not exceed the 200 word limit, however this would require concomitant changes in the DATAC system.
In minor frames 1 - 3 only the 10 msec output and 10 msec input is activated. Minor frame 4 begins by activating the 10 msec I/O. Following the 10 msec I/O a test is made to determine whether or not the interprocessor communication, initiated during the previous minor frame 4, has completed. If not, the missed interprocessor communication counter (MSINTP) is incremented for system monitoring purposes. Next, the interprocessor link communication is enabled. This consists of transferring a portion of the resident NAVCOM common area (DISNAV compool in particular) to the Displays/Data Formatting computer. The size of this interprocessor link communication link varies from 219 words to a maximum of 2919 words, depending on the size and status of the guidance buffers. If the guidance buffers are modified, event flag 34 is set by the flight software to initiate the guidance buffer transfer in addition to the basic data during the next major frame. In addition, to guard against software/hardware anomaly, the guidance buffers are transferred once every 4 seconds to ensure that the Display/Data Formatting computer always has the most recent copy of the guidance buffer. This 4 second update rate is also triggered in the flight software by setting event flag 34.

The I/O handler then executes in a continuous loop until it is terminated via an 'ABO FLTHDL' directive. The actual loop performed begins at the label 'LOOP' and continues through frame 4, 'FRM04'.

When the I/O handler is initiated a status check is performed on the external devices needed by the FM/FC computer, namely the EIU/SIU DATAC. Processing will not begin until the connect bit is clear. If a disconnect is sensed, the following message will be displayed on the terminal from which the I/O handler was activated - 'EIU SIU DATAC DISCONNECTED'. Also, processing cannot begin until a 10 msec interrupt is received from the KW11-P real-time clock. The clock interrupt routine is located in the I/O handler and uses the RSX-11S connect to interrupt vector (CINT$) directive to process the interrupt directly. When an interrupt is received from the real-time clock, the interrupt processor sets event flag 4 to signal the handler to begin processing.

One other error condition, the power fail interrupt is addressed by the I/O handler. When this occurs the 10 msec interrupt is re-enabled, the external device status registers cleared (the EIU/SIU DR11-B) the cold start (COLDST) flag is set and processing begins from a restart location (RESTRT).

The two ATOPS computers are designed to operate, to the extent possible, independently of each other. This means that the FM/FC computer can operate even though the Display/Data Formatting computer is not operational or vice versa. While operating in this mode, the interprocessor link is inoperative and, if the flight software is executing, the following message will be displayed on the task initiating terminal - 'tt:tt nn.% INTERPROCESSOR LINK MISSES, where tt:tt is the time of the message and nn.n is the percentage of interprocessor link misses for the previous 60 seconds. Because the message is actually being printed by the background task (SLOW) and the exact time frame in which it operates varies, the percentage could have a +/- 5% error.
MODULES CONTAINED: IN1OM, OUT1OM

VIRTUAL MEMORY MAP: FLTHDL: 120000 - 124773
IOCOM: 140000 - 157777
TASK NAME: FAST

PURPOSE: Controlling task for the FM/FC Norden.

TASK SIZE: 23328 words

TASK PRIORITY: 160

INVOKED BY: RSX-IIS (RUN FAST)

TASKS INVOKED: RQST$C SLOW (initially)
RQST$C DSTAR (initially)

RESCOMS USED: NAVCOM, IOCOM

DESCRIPTION:

FAST serves as the controlling task for the entire FM/FC ATOPS. It invokes SLOW and calls several procedures that perform system I/O, data recording, mode select panel processing and NCDU functions.

To initiate FAST the user must enter the command "RUN FAST" via the RSX-IIS system utility MCR. FAST waits for the occurrence of a 50 millisecond interrupt before processing can begin. A fifty msec interrupt is communicated by task FLTHDL (the I/O handler) when an attention bit is received from the DMA/DATAC interface. It sets event flag 33 for which FAST has been waiting, signaling that external data is available to be read and fast loop processing can begin. The event flag is cleared by FAST, the frame counter updated, and fast loop procedures called.

The main loop processing checks to see if LABFLG is set and MLSVAL is false. If so, an approximation of XYZ position is computed from raw MLS range, azimuth and ALTCOR. This is used to drive the X-Y plotter in the lab. FAST calls each of its modules in the order required to insure that all time critical signals are computed in the same cycle as they are used. In the few cases where best ordering of the modules did not accomplish this, signal computations were moved from one module to another. A brief description of each module follows, in order as they appear in the calling sequence.

MODULES CONTAINED:

DISFD: DISFD detects momentary discrepancies and failed sensor or serial bus errors for the packed discretes. These discretes are input from triplicated sensors and "debounced" to ensure that any bit change was not a fluke.

IOFLL: This module serves as the link between the Sensor Interface Unit (SIU) and the Input area of the IOPOOL compool. It converts raw sensor data to engineering units for use in the flight software.

MSPLGC: MSPLGC processes inputs from the Control Mode Panel (CMP) to perform the logic associated with the various configurations of the automatic flight mode.
**MLSEX:** This module is called only when Flight Controls is in RUN mode. Its primary outputs are MLSVAL and the vectors POSHAT, VELHAT, and ACCHAT from the third order complementary filter. The navigation and flight controls variables formerly computed in subroutines NAVOT and FCCOT are now computed in the using modules (HNAVFS, MLOG, LATRL, ELEVP). The subroutine which had computed the MLS derived glideslope and localizer errors (DSPOT), has also been removed from MLSEX and is now a module called from FAST.

**MLOG:** MLOG controls the operational mode of the Automatic Flight Controls System. Its primary outputs are the operate mode discretes (ICM, RUNM, HOLDM), the flight mode indices (MODEX, MODE2), and the FLAGS array, which contains discretes corresponding to each possible value of MODEX, plus additional discretes for transitional states. In addition, control law variables which are also required by MLOG (DLPSI, HTDZ), or are used or modified by Navigation routines (HDCF, GPGSOV, GPLOCD), are computed here. MLS mode discretes and selected constants (MSW1, MSW6, XFLARE, XGPIP, TANGSA, HGPIP) are set on the first pass that MLSMOD is true.

If no other mode is selected, the operate mode defaults to IC. In this mode, the flight control laws are called, but critical filters are locked in the IC state. The operational flight mode is RUN, selected by pressing the RUN button on the Flight Controls pallet. HOLD is selected by pressing the HOLD button, or is forced when PRE-FLIGHT is selected on the Systm Test Panel (MSWIT = 1) and AEE is true. In HOLD mode, the control law modules (LATRL and ELEVP), the MLS module and the recording modules (DASOT, SNAP, SNPOUT) are bypassed, and MLOG checks only for input of the IC or RUN switches. Note that the AFD/FFD paddles will not remain engaged when in HOLD unless preflight is active. Also, logic in FAILCP prevents the STP mode switch from latching in the preflight position unless SQUAT is true and WSPIN is false.

For a description of the flight modes, see the MLOG documentation.

**ACCPRC:** This module manipulates the INS Along Track Acceleration (ATKACC) and Cross Track Acceleration (XTKACC) inputs in one of several different ways depending on the bits set in the MLS configuration word (MCONF).

**NAVIG:** This module operates the simulated airplane. When SIMFLG is set appropriately, it initializes the navigation position and velocity estimates to a given point in space, with a given flight path angle and track angle, then 'flies' the airplane by monitoring the commands output by HVGUID and TGUID, updating its internal variables and over-writing certain input sensor values. The Flight Controls modules are not involved in this process, and should be left in IC or HOLD mode. When operating under ACD simulation, NAVIG is used to initialize the NAV position estimate at the ACD IC position. When Flight Controls is moded from IC to RUN, SIMFLG is cleared, providing a smooth transition to active flight conditions. Note that logic exists in various modules throughout the system to resolve conflicts that might occur between signals generated in NAVIG and signals generated in other modules.
HNAVFS: This module computes the aircraft heading, position, velocities, and accelerations based on any of several sources. Position is output as LAT and LON in degrees and ALTCOR in feet; velocity is output as VN and VE in knots and HDCF in feet/sec; acceleration outputs are VGSDOT, XTACC, and HDDOT, all in ft/sec/sec. Additionally, TK, TKMAG, GS, and GSFPS are derived from VN and VE, GAMMA is derived from HDCF and GSFPS, and DFTANG is derived from TK and HDGTRU. HNAVFS also computes the output booleans, NAVFLG, NAV64K, and LLINIT (based on ground speed), MLSVAL and MLSMOD (based on MLSVLD and MLSSLI), and computes the sine and cosine of LAT, LON, TK and HDGTRU.

If MLSMOD is true, all output variables (except true heading) are derived directly from the MLS vectors POSHAT, VELHAT, and ACCHAT, and the latitude, longitude and elevation of the MLS azimuth site from the RWYDEF array in FSTCOM. If MLSMOD is false, the outputs are assigned from an analogous set of IDD___ variables computed from an initial position and the best available velocity source, corrected by the slow loop radio navigation computations. If the simulated airplane is engaged, initial position, velocities and accelerations are set by NAVIG. Otherwise, INS inputs are used if INAVF is true. Lower quality solutions based on true air speed, magnetic heading and MAGVAR are also available. The vertical vector is derived from INS HDD and HBARO in either event.

HVGUID: HVGUID compares the aircraft position calculated by HNAVFS to the desired path (if any) in the guidance buffers, and generates path and track error signals (XTK, TKE, HER, DFPA3D) and steering signals (LATSTR, VERSTR) depending on the state of the GUID2D, GUID3D and NAV64K discretes and the WPPTR indices. It also computes the indices PTR2D and PTR4D (based on WPPTR) and the booleans, BCFLAG, TURN, TEND, ALTARM, MDWARN, PATHND, and PTH4DN. The following variables are also output from HVGUID for use in other modules: AMG, DTG, DTOGO, RALC, NOMBA, DSRKT, DOTAL, SDC.

TGUID: TGUID calculates the speed command when time guidance or speed mode is selected; i.e, ground speeds are specified for each leg of the path, an arrival time is specified, (if TIME PATH is to be selected), CAS is engaged, and TIME PATH or speed is selected on the Mode Control Panel. TGUID then 'flies' the time box along the path at the speed specified for each leg in the guidance buffers. The distance between the time box and the aircraft position (SEPR) is then calculated, and from this the acceleration command (SCMD) required to bring the airplane position into coincidence with the time box. If the airplane is ahead of the time box and the required speed to capture would be below safe limits, the TOSLOW flag is set and PSTIAS is cleared. This, with additional logic in NCFM and MSPLGC, causes speed control to revert to CAS HOLD at the present speed. If the speed has been increased beyond the CAS or MACH limit, SCMD is forced to -.25 until the speed is again within limits. TGUID clears GUID4D and sets PTH4DN at the end of the path. SEPR, SDCC, and PTH4DN are referenced by NVDMOD for display on the NCDU.

NCFM: This module outputs the steering commands used by flight controls depending on flight mode, and conditionally updates IASSUM, TKASUM, FPASUM, and ALTSUM. BACMD is computed from TK and TKASUM if TKSEL is true, and set to HORPTH otherwise. VACMD is computed from GAMMA and FPASUM if FPASEL is true, from ALTCOR and ALTSUM if ALTSEL is
true, and set to VERPTH otherwise. ATCMD is computed from CAS and IASSUM unless TIMPTH is true, in which case it is set to SCMD. NCFM also computes TASFPS, ALFSYN and KALFA.

LATRL: LATRL computes the aileron and rudder commands, based on control inputs selected according to flight mode:

PREENG: AILCMD and RUDCMD are set to ALVDT and DRPOS, respectively.

FFDE: Control input is FWHL and output is AILCMD. RUDCMD is not computed. In this and all higher modes, roll angle and roll rate are feedback terms.

MANEL: Control inputs are DWHL, PEDAL, and ATRIM. In this and all other AFD modes, the sum of ALVDT and ATRIM at AFD engagement is "remembered" as SYNCL, which is subsequently subtracted from the basic aileron command.

ACWSE: Control inputs are as above. Submodes are ATTitude SYNC and ATTitude HOLD. In ATT_SYNC, roll attitude is tracked by PHICMD, which then becomes the reference attitude for ATT_HOLD. If FLAP \( > 20 \) deg, PHICMD is washed out and summed with RUDCMD for turn coordination.

VCWSE: This mode has three submodes. Operation in ATT_SYNC and ATT_HOLD is identical to ACWSE. TRACK_HOLD is entered when roll and roll rate are both near zero. In this mode, cross track acceleration is integrated to provide terms used in maintaining a constant ground track.

AUTOE: Control input is BACMD from NCFM unless LAND is selected. Then, once LOCE becomes true, the controlling term is GPLOCD, if in ILS mode, or DELTY (derived from XHAT and YHAT) if in MLS mode.

ELEVP: ELEVP computes the elevator command, stabilizer trim discretes and the throttle position command. DECMD is set to DEPOS if in PREENG, derived from FCOL if in FFDE, or from DCOL if in ACWS OR VCWS. In AUTO mode, the controlling input is VACMD until GSENG, when GPGSDV (in ILS mode) or DELTH (derived from XHAT and ZHAT in MLS mode) becomes the controlling input. At FLARE, one of two flare laws is used to reduce the sink rate to 2 to 3 ft/sec at touchdown. Finally at RLOUT, DECMD is ramped back to zero to lower the nose wheel. Primary feedback terms are PITCH and pitch rate (Q), except in VCWS, where GAMMA is substituted for PITCH. Vertical acceleration is used for additional damping in VCWS, AUTO and LAND, and filtered HDOT (HDZF) is used during LAND. Radio altitude is also used during FLARE.

The stabilizer trim discretes are generated as required to maintain the stab trim (ASTP) signal near zero in all modes except PREENG and MANEL, unless the aircraft is on the ground (GRD = true).

Throttle control is performed wholly within the subroutine ATHCL. The autothrottle engaged discrete (ATE) is set when GRD is false and either IASSEL or TIMPTH is true (TIMPTH is set true for both 4D (TIME) and ground speed modes). If ATE is false, APCDG is synchronized to throttle handle position via the ATFDBK discrete. Otherwise, it is controlled by ATCMD from NCFM and limited by MXEPR from procedure EPRLMT.
in the slow loop. Feedback terms are TASFPS, ATKACC, EPR1, EPR2, and ROLL. When FLARE becomes true, APCDG is ramped back to zero at 2.8 deg/sec.

DSPOT: DSPOT computes the localizer deviation (ETAH), glide slope deviation (BETAH), lateral beam sensed (DLBS) and vertical beam sensed (DVBS) for transmission to the Display/Data Formatting computer.

FDSTR: This procedure monitors sensor and data path failure indicators, and stores the data for the associated error message in one of two buffers, depending on whether the mode switch on the System Test Panel (STP) is in the FLIGHT or PRE-FLIGHT position. Failures monitored are:
- sensor first and second failures detected by SSFD.
- valid sensor selection index (lower byte of sensor status word < 3).
- sensor validity discretes (e.g., INAVV).
- SSFD interface operational (indicated by toggling of BF+1522).
- watch dog timer wraparound.
- analog (A/D - D/A) wraparound.
- SEIU status (BF+1054).

When a failure is logged, the 'failure read' lamp on the STP is lit, and a flag is set to inhibit logging that same message again. The 'failure logged' flags are cleared only on disengaging/re-engaging the system or pressing the 'ERSET' button on the Flight Controls pallet. Parameters for 16 messages may be logged in each buffer. When the buffer is full, any further errors are ignored. Message buffers are cleared by pressing the 'message clear' button on the STP, and read out by pressing the 'failure read' button.

PRFLT: Subroutine PRFLT contains the control logic for the automated pre-flight tests. There are seven routines associated with pre-flight, all of which are called from FSTTSK when MSWIT (set in FDSTR) = 1. Note that when pre-flight is active the calls to MLSEX, LRLTR, 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.

DINUSE: DINUSE sets the 'discrete-in-use' flags based on flight mode (MODEX) and the LOCE, GSENG, ATE, and MLSMOD discretes. These flags enable error checking of the associated discrete. DINUSE is not called in preflight.
SINUSE: SINUSE sets the in-use flags for variable input signals. These flags, plus the CRSET discrete and the INSST flag, are packed into four words (SINUSO, -1, -2, -3) and shipped to the DS/DF computer for use by the SSFD module. SINUSE is not called in pre-flight, as pre-flight sets its own in-use flags.

F2CMP: This module checks the signal second fail flags (set by SSFD/DISFD) for each signal in use in the engaged mode (indicated by MODEX), and sets the mode second fail flag if any required signal is failed. MODEX is limited to 7 (LANDE) for these checks. Entry 8 of the FAIL2 table is used to record autothrottle failure (if ATE), and entry 9 is used to record MLS mode failure. If CRSET is true the FAIL2 table is cleared, and if ERSET is true the 'failure logged' tables are cleared and the FAILURE READ lamp turned off. If preflight is engaged, only CRSET and ERSET checks are performed.

PANEL: PANEL is the System Test Panel service routine. It builds the SWITCH and LIGHTS words, formats and displays stored messages, and clears the message table on request.

MSPRO: This module contains the logic required to compute the outputs needed to light the appropriate mode lights and airspeed, altitude, flight path angle, and track angle digital displays.

DASOT: This module processes the recording tables set up by DSTAR and loads the DAS recording buffer accordingly. It also sets the variables for MLS valids (MLSOV, DMFLC), the packed discrete word (DISOUT) and the recording selection words (SELWD1, SELWD2, SELWD3). It computes the difference between IDD latitude and MLS latitude (LATDIF) and the difference between IDD longitude and MLS longitude (LONDIF).

NCDKEY: NCDKEY interprets the NCDU inputs, posts alert messages, and synchronizes the navigation display updates with the guidance buffer changes. This module is the foreground executive which controls the NCDU background.

OUTIO: This module serves as the link between the Effector Interface Unit (EIU) and the output area of the IOPOOL compool. It converts output data from engineering units to fixed point data for output to the effectors.

SNAP: SNAP records single-event values, called snapshots, for selected variables and stores them in a data table for output to the line printer. This routine is called only if the run switch has been pressed on the flight controls mode panel (this sets the "RUNM" boolean true). If this is the first pass of the FAST task since the cold start flag, "COLDST", has been set, the system diagnostic variables are cleared. These include the following:

- MSWIT - Flight Controls Mode Switch
- OVER - Timer Overflow counter for 60 seconds
- SUMOVR - Timer overflows, cumulative since last cold start
- MAXF - Maximum Frame Time
- FIFTY - The 50 Msec Interrupt Counter
FRAMES - The 10 Msec Frame Counter
COLDST - Cold Start Flag

A comparison is then made of the current frame time to the maximum frame time and the greater of the two stored in "MAXF". A filtered frame time is then computed and stored in FSTCNT to show the approximate time in milliseconds the fast loop is consuming. A test is made to see if the next 50 msec interrupt has occurred, and if so the overflow variable, "OVER", is incremented and event flag 33 cleared to force a wait until the next 50 msec interrupt.

The SLOW task, which was invoked by FAST at priority 40, runs continually in the background mode. Therefore any time remaining in the 50 msec interval not used by the FAST task will allow the SLOW task to resume execution until the next 50 msec interrupt occurs, which reactivates the FAST task. FAST then goes back to the process of calling its embedded procedures.

To terminate the ATOPS software, the user enters the command "ABORT" via the RSX-11S MCR utility. When FAST recognizes that an abort command has been entered, it aborts the other active task (SLOW) and exits back to the RSX-11S operating system.

MODULES CONTAINED: ACCPRC, CLBIS, CTLCK, DASOT, DINUSE, DISFD, DSPOT, ELEVP, FDSTR, F2CMP, HNAVFS, HVGUID, ILSRC, IOFLL, LATRL, MLOG, MLSEX, MSPLCG, MSPRO, NAVIG, NCDKEY, NCFM, OUTIO, PANEL, PRFLT, RDALT, RGYRO, SINUSE, SNAP, SRVCK, TGUID.

VIRTUAL MEMORY MAP:

FAST: 000000 - 127226
NAVCOM: 160000 - 177777
IOCOM: 140000 - 157777
MODULE NAME: FAST

PURPOSE: Initiate tasks and procedures in NORDEN #2 computer.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: None (main program)

CALLING SEQUENCE: N/A

CALLS TO:

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>AROR (C)</td>
<td>(A) Called every frame (50 milliseconds)</td>
</tr>
<tr>
<td>ACCPRC (A)</td>
<td>(B) Called on alternate frames (100 milliseconds)</td>
</tr>
<tr>
<td>CLBIS (C)</td>
<td>(C) Called conditionally</td>
</tr>
<tr>
<td>CTLCK (C)</td>
<td></td>
</tr>
<tr>
<td>DASOT (C)</td>
<td></td>
</tr>
<tr>
<td>DINUSE (C)</td>
<td></td>
</tr>
<tr>
<td>DISFD (A)</td>
<td></td>
</tr>
<tr>
<td>DSPOT (B)</td>
<td></td>
</tr>
<tr>
<td>ELEV (C)</td>
<td></td>
</tr>
<tr>
<td>ELEVP (C)</td>
<td></td>
</tr>
<tr>
<td>ILSRC (C)</td>
<td></td>
</tr>
<tr>
<td>I0FLL (A)</td>
<td></td>
</tr>
<tr>
<td>LATR (C)</td>
<td></td>
</tr>
<tr>
<td>MLLOG (A)</td>
<td></td>
</tr>
<tr>
<td>MLSEX (A)</td>
<td></td>
</tr>
<tr>
<td>MSPLGC (A)</td>
<td></td>
</tr>
<tr>
<td>MSPRO (B)</td>
<td></td>
</tr>
<tr>
<td>NAVIG (A)</td>
<td></td>
</tr>
<tr>
<td>NCBEY (A)</td>
<td></td>
</tr>
<tr>
<td>NCFM (A)</td>
<td></td>
</tr>
<tr>
<td>NIO (A)</td>
<td></td>
</tr>
<tr>
<td>PANEL (A)</td>
<td></td>
</tr>
<tr>
<td>PR1 (C)</td>
<td></td>
</tr>
<tr>
<td>RDLT (C)</td>
<td></td>
</tr>
<tr>
<td>RGYRO (C)</td>
<td></td>
</tr>
<tr>
<td>SINUSE (C)</td>
<td></td>
</tr>
<tr>
<td>SNAP (B)</td>
<td></td>
</tr>
<tr>
<td>SNPSET (C)</td>
<td></td>
</tr>
<tr>
<td>SRVCK (C)</td>
<td></td>
</tr>
<tr>
<td>TGUID (A)</td>
<td></td>
</tr>
</tbody>
</table>

DESCRIPTION:

FAST is the fast loop task's (FAST.TSK) driver program. The console command "RUN FAST" causes both tasks FAST and SLOW to start. See TASK documentation for information on the interaction between the tasks. FAST also calls fast loop procedures to perform necessary 50 and 100 millisecond functions.

The first thing done in FAST is to set up ASTs (Asynchronous System Traps) to provide for fielding attempted aborts of the task and floating point errors. The console command "ABORT FAST" will cause both the FAST and SLOW Task to terminate. Floating point processor errors which occur in FAST (overflow conversion to integer, and divide by zero) will be displayed on the console by a floating point exception AST. See "RSX-11M EXECUTIVE REFERENCE MANUAL" for information on ASTs.

The main body of FAST is the 50 millisecond loop (major frame). All previously mentioned activities are performed once before entering into a continuous loop. The start of a new major frame is synchronized with the handler (see TASK documentation on FLTHDL) via system event flag 33. The setting of event flag 33 causes FAST to immediately enter into a major frame. Within this major frame, time computations are made and procedures are called. One set of conditional inline code computes an approximation of POSHAT (the X,Y,Z position vector) when LABFLG is true and CFRUN is zero (indicating that MLSEX is not computing POSHAT) from raw MLS azimuth and range inputs. This is used to drive the X-Y plotter in the lab.
Some procedures are called on alternate frames to give a 100 millisecond loop. The time computations involved are the running time count (BINTME), the maximum frame time (MAXF), and the filtered frame time (FSTCNT). The units of BINTME is seconds, therefore an increment is performed every 20 frames. Each major frame is divided into five 10 millisecond minor frames. The current minor frame (0 - 4) is available in the compool variable MFRAME. At the finish of the major frame, MFRAME is used to determine how many minor frames have elapsed since the start of the major frame. The result is filtered to give an approximate average time elapsed in performing all 50 millisecond computations. Also the maximum MFRAME detected at the finish is stored as MAXF. The major frame is concluded by testing whether the processing time exceeded the allowed 50 milliseconds by testing event flag 33 to see if another major frame should have already begun. If so, the overflow count (OVER) is incremented and event flag 33 is cleared, which causes FAST to wait for the next major frame before starting again.

GLOBAL INPUTS: ALTCOR, COLDST, CFRUN, FSTSNA, LABFLG, HOLDM, RUNM, MFRAME, MLSRAW, MSWIT, POSHAT, ZOMLS

GLOBAL OUTPUTS: BF (selected areas), BINTME, COLDST, FIFTY, FRAMES, FSTCNT, MAXF, KHD, MSWIT, OVER, SPBP35, SUMOVR, TAT, TOG100
TASK NAME: SLOW

PURPOSE: To provide the background software functions which include navigation, NCDU functions, and error monitoring.

TAS SIZE: 19968 words

TASK PRIORITY: 40

INVOKED BY: FAST ( RQST$C SLOW - issued by module FAST)

TASKS INVOKED: None

RESCOMS USED: IOCOM, NAVCOM, BULK

This task contains all the background functions for the ATOPS flight software. Included in these functions are the NCDU functions, navigation, earth radius computations, wind speed/direction computations, EPR limit computations, failure message processing and error monitoring.

SLOW is invoked by FAST at the lowest priority in the system (priority 40) so that it runs in the background, i.e.; only when no other tasks are active. Since SLOW is the lowest priority task in the computer, it runs constantly until interrupted by a higher priority task or until stopped by aborting task FAST which, as part of the shutdown process, also aborts SLOW.

While running, SLOW counts the number of 10 millisecond frames required for the background functions to execute. This time is stored in a global variable, SLWCNT, so that it can be monitored at execution time to analyze system performance. SLOW then maps to the appropriate memory resident overlay if SIMFLG is on (Simulated Airplane active), to initialize the simulated airplane (procedure SMINIT). The background functions are then called in the following order:

NCDUEX - NCDU Executive
HNAVSL - Navigation Slow Loop
ERAD - Earth Radius Computations
BLOW - Wind Speed/Direction Computations
EPRLMT - Engine Pressure Ratio Limits Computations

Since SLOW exceeds 32K words of memory it has been organized into memory resident overlays. Memory resident means that the entire task is physically resident in memory but due to limits of the PDP-11/70 architecture, only 32K words of the task can be active at any given time (refer to the RSX-11M Task Builder Manual for a complete discussion of memory resident overlays). Each overlay portion of the task is referred to as a segment. Two procedures, SLOW and NCDUEX, determine which segment will be "mapped" at any given time during execution of SLOW. Procedure MAP handles the manual segment switching.

The NCDU executive (NCDUEX) determines from the NCDU mode switch or the selected NCDU function key which overlay segment to "map".
Following is a list of segment numbers and the procedures contained in each segment:

<table>
<thead>
<tr>
<th>segment #</th>
<th>procedures contained</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ATCMOD, FLTMOD, INIMOD, ACTOPV, LABEL</td>
</tr>
<tr>
<td>2</td>
<td>LOKMOD, NVDMOD</td>
</tr>
<tr>
<td>3</td>
<td>TSTMOD, DEBUG, DECODE, SELMOD, SMINIT, SNPOUT, GMSG</td>
</tr>
<tr>
<td>4</td>
<td>PATHDF, SRPTA, NCDGUD, BUFSWT</td>
</tr>
<tr>
<td>5</td>
<td>FLTEM</td>
</tr>
<tr>
<td>0 (ROOT)</td>
<td>SLOW, NCDUEX, INSELH, WPINVT, HNAVSL, TUNCK, TUNXTK, ERAD, BLOW, EPRLMT</td>
</tr>
</tbody>
</table>

Once a given segment has been "mapped", it remains active until a function is selected via the NCDU which requires that another segment be mapped. (See NCDU documentation for more details.)

The mainline functions of SLOW are contained in the root segment. These are the navigation functions, EPR computations, the wind speed/direction computations, and the earth radius computations, which are called during each iteration of SLOW. During each iteration, a test is also made to determine whether any "snapshot" data needs to be printed. This is determined by comparing the "store" pointer to the "read" pointer. If they are not equal then "snapshot" data needs to be printed. This is accomplished by "mapping" to segment 3 and calling the SNPOUT procedure.

Following completion of the mainline routines, a 60 second timer AST-set flag is tested to see if 1 minute has elapsed and if it has, segment 5 is "mapped" and the error monitor (FLTEM) is called to display any error conditions that have occurred during the previous minute. These errors include missed 10 millisecond interrupts, missed 50 millisecond interrupts, 50 millisecond timer overflows, and missed interprocessor links.

Finally a test is made to determine if a message needs to be displayed on the system test panel. If so, this requires the "mapping" of segment 3 and calling the GMSG procedure.

If at any time during the execution of the SLOW task a floating point exception should occur, the floating point exception procedure (FPEXCP) is invoked which displays a message on the system console device. This message gives the location of the program counter and register contents and the type of exception as a debugging aid.

MODULES CONTAINED: ACTOPV, ATCMOD, BLOW, BUFSWT, DEBUG, DECODE, EPRLMT, ERAD, FLTEM, FLTMOD, GMSG, HNAVSL, INIMOD, INSELH, LABEL, LOKMOD, NCDGUD, NVDMOD, NCDUEX, PATHDF, SELMOD, SLOW, SMINIT, SNPOUT, SRPTA, TSTMOD, TUNCK, TUNXTK, WPINVT

VIRTUAL MEMORY MAP: NAVCOM: 160000 - 177777
IOPOOL: 140000 - 157777
RULK: 100000 - 137777
SLOW: 000000 - 057777

22
MODULE NAME: SLOW

PURPOSE: Calls background procedures.

TASK: SLOW

LANGUAGE: MACRO-11

CALLED BY: None (main program)

CALLING SEQUENCE: None

CALLS TO: SMINIT (B)
          HNAVSL (A)
          ERAD (A)
          BLOW (A)
          EPRLMT (A)
          FLTEM (C)
          SNPOUT (B)
          GMSG (B)
          FPEXCP (B)
          MAP (B)
          NCDUEX (A)
          SNPSET (B)

  (A) Called each time.
  (B) Called conditionally.
  (C) Called every minute.

DESCRIPTION:
SLOW is a background task's (SLOW.TSK) driver program. The module acts mainly as a caller, but also performs a background elapsed time calculation and handles floating point processor exception traps, and controls the breakpoint/SNAP function for the task.

GLOBAL INPUTS: BINTIME, FRAMES, LABFLG, SIMFLG, IOACT, RPTR, SLWSEG, SLWSNA, SPTR, WRDCNT

GLOBAL OUTPUTS: SEG, IOACT, SLWCNT
MODULE NAME: FLTEM

PURPOSE: Display I/O handler related error messages at system console.

TASK: SLOW

LANGUAGE: MACRO-11

CALLED BY: SLOW

CALLING SEQUENCE: CALL FLTEM

CALLS TO: BCDTIM

DESCRIPTION:
FLTEM is called once each minute, from the background task SLOW, to display I/O handler error messages. Error counts for the previous minute are checked and if none exceed a threshold value a return is made after resetting all the error counters to zero. The threshold value for all but the timer overflow counter is 2 percent of the maximum errors that could occur in the 1 minute time span. If any timer overflow errors occur during the preceding minute that error message will be displayed. See FLTHDL documentation for the description of the error counters. Also see description of module FAST for the description of the timer overflow counter.

The following is a sample console print out when all six counters have errors beyond the toleration threshold.

08:47:25 10.0% 10 MSEC MISS
08:47:25 2.8% 50 MSEC MISS
08:47:25 6.4% I/P LINK MISS
08:47:25 5.0% EIU/SIU ERR
08:47:25 12.1% I/P LINK ERR
08:47:25 8.3% OVERFLOWS

GLOBAL INPUTS: EIUERR, INTERR, MS1OML, MS5OML, MSINTP, OVER, SUMOVR

GLOBAL OUTPUTS: None
MODULE NAME: MAP

PURPOSE: Change mapping registers for various overlay segments.

TASK: SLOW

LANGUAGE: MACRO-11

CALLED BY: NCDUEX, SLOW

CALLING SEQUENCE: SEG = N (N - number of desired overlay segment)
  CALL MAP

CALLS TO: $LOAD

DESCRIPTION:
Before a call to a subroutine that resides in a memory resident overlay segment is made, a call to MAP is made to assure the virtual address mapping registers are set to the desired overlay segment. A number corresponding to the desired segment is stored in the global SEG before the call to MAP. The subroutine MAP has the currently mapped segment number in its local variable CSEG. If SEG and CSEG are the same an immediate return results. Otherwise the desired segment is mapped by calling the system subroutine $LOAD with the name of the segment that corresponds to the number in SEG. See RSX-11M Task Builder manual for information about $LOAD and overlays. Also see task documentation for SLOW for the description of the task overlay setup.

GLOBAL INPUTS: SEG

GLOBAL OUTPUTS: None
2.8 COMPOOLS
BCKCOM COMPOOL

BCKCOM is included in the NAVCOM resident common. BCKCOM contains variables that are referenced only from the background navigation software.
BCKCOM COMPOOL VARIABLES

ACMODE  INTEGER  ATC CLEAR MODE
ACTPOS  INTEGER  ACTIVE STR. POSITION
ALRMES  INTEGER  ALERT MESSAGE CODE
ATCMWP  BOOLEAN  THREE WAYPOINT FLAG
ATC2    ARRAY(12) INTEGER  "EXEC OR REJ" MSG
AUOMODE BOOLEAN  NCDU AUTO MODE
BARALR BOOLEAN  BAROSET ALERT FLAG
BAZFLG BOOLEAN  BACK AZIMUTH ILD GAINS
BRGLS  SCALAR  BRG FROM LOCALIZER SHACK
DATNUM  INTEGER  FUNCTION DATA READY
DECNUM  INTEGER  DECODED KEY CODE

DUALNM  INTEGER  DUAL NUMBER FUNCTION
FLSTA2  BIT(16)  STATION 2 FAIL FLAGS
FLSTA3  BIT(16)  STATION 3 FAIL FLAGS
FPMODE  INTEGER  FLIGHT PLAN MODE
FUNUMB BOOLEAN  NUMERIC FUNCTION FLAG
FOG     INTEGER  DME 3 FAIL FLAGS
F2G     INTEGER  DME 2 FAIL FLAGS
F3G     INTEGER  VOR 2 FAIL FLAGS
GDTIME  SCALAR  BUFFER SEND TIMER
INMODE  INTEGER  INITIALIZE MODE
KEYUNL BOOLEAN  KEYBOARD ENABLED
LOKCEN  BOOLEAN  MAP CENTER REF. FLAG
LUMODE  INTEGER  NCDU LOOK-UP MODE
MAMODE  BOOLEAN  NCDU MANUAL MODE
MNAVTY  BIT(16)  NAVIGATION MODE
MODPAG  INTEGER  PAGE MODE
MODUPD BOOLEAN  MODIFIED ERROR CODE
NCDERR INTEGER  NCDU ERROR CODE
NCDSPC INTEGER  NUMBER FUNCTION RECEIVE
NCDFUN INTEGER  FUNCTION INPUT
NCMESSG ARRAY(11) INTEGER  NCDU INPUT LINE
NDMODE  INTEGER  NAV DATA MODE
NUPAGE INTEGER  NEW PAGE REQUEST
PAGSEN BOOLEAN  PAGE SEND FLAG
PROV   BOOLEAN  PROVISIONAL MODE
PRVMP  BOOLEAN  PROV MAP
PTAFLG BOOLEAN  PTA'S CALC FLAG
PTAPOS  INTEGER  PTA WAYPOINT POSITION
PTATIM  SCALAR  PTA AT THE WAYPOINT PTAPOS
PWEND  INTEGER  WAYPOINT TERMINATION POINTER
RINPFL  BOOLEAN  STATION SEARCH IN PROGRESS
RNGLS  SCALAR  RANGE TO LOCALIZER SHACK
RSTDCTR INTEGER  STATION CYCLE INDEX
RTNPRV ARRAY(5) INTEGER  TUNDM2 SCRATCH
RTNSCR INTEGER  SCRATCH VARIABLE
RWYCNT INTEGER  RWF HEADING TO INS CTRL.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RZNCTR</td>
<td>INTEGER</td>
<td>ZONE COUNTER (CYCLIC)</td>
</tr>
<tr>
<td>RZNPTR</td>
<td>INTEGER</td>
<td>ZONE POINTER</td>
</tr>
<tr>
<td>SEMODE</td>
<td>INTEGER</td>
<td>NCDU SELECT MODE</td>
</tr>
<tr>
<td>SLLAT</td>
<td>SCALAR</td>
<td>LOCAL IDDLAT</td>
</tr>
<tr>
<td>SLLON</td>
<td>SCALAR</td>
<td>LOCAL IDDLON</td>
</tr>
<tr>
<td>STATUS</td>
<td>BOOLEAN</td>
<td>NAV SOURCE SIGNAL FAILURE</td>
</tr>
<tr>
<td>ZONELM</td>
<td>INTEGER</td>
<td>ZONE LIMIT</td>
</tr>
</tbody>
</table>
BULK DATA STORAGE

Bulk Data is the organization for storage of navigation and geographic data required for guidance, ND map display, and NCDU display functions of the ATOPS system. 8192 words (2 APRs) of memory are allocated for storage in both the Display and FM/FC Nordens.

Bulk Data is separated into primary categories: longitudinal strips, airways and routes, Standard Instrument Departures (SID) and Standard Terminal Arrival Routes (STAR), airfields, geographic reference points, navaids, terrain objects, and boundaries. Data in each category is tabulated and stored in a dedicated format.

The first 14 words of Bulk Data consists of the pointer table (see Table 2-1). Each location in this table is a pointer to a dedicated buffer zone.

-- Longitudinal Strip Data Zone (Index Block)
-- Airways and Routes Data Zone
-- Boundary Data Zone

INDEX BLOCK POINTER

The index block pointer (IBPTR) points to a category of data in each longitudinal strip. These categories are:

-- West Boundary Longitude
-- East Boundary Longitude
-- Airfields
-- Geographic Reference Points
-- Navigational Aids
-- Terrain Objects

The pointers in each strip contain the memory address of the first data word in the buffer associated with each category of data (see Table 2-2).

AIRWAYS AND ROUTES POINTERS

The Airways and Routes pointers (JETPTR, VICPTR, RTEPTR) point to VOR/DME defined airways, area navigation airways and special predefined routes representative of airline company routes. This buffer is organized to define each airway or route as a list of pointers (addresses) to data in other buffers.

The buffer is subdivided into three areas dedicated to:

-- Routes
-- Jet Airways
-- Victor Airways

BOUNDARY ZONE POINTER

Restricted area (RESPTR), Air Defense Identification Zone (ADZPTR) and Coastal Air Defense Identification Zone (CDZPTR) pointers point to the boundary data zone.

INDEX BLOCK FOR LONGITUDE STRIPS BUFFER

The first four words of data consists of the East and West boundaries for the longitudinal strip. The first strip is geographically the western-most strip. The next four words are pointers to specific data buffers. If a
pointer is zero, there is no data in the buffer for that strip. There can be no zero longitude, so two consecutive zero terminators will indicate an end of the longitude strip buffer. (See Table 2-2.)

AIRWAYS AND ROUTES DATA BUFFER
The airway and route buffers contain the name of the JET or VICTOR airway in ASCII characters, a pointer to the name in the next data block, and a pointer to each waypoint in the airway or route; waypoints include VOR/VORTAC intersections and geographic references. (See Table 2-3.)

SID/STAR INDEX BLOCK AND DATA BUFFER
SIDs and STARs are identified by an alphanumeric designator of up to six characters. To facilitate access to the data, the buffer is preceded by SID and STAR index blocks organized as shown in Table 2-4. The SID index table includes pointers to SID data buffers. The STAR index table follows the same format. The SID/STAR buffer organization is shown in Table 2-5. This buffer contains the following:

-- SID/STAR name, in ASCII format (3 words)
-- pointer to the next SID/STAR in the data block
-- pointer to the associated runway

For each waypoint in the SID/STAR the buffer contains
-- pointer to the waypoint referenced by this SID/STAR
-- altitude of the waypoint (negative altitude denotes the entry of a DME Arc path segment)
-- speed at the waypoint (negative speed indicates that airspeed references from the NCDU should be used for display)
-- radius of turn
-- zero unless a DME turn, in which case it contains bearing from the turn center to the inbound waypoint (turn angle is stored in this position of the outbound waypoint of a DME turn. A negative turn angle denotes a left turn.)

Two consecutive zero terminators indicate the end of the buffer.

AIRFIELD AND RUNWAY BUFFER
The airfield data buffer will include airfield name, latitude and longitude of the reference point, a pointer to the next airfield in the block, the length of the longest runway, azimuth of longest runway, magnetic variation, elevation of reference point, pointer to terminal data (not used), and 4 words of 2x5 frequency data. Potential origin or departure airfields are distinguished from airfields for ND display only by having no terminator (equal to zero) following the last 2x5 frequency (ATIS). Instead, SID and STAR index, name of runway, latitude and longitude threshold, outer marker, and runway data follow. (See Table 2-6.)

NAVAIDS BUFFER
The navaids buffer will store the pertinent data for all in route and terminal area radiating navigation aids except for ILS and runway marker beacons. The navaids buffer contains:

-- VORTAC's
-- VOR's
-- Non-directional Radio Beacons (NDB)
The buffer format is defined in Table 2-8. The format has the navaid name in ASCII characters, frequency in 2x5 code, a formatted word giving the navaid type, where:

- bit 4 0 = low altitude, 1 = high altitude
- bit 2,3 00 = VOR, 01 = VORTAC, 10 = NDB

also the latitude and longitude, magnetic variation, and the elevation in feet.

TERRAIN BUFFER
The buffer contains two categories of data:

- Mountains
- Obstructions

This buffer includes altitude (ASCII characters) in thousands, a zero in bit position 15 for mountains and a one for obstructions, and latitude and longitude of the terrain object. (See Table 2-9.)

BOUNDARY DATA BUFFER
This data is used for ND map display only. The boundary data buffer defines restricted areas, coastal air defense identification zones (CADIZ), and air defense identification zones (ADIZ) boundaries. The first three words in the block define the zone name in ASCII characters, followed by the boundaries in latitude and longitude pairs. Two consecutive zero terminators indicate the end of the buffer. The boundary data buffer is the last information in Bulk Data. It is followed by a global label, B4CKZ2, which contains two ASCII characters for the version I.D. (See Table 2-10.)
KWAL:

WORD "KW"
WORD "AL"
FLT2 37.94277777 N 37 56 34
FLT2 -75.466666 W 75 28 00
WORD G28 GENERATED BLOCK POINTER
FLT2 8748. RUNWAY LENGTH
FLT2 32.18999999 HEADING
FLT2 -9.0 AZIMUTH
FLT2 41. ELEVATION
WORD 0

WORD 005060 126.5 Tower frequency
WORD 024510 0 Clearance frequency
WORD 024510 0 Ground control frequency
WORD 024510 0 ATIS frequency

WORD SIDWAL SID/STAR
WORD STRWAL SID/STAR

WAL04X:

WORD "04" 04

WORD 0 MISSED APPROACH PATH
FLT2 37.92702777 N 37 55 37.3
FLT2 -75.470944 W 75 28 15.4
WORD 0 OUTER MARKER NAME - NOT USED
WORD 0 OUTER MARKER NAME - NOT USED
FLT2 37.95027777 N 37 57 01.0
FLT2 -75.452472 W 75 27 08.9
FLT2 8748. RUNWAY LENGTH
FLT2 32.18999999 AZIMUTH
FLT2 40. ELEVATION
FLT2 3.0 GLIDE SLOPE
WORD 024510 0
WORD 0 BLOCK TERMINATOR

TABLE 2-6.- AIRFIELD/RUNWAY BUFFER

GRW74:

WORD "GA"
WORD "PA"
WORD 116
FLT2 37.89458333 N 37 56 15.4
FLT2 -75.49711111 W 75 28 15.4
WORD SWL

LEE01:

WORD "LE"
WORD "EO"
WORD 61
FLT2 37.8635
FLT2 -75.42188556
WORD SWL

LEE02:

WORD "LE"
WORD "EO"
WORD 62
FLT2 37.83325
FLT2 -75.4616667
WORD SWL

TABLE 2-7.- GEOGRAPHIC REFERENCE POINT BUFFER
CONCOM is included in the SLOW and FAST tasks of both the Flight Management and Displays systems. CONCOM contains standard values for constants that are used throughout the system.
CONCOM COMPOOL VARIABLES

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSETO</td>
<td>INITIAL (29.92) SCALAR</td>
<td>SEA LEVEL PRESSURE (IN)</td>
</tr>
<tr>
<td>C360</td>
<td>INITIAL (360.) SCALAR</td>
<td>360 DEGREES</td>
</tr>
<tr>
<td>DEG15</td>
<td>CONSTANT (15.) SCALAR</td>
<td>15 DEGREES</td>
</tr>
<tr>
<td>DEG25</td>
<td>CONSTANT (25.) SCALAR</td>
<td>25 DEGREES</td>
</tr>
<tr>
<td>DELTAT</td>
<td>INITIAL (0.05) SCALAR</td>
<td>DELTA TIME FRAME (SEC)</td>
</tr>
<tr>
<td>DLT100</td>
<td>INITIAL (0.1) SCALAR</td>
<td>DELTA TIME FOR 100 MSEC LOOP</td>
</tr>
<tr>
<td>DTOR</td>
<td>INITIAL (.0174532925) SCALAR</td>
<td>CONV DEG TO RADIANS</td>
</tr>
<tr>
<td>ELLIP</td>
<td>INITIAL (.0033901) SCALAR</td>
<td>ELLIPTICITY FACTOR</td>
</tr>
<tr>
<td>FTONM</td>
<td>INITIAL (1.6457883E-4) SCALAR</td>
<td>FEET TO NM CONV</td>
</tr>
<tr>
<td>GRAVO</td>
<td>INITIAL (32.1739) SCALAR</td>
<td>NOMINAL ACC OF GRAVITY (FPS2)</td>
</tr>
<tr>
<td>ITIME</td>
<td>CONSTANT (0.5) SCALAR</td>
<td>ITERATION TIME CONSTANT</td>
</tr>
<tr>
<td>KTOFPS</td>
<td>INITIAL (1.687809858) SCALAR</td>
<td>CONV FROM KNOTS TO FPS</td>
</tr>
<tr>
<td>NMTFT</td>
<td>INITIAL (6076.11549) SCALAR</td>
<td>CONV FROM NM TO FT</td>
</tr>
<tr>
<td>ONE80</td>
<td>CONSTANT (180.) SCALAR</td>
<td>180 DEGREES</td>
</tr>
<tr>
<td>PI</td>
<td>INITIAL (3.1415927) SCALAR</td>
<td>PI</td>
</tr>
<tr>
<td>RADIUS</td>
<td>INITIAL (3443.93618) SCALAR</td>
<td>NOMINAL EARTH RADIUS (NM)</td>
</tr>
<tr>
<td>RTOD</td>
<td>INITIAL (57.29578) SCALAR</td>
<td>CONV RADIANS TO DEG</td>
</tr>
<tr>
<td>WPSIZ</td>
<td>INITIAL (90) INTEGER</td>
<td>WAYPT BUFFER SIZE (BYTES)</td>
</tr>
<tr>
<td>ZERO</td>
<td>CONSTANT (0.) SCALAR</td>
<td>ZERO</td>
</tr>
</tbody>
</table>
DISNAV COMPOOL

DISNAV is included in the IOCOM resident common of both Flight Management and Displays systems. DISNAV contains navigation and display variables that are set by the FM/FC Norden #2 and transmitted to Norden #1 over the interprocessor link. In addition to containing many discretes, integer flags and scalar signals, DISNAV contains the active and provisional guidance buffers.
DISNAV COMPOOL VARIABLES

ACTERM                INTEGER INITIAL   0 WORD TERMINATOR TO ACTIVE
ACTIVE                STRUCTURE      THE ACTIVE GUIDANCE BUFFER
ACTIVE.PWAO2          SCALAR         ARC LENGTH OF CURVE 0 2 (FT).
ACTIVE.PWASS          INTEGER        0=AWY 1=ST 2=STR 3=MAP 4=NOMAP
ACTIVE.PWBKG          SCALAR         BEARING
ACTIVE.PWCCD          SCALAR         CENTER TO CENTER DIST (FT).
ACTIVE.PWDMA          BOOLEAN        DME ARC?
ACTIVE.PWDMMI         BOOLEAN        DME ARC STOP CALC
ACTIVE.PWDSS          SCALAR         TANGENT PT TO WPT DIST (FT).
ACTIVE.PWFLLL         BOOLEAN        FLIGHT LEVEL?
ACTIVE.PWGAM          SCALAR         PATH GRADIENT (DEG).
ACTIVE.PWH             SCALAR         ALTITUDE (FT).
ACTIVE.PWIRL          BOOLEAN        IAS REFERENCE?
ACTIVE.PWLAT          SCALAR         LATITUDE (DEG).
ACTIVE.PWLGFR         SCALAR         DME ARC REFERENCE LON (DEG).
ACTIVE.PWLON           SCALAR         LONGITUDE (DEG).
ACTIVE.PWLT            SCALAR         DME ARC REFERENCE LAT (DEG).
ACTIVE.PWMM            SCALAR         MAGNETIC VARIATION (DEG).
ACTIVE.PNWAN          ARRAY(3) INTEGER WPT NAME
ACTIVE.PWPPD          SCALAR         CIRCLE PT TO PT DIST (FT).
ACTIVE.PWPTA          SCALAR         PLANNED TIME OF ARRIVAL (SEC).
ACTIVE.PWRAD          BOOLEAN        RADIUS INHIBIT?
ACTIVE.PWRTN          SCALAR         RADIUS OF TURN (FT).
ACTIVE.PWTA            SCALAR         TURN ANGLE AT WPT (DEG).
ACTIVE.PWDT            SCALAR         DELTA TIME EN ROUTE (SEC).
ACTIVE.PHWVPR          INTEGER        ADDR OF VOR
ACTIVE.PWV              SCALAR         GROUND SPEED (KNOTS)
ACTIVE.PWAY            BOOLEAN        PROVISIONAL?
ALT                   INITIAL (0.) SCALAR INERTIALLY SMOOTHED ALTITUDE (FT).
ALTARM                INITIAL (OFF) BOOLEAN ALTITUDE MODE ARMED
ALTATT                INITIAL (ON) BOOLEAN ALTATT ATTAINED FLAG
ALTCCOR               INITIAL (OFF) SCALAR BARO CORRECTED ALTITUDE (FT).
ALTSEL                INITIAL (OFF) BOOLEAN ALT HOLD MODE DISCRETE
ALTSUM                INITIAL (0.) SCALAR ALTITUDE SUMMER VALUE (FT).
BARSET                INITIAL (29.92) SCALAR BAROMETRIC REFERENCE SETTING (IN)
BETAH                  SCALAR         MLS OR ILS BETA HAT (DEG).
BINTIME                INITIAL (0.) SCALAR REAL TIME CLOCK (SEC).
COSRH                  INITIAL (0.) SCALAR COS (RWY HEADING)
CRAD               INITIAL (0.) SCALAR RADIUS OF CIRCLE (MILES/TENTHS)
DECTY                INITIAL (0.) INTEGER TYPE OF CIRCLE WPT
DESTIN                INITIAL (0.) ARRAY (4) INTEGER DESTINATION AIRFIELD AND RUNWAY
DFETANG               INITIAL (0.) SCALAR DRIFT ANGLE (DEG).
DLATFT                INITIAL (36567.) SCALAR LENGTH OF 1 DEGREE OF LATITUDE (FT).
DLONFT                INITIAL (289693.) SCALAR LENGTH OF 1 DEGREE OF LONGITUDE (FT).
DSRTK                 INITIAL (0.) SCALAR DESIRED TRACK (DEG).
DTOGO                 INITIAL (0.) SCALAR DIST TO MID-TURN (FT).
ETAH                   SCALAR         MLS OR ILS ETA HAT (DEG).

38
DISNAV COMPOOL VARIABLES (cont.)

FCFLGS
  INITIAL (HEX '0') BIT (16) FLIGHT CONTROLS FLAGS TO DISPLAYS

FPASEL
  INITIAL (OFF) BOOLEAN FPA HOLD MODE DISCRETE

FPASUM
  INITIAL (0.) SCALAR FPA SUMMER VALUE (DEG).

GAMC
  INITIAL (0.) SCALAR COMMANDED GAMMA VALUE (DEG).

GAMMA
  INITIAL (0.) SCALAR FLIGHT PATH ANGLE (DEG).

GS
  INITIAL (0.) SCALAR GROUND SPEED (KNOTS).

GSA
  INITIAL (0.) SCALAR GLIDE SLOPE ANGLE (DEG).

GSFPS
  INITIAL (0.) SCALAR GROUND SPEED (FPS).

GUID2D
  INITIAL (OFF) BOOLEAN GUID 2D POSSIBLE.

GUID3D
  INITIAL (OFF) BOOLEAN GUID 3D POSSIBLE.

GUID4D
  INITIAL (OFF) BOOLEAN GUID 4D POSSIBLE.

HDCF
  INITIAL (0.) SCALAR HDOT COMP FILTER (FPS).

HDDOT
  INITIAL (0.) SCALAR VERTICAL ACCEL (FPS2).

HDGTRU
  INITIAL (0.) SCALAR TRUE HEADING (DEG).

HER
  INITIAL (0.) SCALAR ALTITUDE ERROR (FT).

HLDBRG
  INITIAL (0.) SCALAR BRG OF HOLD WPT (DEG).

HLDSEL
  INITIAL (OFF) BOOLEAN HOLD WPT SELECTED?

HLDWPT
  INITIAL (0) INTEGER HOLD WPT

HORPTH
  INITIAL (OFF) BOOLEAN HORIZONTAL PATH MODE

IASSEL
  INITIAL (OFF) BOOLEAN IAS HOLD MODE DISCRETE

IASSUM
  INITIAL (0.) SCALAR MSP IAS SUMMER (KNOTS)

ILSZON
  INITIAL (OFF) BOOLEAN AP WITHIN ILS ZONE

LABFLG
  INITIAL (OFF) BOOLEAN ON=OPERATING IN HOT BENCH

LAT
  INITIAL (0.) SCALAR UPDATED LATITUDE (DEG).

LATCEN
  INITIAL (0.) SCALAR NORTH UP MAP CENTER-LAT (DEG).

LBS
  INITIAL (OFF) BOOLEAN LATERAL BEAM SENSED DISC

LOKBUF
  ARRAY (31) INTEGER INITIAL (0) LOOK-UP WPT/RTF BUFFER

LOKWPT
  ARRAY (2) INTEGER INITIAL (0) LOOK-UP WPT

LON
  INITIAL (0.) SCALAR UPDATED LONGITUDE (DEG).

LONCEN
  INITIAL (0.) SCALAR NORTH UP MAP CENTER-LON (DEG).

MAGVAR
  INITIAL (0.) SCALAR MAGNETIC VARIATION (DEG).

MFREQ
  INITIAL (OFF) BOOLEAN MFD UPDATE REQUEST

MXEPR
  INITIAL (2.23) SCALAR SELECTED EPR LIMIT

NAV64K
  INITIAL (OFF) BOOLEAN GS > 64 KNOTS

NAV2A
  INTEGER PTR TO NAV2 AUTO

NAV2A
  INTEGER PTR TO NAV3 AUTO

OFBIAS
  INITIAL (0.) SCALAR OFFSET BIAS (NM).

OFSEL
  INITIAL (OFF) BOOLEAN OFFSET MODE DISCRETE

ORIGIN
  ARRAY (4) INTEGER INITIAL (0) ORIGIN AIRFIELD AND RUNWAY

PFPA
  INITIAL (0.) SCALAR PLANNED FLIGHT PATH ANGLE (DEG).

PGBUF
  STRUCTURE THE PROVISIONAL GUIDANCE BUFFER

PGBUF.PWA02
  SCALAR ARC LENGTH OF CURVE/2 (FT).

PGBUF.PWASS
  INTEGER 0=AWY1=SID2=STAR3=MAP4=N

PGBUF.PWBRG
  SCALAR BEARING (DEG).

PGBUF.PWCCD
  SCALAR CENTER TO CENTER DIST (FT).

PGBUF.PWDMA
  BOOLEAN DME ARC?

PGBUF.PWDMI
  BOOLEAN DME ARC STOP CALC

PGBUF.PWDTT
  SCALAR TANGENT PT TO WPT DIST (FT).

PGBUF.PWFLL
  BOOLEAN FLIGHT LEVEL?

PGBUF.PWGM
  SCALAR PATH GRADIENT (DEG).
## DISNAV COMPOOL VARIABLES

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGBUF.PWH</td>
<td>SCALAR</td>
<td>ALTITUDE (FT).</td>
</tr>
<tr>
<td>PGBUF.PWRL</td>
<td>BOOLEAN</td>
<td>IAS REFERENCE?</td>
</tr>
<tr>
<td>PGBUF.PWLAT</td>
<td>SCALAR</td>
<td>LATITUDE (DEG).</td>
</tr>
<tr>
<td>PGBUF.PWLGRF</td>
<td>SCALAR</td>
<td>DME ARC REFERENCE LON (DEG).</td>
</tr>
<tr>
<td>GBUF.PWLON</td>
<td>SCALAR</td>
<td>LONGITUDE (DEG).</td>
</tr>
<tr>
<td>PGBUF.PWLTRF</td>
<td>SCALAR</td>
<td>DME ARC REFERENCE LAT (DEG).</td>
</tr>
<tr>
<td>PGBUF.PWMV</td>
<td>SCALAR</td>
<td>MAGNETIC VARIATION (DEG).</td>
</tr>
<tr>
<td>PGBUF.PWNAM</td>
<td>ARRAY (3) INTEGER</td>
<td>WPT NAME</td>
</tr>
<tr>
<td>PGBUF.PWPPD</td>
<td>SCALAR</td>
<td>CIRCLE PT TO PT DIST (FT).</td>
</tr>
<tr>
<td>PGBUF.PWPTA</td>
<td>SCALAR</td>
<td>PLANNED TIME OF ARRIVAL SET</td>
</tr>
<tr>
<td>PGBUF.PWRAD</td>
<td>BOOLEAN</td>
<td>RADIUS INHIBIT?</td>
</tr>
<tr>
<td>PGBUF.PWRTN</td>
<td>SCALAR</td>
<td>RADIUS OF TURN (FT).</td>
</tr>
<tr>
<td>PGBUF.PWT</td>
<td>SCALAR</td>
<td>TURN ANGLE AT WPT (DEG).</td>
</tr>
<tr>
<td>PGBUF.PWTPTR</td>
<td>INTEGER</td>
<td>ADDR OF VOR</td>
</tr>
<tr>
<td>PGBUF.PWV</td>
<td>SCALAR</td>
<td>GROUND SPEED (KNOTS)</td>
</tr>
<tr>
<td>PGBUF.PWAY</td>
<td>BOOLEAN</td>
<td>PROVISIONAL?</td>
</tr>
<tr>
<td>PTERM</td>
<td>INTEGER</td>
<td>0 WORD TERMINATOR TO PGBUF</td>
</tr>
<tr>
<td>PSTALT</td>
<td>INITIAL (OFF) BOOLEAN</td>
<td>PRE-SELECTED ALTITUDE DISCRETE</td>
</tr>
<tr>
<td>PSTFPA</td>
<td>INITIAL (OFF) BOOLEAN</td>
<td>FPA PRESELECT</td>
</tr>
<tr>
<td>PSTITAS</td>
<td>INITIAL (OFF) BOOLEAN</td>
<td>PRE-SELECTED IAS ENGAGE DISCRETE</td>
</tr>
<tr>
<td>PSTTKA</td>
<td>INITIAL (OFF) BOOLEAN</td>
<td>TRACK ANGLE PRESELECT</td>
</tr>
<tr>
<td>PTR2DN</td>
<td>INITIAL (0) INTEGER</td>
<td>2D PTR UPDATED AT BEGIN OF TURN</td>
</tr>
<tr>
<td>PWNUM</td>
<td>INITIAL (0) INTEGER</td>
<td># OF WPTS IN PBUFF</td>
</tr>
<tr>
<td>RADBRG</td>
<td>INITIAL (HEX '0')BIT(16)</td>
<td>BRG OF RADIAL WPT (DEG)</td>
</tr>
<tr>
<td>RADWPT</td>
<td>ARRAY(2) INTEGER INITIAL (0)</td>
<td>RADIAL WPT</td>
</tr>
<tr>
<td>RWPTAD</td>
<td>INITIAL (0) INTEGER</td>
<td>ADDR OF CIRCLE WPT</td>
</tr>
<tr>
<td>RWYHDG</td>
<td>INITIAL (0.) SCALAR</td>
<td>TRUE RUNWAY HEADING (DEG).</td>
</tr>
<tr>
<td>RWYLAT</td>
<td>INITIAL (0.) SCALAR</td>
<td>LAT OF RUNWAY THRESHOLD (DEG).</td>
</tr>
<tr>
<td>RYLEN</td>
<td>INITIAL (0.) SCALAR</td>
<td>LENGTH OF RUNWAY (FT).</td>
</tr>
<tr>
<td>RYELON</td>
<td>INITIAL (0.) SCALAR</td>
<td>LON OF RUNWAY THRESHOLD (DEG).</td>
</tr>
<tr>
<td>RYELEV</td>
<td>INITIAL (0.) SCALAR</td>
<td>RUNWAY ELEVATION (FT).</td>
</tr>
<tr>
<td>SINRH</td>
<td>INITIAL (0.) SCALAR</td>
<td>SIN (RWY HEADING)</td>
</tr>
<tr>
<td>SINUS0</td>
<td>INITIAL (HEX '0')BIT(16)</td>
<td>PACK SENSOR IN USE WORD 0</td>
</tr>
<tr>
<td>SINUS1</td>
<td>INITIAL (HEX '0')BIT(16)</td>
<td>PACK SENSOR IN USE WORD 1</td>
</tr>
<tr>
<td>SINUS2</td>
<td>INITIAL (HEX '0')BIT(16)</td>
<td>PACK SENSOR IN USE WORD 2</td>
</tr>
<tr>
<td>SINUS3</td>
<td>INITIAL (HEX '0')BIT(16)</td>
<td>PACK SENSOR IN USE WORD 3</td>
</tr>
<tr>
<td>TIMPTH</td>
<td>INITIAL (OFF) BOOLEAN</td>
<td>TIME PATH MODE SELECT</td>
</tr>
<tr>
<td>TK</td>
<td>INITIAL (0.) SCALAR</td>
<td>AIRPLANE TRUE TRACK (DEG).</td>
</tr>
<tr>
<td>TKA3SUM</td>
<td>INITIAL (0.) SCALAR</td>
<td>TRACK ANGLE SUMMER VALUE (FT).</td>
</tr>
<tr>
<td>TKSEL</td>
<td>INITIAL (OFF) BOOLEAN</td>
<td>TRACK HOLD MODE DISCRETE</td>
</tr>
<tr>
<td>VBS</td>
<td>INITIAL (OFF) BOOLEAN</td>
<td>VERTICAL BEAN SENSED FLAG</td>
</tr>
<tr>
<td>VERTPH</td>
<td>INITIAL (OFF) BOOLEAN</td>
<td>VERT PATH MODE DISCRETE</td>
</tr>
<tr>
<td>VGSDDOT</td>
<td>INITIAL (0.) SCALAR</td>
<td>ALONG TRACK ACCEL (FPS2)</td>
</tr>
<tr>
<td>WD</td>
<td>INITIAL (0.) SCALAR</td>
<td>WIND SPEED (KNOTS)</td>
</tr>
<tr>
<td>WPNUM</td>
<td>INITIAL (0.) INTEGER</td>
<td># OF WPTS OF ABUFF</td>
</tr>
<tr>
<td>WS</td>
<td>INITIAL (0.) SCALAR</td>
<td>CROSS TRACK ACCEL (FPS2)</td>
</tr>
<tr>
<td>XTACC</td>
<td>INITIAL (0.) SCALAR</td>
<td>CROSS TRACK ERROR (FT).</td>
</tr>
</tbody>
</table>
FCPOOL COMPOOL

FCPOOL is included in the NAVCOM resident common. FCPOOL contains variables that are referenced from the Flight Controls software.
FCPOOL COMPOOL VARIABLES

ACCHAT VECTOR
REPLACE XDDH BY "ACCHAT$(1)"
REPLACE YDDH BY "ACCHAT$(2)"
REPLACE ZDDH BY "ACCHAT$(3)"

ACCEL'N ESTIMATE VECTOR (FPS2)
X ACCELERATION
Y ACCELERATION
Z ACCELERATION

MLS ACCELERATIONS - XDD, YDD, ZDD

ATHCL APC LOWER LIMIT (DEG).

ATHCL APC RATE COMMAND (DPS).
ACWS/VCWS ROLL COMMANDED

TURN COORDINATOR SWITCH
False-ACWS/VCWS ONLY
True-ADD AUTOE

BODY MTD ACCEL FLAG
MLS CFILT RUN COUNTER

 MLS OUTLIER ERROR CNTRS

COSINE OF ROLL
COSINE OF PITCH

XTVEL SOURCE SWITCH
False - INS XRWY VEL
True - CXRVEL

COMPUTED XRWY VEL (FPS)
GAINED & FILTRD DCOL (IN)

MLS DELTA Y (FT).

YAW RELATIVE TO RUNWAY (DEG)

FAIL DISPLAYED FLAGS
1 - AFCS
2 - FFD
3 - MANEL
4 - ACWS
5 - VCWS
6 - AUTO
7 - LAND
8 - AUTO THROTTLE
9 - MLS

DISCRETE STATUS TABLE
VARIABLE COLUMN DEAD ZONE (IN)

EL IN USE SWITCH

GAIN PROGRMD LOCDEV (FT)

CFILT ERROR TERMS - EX, EY, EZ (FT).

SECOND FAILURE STATUS

DISCRETE STATUS TABLE
VARIABLE COLUMN DEAD ZONE (IN)
### FCPOOL COMPOOL VARIABLES (cont.)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIDENT</td>
<td>ARRAY(7) INTEGER</td>
<td>FAILURE ID TABLE</td>
</tr>
<tr>
<td>FLRSEL</td>
<td>BOOLEAN</td>
<td>FLARE CONTROL LAW SWITCH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FALSE - VAR Tau</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRUE - H(X)</td>
</tr>
<tr>
<td>FSIDX</td>
<td>INTEGER</td>
<td>FAIL STATUS INDEX POINTER</td>
</tr>
<tr>
<td>GAMD</td>
<td>SCALAR</td>
<td>GAMMA DOT (DPS)</td>
</tr>
<tr>
<td>GAMER</td>
<td>SCALAR</td>
<td>GAMC - (THETA OR GAMMA) (DEG).</td>
</tr>
<tr>
<td>GRD</td>
<td>BOOLEAN</td>
<td>ON-GROUND DISCRETE</td>
</tr>
<tr>
<td>HDCFSW</td>
<td>BOOLEAN</td>
<td>HDCF SOURCE SWITCH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FALSE - HDCF = FM HDOT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRUE - HDCF = FC HDCF</td>
</tr>
<tr>
<td>HDILS</td>
<td>SCALAR</td>
<td>HDOT = F(H,HDD) FOR FLARE (FPS)</td>
</tr>
<tr>
<td>HTDZ</td>
<td>SCALAR</td>
<td>MLS ALTITUDE (WHEEL HT) (FT)</td>
</tr>
<tr>
<td>INSST</td>
<td>INTEGER</td>
<td>INS SINGLE THREAD FLAG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - NORMAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - SELECT INS 'A'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - SELECT INS 'B'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - SELECT INS 'C'</td>
</tr>
<tr>
<td>KV</td>
<td>SCALAR</td>
<td>AIRSPEED GAIN</td>
</tr>
<tr>
<td>LIGHTS</td>
<td>INTEGER</td>
<td>STP LAMP CONTROL WORD</td>
</tr>
<tr>
<td>LOCVL</td>
<td>SCALAR</td>
<td>ETA/DELTA VARIABLE LIMIT</td>
</tr>
<tr>
<td>MCONF</td>
<td>BIT(16)</td>
<td>MLS CONFIGURATION WORD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 15 - MLS COMPUTE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 14 - REAL MLS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 13 - EL2 FLAG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 12 - BMACC FLAG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 11 - RADAR TRACK FLAG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 10 - VGS FLAG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 9 - 6 UNUSED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 5 - MLS SWITCH 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 4 - 1 UNUSED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 0 - MLS SWITCH 1</td>
</tr>
<tr>
<td>MSLSC</td>
<td>BOOLEAN</td>
<td>MLS COMPUTE SWITCH</td>
</tr>
<tr>
<td>MLSFC</td>
<td>INTEGER</td>
<td>MLS PACKED SIGNAL STATUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 15 - 13 UNUSED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 12 - RNGE OUTLIER VLD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 11 - AZ VALID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 10 - EL1 OUTLIER VALID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 9 - EL2 OUTLIER VALID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 8 - EX COUNT VALID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 7 - EY COUNT VALID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 6 - EZ COUNT VALID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 5 - 4 UNUSED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 3 - RANGE COUNT VALID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 2 - AZMTH COUNT VLD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 1 - EL1 COUNT VALID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIT 0 - EL2 COUNT VALID</td>
</tr>
<tr>
<td>MLMOD</td>
<td>BOOLEAN</td>
<td>MLSMOD SECONDARY</td>
</tr>
</tbody>
</table>

7 - LAND
8 - AUTO THROTTLE
9 - MLS
FCPOOL COMPOOL VARIABLES (cont.)

MLSSVC ARRAY(4) INTEGER MLS SIGNAL VALID COUNTERS
MODEX INTEGER CURRENT MODE INDEX
MSBUF ARRAY(19) INTEGER NEXT STP MESSAGE W/CTRL CHARS
MSGST INTEGER ADDRESS OF NEXT MSG TO STP
MSWIT INTEGER MSP MODE: 0-FLT, 1-PREFLT
MSW1 BOOLEAN MLS SWITCH 1
MSW2 BOOLEAN MLS SWITCH 2
MSW3 BOOLEAN MLS SWITCH 3
MSW4 BOOLEAN MLS SWITCH 4
MSW5 BOOLEAN MLS SWITCH 5
MSW6 BOOLEAN MLS SWITCH 6
NC102 SCALAR ATHCL WINDSHEAR FILTER (FPS2)
NC103 SCALAR ATHCL APC INTEGRATOR (DEG).
NC1 USC SCALAR ATHCL ACCEL DAMPING TERM (FPS2)
OTLERR ARRAY(4) SCALAR MLS SIGNAL OUTLIER ERRORS
PDTCMD SCALAR ROLL COMMAND RATE (DPS)
PHICMD SCALAR COMMANDED ROLL ANGLE (DEG)
PHIERR SCALAR PHICMD-ROLL (DEG)
PMSWIT INTEGER PREVIOUS VALUE OF MSWIT
POSHAT VECTOR POSITION ESTIMATE VECTOR (FT)

REPLACE XHAT BY "POSHAT$(1)" X POSITION
REPLACE YHAT BY "POSHAT$(2)" Y POSITION
REPLACE ZHAT BY "POSHAT$(3)" Z POSITION

QFB1 SCALAR WASHED OUT Q FOR CTL LAWS (DPS)
RADTKF BOOLEAN RADAR TRACK FLAG
RECWD1 INTEGER NORMAL RECORDING SELECT
RECWD2 INTEGER ALTERNATE RECORD SELECT

BITS SELECT GROUP
0-2 0 (VAR1)
3-5 1 (VAR2)
6-7 2 (DISC)

RLMLS BOOLEAN REAL MLS FLAG
RSWADR INTEGER ADRS OF RECORD SELECT DISCRETE
RWYSEL INTEGER RUNWAY SELECT INDEX

1 - WALLOPS
2 - NAFEC
3 - SPARE

SROLL SCALAR SINE OF ROLL
STFAIL ARRAY(26) INTEGER FAILURE STORED FLAGS
STHET SCALAR SINE OF PITCH
SWITCH INTEGER STP SWITCH STATUS WORD
SYNCL SCALAR LATRL TRIM VALUE (DEG).
TANGSA SCALAR TANGENT OF GLIDE SLOPE ANG
TRACK HOLD BOOLEAN VCWS TRACK HOLD ENABLED
VELHAT VECTOR VELOCITY ESTIMATE VECTOR (FPS)

REPLACE XDH BY "VELHAT$(1)" X VELOCITY
REPLACE YDH BY "VELHAT$(2)" Y VELOCITY
REPLACE ZDH BY "VELHAT$(3)" Z VELOCITY

VGAIN ARRAY(6) SCALAR DYNAMICALLY VARIABLE GAINS
WRDCNT INTEGER # OF WORDS TO OUTPUT TO STP
XTK1 SCALAR FIRST INT. OF XTACC (FPS)
<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XTK2</td>
<td>SCALAR</td>
<td>SECOND INT. OF XTACC (FT).</td>
</tr>
<tr>
<td>XYZER</td>
<td>VECTOR</td>
<td>MLS SOLUTION ERROR (FT).</td>
</tr>
</tbody>
</table>
FMBFCM is included in the IOCOM resident common. FMBFCM contains a block of 1156 words through which data is passed to and from the SIR memory. This input/output is done by FLTHDL. FLTHDL also stores in the 1156 word block the selection indices received from the Norden #1 SSFD (Signal Select/Failure Detect) software, for use by IOFLL in formatting the input SIU/RFDIU data.

FMBFCM also contains global variables used in FM/FC Software. There are 5 error counters set in FLTHDL that FLTEM checks to flag hardware/software errors. There is a display status word (at label DISPST) that is passed from the Norden #1, followed by A/C gross weight (WEIGHT) also computed in NORDEN #1. At DSPST2 + 2 is the SSFD toggle word, used to validate the selection indices received from Norden #1.

FMBFCM also contains space for the data recording buffer, DASBF, filled in by DASOT, the high rate data variables processed by IN1OM and OUT1OM, and assorted other variables that did not conveniently fit elsewhere.
### FMRFCM COMPOOL VARIABLES

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AILCMD</td>
<td>SCALAR INTEGER</td>
<td>AILERON COMMAND (DEG).</td>
</tr>
<tr>
<td>BF</td>
<td>INTEGER</td>
<td>0000H - 0483H OF SIR MEMORY.</td>
</tr>
<tr>
<td>BMACC</td>
<td>VECTOR</td>
<td>BODY MOUNTED ACCELEROMETERS. (FPS2)</td>
</tr>
<tr>
<td>CRFADD</td>
<td>INTEGER</td>
<td>HANDLER DATA FOR DEBUG.</td>
</tr>
<tr>
<td>CDATBF</td>
<td>INTEGER</td>
<td>HANDLER DATA FOR DEBUG.</td>
</tr>
<tr>
<td>COLDST</td>
<td>BOOLEAN</td>
<td>COLD START FLAG.</td>
</tr>
<tr>
<td>CWDCNT</td>
<td>INTEGER</td>
<td>HANDLER DATA FOR DEBUG.</td>
</tr>
<tr>
<td>DASBF</td>
<td>ARRAY (150) INTEGER</td>
<td>BOTTOM OF DAS RECORDING BUFFER.</td>
</tr>
<tr>
<td>DATADR</td>
<td>BIT (16)</td>
<td>OPTIONAL DUMP ADDRESS.</td>
</tr>
<tr>
<td>DATIN</td>
<td>ARRAY (35) INTEGER</td>
<td>50 MS MAJOR FRAME INPUT DATA.</td>
</tr>
<tr>
<td>DATOUT</td>
<td>ARRAY (106) INTEGER</td>
<td>START OF OUTPUT BUFFER.</td>
</tr>
<tr>
<td>DATSAV</td>
<td>ARRAY (30) INTEGER</td>
<td>DUMP DATA. (FROM DATADR)</td>
</tr>
<tr>
<td>DECMD</td>
<td>SCALAR INTEGER</td>
<td>ELEVATOR COMMAND (DEG).</td>
</tr>
<tr>
<td>DECMQ</td>
<td>SCALAR INTEGER</td>
<td>ELEVATOR COMMAND-PITCH RATE. (DEG).</td>
</tr>
<tr>
<td>DISPST</td>
<td>INTEGER</td>
<td>DISPLAY STATUS BITS.</td>
</tr>
<tr>
<td>DIUERR</td>
<td>INTEGER</td>
<td>NOT USED.</td>
</tr>
<tr>
<td>DSWST2</td>
<td>ARRAY (2) INTEGER</td>
<td>LAST 2 WORDS OF DISPLAYS STATUS.</td>
</tr>
<tr>
<td>EIUERR</td>
<td>INTEGER</td>
<td>EIU/SIU DMA ERROR.</td>
</tr>
<tr>
<td>EXPIN</td>
<td>ARRAY (98) INTEGER</td>
<td>INPUT DATA FOR EXPERIMENTS.</td>
</tr>
<tr>
<td>EXPOUT</td>
<td>ARRAY (4) INTEGER</td>
<td>OUTPUT DATA FOR EXPERIMENTS.</td>
</tr>
<tr>
<td>FOATIN</td>
<td>ARRAY (23) INTEGER</td>
<td>10 MS DATA INPUTS</td>
</tr>
<tr>
<td>FRAMES</td>
<td>SCALAR</td>
<td>RUNNING 10 MILL FRAME COUNT.</td>
</tr>
<tr>
<td>FRGSAV</td>
<td>ARRAY (6) SCALAR</td>
<td>ACO-AC5 SAVED</td>
</tr>
<tr>
<td>FR4NZ</td>
<td>INTEGER</td>
<td>HANDLER DATA FOR DEBUG.</td>
</tr>
<tr>
<td>FSTCNT</td>
<td>SCALAR</td>
<td>FAST LOOP 10 MILL FRAMES.</td>
</tr>
<tr>
<td>FSTINT</td>
<td>ARRAY (4) INTEGER</td>
<td>10 MS STATUS INPUTS</td>
</tr>
<tr>
<td>FSTSNA</td>
<td>BIT (16)</td>
<td>SNAP ADDRESS FOR FAST.</td>
</tr>
<tr>
<td>F4NRDY</td>
<td>INTEGER</td>
<td>HANDLER DATA FOR DEBUG.</td>
</tr>
<tr>
<td>HDD</td>
<td>SCALAR</td>
<td>VERTICAL ACCELERATION (FPS2).</td>
</tr>
<tr>
<td>INSSAV</td>
<td>BIT (16)</td>
<td>FIRST WORD OF SAVED INSTRUCTION.</td>
</tr>
<tr>
<td>INSSV2</td>
<td>BIT (16)</td>
<td>SECOND WORD OF SAVED INSTRUCTION.</td>
</tr>
<tr>
<td>INTERR</td>
<td>INTEGER</td>
<td>INTERPROCESSOR DMA ERROR.</td>
</tr>
<tr>
<td>INX2</td>
<td>ARRAY (40) INTEGER</td>
<td>BLOCK INDEX (0200H).</td>
</tr>
<tr>
<td>IPERR</td>
<td>INTEGER</td>
<td>HANDLER DATA FOR DEBUG.</td>
</tr>
<tr>
<td>KQ</td>
<td>SCALAR</td>
<td>GAIN FOR Q FEEDBACK.</td>
</tr>
<tr>
<td>MAXF</td>
<td>INTEGER</td>
<td>HIGHEST FRAME FOR FAST LOOP FINISH.</td>
</tr>
<tr>
<td>MFRAME</td>
<td>INTEGER</td>
<td>CURRENT FRAME (0 - 4 ).</td>
</tr>
<tr>
<td>MSINTP</td>
<td>INTEGER</td>
<td>INTERPROCESSOR MISSES.</td>
</tr>
<tr>
<td>MS10MML</td>
<td>INTEGER</td>
<td>10 MILL INTERRUPT MISSES.</td>
</tr>
<tr>
<td>MS55MML</td>
<td>INTEGER</td>
<td>50 MILL INTERRUPT MISSES.</td>
</tr>
<tr>
<td>NCDOUT</td>
<td>ARRAY (128) INTEGER</td>
<td>NC DU OUTPUT BUFFER.</td>
</tr>
<tr>
<td>NDAS</td>
<td>INTEGER</td>
<td>NUMBER OF ENTRIES IN DAS LIST.</td>
</tr>
<tr>
<td>OVER</td>
<td>INTEGER</td>
<td>CURRENT OVERFLOW COUNT.</td>
</tr>
<tr>
<td>P</td>
<td>SCALAR</td>
<td>ROLL RATE (DPS)</td>
</tr>
<tr>
<td>Q</td>
<td>SCALAR</td>
<td>PITCH RATE (DPS)</td>
</tr>
<tr>
<td>QX</td>
<td>SCALAR</td>
<td>Q WASHPOT BIAS (DPS)</td>
</tr>
<tr>
<td>R</td>
<td>SCALAR</td>
<td>YAW RATE (DPS)</td>
</tr>
<tr>
<td>REGSAV</td>
<td>ARRAY (9) INTEGER</td>
<td>RO - R7 AND FP STAT SAVED.</td>
</tr>
<tr>
<td>RFDIN</td>
<td>ARRAY (90) INTEGER</td>
<td>RFD IU INPUTS</td>
</tr>
<tr>
<td>RUDCMD</td>
<td>SCALAR</td>
<td>RUDDER COMMAND (DEG).</td>
</tr>
<tr>
<td>SAVCSR</td>
<td>INTEGER</td>
<td>HANDLER DATA FOR DEBUG.</td>
</tr>
</tbody>
</table>

47
<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLWCNT</td>
<td>SCALAR</td>
<td>SLOW LOOP 10 MILL FRAMES</td>
</tr>
<tr>
<td>SLWSEG</td>
<td>BIT (16)</td>
<td>SEGMENT NUM FOR SLOW SNAP</td>
</tr>
<tr>
<td>SLWSNA</td>
<td>BIT (16)</td>
<td>SNAP ADDRESS FOR SLOW</td>
</tr>
<tr>
<td>SNPFLG</td>
<td>BIT (16)</td>
<td>TRIGGER TO 'SNAP'</td>
</tr>
<tr>
<td>SPBPSW</td>
<td>INTEGER</td>
<td>SPBP CONTROL WORD (0292H)</td>
</tr>
<tr>
<td>STATIN</td>
<td>ARRAY (44) INTEGER</td>
<td>SIGNAL STATUS INPUTS FROM SSFD</td>
</tr>
<tr>
<td>SUMOVR</td>
<td>INTEGER</td>
<td>CUMULATIVE OVERFLOW COUNT</td>
</tr>
<tr>
<td>VDISC</td>
<td>INTEGER</td>
<td>VOTED DISCRETE WORD</td>
</tr>
<tr>
<td>WEIGHT</td>
<td>SCALAR</td>
<td>AIRCRAFT TOTAL WEIGHT (LBS)</td>
</tr>
</tbody>
</table>
FSTCOM is included in the FAST task. FSTCOM contains variables that are referenced only by routines in the FAST task.
FSTCOM COMPOOL VARIABLES

AZ BRG  SCALAR  AZIMUTH ANTENNA BEARING (DEG).
DELTAY  INTEGER  MODE ENGAGE DELAY COUNTER
ERINT   SCALAR  INT(K GAMD + K GAMER) (DEG).
ETAVL   SCALAR  LIMITED ETA (LOCDEV) (DEG).
HGPIP   SCALAR  MLS ELEV OF TD POINT (FT).
KALFA   SCALAR  F(ALPHA): ANTI-STALL GAIN
LAT MLS  SCALAR  AZIMUTH ANTENNA LATITUDE (DEG).
LON MLS  SCALAR  AZIMUTH ANTENNA LONGITUDE (DEG).
MLSVAL  BOOLEAN  MLS VALID FLAG
MODE2   INTEGER  LAST MODE INDEX
NCI01   SCALAR  ATHCL WINDSHEAR FILTER (FPS)
RwyDEF  ARRAY(3,8)SCALAR  RUNWAY CONSTANTS FOR ANY OF THREE POSSIBLE RUNWAYS. CONSTANTS INCLUDE VALUES FOR XFLARE, X GPIP, YPROF, H GPIP, ZO MLS, AZ BRG, LAT MLS, LON MLS
XFLARE  SCALAR  MLS-FLARE @ XGPIP + 884 (FT).
XGPIP   SCALAR  X DIST OF TD POINT (FT).
YPROF   SCALAR  MLS X-RWY CENTER LINE (FT).
ZDIF    SCALAR  ZO + EARTH CURVATURE (FT).
ZOMLS   SCALAR  AZIMUTH ANTENNA MSL ELEVATION (FT).
IOPOOL is included in the NAVCOM resident common. IOPOOL contains the input and output variables for the FM/FC computer. IOPOOL is organized with all the input variables first, followed by all the output variables. The input variables are filled in by IOFLL, using the selection indices and scaled data from the FMRFMC common pool. The output variables are sent to the FMBFCM common pool by OUTIO. High rate data handled by INIOM and OUTIOM are stored in FMBFCM.
IOPOOL COMPOOL VARIABLES

ACWSS BOOLEAN ACWS SELECT
AEE BOOLEAN AFCS ENGAGED (26F:0)
AFCSS BOOLEAN AFCS ENGAGE SELECT
AFCSV BOOLEAN AFCS VALID
AFDTH SCALAR AFD THROTL HDL POS (DEG).
AGCSS BOOLEAN AGCS SELECT
ALFAV SCALAR ANG OF ATTACK [ALPHA] (DEG)
ALTOMP ARRAY(2) INTEGER ALTITUDE TO CMP
ALVDT SCALAR AILERON POS (DEG)
APCDG SCALAR AUTO THROTL POS CMD (DEG)
ASTOMP ARRAY(2) INTEGER AIR SPEED TO CMP
ASTP SCALAR AUTO STAB TRIM POT (DEG)
ATDC BOOLEAN AUTO THROTTLE DISENG
ATE BOOLEAN AUTO THROTTLE ENGAGED
ATFDBK BOOLEAN AUTO THROTTLE FEEDBACK
ATKACC SCALAR ALONG TRK ACCEL (FPS2)
ATRIM SCALAR AFDAILERON TRIM POT (DEG)
ATUNE2 INTEGER DME #2 TUNE FREQUENCY (2X5)
ATUNE3 INTEGER DME #3 FREQ TUNE (2X5)
AUTOS BOOLEAN AUTO SELECT
BETAV SCALAR SIDESLIP ANGLE [BETA] (DEG)
BSLECT BOOLEAN 'B' SYSTEM SELECTED
CALTV BOOLEAN CAOC ALTITUDE VALID
CAS SCALAR CALIBRATED AIRSPEED (KTS)
CASY BOOLEAN CAOC AIRSPEED VALID
CDTISE BOOLEAN CDTI SELECTED
CRSET BOOLEAN COMRESET SELECT
DCOL SCALAR AFD COLUMN POSITION (IN)
DEPOS SCALAR ELEVATOR POS (DEG)
DME2A BOOLEAN DME #2 AUOTUNE
DME2FQ INTEGER DME #2 FREQ SELECTED (2X5)
DME2VD BOOLEAN DME#2 VALID
DME3FQ INTEGER DME #3 FREQ SELECTED (2X5)
DME3VD BOOLEAN DME#3 VALID
DRPOS SCALAR RUDDER SERVO POS (DEG)
DSBV BOOLEAN DIU SERIAL BUS VALID
DSTOMP ARRAY(2) INTEGER DISCRETES TO CMP
DWHL SCALAR AFD WHEEL POSITION (DEG)
EPR1 SCALAR #1 ENGINE PRESS RATIO
EPR2 SCALAR #2 ENGINE PRESS RATIO
ERSET BOOLEAN ERROR RESET ON MODE CHANGE
FALST INTEGER SYSTEM TEST PANEL SWITCH
FCOL SCALAR FFDS COLUMN FORCE (LBS)
FFDS BOOLEAN FWD FLIGHT DECK SELECT
FFLER INTEGER PARITY ERROR COUNTER
FLAGS ARRAY(18) BOOLEAN
    REPLACE PRENG BY "FLAGS$(1:)"      NOT O/P
    REPLACE FFDE BY "FLAGS$(2:)"        NOT O/P
    REPLACE MANEL BY "FLAGS$(3:)"       NOT O/P
    REPLACE ACWSE BY "FLAGS$(4:)" 0425:0+
    REPLACE VCWSE BY "FLAGS$(5:)" 0425:1+
IOPOOL COMPOOL VARIABLES (cont.)

```
REPLACE AUTOE BY "FLAGS$(6:)" 0425:2+
REPLACE LANDE BY "FLAGS$(7:)" 0425:3+
REPLACE DECRB BY "FLAGS$(8:)" 0425:4
REPLACE FLARE BY "FLAGS$(9:)" 0425:5
REPLACE RLOUT BY "FLAGS$(10:)" 0425:6
REPLACE LANDA BY "FLAGS$(11:)" 0425:7+
REPLACE LOCA BY "FLAGS$(12:)" 0425:8
REPLACE LOCE BY "FLAGS$(13:)" 0425:9
REPLACE GSARM BY "FLAGS$(14:)" 0425:10
REPLACE GSENG BY "FLAGS$(15:)" 0425:11
REPLACE LANDR BY "FLAGS$(16:)' NOT O/P
REPLACE GSTRK BY "FLAGS$(17:)" NOT O/P
REPLACE ONCRS BY "FLAGS$(18:)" NOT O/P
REPLACE GSARM BY "FLAGS$(14:)

FLAP SCALAR FFD FLAP HANDLE POS (DEG)
FLPPOS SCALAR FFD FLAP POSITION (DEG)
FPTOMP ARRAY(2) INTEGER FLIGHT PATH ANGLE TO CMP
FSBV BOOLEAN RFDIU (FUEL FLOW) S.B.VLD
FWHL SCALAR FFD WHEEL FORCE (LBS)
GAE BOOLEAN GO AROUND ENGAGED (0425:12)
GAS BOOLEAN GO AROUND SELECT NOT
GEAR BOOLEAN NOSE GEAR DOWN
GSDEV SCALAR GLIDESLOPE DEVIATION (DEG)
GSINS SCALAR GROUND SPEED (KTS)
GSVLD BOOLEAN ILS G/S RCVR VALID
HBARO SCALAR BARO ALTITUDE (FT)
HDOTB SCALAR CADC ALTITUDE RATE (FPS)
HOLD BOOLEAN HOLD MODE SELECT
HOLDM BOOLEAN SYSTEM HOLD MODE (0426:15)
HRAD SCALAR RADIO ALTITUDE (FT)
HRSS INTEGER DAYS/HOURS (BCD)
HRV BOOLEAN RADIO ALTITUDE VALID
IATTV BOOLEAN INS ATTITUDE VALID
IC BOOLEAN SYSTEM IC MODE (0426:13)
ICM BOOLEAN SYSTEM IC MODE (0426:13)
INAVV BOOLEAN INS NAVIGATION VALID
INSAER INTEGER PARITY ERROR COUNTER
INSBER INTEGER PARITY ERROR COUNTER
INSCER INTEGER PARITY ERROR COUNTER
ISBV BOOLEAN INS SERIAL BUS VALID
LAMP BOOLEAN LAMP TEST SELECT
LANDS BOOLEAN LAND SELECT
LATINS SCALAR INS LATITUDE (DEG)
LOCDEV SCALAR LOCALIZER DEVIATION (DEG)
LOCFS BOOLEAN LOCALIZER FREQ SLCTD
LOCVLQD BOOLEAN ILS LOC RCVR VALID
LONINS SCALAR INS LONGITUDE (DEG)
MACH SCALAR MACH NUMBER
MACHV BOOLEAN MACH VALID (NEW TRPLX DISC)
MAGHDG SCALAR MAGNETIC HEADING (DEG)
MDME2 SCALAR DME #2 DISTANCE (NM)
MDME3 SCALAR DME #3 DISTANCE (NM)
```
MDWARN BOOLEAN MODE REVERSION WARNING
MINS INTEGER MINS/SECS (BCD)
MLSER INTEGER PARITY ERROR COUNTER
MLSMOD BOOLEAN MLS VALID AND SELECTED
MLSRAW ARRAY(4) SCALAR MLS SIGNALS: R,AZ,L1,L2
MLSSLI BOOLEAN MLS SELECT
MLSSV ARRAY(4) BOOLEAN MLS SIGNAL VALID
MLSVLD BOOLEAN GATED MLS VALID
MSPER INTEGER PARITY ERROR COUNTER
MVINS SCALAR INS MAG VARIATION (DEG)
MVOR2 SCALAR VOR #2 BEARING (DEG)
NCDUER INTEGER PARITY ERROR COUNTER
NCDUL8 ARRAY(12) INTEGER NCDU #1 INPUTS (CHAR)
NCDUNP BOOLEAN NCDU #1 NEW PAGE DISC
NCUVAL BOOLEAN NCU VALID
NPAGE ARRAY(96) INTEGER NCDU #1 OUTPUTS (CHAR)
PEDAL SCALAR AFD RUD PED + TRIM (DEG)
PITCH SCALAR INS PITCH [THETA] (DEG)
RASELT BOOLEAN RANGE ALTITUDE SELECT [N/U]
RATOMP ARRAY(2) INTEGER RANGE/ALT TO CMP
ROLL SCALAR INS BANK ANGLE [PHI] (DEG)
RSTOMP ARRAY(2) INTEGER RANGE/SPEED TO CMP
RTRIM SCALAR AFD RUDDER TRIM POT (DEG)
RUN BOOLEAN RUN MODE SELECT
RUNM BOOLEAN SYSTEM RUN MODE (0426:14)
RWAYOUT BOOLEAN RUNWAY HEADING SET
SPFINH BOOLEAN SPLER FDBCK INHIBIT (0426:4)
SPI1 SCALAR DELTA PSI RWY [N/U] (DEG)
SPI2 SCALAR RUDDER SURF POS [N/U] (DEG)
SPL2 SCALAR SPOILER PAN #2 POS (DEG)
SPOB1 BOOLEAN NCDU #2 NEW PAGE DISC (NEWD)
SPOB2 BOOLEAN SPARE BOOLEAN (OLD YAWD1)
SRE7 SCALAR SPOILER PAN #7 POS (DEG)
SQUAT BOOLEAN OLEO SQUAT SWITCH
STABP SCALAR STABILIZER POS [N/U] (DEG)
STRU INTEGER SERVO OUTPUT
STRB INTEGER SERVO OUTPUT
STRUC INTEGER SERVO OUTPUT
TAS SCALAR TRUE AIR SPEED (KTS)
TASV BOOLEAN MACH/TAS VALID
TAT SCALAR TOTAL AIR TEMP (DEG-CENT)
TDSER INTEGER PARITY ERROR COUNTER
THDG SCALAR INS TRUE HEADING (DEG)
THROT SCALAR FFD THROTL HDL POS (DEG)
TKTOMP ARRAY(2) INTEGER TRACK ANGLE TO CMP
TOGGLE INTEGER NCDU/CMP INPUTS VALID IND
TRIMD BOOLEAN STAB TRIM DOWN (0426:1)
TRIMT BOOLEAN STAB TRIM RUN (0426:2)
TSBV BOOLEAN TDS SERIAL BUS VALID
VATRD BOOLEAN TRIM DOWN SELECT
VATRL BOOLEAN TRIM LEFT SELECT

54
<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VATRM</td>
<td>BOOLEAN</td>
<td>TRIM UP SELECT</td>
</tr>
<tr>
<td>VATRR</td>
<td>BOOLEAN</td>
<td>TRIM RIGHT SELECT</td>
</tr>
<tr>
<td>VCWSS</td>
<td>BOOLEAN</td>
<td>VCWS SELECT</td>
</tr>
<tr>
<td>VEINS</td>
<td>SCALAR</td>
<td>INS EAST VELOCITY (KTS)</td>
</tr>
<tr>
<td>VNINS</td>
<td>SCALAR</td>
<td>INS NORTH VELOCITY (KTS)</td>
</tr>
<tr>
<td>VORVLD</td>
<td>BOOLEAN</td>
<td>LOC RCVR 'B' VALID</td>
</tr>
<tr>
<td>WPTALR</td>
<td>BOOLEAN</td>
<td>WAYPOINT ALERT</td>
</tr>
<tr>
<td>WSPIN</td>
<td>BOOLEAN</td>
<td>WHEEL SPIN UP</td>
</tr>
<tr>
<td>XTKACC</td>
<td>SCALAR</td>
<td>INS CROSS TRACK ACCEL (FPS2)</td>
</tr>
<tr>
<td>XTVEL</td>
<td>SCALAR</td>
<td>INS CROSS RWY VELOCITY (FPS)</td>
</tr>
<tr>
<td>XYZ</td>
<td>VECTOR</td>
<td>RADAR TRACKING DATA (FT)</td>
</tr>
</tbody>
</table>
NAVCOM is included in the NAVCOM resident common. NAVCOM contains variables and initialized constants that are referenced from all the FM/FC tasks.
NAVCOM COMPOOL VARIABLES

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADMG</td>
<td>SCALAR</td>
<td>ARC DISTANCE MADE GOOD IN TURN (FT)</td>
</tr>
<tr>
<td>ALFSYN</td>
<td>SCALAR</td>
<td>SYNTH. ALFA [GAMMA - THETA] (DEG)</td>
</tr>
<tr>
<td>ALTGSF</td>
<td>BOOLEAN</td>
<td>ALT/GS =0. FLAG 2D/3D</td>
</tr>
<tr>
<td>AMG</td>
<td>SCALAR</td>
<td>ANGLE MADE GOOD (DEG)</td>
</tr>
<tr>
<td>AMG1</td>
<td>SCALAR</td>
<td>GOOD ANGLE IN TURN-TB (FT)</td>
</tr>
<tr>
<td>ANTLAT</td>
<td>SCALAR</td>
<td>LATITUDE OF ANTENNA (DEG)</td>
</tr>
<tr>
<td>ANTLon</td>
<td>SCALAR</td>
<td>LONGITUDE OF ANTENNA (DEG)</td>
</tr>
<tr>
<td>ATCMD</td>
<td>SCALAR</td>
<td>LONGITUDINAL ACC COMM (FPS2)</td>
</tr>
<tr>
<td>ATNAV2</td>
<td>BOOLEAN</td>
<td>AUTO TUNE NAV2</td>
</tr>
<tr>
<td>ATNAV3</td>
<td>BOOLEAN</td>
<td>AUTO TUNE NAV3</td>
</tr>
<tr>
<td>AVECTS.PWP1</td>
<td>VECTOR(3)</td>
<td>WAYPOINT UNIT VECTOR</td>
</tr>
<tr>
<td>AVECTS.PWU12</td>
<td>VECTOR(3)</td>
<td>NORMAL UNIT VECTOR</td>
</tr>
<tr>
<td>AVECTS.PWTC2</td>
<td>VECTOR(3)</td>
<td>CENTER OF TURN UNIT VECTOR</td>
</tr>
<tr>
<td>BACMD</td>
<td>SCALAR</td>
<td>BANK ANGLE COMMAND (DEG)</td>
</tr>
<tr>
<td>BADRAD</td>
<td>ARRAY(3)</td>
<td>BAD RADIUS WPT NAME</td>
</tr>
<tr>
<td>BCFLAG</td>
<td>BOOLEAN</td>
<td>BE CAREFUL GUID ALERT FG</td>
</tr>
<tr>
<td>CDME2</td>
<td>SCALAR</td>
<td>COMPUTED DME2 (NM)</td>
</tr>
<tr>
<td>CDME3</td>
<td>SCALAR</td>
<td>COMPUTED DME3 (NM)</td>
</tr>
<tr>
<td>CENWPT</td>
<td>ARRAY(2)</td>
<td>MAP CENTER WAYPOINT</td>
</tr>
<tr>
<td>CLAT</td>
<td>SCALAR</td>
<td>COS(LATITUDE)</td>
</tr>
<tr>
<td>CLON</td>
<td>SCALAR</td>
<td>COS(LONGITUDE)</td>
</tr>
<tr>
<td>COSTH</td>
<td>SCALAR</td>
<td>COS(AP HEAD)</td>
</tr>
<tr>
<td>COSTKA</td>
<td>SCALAR</td>
<td>COSINE OF TRACK ANGLE</td>
</tr>
<tr>
<td>CVOR2</td>
<td>SCALAR</td>
<td>COMPUTED VOR2 (DEG)</td>
</tr>
<tr>
<td>DELALT</td>
<td>SCALAR</td>
<td>ALTITUDE SELECT ERROR (FT)</td>
</tr>
<tr>
<td>DELCAS</td>
<td>SCALAR</td>
<td>AUTOTHROTTLE CAS ERROR (KTS)</td>
</tr>
<tr>
<td>DELTKA</td>
<td>SCALAR</td>
<td>TRACK ANGLE ERROR SELECT (DEG)</td>
</tr>
<tr>
<td>DFPA3D</td>
<td>SCALAR</td>
<td>3D GUIDANCE PATH ERROR (DEG)</td>
</tr>
<tr>
<td>DLT2</td>
<td>SCALAR</td>
<td>DOUBLE PREC. EXTN. OF IDDLAT</td>
</tr>
<tr>
<td>DLO2</td>
<td>SCALAR</td>
<td>DOUBLE PREC. EXTN. OF IDDLON</td>
</tr>
<tr>
<td>DMG</td>
<td>SCALAR</td>
<td>SEPARATION REFERENCE DISTANCE (FT)</td>
</tr>
<tr>
<td>DPE</td>
<td>SCALAR</td>
<td>EAST POS ERROR (NM)</td>
</tr>
<tr>
<td>DPN</td>
<td>SCALAR</td>
<td>NORTH POS ERROR (NM)</td>
</tr>
<tr>
<td>DTG</td>
<td>SCALAR</td>
<td>ABEAM PT DIST TO 'TO' WPT (FT)</td>
</tr>
<tr>
<td>DTOTL</td>
<td>SCALAR</td>
<td>PROGRESS DISTANCE OF AP WPT (FT)</td>
</tr>
<tr>
<td>DVE</td>
<td>SCALAR</td>
<td>EAST VEL ERROR (KTS)</td>
</tr>
<tr>
<td>DVN</td>
<td>SCALAR</td>
<td>NORTH VEL ERROR (KTS)</td>
</tr>
<tr>
<td>FIFTY</td>
<td>SCALAR</td>
<td>FIFTY MILL. COUNTER CLOCK</td>
</tr>
<tr>
<td>FLADM</td>
<td>BOOLEAN</td>
<td>AIR DATA MODE</td>
</tr>
<tr>
<td>FLMR</td>
<td>BOOLEAN</td>
<td>RADIO MODE ONLY</td>
</tr>
<tr>
<td>FLYFLG</td>
<td>BOOLEAN</td>
<td>O = REAL A/P</td>
</tr>
<tr>
<td>GPGSDV</td>
<td>SCALAR</td>
<td>G.P. GLIDE SLOPE DEVIATION</td>
</tr>
<tr>
<td>GPOCD</td>
<td>SCALAR</td>
<td>G.P. LOC DEVIATION</td>
</tr>
<tr>
<td>HDOT</td>
<td>SCALAR</td>
<td>RATE OF CHANGE OF ALT (FPS)</td>
</tr>
<tr>
<td>IASREF</td>
<td>SCALAR</td>
<td>IAS INIT REF (KTS)</td>
</tr>
<tr>
<td>IDDALT</td>
<td>SCALAR</td>
<td>INERTIALLY SMOOTHED BARO ALT (FT)</td>
</tr>
<tr>
<td>IDDATK</td>
<td>SCALAR</td>
<td>ALONG TK ACC VIA INS DATA (FPS2)</td>
</tr>
<tr>
<td>IDDG5</td>
<td>SCALAR</td>
<td>GS FROM INS DATA (KTS)</td>
</tr>
<tr>
<td>IDDLAT</td>
<td>SCALAR</td>
<td>LAT BASED ON INS DATA (DEG)</td>
</tr>
<tr>
<td>IDDLOLON</td>
<td>SCALAR</td>
<td>LON BASED ON INS DATA (DEG)</td>
</tr>
<tr>
<td>IDDXTK</td>
<td>SCALAR</td>
<td>CROSS TK ACC INS (FPS2)</td>
</tr>
</tbody>
</table>
**NAVCOM COMPOOL VARIABLES (cont.)**

- **KHD** SCALAR: VERTICAL PATH VEL. ERROR GAIN
- **KIP** SCALAR: POSITION GAIN
- **K2P** SCALAR: VELOCITY GAIN
- **LATCN** SCALAR: TRACK UP MAP CENTER-LAT (DEG)
- **LATDF** SCALAR: IDDLAT - MLSLAT (DEG)
- **LATSTR** SCALAR: LATERAL STEERING SIGNAL (DEG)
- **LLINIT** BOOLEAN: LAT/LON INIT ON NCDU
- **LONCN** SCALAR: TRACK UP MAP CENTER-LON (DEG)
- **LONDIF** SCALAR: IDDLON - MLSLON (DEG)
- **MCLEPR** SCALAR: EPR LMT. FOR MAX CLIMB THRUST
- **MCREPR** SCALAR: EPR LMT. FOR MAX CRUISE THRUST
- **MCTEPR** SCALAR: EPR LMT. FOR MAX CONT. THRUST
- **NAVFLG** BOOLEAN: GS > 4 KNOTS FLAG
- **NOMBA** SCALAR: NOMINAL BANK ANGLE (DEG)
- **NVAD2B** INTEGER: PTR TO NEXT NAV2 AUTO
- **PATHND** BOOLEAN: PATH END FLAG
- **PDG3D** INTEGER: 3D GUIDANCE FLAG
- **PDG4D** INTEGER: 4D GUIDANCE FLAG
- **PHISYM** SCALAR: SIMULATION ROLL COMMAND (DEG)
- **PTAREJ** BOOLEAN: REJECT PTA
- **PTAXFG** BOOLEAN: SET PTAPOS/PTATIME APPLY
- **PTH4DN** BOOLEAN: TIMEBOX HAS REACHED PATH END
- **PTR2D** INTEGER: 2D REFERENCE POINTER
- **PTR3D** INTEGER: 3D REFERENCE POINTER
- **PTR4D** INTEGER: AP 4D REFERENCE POINTER
- **PTR4DI** INTEGER: TIMEBOX 4D REFERENCE POINTER
- **PVECTS.PWP1** VECTOR (3): WAYPOINT UNIT VECTOR
- **PVECTS.PWTC2** VECTOR (3): CENTER OF TURN UNIT VECTOR
- **PVECTS.PWU12** VECTOR (3): NORMAL UNIT VECTOR
- **RADFT** SCALAR: LOCAL EARTH RADIUS (FT)
- **RALC** SCALAR: DISTANCE BEFORE TURN TO APPLY ALCBA (FT)
- **RASUM** SCALAR: RSP RANGE/ALT SUMMER (N/U)
- **RETUN2** INTEGER: RETUNE NAV 2 IF SET
- **RMP** SCALAR: LOCAL NORTH RAD OF CURV+ALT (NM)
- **RNP** SCALAR: RNH PROJECTED ON LAT PLANE (NM)
- **RWPTNM** ARRAY(3) INTEGER: WPT NAME CIRCLE
- **SC** SCALAR: DISTANCE FROM BOX TO WPT (FT)
- **SCMD** SCALAR: FLIGHT DIR SPEED CMD (FPS2)
- **SDC** SCALAR: PROGRESS PT VEL COMM (FPS)
- **SDCC** SCALAR: TIMEBOX VELOCITY (FPS)
- **SDD** SCALAR: PROGRESS PT ACC COMM (FPS2)
- **SEPR** SCALAR: DIST BETWEEN BOX AND AP (FT)
- **SIMALT** SCALAR: ALTITUDE OF SIM. A/P (FT)
- **SIMCAS** SCALAR: AIR SPEED SIM. AIRPLANE (KTS)
- **SIMFLG** BIT(16): SIMULATED AP ENGAGED

  - BIT 1 == IC AIRPLANE
  - BIT 2 == FLY AP (200KTS)
  - BIT 3 == FLY CAS OR TIME
  - BIT 4 ==
  - BIT 5 == HOLD AIRPLANE
  - BIT 6 ==
## NAVCOM COMPOOL VARIABLES (cont.)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMHDG</td>
<td>SCALAR</td>
<td>Heading of Sim A/P (deg)</td>
</tr>
<tr>
<td>SIMLAT</td>
<td>SCALAR</td>
<td>Latitude Sim. Airplane (deg)</td>
</tr>
<tr>
<td>SIMLON</td>
<td>SCALAR</td>
<td>Longitude Sim. Airplane (deg)</td>
</tr>
<tr>
<td>SINTH</td>
<td>SCALAR</td>
<td>Sin(Ap Head)</td>
</tr>
<tr>
<td>SINTKA</td>
<td>SCALAR</td>
<td>Sine of Track Angle</td>
</tr>
<tr>
<td>SLAT</td>
<td>SCALAR</td>
<td>Sin(Latitude)</td>
</tr>
<tr>
<td>SLON</td>
<td>SCALAR</td>
<td>Sin(Longitude)</td>
</tr>
<tr>
<td>SMMAGV</td>
<td>SCALAR</td>
<td>Sim. Airplane Magvar (deg)</td>
</tr>
<tr>
<td>SMUHDG</td>
<td>SCALAR</td>
<td>Init Heading for Sim A/P (deg)</td>
</tr>
<tr>
<td>TASFPS</td>
<td>SCALAR</td>
<td>True Air Speed (fps)</td>
</tr>
<tr>
<td>TASGS</td>
<td>SCALAR</td>
<td>GS from TAS (kts)</td>
</tr>
<tr>
<td>TEND</td>
<td>BOOLEAN</td>
<td>2D 2nd Half Turn Flag</td>
</tr>
<tr>
<td>TEND1</td>
<td>BOOLEAN</td>
<td>4D 2nd Half Turn Flag</td>
</tr>
<tr>
<td>TKE</td>
<td>SCALAR</td>
<td>Track Angle Error (deg)</td>
</tr>
<tr>
<td>TKMAG</td>
<td>SCALAR</td>
<td>Magnetic Track (deg)</td>
</tr>
<tr>
<td>TMEFLG</td>
<td>BOOLEAN</td>
<td>Time Flag</td>
</tr>
<tr>
<td>TOG100</td>
<td>BOOLEAN</td>
<td>100 M. SEC. Toggle Flag</td>
</tr>
<tr>
<td>TOSLOW</td>
<td>BOOLEAN</td>
<td>4D Speed Command &lt; 1ASREF</td>
</tr>
<tr>
<td>TOTIME</td>
<td>SCALAR</td>
<td>Clock Time of the WPT (sec)</td>
</tr>
<tr>
<td>TSMODE</td>
<td>INTEGER</td>
<td>NCDU Test Mode</td>
</tr>
<tr>
<td>TURN</td>
<td>BOOLEAN</td>
<td>2D In Turn Flag</td>
</tr>
<tr>
<td>TURN1</td>
<td>BOOLEAN</td>
<td>4D In Turn Flag</td>
</tr>
<tr>
<td>VACMD</td>
<td>SCALAR</td>
<td>Vert Acc Command (fps2)</td>
</tr>
<tr>
<td>VE</td>
<td>SCALAR</td>
<td>Velocity East (kts)</td>
</tr>
<tr>
<td>VERSTR</td>
<td>SCALAR</td>
<td>Vertical Steering Signal (fps2)</td>
</tr>
<tr>
<td>VN</td>
<td>SCALAR</td>
<td>Velocity North (kts)</td>
</tr>
<tr>
<td>VNAVF</td>
<td>INTEGER</td>
<td>Vertical Navigation Flag</td>
</tr>
<tr>
<td>VSTRA</td>
<td>SCALAR</td>
<td>Vert. Path Hdot Command (fps)</td>
</tr>
<tr>
<td>VSTRB</td>
<td>SCALAR</td>
<td>Verpth Hdot Actual (fps)</td>
</tr>
<tr>
<td>WPEND</td>
<td>INTEGER</td>
<td>Active Guidance Buffer End</td>
</tr>
<tr>
<td>WPPTR</td>
<td>ARRAY(2)</td>
<td>Path WP Pointer</td>
</tr>
<tr>
<td>XTKLIM</td>
<td>SCALAR</td>
<td>Horpth Capture Limit (ft)</td>
</tr>
</tbody>
</table>

7 == GOOD DME3
8 == GOOD DME2
9 == GOOD VOR2

59
RECOM is included in the NAVCOM resident common. RECOM contains variables for data recording; snap variables and DAS lists. These include a 450 word block, DASPAR, that is filled in by DSTAR. A maximum of 150 different variables may be selected for data recording and their addresses and scale or shift factor (total of 3 words) are kept in RECOM. DASOT takes these addresses and scales and puts the variables at location DASBF, in the FMBFCM compool. RECOM also contains a 288 word block, ALTPAR, that is also filled by DSTAR, and contains space for twelve, 8-item alternate strip chart recorder tables.
### RECOM COMPOOL VARIABLES

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTPAR</td>
<td>ARRAY(12,8,3) INTEGER</td>
</tr>
<tr>
<td>DASPAR</td>
<td>ARRAY(150,3) INTEGER</td>
</tr>
<tr>
<td>DISOUT</td>
<td>BIT(16)</td>
</tr>
<tr>
<td>DMLFC</td>
<td>BIT(16)</td>
</tr>
<tr>
<td>MLSOV</td>
<td>BIT(16)</td>
</tr>
<tr>
<td>RECWD</td>
<td>BIT(16)</td>
</tr>
<tr>
<td>RPTR</td>
<td>INTEGER</td>
</tr>
<tr>
<td>SCRIT1</td>
<td>ARRAY(20) INTEGER</td>
</tr>
<tr>
<td>SCRIT2</td>
<td>ARRAY(20) INTEGER</td>
</tr>
<tr>
<td>SCRIT3</td>
<td>ARRAY(20) INTEGER</td>
</tr>
<tr>
<td>SCRIT4</td>
<td>ARRAY(20) INTEGER</td>
</tr>
<tr>
<td>SCRIT5</td>
<td>ARRAY(20) INTEGER</td>
</tr>
<tr>
<td>SPTR</td>
<td>INTEGER</td>
</tr>
<tr>
<td>SRST</td>
<td>INTEGER</td>
</tr>
<tr>
<td>SDATA</td>
<td>ARRAY(4.16) SCALAR</td>
</tr>
<tr>
<td>SELWD1</td>
<td>BIT(16)</td>
</tr>
<tr>
<td>SELWD2</td>
<td>BIT(16)</td>
</tr>
<tr>
<td>SELWD3</td>
<td>BIT(16)</td>
</tr>
</tbody>
</table>
SLWCOM COMPOOL

SLWCOM is included in the SLOW task. SLWCOM contains variables that are referenced only from the NCDU software.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARSTAT</td>
<td>BOOLEAN</td>
<td>AIRBLEED STATUS</td>
</tr>
<tr>
<td>AM1, AM0</td>
<td>INTEGER</td>
<td>CONSTANT -1</td>
</tr>
<tr>
<td>A1, A2, A3, A4, A5, A6</td>
<td>INTEGER</td>
<td>CONSTANT 1, 2, 3, 4, 5, 6</td>
</tr>
<tr>
<td>ATCBOT</td>
<td>INTEGER</td>
<td>BOTTOM LINE OF ATC PAGE</td>
</tr>
<tr>
<td>ATCBUF</td>
<td>INTEGER</td>
<td>20 LINES OF ATC PAGE</td>
</tr>
<tr>
<td>ATCLN8</td>
<td>INTEGER</td>
<td>HOLDS ADDR. OF MESSAGE TO BE DISPLAYED ON LINE 8</td>
</tr>
<tr>
<td>ATCLUP</td>
<td>BOOLEAN</td>
<td>CLEARANCE NOT TERMINATED</td>
</tr>
<tr>
<td>BLANKS</td>
<td>INTEGER</td>
<td>8-BIT-ASCII BLANKS</td>
</tr>
<tr>
<td>CASKAL</td>
<td>SCALAR</td>
<td>&quot;CASCADED&quot; ALTITUDES (FT)</td>
</tr>
<tr>
<td>CASKFL</td>
<td>SCALAR</td>
<td>CASCADED FLIGHT LEVEL (FT/100)</td>
</tr>
<tr>
<td>CASKGS</td>
<td>SCALAR</td>
<td>CASCADED GROUND SPEED (KTS)</td>
</tr>
<tr>
<td>CINPFL</td>
<td>BOOLEAN</td>
<td>STATION SEARCH IN PROGRESS</td>
</tr>
<tr>
<td>COP</td>
<td>SCALAR</td>
<td>1/2 DIST. BETWEEN WAYPOINTS (FT)</td>
</tr>
<tr>
<td>COSCB</td>
<td>SCALAR</td>
<td>COS (COMPUTED BRG)</td>
</tr>
<tr>
<td>COSDA</td>
<td>SCALAR</td>
<td>COS (DEPRESSION ANGLE)</td>
</tr>
<tr>
<td>DECADR</td>
<td>INTEGER</td>
<td>DECODED WPT ADDRESS</td>
</tr>
<tr>
<td>DECBNG</td>
<td>SCALAR</td>
<td>DECODED WPT BEARING (DEG)</td>
</tr>
<tr>
<td>DECLAT</td>
<td>SCALAR</td>
<td>DECODED WPT LATITUDE (DEG)</td>
</tr>
<tr>
<td>DECLON</td>
<td>SCALAR</td>
<td>DECODED WPT LONGITUDE (DEG)</td>
</tr>
<tr>
<td>DECMV</td>
<td>SCALAR</td>
<td>DECODED WPT MAGNETIC VAR (DEG)</td>
</tr>
<tr>
<td>DECNAM</td>
<td>INTEGER</td>
<td>DECODED WPT NAME</td>
</tr>
<tr>
<td>DECRNG</td>
<td>SCALAR</td>
<td>DECODED WPT RANGE (FT)</td>
</tr>
<tr>
<td>DECTYP</td>
<td>INTEGER</td>
<td>DECODED WPT TYPE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = NAVAID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 = AIRPORT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 = GRP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 = WPT/BRG/RNG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 = LAT/LON WPT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 = PPOS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 = PPT(N)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 = RTE,AWY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 = SID,STAR</td>
</tr>
<tr>
<td>DECV</td>
<td>SCALAR</td>
<td>DECODED WPT VALUE</td>
</tr>
<tr>
<td>DELBR</td>
<td>SCALAR</td>
<td>VOR DELTA BRG (DEG)</td>
</tr>
<tr>
<td>DELGN</td>
<td>SCALAR</td>
<td>NAV ERROR GAIN</td>
</tr>
<tr>
<td>DLAM1</td>
<td>SCALAR</td>
<td>DME DELTA LONGITUDE (DEG)</td>
</tr>
<tr>
<td>DMERR</td>
<td>INTEGER</td>
<td>DME ERRORS LOGGED</td>
</tr>
<tr>
<td>DPHI1</td>
<td>SCALAR</td>
<td>DME DELTA LATITUDE (DEG)</td>
</tr>
<tr>
<td>DVECT</td>
<td>SCALAR</td>
<td>DIFFERENCE VECTOR</td>
</tr>
<tr>
<td>EPRFLG</td>
<td>INTEGER</td>
<td>0=MCTEPR, 1=MCLEPR, -1=MCREPR</td>
</tr>
<tr>
<td>FPPWP</td>
<td>INTEGER</td>
<td>MIDDLE WPT PTR IN GBUFF</td>
</tr>
<tr>
<td>FRMBUF</td>
<td>INTEGER</td>
<td>FORMATTED BUFFER</td>
</tr>
<tr>
<td>FRMNUM</td>
<td>INTEGER</td>
<td>NUMBER OF CHARACTERS FORMATTED</td>
</tr>
</tbody>
</table>

63
PRESENT POSITION LATITUDE (deg)
LAST WP NUMBER
VOR EF COS LOGGED
UPDATE TIME LIMIT
SCALAR
SCALAR
TRANSPOSE VECTOR
STATION 2 TIMES
STATION 1 TIMES
SELECTING POS IN ADUPE
SEL PAGE = 1, MOD, LINE 6
SEL PAGE = 2, MOD, WP,
SEL PAGE = 3, RADIAL
SEL PAGE = 3, BLANK
SEL PAGE = 3, CIRCLE
SEL PAGE = 2, OFFSET
SEL PAGE = 1, HOLD WP,
SIN (COMPUTED BAG)
CALC AND STORE PLA'S
REQUESTED OVERLAY SEGMENT
STATION VECTOR
PTR TO WP BUFFER
STATION POINTER
PATH DEF, STATION FLAG
A/P VECTOR
PATH DEF TIMER
TRANS STATUS RETURN
CALL POSEXT FLAG
MAKE PROW PATH ACTIVE
EXIT PATH DEF
GUIDANCE DONE
NUMBER OF PLA WP
NEW DESTINATION CHARGE
SCANTH ANO
LOOK UP ADDRESS
ACTIVE PAGE MODE
(SIN (MOD),
(SIN (MOD),
(COS (MOD),
(COS (MOD),
HALF BOOLEAN
GAMMA SALAR
HOLDIN INTEGER
HOLD IN INTEGER
I.S NAV UPDATING
VOR2 FALL PLAS
W3 FALL PLAS
4/P WITHIN DWG H/R CONE
DM2 FALL PLAS
4/P WITHIN DWG H/R CONE
DM3 FALL PLAS
FORMAT VALUE
SLWCOM COMPPOOL VARIABLES (cont.)
**SLWCOM COMPOOL VARIABLES (cont.)**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAYPNT.PPLOM</td>
<td>SCALAR Present Position Longitude (Deg)</td>
</tr>
<tr>
<td>WAYPNT.PPMSG</td>
<td>ARRAY(10) INTEGER NCDU Command Line for PPOS</td>
</tr>
<tr>
<td>WAYPNT.PPNAME</td>
<td>ARRAY(2) INTEGER Name of 'POS' Waypoint</td>
</tr>
<tr>
<td>WAYPNT.PPNAV</td>
<td>INTEGER Address of Assoc. Waypoint</td>
</tr>
<tr>
<td>WTERM</td>
<td>INTEGER O Word Terminator to Waypoint</td>
</tr>
<tr>
<td>XSTCTR</td>
<td>INTEGER Station Cycle Index</td>
</tr>
<tr>
<td>XSTMTR</td>
<td>SCALAR Cross Sta. Timer (4 Sec)</td>
</tr>
<tr>
<td>ZONLIM</td>
<td>INTEGER Zone Limit</td>
</tr>
<tr>
<td>ZONCTR</td>
<td>INTEGER Zone Counter (Cyclic)</td>
</tr>
<tr>
<td>ZONPTR</td>
<td>INTEGER Zone Pointer</td>
</tr>
</tbody>
</table>
I/O OVERVIEW

The I/O routines covered by this section handle the following:

Information coming into the NORDEN #2 computer from hardware sensors and switch/button settings. The interface to the hardware in this case is the SIU (Sensor Interface Unit).

Data to be output from the NORDEN #2 computer to control aircraft surfaces and panel lighting. The EIU (Effector Interface Unit) is the hardware device that interfaces to the NORDEN.

The EIU/SIU interface software uses the compool data block FMBFCM for block transfer of raw information. The actual I/O transfers to/from these units is accomplished by the system executive task (FLTHDL).

The input routines in this section (IOFLL, INIOM) take raw data from FMBFCM and format it to be directly usable by the NORDEN #2 software. Fixed point numeric values are made into floating point values in engineering units and packed discrete words are unpacked to form boolean variables. IOFLL handles data at the 50 millisecond rate, while INIOM takes care of the 10 millisecond cycle.

OUTIO and OUTIOM are the counterparts to IOFLL and INIOM. Values in formats usable by the NORDEN software are scaled/packed, and placed in the FMBFCM compool for transmission to the EIU.

See Table 3-1 for a list of the input signals. This includes signal name, units, source, scaling, SIR address, the number of sensors and other pertinent information.

Table 3-2 contains an analogous list of output signals.
MODULE NAME: IOFLL

PURPOSE: Format and fill IOPOOL with inputs from sensors.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: FAST

CALLING SEQUENCE: CALL IOFLL

CALLS TO: None

DESCRIPTION:

IOFLL is the link between the SIU (Sensor Interface Unit) and the input area of the IOPOOL compool. IOFLL formats raw sensor data obtained from the SIU via the DATAC. Several classes of raw input are handled:

A) Triplex data to be formatted to floating point.
B) Simplex data to be formatted to floating point.
C) Simplex data to be formatted to boolean.
D) Simplex data, stored across two consecutive memory cells, to be formatted to floating point.
E) Triplex data, stored across two consecutive memory cells, to be formatted to floating point and their respective valids stored as booleans.
F) Miscellaneous data: radar altitude, barometric altitude, flap position, and NCDU line #8 inputs.

Note that each triplex data item has its own selection index to specify which sensor will be used. The selection index is stored in FMBFCM compool each 50 millisecond frame. See documentation on SSFD module for more information on signal selection.

Because of the repetitive nature of the data formatting, IOFLL defines several MACROS to be used on the raw inputs. Each MACRO handles one class (A - E above) of data. The MACROS have parameters such as scaling and biases to define particular aspects of the data item to be formatted.

The miscellaneous items are unique data inputs. Flap handle position (FLAP) is found with a table look up on the raw data word. The raw data for radar altitude (HRAD) is duplex and is used to perform a table look up with interpolation followed by compensation for pitch angle to rotate the raw measurement to wheel height reference. Barometric altitude (HBARO) is computed in a local subroutine (HBSUB). A fine and coarse signal are combined, along with boundary error adjustments, to obtain the value. The NCDU input (NCDUL8) must be massaged to delete the label and reorder the characters to PDP memory positions. This is done also in a local subroutine (NCDIN). IOFLL also limits the minimum value of GSINS to 1 knot and computes TAS from CAS and HBARO whenever the raw value of TAS is less than 150 knots.
All the formatting done by IOFLL is repeated at a 50 millisecond rate. A small portion of the data inputs must be done at a 10 millisecond rate as described in the documentation for the module IN1OM.

GLOBAL INPUTS: FMBFCM compool (raw data from SIU).

GLOBAL OUTPUTS: The input section of IOPOOL compool (see IOPOOL documentation)
MODULE NAME: INIOM

PURPOSE: Format sensor inputs to fill IOPOOL

TASK: FLTHDL

LANGUAGE: MACRO-11

CALLED BY: FLTHDL

CALLING SEQUENCE: CALL INIOM

CALLS TO: None

DESCRIPTION:
Seven sensor inputs from the SIU (Sensor Interface Unit) need to be formatted at the 10 millisecond rate. These are handled by INIOM instead of IOFLL (see module documentation on IOFLL). Pitch, roll, and yaw rates, and INS vertical acceleration are all triplex fixed point binary data inputs that need to be scaled and converted to floating point. The X, Y, and Z components of the body mounted accelerometers additionally are filtered in their conversion to floating point.

A selection index, for each triplex data item, is stored in FMBFCM compool. This index is used by INIOM to determine which memory cell to use in the conversion. See module documentation for SSFD (in the DISPLAYS Computer Software Description document) for more information on selection indices.

GLOBAL INPUTS: FMBFCM raw data areas

GLOBAL OUTPUTS: P, Q, R, BMACC, HDD
### TABLE 3-1. - INPUT SIGNALS

<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>UNITS</th>
<th>SOURCE</th>
<th>SCALING</th>
<th>BIAS</th>
<th>SENSE</th>
<th>SENSOR A</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>DPS</td>
<td>RATE GYRO</td>
<td>104.5</td>
<td>-</td>
<td>NU</td>
<td>0001</td>
<td>3</td>
</tr>
<tr>
<td>P</td>
<td>DPS</td>
<td>RATE GYRO</td>
<td>104.5</td>
<td>-</td>
<td>RWD</td>
<td>0002</td>
<td>3</td>
</tr>
<tr>
<td>R</td>
<td>DPS</td>
<td>RATE GYRO</td>
<td>104.5</td>
<td>-</td>
<td>RT</td>
<td>0003</td>
<td>3</td>
</tr>
<tr>
<td>HDD</td>
<td>FPS2</td>
<td>INS ACC</td>
<td>102.4</td>
<td>-</td>
<td>UP</td>
<td>0004</td>
<td>3</td>
</tr>
<tr>
<td>BMACC</td>
<td>-AX</td>
<td>BMACC</td>
<td>64.23</td>
<td>17</td>
<td>FWD</td>
<td>0005</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>-AY</td>
<td>BMACC</td>
<td>64.08</td>
<td>-39</td>
<td>RT</td>
<td>000D</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>-AN</td>
<td>BMACC</td>
<td>101.23</td>
<td>-3005</td>
<td>DN</td>
<td>0015</td>
<td>1</td>
</tr>
</tbody>
</table>

### INPUT SIGNALS - 10 MSEC ANALOG (IN10M)

<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>UNITS</th>
<th>SOURCE</th>
<th>SCALING</th>
<th>BIAS</th>
<th>SENSE</th>
<th>SENSOR A</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>HRAD</td>
<td>FT</td>
<td>RADAR</td>
<td>4.1</td>
<td>-1946.</td>
<td>&gt;475'</td>
<td>002A</td>
<td>2</td>
</tr>
<tr>
<td>FWHL</td>
<td>LBS</td>
<td>FFD WHL</td>
<td>61.44</td>
<td>-</td>
<td>RWD</td>
<td>002B</td>
<td>3</td>
</tr>
<tr>
<td>HDOT</td>
<td>FPS</td>
<td>CADC</td>
<td>12.8</td>
<td>-</td>
<td>UP</td>
<td>002C</td>
<td>3</td>
</tr>
<tr>
<td>GSDEV</td>
<td>DEG</td>
<td>NAV RADIO</td>
<td>2048.</td>
<td>-</td>
<td>FLY DN</td>
<td>002D</td>
<td>3</td>
</tr>
<tr>
<td>LOCDEV</td>
<td>DEG</td>
<td>NAV RADIO</td>
<td>409.6</td>
<td>-</td>
<td>FLY RT</td>
<td>002E</td>
<td>3</td>
</tr>
<tr>
<td>RTRIM</td>
<td>UNITS</td>
<td>AFD TRIM</td>
<td>85.6</td>
<td>-</td>
<td>RT</td>
<td>002F</td>
<td>3</td>
</tr>
<tr>
<td>FCOL</td>
<td>LBS</td>
<td>FFD COL</td>
<td>61.44</td>
<td>-</td>
<td>UP</td>
<td>0030</td>
<td>3</td>
</tr>
<tr>
<td>DCOL</td>
<td>IN.</td>
<td>AFD COL</td>
<td>204.8</td>
<td>-</td>
<td>UP</td>
<td>0031</td>
<td>3</td>
</tr>
<tr>
<td>DWHL</td>
<td>DEG.</td>
<td>AFD WHL</td>
<td>40.96</td>
<td>-</td>
<td>RWD</td>
<td>0032</td>
<td>3</td>
</tr>
<tr>
<td>CAS</td>
<td>KNOTS</td>
<td>CADC</td>
<td>6.86</td>
<td>-412.</td>
<td>&gt;60 KTS</td>
<td>0033</td>
<td>3</td>
</tr>
<tr>
<td>FLAP</td>
<td>DEG</td>
<td>FFD FLP HDL</td>
<td>NON-</td>
<td>-850.</td>
<td>&gt;10 DEG</td>
<td>0034</td>
<td>3</td>
</tr>
<tr>
<td>SPL2</td>
<td>DEG</td>
<td>SPOILER</td>
<td>51.2</td>
<td>-</td>
<td>LWD</td>
<td>0035</td>
<td>3</td>
</tr>
<tr>
<td>SPR7</td>
<td>DEG</td>
<td>SERVO</td>
<td>-51.2</td>
<td>-</td>
<td>RWD</td>
<td>0036</td>
<td>3</td>
</tr>
<tr>
<td>PEDAL</td>
<td>IN.</td>
<td>AFD RUN</td>
<td>409.6</td>
<td>-</td>
<td>RT</td>
<td>0037</td>
<td>3</td>
</tr>
<tr>
<td>ATRIM</td>
<td>UNIT</td>
<td>AFD TRIM</td>
<td>204.8</td>
<td>-</td>
<td>RWD</td>
<td>0038</td>
<td>3</td>
</tr>
<tr>
<td>PITCH</td>
<td>DEG</td>
<td>INS</td>
<td>182.04</td>
<td>-</td>
<td>NU</td>
<td>0039</td>
<td>3</td>
</tr>
<tr>
<td>ROLL</td>
<td>DEG</td>
<td>INS</td>
<td>182.04</td>
<td>-</td>
<td>RWD</td>
<td>003A</td>
<td>3</td>
</tr>
<tr>
<td>MACH</td>
<td>-</td>
<td>CADC</td>
<td>65536. ±</td>
<td>-13107</td>
<td>&gt;.2</td>
<td>003C</td>
<td>3</td>
</tr>
<tr>
<td>HBARO</td>
<td>-HBC</td>
<td>FT</td>
<td>48545±</td>
<td>-</td>
<td>ABOVE MSL</td>
<td>003D</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>-HBF</td>
<td>FT</td>
<td>13.107±</td>
<td>-</td>
<td>MODULO 5000</td>
<td>003E</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>TAS</td>
<td>KTS</td>
<td>55.536±</td>
<td>-13107</td>
<td>&gt;200</td>
<td>003F</td>
<td>3</td>
</tr>
<tr>
<td>XTKACC</td>
<td>FPS2</td>
<td>INS</td>
<td>128</td>
<td>-</td>
<td>RT</td>
<td>0041</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>UNITS</th>
<th>SOURCE</th>
<th>SCALING</th>
<th>BIAS</th>
<th>SENSE</th>
<th>SENSOR A</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUT SIGNALS - 50 MSEC ANALOG (IOFLL)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>UNITS</th>
<th>SOURCE</th>
<th>SCALING</th>
<th>BIAS</th>
<th>SENSE</th>
<th>SENSOR A</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>HRAD</td>
<td>FT</td>
<td>RADAR</td>
<td>4.1</td>
<td>-1946.</td>
<td>&gt;475'</td>
<td>002A</td>
<td>2</td>
</tr>
<tr>
<td>FWHL</td>
<td>LBS</td>
<td>FFD WHL</td>
<td>61.44</td>
<td>-</td>
<td>RWD</td>
<td>002B</td>
<td>3</td>
</tr>
<tr>
<td>HDOT</td>
<td>FPS</td>
<td>CADC</td>
<td>12.8</td>
<td>-</td>
<td>UP</td>
<td>002C</td>
<td>3</td>
</tr>
<tr>
<td>GSDEV</td>
<td>DEG</td>
<td>NAV RADIO</td>
<td>2048.</td>
<td>-</td>
<td>FLY DN</td>
<td>002D</td>
<td>3</td>
</tr>
<tr>
<td>LOCDEV</td>
<td>DEG</td>
<td>NAV RADIO</td>
<td>409.6</td>
<td>-</td>
<td>FLY RT</td>
<td>002E</td>
<td>3</td>
</tr>
<tr>
<td>RTRIM</td>
<td>UNITS</td>
<td>AFD TRIM</td>
<td>85.6</td>
<td>-</td>
<td>RT</td>
<td>002F</td>
<td>3</td>
</tr>
<tr>
<td>FCOL</td>
<td>LBS</td>
<td>FFD COL</td>
<td>61.44</td>
<td>-</td>
<td>UP</td>
<td>0030</td>
<td>3</td>
</tr>
<tr>
<td>DCOL</td>
<td>IN.</td>
<td>AFD COL</td>
<td>204.8</td>
<td>-</td>
<td>UP</td>
<td>0031</td>
<td>3</td>
</tr>
<tr>
<td>DWHL</td>
<td>DEG.</td>
<td>AFD WHL</td>
<td>40.96</td>
<td>-</td>
<td>RWD</td>
<td>0032</td>
<td>3</td>
</tr>
<tr>
<td>CAS</td>
<td>KNOTS</td>
<td>CADC</td>
<td>6.86</td>
<td>-412.</td>
<td>&gt;60 KTS</td>
<td>0033</td>
<td>3</td>
</tr>
<tr>
<td>FLAP</td>
<td>DEG</td>
<td>FFD FLP HDL</td>
<td>NON-</td>
<td>-850.</td>
<td>&gt;10 DEG</td>
<td>0034</td>
<td>3</td>
</tr>
<tr>
<td>SPL2</td>
<td>DEG</td>
<td>SPOILER</td>
<td>51.2</td>
<td>-</td>
<td>LWD</td>
<td>0035</td>
<td>3</td>
</tr>
<tr>
<td>SPR7</td>
<td>DEG</td>
<td>SERVO</td>
<td>-51.2</td>
<td>-</td>
<td>RWD</td>
<td>0036</td>
<td>3</td>
</tr>
<tr>
<td>PEDAL</td>
<td>IN.</td>
<td>AFD RUN</td>
<td>409.6</td>
<td>-</td>
<td>RT</td>
<td>0037</td>
<td>3</td>
</tr>
<tr>
<td>ATRIM</td>
<td>UNIT</td>
<td>AFD TRIM</td>
<td>204.8</td>
<td>-</td>
<td>RWD</td>
<td>0038</td>
<td>3</td>
</tr>
<tr>
<td>PITCH</td>
<td>DEG</td>
<td>INS</td>
<td>182.04</td>
<td>-</td>
<td>NU</td>
<td>0039</td>
<td>3</td>
</tr>
<tr>
<td>ROLL</td>
<td>DEG</td>
<td>INS</td>
<td>182.04</td>
<td>-</td>
<td>RWD</td>
<td>003A</td>
<td>3</td>
</tr>
<tr>
<td>MACH</td>
<td>-</td>
<td>CADC</td>
<td>65536. ±</td>
<td>-13107</td>
<td>&gt;.2</td>
<td>003C</td>
<td>3</td>
</tr>
<tr>
<td>HBARO</td>
<td>-HBC</td>
<td>FT</td>
<td>48545±</td>
<td>-</td>
<td>ABOVE MSL</td>
<td>003D</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>-HBF</td>
<td>FT</td>
<td>13.107±</td>
<td>-</td>
<td>MODULO 5000</td>
<td>003E</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>TAS</td>
<td>KTS</td>
<td>55.536±</td>
<td>-13107</td>
<td>&gt;200</td>
<td>003F</td>
<td>3</td>
</tr>
<tr>
<td>XTKACC</td>
<td>FPS2</td>
<td>INS</td>
<td>128</td>
<td>-</td>
<td>RT</td>
<td>0041</td>
<td>3</td>
</tr>
</tbody>
</table>
### TABLE 3-1. - (CONTINUED)

<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>UNITS</th>
<th>SOURCE</th>
<th>SCALING</th>
<th>BIAS</th>
<th>SENSE</th>
<th>SENSOR A #</th>
<th>SENSORS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSINS</td>
<td>KTS</td>
<td>INS</td>
<td>5</td>
<td>-</td>
<td>ALWAYS</td>
<td>0043</td>
<td>3</td>
</tr>
<tr>
<td>XTVEL</td>
<td>FPS</td>
<td>INS</td>
<td>10.28</td>
<td>-</td>
<td>RT OF Rwy</td>
<td>0047</td>
<td>3</td>
</tr>
<tr>
<td>ATKACC</td>
<td>FPS2</td>
<td>INS</td>
<td>128</td>
<td>-</td>
<td>FWD</td>
<td>0049</td>
<td>3</td>
</tr>
<tr>
<td>VNINS</td>
<td>KTS</td>
<td>INS</td>
<td>10</td>
<td>-</td>
<td>NORTH</td>
<td>0048</td>
<td>3</td>
</tr>
<tr>
<td>VEINS</td>
<td>KTS</td>
<td>INS</td>
<td>10</td>
<td>-</td>
<td>EAST</td>
<td>004D</td>
<td>3</td>
</tr>
<tr>
<td>THDG</td>
<td>DEG</td>
<td>INS</td>
<td>182.04</td>
<td>-</td>
<td>EAST</td>
<td>004F</td>
<td>3</td>
</tr>
<tr>
<td>LATINS</td>
<td>DEG</td>
<td>INS</td>
<td>182.04</td>
<td>-</td>
<td>N. OR EQ.</td>
<td>0051</td>
<td>3</td>
</tr>
<tr>
<td>LONINS</td>
<td>DEG</td>
<td>INS</td>
<td>182.04</td>
<td>-</td>
<td>E. OF GRWCH 0053</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>MVINS</td>
<td>DEG</td>
<td>INS</td>
<td>182.04</td>
<td>-</td>
<td>THDG RT.</td>
<td>0055</td>
<td>3</td>
</tr>
<tr>
<td>MLSRAW</td>
<td>MLS RCVR</td>
<td>.16458</td>
<td>-</td>
<td>ALWAY S</td>
<td>00EC</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>- RANGE</td>
<td>MLS RCVR</td>
<td>200.0</td>
<td>-</td>
<td>LFT OF CTR</td>
<td>00FO</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>- EL1</td>
<td>MLS RCVR</td>
<td>200.0</td>
<td>-</td>
<td>ABOVE RWY</td>
<td>00F4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>- EL2</td>
<td>MLS RCVR</td>
<td>443.41</td>
<td>-</td>
<td>ABOVE RWY</td>
<td>00F8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>STABP</td>
<td>DEG</td>
<td>STABILIZER</td>
<td>96.67</td>
<td>372</td>
<td>&gt;7.25(ND)</td>
<td>0089</td>
<td>1 (N/U)*</td>
</tr>
<tr>
<td>TAT</td>
<td>DEG-C</td>
<td>TEMP PROBE</td>
<td>20.06</td>
<td>-</td>
<td>POS TEMP</td>
<td>0109</td>
<td>1</td>
</tr>
<tr>
<td>BETAV</td>
<td>DEG</td>
<td>BETA VANE</td>
<td>35.31</td>
<td>-</td>
<td>NOSE RT.</td>
<td>008B</td>
<td>1 (N/U)</td>
</tr>
<tr>
<td>ALFAV</td>
<td>DEG</td>
<td>L. ALPHA VANE</td>
<td>34.13</td>
<td>529</td>
<td>&lt;15.5</td>
<td>010A</td>
<td>1</td>
</tr>
<tr>
<td>- ALV1</td>
<td>DEG</td>
<td>R. ALPHA VANE</td>
<td>34.13</td>
<td>34</td>
<td>&lt;1.0</td>
<td>010B</td>
<td>1</td>
</tr>
<tr>
<td>DEPOS</td>
<td>DEG</td>
<td>ELEVATOR PCU</td>
<td>102.4</td>
<td>-</td>
<td>ND</td>
<td>008C</td>
<td>1</td>
</tr>
<tr>
<td>ALVDT</td>
<td>DEG</td>
<td>AILERON PCU</td>
<td>102.4</td>
<td>-</td>
<td>RWD</td>
<td>010C</td>
<td>1</td>
</tr>
<tr>
<td>THRTO</td>
<td>DEG</td>
<td>FFD LF THROT</td>
<td>19.25</td>
<td>140</td>
<td>&lt;7.29</td>
<td>010F</td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>DEG</td>
<td>FFD RT THROT</td>
<td>19.66</td>
<td>40</td>
<td>&lt;2.02</td>
<td>0110</td>
<td>1</td>
</tr>
<tr>
<td>MDME2</td>
<td>NM</td>
<td>NAV RCVR</td>
<td>100</td>
<td>-</td>
<td>ALWAYS</td>
<td>0095</td>
<td>1</td>
</tr>
<tr>
<td>MDME3</td>
<td>NM</td>
<td>NAV RCVR</td>
<td>100</td>
<td>-</td>
<td>ALWAYS</td>
<td>0115</td>
<td>1</td>
</tr>
<tr>
<td>ASTP</td>
<td>DEG</td>
<td>STAB TRIMPOT</td>
<td>20.48</td>
<td>-</td>
<td>TRIMUP</td>
<td>0118</td>
<td>1</td>
</tr>
<tr>
<td>FLPPPOS</td>
<td>DEG</td>
<td>FLAP SERVO</td>
<td>NON-LIN</td>
<td>-1687</td>
<td>NEVER</td>
<td>016B</td>
<td>1 (N/U)</td>
</tr>
<tr>
<td>AFDTH</td>
<td>DEG</td>
<td>AFD THROT</td>
<td>20.48</td>
<td>-</td>
<td>ALWAYS</td>
<td>016C</td>
<td>1 (N/U)</td>
</tr>
<tr>
<td>MAGHDG</td>
<td>DEG</td>
<td>SHIP COMPASS</td>
<td>182.04</td>
<td>-</td>
<td>EAST</td>
<td>016B</td>
<td>1</td>
</tr>
<tr>
<td>MVOR2</td>
<td>DEG</td>
<td>NAV RCVR</td>
<td>182.04</td>
<td>-21845</td>
<td>&gt;60</td>
<td>019A</td>
<td>0</td>
</tr>
<tr>
<td>EPR1</td>
<td>-</td>
<td>LFT ENG.</td>
<td>2171.8</td>
<td>3594</td>
<td>&lt;1.65</td>
<td>01DC</td>
<td>1</td>
</tr>
<tr>
<td>EPR2</td>
<td>-</td>
<td>RT ENG.</td>
<td>2149.0</td>
<td>-3557</td>
<td>&gt;1.65</td>
<td>01DD</td>
<td>1</td>
</tr>
</tbody>
</table>

**NOTES:**
* NOT USED BY FLIGHT SOFTWARE, BUT IS CHECKED BY SSFD AND/OR PREFLIGHT SOFTWARE
+ 14 BIT SYNCHRO - DIGITAL INPUT: ACTUAL RESOLUTION IS 1/4 THAT SHOWN AS 2 LSB's ARE ZERO.
<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>TYPE</th>
<th>SOURCE</th>
<th>INPUT BIT(S)</th>
<th>SENSOR A ADDR</th>
<th># SENSORS</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOGGLE</td>
<td>BIT</td>
<td>SIU</td>
<td>0-15</td>
<td>0021</td>
<td>1</td>
<td>Based on NCDU/MSP inputs</td>
</tr>
<tr>
<td>HRV</td>
<td>BOOLEAN</td>
<td>RADAR</td>
<td>0</td>
<td>0022</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>DME2VD</td>
<td>BOOLEAN</td>
<td>NAV RCVR</td>
<td>1</td>
<td>0022</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DME3VD</td>
<td>BOOLEAN</td>
<td>NAV RCVR</td>
<td>1</td>
<td>0122</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>LANDS</td>
<td>BOOLEAN</td>
<td>MSP</td>
<td>2</td>
<td>0022</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>AUTOS</td>
<td>BOOLEAN</td>
<td>MSP</td>
<td>3</td>
<td>0022</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>VCWSS</td>
<td>BOOLEAN</td>
<td>MSP</td>
<td>4</td>
<td>0022</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>ACWSS</td>
<td>BOOLEAN</td>
<td>MSP</td>
<td>5</td>
<td>0022</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>GSVLD</td>
<td>BOOLEAN</td>
<td>NAV RCVR</td>
<td>6</td>
<td>0022</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>LOCVLD</td>
<td>BOOLEAN</td>
<td>NAV RCVR</td>
<td>7</td>
<td>0022</td>
<td>3</td>
<td>AND'ED W/LOCFS</td>
</tr>
<tr>
<td>VORVLD</td>
<td>BOOLEAN</td>
<td>NAV RCVR</td>
<td>7</td>
<td>0022</td>
<td>3</td>
<td>ALWAYS FALSE</td>
</tr>
<tr>
<td>LAMP</td>
<td>BOOLEAN</td>
<td>MSP</td>
<td>8</td>
<td>0022</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>IATTV</td>
<td>BOOLEAN</td>
<td>INS</td>
<td>9</td>
<td>0022</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CASV</td>
<td>BOOLEAN</td>
<td>CADC</td>
<td>10</td>
<td>0022</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CALTV</td>
<td>BOOLEAN</td>
<td>CADC</td>
<td>11</td>
<td>0022</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>LOCF5</td>
<td>BOOLEAN</td>
<td>NAV RCVR</td>
<td>12</td>
<td>0022</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>INAVV</td>
<td>BOOLEAN</td>
<td>INS</td>
<td>13</td>
<td>0022</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>MACHV</td>
<td>BOOLEAN</td>
<td>CADC</td>
<td>14</td>
<td>0022</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>TASV</td>
<td>BOOLEAN</td>
<td>CADC</td>
<td>15</td>
<td>0022</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>IC</td>
<td>BOOLEAN</td>
<td>FC PALLET</td>
<td>0</td>
<td>0023</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>RUN</td>
<td>BOOLEAN</td>
<td>FC PALLET</td>
<td>1</td>
<td>0023</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>HOLD</td>
<td>BOOLEAN</td>
<td>FC PALLET</td>
<td>2</td>
<td>0023</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>CRSET</td>
<td>BOOLEAN</td>
<td>FC PALLET</td>
<td>3</td>
<td>0023</td>
<td>1</td>
<td>ALSO SET BY MLOG</td>
</tr>
<tr>
<td>ERSET</td>
<td>BOOLEAN</td>
<td>FC PALLET</td>
<td>4</td>
<td>0023</td>
<td>1</td>
<td>ALSO SET BY MLOG</td>
</tr>
<tr>
<td>MINS</td>
<td>BCD</td>
<td>SYSTEM</td>
<td>0-15</td>
<td>0024</td>
<td>1</td>
<td>MIN'S/SECS (4 DIGITS)</td>
</tr>
<tr>
<td>HRSS</td>
<td>BCD</td>
<td>SYSTEM</td>
<td>0-15</td>
<td>00A4</td>
<td>1</td>
<td>(2 DIGITS)</td>
</tr>
<tr>
<td>DME2A</td>
<td>BOOLEAN</td>
<td>NAV SELSW</td>
<td>0</td>
<td>0124</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ATDC</td>
<td>BOOLEAN</td>
<td>AFD THROT</td>
<td>1</td>
<td>0124</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>GEAR</td>
<td>BOOLEAN</td>
<td>NOSE GEAR</td>
<td>2</td>
<td>0124</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>MLSSLI</td>
<td>BOOLEAN</td>
<td>NAV PALLET</td>
<td>3</td>
<td>0124</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>RASELT</td>
<td>BOOLEAN</td>
<td></td>
<td>5</td>
<td>0124</td>
<td>0</td>
<td>NOT USED</td>
</tr>
<tr>
<td>ATFDKB</td>
<td>BOOLEAN</td>
<td>AFD THROT</td>
<td>6</td>
<td>0124</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>CDTI</td>
<td>BOOLEAN</td>
<td></td>
<td>7</td>
<td>0124</td>
<td>0</td>
<td>NOT USED</td>
</tr>
<tr>
<td>FALST</td>
<td>BIT</td>
<td>STR</td>
<td>8-15</td>
<td>0124</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DME2FQ</td>
<td>2X5 CODE</td>
<td>NAV RCVR</td>
<td>0-15</td>
<td>0026</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DME3FQ</td>
<td>2X5 CODE</td>
<td>NAV RCVR</td>
<td>0-15</td>
<td>00A6</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

TABLE 3-1. - (CONTINUED)
### INPUT SIGNALS - 50 MSEC BINARY (IOFLL)

<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>TYPE</th>
<th>SOURCE</th>
<th>INPUT BIT(S)</th>
<th>SENSOR A ADDR</th>
<th>SENSORS</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGCSS</td>
<td>BOOLEAN</td>
<td>FC PALLET</td>
<td>0</td>
<td>0126</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>GAS</td>
<td>BOOLEAN</td>
<td>AFD THROT</td>
<td>1</td>
<td>0126</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>WSPIN</td>
<td>BOOLEAN</td>
<td>MAIN GEAR</td>
<td>2</td>
<td>0126</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>BSLECT</td>
<td>BOOLEAN</td>
<td>FFD CCP</td>
<td>3</td>
<td>0126</td>
<td>1</td>
<td>NOT USED</td>
</tr>
<tr>
<td>VATRD</td>
<td>BOOLEAN</td>
<td>AFD BROLLY</td>
<td>4</td>
<td>0126</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>AFCSS</td>
<td>BOOLEAN</td>
<td>FFD CCP</td>
<td>6</td>
<td>0126</td>
<td>1</td>
<td>ALWAYS TRUE</td>
</tr>
<tr>
<td>AFCSS</td>
<td>BOOLEAN</td>
<td>FFD CCP</td>
<td>7</td>
<td>0126</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>FFDS</td>
<td>BOOLEAN</td>
<td>OLEO STRUTS</td>
<td>8</td>
<td>0126</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SQUAT</td>
<td>BOOLEAN</td>
<td>AFD BROLLY</td>
<td>10</td>
<td>0126</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>VATRM</td>
<td>BOOLEAN</td>
<td>AFD BROLLY</td>
<td>11</td>
<td>0126</td>
<td>1</td>
<td>NOT USED</td>
</tr>
<tr>
<td>VATRR</td>
<td>BOOLEAN</td>
<td>AFD BROLLY</td>
<td>12</td>
<td>0126</td>
<td>1</td>
<td>NOT USED</td>
</tr>
<tr>
<td>VATRL</td>
<td>BOOLEAN</td>
<td>SIU</td>
<td>0-2</td>
<td>0117</td>
<td>3</td>
<td>BITS VOTED BY DISFD</td>
</tr>
<tr>
<td>ISBV</td>
<td>BOOLEAN</td>
<td>SIU</td>
<td>3</td>
<td>0117</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DSBV</td>
<td>BOOLEAN</td>
<td>SIU</td>
<td>4</td>
<td>0117</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>TSBV</td>
<td>BOOLEAN</td>
<td>SIU</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSBV</td>
<td>BOOLEAN</td>
<td>SIU</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTE**: FAILURE OCCURRENCES COUNTED IN MLSER, TDSER, FFLER, RESPECTIVELY BY IOFLL, BUT OTHERWISE UNUSED.

---

**TABLE 3-1. - (CONTINUED)**
MODULE NAME: OUTIO

PURPOSE: Formal IOPOOL outputs for DATAC transmission to EIU.

TASK: FAST

LANGUAGE: MACRO-11

 CALLED BY: FAST

 CALLING SEQUENCE: CALL OUTIO

 CALLS TO: None

DESCRIPTION:

OUTIO is the link between the output area of IOPOOL compool and the EIU (Effector Interface Unit). PDP machine formatted data is reformatted to fixed point binary and packed discrete words to be sent via DATAC to the EIU. Most of the formatting is done with MACROS defined in OUTIO. Each has a set of parameters to specify certain characteristics of the particular data item being formatted.

The NCDU page output is accomplished with a local subroutine (NCDOUT). The ASCII data stored at NPAGE is rearranged around label bytes and stored in the output area to be sent to the EIU.

Several software tests of the EIU/SIU hardware are initiated in OUTIO. A discrete and analog set of values is output to the EIU to be wrapped back through the SIU (Sensor Interface Unit) and tested in FAILCP for hardware transmission errors.

All the formatting done by OUTIO is repeated at a 50 millisecond rate. Several outputs must be repeated every 10 milliseconds. See documentation for the module OUTIO for information about these data items.

GLOBAL INPUTS: Output area of IOPOOL compool (see IOPOOL Description)

GLOBAL INPUTS: FMBFCM compool (raw data to SIU)
MODULE NAME: OUTIOM

PURPOSE: Format IOPOOL outputs for transmission to EIU

TASK: FLTHDL

LANGUAGE: MACRO-11

CALLED BY: FLTHDL

CALLING SEQUENCE: CALL OUTIOM

CALLS TO: None

DESCRIPTION:

Three data items must be formatted and output to the EIU (Effector Interface Unit) at a 10 millisecond rate. These items are done by OUTIOM instead of OUTIO (the 50 millisecond output formatter). Rudder, aileron, and elevator commands are formatted to fixed point integer using a MACRO defined to accept parameters for individual items.

Elevator Command (DECMD) output requires special handling as the pitch rate (Q) feedback is applied to the basic elevator command (DECMO) in this routine. The equation implemented is:

\[ DECMD = DECMO + (Q - QX) \times KQ \]

where QX is a 16 second lag on Q (computed in ELEVP) used to wash out possible bias on the pitch rate signal, and KQ is the mode dependent pitch rate gain also computed in ELEVP. The Digital Systems Diagram for this operation is shown in the NSD for ELEVP.

GLOBAL INPUTS: RUDCMD, AILCMD, DECMO, Q, QX, KQ

GLOBAL OUTPUTS: FMBFCM raw data areas, DECMD
### OUTPUT SIGNALS —— 10 MSEC ANALOG — (OUT10M)

<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>UNITS</th>
<th>DESTIN.</th>
<th>SCALING</th>
<th>BIAS</th>
<th>SENSE</th>
<th>'A' OUTPUT</th>
<th>'B' OUTPUT</th>
<th>SIR ADDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUDCMD</td>
<td>DEG</td>
<td>Rudder</td>
<td>81.92</td>
<td>-</td>
<td>YAW RT</td>
<td>0401</td>
<td>0409</td>
<td></td>
</tr>
<tr>
<td>AILCMD</td>
<td>DEG</td>
<td>Aileron</td>
<td>102.4</td>
<td>-</td>
<td>LWD</td>
<td>0402</td>
<td>040A</td>
<td></td>
</tr>
<tr>
<td>DECMD</td>
<td>DEG</td>
<td>Elevator</td>
<td>102.4</td>
<td>-</td>
<td>ND</td>
<td>0403</td>
<td>040B</td>
<td></td>
</tr>
</tbody>
</table>

### OUTPUT SIGNALS —— 50 MSEC ANALOG — (OUT10)

<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>UNITS</th>
<th>DESTIN.</th>
<th>SCALING</th>
<th>BIAS</th>
<th>SENSE</th>
<th>OUTPUT</th>
<th>SIR ADDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>APCDG</td>
<td>DEG</td>
<td>Throttle</td>
<td>20.48</td>
<td>-</td>
<td>ALWAYS NEG.</td>
<td>040C</td>
<td></td>
</tr>
<tr>
<td>REF</td>
<td>VOLTS</td>
<td>SIU/EIU</td>
<td>204.8</td>
<td>*SAWTOOTH FOR TEST</td>
<td>042F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RWYHDG</td>
<td>DEG</td>
<td>INS</td>
<td>182.04</td>
<td>-</td>
<td>EAST</td>
<td>0462/0463</td>
<td></td>
</tr>
</tbody>
</table>

### OUTPUT SIGNALS —— 50 MSEC BINARY — (OUT10)

<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>TYPE</th>
<th>DESTIN.</th>
<th>OUTPUT</th>
<th>SIR</th>
<th>BIT(S)</th>
<th>ADDR</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRUA</td>
<td>BIT STR</td>
<td>NOTE 1</td>
<td>0-15</td>
<td>0413</td>
<td>1. SELF TEST BITS FOR RATE GYROS (A,B,C), ILS RCVRS (A,B,C). RADIO ALTIMETER (A,B) YAW DAMPER AND MACH TRIM. SET BY PREFLT ROUTINES. ELSE ZEROED BY OUTIO.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRUB</td>
<td>BIT STR</td>
<td>NOTE 1</td>
<td>0-15</td>
<td>0414</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRUC</td>
<td>BIT STR</td>
<td>NOTE 1</td>
<td>0-15</td>
<td>0415</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATUNE2</td>
<td>2X5 CODE</td>
<td>NAV RCVR</td>
<td>0-15</td>
<td>0417</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WPTALR</td>
<td>BOOLEAN</td>
<td>NCDU</td>
<td>0</td>
<td>041B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RWYOUT</td>
<td>BOOLEAN</td>
<td>INS</td>
<td>1</td>
<td>041B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NCOUNP</td>
<td>BOOLEAN</td>
<td>NCDU</td>
<td>2</td>
<td>041B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MDWARN</td>
<td>BOOLEAN</td>
<td>AVON LADY</td>
<td>4</td>
<td>041B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MLSVLD</td>
<td>BOOLEAN</td>
<td>NAV Pallet</td>
<td>5</td>
<td>041B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATE</td>
<td>BOOLEAN</td>
<td>THROTTLE</td>
<td>6</td>
<td>041B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MLSMOD</td>
<td>BOOLEAN</td>
<td>NAV Pallet</td>
<td>7</td>
<td>041B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NCUVAL</td>
<td>BOOLEAN</td>
<td>NCDU</td>
<td>8</td>
<td>041B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPBPSW</td>
<td>BIT STRING</td>
<td>SIU</td>
<td>0-9</td>
<td>041D</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE 3-2. — OUTPUT SIGNALS
<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>TYPE</th>
<th>DESTIN.</th>
<th>OUTPUT BIT(S)</th>
<th>SIR ADRS</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACWSE²</td>
<td>BOOLEAN</td>
<td>VARIOUS</td>
<td>0</td>
<td>0425</td>
<td>2. THESE BITS ARE ALSO SET IN FCFLAGS</td>
</tr>
<tr>
<td>VCWSE²</td>
<td>BOOLEAN</td>
<td>PALLET</td>
<td>1</td>
<td>0425</td>
<td>FOR TRANSMISSION TO DISPLAYS NORDEN</td>
</tr>
<tr>
<td>AUTOE²</td>
<td>BOOLEAN</td>
<td>LIGHTS</td>
<td>2</td>
<td>0425</td>
<td></td>
</tr>
<tr>
<td>LANDE²</td>
<td>BOOLEAN</td>
<td>3</td>
<td>0425</td>
<td>0425</td>
<td></td>
</tr>
<tr>
<td>DECRB</td>
<td>BOOLEAN</td>
<td>4</td>
<td>0425</td>
<td>0425</td>
<td></td>
</tr>
<tr>
<td>FLARE</td>
<td>BOOLEAN</td>
<td>5</td>
<td>0425</td>
<td>0425</td>
<td></td>
</tr>
<tr>
<td>RROUT</td>
<td>BOOLEAN</td>
<td>6</td>
<td>0425</td>
<td>0425</td>
<td></td>
</tr>
<tr>
<td>LANDA</td>
<td>BOOLEAN</td>
<td>7</td>
<td>0425</td>
<td>0425</td>
<td></td>
</tr>
<tr>
<td>LOCA</td>
<td>BOOLEAN</td>
<td>8</td>
<td>0425</td>
<td>0425</td>
<td></td>
</tr>
<tr>
<td>LOCE</td>
<td>BOOLEAN</td>
<td>9</td>
<td>0425</td>
<td>0425</td>
<td></td>
</tr>
<tr>
<td>GSARM</td>
<td>BOOLEAN</td>
<td>10</td>
<td>0425</td>
<td>0425</td>
<td></td>
</tr>
<tr>
<td>GSENG</td>
<td>BOOLEAN</td>
<td>11</td>
<td>0425</td>
<td>0425</td>
<td></td>
</tr>
<tr>
<td>GAE</td>
<td>BOOLEAN</td>
<td>12</td>
<td>0426</td>
<td>0426</td>
<td></td>
</tr>
<tr>
<td>AEE²</td>
<td>BOOLEAN</td>
<td>FFD CCP</td>
<td>0</td>
<td>0426/0427</td>
<td></td>
</tr>
<tr>
<td>TRIMD</td>
<td>BOOLEAN</td>
<td>STAB TRIM</td>
<td>1</td>
<td>0426/0427</td>
<td></td>
</tr>
<tr>
<td>TRIMT</td>
<td>BOOLEAN</td>
<td>STAB TRIM</td>
<td>2</td>
<td>0426/0427</td>
<td></td>
</tr>
<tr>
<td>SPFWINH</td>
<td>BOOLEAN</td>
<td>SPOILER FDBK</td>
<td>4</td>
<td>0426/0427</td>
<td></td>
</tr>
<tr>
<td>ICM</td>
<td>BOOLEAN</td>
<td>FC PALLET</td>
<td>13</td>
<td>0426</td>
<td></td>
</tr>
<tr>
<td>RUNM</td>
<td>BOOLEAN</td>
<td>FC PALLET</td>
<td>14</td>
<td>0426</td>
<td></td>
</tr>
<tr>
<td>HOLDM</td>
<td>BOOLEAN</td>
<td>FC PALLET</td>
<td>15</td>
<td>0426</td>
<td></td>
</tr>
<tr>
<td>(WDTV)</td>
<td>BIT STRING</td>
<td>SIU</td>
<td>0-15</td>
<td>0426/0420</td>
<td>ALTERNATE 1's &amp; 0's</td>
</tr>
<tr>
<td>(EOT-001)</td>
<td>BIT STRING</td>
<td>SIU</td>
<td>0-15</td>
<td>042E</td>
<td>WALKING BIT</td>
</tr>
<tr>
<td>ALTOMP</td>
<td>BIT STRING</td>
<td>MSP</td>
<td>0-15</td>
<td>0432</td>
<td>32 BIT OUTPUTS</td>
</tr>
<tr>
<td>FPTOMP</td>
<td>BIT STRING</td>
<td>MSP</td>
<td>0-15</td>
<td>0434</td>
<td></td>
</tr>
<tr>
<td>TKTOMP</td>
<td>BIT STRING</td>
<td>MSP</td>
<td>0-15</td>
<td>0436</td>
<td></td>
</tr>
<tr>
<td>ASTOMP</td>
<td>BIT STRING</td>
<td>MSP</td>
<td>0-15</td>
<td>0438</td>
<td></td>
</tr>
<tr>
<td>DSTMOP</td>
<td>BIT STRING</td>
<td>MSP</td>
<td>0-15</td>
<td>043A</td>
<td></td>
</tr>
<tr>
<td>RSTOMP</td>
<td>BIT STRING</td>
<td>N/U</td>
<td>0-15</td>
<td>043C</td>
<td></td>
</tr>
<tr>
<td>RATOMP</td>
<td>BIT STRING</td>
<td>N/U</td>
<td>0-15</td>
<td>043E</td>
<td></td>
</tr>
</tbody>
</table>
4.0 FLIGHT CONTROLS
This section describes the software that receives sensor inputs from the Sensor/Effector Interface Unit (SEIU), calculates according to mode control logic and control laws, 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 discretes (TRIMR, TRIMD). The computation of these commands, as well as the accompanying mode logic and error reporting are covered in detail in the following pages.
MODULE NAME: DINUSE

PURPOSE: To tell DISFD which discrete sensors it is required to check at any given time.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: FAST

CALLING SEQUENCE: If not in preflight CALL DINUSE.

CALLS TO: None

DESCRIPTION:
Based on the current flight mode and condition of the aircraft, DINUSE sets the sensor in use bit (#2000 octal) of the appropriate sensor status word. Each discrete signal is allocated one status word. This is then detected by DISFD which votes the signal if the bit is set.

GLOBAL INPUTS: MODEX, FLAGS, MLSMOD, LOCFS, LOCVLD, GSVLD, VBS, LBS

GLOBAL OUTPUTS: DSTAT
NAME: DISFD (Discrete Select and Failure Detect)

PURPOSE: To detect momentary discrepancies and failed sensor or serial bus errors for the packed discretes.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: FAST

CALLING SEQUENCE: CALL DISFD

CALLS TO: None

DESCRIPTION:
Raw discretes 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 insure that any bit change was not a fluke. The output is a single packed word with each bit set with slightly different logic depending on whether it is a select, a valid, or the duplex HRV. The output is stored in the compool BF at VDISC (BF + 60) for subsequent unpacking by the routine IOFLL.

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 invalid 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.

If the INS Single Thread flag has been set, then IATTV, INAVV and ISBV in all 3 raw discrete words are set to agree with the selected channel as determined by bits 0-1 (INSST) in the global word SINUS3.

PACKED DISCRETES WORD FORMAT

<table>
<thead>
<tr>
<th>BIT</th>
<th>SIGNIFICANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>HRV</td>
</tr>
<tr>
<td>1</td>
<td>ISRV</td>
</tr>
<tr>
<td>2</td>
<td>LANDS</td>
</tr>
<tr>
<td>3</td>
<td>AUTOS</td>
</tr>
<tr>
<td>4</td>
<td>VCWSS</td>
</tr>
<tr>
<td>5</td>
<td>ACWSS</td>
</tr>
<tr>
<td>6</td>
<td>GSVLD</td>
</tr>
<tr>
<td>7</td>
<td>LOCVLD</td>
</tr>
<tr>
<td>8</td>
<td>LAMPS</td>
</tr>
<tr>
<td>9</td>
<td>IATTV</td>
</tr>
<tr>
<td>10</td>
<td>CASV</td>
</tr>
</tbody>
</table>
DEBOUNCE ALGORITHM

\[ A_3 = A_2 \]
\[ A_2 = A_1 \]
\[ A_1 = A \oplus MLO \]
\[ A_2 = A_2 \land A_1 \]
\[ AF = AF \land \neg A_3 \lor A \land A_3 \]

(REPEAT FOR CHANNELS B & C)

\[ MLV = AF \land BF \lor AF \land CF \lor BF \land CF \]
\[ MLO = MLO \land PMLV \lor MLV \land PMLV \]

WHERE: 
\[ A = \text{NEW INPUT} \]
\[ A_1 = \text{FIRST BOUNCE, ETC.} \]
\[ AF = \text{DEBOUNCED VALUE} \]
\[ MLO = \text{MAJORITY LOGIC OUTPUT} \]
\[ MLV = \text{MAJORITY LOGIC VOTE} \]
\[ PMLV = \text{MLV FROM LAST PASS} \]

GLOBAL INPUTS: BF, CRSET, DSTAT, SINUS3.

GLOBAL OUTPUTS: DSTAT, VDISC.
MODULE NAME: DSTOR

PURPOSE: Stores data for error messages into failure data table.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: FDSTR, ILSRC, SRVCK

CALLING SEQUENCE: In FIDENT load address of failure message.
In FIDENT+10. load failed data value.
In FIDENT+12. load voted data value.
CALL DSTOR

CALLS TO: None

DESCRIPTION:
DSTOR packs eight words of information needed to create an error message into a data area for retrieval later by FMTMG. The arrangement of these eight words is as follows.

Word 1 Failure ID (address of failure message in MESG)
Word 2 Sensor channel failed/MODEX mode at failure time
Word 3 Hour of failure
Word 4 Minute of failure
Word 5 Second of failure
Word 6 Failed data value
Word 7 Voted data value
Word 8 Null word (set boolean true by FDSTOR if system reset)

This routine shares common local data with FDSTR, PANEL, FMTMG, and F2CMP.

GLOBAL INPUTS: FIDENT, MODE2, HRSS, MINS

GLOBAL OUTPUTS: FLTBL or PFLTBL, FIDENT

84
NAME: ELEVP -- ELEVator control laws (Pitch)

PURPOSE: To calculate the elevator command and to calculate
the stabilizer trim and autothrottle commands.

LANGUAGE: HAL/S

TASK: FAST

CALLING SEQUENCE: CALL ELEVP

CALLED BY: FAST

CALLS TO: ATHCL, HXFCL, LIMINT, PAFD, PAL, PFFD, PVPC, SCOSD, STABT,
VTFCL, $$EXP.

DESCRIPTION:
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 FFDE mode are described in PFFD. All inputs shown are voted from the
"outside world" except for KV and PHIVS, which are computed in the ELEVP
executive routine.

There are five distinct modes that can be flown from the aft flight
deck: MANEL, ACWSE, VCWSE, AUTOE and LANDE. The overall DSD showing the
computational requirements for the elevator command for these AFD modes
is given in the DSD section in the back of this document. The overall
static gains are included in these figures.

Procedure ELEVP is the driver for all the sub-procedures which
comprise the longitudinal axis control law. It consists of
initialization routines, the glide slope engage (GSENG) and track
(GSTRK) logic, and procedure calls.

ELEVP 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 comand 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
ICM flag is set, the filters and command outputs are initialized. Then,
for all modes, pitch rate (Q) is filtered and stored in QFB1 and QX.
Versine (PHI) is approximated and stored in PHIVS. The ILS vertical
velocity, HDILS, is initialized as required; the "flare set on last
pass" flag, FLARE_1, is updated, and filtered vertical acceleration
(HDIF) is computed. GSENG logic is executed; airspeed gain (KV) is
calculated, and the ten second countdown from GSENG to GSTRK is begun
(if GSENG is TRUE). Subsequently, through a case statement, ELEVP
processes the active mode as indicated by the value of MODEX.
For each of the modes described below, the intermediate output is 
DECGL, which is passed through the 1/2 second easy-on/off switch 
initialized at ICM or mode change. Except for LANDE, DECRB, and FLARE, 
the final elevator command is phased in as DECMQ. Then, the switch in 
control law source (and setting of INIT) occurs at GSENG. The next 
switch occurs at FLARE, but the control law itself takes care of the 
easy-on.

For PRENG mode, the elevator servo position (DEPOS) is simply 
passed to DECGL, without modification.

For MANEL mode, the column force input, DCOL, is processed through 
a deadzone, and, if out-of-detent, it is multiplied by -6.96.

For FFDE mode, the sub-procedure PFFD is called.

For ACWSE and VCWSE modes, sub-procedure PAFD is called.

For AUTOE mode, if GSENG is set for the first time, then the easy-
on switch is initialized and sub-procedure PAL is called. Otherwise, 
sub-procedure PVPC is called to generate the elevator command. The 
present setting of GSENG is saved in GSENG 1 for the next iteration.

For LANDE mode, the pitch autoland control law sub-procedure, PAL, 
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, if the flare control law flag, FLRFLG, 
is TRUE, then the H(x) control law, sub-procedure HXFCL, is used. If 
FLRSEL is FALSE, then the variable-tau law, sub-procedure VTFCL, is 
called.

For DECRB mode, if the FLARE flag is FALSE, sub-procedure PAL 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 1 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 ELEV outputs KQ, QX, DECMQ, and the 10 Msec 
pitch rate (Q) input.

To conclude ELEV processing, the stabilizer trim sub-procedure, 
STABT, and the authrottle control logic sub-procedure, ATHCL, are both 
called unconditionally.
SUB-PROCEDURES:

PFFD (Pitch, Forward Flight Deck)

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, then filtered to give FFDI. 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 FCOL is less than 5 lbs, and if the roll angle command (PHICMD) is less than 5 deg. The result of this integration, PINT2, is multiplied by .05, added to PFFDE, reduced by .1904 FFD1, and the result multiplied by 4.32 to give FCWSE. This signal is reduced by PHIVS to produce FDED. It is then used to set the elevator authority limit flag (DEAL) if the effective elevator command is greater than 15 degrees. Finally, it is attenuated as a function of KV and output as DECGL.

PAFD PROCEDURE (Pitch, Aft Flight Deck)

This procedure computes the delta elevator command (DECGL) in the velocity and attitude CWS modes and gamma commanded (GAMC) in the velocity CWS mode. 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. When the column is released, the commanded GAMMA (or PITCH) holds constant. The control law takes column input in a proportional and in an integral path. The proportional path gives the initial elevator command to initiate the maneuver to change the path angle or pitch angle.

The AFD column deflection (DCOL) is passed through a 0.1 inch deadzone and programmed with ground speed such that the commanded normal acceleration per inch of column input remains constant over the speed range. If GAE (go-around engaged) is set then a 2 degree bias is integrated into GAMC (gamma commanded) calculation until the 2 degree fly-up is achieved. At this point GAE is reset. The appropriate term for column deflection is filtered, integrated, and stored in GAMC. The GAMC integrator is modified if the VATRM (manual stab trim) flag is set; a 0.5 degree/sec bias is added. If VATRD (manual auto stab trim down) is set, a 0.5 degree/sec bias is subtracted.

The commanded gamma is subtracted from either GAMMA or PITCH to produce the error signal, GAMER. This is used in both the proportional and integral paths to develop the elevator command. Feedback of filtered pitch rate (QFB1) and the rate of change of error signal GAMD provide damping terms for the control law. DECGL is returned to ELEVP.

PAL (Pitch Auto Land control law)

This procedure computes the delta elevator command, DECGL, in auto-engage with glide slope engage, land engage or decrab modes.
If in MLS mode, the beam error is derived from DELTH (the MLS altitude error) gained as follows:

$$\text{GSPF} = 1.06 \times 0.0666 \times \text{DELTH}$$

For ILS mode, it is derived from GPGSDV with modifications depending on the state of GSTRK. If GSTRK (which is set 10 secs. after GSENG) the difference between the altitude rate of a path parallel to the glideslope ($\text{GSFPS} \times \tan(\text{GSA})$) and the filtered rate of altitude change ($\text{HDCF}$), is integrated with the beam error, and the output is filtered over 15 seconds to produce GSPF. If NOT GSTRK, the slope deviation is calculated as 1.06 $\text{GSGPA}$ (gain programmed glideslope deviation) and filtered over 1.5 seconds to produce GSPF.

Subsequent processing is common to both MLS and non-MLS modes. ALC is generated by summing 4.0 ($\text{GSPF} + 0.39 \times \text{HDCF}$) with the filtered vertical acceleration ($\text{HDDF}$). A portion of the original gain programmed deviation signal ($0.28 \times \text{GSGPA}$) is integrated and limited to give GSI. (GSI was initialized as $\text{DECMD} - \text{ALC}$.) This serves as an easy-on during glideslope capture and compensates for inaccuracies in the slope of the glideslope beam.

The final output, DECGL, is calculated as follows:

$$\text{DECGL} = KV(\text{ALC} + \text{GSI} + 2.16 \times \text{QFB1} - \text{PHIVS})$$

Where:

- QFB1 is the filtered pitch rate feedback.

DECGL is returned to ELEVP.

PVPC (Pitch Vertical Path Command)

This routine computes the elevator command DECGL for AUTOE mode when the glide slope has not been engaged. (PAL is called when GSENG IS TRUE.)

A vertical path command (EVPC) is derived as the difference between the vertical acceleration signal, HDDF, and the vertical acceleration command, VACMD. This difference 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). The integrator output 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.

HXFCL (H(x) FLARE CONTROL LAW)

HXFCL is the newer of the two flare control laws. It is called if the flag FLRSEL (IC = FALSE) is TRUE. The H(x) routine initializes itself, and sets the FLARE flag, at descent through 42 feet of altitude (in ILS mode), or when XHAT is less than XFLARE (MLS mode). It commands altitude as a function of distance (XFF) from FLARE to touchdown and thereby provides better control over the touchdown point than does the variable tau law. For MLSM, XFF is computed as the difference between
XFLARE and XHAT, where the former is a predefined (RWYDEF) distance from
the MLS azimuth transmitter to a TD point at a nominal distance past
XGPIP. For ILS, XFF is computed indirectly as the integral of INS
ground speed. Subsequent processing is identical for both ILS and MLS
modes.

The altitude command computations generate the commanded altitude
(HC), rate (HDC), and acceleration (HDDC) as follows:

\[
HC = \left(\frac{K1}{K2^{**2}}\right) (\text{EXPO} -.25 \text{EXPOSQ}) + K3 \ XFF + K4, \\
HDC = \text{VGF} \left(\frac{K1}{K2}\right) (.5(\text{EXPOSQ} - \text{EXPO}) + K3), \\
HDDC = (\text{VGF}^{**2}) K1 \ (\text{EXPO} - \text{EXPOSQ})
\]

Where:

- \( \text{EXPO} = e^{(-K2 \ XFF)} \)
- \( \text{EXPOSQ} = \text{EXPO}^{**2} \)
- \( K1 = .0001816455 \)
- \( K2 = .002047959 \)
- \( K3 = -.0079918 \)
- \( K4 = 9.51766 \)
- \( \text{VGF} = \text{Ground speed in ft/sec:} \)
  - For MSW1. EL2F. \( \text{VGF} = -\text{XDH}, \)
  - ELSE, \( \text{VGF} = \text{GSINS KTOPYS}. \)
- \( XFF = \text{Runway dist. in feet from flare pt.} \)

The difference between the altitude and commanded altitude \((KHXH - HC)\) is computed and stored in HEPS. HDERX and DEFLIF are then
calculated as:

\[
\text{HDERX} = \text{HDILS} - \text{HDC} \\
\text{DEFLIF} = 4.0 \ \text{HDERX} + 2 \ \text{HEPS} - 2.5 \ \text{HDDC}
\]

The quantity \((KHXDD + 5.46 \ \text{QFBI} - \text{HDDQF})\) is integrated and the
result added to .72 \text{HEPS (HEK)} and filtered \text{QFBI (KDTW)}. This is stored
in HDDQF.
The output of the HXBIAS filter, which is zero at initialization and
decreases to -5 degrees, is added to DEFLIF to produce DEFL2F. This is
added to HDDQF and multiplied by KV to give the elevator command DECGL.

**VTFCL (Variable Tau Flare Control Law)**

VTFCL is the default flare control law, called when FLRSEL is FALSE
(the initial state). 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 functions to drive the rate of descent
exponentially to 2 fps at touchdown.

On the first pass, the filters and integrators are initialized. The
height is biased up 10 feet, multiplied by the ground speed, added to
the vertical velocity, HDILS, and the result stored in DEFL1. Errors in
the sink rate are then corrected by use of a complementary filter using
HDD for ILS OR ZDDH for MLS, and filtered pitch rate, QFB1. The result
is stored in HDDQ. The output of the FTAUF3 filter, which is initially
set to zero and decreases to -4 degrees to set the elevator command
bias, is added to DEFL1 and HDDQ.

This, multiplied by KV, yields the output DECGL.

STABT (STABilizer Trim)

This routine determines the stabilizer trim command, ASTP, 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 discrete, TRIMT, is cleared if GRD, MANEL, or PRENG are set. If
the magnitude of the auto stabilizer trim position, ASTP, is less than
one degree, then TRIMT is cleared; if the 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.

ATHCL (AutoTHrottle Control Law)

ATHCL is called unconditionally on each cycle of ELEVP. There are
three operating modes which depend on authrottle engagement (ATE) and on
the state of the FLARE flag. The routine first evaluates the indicators
for authrottle 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.

\[
AFTLIM = \begin{cases} 
10 \text{ degrees,} & \text{for } CAS < 200 \text{ knots.} \\
0.2 \ (250. - CAS), & \text{for } 200 \leq CAS \leq 250 \text{ knots.} \\
0, & \text{for } CAS > 250 \text{ knots.}
\end{cases}
\]

The complementary filtered longitudinal acceleration signal, NC101,
is derived by washing out the inertial longitudinal acceleration,
VGSDOT, with the true airspeed signal, TASFPS.

\[
\text{TEMP} = 5. \ (\text{TASFPS} - \text{NC101})
\]

\[
\text{NC101} = \text{NC101} + \text{DELTAT} \ (\text{TEMP} + \text{VGSDOT})
\]

This signal is then passed through a turbulence filter to limit high
frequency components:

\[
\text{TEMP} = \text{TEMP} - \text{NC102}, \quad \text{limit} = +/- 1.
\]
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 ft per sec per sec (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 . FLARE), then a series of actions occur. The raw throttle position rate command, APCPRM, is calculated as

\[
APCPRM = 3.0 \times (VSPHI - LAG16S + ATCMD - NCL1)
\]

Where:

\[
VSPHI = 5 - (5 \cos(ROLL))
\]

\[
LAG16S = \text{last cycle filter output}
\]

VSPHI is then passed through a 16 second washout filter to generate a new roll compensation signal:

\[
LAG16S = e^{(-0.05/16.0)} (LAG16S - VSPHI) + VSPHI.
\]

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 (SEPR) is calculated as the difference between the maximum EPR and EPR, gained by 3.3333, and limited to 1.0. APCPRM is then gained by SEPR. This attenuates the rate command as the engine approaches maximum output. However, if one EPR already exceeds MXEPR, (SEPR less than zero), then the raw throttle command is immediately set to 10 times SEPR.

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.

\[
APCDG = 2.4 \times NCL1 - NCI03
\]

If FLARE is set,

\[
APCDG = -NCI03.
\]

GLOBAL INPUTS: ACCHAT, ASTP, ATCMD, ATE, ATFDBK, CAS, CROLL, DCOL, DCMDC, DEPOS, EL2F, EPR1, EPR2, FAIL2, FCOL, FLAGS, FLRSEL, GAE, GPGSDV, GS, GSDEV, GSFPS, GSVLD, HDCF, HDD, HGPIP, HRAD, HTDZ, IASSEL, ICM, MLSC, MLSM,
GLOBAL OUTPUTS: MODE2, MODEX, MSW1, MSW6, MXEPR, PHICMD, PITCH, POSHAT, Q, ROLL, SQUAT, TANGSA, TASFPS, TIMPTH, VACMD, VATRD, VATRM, VELHAT, VGSDOT, XFLARE, XGPIP.

GLOBAL OUTPUTS: APCDG, ATE, DECMQ, DSPLF, FLAGS, GAE, GAMC, GAMMA, GRD, KQ, KV, MODEX, QX, TRIMD, TRIMT, VBS.
NAME: EPRLMT (Engine Pressure Ratio Limit)

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

LANGUAGE: HAL/S

TASK: SLOW

CALLED BY: SLOW

CALLING SEQUENCE: CALL EPRLMT

CALLS TO: None

DESCRIPTION:
EPRLMT produces maximum EPR limits for climb (MCLEPR), cruise (MCREPR), and continuous thrust (MCTEPR). Depending upon the setting of EPRFLG, one of these is selected as the maximum EPR (MXEPR) for display.

The program is organized in two logical sections, for engine bleed-air on, and for engine bleed-air off. The calculations consider the effect of static air pressure (STAT_PRES), true 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 flow chart.

GLOBAL INPUTS: ABSTAT, ALT, BARSETO, EPRFLG, TAT.

GLOBAL OUTPUTS: MCLEPR, MCREPR, MCTEPR, MXEPR.
MODULE NAME: FDSTR

PURPOSE: Examines status of redundant sensors to detect failure flags and record failure information in a table of failure data.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: FAST

CALLING SEQUENCE: CALL FDSTR

CALLS TO: DSTOR

DESCRIPTION:
This routine has five major parts. The first four incorporate checking the sensor status as reported by the SSFD. These four parts are segregated mainly by the allocation of the sensors they each process. The first section involves testing of the 10 ms sensors. The second and third test the 50 ms analog and 50 ms digital INS sensors, respectively. The fourth 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 two NORDEN 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 a description of the meaning of first and second failures see the description of SSFD. If a failure is indicated and has not already been recorded, the needed failure data is formatted into eight words and stored in a failure data table by the routine DSTOR for later retrieval by FMTMG when a failure display is requested.

A series of tests is performed on the SEIU/RF DIU system. The first test checks communication between the Norden #2 and the SEIU. Second is a check for power failure on the SEIU. Thirdly, a test is run on the six SEIU and two RFDIU analog to digital converters. Fourth is a test of Norden #2 to RFDIU communication. Fifth, a test for RFDIU power fail is made. Last, a test of the DATAC link from Norden #1 to Norden #2 is made.

This routine shares common local data with DSTOR, PANEL, FMTMG, and F2CMP.

GLOBAL INPUTS: SQUAT, WSPIN, FALST, MSWIT, STFAIL, BF, DSTAT, HRV, ANDS, AUTOS, VCWSS, ACWSS, GSVLD, LOCVL, IATTV, CASV, CALTV, LOCFS, INAVV, ISBV, DSTAT, TAVV, FLDS, CRSET

GLOBAL OUTPUTS: PMSWIT, MSWIT, STFAIL, FIDENT, LIGHTS
MODULE NAME: FMTMG

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.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: PANEL, PRFLT, CTLCK, ILSRC, DETNT (MLOG)

CALLING SEQUENCE: In R1 load address of message to be formatted. It is used to determine if the message is of the demand type. In R2 load address of first word of eight stored in the failure data table (R2 is required only if the message is of the type stored in the failure data tables). CALL FMTMG

CALLS TO: None

DESCRIPTION:
Four 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, flight mode when failure occurred.

This routine uses a local subroutine called ICO to convert octal values into decimal ASCII.

This routine shares common local data with FDSTR, DSTOR, PANEL, and F2CMP.

GLOBAL INPUTS: MSWIT, LIGHTS, MESG (pool of error messages)

GLOBAL OUTPUTS: MSBUF, MSGST, WRDCNT
MODULE NAME: F2CMP

PURPOSE: Checks individual sensor second fails to determine if a mode failure is required.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: FAST

CALLING SEQUENCE: (If in flight mode) CALL F2CMP

CALLS TO: FMTMG

DESCRIPTION:
Given the existing flight mode this routine 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 routine also clears all mode failure flags upon CRSET and clears the error message table upon ERSET.

This routine shares common local data with FDSTR, DSTOR, PANEL, and FMTMG.

GLOBAL INPUTS: BF, MODEX, MLSVAL, BMAFLG, AFCSV, CASV, IATTV, DSTAT, INAVV, CALTV, MLSMOD, GSVLD, LOCVLD, ATE, TAS, TASV, FAIL2, DSPLF, FSIDX, MESG (pool of error messages), CRSET, ERSET

GLOBAL OUTPUTS: FAIL2, DSPLF, LIGHTS, STFAIL
MODULE NAME: GMSG

PURPOSE: To output data to the system test panel and printer.

TASK: SLOW

LANGUAGE: MACRO-11

CALLED BY: SLOW

CALLING SEQUENCE: MSGST contains address of message start. WRDCNT contains the number of bytes to be output. If WRDCNT <> 0 CALL GMSG

CALLS TO: None

DESCRIPTION:
GMSG drives the system test panel display and lights with data stored in a message buffer. This buffer's location is dynamic and its address must be stored in the word MSGST. The length of the buffer is also dynamic and must be stored in WRDCNT (number of bytes). A special case calling this procedure is when the test pattern is required on the system test panel. This is accomplished by clearing MSGST before calling GMSG.

GMSG also outputs to a line printer all messages (except panel test pattern).

GLOBAL INPUTS: WRDCNT, MSGST

GLOBAL OUTPUTS: WRDCNT, IOACT
MODULE NAME: LATRL (Lateral axis control law)

PURPOSE: To calculate aileron and rudder commands for the flight control system.

LANGUAGE: HAL/S

TASK: FSTTSK

CALLING SEQUENCE: CALL LATRL

CALLED BY: Fast loop executive (FAST)

CALLS TO: FRCWS, RCOM, RBASC, CMPF, DCRAB, LIMINT, SRLMT

DESCRIPTION:

LATRL is the scheduler which calls the other procedures which make up the lateral axis control laws. It is composed primarily of logic checks, initialization routines, and procedure calls.

The sub-procedures are:

- FRCWS  Forward flight deck control wheel steering
- RCOM    Roll command computer
- RBASC   Basic roll computer
- CMPF    Complementary filter
- DCRAB   Decrab flight mode calculations

LATRL first assigns or calculates the parameters for the MLS or ILS modes, especially determining whether the localizer is engaged (LOCE) and the aircraft on-course (ONCRS). For both MLS and ILS, if LOCE is TRUE the localizer deviation logic is executed and the complementary filter procedure CMPF is called.

Next, the lower flight modes are checked and processed as appropriate. ICM and PRENG are done first. Then, if FFDE mode is active, procedure FRCWS is called. If the mode is MANEL, that logic is performed in-line and LATRL terminates.

For the higher flight modes, procedure RCOM is called unconditionally, followed by DCRAB if DECRB flight mode is active and by RBASC. The aileron servo command (AILCMD) and rudder servo command (RUDCMD) are determined, the initialization flags are reset, and LATRL terminates.

Various combinations and levels of automatic flight control assistance are available to the AFD and FFD pilots. The lateral control laws for each mode are depicted in the DSD's found in the back of this manual. Each mode is listed below, with the source of inputs and the control law used. Note that the AFD cab does not have an aileron trim mechanism; therefore the aileron trim knob command is brought into the FCC separately and is summed with the aileron command.
MODE PROCESSING:

PRENG:

The aileron trim command is initialized to zero at auto-pilot engagement regardless of aileron trim knob position. This is done by summing the aileron trim command with the aileron position signal. This is not done outside the pre-engage mode but the pre-engage value of the aileron position and trim command is available in SYNCL. The aileron synchronization routine is:

1. SYNCL = 1.52 ATRIM (Limited to +/- 5.0)
   SYNCL = ALVDT + SYNCL
2. AILCMD = ALVDT
3. RUDCMD = DRPOS

FORWARD FLIGHT DECK CWS (FFD):

The lateral axis control signals for the forward flight deck CWS mode is programmed entirely in one procedure. The FFD CWS has two submodes, attitude hold and attitude synchronization. In the ATT.Sync submode, roll attitude is controlled by the pilot’s wheel inputs. Roll--command (FPHCMD) is forced to track roll attitude (PHI) by feeding the difference between the two signals as an error signal to the FPHCMD integrator. ATT.Sync submode is entered when the pilot applies a control wheel force sufficient to exceed the wheel input deadzone. When the wheel force is released, the control wheel centers, roll rate is reduced, and the difference between PHI and FPHCMD becomes increasingly small. When the difference (FPHIER) is sufficiently small, the altitude hold submode (unlabeled) is entered by default. In this submode, the roll command (FPHCMD) is held constant by removing the input to the roll command integrator. This submode is transparent in that the FRCWS procedure normally generates a "hold" roll command and changes it only if ATT_SYNC is TRUE.

AFT FLIGHT DECK MODES:

MANEL:

In the manual-electric mode, pilot inputs (DWHL) and aileron trim knob input (ATRIM) are processed in the roll basic (RBASC) procedure and are summed to form aileron command.

WHLINP = 0, IF DWHL < 1.0
   WHLINP = -.448 * (DWHL - SIGN(DWHL)) otherwise.
AILTRM = ATRIM * 1.52
RUDCMD = PEDAL * 6.55
AILCMD = -(WHLINP + AILTRM - SYNCL)
ACWS:

The Attitude Control Wheel Steering mode uses sensor feedback to stabilize the airplane. Roll rate feedback is computed in mainline code. The difference between roll command and roll angle is PHIERR and is computed in procedure RCOM. The main stabilizing term is roll rate (PHDOT) which is filtered, then summed with PHIERR to form the aileron command (AILCD1). The roll rate signal would tend to cancel the pilot's input, so roll rate feedback is removed when RCWOD becomes TRUE. In order to maintain the same performance at all speeds, aileron command is gain programmed as a function of airspeed.

In ATTCWS, the roll computer, procedure RCOM, has two submodes. These are attitude-synchronization (ATT_SYNC) and attitude-hold, which is the normal mode. When the pilot moves the panel mounted controller (PMC) far enough to exceed the DWHL deadzone, the RCWOD logic is satisfied and the ATT_SYNC submode is entered. In ATT_SYNC, the difference between PHI and PHICMD is used to drive PHIERR to zero, i.e., to make PHICMD track PHI. In this submode, the roll computer has a loop gain of 15. This value was chosen empirically for best pilot response. The limit (ROL_LIM) on the PHICMD integrator is 50°.

When the airplane is at the desired roll attitude, the pilot centers the PMC, RCWOD is no longer TRUE, and roll rate decreases. When PHIERR is sufficiently small, ATT_SYNC changes to the attitude hold submode. In this mode, the input (PTDCMD) to the PHICMD integrator is zero.

The roll basic (RBASC) procedure processes the pilot's ATRIM and PEDAL inputs and also contains the turn coordinator. Turn coordination is used in the control wheel steering modes when the flaps are down to improve lateral handling. Since roll command anticipates a turn, it is used to command some corresponding yaw through the rudder. The rudder input is only required for turn initiation, however, so roll command is passed through a washout circuit. The output of the turn coordinator is summed with the pilot's pedal input to form the rudder command.

VCWS:

Velocity control wheel steering has three submodes. The first two, ATT_SYNC and attitude-hold, are the same as in ACWSE, where attitude-hold is the unlabelled default mode. The third submode is called TRACK_HOLD. In procedure RCOM, upon leaving ATT_SYNC submode, either attitude-hold or TRACK_HOLD is entered. If roll attitude is greater than 5 degrees, attitude-hold is entered; if less, TRACK_HOLD is entered. The purpose of TRACK_HOLD is to maintain a constant ground track. This is achieved by keeping average cross track acceleration (XTACC) zero. XTACC is calculated in HNAVFS and is based on INS or MLS sources, depending on the active mode. XTACC is integrated once (XTK1) and used as an input to the roll computer to command a roll angle (PHICMD).

If the airplane were mistrimmed in the lateral axis, some XTK1 signal could be required to command sufficient aileron to return the airplane to trim. This would result in some cross track drift. To remove this error source, XTK1 is integrated to form XTK2 which is summed in the mainline code with aileron command. When the pilot makes a wheel input to turn to a new heading, the XTK1 integrator is reset to zero. Since the mistrim will change...
minimally with heading, the XTK2 integrator is "washed out" with a 5 second
time constant whenever the wheel is out of detent and it is reset to zero only
when switching out of the VCWSE mode.

AUTO:

In the AUTO mode the horizontal path command is processed in procedure
RCOM. It is derived from LOCCMD if the localizer is engaged (LOCE) or from
BACMD otherwise. It is then limited to a maximum angle of 5 degrees if in
FLARE mode, 10 degrees if ONCRS is TRUE, or 25 degrees otherwise. Maximum
roll angle and roll rate limits are determined and the roll angle command
(PHICMD) is produced with respect to those limits.

AUTOLAND:

The computation of the localizer command signal (LOCCMD) is different for
ILS and MLS operation. Hence, these processes will be discussed
separately. Subsequent lateral axis computations are essentially the same for
ILS and MLS and will be discussed in combination.

ILS PROCESSING:

Localizer capture is initiated at localizer-engage (LOCE). The gain
programmed and limited localizer deviation is calculated in the mode logic
procedure (MLOG) and input as ETAFT.

ONCRS logic is used to change from localizer capture to localizer track
submode. It is computed as:

ONCRS = (ABS(RF4) < .0271) . (ABS(LOCDEV) < LOCVL)

                      . (ABS(ROLL) < 3.0)

Where:

RF4 is the estimate of localizer beam rate.

At localizer capture (LOCE), ETAFT is used to generate a guidance signal
to capture the beam center. It is input to the complementary filter procedure
(CMPF) where it is passed through a noise filter composed of a first order
filter with a 2 second time constant. A cross track damping term (XTKDMP) is
added to the filtered localizer beam error to form the localizer command
signal, LOCCMD.

When ONCRS logic is satisfied, the localizer-track submode is entered.
Localizer beam error limit changes from 2 degrees to less than 0.8 degrees.
The gain programmed beam error is complementary filtered with the selected
cross track velocity signal, SEL XTV. The complementary filtering takes place
in a first order filter with a 20 second time constant. The output of the 20
second lag (LOCCF2) is initialized to be the same as the 2 second lag
(LOCCF1), so that no transients will occur at ONCRS. The gain on XTKDMP is
increased at ONCRS and XTKDMP is added to the complementary filtered beam
error. At ONCRS a localizer beam integral path is added to the control law.
Localizer beam error is integrated and the output of the integrator (LOCINT)
is added to complementary filter beam error and XTKDMP to form LOCCMD.
MLS PROCESSING:
Localizer capture is initiated at localizer engage (LOCE) as in ILS operation. However, for MLS operations, on course (ONCRS) occurs simultaneously with LOCE. The LOCE/ONCRS logic is the first code block in LATRL and equates to:

\[ \text{ONCRS/LOCE} = \text{MLSM} \cdot \text{LOCA} \cdot (\text{ABS}(\text{DELTY}) < (\text{LOCVL} - 2.67)) \]
\[ \cdot (\text{ABS}(\text{ROLL}) < 3.0)) \cdot (\text{ABS}(\text{YDH}) < (0.000473 \times \text{XHAT})) \]

When ONCRS/LOCE logic is satisfied, the MLS derived lateral position error (DELTY) is brought directly into procedure CMPF and becomes equivalent to the ETAFT signal from ILS operations. Complementary filter processing is not performed on the lateral position error. DELTY is integrated to form LOCINT which is summed with DELTY and a cross track damping term (XTKDM) to form the localizer command (LOCCMD) signal. The MLS derived YDH (Est. Xtrack Vel.) is used to develop XTKDM.

MLS/ILS COMPUTATIONS:
The bank angle limiter in procedure RCOM limits LOCCMD to 25 degrees of roll command at LOCE and 10 degrees of roll command at ONCRS. Since LOCE and ONCRS occur simultaneously for MLS, the lower limit will be used starting at localizer capture. The limit is reduced to 5 degrees at FLARE. In procedure RCOM the roll computer rate limit is raised from 4 degrees/sec to 7 degrees/sec at ONCRS. The effect of these changes is to allow the airplane to track localizer beam center more closely, and at the same time reduce the possibility of a violent maneuver by limiting command authority.

At 150 feet of altitude the DECRB mode is selected by mode logic and the DCRAB procedure by LATRL. The purpose of the decrab maneuver is to align the centerline of the airplane with the centerline of the runway. This is done by bringing the difference between airplane heading and runway heading (DLPSI) into the DCRAB routine where it is passed through a 5 degree limiter. The output of the limiter is called PSILIM. To avoid transients, PSILIM is passed through a 6 second "easy-on". The output of the easy-on (YTEMP) is passed through parallel integral paths, then incorporated with YTEMP to form YAWCMD. This is fed to the rudder to decrab the airplane. In a crosswind situation, unless sideslip compensation is used, lateral deviation will occur as soon as rudder is applied to decrab the airplane. Therefore, YAWCMD is multiplied by 1.4 and used as an aileron crossfeed (AILXFD) signal to command the appropriate roll angle. Note that localizer beam error is still used, outside this procedure, for lateral guidance in DCRAB submode.

The DLPSI limiter referred to earlier limits the amount of decrab to 5 degrees from the airplane heading existing at the time DECRB occurred (150 ft.). If, due to a cross wind, the airplane heading was 7 degrees from the runway centerline heading when DECRB occurred, then after the decrab maneuver the airplane heading would be 7 - 5 or 2 degrees. The limit is necessary to retain sufficient control authority for lateral guidance.

The roll-out (RLOUT) submode is the natural sequel to the decrab routine and occurs when weight on the landing gear causes the SQUAT switch to activate and when wheel spin (WSPIN) is sensed. In procedure RCOM, PHIERR now commands
wings level. Lateral guidance on the ground is achieved by summing lateral beam error, suitably scaled, with YAWCMD to drive the rudder. A damping term is needed for lateral stability, so DLPSI is differentiated in YF1 to form PSIDMP. PSIDMP is summed with PHICMD to form YAWDMP. The original decrab command (PSILIM) was passed through a proportional and a parallel integral path. At roll-out, the input to the integral path is removed. Thus, rudder commands stored on the integrator output will be held constant during the roll out.

SUB-PROCEDURES:

LOCAL PROCEDURE: FRCWS (FoRward flight deck Control Wheel Steering)

If the first pass flag is set, the roll rate filter and the roll command integrator are initialized. If the pilot's control wheel is out of detent the wheel force (FWHL) is integrated, limited and stored as FCOM. Next, roll rate (P) is filtered and stored in FRFI.

There are two submodes within FRCWS: attitude-hold and attitude-synchronization (ATT_SYNC). The difference is that ATT_SYNC tracks the roll angle commanded when the control wheel is out of detent, whereas the attitude-hold mode maintains whatever roll attitude is established when the wheel is returned to detent. In the ATT SYNC submode the ROLL input is filtered, limited and output as FPHCMD. FPHIER is calculated as FPHCMD - ROLL. If the wheel is out of detent, FPHIER is doubled, otherwise, it is doubled and reduced by 1.5 * FRFI. This value, AIL1, is input to the final aileron command calculation:

$$\text{AILCMD} = -(\text{KV} \times \text{AIL1} + \text{FCOM}).$$

The result is negated because of aircraft wiring polarity requirements.

In the attitude-hold sub mode, the wheel is in detent and the minimized value of AIL1 is used. FRCWS will remain latched in this mode until the pilot moves the wheel out of detent.

LOCAL PROCEDURE: CMPF (CoMPlementary Filter)

This routine provides a noise filter and first order lag filter for the ETAFT (720. * GPOCD) signal upon localizer engage and integrates that signal (in MLS the DELTY signal), and adds the cross track damping term.

On first pass, LOCINT is initialized to zero, LOCCF to .0171 ETAFT, and XTKDMP to SEL XTV. Subsequent CMPF processing is dependent on the state of MLSM, as follows:

ILS MODE:

Prior to localizer on course (ONCRS), gain programmed localizer error, ETAFT, is filtered into LOCCF using a first order lag with a 2 second time constant. The cross track damping term, XTKOMP, is derived from cross track velocity (XTVEL). XTKDMP is added to LOCCF to form localizer command (LOCCMD). Subsequent to ONCRS, complementary filtered localizer beam error is obtained by summing localizer beam error (ETAFT) and XTVEL and filtering them into...
LOCCF using a 20 second time constant. ETAFT is integrated and stored in LOCINT. LOCINT, LOCCF, and XTKDMP are summed to form the output LOCCMD.

MLS MODE:
Lateral position error (DELY) is stored in LOCCF and integrated to form LOCINT. XTKDMP is derived from MLS cross track velocity (YDH). When RLOUT is detected and XHAT is less than 10743 feet, DELTY is scaled up by 10743/XHAT. LOCINT, LOCCF and XTKDMP are summed to form the output LOCCMD.

In both modes, LOCCMD is passed to procedure RCOM as an input to the roll command signal PHICMD.

LOCAL PROCEDURE: RCOM (Roll COMputer)

The routine first checks the first pass flag and initializes the filters and mode discretes if true. It then checks whether in AUTO or one of the control wheel steering (CWS) modes, attitude CWS (ACWS), or velocity CWS (VCWS). If in CWS, further checks are made to determine the appropriate submode: attitude synchronization (ATT_SYNC), attitude hold, and track hold (TRACK_HOLD). Note that attitude hold does not have an explicit boolean, but is inferred when AUTOE, ATT_SYNC and TRACK_HOLD are all false.

Engage criteria for the CWS submodes are as follows:

\[
\text{ATT}_\text{SYNC} = \text{RCWOD} = (\text{ABS(DWHL)} > 1.0) \\
\text{ATT}_\text{HOLD} = \text{RCWOD} \cdot (\text{ABS(PDTCMD)} < 0.75) \\
\text{TRACK}_\text{HOLD} = \text{VCWSE} \cdot \text{ATT}_\text{SYNC} \cdot (\text{ABS(PHICMD)} < 5.0)
\]

Attitude sync is exited by obtaining the conditions required for attitude hold; track hold (not available in ACWS) is exited when RCWOD becomes true, and attitude hold is exited by obtaining the conditions required for either ATT_SYNC or TRACK_HOLD.

Input to the PHICMD integrator (PDTCMD) is computed based on the selected submode. For ATT_SYNC, PDTCMD = -15. PHIERR. In effect, then, PHICMD is a .067 second lag on roll attitude. In attitude hold, PDTCMD is set to zero, forcing PHICMD to hold the last commanded roll angle. In TRACK HOLD, PDTCMD is set to the gained and limited difference of PHICMD and the first integral of cross track acceleration (XTACC). PHICMD is thus a 0.36 second lag on cross track velocity with a 4 degree/second rate limit. Note that XTACC is selected from MLS or INS cross track acceleration in the Navigation fast loop (HNAVFS). The second integral of XTACC (XTK2) is also computed when in TRACK HOLD. This is summed with AILCD1 in mainline to compensate for any lateral mistrim. This integrator is washed out with a 5 second time constant when in ATT_SYNC, and cleared when not in VCWS.

If in AUTO mode, another set of submodes is available. If LOCE is false, the limited roll command (RCLIM) is derived from BACMD. This signal is generated in NFCM from track angle command or horizontal path command (depending on the TKSEL discrete) and limited to 25 degrees. If LOCE is true, RCLIM is derived from the LOCCMD output of the complementary filter procedure.
(CMPF), limited to 25 degrees until ONCRS, then limited to 10 degrees until 
FLARE, then limited to 5 degrees. In any case, PDTCMD is then set as the 
gained and limited difference of RCLIM and PHICMD. PHICMD then becomes a 0.2 
second lag on RCLIM with a rate limit of 7 degrees/second if ONCRS is true, 
and 4 degrees/second otherwise.

Finally, PHICMD is computed as the integral of PDTCMD, and PHIERR is 
computed as the difference between PHICMD and ROLL. Exception processing sets 
PHIERR to -ROLL to command wings level at RLOUT.

PHIERR is the principal output of the RCOM routine, although PHICMD is 
used during de-crab and XTK2 is used in velocity CWS mode.

LOCAL PROCEDURE: RBASC (Roll BASIC)

This routine processes the aileron trim knob and rudder pedal plus rudder trim. It also computes the turn coordinator.

On first pass, the turn coordinator filter is zeroed. Next, aileron trim input (ATRIM) is multiplied by 1.52 and stored in AILTRM. The turn coordinator (DELRF) is calculated in a washout filter as follows:

\[
\text{DELRF} = \text{PHICMD} - \text{RF2} \\
\text{RF2} = \text{PHICMD} - \exp(-0.3 \times \text{ITIME}) \times \text{DELRF} \\
\text{DELRF} = 0.235 \times \text{DELRF}
\]

The PEDAL input is multiplied by 6.55 and stored in RUDCMD.

If a CWS mode is engaged, or if AUTOTC and AUTOE are true while ONCRS is false, and the flaps are extended at least 20 degrees, and PHICMD is less than 2 degrees, then the turn coordinator is engaged. Turn coordination remains latched until one of the conditions above is not satisfied. If DECRB mode is active, then RUDCMD is decreased by the DECRB value. (Note that AUTOTC exists for experimental purposes and may be set through TERMX to enable AUTO mode turn coordination.)

LOCAL PROCEDURE: DCRAB (DeCRAB)

This routine generates de-crab and aileron crossfeed signals to align the aircraft and runway centerline headings 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, PSILIM = DLPSI. Otherwise, PSIDZ is subtracted from DLPSI and the difference is stored in PSILIM. PSILIM is passed through a 6 second easy-on to produce YTEMP. DLPSI is differentiated and stored as YF1. Prior to RLOUT the yaw damping term YAWDMP is zero and the output of the 6 second easy-on, YTEMP, is integrated, limited, and stored in YINT1. Subsequent to RLOUT, YINT1 is held constant and the output of the 5 degree deadzone is ignored. YAWDMP is derived from YF1.
and PHICMD. Finally, for both values of RLOUT, the aileron cross-feed and DECRAB output are generated as follows:

\[ \text{YAWCMD} = \text{YTEMP} + 0.756 \text{YINT1} \]
\[ \text{DECRAB} = \text{YAWCMD} + \text{YAWDMP} \]
\[ \text{AILXFD} = 1.4 \text{YAWCMD} \]

GLOBAL INPUTS: ALVDT, ATRIM, BACMD, DELTY, DLPSI, DRPOS, DWHL, FLAGS, FLAP, FWHL, GPLOCD, ICM, KV, LOCDEV, LOCFS, LOCVDL, MLSM, MODEX, P, PEDAL, POSHAT, ROLL, VELHAT, XTACC, XTVEL, YPROF.

GLOBAL OUTPUTS: AILCMD, LBS, PHICMD, RUDCMD, SPFINH, TRACK_HOLD, ATT_SYNC, SYNCL, XTK1, XTK2, POTCMD, PHIERR, PHICMD.
MODULE NAME: MESG

PURPOSE: ASCII data area for error messages.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: None

CALLING SEQUENCE: None.

CALLS TO: None

DESCRIPTION: MESG contains a pool of ASCII error messages to be displayed by the system test panel.

GLOBAL INPUTS: None

GLOBAL OUTPUTS: None
NAME: MLOG (Mode Logic)

PURPOSE: To determine and control the flight mode for operation of the Flight Controls programs.

TASK: FSTTSK

LANGUAGE: HAL/S

CALLED BY: FSTTSK

CALLING SEQUENCE: CALL MLOG

CALLS TO: ANGL, DETNT, TAND, UNPK

DESCRIPTION:
Mode Logic (MLOG) determines and selects a flight control mode under the Advanced Guidance Control System (AGCS). It first checks the system run status and terminates if it is in HOLD Mode, otherwise, processing continues.

Preliminary matters include the assignment of alternate sources for some variables depending on whether or not an airport has been selected, MLS has been selected, MLS operation is real or simulated, and whether in IC or Run mode. This processing is done here, rather than by LATRL/ELEVP as was done previously, since they are also used by some navigation functions which run after MLOG and before the flight controls procedures.

For MLSMOD, the MLS runway parameters are initialized and MLSM is set TRUE. For non-MLS mode, the ILS parameters are processed. Prior to localizer ONCRS, localizer deviation is limited to 2 degrees of beam error. At ONCRS, localizer deviation has a variable limit applied to it which is a function of altitude. The limiter is used to reduce localizer deviation due to external disturbances. The localizer variable limit function is:

\[ \text{LOCVL} = \begin{cases} 0.2 \text{ degrees, for } H < 500 \text{ feet.} \\ 0.0006 \times H - 0.1 \text{ degree, for } H \geq 500 \text{ feet.} \end{cases} \]

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 0.20 because localizer beam error is used for rollout guidance. After gain programming, localizer deviation (GPLOCD) is "per foot", rather than "per degree".

If the DELAY variable is non-zero, it indicates that there was a mode logic "dropout" and that mode determination is being delayed for four iterations (200 Msec) to allow time for the engage paddle to drop.

For AGCS, either the forward or aft flight deck paddle must be raised on the FFD Control & Command panel. Successful engagement of the FFD panel
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 .15 seconds. If AEE is FALSE, the paddle will spring down to the OFF position. AEE TRUE is a precondition for all flight modes except PRENG. The logic for initial AEE determination is:

\[
\text{AFCSS} = \text{FFDS + AFDS} \quad \text{(performed in hardware)}
\]

\[
\text{AEE} = \text{AFCSS } \cdot \text{AFCSV } \cdot \text{DETNT(1.5,4.5)} \cdot \text{FAIL2(AFCS)}
\]

Where:

\[
\text{AFCSV} = \text{Auto Flight Control System Valid}
\]

\[
\text{DETNT} \text{ is a function which returns TRUE if both DCOL & DWHL, respectively, are within parameter limits.}
\]

And to remain valid:

\[
\text{AEE} = \text{AFCSS } \cdot \text{AFCSV } \cdot \text{FAIL2(AFCS)}
\]

MODE OVERVIEW:

There are ten flight modes available. Pre-engage (PRENG) is the default when no other mode can be sustained. Forward Flight Deck Engage (FFDE) permits control wheel steering (CWS) from the forward flight deck (FFD). All other modes pertain only to the aft flight deck. Manual-Electric mode (MANEL) permits manual control with minimum computer aid. The AFD CWS modes permit automatic flight control with reference to attitude (ACWS) or to the velocity vector (VCWS). The Autoland modes provide automatic flight control for the 3D guidance and the landing phase, through decrab (DECRB), FLARE, and rollout (RLOUT).

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) and the 2nd fail flag is clear, the FFDE mode is selected and further mode determination is bypassed. The pilot then has CWS in the forward cockpit. The logic for FFDE is:

\[
\text{FFDE} = \text{AEE } \cdot \text{FFDS } \cdot \text{FAIL2(FFD)}
\]
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 AFD when no other mode is selected. ACWS will engage when:

$$ACWSE = AEE \cdot FFDS \cdot (ACWSS + ACWSE + MODEX = 0)$$

$$(ACWSE \cdot ACWSS) \cdot AGCSS \cdot FAIL2(ACWS)$$

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.

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 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 selected when VCWSE is FALSE and AEE is TRUE.
2) GAS is selected with AEE TRUE.
3) AUTOE becomes FALSE when AUTO is TRUE.
4) AUTO is selected while in AUTOE Mode.
5) LANDE goes from TRUE to FALSE.

Disengagement occurs when:

1) AEE is lost, causing reversion to PRENG.
2) ATT CWS, VEL CWS or AUTO is selected while in VCWSE Mode.
3) VCWS second fail occurs, causing reversion to ACWSE.
AUTOE (AUTO Engage Mode):

The AUTOE mode is a precondition for the higher modes: LAND, DECRB, FLARE and RLOUT. It can be set TRUE only by pressing the AUTO button on the Mode Control Panel (MCP). The initial criteria are:

\[ \text{AUTOE} = \text{AUTOS} \cdot \text{AUTOE} \cdot \text{AEE} \cdot \text{FFDS} \cdot \text{ACWSS} \cdot \text{GAS} \]

\[ \cdot \text{DETNT(DZNE,1.0)} \cdot \text{FAIL2(AUTO)} \]

Where:
- DZNE = selected column dead zone.
- AUTOE selection is completed when AUTO mode remains set until read into the FLAGS array at the end of MLOG.

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) FAIL2(AUTO) becomes TRUE, causing reversion to VCWSE.
7) LANDE becomes FALSE, having been TRUE, causing reversion to VCWSE.

LANDE (Land Engage Mode):

When AUTOE 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:

1) LANDR (LAND READY):

\[ \text{LANDR} = \text{AUTOE} \cdot (\text{LANDE} \cdot \text{LANDS}) \cdot (\text{LANDS} + \text{LANDR}) \]

\[ \cdot \text{DETNT(.78,8.0)} \cdot \text{FAIL2(AUTO)} \cdot \text{FAIL2(LAND)} \]

2) LOCA (Localizer Armed):

\[ \text{LOCA} = \text{LANDR} \cdot (\text{ABS(DLPSI)} < 90.) \]

\[ \cdot (\text{MLSMOD} + \text{LOCVLD}) \cdot \text{LOCE} \]
3) GSARM (Glide Slope Armed):

\[ \text{GSARM} = \text{LANDR} \cdot (\text{ABS}(\text{DLPSI}) < 90.) \]
\[ \cdot (\text{MLSMOD} + \text{GSYLD}) \cdot \text{GSENG} \]

4) LANDE (Land Engage Mode):

\[ \text{LANDE} = \text{LANDR} \cdot \text{LOCE} \cdot \text{ONCRS} \cdot \text{GSENG} \cdot \text{GSTRK} \]

Where:
- GSENG and, 10 seconds later, GSTRK are set by ELEVP. (See Sec 5.0, ELEVP.)
- LOCE and ONCRS are set by LATRL. (See Sec 6.0, LATRL.)

Furthermore, IF LANDE was TRUE on the last pass, but FALSE on the current pass, then control reverts to VCWSE.

THE AUTOLAND MODES:

When LANDE mode is established, the autoland modes are entered, in sequence, as their associated logic is satisfied. At 150 feet altitude, DECRB mode should be activated to align the aircraft with the runway heading. Next, FLARE mode raises the nose of the aircraft and establishes a landing attitude. Finally, the Rollout (RLOUT) mode maintains a course down the runway centerline as the aircraft rolls to a stop. 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):

\[ \text{DECRB} = \text{LANDE} \cdot (\text{H} < 150.) + \text{DECRB} \]

2) FLARE (Flare Mode):

FLARE criteria are evaluated in ELEVP once DECRB is TRUE. If MLSM and H(x) flare law is selected (FLRSEL = TRUE), then FLARE is set when XHAT becomes less than XFLARE. Otherwise, it is set when the selected altitude reference (HRAD or HTDZ) becomes less than 42 feet.

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 AFD 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:

\[
\text{MANEL} = \text{AEE} \cdot \text{FFDS} \cdot (\text{ACWSS} \cdot \text{ACWSE}) \cdot (\text{MODEX} = 0) \cdot \text{AGCSS} \cdot \text{FAIL2(MANEL)}
\]

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 the variable DELAY is set to four to allow the 200 Msec intermission for the engage paddle to drop, and PRENG Mode is set. If the VCWS Mode was not selected, the Go-Around flag (GAE) is set FALSE. Last, the FLAGS array is updated to reflect the selected mode and to clear any flags not appropriate to that mode.
### BIT/ARRAY/INDEX ALIGNMENTS:

<table>
<thead>
<tr>
<th>VALUE</th>
<th>MODEX</th>
<th>FAIL2/DSPLF</th>
<th>FLAGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>UNDEF</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>PRENG</td>
<td>AFCS</td>
<td>PRENG</td>
</tr>
<tr>
<td>2</td>
<td>FFDE</td>
<td>FFD</td>
<td>FFDE</td>
</tr>
<tr>
<td>3</td>
<td>MANEL</td>
<td>MANEL</td>
<td>MANEL</td>
</tr>
<tr>
<td>4</td>
<td>ACWS</td>
<td>ACWS</td>
<td>ACWSE</td>
</tr>
<tr>
<td>5</td>
<td>VCWS</td>
<td>VCWS</td>
<td>VCWSE</td>
</tr>
<tr>
<td>6</td>
<td>AUTO</td>
<td>AUTO</td>
<td>AUTOE</td>
</tr>
<tr>
<td>7</td>
<td>LAND</td>
<td>LAND</td>
<td>LANDE</td>
</tr>
<tr>
<td>8</td>
<td>DECRB</td>
<td>AUTO THROTTLE*</td>
<td>DECRB</td>
</tr>
<tr>
<td>9</td>
<td>FLARE</td>
<td>MLS*</td>
<td>FLARE</td>
</tr>
<tr>
<td>10</td>
<td>RLOUT</td>
<td>---</td>
<td>RLOUT</td>
</tr>
<tr>
<td>11</td>
<td>---</td>
<td>---</td>
<td>LANDA</td>
</tr>
<tr>
<td>12</td>
<td>---</td>
<td>---</td>
<td>LOCA</td>
</tr>
<tr>
<td>13</td>
<td>---</td>
<td>---</td>
<td>LOCE</td>
</tr>
<tr>
<td>14</td>
<td>---</td>
<td>---</td>
<td>GSARM</td>
</tr>
<tr>
<td>15</td>
<td>---</td>
<td>---</td>
<td>GSENG</td>
</tr>
<tr>
<td>16</td>
<td>---</td>
<td>---</td>
<td>LANDR</td>
</tr>
<tr>
<td>17</td>
<td>---</td>
<td>---</td>
<td>GSTRK</td>
</tr>
<tr>
<td>18</td>
<td>---</td>
<td>---</td>
<td>ONCRS</td>
</tr>
</tbody>
</table>

* Entries 8 & 9 of FAIL2/DSPLF are used to record autothrottle & MLS failures, but this has no relationship to MODEX or the flight modes.

### GLOBAL INPUTS:

ACWSS, AFCSS, AFSV, AGCSS, AUTOS, COSRH, CXTSW, DESTIN, DZNE, FAIL2, FCUL, FFDS, FLAGS, FLYFLG, FWHL, GAS, GSA, GSDEV, GSVLD, HDD, HDTGRU, HDOTB, HOLD, IC, INAVY, LABFLG, LANDS, LUCDEV, LOCVL, LOCYLD, MCONF, MLSDMD, MLSRAW, MSWIT, POSHAT, RLMLS, RUN, RWHDEF, RWHYDG, RWHYSEL, SIRH, SQUAT, VE, VN, WSPIN.

### GLOBAL OUTPUTS:

AEE, CRSET, CXTVEL, DELAY, DLPSI, DSPLF, ERSET, ETAVL, FLAGS, GAE, GPGSDV, GPOCD, HDGF, HGPIP, HOLOM, HTDZ, ICM, LBS, LOCVL, MLSC, MLSM, MLSVAL, MODE2, MODEX, MSW1, MSW6, RUNM, SIMFLG, VBS, VCWSS, XFLARE, XGPIP, XTVEL, YPROF.
MODULE NAME: PANEL

PURPOSE: Services the system test panel requests.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: FAST

CALLING SEQUENCE: CALL PANEL

CALLS TO: FMTMG

DESCRIPTION:
The following system test panel buttons are checked, and if they have been pushed the appropriate operation is performed:

1.) PANEL RESET - Results in a hardware reset of the system test panel. If this button is depressed, PANEL is exited with no further button checks.

2.) MODE FAIL - Turns off the mode fail lamp.

3.) PANEL TEST - Produces a momentary illumination of all lights and display segments for test purposes.

4.) DATA CLEAR - Clears failure data table and blanks display.

5.) STATUS ALERT - Displays status alert (mode-failure) messages on the system test panel.

6.) FAILURE READ - Displays recorded sensor failure messages on the system test panel.

This routine shares common local data with FDSTR, DSTOR, FMTMG, and F2CMP.

GLOBAL INPUTS: FALST, MSWIT, SWITCH, WRDCNT, FSIDX

GLOBAL OUTPUTS: SWITCH, LIGHTS, MSGST, WRDCNT, MSBUF
MODULE NAME: SINUSE

PURPOSE: To tell the SSFD which analog sensors it is required to check at any given time.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: FAST

CALLING SEQUENCE: (If not in preflight) CALL SINUSE

CALLS TO: None

DESCRIPTION:
SINUSE outputs four packed discrete words called SINUSO, SINUS1, SINUS2, and SINUS3 which contain information on which redundant sensors the SSFD should vote. The selection of which sensors to vote is made by virtue of the flight mode and condition of the aircraft at any given time. These four words are sent to NORDEN #1 for use by the SSFD. The last of the four words contains no sensor information but does contain the computer reset, hardware SSFD enabled, displays inoperative, and INS single thread flags. These four words are packed as follows.

<table>
<thead>
<tr>
<th>BIT</th>
<th>SINUSO</th>
<th>SINUS1</th>
<th>SINUS2</th>
<th>SINUS3</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Q</td>
<td>SPR7</td>
<td>VEINS</td>
<td>CRSET</td>
</tr>
<tr>
<td>14</td>
<td>P</td>
<td>PEDAL</td>
<td>THDG</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>R</td>
<td>ATRIM</td>
<td>LATINS</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>HDD</td>
<td>PITCH</td>
<td>LONINS</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>HRAD</td>
<td>ROLL</td>
<td>INSMV</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>FWHL</td>
<td>DRPOS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>HDOTB</td>
<td>MACH</td>
<td></td>
<td>HRDW-SSFD</td>
</tr>
<tr>
<td>8</td>
<td>GSDEV</td>
<td>HBARO-C</td>
<td>DISP-INOP</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>LOCDEV</td>
<td>HBARO-F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>RTRIM</td>
<td>TAS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>FCOL</td>
<td>XTKACC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>DCOL</td>
<td>GSINS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>DWHL</td>
<td>DLPSI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CAS</td>
<td>XTVEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>FLAP</td>
<td>ATKACC</td>
<td></td>
<td>INSST</td>
</tr>
<tr>
<td>0</td>
<td>SPL2</td>
<td>VNINS</td>
<td></td>
<td>INSST</td>
</tr>
</tbody>
</table>

GLOBAL INPUTS: FLAGS, MLSMOD, VBS, LBS, INSST, CRSET

GLOBAL OUTPUTS: SINUSO, SINUS1, SINUS2, SINUS3
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.
MODULE NAME: CLBIS

PURPOSE: This preflight procedure sets the airplane control surfaces for testing by SRVCK.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: FAST

CALLING SEQUENCE: IF MSWIT = 1 (System Test Panel mode switch in pre-flight and SQUAT and not WSPIN) CALL 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 PRFLT. If its INHIBT bit is set this program is exited. If not, CLBIS will either reset the control surfaces (ailerons, 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
ELEVATOR : 13.84 DEG FLY UP

This routine shares common local data with all other preflight modules.

GLOBAL INPUTS: None

GLOBAL OUTPUTS: AILCMD, RUDCMD, DECMD
MODULE NAME: CTLCK

PURPOSE: This preflight test checks the various controls for proper displacement by the operator.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: FAST

CALLING SEQUENCE: IF MSWIT = 1 (System Test Panel mode switch in preflight and SQUAT and not WSPIN) CALL 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 RUDR TRM RT-LT (Turn rudder trim knob right and left > 7.4 DEG)
PUSH RUDR PEDALS RT-LT (Push rudder pedals right and left > 2.64 DEG)
TURN AILRN TRM RT-LT (Turn aileron trim knob right and left > 5.05 DEG)
ROLL PMC RWD-LWD (Turn wheel panel mounted controller RT-LT > 27.7 DEG)
PUSH-PULL PMC ND-NU (Push-pull column panel mounted controller > 1.7 IN)
RE-CTR AILRN TRM (Re-center aileron trim < 1.01 DEG)
RE-CTR RUDR TRM (Re-center rudder trim < 0.83 DEG)

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: WRDCNT, RTRIM, ATRIM, PEDAL, DWHL, DCOL, MESG (pool of messages)

GLOBAL OUTPUTS: FLAGS
MODULE NAME: ILSRC

PURPOSE: To test the ILS receivers during preflight testing.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: FAST

CALLING SEQUENCE: IF MSWIT = 1 (System Test Panel mode switch in preflight and SQUAT and not WSPIN) CALL 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 (LOC between 0.8911 and 1.074 deg. and GS between 0.3274 and 0.4126 deg.) to the stimulus provided by PRFLT immediately after pushing the start-test button. 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: LOCFS, LOCVD, LOCDEV, GSVLD, GSDEV, MESG (pool of messages)

GLOBAL OUTPUTS: FIDENT, STRUA, STRUB, STRUC, FLAGS
MODULE NAME: PRFLT

PURPOSE: To initiate and control the system preflight tests.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: FAST

CALLING SEQUENCE: IF MSWIT = 1 (System Test Panel mode switch in pre-flight and SQUAT and not WSPIN) CALL PRFLT

CALLS TO: FMTMG

DESCRIPTION:
The preflight system is composed of seven modules that are called serially from FAST (if in preflight mode) in the following order:

CALL PRFLT ; Preflight control procedure.
CALL CTLCK ; Control check test.
CALL CLBIS ; Control bias procedure.
CALL RDALT ; Radio altimeter test.
CALL ILSRC ; ILS self test.
CALL RGYRO ; Rate gyro test.
CALL 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:

BIT 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 the first 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 on 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 and once detected clears the INHIBIT word so that all tests begin running. After all the tests are completed (each test sets its own INHIBIT 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: PMSWIT, MSWIT, AEE, AFCSV, WRDCNT, FAIL2, SWITCH, LIGHTS, MESG (pool of messages)

GLOBAL OUTPUTS: MODEX, MODE2, FLAGS, MSGST, WRDCNT, LIGHTS, STRUA, STRUB, STRUC, AEE, SWITCH, MSBUF
MODULE NAME: RDALT

PURPOSE: To test the radio altimeters during preflight.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: FAST

CALLING SEQUENCE: IF MSWIT = 1 (System Test Panel mode switch in preflight and SQUAT and not WSPIN) CALL 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 ft. 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: BF

GLOBAL OUTPUTS: BF, STRUA, STRUB, DSTAT
MODULE NAME: RGYRO

PURPOSE: To test the pitch, roll, and yaw rate gyros during preflight testing.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: FAST

CALLING SEQUENCE: IF MSWIT = 1 (System Test Panel mode switch in preflight and SQUAT and not WSPIN) CALL 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: Q, P, R

GLOBAL OUTPUTS: STRUA, STRUB, STRUC, BF
MODULE NAME: SRVCK

PURPOSE: Tests the aileron, elevator, and rudder servos during preflight testing.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: FAST

CALLING SEQUENCE: IF MSWIT = 1 (System Test Panel mode switch in pre-flight and SQUAT and not WSPIN) CALL 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. After being called 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.) LATAX - 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: SPR7, ALVDT, DEPOS, DRPOS, STABP, FLAP, MESG (pool of messages)

GLOBAL OUTPUTS: AILCMD, RUDCMD, DECMD, FIDENT, LIGHTS, TRIMT, TRIMD
6.0 FLIGHT MANAGEMENT
FLIGHT MANAGEMENT OVERVIEW

The Flight Management routines provide 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 NCDU (Navigation Control and Display Unit) and MSP (Mode Select Panel). The look-up and usage of the system data base (Bulk Data) is done via the NCDU.

The NCDU can be used to create a flight path that the aircraft will follow when some of the various automatic modes of flight are chosen.

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.
MODULE NAME: ALTGSX

PURPOSE: Set or clear guidance possible and path incomplete flags.

TASK: SLOW

LANGUAGE: MACRO-11

CALLED BY: PATHDF, ATCMOD

CALLING SEQUENCE: CALL ALTGSX

CALLS TO: None

DESCRIPTION:

When path changes have been made, this routine determines if guidance modes 3D and 4D are possible for the current provisional path. Guidance 3D is possible if there are 3 or more waypoints on the path and each has a non-zero altitude entry. In this routine 4D is possible (PDG4D is ON) when 3D is valid (PDG3D is ON) and each waypoint has a nonzero ground speed entry. Prior to making the provisional path the active path, procedure SRPTA is called from procedure NCDGUD which will clear PDG4D if no planned time of arrival has been entered. When the provisional path is made the active path, the provisional guidance possible flags (PDG3D, PDG4D) are copied into the active guidance possible flags (GUID3D, GUID4D). To engage guidance modes in actual flight, the horizontal and vertical path buttons on the mode control panel must be manually selected.

If either of the provisional guidance possible flags is cleared (mode not allowed), the altitude/groundspeed failure flag (ALTGSF) is set to cause the "PATH INCOMPLETE" message to be displayed on the NCDU.

GLOBAL INPUTS: PWNUM, PWEND, PGRUF, WPSIZ

GLOBAL OUTPUTS: PDG3D, PDG4D, ALTGSF
MODULE NAME: BTOBCD

PURPOSE: To convert floating point values into three digit BCD for MSPRO.

TASK: FAST

LANGUAGE: MACRO-11

 CALLED BY: MSPRO

 CALLING SEQUENCE: CALL BTOBCD

 CALLS TO: None

DESCRIPTION:

 DTOBCD converts a PDP-11 floating point number into a three digit binary coded decimal representation for use by the mode control panel display windows. The result replaces the input.

GLOBAL INPUTS: None

GLOBAL OUTPUTS: None
MODUEL NAME: BUFSWT

PURPOSE: Move the provisional guidance and vector buffers to their active counterparts.

TASK: SLOW

LANGUAGE: MACRO-11

CALLING SEQUENCE: CALL BUFSWT

CALLS TO: None

DESCRIPTION:

BUFSWT is called when a provisional path is being made active. The provisional path will have all its static features computed (PATHDF) and all automatic guidance modes turned off before BUFSWT is called. Exact copies of the provisional waypoint and guidance vector buffers (PGBUF,PVECTS) are moved to their active counterparts, ACTIVE and AVECTS. The modes of guidance that will be allowed are determined from PATHDF flags (PDG3D,PDG4D), and the guidance possible flags (GUID2D,GUID3D,GUID4D) are set appropriately.

GLOBAL INPUTS: BADRAD, PDG3D, PDG4D, PGBUF, PTRSTA, PVECTS, PWEND, PWNUM

GLOBAL OUTPUTS: ACTIVE, AVECTS, GUID2D, GUID3D, GUID4D, WPEND, WPNUM
MODULE NAME: MSPLGC

PURPOSE: To process inputs from the Control Mode Panel (CMP) and to perform the logic associated with the various configurations of the AUTO flight mode.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: FAST

CALLING SEQUENCE: CALL MSPLGC

CALLS TO: None

DESCRIPTION:

MSPLGC consists of two fundamental parts. The first part handles inputs from the CMP knobs (using the local subroutine KNOBER) and buttons (less the bottom-left four which are handled by MLOG). The second section contains the logic used to calculate which AUTO flight 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.

\[
\text{PSTTKA} = \text{NOT(TKSEL)} \land (\text{RTKNOB} \lor (\text{NOT(D2D)} \land \text{PSTTKA}))
\]

where RTKNOB is a boolean set by turning the track knob, and D2D is a momentary boolean set when the horizontal button is pushed.

\[
\text{HORARM} = \text{GUID2D} \land \text{AUTOE} \land \text{NOT(LOCENG)} \land ((\text{NOT(HORARM)} \land \text{D2D}) \lor (\text{NOT(DTKSEL)} \land \text{NOT(D2D)} \land \text{HORARM}))
\]

where DTKSEL is a momentary boolean set when the track button is pushed.

\[
\text{HORPTH} = \text{HORARM} \land \text{NOT(BCFLAG)}
\]

\[
\text{TKSEL} = \text{AUTOE} \land \text{NOT(LOCENG)} \land \text{NOT(HORPTH)}
\]

\[
\text{PSTALT} = \text{NOT(ALTARM)} \land (\text{RAKNOB} \lor (\text{NOT(D3D)} \lor \text{DFPSEL}))
\]

where RAKNOB is a boolean set when the altitude knob is turned, D3D is a momentary set when the vertical button is pushed, and DFPSEL is a momentary set when the FPA button is pushed.

\[
\text{PSTFPA} = \text{NOT(FPASEL)} \land (\text{RFKNOB} \lor (\text{NOT(D3D)} \lor \text{DALSEL}))
\]

where RFKNOB is set when the FPA knob is turned and DALSEL is a momentary set when the altitude button is pushed.

\[
\text{VERARM} = \text{GUID3D} \land \text{AUTOE} \land \text{NOT(GSENG)} \land (\text{HORPTH} \lor \text{LOCENG}) \land ((\text{D3D} \land \text{NOT(VERARM)}) \lor (\text{VERARM} \land \text{NOT(D3D} \lor \text{DALSEL} \lor \text{DFPSEL})))
\]

\[
\text{VERPTH} = \text{VERARM} \land (\text{VERPTH} \lor ((\text{ABS(HER)} < 152') \lor ((\text{HER} > 0) \land (\text{VSTRA} < \text{VSTRB})) \lor ((\text{HER} < 0) \land (\text{VSTRA} > \text{VSTRB})))
\]

where HER is vertical path error, VSTRA is vertical velocity commanded, and VSTRB is gained vertical velocity.
ALTARM = AUTOE AND NOT(GSENG) AND ((NOT(ALTARM) AND DALSEL) OR
(ALTARM AND NOT(DALSEL OR D3D OR DFPSEL)))

ALTSEL = ALTARM AND
((ABS(DELALT) < 1200') AND ALTSEL) OR ((ABS(DELALT) <152')) OR
((DELALT >= 0) AND (DELALT < 20*HDCF)) OR
((DELALT < 0) AND (DELALT > 20*HDCF))

where DELALT is altitude select/hold error and HDCF is filtered vertical velocity.

FPASEL = AUTOE AND NOT(GSENG) AND NOT(VERPTH OR ALTSEL)

PSTIAS = NOT(IASSEL) AND (RCKNOB OR (NOT(D4D) AND PSTIAS))

where RCKNOB is set when the CAS knob is turned and D4D is a momentary set when the time path button is pushed.

TIMARM = AUTOE AND GUID4D AND VERARM AND NOT(TOSLOW OR ATDC) AND
((NOT(TIMARM) AND D4D) OR (TIMARM AND NOT(D4D OR DIASEL)))

where DIASEL is a momentary set when the CAS button is pushed.

TIMPTH = TIMARM AND VERPTH AND (NOT TIMPTH OR (TIMPTH AND ATE))

where ATE is autothrottle engaged.

IASSEL = NOT(ATDC) AND ((IASSEL AND NOT(DIASEL OR TIMPTH)) OR
(NOT(IASSEL) AND DIASEL) OR (NOT(TIMPTH) AND PTIMPT AND
NOT(IASSEL) AND NOT(DIASEL)))

where ATDC is autothrottle disconnect flag and PTIMPT is TIMPTH from the previous frame.

These equations must be executed in the order they appear here.

This routine shares common local data with MSPRO.

GLOBAL INPUTS: BF, COLDST, TOGGLE, ALTSUM, FPASUM, TKASUM, IASSUM, FLAGS,
TKSEL, IASREF, GUID2D, GUID3D, GUID4D, BCFLAG, TOSLOW,
ALTARM, FPASEL, HORPTH, VERPTH, TIMPTH, ALTCOR, VSTRA, VSTRB,
HER, ALTSEL, DELALT, HDCF, IASSEL, ATDC, MSPER, ATE.

GLOBAL OUTPUTS: BF, ALTSUM, FPASUM, TKASUM, IASSUM, ALTATT, PSTTKA,
HORPTH, VERPTH, TIMPTH, TKSEL, PSTALT, PSTFPA, ALTARM,
ALTSEL, FPASEL, PSTIAS, IASSEL, MDWARN, MSPER
MODULE NAME: MSPRO

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.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: FAST

CALLING SEQUENCE: CALL MSPRO

CALLS TO: BTOBCD

DESCRIPTION:
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).

<table>
<thead>
<tr>
<th>BIT</th>
<th>BOOLEAN</th>
<th>MEANING</th>
<th>LAMP COLOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>PSTTKA</td>
<td>PRESELECT TRACK ANGLE MODE</td>
<td>BLUE</td>
</tr>
<tr>
<td>10</td>
<td>TKASEL</td>
<td>TRACK ANGLE MODE ENGAGED</td>
<td>GREEN</td>
</tr>
<tr>
<td>11</td>
<td>PSTFPA</td>
<td>PRESELECT FLIGHT PATH ANGLE</td>
<td>BLUE</td>
</tr>
<tr>
<td>13</td>
<td>FPASEL</td>
<td>FLIGHT PATH ANGLE MODE ENGAGED</td>
<td>GREEN</td>
</tr>
<tr>
<td>14</td>
<td>PSTALT</td>
<td>PRESELECT ALTITUDE HOLD</td>
<td>BLUE</td>
</tr>
<tr>
<td>15</td>
<td>ALTARM</td>
<td>ALTITUDE HOLD MODE ARMED</td>
<td>AMBER</td>
</tr>
</tbody>
</table>

DSTOMP+2

<table>
<thead>
<tr>
<th>BIT</th>
<th>BOOLEAN</th>
<th>MEANING</th>
<th>LAMP COLOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ALTSEL</td>
<td>ALTITUDE HOLD MODE ENGAGED</td>
<td>GREEN</td>
</tr>
<tr>
<td>1</td>
<td>PSTCAS</td>
<td>PRESELECT CALIBRATED AIR SPEED</td>
<td>BLUE</td>
</tr>
<tr>
<td>3</td>
<td>IASSEL</td>
<td>CALIBRATED AIR SPEED ENGAGED</td>
<td>GREEN</td>
</tr>
<tr>
<td>4</td>
<td>TIMARM</td>
<td>TIME PATH MODE (4-D) ARMED</td>
<td>AMBER</td>
</tr>
<tr>
<td>5</td>
<td>TIMPTH</td>
<td>TIME PATH MODE (4-D) ENGAGED</td>
<td>GREEN</td>
</tr>
<tr>
<td>6</td>
<td>VERARM</td>
<td>VERTICAL PATH MODE (3-D) ARMED</td>
<td>AMBER</td>
</tr>
<tr>
<td>7</td>
<td>VERPTH</td>
<td>VERTICAL PATH MODE (3-D) ENGAGED</td>
<td>GREEN</td>
</tr>
<tr>
<td>8</td>
<td>HORARM</td>
<td>HORIZONTAL PATH MODE (2-D) ARMED</td>
<td>AMBER</td>
</tr>
<tr>
<td>9</td>
<td>HОРPTH</td>
<td>HORIZONTAL PATH MODE (2-D) ENGAGED</td>
<td>GREEN</td>
</tr>
</tbody>
</table>

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 no mode is preselected or engaged).

This routine shares common local data with MSPLGC.
GLOBAL INPUTS: PSTTKA, TKSFL, PSTFPA, FPASEL, PSTALT, ALTSEL, ALTARM, PSTIAS,
ATE, IASSEL, TIMPTH, VERPTH, HER, HORPTH, ALTSM, FPASUM,
TKASUM, MAGVAR, IASSUM, ALTCOR, IASREF, ALTOMP, FPTOMP,
TKTOMP, ASTOMP

GLOBAL OUTPUTS: DSTOMP, ALTOMP, FPTOMP, TKTOMP, ASTOMP
MODULE NAME: PATHDF

PURPOSE: To process flight plan parameters entered via the NCDU, or extracted from Bulk Data, and perform preliminary calculations required for path guidance.

LANGUAGE: HAL/S

CALLED BY: NCDGUD

CALLING SEQUENCE: Call PATHDF

CALLS TO: SRPTA

DESCRIPTION:
The Path Definition routine uses, as input, a provisional guidance buffer (PGBUF) that has been partially built via the Air Traffic Clearance (ATC) Page of the Navigation Control Display Unit (NCDU) and computes the following parameters:

1. The waypoint turn radius (PWRTN).
2. The change in track angle required at each waypoint (PWTA).
3. The length of the circular arcs to be flown at each waypoint (PWAO2).
4. The distance between the tangent point inbound at the next waypoint and the next waypoint on the path (PWDTT).
5. The great circle distance between each pair of successive waypoints (PWPPD).
6. The flight path gradient between each pair of successive waypoints (PWGAM).
7. The center of turn to the center of turn distance for each pair of successive waypoints (PWCCD).
8. The incremental time to fly from center of turn to the center of turn at each successive pair of waypoints on the path (PWT).
9. The waypoint unit vector from the center of the earth pointing toward each waypoint (PWP1).
10. The path normal unit vector which is normal to the plane formed by the waypoint it is associated with, the previous waypoint, and the center of the earth (PWU12).
11. The waypoint center of turn unit vector from the earth center to the center of turn at each waypoint (PWTC2).

This routine is activated by NCDGUD as a result of the provisional execution flag (PRVDEF) being set when a function has been entered on the ATC page or Flight Plan page. It is also called when a request has been made to make the provisional guidance buffer the active guidance buffer by depressing the execute button on the ATC or Flight Plan pages. During the path activation process the following flags are set to determine the possible guidance modes:

(GUID2D) set indicates 2D guidance possible,
(GUID3D) set indicates 3D guidance possible,
(GUID4D) set indicates 4D guidance possible.
Each higher mode requires all lower mode information plus additional data. A 2D path needs altitude information to become a 3D path and the addition of a planned time of arrival makes a 3D path a 4D path.

When defining a path via the ATC page, either one waypoint or a group of waypoints may be added to the path with each line of data entered by the pilot. A group of waypoints is entered by selecting the type of group (SID, STAR, etc.) followed by the name. Procedures are then called by the ATC subroutines to search Bulk Data and add the waypoint descriptor information to the provisional guidance buffer.

The guidance system utilizes scalar and vector waypoint descriptor blocks. The scalar waypoint descriptor block contains 26 descriptors per waypoint. These descriptors consist of integer, boolean, and floating point variables occupying a total of 45 words of storage for each waypoint. The vector waypoint descriptor block contains 3 unit vector descriptors of 3 scalars each, or 18 words (2 words per scalar entry) per waypoint descriptor. Each buffer handles a maximum of 30 waypoints, therefore the scalar buffer requires \((45\times30 = 1350)\) words and the vector buffer requires \((18\times30 = 540)\) words. These buffers are defined via HAL/S structures GD_BUFFER for the scalar buffer and VECTOR_BUFFER for the vector buffer (see Table 6-1).

On the first pass of PATHDF, the initialization code path is taken. This is determined by the next waypoint pointer (NXWP) being less than or equal to one in the PD2 procedure. Thereafter processing continues by calling PD3A, PD4, and PD5A which are all imbedded in the PATHDF procedure. The path definition process is terminated by the last pass flag (LASTPASS) being set in the PDEND procedure as a result of a zero name being detected in the guidance buffer or an active waypoint detected in the provisional buffer. An active waypoint will have the boolean (PWAY) provisional waypoint flag off. Each pass through path definition (controlled by PDLOOP) will complete the calculations for one waypoint. Except at the path endpoints, the computations require vector and scalar information on the two waypoints adjacent to the waypoint being processed. For the first and last waypoints on the path those parameters requiring a preceding and following waypoint respectively are not computed since such waypoints do not exist.

After the first call to the PD2 procedure, a test is made for a DME arc type waypoint as indicated by a boolean flag (PWAYMA). Special computations are required for DME arc turns which are discussed later. The normal code path is then followed through Path Definition. If after an entire path has been entered and any waypoint does not have an altitude or groundspeed, PATHDF causes a "PATH INCOMPLETE" message to appear on the NCDU via the Boolean variable ALTGSF.

Following the computation of the great circle distance between waypoints, a test is made to determine if the ARC radii are acceptable. The test is made by comparing the point to point distance between two waypoints to the combined tangent distances for the waypoints. If the point to point distance is less than the combined tangent distances then a bad radius message will be displayed on the NCDU for the current waypoint. The problem may be resolved by using a smaller radius at WPT(i) and/or WPT(i-1).
This and the following pages describe the computations required for building the guidance buffer and vector guidance buffer using the following symbology:

### TABLE 6-1

<table>
<thead>
<tr>
<th>Structure GD Buffer:</th>
<th>/* Guidance Buffer */</th>
<th>*/</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 PWNAM ARRAY(3) INTEGER,</td>
<td>/* WPT NAME */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWWAY BOOLEAN,</td>
<td>/* PROVISIONAL? */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWFL BOOLEAN,</td>
<td>/* FLIGHT LEVEL? */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWIRL BOOLEAN,</td>
<td>/* IAS REFERENCE? */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWDMA BOOLEAN,</td>
<td>/* DME ARC? */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWDMI BOOLEAN,</td>
<td>/* DME ARC STOP CALC */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWRAI BOOLEAN,</td>
<td>/* RADIUS INHIBIT? */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PNAS INTEGER,</td>
<td>/* 0=AWY1=SID2=STAR3=MAP4=N */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWLAT SCALAR,</td>
<td>/* LATITUDE DEG. */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWLON SCALAR,</td>
<td>/* LONGITUDE DEG. */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWH SCALAR,</td>
<td>/* ALTITUDE FT. */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWV SCALAR,</td>
<td>/* GROUND SPEED KNOTS */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWRTN SCALAR,</td>
<td>/* RADIUS OF TURN FT. */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWT SCALAR,</td>
<td>/* DELTA TIME EN ROUTE SEC. */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWPPD SCALAR,</td>
<td>/* CIRCLE PT TO PT DIST FT. */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWCCD SCALAR,</td>
<td>/* CENTER TO CENTER DIST FT. */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWGAM SCALAR,</td>
<td>/* PATH GRADIENT DEG. */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWTA SCALAR,</td>
<td>/* TURN ANGLE AT WPT DEG. */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWAO2 SCALAR,</td>
<td>/* ARC LENGTH OF CURVE 0 2 FT. */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWDTT SCALAR,</td>
<td>/* TANGENT PT TO WPT DIST FT. */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWPTA SCALAR,</td>
<td>/* PLANNED TIME OF ARRIVAL SEC. */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWMV SCALAR,</td>
<td>/* MAGNETIC VARIATION DEG. */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWVTR INTEGER,</td>
<td>/* ADDR OF VOR */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWBREG SCALAR,</td>
<td>/* BEARING DEG. */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWLTRF SCALAR,</td>
<td>/* DME ARC REFERENCE LAT DEG. */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWLGRF SCALAR,</td>
<td>/* DME ARC REFERENCE LON DEG. */</td>
<td>*/</td>
</tr>
</tbody>
</table>

**Scalar Guidance Buffer**

<table>
<thead>
<tr>
<th>Structure Vector Buffer:</th>
<th>/* Waypoint Unit Vector */</th>
<th>*/</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 PWPI VECTOR(3),</td>
<td>/* Normal Unit Vector */</td>
<td>*/</td>
</tr>
<tr>
<td>1 PWU12 VECTOR(3),</td>
<td>/* Center of Turn Unit Vector */</td>
<td>*/</td>
</tr>
</tbody>
</table>

**Vector Guidance Buffer**
\hat{\text{a}} \text{ implies a unit vector}
- \text{ implies a vector}
\times \text{ denotes cross product}
. \text{ denotes dot product}

The waypoint unit vector is defined by:

\[ \hat{\text{WP}} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \sin \text{LAT} \\ -\cos \text{LAT} \sin \text{LON} \\ \cos \text{LAT} \cos \text{LON} \end{bmatrix} \]

The X, Y, Z components for WP are stored at the base address identified by PWPL.

The vector to the waypoint is described in terms of the unit vector WP as follows:

\[ \text{WP} = \text{Re} \hat{\text{WP}}, \text{ where Re is the earth's radius.} \]

The relationship of these vectors to the guidance software coordinate system is shown in figure 6-1. The positive X-axis pierces the North Pole and positive longitudes are measured eastward from the Greenwich Meridian.

\[ \text{FIGURE 6-1. GUIDANCE SOFTWARE COORDINATE SYSTEM} \]
Normal Unit Vector

The normal unit vector is computed as follows:

\[ \mathbf{U}_{12}(i+1) = \frac{\mathbf{W}_{P}(i) \times \mathbf{W}_{P}(i+1)}{|\mathbf{W}_{P}(i) \times \mathbf{W}_{P}(i+1)|} \]

where \( i \) denotes the current waypoint (CWP) and \( (i+1) \) denotes the next waypoint (NXWP).

The components of the cross product \( \mathbf{W}_{P}(i) \times \mathbf{W}_{P}(i+1) \) are stored at the base address identified by the label \( \text{PWU12} \) (see TABLE 1)

\[ \text{FIGURE 6-2. NORMAL UNIT VECTOR} \]
Track Angle Change at Each Waypoint

The normal unit vector and the waypoint unit vectors are used to compute the track angle change (i) at each waypoint using the following equations:

\[
\sin \Delta \psi(i) = -\hat{U}_{12}(i) \times \hat{U}_{12}(i+1) \cdot \hat{W}(i) \\
\cos \Delta \psi(i) = \hat{U}_{12}(i) \cdot \hat{U}_{12}(i+1)
\]

where (\cdot) is the dot product and (x) is the cross product between vectors. The track angle change is given by:

\[
\Delta \psi(i) = \arctan2(\sin \Delta \psi(i), \cos \Delta \psi(i))
\]

This value is stored at provisional base address identified by the label PWTA (see TABLE 6-1).

The relationship between the waypoint vectors, the normal unit vectors which define the plane of the path and the track angle change is shown in FIGURE 6-2. These computations are performed for all but the first and last waypoints on the path.
The Length of the Circular Arches to be flown at each Waypoint

The ARC length of the curve over 2 is computed using the following equation:

$$A_02(i) = RTN*\left(\frac{PW(i)}{360}\right)\pi$$

where RTN is the radius of turn computed in local routine SRRTN. This value is stored at the provisional base address identified by the label PWA02.

Figure 6-3. Arc Distance of Curve/2
The Tangent Point to Waypoint Distance

The tangent point to waypoint distance is computed from the track angle change and the radius of the turn:

\[ DTT(i) = RTN \times \left[ \frac{\sin(\Delta \psi_i)}{1 + \cos(\Delta \psi_i)} \right] \]

This value is stored at the provisional base address identified by the label PWDTT (see TABLE 6-1).

Figure 6-4. Tangent Point to Waypoint
The Turn Center Vectors

The waypoint to tangent point vector is computed from the tangent point to waypoint distance, the waypoint unit vectors and the normal unit vector by:

\[ \text{TP} = R \text{e} \text{WP}(i) + \text{DTT}(i) [\text{WP}(i) \times \text{U}_12(i)] \]

where \( R \) is the earth's radius and \( \times \) is the cross product between the two vectors. This vector is then used to compute the center of turn unit vector in the following manner:

\[ \hat{C}(i) = \frac{\text{TP}(i) + \text{RTN} \times \text{U}_12(i)}{\text{TP}(i) + \text{RTN} \times \text{U}_12(i)} \]

where the + sign applies if \( \Delta \psi(i) \) is negative and the - sign applies if \( \Delta \psi(i) \) is positive. The relationship between the waypoint vectors, the waypoint to tangent point vector and the center of turn vector where:

\[ \hat{C}(i) = \text{TP}(i) + \text{RTN} \times \text{U}_12(i) \]

is shown in FIGURE 6-5. This value is stored at the provisional base address identified by the label PTTC2 (see TABLE 6-1).
Figure 6-5  Waypoint Center of Turn Vector
The Great Circle Distance Between Waypoints

The great circle distance between two successive waypoints is found by computing the angle between the unit vectors WP(i-1) and WP(i) and multiplying the result by the earth's radius. For maximum accuracy the sine and cosine of the angles are found, and these results are used in a double argument arctangent routine:

\[
\sin(r) = \hat{\text{WP}}(i-1) \times \hat{\text{WP}}(i) \\
\cos(r) = \hat{\text{WP}}(i-1) \cdot \hat{\text{WP}}(i)
\]

The great circle distance is then given by:

\[
\text{PPD}(i) = \Re \arctan2(\sin(\theta),\cos(\theta))
\]

which is stored at the base address in the provisional guidance buffer identified by the label PWPPD (see TABLE 6-1).

![Diagram of great circle distance](image)

**Figure 6-6** Great Circle Distance

After the great circle distance is computed a test is made to determine if the ARC radii are acceptable. The test is made on the tangent distances and failure implies that the tangents overlap as shown in FIGURE 6-7. If there is an overlap, the message "BAD RADIUS (WPT NAME)" is displayed on the NCDU.
Normal Radii

\[ PPD(I) \geq DTT(I-1) + DTT(I) \]

Bad Radii

\[ PPD(I) < DTT(I-1) + DTT(I) \]

FIGURE 6-7 TANGENT OVERLAP
Center-of-Turn to Center-of-Turn Distance

The center-to-center of turn distance is computed by adding to the great circle distance one-half the arcs at each end of the path leg and subtracting the straight segments from the tangent points to the waypoints. One-half the length of each arc is found by multiplying the track angle change by one-half the turn radius. Since these computations have been performed previously to build the waypoint descriptors (see TABLE 1), the provisional waypoint buffer is accessed to compute the center to center distance as follows:

\[ PWCCD(i) = PWPPD(i) - PWDTT(i) + PWAO2(i) - PWDTT(i-1) + PWAO2(i-1) \]

Figure 6-8 Center-to-Center Distance
The Flight Path Gradient Between Waypoints

The flight path gradient required between two successive waypoints is computed from the required altitude change and the center-of-turn to center-of-turn distance:

$$PWGAM(i) = \frac{PWH(i) - PWH(i-1)}{PWCCD(i) \times (\pi/180)}$$

Figure 6-9  Flight Path Gradient
The Incremental Time

The incremental time associated with the ith waypoint is a time computed from the groundspeeds assigned to the (i-1)th and ith waypoints and the center-of-turn to center-of-turn distance between them. The computation assumes a linear speed transition with distance as given by the equation:

\[ PWT(i) = \frac{PWCCD(i)}{((PWV(i-1)+PWV(i))/KTOFPS)/2} \]

where \( KTOFPS=1.687809858 \)

If a planned time of arrival at one waypoint is entered from the NCDU before Path Definition is executed, the Path Definition software assigns that time to the appropriate waypoint and by using the PWT(i)'s proceeds to assign arrival times to the remaining waypoints.
The DMEARC Label

In the beginning of the PATHDF routine the second waypoint word (PWNAM2) in the provisional waypoint buffer (see TABLE 6-1) is fetched and checked for the DME ARC bit; if it is set a transfer is made to the DMEARC label. This module has two basic tasks:

1. To compute certain DME ARC descriptive parameters
2. To set flags to modify the Path Definition logic to fulfill DME ARC requirements

A typical DME ARC is shown below:

where:

PWTA(OUT) is the angle turned from WPT(IN) to WPT(OUT), \( \theta \) is the bearing center to WPT(IN), \( \psi \) is the bearing center to WPT(OUT), PWRTN is the turn radius and (PWLTRF, PWLGRF) is the center of turn latitude and longitude.
Since the radius of turn of the DME ARC is provided in feet, this distance must be converted to degrees of latitude and longitude for several different calculations as given by:

\[ T_1 = \frac{PWRTN}{DLATFT} \]

\[ T_2 = \frac{PWRTN}{\cos(PWLTRF) \cdot DLATFT} \]

where DLATFT is the length of 1 degree of latitude in feet.

After the conversion, the latitude and the longitude of the inbound and outbound waypoints are computed as follows:

\[ PWLT(IN) = PWLTRF(IN) + T_1 \cdot \cos(PWBRG(IN)) \]

\[ PWLG(IN) = PWLGRF(IN) + T_2 \cdot \sin(PWBRG(IN)) \]

\[ PWLT(OUT) = PWLTRF(OUT) + T_1 \cdot \cos(PWBRG(OUT)) \]

\[ PWLG(OUT) = PWLGRF(OUT) + T_2 \cdot \sin(PWBRG(OUT)) \]

The DME ARC length and DME ARC Half Arc length are then computed as follows:

\[ PWCCD(OUT) = PWTA(IN) \cdot DTOR \cdot PWRTN(IN) \]

\[ PWAO2(IN) = PWCCD(OUT) / 2 \]

The DME ARC flag (ARCSKIP) is then loaded and tested; if = 0, set it to 3; else set to -1.
SRRTN - Compute Radius of Turn

There are three ways in which a turn radius may be assigned to a waypoint:

1. It may be assigned to the waypoint directly via the NCDU or a stored path such as a SID or a STAR. If a radius has been assigned, the boolean PWRAD in the provisional waypoint buffer is set inhibiting the radius of turn computation in this routine.

2. If no radius is assigned, but a groundspeed is designated, the turn radius is calculated from the following formulas:

   \[ R = \frac{(V^{**2})}{(g \cdot \tan(15 \text{ deg}))} \]

   where:

   \( R \) = radius of turn in feet

   \( V \) = groundspeed in ft/sec

   \( g \) = gravitational acceleration (32.174 ft/sec^2)

3. If neither a groundspeed nor turn radius is entered, the radius is assigned as follows:

   if the waypoint altitude is below 15,000 feet, \( R=15,000 \) feet

   if the waypoint altitude is 15,000 feet or higher, \( R = 50,000 \).
MODULE NAME: SRPTA

PURPOSE: To Compute the planned time of arrival (PTA) for each waypoint on the path, or if none has been entered, set the PTA to zero for each waypoint.

TASK: SLOW

LANGUAGE: HAL/S

CALLED BY: NCDGUD

CALLING SEQUENCE: CALL SRPTA

CALLS TO: None

DESCRIPTION:
This routine starts with a PTA at a given waypoint as defined on the ATC page or Flight Plan page and cascades times forward and/or backward from that point. Therefore, this routine is called each time a new PTA is entered via the NCDU and thus provides the capability to change the time guidance profile while remaining in the 3D guidance mode.

The delta time enroute (PWT) from the provisional guidance buffer is used to adjust the time at each waypoint based on the starting time (PTATIM) and the starting waypoint position (PTAPOS) as set by the ATC or Flight Plan page.

If a time greater than or equal to the current time (BINTME) is not found in the guidance buffer, then the pointer for the time box position is left at the first waypoint and 'TOTIME' set to zero to indicate that the time path is invalid. If 'PTAPOS', the position of the entered PTA on the path is zero upon entry to this routine, then the entire path has its PTA at each waypoint set to zero. In addition, the 4D guidance possible flag, PDG4D, is also cleared when PTAPOS is zero. When a valid time is entered, the PDG4D flag is set to permit the time guidance mode.

GLOBAL INPUTS: Provisional Guidance Buffer(PGBUF), PTAPOS, PWNUM, PTATIM, BINTME,PTAXFG,PTAFLG

GLOBAL OUTPUTS: PGBUF, WPPTR(2), PDG4D, TOTIME, PTAXFG, PTAPOS, PTATIM, PTAFLG
6.1 NCDU
MODULE NAME: ACTOPV
PURPOSE: Copy active waypoints to provisional guidance buffer.
TASK: SLOW
LANGUAGE: MACRO-11
CALLED BY: ATCMOD
CALLING SEQUENCE: CALL ACTOPV
CALLS TO: None

DESCRIPTION:
ACTOPV copies the active path, starting with the waypoint designated by the offset in STRACT, to the position in the provisional guidance buffer marked by STRPRV. The provisional guidance buffer end markers are updated to the new buffer-end position.

The following is a pictorial example of ACTOPV execution. Initially, the active guidance buffer contains waypoints 1 - 8 while the provisional buffer has a - f.

![Pictorial example of ACTOPV execution](image)

GLOBAL INPUTS: ACTIVE, STRACT, STRPRV, WPEND
GLOBAL OUTPUTS: PGBUF, PWEND, PNUM
MODULE NAME: ATCMOD

PURPOSE: Allows the entry of a flight plan via the NCDU.

TASK: SLOW

LANGUAGE: MACRO-11

CALLED BY: NCDUEX

CALLING SEQUENCE: CALL ATCMOD

CALLS TO: ACTOPV, C7T08, FRMTIM, INSELH, LUGUID, LURWY, WPINVT

DESCRIPTION:
When the "ATC" button on the NCDU has been pressed the ATC page display appears and the subroutine ATCMOD interprets user function and data entries for the creation of an aircraft flight plan. There are 15 legal function keys that may be used while on the ATC page. These functions can be divided into the following two classes.

Class #1 - Functions that require user text entry immediately following selection: WPT, AWY, FL, ALT, RTE, RWY, GS, SID, STAR, PTA

Class #2 - Functions having immediate effect upon selection: EXEC, UP, DOWN, CLR, REJ

As a prerequisite to reading on, the module document for DECODE should be read for a description of class #1 functions. Class #1 functions are used to select waypoints or their characteristics in defining the flight plan. The WPT, AWY, RTE, RWY, SID and STAR function keys are those used to define the path (waypoints) while FL, ALT, GS and PTA describe or override certain waypoint characteristics.

The following is an example ATC page display. The key strokes used to obtain this display are: (function keys noted by <> , <ENT> means pressing the data entry complete key)

<WPT>WFBBA<ENT><ALT>3000<ENT><GS>145<ENT><PTA>11.34<ENT><WPT>WFBBB<ENT><WPT>WFBBC<ENT>

As can be seen the ATC page display scrolls upward for each new waypoint entry.
Class #2 functions are used to perform desired functions once a flight plan has been started by the use of class #1 functions. The UP/DOWN keys are used to scroll the display when the number of entries made exceed the 6 allocated lines on the display. The CLR key is used to cancel a class #1 function entry before its completion.

As waypoint entries are being made via class #1 functions, the generated flight plan is considered provisional. If a flight plan was previously entered and accepted, it will still be considered the active flight plan during the entries of the provisional path. If no path was previously activated before the provisional entry, the NAV software does not recognize any flight plan as valid for flight. When enough information has been entered on the provisional flight plan, the bottom line message "EXEC OR REJ" appears. At this point the pilot may enter more waypoint information, press "EXEC" to activate the provisional path, or press "REJ" to clear out the last provisional entry. Two consecutive REJects will clear all provisional entries made. A provisional path is considered having adequate information to form an active flight plan when the path contains three or more waypoints and each waypoint has a non-zero altitude and ground speed entry. Note that altitude and ground speed data may be associated with a waypoint in the following three ways:

1) Direct user entry.
2) Default values obtained from waypoint group entries such as a STAR.
3) Cascaded values obtained from direct entries on previous waypoints.
When there is no active path entered in the system, provisional entries made on the ATC page are called "Original Clearance". Otherwise the entries made are called "Reclearance" operations. There are three methods of initiating the reclearance mode. One allows alterations to be made to the active path, another uses the current position of the aircraft to start a totally new path, and the last one allows the pilot to enter a Standard Instrument Departure (SID) from a runway following the successful touch down from the last path. In the first two cases the WPT function is pressed to start the reclearance operation. The last starts with pressing "SID".

An active path is modified by entering the name of a waypoint that exists on the active path somewhere ahead of the current aircraft position. This designates the position that the new path will break away from the old one. More waypoint information may then be entered through any of the class #1 functions. If the last waypoint inserted lies on the active path, the rest of the active waypoints after that point are automatically included in the new path. The pilot may execute or reject the provisional path once enough information has been entered.

To create a totally new path by reclearance, the WPT function is selected and the entry "POS" is made in response to the prompt. This creates a new waypoint at the current position of the aircraft. Then waypoint information is entered via class #1 functions as in standard original clearance. Note that there is no mechanism for rejoining the current active path.

The "SID" reclearance mode is identical in effect to the "POS" method. The only difference is that a group of waypoints forming a departure route becomes the new start of the provisional flight plan.

GLOBAL INPUTS: ACMODE, ACTIVE, ALTCOR, BADRAD, BINTME, CASKAL, CASKFL, CASKGS, DECADR, DECLAT, DECLON, DECMV, DECNAM, DECEAV, DECN, DECTYP, DECV, DESTIN, GS, GUID2D, GUID3D, GUID4D, LAT, LON, LUADDR, NCMESG, NEWDES, NEWPTA, NGIDON, ORIGIN, PGBUF, PTAPOS, PTAREJ, PTR2DN, PWEND, PWNUM, WAYADA, WAYNEW, WAYPNT, WPEND, WPNUM

GLOBAL OUTPUTS: ACTIVE, ACTPOS, ALRMES, ALTGSF, ANTLAT, ANTLON, ATCLN8, ATCLMWP, CASKAL, CASKFL, CASKGS, FPPWP, GDTIME, GSA, LATTML, LATEAD, LONCN, LONEAD, NCDERR, NEWPTA, NPAGE, NUPAGE, PATHN, PDG3D, PDG4D, PGBUF, PROV, PRVDEF, PRVEXC, PRVMP, PTAPOS, PTATIM, RWYCN, RWYHDG, RYELEV, SET4D, SMER61, SMER66, STRACT, STRPRV, WPPT
MODULE NAME: DEBUG
PURPOSE: Allows the display and modification of compool data via the NCDU
TASK: SLOW
LANGUAGE: MACRO-11
CALLED BY: NCDUEX
CALLING SEQUENCE: CALL DEBUG
CALLS TO: None

DESCRIPTION:
When the NCDU is to be used for monitoring compool data, the T-M-A knob (Test-Manual-Automatic) on the NCDU must be turned to the manual mode setting. While in manual mode the DEBUG display, consisting of 7 lines for compool entries and a message line, will always be displayed on the NCDU. To show the contents of a memory word, the number corresponding to a line on the display must be pressed. The prompt "C/OFF" then appears on the bottom line of the NCDU to queue the user for the compool and offset numbers for the desired memory cell. Note that the offsets for all compool variables are listed in a table contained on the file COMMON.DAT which is automatically created when the system is built. The compool numbers are assigned as follows:

<table>
<thead>
<tr>
<th>Number</th>
<th>Compool</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IOPOOL</td>
</tr>
<tr>
<td>2</td>
<td>NAVCOM</td>
</tr>
<tr>
<td>3</td>
<td>SLWCOM</td>
</tr>
<tr>
<td>4</td>
<td>FCPOOL</td>
</tr>
<tr>
<td>5</td>
<td>BCKCOM</td>
</tr>
<tr>
<td>6</td>
<td>FMRFCM</td>
</tr>
<tr>
<td>7</td>
<td>DISNAV</td>
</tr>
</tbody>
</table>

When the compool and offset has been entered, the chosen line on the display will have the line number, compool number, offset and contents of the memory location. Note that all numbers entered and displayed on the DEBUG page are octal values.

To change the contents of a memory location the item must first be displayed on one of the 7 lines. The user then presses the the "8" key to select change mode which causes the prompt "L/VAL" to be issued on the bottom line. At this point the line number of the data item and the new octal value are entered.

The entire page may be reset by pressing the reject key, and the clear key will cancel an error message given from an illegal entry.

The following is an example of displaying the memory contents at 326 bytes into DISNAV compool:

<table>
<thead>
<tr>
<th>USER ENTRY</th>
<th>LINE 8 MESSAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1&lt;ENT&gt;</td>
<td>PRESS #(1-8)</td>
</tr>
<tr>
<td>1&lt;ENT&gt;</td>
<td>C/OFF</td>
</tr>
<tr>
<td>7/326&lt;ENT&gt;</td>
<td>PRESS #(1-8)</td>
</tr>
</tbody>
</table>
(Manual Mode Example)

GLOBAL INPUTS: DECNUM, KEYUNL, MAMODE, NCDSPC, NCMESG, PAGSE

GLOBAL OUTPUTS: FUNUMB, NPAGE, NUPAGE
MODULE NAME: DECODE

PURPOSE: Parse NCDU data entries and set values for page routines to use.

TASK: SLOW

LANGUAGE: MACRO-11

CALLED BY: NCDUEX

CALLING SEQUENCE: CALL DECODE

CALLS TO: ANGL, ASCBIN, FORMAT, FORMLL, LLBIN, LUARP, LUGRP, LUJET, LUNAVA, LURTE, LUVIK, LUWAY, SCOSD, TIMBIN, WPINVT

DESCRIPTION:

DECODE parses a text string input that was entered in response to one of 12 NCDU line #8 prompts. When the variable DATNUM contains a non-zero value, a call to DECODE is made by the NCDU executive (NCDUEX) prior to calling the currently active NCDU page subroutine. DATNUM will contain a number designating the item type of the entry made, with values ranging from 1 to 12. The input ASCII text corresponding to the code in DATNUM is contained in the array NCMESG. The following is a list of the DATNUM values, prompt text, response examples and definitions of the entry.

<table>
<thead>
<tr>
<th>#</th>
<th>PROMPT</th>
<th>RESPONSE</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WPT</td>
<td>WFBB0710.5</td>
<td>waypoint specification</td>
</tr>
<tr>
<td>2</td>
<td>AWY</td>
<td>V38</td>
<td>airway name</td>
</tr>
<tr>
<td>3</td>
<td>RAD</td>
<td>4000</td>
<td>radius value (feet)</td>
</tr>
<tr>
<td>4</td>
<td>F/L</td>
<td>50</td>
<td>flight level value (hundreds of feet)</td>
</tr>
<tr>
<td>5</td>
<td>ALT</td>
<td>5000</td>
<td>altitude value (feet)</td>
</tr>
<tr>
<td>6</td>
<td>RTE</td>
<td>FAAR1</td>
<td>route name</td>
</tr>
<tr>
<td>7</td>
<td>RWY</td>
<td>22</td>
<td>runway number</td>
</tr>
<tr>
<td>8</td>
<td>GS</td>
<td>150</td>
<td>ground speed value (knots)</td>
</tr>
<tr>
<td>9</td>
<td>SID</td>
<td>ISL25</td>
<td>standard instrument departure route</td>
</tr>
<tr>
<td>10</td>
<td>STAR</td>
<td>WFB13</td>
<td>standard terminal arrival route</td>
</tr>
<tr>
<td>11</td>
<td>PTA</td>
<td>14.32.55</td>
<td>planned time of arrival</td>
</tr>
<tr>
<td>12</td>
<td>ARPT</td>
<td>KWAL</td>
<td>airport name</td>
</tr>
</tbody>
</table>

The DECODE processing of each of these items is described in the following sections. DATNUM is stored into the output variable DECNUM once DECODE has finished so the active NCDU page routine, looking for a non-zero DECNUM, will perform its function on the other DECODE outputs.

*** WPT ***

The waypoint entry is the most diverse of all the various entry types, having the following four different classes of acceptable inputs:

- Class #1: specific waypoint name
- Class #2: present position
- Class #3: waypoint, bearing and range
- Class #4: designated latitude and longitude
DECODE creates the following outputs for all four classes.

- **DECADR**: Address in memory of the item.
- **DECNAV**: Address of the navaid associated with the item.
- **DECMV**: Value of the magnetic variation at the item's position.
- **DECTYP**: Type code (NAVAID, ARPT, GRP, PPT).
- **DECLAT**: Latitude of the item.
- **DECLON**: Longitude of the item.
- **DECNAM**: Name of the item (from NCMESG).

Valid entries for class #1 are the names of existing navaids, airports, geographic reference points and pilot defined waypoints from the system database.

The user entry text for class #2 has two equivalent forms, POS or PPOS. Each causes a new waypoint, called a PPT for pilot defined waypoint, to be created in the system database with the aircraft's current position, speed and altitude.

The waypoint/bearing/range entry of class #3 allows the creation of a new waypoint positioned relative to an existing waypoint (navaid, GRP, airport). For example, if a waypoint is desired 5 1/2 nautical miles from the navaid CCV at a bearing of 15 degrees, the entry "CCVO155.5" is made. Note the bearing must be a three digit number.

A pilot defined waypoint may also be created by entering the desired latitude and longitude of its position (class #4). For example, the entry "N3745.9W0714.4" causes a waypoint to be created at north latitude 37 degrees 45.9 minutes and west longitude 75 degrees 14.4 minutes. Since longitude values can range up to 180 degrees, a three digit value must always be supplied.

*** AWY ***

The only entry format for the airway is the name of an existing airway defined in the system database. Airway names may be from two to six characters long, starting with the letters J or V. The J and V designate jet (above 18,000 feet) and victor (below 18,000 feet) airways. Airways define sets of waypoints connecting VOR radio beacons and are FAA recognized routes that are shown on navigational charts. The entry and exit waypoints on the airway must be supplied immediately before and after the airway entry. On exit, DECODE stores the number 8 in DECTYP and the address of the airway in DECADR.

*** RAD ***

A radius entry is made when the pilot wishes to override the default radius of a turn between two waypoints. The number of feet, up to six digits, is entered and the ASCII alpha-numeric string is converted into a machine format floating point number stored in the output variable DECVAL.

*** F/L ***

A flight level is an altitude entry in hundreds of feet. DECODE processes the three digit flight level by converting it to a floating point value, multiplying by 100 and storing the result in the output variable DECVAL.

*** ALT ***
An alpha-numeric string up to five digits in length must be in NCMESG for a valid ALT input. DECODE converts the string to a floating point value and stores the result in DECVAL.

*** RTE ***

Routes are groupings of airways defined by individual organizations to define paths connecting one's origin and destination. Refer to AWY segment for more information about the RTE entry.

*** RWY ***

When a runway entry has been received, only the name and DATNUM value are saved in DECNAM and DECNUM respectively. The page routines (ATC, LK-UP) perform the data base search for themselves in this instance.

*** GS ***

Ground speed is an alpha-numeric entry that is converted and stored in DECVAL for use by either ATCMOD or FLTMOD.

*** SID ***

A valid SID entry is the name of a data base defined group of waypoints forming a departure route from an airport. The search for a SID in the database is confined to one airport. If the SID request was made from the ATC page the origin airport is used, while the destination airport is used when on the Look-up page. The origin and destination airports are those defined by pilot entry on the INIT page of the NCDU. DECODE outputs are DECTYP set to 9 and DECADR containing the SID address.

*** STAR ***

A STAR consists of a set of waypoints converging on a particular airport and a runway number designating the runway at that airport to be used for a landing. The STAR name in NCMESG is found in the data base for the destination airport chosen on the INIT page of the NCDU. The outputs associated with this entry are DECTYP set to 9 and the address of the STAR in DECADR.

*** PTA ***

The PTA entry is an ASCII string containing a time to be formatted into the floating point number of seconds it represents. The general form is "HH.MM.SS" for hours minutes and seconds, however since blanks are considered zeroes, short hand entries may be made. The entry "09" would be considered exactly nine o'clock and the number 32,400 would be stored in DECVAL.

*** ARPT ***

The ARPT entry is made when the pilot wishes to look-up all STAR names associated with a particular airport. This is done by pressing the PTA button while the NCDU is on the look-up page #2. The same outputs are created for this entry as are when an airport is used for a waypoint (see WPT segment).
MODULE NAME: FLTMOD

PURPOSE: Format the display for the flight plan on the NCDU

TASK: SLOW

LANGUAGE: MACRO-11

CALLED BY: NCDUEX

CALLING SEQUENCE: CALL FLTMOD

CALLS TO: ATCXO, C7T08, FORMAT, FRMTIM

DESCRIPTION:

FLTMOD formats the Flight Plan page display buffer to show up to three waypoints of the current provisional or active flight plan contained in the guidance buffer (PGBUF). The information shown for each waypoint depends on which of two flight plan pages is being displayed. Flight Plan page #1 is displayed when the "FLT" button of the NCDU keyboard is pressed and the NCDU is not currently on FLT page #1. If the NCDU is on FLT page #1, a "FLT" key stroke sends the NCDU display to FLT page #2. Both pages have a banner line at the top of the display. Text to identify the page number is on the right hand side of the banner line while the origin and destination airport names, if selected, are shown on the left.

Flight Plan page #1 displays six items, contained on two lines, for each waypoint on the display. The first of the two lines contains the waypoint name, altitude and planned time of arrival while the second line contains a message, flight path angle and ground speed. A field of blanks may be present if one of the five parameters associated with the waypoint name is null.

Flight Plan page #2 also contains six items on two lines for each waypoint. The waypoint name, remaining distance to final waypoint (nautical miles), and the estimated elapsed time to the final waypoint are on the first of the two lines. The next line has the message, distance to the next waypoint, and the estimated elapsed time to the next waypoint.
Either page may have one of the following three messages in the message field directly under the waypoint name.

- **TURN L**  Left DMA turn starts at this waypoint
- **TURN R**  Right DMA turn starts at this waypoint
- **HOLD**    Engage holding pattern at this waypoint

The waypoint names may have a one character prefix inserted by the Flight Plan page. A question mark means the waypoint is considered provisional while a star means the waypoint is the current destination waypoint on the active path.

When the Flight Plan page is initially activated, the center waypoint on the display will be the "to" waypoint, marked by the star, if all the waypoints on the path are active. Otherwise the center waypoint will be the most recent provisional waypoint entered. As a path is flown and the "to" waypoint updates to each successive waypoint, the Flight Plan Page display will automatically update to show the new "to" waypoint in the center position. If the pilot wishes to review more of the flight plan than the 3 waypoints shown on the display, he may press the "UP" or "DOWN" keys on the NCDU keyboard to scroll the display to other waypoints. Once either the up or down key has been used to alter the default display, "to" waypoint changes will not affect the display any more. An exit and re-entry of the Flight Plan page must be performed to enable the automatic scrolling again.

There are seven NCDU functions that may be used to affect the flight plan while on either of the Flight Plan pages. All but the execute and reject functions can be used on a provisional or active path. See NCDU function table in the module documentation for DECODE for the description of the RAD, F/L, ALT, GS, and PTA. See ATCMOD documentation for a description of the EXEC and REJ functions. Note that when an alteration is made to an active path, appropriate guidance modes are dropped and must be re-selected, after the path definition stage is complete, to engage automatic navigation of the path.

**GLOBAL INPUTS:** ACTPOS, ARPLIN, ATCLNB, ATCLUP, ATCREJ, BADRAD, DECNUM, DECVAL, FPMODE, HLDSEL, HLDWPT, MDPAG, NGIDON, PROV, PTR2DN, PWEND, WPPTR

**GLOBAL OUTPUTS:** CASKAL, CASKFL, CASKGS, FPPWP, KEYUNL, GDTIME, LATCN, LONCN, NCDERR, NEWPTA, NPAGE, NUPAGE, PRVDEF, PRVEXC, PTAPOS, PTATIM, SET4D
MODULE NAME: INIMOD

PURPOSE: To process INIT page data entered via the NCDU and format the INIT page for the NCDU.

LANGUAGE: Macro-11

CALLED BY: NCDUEX

CALLING SEQUENCE: CALL INIMOD

CALLS TO: ASCBIN,BCDTIM,C7TO8,FRMTIM,LUARP,PROVOG,INSELH,TIMBIN

DESCRIPTION:

INIMOD processes INIT page data entered via the NCDU and displays appropriate cue messages on line 8 of the NCDU. Five parameters may be entered on the INIT page corresponding to the five numeric lines displayed as follows:

1 ORIGIN
2 DESTIN
3 GMT
4 BAROSET
5 IASLIM.

When INIMOD is entered for the first time version and title information are retrieved from a data block that is generated by the system build process. This information is displayed at the top of the INIT page and the message "CHECK GMT PRESS 3" is displayed on line eight of the NCDU. The GMT entry functions as an interlock to ensure that the real-time clock in the flight management software is started. No other page can be selected until after a GMT entry has been made. Once the GMT has been entered, the message "PRESS # FOR DATA ENTRY" is displayed on line eight. The actual entry of Greenwich Mean Time (GMT) can be by default, i.e., pressing 3 then the ENT key which causes the system time, supplied by the Data Acquisition System (DAS), to be displayed on the NCDU. To override the default system time, the time may be entered in hours, minutes and seconds in the following format - HH,MM,SS.

As each parameter is entered its validity is checked and an appropriate error message displayed for invalid entries. The following error messages may appear for the respective parameter:

1 - "NOT IN MEMORY PRESS #" - this message indicates that the selected origin airport could not be found in bulk data. To reenter the origin press 1.

2 - "NOT IN MEMORY PRESS #" - this message indicates that the selected destination airport could not be found in bulk data. To reenter the destination press 2.

3 - "FORMAT ERROR PRESS #" - the entered time was incorrectly formatted. The correct format is HH,MM,SS. To reenter the time press 3.
4 - "FORMAT ERROR PRESS #" - the entered barometric pressure setting was incorrectly formatted and/or differed from the default setting of 29.92 by more than |1.5|. To reenter the Baroset press 4.

5 - "FORMAT ERROR PRESS #" - the entered speed was incorrectly formatted or the entered value was less than 105 knots. To reenter IASLIM press 5.

In addition to the error messages and INIT page cue messages previously entered, other informative messages may appear on NCDU line eight announcing system status and/or directives.

GLOBAL INPUTS: BINTME, BSETO, B4CKZ2, DECNUM, DESTIN, FRMVAL, INMODE, KEYUNL, LABEL, LATINS, LONINS, NAVFLG, NCDSPC, NCMESG, NPAGE, ORIGIN, TMEFLG

GLOBAL OUTPUTS: ATNAV2, ATNAV3, RARALR, BARSET, BINTME, CENWPT, DECNUM, DESTIN, FUNUMB, IASREF, IDDLAT, IDDLON, INMODE, KEYUNL, LLINIT, LOKBUF, LOKCEN, LOKPGN, LOKWPT, MODUPD, NCDERR, NCDSPC, NEWDES, NVAD2A, NVAD2B, NVAD3A, NUPAGE, ORIGIN, PATHND, SMER61, SMER62, SMER63, SMER65, SMER66, TMEFLG
MODULE NAME: INSELH

PURPOSE: Initialization of select page #1 options.

TASK: SLOW

LANGUAGE: MACRO-11

 CALLED BY: SELMOD, ATCMOD, INIMOD

 CALLING SEQUENCE: CALL INSELH

 CALLS TO: None

DESCRIPTION:

INSELH selectively resets memory areas pertaining to Select page #1 options. Each of the five options numbered 1 through 6 (option #4 is undefined) has text data on the page and variables that need to be reset at certain instances. The flags SMER61 - SMER66 are tested for non-zero values to initiate the reset of the corresponding option.

The three areas of memory (SE11, SE21, SE31) used to store the text data used for the Select page display are contained within this module.

GLOBAL INPUTS: SMER61, SMER62, SMER63, SMER65, SMER66

GLOBAL OUTPUTS: CRAD, HLDSEL, HLDWPT, HOLDIN, OFBIAS, OFSSEL, RADBRG, RADWPT, RWPTAD, SE11, SMF61

170
MODULE NAME: LOKMOD

PURPOSE: Display status and allow pilot look-up of data base items.

TASK: SLOW

LANGUAGE: MACRO-11

CALLED BY: NCDUEX

CALLING SEQUENCE: CALL LOKMOD

CALLS TO: C7T08, FORMAT, FORMLL, FRMFRQ, LURWY

DESCRIPTION:
The look-up pages of the NCDU have two distinct functions that are performed depending on which of the two pages is being displayed. Consecutive presses of the "LOOK" button on the NCDU keyboard toggle the display between pages #1 and #2.

Look-up page #1 allows the examination of a system status report. Two columns of entries are displayed along with either "OK" or "BAD" following each entry. An example display is shown below followed by a description of each entry.

```
STATUS   LOOK UP #1
* ILS ON   OK  DME3 EAD *
* DME2 BAD *
* G/S OK VOR/LOC BAD *
* ALT OK *
* INS OK *
* LOOK-UP STATUS *
```

LOOK-UP PAGE #1

ILS ON  The tuning heads have been set so that the Instrument Landing System (ILS) frequency will be tuned instead of a VHF Omni-range beacon (VOR).

G/S  Glide slope landing signal valid.

DME3  Distance measuring navaid beacon #3 (cross station) valid.

DME2  DME path station valid.

VOR/LOC  VOR signal valid ORed with localizer signal valid. (VOR hardware not presently implemented on aircraft).

INS  Inertial Navigation System operative.
Whenever the status of any of the items is "BAD", the message "LOOK UP STATUS" is displayed on line #8 of the NCDU no matter which NCDU display is active. This alert may be canceled by pressing CLEAR while the message is shown.

Look-up page #2 is used to examine information stored in the system database (rescom BULK.TSK). There are seven types of data base items that can be looked-up, six of which are shown below in the example look-up page #2 menu.

```
* KLFI -> KWAL  LOOK UP#2 *
* WPT           *
* AWY           *
* RTE           *
* SID(DEST)     *
* STAR(DEST)    *
* RWY(DEST)     *
* PRESS FUNCTION KEY *
```

The page #2 menu is the default page #2 display which will be shown until one of the seven items is looked-up. The seventh item, not shown on the menu, is the ARPT function which displays all STAR names at a particular airport. Unlike the other six functions, which have a designated NCDU key, the ARPT look-up is initiated by the PTA button. The module description of DECODE may be consulted for definition of the seven items.

All seven look-ups are performed in the same manner. The function button on the NCDU keyboard is selected. the appropriate prompt is issued on line #8, a name in response to the prompt is entered, and information is displayed instead of the menu. Pressing the reject key causes the menu to re-appear. All of the seven item types except WPT and RWY cause the display of a list of the waypoints associated with the name. If more information is desired the WPT look-up must be performed on each particular waypoint in the list. In many cases the list of waypoints will be too long for the display. The line #8 message "KEY UP PAGE CONTINUES" will appear and the UP/DOWN keys will be enabled for display scrolling.

If the Navigation display unit (ND) is in the "plan" mode, then when a look-up has been completed, the ND will have a pictorial layout of all waypoints that fall in its current screen boundaries. The ND display may be centered at one of these waypoints by scrolling the NCDU display until the desired waypoint is moved within the question mark field and pressing the EXEC button on the NCDU keyboard. The question marks will change to angle brackets and the Nav display center will update.
The following are example NCDU displays of STAR and WPT look-ups:

```
KWAL- KWAL LOOK UP#2
* STAR WTB13
* WFB13 4000 PGS150
* WFT2E 4000 PGS150
* WFB1D 4000 PGS150
* WFB1E 2745 PGS150
* KEY UP PAGE CONTINUES

GLOBAL INPUTS: BLNTB, DECADR, DECLAT, DECLON, DECNAM, DECNAV, DECNUM, DECTYP, DESTIN, LOKPGN, LUADDR, LUMODE, MODPAG, ORIGIN

GLOBAL OUTPUTS: CENWPT, GDTIME, LATCEN, LOKBUF, LOKCEN, LOKWPT, LONCEN, LOKWPT, LONCEN, NCDERR, NPAGE, NUPAGE
```

ORIGINAL PAGE IS OF POOR QUALITY
MODULE NAME: NCDGUD

PURPOSE: Interface the NCDU and the various path definition functions

TASK: SLOW

LANGUAGE: HAL/S

CALLED BY: NCDUEX

CALLING SEQUENCE: CALL NCDGUD

CALLS TO: PATHDF, PRVSIZ, SRPTA, BUFSWT

DESCRIPTION:
NCDGUD calls path definition routines to perform functions requested by NCDU modules ATCMOD and FLTMOD when changes have been made to the flight path in the Active or Provisional guidance buffers. One of the following three variables will be set to a non-zero value to initiate a call to NCDGUD.

PRVDEF Define static path features for waypoints in the provisional guidance buffer only.

PRVEXC Same as PRVDEF plus activating the provisional path.

SET4D Cascade planned times of arrival for each waypoint in the active guidance buffer when a PTA has been entered on ATC or Flight Plan page.

The guidance possible flags (GUID2D, GUID3D, GUID4D) are cleared for a minimum of 50 milliseconds to cause the appropriate loss of guidance mode when PRVEXC is used. Only GUID4D is cleared in response to the setting of SET4D. This causes the pilot to reselect guidance modes after a change was made that invalidated previously selected modes.

After any of the three operations have completed, the navigation interface done flag (NGIDON) is set to cause NCDU page software to continue from the NCDU busy mode entered upon initiating action from NCDGUD.

GLOBAL INPUTS: PRVDEF, PRVEXC, FIFTY, SET4D, PDG4D, PGBUF, PNUM

GLOBAL OUTPUTS: NGIDON, GUID2D, GUID3D, GUID4D, PTAXFG, STRACT, STRPRV, ACTIVE, PTAFLG
MODULE NAME: NCDKEY

PURPOSE: Perform NCDU fast loop functions such as the interpretation of NCDU inputs, posting alert messages, and synchronizing the NAV display updates with guidance buffer changes.

TASK: FAST

LANGUAGE: HAL/S

CALLED BY: FAST

CALLING SEQUENCE: CALL NCDKEY

CALLS TO: CLREF, C7TO8, NCDUIO, PAGRST, RESETM, RESET2, SETEF

DESCRIPTION:
NCDKEY is called every 50 milliseconds to perform miscellaneous NCDU functions. The following discussion handles NCDKEY in the sequential order found on the program listing.

First a test is made to determine if a change has been made to the guidance buffers, a new symbol for the navigation display has been requested by NCDU software, or 4 seconds have elapsed since the last guidance buffer transfer (see FLTHDL task description for information on the DMA transfer between the two NORDEN computers). The 4 second update is accomplished by comparing the difference between the current time of day (BINTME) and the time of the last update (GDTIME) to 4 seconds. The other two forms of update requests are achieved when a NCDU routine sets the variable GDTIME to zero. This causes the appearance of 4 seconds expiring and an update sequence is started. The boolean MFDREQ and RSX-11S event flag 34 are set to inform the NORDEN #1 display computer and the I/O handler of the update. MFDREQ is "ON" for two 50 millisecond frames, while event flag 34 is on for one frame.

When NCDU routines have detected an error on user entry, a non-zero index value will be stored in NCDERR. A call to the local subroutine KEYERR is made to post the appropriate error message on the bottom line of the NCDU when the non-zero NCDERR is found.

NCDU variables that need to be computed every 50 milliseconds are done at this point with a call to NCDK3. The following is a summary of the values computed by NCDK3.

PTR2DN The byte offset into the guidance buffer of the "to" waypoint.

WPTALR Alert boolean that is set for the final 10 seconds of a waypoint approach. The boolean causes the ALERT light on the NCDU to be lit.

ALRMES The index value corresponding to an alert message that will be posted on NCDU display line #8. Various logic is performed for the six possible alert messages, including the check of message inhibit bits.

LATCEN/LONCEN The position values for NAV Display center. These values are set corresponding to what functions have been performed on the NCDU.
The address of the next navaid to be used as VOR #2.

Flag set to cause the Inertial Navigation System (INS) to read runway heading.

NCDU I/O interfacing is done by a call to NCDUIO (see module documentation). When a function entry has been made on the NCDU, the variable NCDUFN is set to a non-zero function code by NCDUIO. The cold start flag is checked and NCDU variables are reset if the condition is true. The NCDU is forced to the INIT page until a new time has been entered since the computer time will be incorrect after recovering from power off. NCDKEY tests NCDUFN and returns when no function entry was made. When a non-numeric function entry was made and a valid time entry was made on the INIT page, the subroutine FUNCTION CODE is called to process the entry. On numeric function entry (NCDU buttons 1 - 9) a test is made to see if numeric options are enabled on the current page. If enabled, the input function is saved in NCDSPC for use by the active page and in DUALNM to wait for completion of a NCDU data entry. When not enabled the INVALID FUNCTION message is displayed. The following is a list of the different classes of function entries.

CLASS #1 - Functions causing a NCDU bottom line prompt.

Function Keys: WPT, AWY, RTE, RWY, SID, STAR, RAD, FL, GS, ALT, PTA

NCDKEY checks if the active NCDU page allows the function key. If so, the prompt is issued and a unique function code is placed in DUALNM to be saved until the user entry is complete. The function code is stored in DATNUM to enable the input parsing routine DECODE after the input is finished.

CLASS #2 - Page Selection Functions.

Function Keys: INIT, FLT, SEL, ATC, NAV, LOOK

NCDU variables are reset and the page corresponding to the function becomes active. Subsequent presses of the same function cause the various sub-pages of the active page to be displayed.

CLASS #3 - Numeric Functions.

Function Keys: 0 - 9

Numeric functions have specific page dependent meanings. The function number is stored in DUALNM like CLASS #1 functions and also in NCDSPC to notify the active page of the function entry. The page routine has the responsibility of issuing the NCDU prompt.

CLASS #4 - Miscellaneous Functions.

Function Keys: EXEC, UP, DOWN, CLR, REJ

These functions cause no user prompts and are passed along to NCDU background routines through the variable DECNUM. Each cause
the cancelation of NCDU key entry mode. The clear function also is used to inhibit the display of alert messages.

CLASS #5 - NCDU mode dial

Mode Positions: Automatic, Manual, Test

Determines the mode of the NCDU display.
Automatic: standard NCDU page.
Manual: Debug mode. (When in Debug mode, 0 key can be used as Debug page Select key.)
Test: NCDU test pattern display.

GLOBAL INPUTS: ACTIVE, ACTPOS, ALTG, ALTGSF, ATCMWP, ATNAV2, BADR, BFLAG, BINTME, COLST, DESTIN, DTG, DTOG, FNUMB, GDTIME, GS, GSFPS, NAV64K, NCDERR, GUID2D, NCDUFN, LAT, LOKCEN, LON, PGBUF, PROV, PTR2D, PTR4D, RETUN2, RWYCNT, STATUS, TMEFLG, TURN, WPSIZ

GLOBAL OUTPUTS: ACMODE, ALRMES, ATCMWP, AUMODE, BARALR, DECNUM, DUALNM, MAMODE, FPMODE, MFDREQ, MODPAG, INMODE, NCDSPC, KEYUNL, LATCN, NDMODE, LATCN, NPAGE, NUPAGE, NVAD2B, LONCEN, LONCN, LUMODE, PRVMP, PTAREJ, PTR2DN, RETUN2, RWYOUT, TSMODE, SEMODE, WPTALR
MODULE NAME: NCDUEX

PURPOSE: Executive for NCDU routines.

TASK: SLOW

LANGUAGE: MACRO-11

CALLED BY: SLOW

CALLING SEQUENCE: CALL NCDUEX

CALLS TO: MAP, TSTMOD, DEBUG, NCDGUD, DECODE, ATCMOD, FLTMOD, INIMOD, LOKMOD, PRVMAP, NVDMOD, POSERT, SELMOD

DESCRIPTION:

NCDUEX calls the various page formatting and path definition interface routines. Also the valids associated with Look-up page #1 are computed and reflected in the variable STATUS.

The first thing done in NCDUEX is to compute the boolean variable VORLOC, which is the logical OR of localizer and VOR valids. The logical NOT of this and six other valids (LOCFS, DME3VD, DME2VD, GSVLD, CALTV, INAVV) are ORed to form a failure alert boolean (STATUS) which is used by module NCDKEY to post the message "LOOK-UP STATUS" on line #8 of the NCDU. When this message appears the NCDU should be changed to Look-up page #1 to note which signals are invalid.

Next, conditions are checked to determine which NCDU subroutines need to be called. The T/M/A knob (Test/Manual/Automatic) at the upper left of the NCDU keyboard controls the NCDU mode. In test mode a moving pattern of the NCDU character set is displayed and nothing can be entered from the keyboard. Manual mode provides the capability of displaying and altering compool memory cells (see documentation for module DEBUG). Almost all NCDU activity is done while in automatic mode. One of the six NCDU pages will always be displayed, depending on the page buttons located on the bottom row of the keyboard, while in automatic mode. The various pages are called Air Traffic Clearance, Flight Plan, Initialization, Look-up, Navigation Data and Select. Information on these can be found in the documentation for modules ATCMOD, FLTMOD, INIMOD, LOKMOD, NVDMOD and SELMOD respectively. Two other subroutines involved with guidance interfacing are called if conditions are met. NCDGUD is called when changes to waypoints in one of the guidance buffers have been made and PRVMAP is called when a missed approach path (MAP) is to be added at the end of the path.

All subroutines called from NCDUEX are preceded by a call to MAP to determine if the correct overlay segment is mapped. Task documentation for SLOW and the RSX-11S Executive Reference Manual can be consulted for information on overlays.

GLOBAL INPUTS: VORVLD, LOCFS, LOCVLD, TSMODE, MAMODE, PRVDEF, PRVEXC, SET4D, DATNUM, ACMODE, FPMODE, INMODE, LUMODE, NDMODE, PSRTFG, GUID2D, ACTIVE, PTR2DN, PRVMP, AUMODE, BINTME, CALTV, DME2VD, DME3VD, GSVLD, INAVV

GLOBAL OUTPUTS: BLNTB, VORLOC, STATUS, SEG, NPAGE, NUPAGE, KEYUNL, GDTIME
MODULE NAME: NCDUIO

PURPOSE: Prepares NCDU I/O memory buffers for transmission.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: NCDKEY

CALLING SEQUENCE: CALL NCDUIO

CALLS TO: None

DESCRIPTION:
NCDUIO manipulates NCDU input and output data areas and controls the timing sequence of I/O operations.

A non-zero first element of the 12 element array NCDUL8 means a function or data entry has been made. Function entries are single key strokes of the keyboard that request a particular action by the NCDU. Data entries are character strings submitted in response to a previous function entry. Bit 6 of NCDUL8 element 1 is set by the hardware when the input is a function. Upon receiving an entry, the function code is moved from NCDUL8 into NCDUFN or the text data is moved to NCMESG. When a function that requires data entry is used, its code is stored in DUALNM. Upon completing a data entry, NCDUIO moves the code stored in DUALNM into either DATNUM or DECNUM depending on its value. This causes the function and its corresponding text string to be used as a pair in the currently active NCDU page routine. DECNUM is used for numeric functions (option 1, 2, ...) and DATNUM for all others.

The next section handles NCDU outputs. An output is initiated when the flag NUPAGE is set and the previous output cycle has completed (PAGSEN = 0). NUPAGE is set by the currently active NCDU page routine when a change to the display is needed, however NCDUIO can also set NUPAGE if the one second update mode is not active (MODUPD = 0), no NCDU I/O has occurred for 2 seconds, and the NCDU is not currently waiting for a text data entry in response to a function prompt. When in the one second update mode, NUPAGE is set automatically every second by the currently active page routine. The flag KEYUNL is tested next to determine if the NCDU needs to be placed in data entry mode. This is done by setting bit 14 of the next to last element of the array NPAGE, when KEYUNL is non-zero. Upon receiving the transmission with bit 14 set, the NCDU hardware interprets key strokes as ASCII text until the "ENTER" key is pressed. If the NCDU is not to be placed in input mode, NCDUIO determines if an alert message needs to be put out on line #8 of the NCDU display. A non-zero value in ALRMSG, set by NCDKEY, causes one of the following seven messages to be displayed depending on the value of ALRMSG.

"LINK UP PPOS TO WPT"
"LOOK-UP STATUS"
"RETUNE NAVAID #2"
"EXEC OR REJ"
"PATH INCOMPLETE"
"CHECK BAROSET"
"SIMULATED A/P ENGAGED"
The last message is displayed once every four output cycles if the boolean SIMFLG is true. Finally the NCDU output cycle is started by setting the boolean variables PAGSEN and NCDUNP. NCDUNP is an IOPOOL variable that is packed into a discrete word by OUTIO and sent to the Effecter Interface Unit (EIU). The cycling of the discrete (ON, OFF) causes the hardware to transmit the SIR memory buffer area designated for the NCDU. PAGSEN is used to hold any new attempts to initiate an output before the current output cycle is done. The output cycle consists of ten 50 millisecond frames. NCDUNP will be on for the first two frames and set off for the remaining eight. PAGSEN is on for all ten frames to keep another cycle from starting. After the tenth frame PAGSEN is cleared so the next output request will be accepted.

GLOBAL INPUTS: ALRMES, ATC2, DUALNM, KEYUNL, MODUPD, NCDUL8, NUPAGE, SIMFLG

GLOBAL OUTPUTS: Datnum, Decnum, NCDUFN, NCDUNP, NCMESS, NPAGE, PAGSEN, PTAREJ
MODULE NAME: NVDMOD

PURPOSE: Display dynamic navigation information pertaining to the current flight plan

TASK: SLOW

LANGUAGE: MACRO-11

CALLED BY: NCDUEX

CALLING SEQUENCE: CALL NVDMOD

CALLS TO: C7TO8, FORMAT, FORMLL, FRMFRQ, FRMTIM, LLBIN, LUNAVA, SCOSD

DESCRIPTION:
Each of three Navigation Data pages are displayed in order by consecutive key strokes of the "NAV" button on the bottom row of the NCDU. The first two pages are for information display only, allowing no user entries. Page #3 however displays information subject to user modification.

Nav Data page #1 displays 13 values along with header and message text. The header is shown on the top line, right justified. The bottom line of the display contains one of several possible messages. It may be a general NCDU alert message or a special Nav Data message (see documentation for the module NCDUIO for alert messages). The Nav Data message contains three fields which are used to display the name of the last waypoint reached, the next waypoint to be reached and the three letter code for the current navigation mode (see HNAVSL documentation for information about navigation modes). The word "HOLD" may replace either the "from" or "to" waypoint name if a holding pattern has been selected at that waypoint. The following are the 13 items displayed:

| GMT   | Current time of day. |
| ETA   | Estimated time of arrival at the next waypoint. |
| TE    | Time error when flying time guidance. |
| TTC   | Time to capture. (How long to capture time path position). |
| TK    | Track angle in degrees (magnetic north). |
| TKE   | Track angle error in degrees left or right of the path. |
| XTKE  | Cross track error. Displacement off the path (nautical miles). |
| ALT   | Altitude in feet. |
| AE    | Altitude error in feet when flying 3D path. |
| FPVE  | Flight path angle error in degrees. |
| OVTK  | Over take speed in capturing the time path position. |
| GS    | Ground speed in knots. |
| GSE   | Ground speed error (knots) from the time path ground speed. |
Nav Data page #2 has a banner and message line like page #1. It displays the value of the following 10 variables.

- **HDG**: True and magnetic heading.
- **DA**: Drift angle in degrees left or right.
- **FPA**: Flight path angle in degrees above or below horizontal.
- **GRAD**: Gradient. Altitude change (feet) in 1 nautical mile at the present FPA.
- **WD**: Wind direction (magnetic).
- **WS**: Wind speed in knots.
- **ACC**: Acceleration. Change in speed (knots/sec).
- **TTG**: Time to go to the next waypoint.
- **DTG**: Distance to go to the next waypoint (nautical miles).

Nav Data page #3 shows the banner line and navigation mode like pages one and two. The left of the bottom line, if not containing an alert message, will contain one of the following user prompts:

- **PRESS #**
- **KEY LAT**
- **KEY LON**
- **KEY VOR**

The first prompt notifies the user that the buttons 1 - 5 on the NCDU may be pressed to select new values for the items displayed on the lines. Once a numeric key has been pressed, one of the other prompts is given to note that...
the NCDU is ready for input. The aircraft position may be initialized manually by using keys 1 and 2 to alter the latitude and longitude. Since the degree symbol, appearing in the displayed LAT/LON, does not have a corresponding key on the NCDU keyboard, entered LAT/LON values have no delimiter between the degrees and minutes. Otherwise position entries appear just as displayed on the screen. The system automatically attempts to tune navigation radio aids (see HNAVSL documentation), however manual overrides can be made on Nav Data page #3. NCDU key 3, 4 and 5 are used to initiate manual tuning of nav aids. Once the "KEY VOR" prompt has been displayed, the name of a data base defined navaid waypoint is entered. Following a manual tune, the letter "M" will appear on the line of the manually tuned navaid. Whether manually or automatically tuned, the lines containing nav aids will also have displayed the frequency of that navaid.

GLOBAL INPUTS: ALTCOR, ATNAV2, ATNAV3, BINTME, DECNUM, DFPA3D, DFTANG, DTOGO, FRMVAL, GAMMA, GS, GSFPS, GUID2D, GUID3D, GUID4D, HDGTRU, HLDSEL, HLDWPT, KEYUNL, LAT, LON, MAGVAR, MODPAG, NAVFLG, NAVTP, NAV64K, NCDSPEC, NCMESG, NMODE, NPAGE, NVAD2A, NVAD2B, NVAD3A, OFFSEL, PATHND, PTH4DN, PTR2DN, PTR4D, SDCC, SEPR, TK, TKE, VGSDOT, WD, WS, XTK, ACTIVE, A0, A1, A2, A3, A4, A5, A6, BLANKS, FTONM, HER, IDDLAT, IDDLON, KTOFPS, WPSIZ

GLOBAL OUTPUTS: ATNAV2, ATNAV3, FUNUMB, IDDLAT, IDDLON, LLINIT, MODUPD, NCDERR, NUPAGE, NVAD2A, NVAD2B, NVAD3A
MODULE NAME: PAGRST/RESTM/RESET2

PURPOSE: Initialize NCDU variables after some NCDU function entries.

TASK: FAST

LANGUAGE: MACRO-11

CALLED BY: NCDKEY

CALLING SEQUENCE: CALL PAGRST
                CALL RESTM
                CALL RESET2

CALLS TO: none

DESCRIPTION:
These routines are separate entry points into one subroutine that exists on the file RESET.MAC. They are all used to initialize some NCDU variables when the NCDU software has been in some transitional state. RESET2 is performed no matter which of the three entry points is used.

PAGRST is used when the NCDU mode is changed from either TEST or MANUAL to AUTOMATIC (the standard NCDU page display). It finds whichever page was last active and sets the first pass bit for that page to enable specialized initialization. RESTM resets NCDU variables after a display page change and RESET2 resets other variables when a mode change from AUTOMATIC to either TEST or MANUAL is made.

GLOBAL INPUTS: ACMODE, FPMODE, INMODE, LUMODE, NDMODE, SEMODE

GLOBAL OUTPUTS: ACMODE, FPMODE, INMODE, LUMODE, NDMODE, SEMODE, DATNUM,
                 DECNUM, NCDSPC, FUNUMB, KEYUNL, MODUPD, PTAREJ
MODULE NAME: PRVSIZ

PURPOSE: Set provisional guidance buffer end pointers and optionally set/clear provisional bit for each waypoint.

TASK: SLOW

LANGUAGE: MACRO-11

CALLED BY: NCDGUD

CALLING SEQUENCE: CALL PRVSIZ(DO_BIT,POLARITY);

C... DO_BIT Boolean for enabling the changing of the
C... provisional bit.
C... POLARITY The boolean value that the provisional bit
C... will be set to.

CALLS TO: none

DESCRIPTION:
PRVSIZ is called, after changes have been made to the provisional guidance buffer, to set the global variables PWNUM and PWEND. PWNUM is the number of waypoints in the buffer and PWEND is the byte offset from the start of the buffer to the location of the first empty waypoint.

If the first of the two input parameters has the value 'ON', then the provisional bit for each waypoint is set to the boolean value contained in the second input parameter.

GLOBAL INPUTS: PGBUF, WPSIZ

GLOBAL OUTPUTS: PWNUM, PWEND
MODULE NAME: SELMOD

PURPOSE: Perform NCDU 'Select Page' I/O

TASK: SLOW

LANGUAGE: MACRO 11

CALLED BY: NCDUEX

CALLING SEQUENCE: CALL SELMOD

CALLS TO: INSELH, LUGUID, ATN2D, ASCBIN, FORMAT, LUGRP, LUNAVA

DESCRIPTION:
The select pages of the NCDU allow the AFD pilot to select desired horizontal navigation, vertical guidance and power setting options. There are three separate pages that can be displayed by consecutive presses of the SEL button on the NCDU.

***** PAGE #1 *****

The first select page has six pilot options, however option #4 is currently undefined. An option is chosen by pressing the numeric button on the NCDU corresponding to the number displayed preceding each option. Any option in effect can be cancelled by pressing its function number followed by a REJECT.

```
*                                *
*       SEL#1       *
*  1 WPT HOLD WFBBF/L/090  *
*  2 OFFSET   L/ 4   *
*  3 CIRCLE   WFBBD/30  *
*  4   *
*  5 RADIAL   WFBBC/090  *
*  6 'TO' WPT WFBBE   *
* FRESS # FOR MODE SELECT *
*                                *
```

(NCDU SELECT PAGE #1)

Option #1 causes a holding pattern to be displayed on the navigation display unit. The pattern consists of two parallel lines appropriately oriented with respect to the designated waypoint on the active flight path. One line is joined to the waypoint while the other is optionally to the left or right. The lines are oriented on a magnetic bearing chosen by the pilot. To enable a holding pattern function #1 is pressed and the prompt "KEY WPT" is displayed on NCDU line eight. After entering a waypoint name or POS, for present position, "L-R&BGR" is displayed on line eight. Upon entering position and bearing the option to "EXEC OR REJ" is given.
Option #2 causes an offset flight path to be drawn on the navigation display unit. The original flight path is drawn dashed as if it were provisional and the offset path is drawn as a series of solid lines and arcs at a fixed distance from the flight path.

To choose an offset flight path, press button #2 on the NCDU. Enter L or R (for left or right) and offset in nautical miles in response to the prompt "L-R&RFS". The prompt "EXEC OR REJ" is issued to allow a final choice of displaying the offset path.

Option #3 causes a circle to be drawn about any waypoint stored in the NAV resident common data base (BULK.TSK). Press button #3 and enter a waypoint name and circle radius, in nautical miles, in response to the prompt "WPT&RAD". The waypoint and circle will immediately be displayed if it lies within the screen boundaries of the display.

Option #5 is used to select the display of a radial symbol through any waypoint in the data base. A series of small circles passes through the chosen waypoint in a line oriented to a chosen magnetic bearing.

Select the radial option by pressing button #5 on the NCDU. Enter waypoint name and bearing in response to the "WPT&BRG" prompt. The waypoint and symbol will appear immediately on the navigation display if its position lies within the screen boundaries.

Option #6 allows the selection of the waypoint on the path that the airplane will fly to. By default the 'TO' waypoint is initially the second on the path. The altering of the 'TO' waypoint can be done only when the A/P is not moving, therefore this option is only useful when flying EASILY lab simulations or departing from an airport.

Press button #6 to get the prompt "KEY WPT" and enter the name of a waypoint on the path. "EXEC OR REJ" is displayed next for finalizing the decision.

***** PAGE #2 *****

The second select page has three vertical guidance options which are presently unused. These were used in previous versions of the displays software but at the present time these options have no effect on the primary flight display.

Q*******************************
* EADI OPTIONS    SEL#2 *
* 1 VNAV#1         *
* 2 VNAV#2         *
* 3 VNAV#3         *
*                *
*                *
* PRESS # FOR MODE SELECT *
*******************************

(NCDU SELECT PAGE #2)
The third select page allows the selection of power settings. The current state of AIRBLEED, engine pressure limits and pressure ratios of both engines are displayed on this page. The status of AIRBLEED can be changed and a desired EPR limit chosen by pressing the numeric NCDU button corresponding to the numbers displayed preceding the options. See module documentation of EPRLMT for the description of maximum continuous thrust, climb and cruise engine pressure ratio limits.

```
* EPR STATUS  SEL#3 *
* 1 AIRBLEED  OFF  *
* 2 MCT  1.94  SELECTED *
* 3 CLIMB  1.94  *
* 4 CRUISE  1.85  *
*  *
* EPR#1 0.00  EPR#2 0.00  *
* CHECK AIRBLEED STATUS  *
```

(NCDU SELECT PAGE #3)

GLOBAL INPUTS: ACTIVE, ATC2, BINTME, CLAT, DATNUM, DECADR, DECBNG, DECMV, DECNAM, DECNUM, DECTYP, DSRTK, DUALNM, EPR1, EPR2, FNMVAL, FUNUMB, GUID2D, GUID3D, LUADDR, MCLEPR, MCREPR, MCTEPR, MODPAG, MODUPD, NCDSPC, NCMESG, NGIDON, SEMODE, SE11, SE21, SE31, WPEND, WPSIZ

GLOBAL OUTPUTS: ABRSTAT, CRAD, DECTY, EPRFLG, GDTIME, HLDBRG, HLDSEL, HLDWPT, HOLDIN, KEYUNL, NCDERR, NPAG, NUPAGE, OFBIAS, OFSSEL, PSRTFG, RADBRG, RADWPT, RWPTAD, RWPTNM, SMER61, SMER62, SMER63, SMER65, SMER66, SMF61, VNAV, WPPT
MODULE NAME: TSTMOD

PURPOSE: Display test pattern on the NCDU

TASK: SLOW

LANGUAGE: MACRO-11

CALLED BY: NCDUEX

CALLING SEQUENCE: CALL TSTMOD

CALLS TO: None

DESCRIPTION:
The test pattern on the NCDU is displayed when the T-M-A (Test-Manual-Automatic) knob on the NCDU is turned to "T". A repeated pattern of all 64 NCDU characters is then displayed on the screen. Every second the display is shifted 1 character to the left giving the appearance of a moving string of characters that wrap around at the right and left screen boundaries.

***
BCDEFGHIJKLMNOPQRSTUVWXYZ *
Z[\]^_!"#$%&'()++,-./01 *
23456789;<=?>@ABCDEFGHIJKLMNOPQRSTUVWXYZ[^_!
JKLmnopqrstuvwxyz[^_!
!*+,-,/0123456789;<=?>@ABCDEFGHIJKLMNOPQRSTUVWXYZ[^_!
RKLVWXYZ[^_!"#$%&'()++,-./0123456789;<=?>@ABCDEFGHIJKLMNOPQRSTUVWXYZ[^_!

(Test Pattern)

GLOBAL INPUTS: BINTME, MODUPD, TSMODE

GLOBAL OUTPUTS: DECNUM, NCDSPC, NPAG, NUPAG
MODULE NAME: WPINVT

PURPOSE: Create a new waypoint.

TASK: SLOW

LANGUAGE: MACRO-11

CALLED BY: ATCMOD, DECODE

CALLING SEQUENCE: MOV #N,RO (N - number)
                  CALL WPINVT

CALLS TO: None

DESCRIPTION:
When the pilot has entered commands to the NCDU that call for the
creation of a new waypoint, WPINVT defines a waypoint with the name "PPTn"
where "n" is a number from 1 to 9. The latitude, longitude and address of a
navaid to be associated with the waypoint are set up prior to calling WPINVT
in the variables DECLAT, DECLON and DECNAV. A code number, passed in
through register #0, used to differentiate between various types of pilot
defined waypoints is stored in the variable DECTYP. The following items for
each created waypoint are stored in the pilot defined waypoint buffer (WAYPNT)
which can contain up to nine waypoints before wrapping back around to the
start.

<table>
<thead>
<tr>
<th>NAME</th>
<th>The name of the waypoint (created by WPINVT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LATITUDE</td>
<td>The latitude of the waypoint (from DECLAT)</td>
</tr>
<tr>
<td>LONGITUDE</td>
<td>The longitude of the waypoint (from DECLON)</td>
</tr>
<tr>
<td>NAVAID</td>
<td>The address of the navaid to be used for guidance (from DECNAV)</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>The text entered causing the creation (from NCMESG)</td>
</tr>
</tbody>
</table>

GLOBAL INPUTS: DECLAT, DECLON, DECNAV

GLOBAL OUTPUTS: DECADR, DECNAM, DECTYP
7.0 NAVIGATION
NAVIGATION OVERVIEW

The navigation routines find aircraft position, altitude and velocities via various position measuring techniques. Factors external to the aircraft are computed such as wind speed/direction and local earth radius of curvature.

The tuning of ranging stations such as DME (Distance Measuring Equipment) and VOR (VHF Omni-range) is done by HNAVSL to find position estimates relative to known transmitter locations. These values are combined with sensor data such as INS (Inertial Navigation System) or MLS/ILS ground telemetry data by HNAVFS every 50 ms to produce the aircraft position, altitude and velocities.
MODULE NAME: ACCPRC

PURPOSE: To compute alternate values of the along track and crosstrack acceleration terms.

TASK: FAST

LANGUAGE: HAL/S

CALLED BY: FAST

CALLING SEQUENCE: JSR PC_ACCPRC

CALLS TO: ACCSTP, IAND, SCOSD, SQRT

DESCRIPTION:
This module manipulates the INS ATKACC and XTKACC inputs in one of several different ways depending upon the bits set in the MLS configuration word (MCONF). If MCONF$9 is set, these inputs are modified to simulate those inputs that might be expected from a standard LTN-51 INS using an algorithm supplied by Bill Lynn. If MCONF$10 is set, the inputs are modified according to Harry Fuller. If MCONF$11 is set, these inputs are simply replaced by analogous signals derived from the body mounted accelerometers. If none of the above bits are set, the routine simply exits.

GLOBAL INPUTS: ATKACC, BMACC, CTHET, CROLL, DFTANG, MCONF, MLSC, PITCH, ROLL, STHET, SROLL, XTKACC

GLOBAL OUTPUTS: ATKACC, XTKACC
MODULE NAME: BLOW
PURPOSE: Calculate wind speed and direction.
TASK: SLOW
LANGUAGE: HAL/S
CALLED BY: Slow Loop Executive (SLOW)
CALLING SEQUENCE: JSR PC,BLOW
CALLS TO: None

DESCRIPTION:
If the radio altitude HRAD is less than five feet then wind direction
remains unchanged and wind speed is set to zero.
If HRAD is greater than or equal to five feet then wind speed and
direction are computed as:

\[
\begin{align*}
WD &= \text{ATN2D} (X,Y) \\
WS &= \sqrt{X^2 + Y^2}
\end{align*}
\]

WHERE: 
\[
\begin{align*}
X &= \text{SINTH \ TASGS} - \text{VE} \\
Y &= \text{COSTH \ TASGS} - \text{VN} \\
\text{VE} &= \text{velocity east in knots} \\
\text{VN} &= \text{velocity north in knots} \\
\text{SINTH} &= \sin \text{ of true heading} \\
\text{COSTH} &= \cos \text{ of true heading} \\
\text{TASGS} &= \text{true airspeed corrected for gamma}
\end{align*}
\]

GLOBAL INPUTS: COSTH, HRAD, SINTH, TASGS, VE, VN
GLOBAL OUTPUTS: WD, WS
MODULE NAME: ERAD

PURPOSE: Compute the radii of curvature in the east-west and north-south planes.

TASK: SLOW

LANGUAGE: HAL/S

CALLED BY: Slow loop executive (SLOW)

CALLING SEQUENCE: JSR PC,ERAD

CALLS TO: GETDAT

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 INS navigation is valid (INAVV true) then MAGVAR = MVINS; else if a path has been entered (GUID2D true) then MAGVAR is set from the guidance buffers (PWMVS(PTR2D)); else if a valid DME is tuned (FLSTA2 or FLSTA3 = 0) then MAGVAR is set from the bulk data address pointed to by NVAD2A or NVAD3A, respectively, via a call to GETDAT. If none of the above conditions prevail, then MAGVAR is not set.
The local north and east radius of curvature and several related variables used by the navigation and guidance procedures are then set by evaluating the following equations:

\[
\begin{align*}
RN &= RADIUS \times (1. + ELLIP \times SLAT^2) \\
RM &= RADIUS \times (1. - 2. \times ELLIP + 3. \times ELLIP \times SLAT^2) \\
RNP &= (ALTCOR \times FTONM + RN) \times CLAT \\
RMP &= ALTCOR \times FTONM + RM
\end{align*}
\]

where: RADIUS is the nominal earth radius in NM. (taken as 3443.9362 NM)
ELLIP is the eccentricity (3.3901E-3).
SLAT is the sine of the present latitude.
CLAT is the cosine of the present latitude

\[
\begin{align*}
RADFT &= RMP \times NMTFT \\
DLATFT &= RADFT \times DTOR \\
DLONFT &= RNP \times NMTFT
\end{align*}
\]

where: RADFT is the local earth radius in feet,
DLATFT is the number of feet per degree of latitude,
DLONFT is the number of feet per degree of longitude.

GLOBAL INPUTS: ALTCOR, CLAT, ELLIP, FLSTA2, FLSTA3, FLYFLG, FTONM, GUID2D, INAVV, MVINS, NVAD2A, NVAD3A, SLAT

GLOBAL OUTPUTS: MAGVAR, RMP, RNP, RADFT, DLATFT, DLONFT
NAME: HNAVFS (Horizontal/vertical NAVigation, FaSt)

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

TASK: FAST

LANGUAGE: HAL/S

CALLED BY: FAST

CALLING SEQUENCE: CALL HNAVFS

CALLS TO: ANGL, ATN2D, DPINTG, SCOSD.

DESCRIPTION:

The Horizontal/Vertical Navigation routine (HNAVFS) is the only fast loop
navigation routine. All integrations in the navigation filter take place
here. The radio corrections to the INS positions and velocities are received
from HNAVS, and the east-west and north-south local radii of curvature are
received from ERAD. These are combined and integrated to form the corrections
to the velocities (knots) and position (degrees). A simplified overview is
depicted in figure 7-4.

HNAVFS first initializes (OFF) FLRM and INSVAL and, if COLDST is TRUE, it
also initializes NAVFLG, LLINIT and PASS1. The value for TASGS (ground speed
based on true airspeed) is computed as the square root of TASFPS**2 minus
HDOT**2, converted to knots. Subsequent initializations occur as required
after the determination of flight mode.

Air Data or Radio Mode must be engaged if neither FLYFLG nor INAVV is
TRUE. The acceleration inputs, normally input from the INS, are derived from
the body mounted accelerometers by external code and assigned to ATKACC and
XTKACC if bit 13 of MCONF is set. If so, these accelerations will be assigned
to IDDATK and IDDXTK. Otherwise, no acceleration input is available; IDDATK
and IDDXTK are zeroed and NCUVAL is set FALSE. In either case, true heading
is taken as mag heading corrected for local variation, and external procedure
SCOSD is called to form the sine and cosine of true heading (SINTH,COSTH).

Air Data Mode will be selected if the true airspeed input is valid (TASV
= TRUE) and the directional velocities are then calculated as:

EWAVE = TASGS SINTH
NSAVE = TASGS COSTH.

Furthermore, on the first pass for Air Data Mode, DVN and DVE are initialized
as the difference between the directional velocities and the estimated INS
velocities:

DVN = IDDVN - NSAVE
DVE = IDDVE - EWAVE.
FIGURE 7-4: RELATIONSHIP BETWEEN HNAVFS AND THE REST OF THE NAVIGATION SOFTWARE AND HARDWARE.
Radio Mode is selected by default if there is no valid true airspeed input. No initializations are performed except for the mode flags FLRM (ON) and FLADM (OFF).

The Simulated Airplane is engaged if FLYFLG is TRUE. Otherwise, INAVV must be TRUE and Inertial Mode engaged.

The Simulator provides its own true heading (SMUHDG), calibrated airspeed (CAS), and cross track velocity (IDDXTK). Inertial Mode, naturally, uses INS derived values. Initializations, for one mode or the other, occur as depicted below:

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>SIMULATOR (FLYFLG)</th>
<th>INERTIAL (INAVV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDGTRU</td>
<td>SMUHDG</td>
<td>THDG</td>
</tr>
<tr>
<td>SINTH</td>
<td>SCOSD(HDGTRU)</td>
<td>SCOSD(HDGTRU)</td>
</tr>
<tr>
<td>COSTH</td>
<td>SCOSD(HDGTRU)</td>
<td>SCOSD(HDGTRU)</td>
</tr>
<tr>
<td>NSVAVE</td>
<td>CAS COSTH</td>
<td>VNINS</td>
</tr>
<tr>
<td>EWVAVE</td>
<td>CAS SINTH</td>
<td>VEINS</td>
</tr>
<tr>
<td>IDDATK</td>
<td>ZERO</td>
<td>ATKACC</td>
</tr>
<tr>
<td>IDDXTK</td>
<td>IDDXTK</td>
<td>XTKACC</td>
</tr>
</tbody>
</table>

Additionally, for both modes, INSVAL is set ON and FLADM is set OFF. This concludes the initialization and mode selection for Air Data, Radio, and Simulator modes.

Next, the conditions for the Microwave Landing System (MLS) mode are evaluated. The MLS data valid flag, MLSVLD, is initialized as MLSVAL AND RUNM. If the simulator is engaged or if MLS has not been selected, the mode flag (MLSMOD) is set OFF. Otherwise, MLSMOD is set ON if MLS data is valid and enabled, and destination runway data is available. If the MLS data flag is not valid or MLS is not selected, then MLSMOD is turned OFF. MLS mode may be selected via the toggle switch on the NAV pallet (MLSSLI becomes true) or by depressing the MLS bezel button on the ND display unit. Code in the DS/DF Norden sets the sign bit of the DISPST word for one iteration when the bezel button is depressed. This word is transmitted from the DS/DF to the FM/FC Norden via the DATAAC 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 MLS data flag is not valid or MLS is not selected, then MLSMOD is turned OFF.

If MLSMOD is TRUE then sub-procedure HNAVML is called to calculate the navigation variables. HNAVML sets the MLS first-pass flag, PASS1.

Sub-procedure HNAVB is called unconditionally to calculate the navigation variables based on INS and radio nav data. Then, if MLS is deselected, this set of variables is used.

Once all the required data have been selected, the sines and cosines of latitude and longitude are derived through calls to SCOSD. Then, if the groundspeed equals or exceeds 64 knots, or if both MLSMOD and NAVFLG (GS > 4) are set, the track angle is calculated from the north and east velocities. The east and north velocities are divided by groundspeed to produce the sine and cosine of the track angle, respectively. Otherwise, the track angle, and the sine and cosine of the track angle, are based on the true heading as it
was set during initial mode determination and initialization. The flight path angle, \( \Gamma \), is calculated as the quotient of the filtered rate of altitude change (HDCF), divided by groundspeed and converted to degrees.

Unconditionally, the drift angle and magnetic track angle are calculated through calls to external procedure ANGL. If the groundspeed is below 4 knots, both the navigation mode flag, NAVFLG, and the 64 knot flag, NAV64K, are turned OFF and the ground speed references (GS & GSFPS) are limited, respectively, to a minimum 1 knot and the corresponding value in feet per second. The limit protects against subsequent division by zero. If the groundspeed exceeds 4 knots, NAVFLG is turned ON, but NAV64K is kept OFF until 64 knots is reached. At that point, the Lat/Lon initialization flag, LLINIT, is also turned ON to indicate that the latitude and longitude variables have been initialized either from the INS or Nav data page 3.

PROCEDURE HNAVML:

This routine calculates velocities, accelerations, and positions based on MLS data inputs. It is called only when MLS mode has been selected. It consists of only eleven equations, mostly trigonometric, to calculate GSFPS, VGSDOT, XTACC, VN, VE, GS, LAT, LON, HDCF, and ALTCOR.

PROCEDURE HNAVLB:

This routine calculates the velocity, acceleration, and position values based on the selected source velocities and radio updates. If MLSMOD and PASS1 are both TRUE, the radio nav gains and error terms are replaced by terms calculated from the MLS position estimates. If MLSMOD is FALSE, PASS1 is cleared. The output depends on several conditions.

Depending on the state of the navigation mode flag, NAVFLG, the north/east velocities, IDDVN and IDDVE, and the position coordinates IDDLAT and IDDLON are computed. If NAVFLG is 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. Then, these velocities are converted to earth coordinates and, through calls to external procedure DPINTG, integrated to form the inertially derived latitude, IDDLAT, and longitude, IDDLON. However, if NAVFLG is OFF, the velocity estimates are taken directly from NSVAVE and EWVAVE. Then, if the INS signals are valid but the position coordinates not yet initialized (GS < 4 KTS), the raw INS latitude and longitude are used for IDDLAT and IDDLON.

The inertial ground speed estimate, IDDGS, is figured with the aid of Pythagoras. IDDVN and IDDVE are squared and summed, and the SQRT function outputs IDDGS.

If the simulated airplane is engaged, then HDOT is taken from the complementary filtered vertical velocity, HDCF (set 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. This 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, is formed by integrating the sum of HDDOT and XKV times DELH. Finally, the 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. FBARC is derived by converting the deviation from the standard barometric pressure to feet, taking the difference between that delta and the current value of FBARC, limiting that to a 10 foot maximum, and integrating.

GLOBAL INPUTS:

ACCHAT, ALT, ATKACC, AZ BRG, BARSET, BSETO, CAS, COLDST, DESTIN, DLATFT, DLONFT, DPE, DPN, FLYFLG, HBARO, HDCFSW, HDD, HDOTB, INAVV, K1P, K2P, LATINS, LLINIT, LONINS, MAGHDG, MAGVAR, MCONF, MLSSLI, MLSVAL, POSHAT, RADFT, RLMLS, RWYDEF, RWYSEL, SMUHDG, TASFPS, TASV, THDG, VEINS, VELHAT, VNINS, XTKACC.

GLOBAL OUTPUTS: ALT, ALTCOR, CLAT, CLON, COSTH, COSTKA, DFTANG, DLAT2, DLO2, DPE, DPN, DVE, DVN, FLADM, FLRM, GAMMA, GS, GSFPS, HDCF, HDDOT, HDGTRU, HDOT, IDDALT, IDDATK, IDDGS, IDDLAT, IDDLON, IDDXTK, LAT, LAT MLS, LLINIT, LON, LON MLS, MLSMOD, MLSVLD, NAV64K, NAVFLG, NCUVAL, SINTH, SINTKA, SLAT, SLON, TASGS, TK, TKMAG, VE, VGSDOT, VN, XTACC, ZOMLS, ZDIF
MODULE NAME: HNAVSL

PURPOSE: The radio update navigation routine is the navigation update executive. It calls the routines that manually or automatically tune and monitor the path defined and cross track stations. Information from these stations such as range and bearing are used to update aircraft position and velocity estimations. In addition, HNAVSL performs the ILS update computations.

TASK: SLOW

LANGUAGE: HAL/S

CALLED BY: SLOW

CALLING SEQUENCE: CALL HNAVSL

CALLS TO: TUNXTK, TUNDM2

DESCRIPTION:

HNAVSL is logically partitioned into the following modules:

Initialization Code:
Boolean error flags F0, F2, F3, DMERR, and VORER to be used in the current HNAVSL pass are cleared. Local variables are assigned the most recent position estimate from fast loop navigation.

TUNPATH:
TUNPATH is an internal procedure which either manually tunes station #2 or automatically tunes a path defined station.

* Manual Mode *
If manual tune conditions are met, a check is made to see if certain station #2 error conditions exist. Under these conditions, the procedure RETMSG is called and the retune-station-2 flag for the NCDU is set if the ground speed is greater than 140 knots. The station #2 frequency is fetched and assigned to the output frequency variable ATUNE2. Proper tuning of the station is then confirmed by substantiating equality between ATUNE2 and the input frequency variable DME2FQ. If ATUNE2 does not equal DME2FQ, HAL bit 3 of fail flag FLSTA2 is set and a timer which allows four seconds for proper tuning is initiated.

* Auto Mode / NAV64K not set *
If automatic tune conditions are met and aircraft speed is less than or equal to 64 knots. 2-D guidance validity is examined. If 2-D guidance is possible (GUID2D ≠ 0), the pointer to the station defined by the next waypoint in the path is retrieved. The station #2 frequency is then fetched, assigned, and checked for proper tuning. If 2-D guidance is not possible (GUID2D = 0), the station pointer is set to zero and the station-not-tuned bit of fail flag F2 is set. The less-than-64 knots bit of FLSTA2 is set in any case.
* Auto Mode / NAV64K set *

If automatic tune conditions are met and aircraft speed is greater than 64 knots, the less-than-64 knots bit of FLSTA2 is cleared. If the timer that allows four seconds for proper tuning is zero and FLSTA2 registers an error, the boolean PSFAIL is set to true; otherwise, it is set to false.

Next, the waypoint indicator PTRSTA is compared to the current PTR2D to check for a waypoint change.

If PTRSTA is greater than PTR2D, PSFAIL is tested and if true, the auto-tune routine TUNDM2 is called to find a new station. If PSFAIL is false, the current station is maintained.

If PTRSTA is less than PTR2D, PTRSTA is updated to equal PTR2D and the station defined by the next waypoint is retrieved. If PTRSTA equals PTR2D, the station is not updated. If the aircraft has not surpassed half the total distance to the upcoming waypoint and PSFAIL is false (indicating no station #2 failures), the current station is maintained. If the aircraft has passed the halfway point to the next waypoint or PSFAIL is true, a check is made to see if the station defined by the next waypoint is already tuned. If this station is not already tuned and 2-D guidance is possible (GUID2D ≠ 0), the station is tuned. If this station is already tuned or GUID2D = 0, PSFAIL is examined. If no failures were detected, (PSFAIL is false) this station is maintained; however, if a failure exists, PTRSTA is incremented and the next path defined station is retrieved. At this point, if 2-D guidance is valid, this new path defined station is tuned; otherwise, TUNDM2 is called to find a new station.

TUNDM2:

TUNDM2 is an external procedure which automatically tunes a station #2. It is called by HNAVSL under certain conditions as described in the TUNPTH section above. See TUNDM2 documentation for more information.

TUNXTK:

TUNXTK is an external procedure which manually or automatically tunes the cross-track station. See TUNXTK documentation for more information.

CRBSC:

CRBSC is an internal routine that calculates range and bearing parameters of a navaid relative to the aircraft.

If the boolean PTHSTA is set, the delta latitude (DPHI1) and delta longitude (DLAM1) between the aircraft and the navaid are computed. PTHSTA is cleared.

The sines and cosines of the navaid latitude and longitude are computed. They are used in calculating the components of a vector (PTVECT) whose origin is the center of the earth and whose endpoint is the navaid. DVECT is computed by subtracting PTVECT from PSVECT, where PSVECT is the vector from the earth's center to the aircraft. The magnitude, or slant range of DVECT is assigned to SRMAG.
Using DVECT, the north and east components from the aircraft to the navaid are derived and assigned to the array TRANSV. Taking the square root of the sum of the squared components yields the ground range magnitude (GRMAG). Also, the components are used to calculate the magnetic bearing (MAGBEAR) of the navaid.

Radcal Processing Code:
The extended radius of the geoid to the aircraft is calculated. The vector PSVECT contains the components of the extended radius. PSVECT is derived in the same manner as PTVECT of the CRBSC routine.

Station #3 Processing:
If HAL bit 7 of SIMFLG is set, the local copy of MDME3 (DME3) is set equal to the canned value of CDME3. The appropriate bits of error flag DMEER are set if:

* Error flag FLSTA3 is not zero
* Boolean DME3VD is invalid
* DME3 > 200 nm
* DME3 <= 0
If the navaid address pointer NVAD3A is zero, CMDE3 is set to zero; otherwise, the internal routine CRBSC is called to calculate the slant range magnitude and assign it to CDME3. If CDME3 is greater than 327.67 nm or equal to zero, bits in DMEER are set to indicate an out-of-range error.

If CDME3 has a valid value, the delta range (DRANGE) between CDME3 and MDME3 is computed. If the delta range is greater than or equal to 5 nm, a bit in DMEER is set. If DRANGE is less than 5 nm, two elements of the position error component array (H) are computed as follows:

\[
\begin{align*}
H(1) &= DRANGE \times \cos CB \times DELGN \\
H(5) &= DRANGE \times \sin CB \times DELGN
\end{align*}
\]

where \(\cos CB\) and \(\sin CB\) are the cosine and sine of the navaid bearing and \(DELGN\) is a delta gain. If FLADM or FLRM is set, \(DELGN = 0.5\); otherwise \(DELGN = 0.25\). The array \(H\) is used for position and velocity updating in a later section. At this point, if aircraft altitude is greater than the ground range magnitude (GRMAG) calculated in CRBSC, HAL bit 8 of FO is set.

FO is updated to reflect all station #3 errors registered during the current pass. If errors are registered in DMEER, a fifteen second timer is initiated. If the errors persist after the timer has elapsed, the appropriate bit of FLSTA3 is set so that a new station #3 will be tuned (by TUNXTK).

Station #2 Processing:

If HAL bit 8 of SIMFLG is set, the local copy of MDME2 (DME2) is set equal to the canned value of CDME2. Likewise, if HAL bit 9 of SIMFLG is set, the local copy of MVOR2 (VOR2) is set equal to the canned value of CVOR2. The boolean PTHSTA is set and DMEER is cleared of all previously detected errors so that it may be reused for station #2 error logging. The appropriate bits of DMEER are now set if:

* Error flag FLSTA2 is not zero
* Boolean DME2VD is invalid
* DME2 > 200 nm
* DME2 <= 0

The appropriate bits of fail flag VORER are set if the boolean VORVLD is false. If the navaid address pointer NVAD2A is zero, CDME2 and CVOR2 are set to zero; otherwise, the internal routine CRBSC is called. CRBSC returns the slant range (SRMAG) from the aircraft to the navaid and assigns it to CDME2. CVOR2 is assigned the magnetic bearing to the navaid.

If CDME2 is greater than 327.67 nm or equal to zero, bits in DMEER and VORER are set to reflect an out-of-range error. Also, an additional fail bit in VORER is set if CDME2 is greater than or equal to 150 nm. If CDME2 has a valid value, the delta range (DRANGE) between CDME2 and MDME2 is computed. If the delta range is greater than or equal to 5 nm, bits in VORER and DMEER are set. If DRANGE is less than 5 nm, two more elements of the position error component array (H) are computed as follows:

\[
\begin{align*}
H(1) &= DRANGE \times \cos CB \times DELGN \\
H(5) &= DRANGE \times \sin CB \times DELGN
\end{align*}
\]
\[ H^{(3)} = DRANGE \times \cos(CB) \times DELGN \]
\[ H^{(7)} = DRANGE \times \sin(CB) \times DELGN \]

where \( \cos(CB) \) and \( \sin(CB) \) are the cosine and sine of the navaid bearing and \( DELGN \) is a delta gain. \( DELGN \) is the same for station \#2 as it was for station \#3. At this point, if aircraft altitude is greater than the ground range magnitude (GRMAG) calculated in CRBSC, HAL bit 8 of FO is set.

If VORVLD is valid and the delta bearing between MVOR2 and CVOR2 (DELBR) is less than 30 degrees, the weighted difference used for position and velocity updating is calculated as:

\[ VOR\_WEIGHT = 0.125 \times \frac{\text{ABS}(DME - DMEMAX) \times \text{DELBR} \times \text{DELGN}}{DMEMAX} \]

Two more error components of \( H \) are calculated as:

\[ H^{(4)} = VOR\_WEIGHT \times \text{TRANSV}^{(1)} \]
\[ H^{(8)} = VOR\_WEIGHT \times \text{TRANSV}^{(2)} \]

where \( \text{TRANSV} \) contains the north and east components from the aircraft to the station.

\( F2 \) is updated to reflect the errors registered in DMEER and \( F3 \) is updated to reflect the errors registered in VORER during the current pass. If errors are registered in DMEER, a fifteen second timer is initiated. If errors persist after the timer has elapsed, the appropriate bit of FLSTA2 is set so that a new station will be tuned.

ILS Update Computations:
The boolean \( F5 \) is cleared. If the aircraft is not within an ILS zone (ILSZON is false), it must be ensured that the delta latitude and delta longitude between the aircraft and the localizer shack are both less than 1.40625 degrees before further processing. If this holds true, \( F5 \) is set and the tangent of the glide slope angle (TANGSA) is assigned.

ILS processing stops at this point if ILSZON and \( F5 \) are cleared. If either is set the parameters (CRBLT, CRBLG, CRBMV, CRBEL) used in the internal routine CRBSC are assigned localizer data. Thus, when CRBSC is called, the slant range (RNGLS) and bearing (BRGLS) to the localizer
shack will be computed. If RNGLS is assigned a value that is less than or equal to 1000 feet, or greater than 10 nm, or the absolute value of BRGLS is greater than or equal to 60 degrees, ILS processing is discontinued.

If ILS processing conditions are valid thus far, boolean ILSTMP is set to true, F5 is cleared, and the localizer deviation angle (SLOCDV) is compared to 1.9 degrees. Processing continues if SLOCDV is less than 1.9 degrees, and LOCFS and LOCVLD are both true. F4 is set. Using SLOCDV and the north and east components from the aircraft to the localizer shack (computed by CRBSC and stored in TRANSV), estimates are made for the aircraft's north (DLNPP) and east (DLEPP) position components. A second corrected estimate of the components using glide slope data is performed if VBS is true, F2 indicates station #2 errors, and the current 2-D waypoint (PTR2D) equals the number of waypoints on the path (WPNUM).

If the ILS receiver is not tuned or the aircraft is not receiving valid data, the aircraft position estimates are not computed and F4 is cleared.

Regardless of the occurrence of ILS update computations, ILSZON is set to ILSTMP, and if false, F4 and F5 are cleared. If F4 is cleared and roll is greater than 15 degrees, or NAV64K is false, the appropriate bits of F0 and F2 are set and F4 is cleared.

![Figure 5 Error Vector Perpendicular to Runway Resolved Into Its North and East Components](image-url)
Error Component Setting:
The navigation mode determines the values of the error component variables used for aircraft position and velocity updating in the fast loop navigation computations.

If FO, F2, and F4 all register failures, the position error components DPN and DPE are set to zero indicating no updates. The velocity error components DVN and DVE are zeroed if these failures persist for five minutes, and FLADM or FLRM are not set.

If FO, F2, or F4 do not register failures, the position error components will be set according to navigation mode. These modes and settings are:

**ILS:**
- \( DPN = DLNPP \)
- \( DPE = DLEPP \)

**ILS / DME #2:**
- \( DPN = DLNPP + H$(3) \)
- \( DPE = DLEPP + H$(7) \)

**DME #2 / DME #3:**
- \( DPN = H$(1) + H$(3) \)
- \( DPE = H$(5) + H$(7) \)

**DME #3 / VOR:**
- \( DPN = H$(1) + H$(4) \)
- \( DPE = H$(5) + H$(8) \)

**DME #2 / VOR:**
- \( DPN = H$(3) + H$(4) \)
- \( DPE = H$(7) + H$(8) \)

**DME #3 only:**
- \( DPN = H$(1) \)
- \( DPE = H$(5) \)

**NOTE:** If CDME3 < 2 nm, DPN and DPE are zeroed and HAL bit 14 is set in FO.

**DME #2 only:**
- \( DPN = H$(3) \)
- \( DPE = H$(7) \)

**NOTE:** If CDME2 < 2 nm, DPN and DPE are zeroed and HAL bit 14 is set in F2.
Position and Velocity Gain Settings:

KIP and K2P are velocity and position gains that are set equal to locally declared constants and used in the fast loop navigation computations. Their values are dependent upon the system mode of operation.

NCDU Message:

Two variables NAVTYP and MNAVTY are used to pass navigation mode information to the NCDU. (See DISNAV documentation for details). The NCDU code is a three letter acronym. The first letter represents the system mode. The second and third letters represent station validity. The table below depicts the various combinations.

<table>
<thead>
<tr>
<th>LETTER POSITION</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIRST SECOND THIRD</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>I</td>
<td>Inertial Navigation Mode</td>
</tr>
<tr>
<td>A</td>
<td>Air Data Computer Mode or MLS</td>
</tr>
<tr>
<td>X</td>
<td>Radio Receiver Mode</td>
</tr>
<tr>
<td>D</td>
<td>Valid Station #3</td>
</tr>
<tr>
<td>V</td>
<td>Valid VOR #3</td>
</tr>
<tr>
<td>L</td>
<td>Cross runway position estimated from Localizer</td>
</tr>
<tr>
<td>X</td>
<td>Station #3 Not Tuned</td>
</tr>
<tr>
<td>M</td>
<td>MLS Mode</td>
</tr>
<tr>
<td>G</td>
<td>Along runway position estimated from Glideslope</td>
</tr>
</tbody>
</table>
### TUNE STATION FAIL FLAG TABLE

<table>
<thead>
<tr>
<th>HAL BIT</th>
<th>F0</th>
<th>F2</th>
<th>F3</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>Station not tuned.</td>
</tr>
<tr>
<td>3</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>Receiver not valid.</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>*</td>
<td>MDME &lt;= 0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CDME &gt;= 150 nm.</td>
</tr>
<tr>
<td>5</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>MDME &gt; 200 nm.</td>
</tr>
<tr>
<td>6</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>CDME out of range.</td>
</tr>
<tr>
<td>7</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>CDME - MDME &gt;= 5 nm.</td>
</tr>
<tr>
<td>8</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>Altitude &gt;= Calculated ground range.</td>
</tr>
<tr>
<td>9</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>CDME = 0.</td>
</tr>
<tr>
<td>10</td>
<td>*</td>
<td>*</td>
<td></td>
<td>(GS &lt; 64 kt) or (Roll &gt; 15 deg and ILS invalid).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>Invalid ILS and Stations</td>
</tr>
<tr>
<td>11</td>
<td>*</td>
<td>*</td>
<td></td>
<td>CDME &lt; Single DME minimum range.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HAL BIT</th>
<th>FLSTA3</th>
<th>FLSTA2</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>*</td>
<td>*</td>
<td>Invalid data for 15 seconds.</td>
</tr>
<tr>
<td>3</td>
<td>*</td>
<td>*</td>
<td>Tune output not equal input.</td>
</tr>
<tr>
<td>4</td>
<td>*</td>
<td>*</td>
<td>GS &lt; 64 knots.</td>
</tr>
<tr>
<td>5</td>
<td>*</td>
<td>*</td>
<td>Bad station geometry.</td>
</tr>
</tbody>
</table>

**GLOBAL INPUTS:**
- ACTIVE
- ALTCOR
- ANTLAT
- ANTLPN
- ATNAV2
- BAZFLG
- BINTME
- BRGLS
- CDME2
- CDME3
- COSCB
- COSDA
- COSRH
- DESTIN
- DME2VD
- DME3VD
- DTG
- DTOGO
- DTOR
- ELLIP
- FLADM
- FLRM
- FTONM
- F0
- F2
- F3
- GS
- GSA
- GSDEV
- GUID2D
- IIDALT
- IDDLAT
- IDDLON
- LOCDEV
- LOCFS
- LOCVLD
- MDME2
- MDME3
- MSLMOD
- MVOR2
- NAV64K
- NVAD2A
- NVAD3A
- PSTMR
- PTRPS
- PTR2D
- RADIUS
- ROLL
- RWYHDG
- RYELEV
- SIMFLG
- SINCB
- SINRH
- TIMS1
- TIMS2
- TRANSV
- UPDTMR
- VBS
- VORVLD
- WPNUM

**GLOBAL OUTPUTS:**
- ATUNE2
- CDME2
- CDME3
- COP
- COSCB
- COSDA
- CVOR2
- DELBR
- DELGN
- DLAM1
- DMEER
- DME2FD
- DPE
- DPHI1
- DPN
- DVE
- DVECT
- DYN
- FLSTA2
- FLSTA3
- F0
- FOG
- F2
- F2G
- F3
- F3G
- F4
- GRMAG
- ICLAT
- ICLON
- ILSZON
- ISLAT
- ISLON
- K1P
- K2P
- MNAVTY
- NAVTYP
- NVAD2A
- PSFAIL
- PSVECT
- PTHSTA
- PTRPS
- PTVECT
- PTRSTA
- RETUN2
- RNGLS
- SLLAT
- SLLON
- TANGSA
- TIMS1
- TIMS2
- TRANSV
- UPDTMR
- VORER
MODULE NAME: MLSEX (Microwave Landing System (MLS) Executive

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

TASK: FAST

LANGUAGE: HAL/S

CALLED BY: FAST

CALLING SEQUENCE: CALL MLSEX

CALLS TO: ATND, ATN2D, DISMF, SCOSD, TAND, UNPK

MLS SUMMARY:

The conventional MLS ground equipment transmits three cone-shaped time-reference scanning radio beams from the runway area (Figure 7-3). 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). The third beam (if available) scans up 7 1/2 degrees and down to the reference plane for flare guidance (EL2). A fourth 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, etc., which would vary from airport to airport.

The MLS coordinate system (Figure 7-1) 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 site is located 6 feet behind and 62 feet to the right of the azimuth antenna at the MLS origin. The EL1 site is offset from 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 7-2.

Figure 7-3 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
<table>
<thead>
<tr>
<th>ITEM</th>
<th>LAT</th>
<th>LON</th>
<th>h MSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Az</td>
<td>37.923962°</td>
<td>-75.473675°</td>
<td>41.1 ft.</td>
</tr>
<tr>
<td>DME</td>
<td>37.924066°</td>
<td>-75.473845°</td>
<td>49.7 ft.</td>
</tr>
<tr>
<td>EL</td>
<td>37.944799°</td>
<td>-75.455473°</td>
<td>34.7 ft.</td>
</tr>
<tr>
<td>T.P.6</td>
<td>37.926944°</td>
<td>-75.471304°</td>
<td>34.19 ft.</td>
</tr>
<tr>
<td>T.P.12</td>
<td>37.929268°</td>
<td>-75.469457°</td>
<td>34.75 ft.</td>
</tr>
<tr>
<td>T.P.74</td>
<td>37.946930°</td>
<td>-75.455418°</td>
<td>32.6 ft.</td>
</tr>
<tr>
<td>T.P.75</td>
<td>37.947274°</td>
<td>-75.455145°</td>
<td>32.18 ft.</td>
</tr>
</tbody>
</table>

True Heading Runway 22 = 212.193°

FIGURE 1. WALLOPS FLIGHT CENTER MLS Antenna and Test Point Locations on Runway 22
G/S DEV, CALCULATIONS

Figure 7-2 MLS COORDINATE SYSTEM
FIGURE 7-3

\[
\begin{align*}
0 &= \theta \\
0 &= \phi \\
0 &= \psi
\end{align*}
\]
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 7-3 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:

- positive EL1 and EL2
- positive X, range, and Z displacement
- positive azimuth and 'delta Y'
- 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, which is 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 7-3, Z may be either positive or negative. Likewise, X may be either positive or negative, although flight testing to date has been confined to regions where X is positive.

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:

1) In the normal landing attitude the X-body axis is in a direction approximately 180 degrees to the X-axis of the inertial MLS reference frame.

2) In the normal landing attitude the Z-body axis is in a direction approximately 180 degrees to the Z-axis of the MLS reference frame.

These factors result in the following:

\[
\begin{align*}
X(mls) &= -X(body) \\
Y(mls) &= Y(body) \\
Z(mls) &= -Z(body) = h
\end{align*}
\]
The parameters XDME, YDME and ZEIG 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 ZEIG parameter will have positive/negative sense.

Table 7-1 lists several of the frequently used parameters in the MLS software and their respective polarities.

MLS MODULE 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 the control of the MLS Configuration word (MCONF), which is normally set via the TERMX 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 Table 7-1.

MLSEX first checks the MLS Compute discrete (MLSC). If false, the MLS valid discrete (MLSVAL) is cleared and the first pass flag (FPF) is set to force re-initialization when computations are next begun. The MSB of the configuration word (MCONF$1) is then checked. If clear, MLSC is cleared and processing ends. Otherwise, processing continues by checking the FPF flag.

If FPF is true, MACRO procedure UNPK is called to unpack the upper five bits of the configuration word into the respective booleans. (The lower bits of MCONF affect only the usage of MLS parameters by other modules, and are unpacked in procedure MLOG when MLS mode is selected.) Subroutine RSCON is then called to set up the airport and receiving antenna parameters according to the selection words RWYSEL and ANTSEL.

In either event, MLSEX next calls each of the remaining subroutines in the order as described below, and returns to the executive.

SUBROUTINE RSCON:

RSCON is called only on the first pass after MLSC becomes true. It loads an entry of the AIRPORTS array into the RWYDEF vector according to the RLMLS flag and the RWYSEL index, and loads an entry of the ANT OFF array into the ANT POS vector according to the ANTSEL index. The ANT POS vector describes the position of the selected aircraft receiving antenna relative to the aircraft center of gravity (C.G.), and the RWYDEF vector describes the location of the Range, EL1 and EL2 transmitter antennas relative to the MLS Azimuth antenna (MLS origin). It also sets the loop index (K) to control the number of signals processed, depending on whether EL2 processing was specified in the configuration word (EL2F set).
Note that another table of runway specific constants - also bearing the name RWYDEF - exists in compool FSTCOM. This latter is used by Flight Controls and navigation processing, and is referenced by procedures MLOG and HNAVSL on the first pass after MLS mode is selected.

SUBROUTINE CNTRM:

This module maintains a 128-cycle history of the valid flags associated with the MLS data (MLSSV = [RV, AZV, ELIV, EL2V]), and the complementary filter position limit exceedants (CFXCV), and sets the IC_PF, PF_IC, and PF_CVAL Boolean arrays. Each history consists of a 128-bit binary string (8 words). The associated counter (CFX_CC) is updated only when the state of the validity bit differs from the state of the history bit for that cycle.

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 CTLBK). Otherwise, processing continues. Processing of the signal valids is in a loop 'DO FOR I = 1 TO K;' where K is set by RSCON to 4 if EL2F is true, otherwise to 3. Initially, the local signal valid (MLSLSV) is set equal to the raw signal valid. Then, once the complementary filter has run at least 120 cycles (CFRUN > 120), outlier errors are computed as (MLSRaw - MLSPred). OUTLR V is set true if the corresponding OUTLR ERR is less than OUTLR_LIM, and MLSLSV is set to the 'AND' of MLSSV and OUTLR V.

Next the validity histories (MLSSHIST) and valid counters (MLSS_VC) are updated. The counter value is then checked to set the operate discretes. If MLSLSV is true, the following is performed:

- If the validity count 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.
- If the validity count is >= 115, the PF_CVAL flag is set to indicate that the corresponding signal is usable for position computation.

If MLSLSV is false the following is performed:

CFRUN > 120:
- If the validity count is < 18, the PF_IC and PF_CVAL flags are cleared.
CFRUN <= 120:
- If the validity count is < 115, the PF_CVAL flag is cleared.
- If the validity count is < 58, the PF IC flag is cleared.
- If the validity count = 0, the IC_PF flag is set.

If MLS is valid (MLSVAL set by CTLBK when CFRUN reaches 200), the complementary filter exceedance counters and histories are updated next. 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. 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.

Finally, the history pointers are updated and DISMF is called to pack the OTLIR V, CFXCF and PF_CVAL boolean arrays into the packed discrete word MLSFC for recording.

SUBROUTINE CTLBK:

This module controls selection of the altitude ('Z') reference, initialization of the MLS complementary filter, operation of the CF run counter, and setting of the MLS valid discrete, MLSVAL.

Initially, the first pass flag is checked. If set, it is cleared. the IC MLS 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 clear, processing continues by setting the CF Count Valid discrete initially equal to the 'AND' of the three exceedance count valid flags, the PF count valid flags for Range and Azimuth, and the 'NOT' of FAIL2$9. If CF_CVAL is true, the altitude reference is selected as follows:

EL2 Processing Selected (EL2F true):
- If altitude is above ZUPPER, EL1 is primary and EL2 secondary;
- If ZLOW < altitude < ZUPPER, EL2 is primary and EL1 secondary;
- If 0 < altitude < ZLOW, EL2 is primary and HRAD secondary;
- If GRD = true, HRAD is primary and only.

EL2 Not Selected (EL2F false):
- If the MLS 'X' position is greater than X_HRSW, EL1 is primary and only;
- Else HRAD is primary and only.

If EL2 is primary and not valid, the EL2F flag is cleared. If neither the primary nor secondary (if one exists) altitude reference is valid, CF_CVAL is cleared. Otherwise, the switch ALTREF is set to the selected reference. A check is then made of the INS validity flags. If the INS Attitude Valid is false, or either INS NAV Valid or INS serial bus Valid is false when BMAFLG is not true, a counter is incremented. If this counter reaches 5, CF_CVAL is cleared.

If, after all checks, CF_CVAL is true, the CFRUN counter is checked:
- If CFRUN = 0, ICCF is set true (initializing the complementary filter);
- If CFRUN = 200, MLSVAL is set true; else CFRUN = CFRUN + 1. If, however, CF_CVAL is false, MLSVAL is set false and CFRUN is cleared to 0. If CFRUN was previously non-zero, ICMLS is set true, causing complete reinitialization of the MLS logic.
## TABLE 7-1 MLS PARAMETER INFORMATION

### VECTOR:

<table>
<thead>
<tr>
<th>Vector name</th>
<th>Computed by</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLSRAW</td>
<td>IOFLL</td>
<td>R, AZ, EL1, EL2</td>
</tr>
<tr>
<td>MLSS_P</td>
<td>PFILT</td>
<td>Phat, ... , EL2hat</td>
</tr>
<tr>
<td>MLSS_PRED</td>
<td>PRINV</td>
<td>Rp, ... , EL2P</td>
</tr>
<tr>
<td>OTELERR</td>
<td>CNTRM</td>
<td>[MLSS_P] - [MLSS_PRED]</td>
</tr>
<tr>
<td>XYZ</td>
<td>IOFLL</td>
<td>X, Y, Z (Radar Tk)</td>
</tr>
<tr>
<td>POS CG</td>
<td>XYZIN</td>
<td>X, Y, Z (Compt'd)</td>
</tr>
<tr>
<td>POSHAT</td>
<td>CFILT</td>
<td>Xhat, Yhat, Zhat</td>
</tr>
<tr>
<td>XYZER</td>
<td>XFORM</td>
<td>[POSHAT] - [XYZ]</td>
</tr>
<tr>
<td>VELHAT</td>
<td>CFILT</td>
<td>XDH, YDH, ZDH</td>
</tr>
<tr>
<td>BMACC</td>
<td>IOFLL</td>
<td>Ax, Ay, Fn</td>
</tr>
<tr>
<td>ACC</td>
<td>CFILT</td>
<td>XDD, YDD, ZDD</td>
</tr>
<tr>
<td>ACCHAT</td>
<td>CFILT</td>
<td>XDDH, YDDH, ZDDH</td>
</tr>
</tbody>
</table>

### POLARITY INFORMATION:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Polarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>R, Rhat, Rp</td>
<td>always positive</td>
</tr>
<tr>
<td>Az, Azhat, Azp</td>
<td>+ left of Rwy C.L.</td>
</tr>
<tr>
<td>EL1, EL1hat, EL1hat</td>
<td>+ above phase center</td>
</tr>
<tr>
<td>EL2, EL2hat, EL2hat</td>
<td>+ above phase center</td>
</tr>
<tr>
<td>X, Xhat, HGPIP</td>
<td>+ from Az antenna to threshold</td>
</tr>
<tr>
<td>Y, Yhat, YPROF</td>
<td>+ right of Rwy centerline</td>
</tr>
<tr>
<td>Z, Zhat</td>
<td>+ above MLS X-Y Plane</td>
</tr>
<tr>
<td>XDH, YDH, ZDH</td>
<td>+ on increasing X, Y, Z</td>
</tr>
<tr>
<td>XDDH, YDDH, ZDDH</td>
<td>+ on increasing XDH, YDH, ZDH</td>
</tr>
<tr>
<td>HGPIP</td>
<td>+ GPIP below MLS X-Y Plane</td>
</tr>
<tr>
<td>ETAH</td>
<td>+ above glideslope</td>
</tr>
<tr>
<td>BETAH</td>
<td>+ left of Rwy centerline</td>
</tr>
</tbody>
</table>

### MCONF BIT CONFIGURATION:

<table>
<thead>
<tr>
<th>HAL BIT</th>
<th>BOOLEAN SET</th>
<th>USE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MLSC</td>
<td>Enable MLS computations</td>
</tr>
<tr>
<td>2</td>
<td>RLMLS</td>
<td>Use real vs. simulated MLS data</td>
</tr>
<tr>
<td>3</td>
<td>EL2F</td>
<td>Enable EL2 computations</td>
</tr>
<tr>
<td>4</td>
<td>BMAFLG</td>
<td>Use body MTD vs. INS accelerations in CFILT</td>
</tr>
<tr>
<td>5</td>
<td>RADTKF</td>
<td>Enable radar tracking computations</td>
</tr>
<tr>
<td>6</td>
<td>VGS_FLG</td>
<td>Use VN and VE vs. prefilter velocities to initialize CFILT X and Y velocities unused</td>
</tr>
<tr>
<td>7-9</td>
<td>--</td>
<td>Use MLS ZDDH vs. HDDF in VCWS, Auto, and glidescope tracking</td>
</tr>
<tr>
<td>10</td>
<td>MSW6</td>
<td></td>
</tr>
</tbody>
</table>
SUBROUTINE PFILT:

The Prefilter module uses alpha-beta filters to prefilter the MLS input parameters (Range, Azimuth, EL1 and EL2 (if in use)). The IC PF flags are OR'ed with the ICMLS flag to determine initialization processing. If the result is one, 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 (\( MLSRAW \)). Note that IC PF will be true on the first and 58th cycles of initialization. If both IC PF and ICMLS are true, the filter velocity (\( MLSS_{DHAT} \)) is also zeroed.

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

If MLSVAL is false, filter input (TEMP) is the difference between the measured and predicted values when the input signal valid is true, and zero otherwise. If MLSVAL is true, filter input is the difference between the measured and predicted (\( MLSS_P \)) values when the input signal valid is true, and the difference between \( MLSS_P \) and the prediction from PRINV (\( MLSS\_{PRED} \)) 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:

\[
S_{HAT} = MLSS_P + \text{ALPHA} \times \text{TEMP}; \\
MLSS_{DHAT} = MLSS_{DHAT} + \text{BETA} \times \text{TEMP}; \\
MLSS_P = S_{HAT} + MLSS_{DHAT} \times \text{DELTAT};
\]

Finally, if MLSVAL is true and MLSSLV is also true, the filter output for that signal 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.

SUBROUTINE XFORM:

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. If the Radar Tracking Flag is set, the TDS input position vector (XYZ) is also translated from the tail antenna position to the aircraft C.G. and corrected for transport lag. The
error vector, \( XYZER \), is then computed as the difference between \( XYZ \) and the MLS position estimate, \( POSHAT \). The following equations are evaluated:

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

where: \( C\theta = \cos \theta \), \( S\theta = \sin \theta \), etc.

\[
a = DLPSI + 180
\]

\[
\{ \text{ANT}_\text{VEC} \} = [LMB] \{ \text{ANT}_\text{POS} \}
\]

where: \( \{ \text{ANT}_\text{POS} \} \) is the X,Y,Z position of the MLS receiving antenna relative to the aircraft C.G.

( if RADTKF )

\[
\{ \text{XYZ} \} = \{ \text{XYZ} \} + [LMB] \{ \text{TAIL V} \} + \text{VGA}N\$4 \{ \text{VELHAT} \}
\]

\[
\{ \text{XYZER} \} = \{ \text{POSHAT} \} - \{ \text{XYZ} \}
\]

where: \( \text{TAIL V} \) is the position vector of the radar reflector on the TCV aircraft;
\( \text{VELHAT} \) is the MLS velocity vector;
\( \text{VGA}N\$4 \) is an estimate of the TDS data transport lag in seconds.

SUBROUTINE XYZIN:

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

0) range, azimuth and radio altitude data
ALTREF = 1) range, azimuth and elevation 1 data
2) range, azimuth and elevation 2 data

XYZIN begins calculations when both range and azimuth prefilters have run at least 58 iterations (PF IC), even though the solution is not used until 115 iterations (PF CVAL). 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.

\[
Raz = R + Xdme \cos(Az) - Ydme \sin(Az)
\]

\[
Ya = -Raz \sin(Az)
\]

\[
Y = \sqrt{(Raz - Ya - Za)^2}
\]

\[
Xa = \sqrt{(Raz - Ya - Za)^2}
\]
ALTREF = 0:
\[ Z_a = \text{Hrad} - \text{Htdc} - \text{ANT}_{\text{VEC}3} \]

ALTREF = 1:
\[ Z_a = \tan(\text{EL1}) \sqrt{(X_a - X_{\text{ell1}})^2 + (Y_a - Y_{\text{ell1}})^2} + Z_{\text{ell1}} \]

ALTREF = 2:
\[ Z_a = \tan(\text{EL2}) \sqrt{(X_a - X_{\text{ell2}})^2 + (Y_a - Y_{\text{ell2}})^2} + Z_{\text{ell2}} \]

(all cases)
\[ \{\text{POS}_\text{CG}\} = \{\text{ANT}_\text{VEC}\} + \begin{pmatrix} X_a \\ Y_a \\ Z_a \end{pmatrix} \]

SUBROUTINE CFILT:

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).

Filter initialization occurs when ICCF is set true by CTLBK. During initialization, position output is set to POS CG, acceleration biases are zeroed, and vertical velocity (ZDH = VELHAT$3$) is set to HDOT. XDH and YDH may be initialized from MLS prefilter velocity terms or from INS VN and VE, depending on the setting of VGS_FLAG.

The filter is operated whenever CFRUN is non-zero. Input accelerations in the MLS coordinate frame may be derived from the INS or from the Body Mounted Accelerometers. depending on BMAFLG. BMA accelerations are rotated into the MLS frame using the Lambda (LMB) matrix developed by module XFORM. INS cross track and along track accelerations are rotated into the MLS frame using the sine and cosine of delta track. (Vertical acceleration from the INS (HDD) is already in the proper coordinate frame).

The residual term (EX) is limited, and a flag (CF_LIM_XC) set for each element which exceeds 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.
SUBROUTINE PRINV:

PRINV computes predicted values for the next set of MLS measurements based on the present position estimate (POSHAT) 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 transformed to the receiving antenna location using the ANT_VEC output of module XFORM. Slant range to the DME antenna is next computed using the Pythagorean theorem.

\[ Rp = \sqrt{(X_a - X_{dme})^2 + (Y_a - Y_{dme})^2 + Z_a^2} \]

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

The angle transmitters (Az, EL1, EL2) define a conical angle. Thus Azimuth is computed as follows, using the four quadrant arctan.

\[ Azp = \text{atan2}(-Y_{AH}, \sqrt{X_a + Y_a}) \]

EL1 (and EL2, if EL2F is true) is computed using the small angle arctan:

\[ \text{EL1p} = \frac{Z_a - Z_{ell1}}{\sqrt{(X_a - X_{ell1})^2 + (Y_a - Y_{ell1})^2}} \]

\[ \text{EL2p} = \frac{Z_a - Z_{ell2}}{\sqrt{(X_a - X_{ell2})^2 + (Y_a - Y_{ell2})^2}} \]

GLOBAL INPUTS: ANTSEL, ATKACC, BMACC, COSRH, DLPSI, FAIL2, FLAGS, GRAVO, GRD, GSINS, HDD, HDDOT, HRAD, HRV, IATTV, IDDALT, INAVV, ISBV, MCONF, MLSRAW, MLSSV, PITCH, ROLL, RWYSEL, SIRH, VE, VEINS, VN, VNINS, XTKACC, XYZ.

GLOBAL OUTPUTS: ACCHAT, BMAFLG, CFRUN, CFXCC, CROLL, CTHET, EL2F, EX, FAIL2, MLS,MLSFC, MLSSVC, MLSSVAL, OTLERR, POSHAT, RADTKF, RLMLS, SROLL, STHET, VELHAT, XYZER.
MODULE NAME: NAVIG

PURPOSE: To permit ground checkout of several dynamic functions of the Flight Management/Flight controls computer.

TASK: FAST

LANGUAGE: HAL/S

CALLED BY: FAST

CALLING SEQUENCE: CALL NAVIG

CALLS TO: ANGL, TAND, IAND

DESCRIPTION:
The simulated airplane NAVIG is controlled via the Debug Page of the NCDU, or the utility program TERMX, by setting certain bits of the simulation control word SIMFLG. The actions initiated by setting different bits in SIMFLG are enumerated below:

- Bit 0 - Initialize airplane
- Bit 1 - Fly airplane 200 knots
- Bit 2 - Fly calibrated air speed or 4D path
- Bit 3 - Not used
- Bit 4 - Hold airplane

NOTE: The setting of bits in SIMFLG follows the PDP 11/70 word structure, which has bit 0 as the rightmost bit and bit 15 as the leftmost or most significant. It should be further noted that the HAL/S language follows a bit pattern that has bit 1 starting as the leftmost position of the word and bit 16 as the rightmost or least significant.

Example: SIMFLG = 21 (octal) places airplane in hold and also initializes airplane.

Upon entry a check is made to determine if the flight controls RUN mode is selected. If so, the simulation control word SIMFLG is zeroed. FLYFLG is turned off which signifies that the real airplane is in operation, and control returns to FAST.

If bit 0 of SIMFLG is set, which signifies initialization of the airplane, the following takes place:
A branch is made to routine FLYIC which sets the following:
- Latitude (IDDLAT) is set to the value stored in SIMLAT.
- Longitude (IDDLON) is set to the value stored in SIMLON.
- Altitude (ALT) is set to the value stored in SIMALT.

Airplane Heading (SMUHDG) is set to the value stored in SIMHDG.
Magnetic Variation (MAGVAR) is set to the value stored in SMMAGV.
The NCDU latitude/longitude initialization flag LLINIT is turned on, 2D guidance, 3D guidance and 4D guidance flags are turned off. The simulation roll command (PHISYM) and HDOT complimentary filter (HDCF) are set to zero. Auto engage (AUTOE), path and (PATHND), localizer engage (LOCE), and glide slope engage (GSENG) flags are turned off. The
calibrated air speed (CAS) is set to 2 knots. NAVFLG, which specifies ground speed greater than 2 knots is turned off. Procedure ENGAGE_CAS is called. The IC bit of SIMFLG (bit 0) is turned off.

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

FLYFLG and AUTOE are turned on signifying that the simulated airplane is now in operation.

If the hold bit is on calibrated air speed (CAS) is set to 2 knots and NAVFLG (GS greater than 4 knots flag) is turned off. If the hold bit is off a check is made to determine if the CAS bit is selected. If so, the calibrated air speed is computed from the longitudinal acceleration command (ATCMD) and ground speed feet per second variable (GSFPS). If not, the calibrated airspeed is set to a constant 200 knots, or the value stored in SIMCAS.

Procedure ENGAGE_CAS is called.

Following this control returns to program FAST.

LOCAL ENGAGE_CAS PROCEDURE:

If the hold or initialize bits have not been selected via SIMFLG the following variables are computed:

The simulated roll command (PHISYM)
The bank angle (ROLL)
The heading for the simulated airplane (SMUHDG)
The cross track acceleration (IDDXTK)
The vertical acceleration (HDDOT)
The HDOT complimentary filter (HDCF)
The inertial altitude (ALT)

Following the computation of the above variables control returns to the main routine NAVIG.

GLOBAL INPUTS: ALT, ATCMD, BACMD, CAS, DELTAT, GRAVO, GSFPS, HDCF, HDGIC, IDDXTK, KTQFPS, PHISYM, PDG3D, RUN, RUNM, SIMALT, SIMCAS, SIMFLG, SIMHDG, SIMLAT, SIMLON, SMMAGV, VACMD

GLOBAL OUTPUTS: ALT, CAS, FLAGS, FLYFLG, GUID2D, GUID3D, GUID4D, HDCF, HDDOT, IDDLAT, IDDLON, IDDXTK, LLINIT, MAGVAR, NAVFLG, PATHND, PHISYM, ROLL, SIMFLG, SMUHDG
MODULE NAME: SMINIT
PURPOSE: Initialize simulated airplane (NAVIG).
TASK: SLOW
LANGUAGE: MACRO-11
CALLED BY: SLOW
CALLING SEQUENCE: CALL SMINIT
CALLS TO: None

DESCRIPTION:
SMINIT is used to initialize the variables used by the navigation simulated airplane (NAVIG). The variable SIMFLG must be manually modified, via TERMX or the DEBUG page, to 1 of 4 octal values to choose a corresponding test case. This test case is used to uniquely identify initial conditions corresponding to those used by NAVIG as well as those used by the airplane simulator which executes on a remote computer.

<table>
<thead>
<tr>
<th>OCTAL VALUE</th>
<th>CASE</th>
<th>SPEED</th>
<th>HDG</th>
<th>ALT</th>
<th>LAT</th>
<th>LON</th>
</tr>
</thead>
<tbody>
<tr>
<td>100000</td>
<td>E</td>
<td>130</td>
<td>-117</td>
<td>1440</td>
<td>38.02</td>
<td>75.37</td>
</tr>
<tr>
<td>120000</td>
<td>I</td>
<td>150</td>
<td>106</td>
<td>2745</td>
<td>38.04</td>
<td>75.48</td>
</tr>
<tr>
<td>140000</td>
<td>B</td>
<td>130</td>
<td>-17</td>
<td>2745</td>
<td>37.95</td>
<td>75.36</td>
</tr>
<tr>
<td>160000</td>
<td>J</td>
<td>130</td>
<td>-147</td>
<td>2775</td>
<td>38.09</td>
<td>75.44</td>
</tr>
</tbody>
</table>

GLOBAL INPUTS: SIMFLG

GLOBAL OUTPUTS: SIMCAS, SIMALT, SIMHDG, SIMLAT, SIMLON, SIMFLG
MODULE NAME: TUNCK

PURPOSE: To check for favorable geometry between a path defined station and a cross track station.

TASK: SLOW

LANGUAGE: HAL/S

CALLED BY: TUNXTK

CALLING SEQUENCE: CALL TUNCK (stalat, stalon, max range) ASSIGN (test boolean)

where STALAT = station latitude
STALON = station longitude

CALLS TO: None

DESCRIPTION:
The position of a cross track station with respect to the path defined station is checked to determine if the separation angle (relative to the aircraft) is within limits.

Figure 7-13. Minimum Separation Angle For Cross Path Tuning
The separation angle $\theta$ is calculated as:

$$\theta = \theta_{Ap} - \theta_{Ax}$$

Where: $\theta_{Ap}$ is the bearing of the path defined station.
$\theta_{Ax}$ is the bearing of the cross track station.

The geometry will be favorable if $\theta$ is greater than the minimum separation wedge angle ($\text{WEDGAN} = 30$ degrees) and less than $150$ degrees. Following is the test for these conditions:

If $\text{ABS}(\cos(\theta)) \cdot \tan(\text{WEDGAN}) - \text{ABS}(\sin(\theta)) > 0$ then

$\theta \leq \text{WEDGAN}$ or $\theta \geq 150$ degrees and the test fails.

If $\text{ABS}(\cos(\theta)) \cdot \tan(\text{WEDGAN}) - \text{ABS}(\sin(\theta)) < 0$ then

$150$ degrees $> \theta > \text{WEDGAN}$ and is the favorable condition.

Use of the absolute values of the sine and cosine of $\theta$ are necessary so that the calculation will be correct regardless of whether the separation angle is to the left, right, or behind the aircraft.

The sine and cosine of the separation angle are calculated as follows:

$$\sin(\theta) = \sin(\theta_{Ap} - \theta_{Ax})$$
$$= \sin(\theta_{Ap}) \cdot \cos(\theta_{Ax}) - \cos(\theta_{Ap}) \cdot \sin(\theta_{Ax})$$

$$\cos(\theta) = \cos(\theta_{Ap} - \theta_{Ax})$$
$$= \cos(\theta_{Ap}) \cdot \cos(\theta_{Ax}) + \sin(\theta_{Ap}) \cdot \sin(\theta_{Ax})$$

In this routine the sines and the cosines of the path defined station and cross track station are computed as:

$$\cos(\theta_{Ap}) = \text{path defined station latitude} - \text{A/C latitude}$$
$$= \text{DPHI1}$$

$$\sin(\theta_{Ap}) = (\text{path defined station longitude} - \text{A/C longitude}) \cdot \cos(\text{A/C latitude})$$
$$= \text{DLAM1} \cdot \text{ICLAT}$$

$$\cos(\theta_{Ax}) = \text{cross track station latitude} - \text{A/C latitude}$$
$$= \text{STALAT} - \text{SLLAT}$$
$$= \text{DTLAT}$$

$$\sin(\theta_{Ax}) = (\text{cross track station longitude} - \text{A/C longitude}) \cdot \cos(\text{A/C latitude})$$
$$= (\text{STALON} - \text{SLLON}) \cdot \text{ICLAT}$$
$$= \text{DTLON}$$

NOTE: While latitudes are equidistant, longitudes are not. The difference in longitude must be multiplied by the cosine of the latitude to determine the correct distance.
The test now becomes a simple matter of substitution. If the test fails, a test failure boolean is set to true and control is returned to the calling routine. The geometry is not favorable because the separation angle is invalid.

If the separation angle is valid, the distance between the aircraft and the cross track station is tested to see if it exceeds the maximum range. If the cross track station is greater than 200 miles away, the test failure boolean is set true and control is returned to the calling routine.

GLOBAL INPUTS: DLAM1, DPHI1, SLLAT, SLLON, ICLAT

GLOBAL OUTPUTS: NONE
MODULE NAME: TUNDM2

PURPOSE: To automatically select and tune the path station (station #2).

TASK: SLOW

LANGUAGE: MACRO-11

CALLED BY: HNAVSL

CALLING SEQUENCE: CALL TUNDM2

CALLS TO: SQRT

DESCRIPTION:
HNAVSL calls TUNDM2 when it becomes necessary to auto-tune a new station #2.

Search In Progress:
If there is a search in progress (RINPFL is set), the high altitude flag (HIALTF) is cleared. If the aircraft altitude is greater than or equal to 18,000 feet, HIALTF is set. A transfer of control to label TUN8 is executed to obtain the next navaid in Bulk Data to be tested and tuned.

Search Not In Progress:
If RINPFL is not set (signifying the beginning of a search), the maximum allowed distance between the aircraft and a new station (ZONELM) is initialized to 40 miles. The zone counter (RZNCTR) is cleared.

Navaids in bulk data are organized by longitudinal strips which are two degrees wide. IBPTR points to the first strip index in Bulk Data. A search is initiated to find the strip containing the aircraft. This is done by comparing the IDD longitude of the aircraft to the longitudinal boundaries of the index. If it is not found, the index block pointer is incremented to point to the next strip index and set of longitudinal boundaries until the correct strip is found or the end of the longitudinal strips is reached.

Once the correct strip is found, the pointer to those navaids lying within that strip is stored in RZNPtr and the address of the first navaid within that strip is stored in RTNSCR. RINPFL is then set and the station counter (RSTCTR) is initialized to one. Testing and tuning of the new navaid can now be done.

Navaid Testing and Tuning:
Once a new navaid has been found, it must pass a number of tests before it can be tuned. If the navaid is not a high altitude type, and the aircraft is flying at the high altitude range (above 18,000 feet), a different navaid must be found. Also, if this navaid is the same as the one currently being used, the search must continue. If the navaid passes these tests, its address is stored in RTNSCR.
The distance between the navaid and aircraft position is computed next. A new navaid must be sought if the distance is greater than the current ZONELM. If the range is valid, the RTNPRV array which contains the four most recently tuned path stations is scanned to see if the navaid in question has been recently tuned. If it is found in the table, a new navaid must be found; otherwise, it is ready to be tuned. NVAD2A is set to the address of the navaid in Bulk Data, the station counter (XSTCTR) is incremented (unless it is equal to four in which case it is reset to one). Control then returns to the navigation slow loop.

Strip and Zone Limit Modifications:

In the case that a condition for tuning is not fulfilled, a transfer is made to TUN8 to find a different station. In the event that a suitable navaid is not found within the original strip, the two adjacent longitudinal strips to the east (if present) are cycled through next. Once searching progresses past these strips, the two strips adjacent to the west (if present) of the original are cycled through. At most, two strips to the east and two strips to the west of the strip containing the aircraft can be searched. Once the search reaches the end of the last allowable strip, the zone limit (ZONELM) is increased by forty miles and the process repeats.

If the range reaches 200 miles, RINPFL is cleared and RZNCTR is reset to 1. Clearing RZNCTR will cause an update to the address pointer of the first strip to be searched (RZNPTR) if the aircraft has moved into a different longitudinal strip. Thus, a new set of longitudinal strips will be searched.

GLOBAL INPUTS: IBPTR, ICLAT, IDDALT, RINPFL, RTNPRV, RTNSCR, RZNPTR, SLLAT, SLLON

GLOBAL OUTPUTS: HIALTF, NVAD2A, RINPFL, RSTCTR, RTNPRV, RTNSCR, RZNCTR, RZNPTR, ZONELM
MODULE NAME: TUNXTK

PURPOSE: To select and tune the cross track station (station #3), either manually (VOR) or automatically (DME).

TASK: SLOW

LANGUAGE: MACRO-11

CALLED BY: HNAVSL

CALLING SEQUENCE: CALL TUNXTK

CALLS TO: TUNCHK

DESCRIPTION:
If the groundspeed is less than 64 knots, the local tune cross station routine (TUNEXS) is called. The groundspeed fail bit of flag FLSTA3 is set, and control is returned to the navigation slow loop.

If the groundspeed is greater than or equal to 64 knots, a test is made to determine if manual or auto tuning is desired. Auto tuning is performed if ATNAV3 is not equal to zero, or if NVAD3A equals zero. Manual tuning is performed if ATNAV3 equals zero and NVAD3A does not equal zero. A VOR may only be tuned manually.

Manual Mode:
If manual tuning, a transfer is made to label TUN12. The variable F2 is tested and if non-zero (indicating an invalid path defined station), the geometry test becomes unnecessary and a jump to label T2STA is executed to maintain the current cross station. If F2 equals zero (indicating a valid path defined station), the Bulk Data navaid NVAD3A is used to obtain the selected navaid latitude and longitude. These arguments, along with the maximum tuning range (200 miles), are sent to the TUNCHK routine where they are used to check for favorable geometry between the navaid and the current path defined station.

TUNCHK returns a boolean value which indicates geometry and range validity. If the boolean is set, the bad geometry bit of FLSTA3 is set and control is returned to the navigation slow loop. If the geometry is valid, a transfer is made to label T2STA to tune to the station.

Station Tuning:
Under T2STA, TUNEXS is called to get the cross station frequency. A comparison is made to verify that the tuned output frequency (ATUNE3) equals the tuned input frequency (DME3FQ). If they are equal, the error timer which allows four seconds for proper station tuning is cleared and control is returned to the navigation slow loop. If ATUNE3 does not equal DME3FQ, the appropriate bit of FLSTA3 is set. If the error timer is zero, it is reset to BINTME; otherwise, it remains unchanged. In the manual tuning mode, the error timer has no effect. In the automatic tuning mode, the expiration of the error time causes the algorithm to search for a new station.
TUNEXS loads in a frequency for a navaid from bulk data via the navaid address pointer NVAD3A and stores it in ATUNE3. The fail flag FLSTA3 is cleared and control passes back to TUNXTK.

Auto Mode:
The auto tune logic begins with label T2AUTO. If a failure has not been detected, a test is performed to determine if the aircraft is within the height (H) > range (R) cone for the path defined station. If it is within this "zone of confusion", the current cross station is maintained through a transfer to T2STA.

If a failure has been detected or the aircraft is not in the H > R cone, the station search in progress flag (CINPFL) is tested. If the search is in progress, the address of the most recently tested navaid is fetched and the search is allowed to proceed. If a search is not in progress and there are no failures, a station has been found and its geometry is checked by branching to TUN12. Tuning proceeds at T2STA if the station geometry is good.

If there is a station search in progress and a station failure has been detected, it must be determined whether the tuning-output-not-equal-input failure is the only failure. If this is the sole failure and the error timer has not surpassed its limit of four seconds, the station geometry is checked at TUN12. Tuning proceeds at T2STA if the station is good. If the error time has expired or there was an additional failure, a new station will be sought.

The search for another cross station starts at label T2AI. First, the error timer is cleared. Bulk Data will be cycled through to obtain a navaid that is within a certain range of the aircraft and has good geometry.

The station cycle index (XSTCTR) varies between zero and four. If it is zero (search is just starting), the high altitude flag (HIALTF) is cleared. If aircraft altitude is greater than or equal to 18,000 feet, HIALTF is set. The first searching range, 40 miles, is stored in ZONLIM and the zone counter (ZONCTR) is cleared.

Navaids in bulk Data are organized by longitudinal strips which are two degrees wide. IBPTR points to the first strip index in Bulk Data. A search is initiated to find the strip containing the aircraft. This is done by comparing the IDD longitude of the aircraft to the longitudinal boundaries of the index. The index block pointer is incremented to point to the next strip index and set of longitudinal boundaries until the correct strip is found.

Once the correct strip is found, the pointer to those navaids lying within that strip is stored in ZONPTR. The search in progress flag (CINPFL) is set and the station cycle index (XSTCTR) is set to 1.

The frequency word of the navaid is obtained and tested to determine if the navaid is a VORTAC. The next station in Bulk Data will be examined if the navaid in question is not a VORTAC, or if it is a low altitude range (above 18,000 feet). If it is a high altitude VORTAC, the geometry is checked by TUNCX with the range ZONLIM.
If the geometry of the VORTAC with the path defined station is acceptable and it is not the same as the path defined station or present cross station, it is necessary to see if it was one of the past stations tuned in the cycle.

The array TUNPRV contains a maximum of four previously tuned stations. The TUNPRV array is scanned to see if the VORTAC has been used before. If the VORTAC is found in TUNPRV, a new station must be found. If the station has not previously appeared in the cycle, it is ready to be tuned. The pointer to the navaid is stored in NVAD3A. The station counter is incremented unless it equals four, in which case it is reset to one. A transfer is made to T2STA to complete the tuning process.

CAUTIONARY NOTE: The code described in the paragraph above that tests for previous tuning of a navaid is not currently implemented.

In the case that a condition for tuning is not fulfilled, a transfer is made to TUN8 to find a different station. In the event that a suitable navaid is not found within the original strip, the two adjacent longitudinal strips to the east (if present) of the original are cycled through. At most, two strips to the east and two strips to the west of the strip containing the aircraft can be searched. Once the search reaches the end of the last allowable strip, the zone limit (ZONLIM) is increased by forty miles and the process repeats.

If the range reaches 200 miles, CINPFL is cleared and XSTCTR is reset is 1.

GLOBAL INPUTS: ATNAV3, ATUNE3, BINTME, CINPFL, DME3FQ, F2, FLSTA3, IBPTR, IDDALT, NAV64K, NVAD2A, NVAD3A, SLLON, TUNPRV, TUNSCR, XSTCTR, XSTMR

GLOBAL OUTPUTS: ATUNE3, CINPFL, FLSTA3, HIALTF, NVAD3A, TUNSCR, XSTCTR, XSTMR, ZONCTR, ZONLIM, ZONPTR
GUIDANCE OVERVIEW

The guidance routines use the position, altitude and velocity from navigation to find errors associated with desired position. The desired position, depending on which automatic mode is in effect, is defined from either a previously defined flight path or the knob settings on the Mode Control Panel. Horizontal/vertical steering commands along with the auto-throttle position command will be used to correct for the position errors. 2D and 3D guidance (horizontal and vertical) are calculated by HVGUID while 4D (time) is done by TGUID. NCFM is the interface between the flight control software and the 2D, 3D and 4D commands from HV GUID and TGUID.
MODULE NAME: DSPOT

PURPOSE: To compute analogs of the glideslope and localizer deviation variables to be used by the Displays task.

TASK: FAST

LANGUAGE: HAL/S

CALLED BY: FAST

CALLING SEQUENCE: CALL DSPOT

CALLS TO: SRLMT

DESCRIPTION:
Prior to glideslope engage, the vertical path deviation (BETAH) is set to the negative of RNAV altitude error (HER). Prior to localizer engage the cross-track error (ETAH) is set to the negative of XTK.

Once within .7 degree of the glideslope, BETAH is defined as:

For MLS mode:
\[ \hat{\beta} = \frac{1}{81850} \left( \frac{ZHAT + HGPIP}{XHAT - XGPIP} - TANGSA \right) \]

For ILS mode:
\[ \hat{\beta} = \frac{1428.5 \text{ GSDEV}}{1.25} \]

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 mode:
\[ \hat{\eta} = \frac{28645 \cdot YPROF - YHAT}{1.25 \cdot XHAT} \]

For ILS mode:
\[ \hat{\eta} = \frac{500 \cdot \text{LOCDEV}}{1.25} \]

In any case, both BETAH and ETAH are limited to +/- 1000.
GLOBAL INPUTS:  GSA, GSDEV, GSVLD, HGPIP, LOCDEV, LOCVLD, MLSMOD, MLSRAW(AZ, EL), POSHAT(XHAT, YHAT, ZHAT), TANGSA, XGPIP, YPROF.

GLOBAL OUTPUTS:  BETAH, ETAH, FCFLGS(DLBS, DVBS)
MODULE NAME: HVGUID

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

TASK: FAST

LANGUAGE: HAL/S

CALLED BY: FAST

CALLING SEQUENCE: CALL HVGUID

CALLS TO: None

DESCRIPTION:
The HVGUID software receives information about the present position of the airplane from the navigation software (HNAVFS, HNAVSL) via compool data. 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 two major outputs of this module - Lateral Steering Signal (LATSTR) and Vertical Steering Signal (VERSTR).

LATSTR is the solution to a linear second order differential equation. It is an acceleration command that returns the airplane to the horizontal path near the abeam point.

VERSTR represents the solution of a linear second order differential equation. This solution is a command that takes the airplane back to the planned flight path in the vertical plane.
Since HVGUID has two major outputs and the logic involved in computing the two major outputs is clearly divided and defined, the horizontal and vertical axes will be addressed 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 the flags PATHND, GUID2D, and NAV64K are not set. PATHND is set when the final waypoint on the path has been reached. GUID2D is set when there are at least three waypoints in the guidance buffer and NAV64K is set when the groundspeed is greater than 64 knots.

During initialization, the "to" waypoint pointer (WPPT) 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 NCDU select page.

Once the proper "to" waypoint has been determined, the 2D pointers and the 4D pointers for the airplane are set and the HVGP1 and PTH4DN flags are cleared. The HVGP1 flag is used to indicate that one pass has been made through the HVGUID routine and the PTH4DN flag indicates whether or not the time box has reached the end of path. After the HVGP1 and PTH4DN flags are cleared the program drops into Path Termination loop, clears certain variables and transfers to the Vertical Guidance Off procedure.
II. Straight Segment Calculations

At the beginning of the Straight Segment the following flags are turned off and variables are zeroed in the LEGSW procedure:

- TURN - Airplane in the Turn Flag
- TEND - Airplane halfway through Turn Flag
- ALCFLG - Advanced Lateral Control Flag
- ALCBA - Advanced Lateral Commanded Bank Angle
- AMG - Angle Made Good

In addition DTOGO (Abeam Point to Next Waypoint Distance) is initialized to the initial value of DTG (Distance to Go), HVGPI 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. The Airplane Position Unit Vector ($\hat{PO}$)
2. The Abeam Point Unit Vector ($\hat{POP}$)
3. The Abeam Point to Next Waypoint Distance (DTG)
4. The Abeam Point to Middle of Arc at the Next Waypoint Distance (DTOGO)
5. The Crosstrack Distance to the Planned Flight Path (XTK)

6. The Desired Track along a Great Circle Path (DSRTK)

7. The Track Angle Error (TKE)

8. The Be Careful Flag (BCFLAG)
Airplane Position Unit Vector ($\mathbf{\hat{P}_0}$)

The airplane position vector is one of the primary references in the Guidance Equations. The Airplane Position Unit Vector ($\mathbf{\hat{P}_0}$) is calculated from the airplane latitude (LAT) and longitude (LON) supplied by the navigation algorithms. This value is computed at label HVGI in procedure HVGUID as follows:

$$\mathbf{\hat{P}_0} = \begin{bmatrix} \sin(\text{LAT}) \\ -\cos(\text{LAT}) \sin(\text{LON}) \\ \cos(\text{LAT}) \cos(\text{LON}) \end{bmatrix} = \begin{bmatrix} \text{SLAT} \\ -\text{CLAT} \ast \text{SLON} \\ \text{CLAT} \ast \text{CLON} \end{bmatrix}$$

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 $\mathbf{\hat{P}_0}$ in relationship to the position of the airplane.

Figure 8-1. Position Unit Vector
Abeam Point Unit Vector (POP)

The Abeam Point Unit Vector plays an important role in the horizontal guidance computations. The Abeam Point Unit Vector (POP) points to the point 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 PO which is perpendicular to the plane of the great circle (i.e., parallel to U₁₂) from the vector PO as follows:

\[ \hat{\text{POP}} = \hat{\text{PO}} - (\hat{\text{PO}} \cdot \hat{\text{U₁₂}(i)}) \times \hat{\text{U₁₂}(i)} \]

where: \( \hat{\text{U₁₂}(i)} \) is the Flight Path Normal Unit Vector calculated in the Path Definition routine.

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 8-2.
Figure 8-2. GEOMETRY OF THE ABEAM VECTOR
Abeam Point to Next Waypoint Distance (DTG)

The great zero circle distance from POP to WP(i) is found by computing the angle subtended at the earth center by the unit vectors \( \hat{POP} \) and \( \hat{WP}(i) \), 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 \( \hat{U}_{12}(i) \). Since \( \hat{POP} \) and \( \hat{WP}(i) \) both lie in the plane of the great circle, their cross is perpendicular to that plane and therefore parallel to \( \hat{U}_{12}(i) \) as given by:

\[
\sin(\beta) = (\hat{POP} \times \hat{WP}(i)) \cdot \hat{U}_{12}(i)
\]

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 \( \beta \) is found from dot product as follows:

\[
\cos(\beta) = \hat{POP} \cdot \hat{WP}(i)
\]

The angle is found from a double argument arctangent subroutine, hence the distance is given by:

\[
DTG = Re \times \text{arctangent} \left( \frac{\sin(\beta)}{\cos(\beta)} \right)
\]

where \( Re \) is the local radius of the earth.
Figure 8-3. Abeam Point to Waypoint Distance
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 A02(i) as given:

\[ \text{DTOGO} = \text{DTG} - \text{DTT}(i) + \text{A02}(i) \]

Figure 8-4. 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:

\[ \text{DTOGO} = \text{DTG} \]

Figure 8-5. DME-Arc Distance
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 XTK 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 (PO) and the Abeam Point Unit Vector (POP), in radians, by the radius of the earth as given:

\[
\text{Re} = \text{local radius of the earth} \\
\sin(\alpha) = \hat{P}_0 \cdot U_{12}(i) \\
\cos(\alpha) = \hat{P}_0 \cdot \hat{P}_\text{POP}
\]

then:

\[
\text{XTK} = \text{Re} \cdot \arctangent \left( \frac{\sin(\alpha)}{\cos(\alpha)} \right)
\]

Figure 8-6. Crosstrack Distance
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 \( \hat{\text{POP}} \), pointing in the forward direction along the path. The computation of the desired track involves the unit normal vector \( \hat{U_{12}}(i) \) and a unit vector \( \hat{M} \) which points westward at \( \hat{\text{PO}} \). 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 \( \hat{M} \) is given by the equations:

\[
\begin{align*}
\hat{M} &= \cos(\text{LON}) \\
\hat{M} &= \sin(\text{LON})
\end{align*}
\]

where \( \text{LON} \) is the longitude at the abeam point \( \hat{\text{PO}} \). The orientation of \( \hat{M} \), \( \hat{U_{12}}(i) \) and the path are shown below.

Figure 8-7. Desired Track
The desired track DSRTK is found from the equation:

\[ DSRTK = \arctan \left( \frac{\sin A}{\cos A} \right) \]

where

\[ \sin A = (\hat{U}_{12}(i) \times \hat{M}) \cdot \hat{P} \]
\[ \cos A = \hat{U}_{12}(i) \cdot \hat{M} \]

Figure 8-8. End View of POP

The computation of \( \sin A \) utilizes a dot product with \( \hat{P} \) to convert the vector \( \hat{M} \times \hat{U}_{12}(i) \) to a scalar quantity. This is legitimate because \( \hat{P} \) is a unit vector parallel to \( \hat{M} \times \hat{U}_{12}(i) \).
**Track Angle Error (TKE)**

The Track Angle Error is used as one of the parameters in the Lateral Steering Command LATSTR calculations; it is also used to compute the Overtake Rate (OVTK) and the Star and Circle parameters. This value is computed from the airplane's track (TK) and the Desired Track Angle (DSRTK).

\[ TKE = TK - DSRTK \]

![Diagram showing Track Angle Error](image)

**Figure 8-9. Track Angle Error**
**Be Careful Flag (BCFLAG)**

The Be Careful Flag inhibits Horizontal Path engage by the AGCS mode equations. This flag is set when the Crosstrack Distance (XTK) is greater than 15,000 feet or the Track Angle Error (TKE) is greater than 90 degrees.

If $|TKE| > 90$ degrees or

$$|XTK| > XTKLIM \text{ and } |XTK| > (GSFPS^2/0.268 \text{ GRAV0})(1-\text{CTKE})$$

Then: $BCFLAG = ON$
Else: $BCFLAG = OFF$

where $XTKLIM$ is initialized to 15000'
$CTKE$ is the $\cos(TKE)$

![Figure 8-10. Be Careful Flag](image-url)
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 8-11. Distances to the Turn
There are three criteria for determining that the guidance routine should be making turn calculations:

1. **TURN ≠ 0**

2. **VGS * DELTAT ≥ DTG**

3. **DTG < WPDDT**

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 traveled since the last 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 Nominal Bank Angle (NOMBA)
2. The Magnitude of the Track Angle Change (MAGTA)
3. The Turn Normal Unit Vector (UL2C)
4. The Crosstrack Distance to Planned Flight Path During Turns (XTK)
5. The Desired Track During Turns (DSRTK)
6. The Turn Angle Made Good (AMG)
7. The Airplane Distance Made Good (DMG)
8. The Progress Distance of the Airplane (DTOTL)
9. The Abeam Point to Middle of Arc to Next Waypoint Distance (DTOGO)
10. The Distance Before the Turn to Apply Advanced Lateral Control (RALC)
11. The Advanced Lateral Control Exit Angle (ALCXA)
12. The Advanced Lateral Control Bank Angle (ALCBA)
13. The Lateral Steering Signal (LATSTR)
Nominal Bank Angle (NOMBA)

The Nominal Bank Angle (NOMBA) 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:

\[ \text{NOMBA} = \arctangent \left( \frac{\text{GSFPS}^2}{\text{GRAVO} \times \text{WPRTN}(i)} \right) \]

where \( \text{GRAVO} = \) Nominal acceleration of gravity (32.1739).

Magnitude of the Track Angle Change (MAGTA)

The Magnitude of the Track Angle Change (MAGTA) is used to determine the mid-point and end of a turn. It is calculated by taking the absolute value of the Track Angle Change \( \text{WPTA}(i) \) (see TABLE 4.9-1) defined for the next waypoint, as follows:

\[ \text{MAGTA} = |\text{WPTA}(i)| \]
Turn Normal Unit Vector \( (U_{12C}) \)

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 the Turn Center Unit Vector \( \hat{WPTC}_2 \), the component of that vector in the Turn Center Unit Vector direction \(( (\hat{P}_0 - \hat{WPTC}_2) \cdot \hat{WPTC}_2) \times \hat{WPTC}_2 \) as given:

\[
U_{12C} = ((\hat{P}_0 - \hat{WPTC}_2(i))) - (( (\hat{P}_0 - \hat{WPTC}_2(i)) \cdot \hat{WPTC}_2(i)) \times \hat{WPTC}_2(i))) \text{SIGN}(TA)
\]

where \( TA \) positive for right turn, negative for left turns

\[\text{Figure 8-13. Turn Normal Unit Vector}\]
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 adjusted during the turn to reflect the distance from the airplane PO to the Abeam Point POP, as given:

\[
XTK = Re \times \text{arctangent} \left( -\frac{\hat{PO} \cdot \hat{U_{12C}}}{\hat{PO} \cdot \hat{POP}} \right)
\]

where \( Re \) is the local radius of the earth.

![Diagram of Crosstrack Distance to POP](image)

Figure 8-14. Crosstrack Distance to POP
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 WPU12 with WPU12C and PO with WPTC2 (refer to DSRTK for straight segments), as follows:

\[
\text{DSRTK} = \arctangent\left(\frac{(\text{WPU12C} \times M \cdot \text{WPTC2})}{(\text{WPU12C} \cdot M)}\right)
\]

Figure 8-15. Desired Track during Turns
Figure 8-16  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 ($\mathbf{u}_{12}$) and the Turn Normal Unit Vector ($\mathbf{u}_{12C}$), as given:

$$
\sin(\text{AMG}) = \frac{\mathbf{WP} \times \mathbf{WP}_{12}(t)}{||\mathbf{WP}|| \cdot ||\mathbf{WP}_{12}(t)||}
$$

$$
\cos(\text{AMG}) = \frac{\mathbf{WP} \cdot \mathbf{WP}_{12}(t)}{||\mathbf{WP}|| \cdot ||\mathbf{WP}_{12}(t)||}
$$

AMG = arctangent ($\sin(\text{AMG})$/cos (AMG))

AMG = 360 - AMG (if the track angle is negative, ($\mathbf{WP}_{TA}(t)$) $< 0$), then:

WP$(t)$

WP$_{12}(t)$

WP$_{12C}(t)$

P0

\[\text{Turn Angle Made Good (AMG)}\]
Airplane Distance Made Good (DMG)

The Airplane Distance Made Good (DMG) is the length of the path segments between 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. In the TGUID routine the DMG is compared to the Distance Made Good of the time box (DMG + 1) and they are used to update each other, as given:

\[
\text{when: } 2 \times \text{AMG} > \text{MAGTA} \\
\text{then: } \text{DMG} = \text{DMG} + \text{WPCCD}
\]

Figure 8-17 Distance Made Good
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 Distance, minus the Abeam Point to the Next Waypoint Center of Turn Distance (DTOGO), as given:

\[
\text{DTOTL} = \text{DMG} + \text{WPCCD}(i) - \text{DTOGO}
\]

Figure 8-18  Progress Distance
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:

\[ t = \frac{\text{NOMBA}}{\text{ROLL RATE LIMIT}} \]

Then the distance before the tangent point is:

\[ \text{RALC} = \text{NOMBA} \times \text{CP25} \times \text{GSFPS} \]

where CP25 is the constant .25.

Figure 8-19
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:

\[ ALCXA = \frac{\text{NOMBA} \times 180}{\text{Roll Rate Limit}} \times \frac{\text{GSFPS}}{\text{WPRTN}(i-1)} = \frac{\text{RTOD} \times \text{RALC}}{\text{WPRTN}(i-1)} \]

Figure 8-20
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 shown in the following Digital System Diagram (DSD):

\[ \text{NOMBA} \rightarrow \text{sign(WPTA(i))} \rightarrow \text{ALCBA} \]
LATERAL = \sqrt{\gamma^2 + \dot{\gamma}^2 + \gamma D}^2

Given a linear second order differential equation as given:

The Lateral Steering Signal (LATERAL) represents the solution

Lateral Steering Signal (LATERAL)

Then:

Beyond the roll out point and AlcBA = 0

\[ \text{when } MAGA \leq AMG \Rightarrow \text{ALCXA} \]

Then:

In the turn, not to roll out point and

\[ \text{when } MAGA \leq AMG \Rightarrow \text{ALCXA} \]

\[ (\text{MAGA}) (\text{MAGA}) \]

Then:

In the first half of turn and AlcBA = 0

\[ \text{when } 060 > \text{RALC} + \text{MPA02} \]

Then:

The airplane has not reached the roll-up

\[ \text{when } 060 \geq \text{RALC} + \text{MPA02} \]

The bank angle is computed under the following conditions:
LATERAL = KY * KY + TAKE * DTO + GSP + KYD + ACA + FDPAH

angle (lateral acceleration) components, as given:
error, cross-track rate (lateral velocity) error and bank

The equation is made up of cross-track (lateral distance)
IV Horizontal Guidance:

The following computations are done in horizontal guidance for both straight segments and turns. The following quantities are computed:

1. The Lateral Steering Signal Intercept Angle (PSIi)
2. The Lateral Steering Course Cut Limit (limit)
3. The Lateral Crosstrack Error Signal ($\Delta y$)
4. The Forward Path Integrator (FWDPTH)
5. The Lateral Steering Crosstrack Rate Error Signal ($\Delta y$)
6. The Lateral Steering Signal (LATSTR)
Figure 8-21. Intercept Angle

Planned Flight Path

\[
\text{where: } x = \frac{\left(470^*|\text{XTK}|/\text{GSfps}\times 2\right) - 30}{\text{If } X < 90 \text{ then PSI} \text{I} = 90, \text{ If } X > 90 \text{ then PSI} \text{I}} \\
\]

The intercept angle is the component of the lateral steering signal which determines the direction that the airplane approaches the planned flight path. The angle is determined by the cross track error (XTK) and the ground speed (GSfps) and is limited from 0 to 90 degrees as described below.
Lateral Steering Course Cut Limit (limit)

The purpose of the Course Cut Limit is to limit the crosstrack error signal preventing it from predominating the crosstrack rate signal. The limit is varied with groundspeed (VGS) and intercept angle (PSII), as given:

\[ \text{limit} = \text{GSFPS} \times \text{DTOR} \times \text{PSII} \times \text{KYD} \]

where DTOR is the degrees to radians conversion factor.

Lateral Steering Crosstrack Error Signal (\( \Delta y \))

The Crosstrack Error Signal is one of the three inputs that make up the Lateral Steering signal (LATSTR). The Crosstrack Error Signal is made up of the scaled crosstrack error (XTK * KY), limited by the Course Cut Limiter (limit) as shown in the DSD below.

\[ \text{XTK} \rightarrow \text{KY} \rightarrow \frac{1}{\text{limit}} \rightarrow \Delta y \]

The Error Signal is computed under the following conditions:

\[
\begin{align*}
\Delta y &= -\text{limit} \quad \text{for } -\text{limit} \leq \text{XTK} \times \text{KY} \\
\Delta y &= \text{XTK} \times \text{KY} \quad \text{for } -\text{limit} < \text{XTK} \times \text{KY} < \text{limit} \\
\Delta y &= \text{limit} \quad \text{for } \text{XTK} \times \text{KY} \geq \text{limit}
\end{align*}
\]
Forward Path Integrator (FWDPTH)

Guidance errors caused from mistrim and biases in the autopilot that result in differences between the actual bank angle and the commanded bank angle are referred to as stand-off error. The purpose of the Forward Path Integrator (FWDPTH) is to improve the tracking accuracy of Horizontal Path by nulling the cross path stand-off error. The FWDPTH is the summation of the Crosstrack Error Signal $\Delta y \cdot Ky$ with respect to time, as given:

$$(PSCRATCH1 \cdot DTOR \cdot GSFPS \cdot KYD)$$

where

$PSCRATCH1 = \frac{|XTK|}{GSFPS^2} - 30$  
$RESET = AUTO + TKSEL + |XTK| > 1000FT + |TKE| > 5$ deg

The bank angle command increment generated by the FWDPTH is limited to $\pm 5$ degrees, regardless of the actual integrator output.
Crosstrack Rate Error Signal (Δy*)

The Crosstrack Rate Signal is one of the three main inputs to the Lateral Steering Signal (LATSTR). The Signal Δy is computed from the Ground Speed of the Airplane (GSFPS) and the Track Angle Error (TKE) as given:

$$\Delta y = \text{GSFPS} \times \text{TKE} \times \text{DTOR}$$
(1) The crosstrack rate error signal (\( \dot{\gamma} \)) is limited by the course cut limit of the flight controller equations; \( \text{LATSTR} \) is made up of the lateral crosstrack error signal (\( \dot{\gamma} \)) and the advanced lateral control bank angle (\( \text{ALCBA} \)). As follows:

\[ \text{LATSTR} = \text{LATSTR} - (\text{ITR} \times \text{TOR} \times \text{XYD} \times \text{PSCTRCH}) \times 50. \]

Respectively.

Where: \( \text{KV} \) and \( \text{KYD} \) are the \( \text{KV} \) and \( \text{XYD} \) gains.

\[ \text{LATSTR} = (\dot{\gamma} \times \text{KV} + \dot{\gamma} \times \text{KYD} \times \text{XYD} + \text{ALCBA} + \text{FMDPTH}) \]

The lateral steering signal (\( \text{LATSTR} \)) is the primary output of the lateral steering controller (\( \text{ALCBA} \)).
Abeam Point Commanded Altitude (HC)

The change from WPH(i-1) and WPH(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 (WPGAM) 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 (DTOGO), as given:

\[
HC = WPH(i) - WPGAM(i) \times DTOGO \times DTOR
\]

where: the Gradient WPGAM is illustrated in the following figure.
Because the DME-Arc parameters are defined and computed differently, the computation of HC during the first half of the DME-Arc becomes:

\[ HC = WPH(i+1) - WPGAM(i+1) \times DTOGO \times DTOR \]
**Vertical Guidance Switch Point Distance (HDIS)**

The 3D pointers to the Waypoint to Waypoint Path Gradient (WPGAM) are switched (or updated) before the midpoint of the turn to anticipate vertical maneuvers. There are three different ways that the switch point is determined; first, if the waypoint half arc distance of the 'to' waypoint WPA02 is greater than 1000 feet, the switch point will be set to the half arc distance and the descriptor pointer will switch at the tangent point; secondly, if WPA02 is less than 1000 feet, the switch distance is forced to 1000 feet along the path from the midturn point; finally, if the waypoint to tangent point distance WPDTT is zero, the switch point is also forced to the minimum 1000 feet distance.

\[
HDIS = 1000 \quad \text{if: } WPDTT(i) = 0 \\
\text{or} \\
WPA02(i) < 1000
\]

\[
HDIS = WPA02(i) \quad \text{otherwise}
\]

*Figure 8-23. Switch Point Distance*
Abeam Point Commanded Ground Speed (SDC)

\[
SDC = \left( \frac{-WPV(4D-1) \times DTOGO + WPV(4D)}{WPCCD(4D)} \right) \times KTOFPS
\]

During the first half of a DME-arc turn, the abeam point commanded groundspeed equation is as follows:

\[
SDC = \left( \frac{-WPV(4D+1) - WPV(4D) \times DTOGO + WPV(4D+1)}{WPCCD(4D+1)} \right) \times KTOFPS
\]

NOTE: It is necessary to modify the equation when in the first half of a DME turn because PTR4D is not updated until the middle of the turn is reached.
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 Guidance Switch Point Distance (HDIS)
3. The Raw Vertical Speed Command (HDTCR)
4. The Vertical Acceleration Command (HDDC)
5. The Vertical Path Error (HER)
6. The Vertical Speed Command (HDTC)
7. The Planned Flight Path Angle (PFPA)
8. The Flight Path Angle Error (DFPA3D)
9. The Altitude Rate Command (VSTRA)
10. The Altitude Rate Response (VSTRB)
11. The Vertical Steering Command (VERSTR)
Raw Vertical Speed Command (HDT CR)

The purpose of the Vertical Speed Command (HDT CR) is to derive a desired vertical speed command for comparison with the actual vertical Guidance Speed Command (HDT C). The HDT CR is derived from the Waypoint to Waypoint Path Gradient (WPGAM(i)) and the current ground speed of the airplane (GS FPS), as given:

\[
HDT CR = GS FPS \times WPGAM(i) \times DT OR
\]

\[
WPGAM(i) \quad DT OR \quad x \quad HDT CR
\]

\[
GS FPS \quad + \quad HDT CR
\]
Vertical Acceleration Command (HDDC)

The purpose of the Vertical Acceleration Command is to compute the vertical acceleration portion of the Vertical Steering Command (VSTRA). The Vertical Acceleration Command is derived from the ground acceleration (VGSDOT) and the Waypoint to Waypoint Path Gradient (WPGAM), as given:

\[ HDDC = VGSDOT \times WPGAM(i) \times DTOR \]
Vertical Path Error (HER)

The Vertical Path Error is one of the major elements used in the calculation of the Altitude Rate Command (VSTRA). The HER is the difference between the Abeam Point Commanded Altitude (HC) and the actual altitude (ALTCOR), as given:

\[ \text{HER} = \text{HC} - \text{ALTCOR} \]
The Vertical Speed Command is one of the three major elements which make up the Altitude Rate Command (VSTRA). HDTC is developed by scaling, limiting and integrating the Raw Vertical Speed Command (HDTR). The following steps are taken to develop HDTC.

scaled by KHI:

\[
HDTCl(i) = (HDTCR(i) - HDTCl(i-1)) \times KHI
\]

where \( KHI = 0.16 \)

limited by HDDLMT:

if \( HDTCl(i) \leq -|HDDLMT| \)
then \( HDTCl(i) = -|HDDLMT| \)

if \( -|HDDLMT| < HDTCl(i) < |HDDLMT| \)
then \( HDTCl(i) = HDTCl(i) \)
if \( HDTC_1(i) > |HDDLMT| \)

then \( HDTC_2(i) = HDDLMT \)

where \( HDDLMT = 2 \)

and integrated:

\[
HDTC(i) = HDTC(i-1) + HDTC_2(i) \cdot DELTAT
\]
**Planned Flight Path Angle (PFPA)**

The Planned Flight Path Angle is used in the calculation of the Flight Path Angle Error (DFPA3D) and to generate the gamma bars for the EADI. The PFPA is determined by dividing the Vertical Speed Command (HDTc) by the present Ground Speed of the Airplane (GSFPS).

\[
PFPA = \frac{HDTc}{GSFPS} \text{ RTOD}
\]

**Flight Path Angle Error (DFPA3D)**

The purpose of the Flight Path Angle Error is to be available for display on the NCDU navigation data pages. The Flight Path Angle Error is computed from the difference between the Planned Flight Path Angle (PFPA) and the actual Flight Path Angle (GAMMA).

\[
DFPA3D = PFPA - GAMMA
\]
Where KHA is the vertical path position error gain when AUTO is on; KHD is the vertical path velocity error gain when AUTO is off; KHDA is the vertical path velocity error gain when Auto is off; KHDB is the vertical path position error gain when AUTO is on.

\[
\text{VSTRA} = \text{HER} \times \text{KH} + \text{HTDC} \times \text{KHD} + \text{HDDC}
\]

The attitude rate command is one of the major elements that makes up the vertical steering signal (VSTRA). The attitude rate command signal (VSTRA) is the sum of the scaled attitude error rate command (HER), the vertical speed command (HTDC) and the vertical acceleration command (HDDC), as given:

\[
\text{HER} = \text{KHA} \times \text{HER} + \text{KHD} \times \text{KHDB}
\]

\[
\text{HTDC} = \text{KHDA} \times \text{KHD} + \text{HDDC}
\]

\[
\text{HDDC} = \text{KHDB} \times \text{KHD} + \text{HER}
\]
The Altitude Rate Response is the feedback signal for the Vertical Steering Signal (VERSTR). The Altitude Rate Response is Baro-Inertial Altitude Rate, complementary Filtered (HDCF) and scaled to correspond to the Altitude Rate Command (VSTRA) input to the Vertical Steering Signal, as given:

\[ VSTRB = \text{HDCF} \times \left\{ \begin{array}{l} \text{KHDA} \text{ AND AUTO} \\ \text{OR} \\ \text{KHDB} \text{ AND AUTO} \end{array} \right\} \]

*where KHDA is the vertical path velocity error gain when AUTO is on; KHDB is the vertical path velocity error gain when AUTO is off.*
Vertical Steering Signal (VERSTR)

\[ \text{VERSTR} = \text{VSTRA} - \text{VSTRB} \]

where KALFA is an anti-stall gain processed only for a pitch up command and LHDC is the vertical path velocity command limit.

The Vertical Steering Signal (VERSTR) is the difference between the Altitude Rate Command (VSTRA) and the Altitude Rate Response (VSTRB) after the command has been limited and de-gained if the command was pitch up.

The Vertical Steering Signal is the main output of the Vertical Guidance equations. It is one of the possible inputs to the Flight Controls ELEVP procedure.
GLOBAL INPUTS: ACTIVE, ALTCOR, AMG, CLAT, CLON, COLDST, DELTAT, DMG, DSRTK, DTG, DTVGO, DTOTL, FLAGS, GAMMA, GSFPS, GUID2D, GUID3D, HDCF, HER, KALFA, KHD, NAVG4K, NOMBA, PATHND, PFPA, PTR2D, PTR3D, PTR4D, RALC, SLAT, SLON, TEND, TK, TRE, TKSEL, TURN, VERPTH, VGSDOT, VSTRA, VSTRB, WPPTR, XTK

GLOBAL OUTPUT: ALTARM, AMG, AVECTS, BCFLAG, DFPA3D, DMG, DSRTK, DTG, DTGO, DTOTL, GUID2D, GUID3D, GUID4D, HER, KHD, LATSTR, MDWARN, NCUVAL, NOMBA, PATHND, PFPA, PTH4DN, PTR2D, PTR3D, PTR4D, RALC, SDC, TEND, TKE, TURN, VERSTR, VSTRA, VSTRB, WPPTR, XTK
MODULE NAME: Non Critical Flight Modes (NCFM)

PURPOSE: To calculate guidance commands to the flight controls modules according to the selected mode.

LANGUAGE: HAL/S

TASK: FAST

CALLING SEQUENCE: CALL NCFM

CALLED BY: FAST

CALLS TO: ANGL, SRLMT

DESCRIPTION:

NCFM provides intermediate calculations of the flight path and acceleration command inputs to the flight controls modules based on the current flight situation and on the mode control panel inputs. The primary outputs are:

BACMD  Bank Angle Command,
VACMD  Vertical Acceleration Command, and
ATCMB  Autothrottle Command.

Preliminary calculations determine the angle of attack (ALFSYN) and the anti-stall gain (KALFA). Below 64 knots, ALFSYN equates to PITCH. Above that speed, it is calculated as GAMMA - PITCH, where GAMMA is approximated as HDCF/TASFPS. The anti-stall gain is based on the existing angle of attack. Up to 7 degrees, the gain is one. Above 7 degrees, it is decreased from one to 0.0 at 10 degrees. It remains zero for all angles above 10 degrees.

The logic is generally similar for all three command outputs. The user input comes from the values set by the rotary knobs on the mode control panel. These inputs are stored in the "summers" TKASUM, ALTSUM, FPASUM, and IASSUM, which are incremented or decremented as the knobs are rotated. If a mode is active or preselected, then the relevant command is based on the delta between the summer and the current value. Then, a gain, a limit, or both may be applied. When the mode is not active or preselected, the "summer" is preset to the present value of CAS, FPA, etc.

For BACMD, the calculations are essentially as indicated above. If Track Hold mode is active, DELTKA (TKASUM - TKA) is Gained for groundspeed (.00475 GSFPS). Otherwise, BACMD is taken directly from the lateral steering signal (LATSTR). In either case, BACMD is limited to +/- 25 degrees.

The VACMD calculations are more complex. For altitude hold, the delta value (DELA) is gained by KHALT. If the command is to pitch up (DELA > 0) then the anti-stall gain (KALFA) is applied. This intermediate value (ASC) is the altitude select signal. If Flight Path Angle Hold is selected or active, the delta FPA (DFPAHM) is taken from FPASUM and GAMMA, gained by a function of groundspeed. If the FPA Hold mode is active, this delta FPA becomes the VACMD. Otherwise, if the Altitude Hold mode is active, the altitude select
signal (ASC) is used. Finally, if neither mode is active, the vertical steering signal (VERSTR) is used. In any case, VACMD is limited to 5 ft/sec**2.

The Airspeed Select/Hold calculations are straightforward. The delta is figured normally, and if Time Path mode is not active, ATCMD is the delta (DELCAS) in FT/SEC, times the gain (.1). Otherwise, the Time Guidance speed command (SCMD) is used. Limiting is performed by the autothrottle logic.

GLOBAL INPUTS: ALTARM, ALTCOR, ALTSEL, ALTSUM, CAS, FPASEL, FPASUM, GAMMA, GSFPS, HDCF, IASSEL, IASSUM, LATSTR, NAV64K, PITCH, PSTALT, PSTFPA, PSTIAS, PSTTKA, SCMD, TAS, TIMPTH, TK, TKASUM, TKSEL, VERSTR

GLOBAL OUTPUTS: ALFSYN, ALTARM, ATCMD, BACMD, DELCAS, DELALT, FPASUM, IASSUM, KALFA, TASFPS, TKASUM, VACMD
MODULE NAME: TGUID

PURPOSE: Process time box calculations and compute flight director speed command.

TASK: FAST

LANGUAGE: HAL/S

CALLED BY: FAST

CALLING SEQUENCE: CALL TGUID

CALLS TO: SRLMT

DESCRIPTION:
Upon entry there are three possible branches:
1. time guidance off
2. time guidance first pass
3. time guidance integration loops

Each of these is explained below.

TIME GUIDANCE OFF:
If GUID4D is off the logic takes this path. The following is performed: the time box waypoint pointer (PTR4D1) is set to two if it is not already greater than or equal to two. PTR4D1 is computed to determine the address of the data for the "to" waypoint. The speed command, separation distance of the box and airplane, and first pass flag are all initialized.

TIME GUIDANCE FIRST PASS:
The first pass loop is only executed the first time TGUID is called after the 4D guidance flag is set. The 4D flags for a half and full turn (TEND1 and TURN1) are reset. The distance made good for the box and airplane (DMG1, DMG) are computed according to whether the airplane is ahead or the time box is ahead.
Following this the time guidance first pass flag (TGPI) is set. The initial values of the time box velocity (SDDC) and the time box position (SC) are computed.
The algorithm determines whether or not the box is in a turn or on a straight segment, or if the "to" waypoint is a dme-arc inbound waypoint.

If the "from" waypoint is a dme-arc inbound waypoint, the box is in the second half of a turn. TEND1, a flag which signifies the second half of the turn, is set. The time box magnitude of change value (MAGTA1) and the time box arc distance made good (ADMG) are initialized before branching to the turn loop (AAT).

If the box is in a turn or in a dme-arc, the arc distance made good is set (ADMG), and a branch to the turn loop (AAT) is made.
If the box is on a straight segment, a branch is made to the speed command loop (CDG).

If the "to" waypoint is a dme-arc incound waypoint, or if the "from" waypoint is a dme-arc inbound waypoint, the time box is either on a straight segment, in a turn, or in the first half of a dme-arc. In these cases the time box magnitude of the track change (MAGTA1) is set prior to branching to the turn or straight segment processing.

**TIME GUIDANCE INTEGRATION:**

**PROCEDURE IAVS:**

The time box velocity, and the distance from the box to the "to" waypoint is are updated (SDCC,SC). If the box is already in a turn, loop is called (AAT). If not, a check is made to determine if the box is entering a turn. If it is not, control transfers to the flight director speed command loop (CDG). There are two conditions that signify if the box is entering a turn: the distance from the box to the next waypoint is less than the distance from the waypoint to the tangent point, or the distance from the next waypoint is negative.

When the time box reaches the end of the planned flight path, the 4D guidance flag (GUID4D) is turned off and the end of the path flag (PATHDN) is set. Control then passes to the calling routine.

**PROCEDURE ITN:**

Start of turn calculations are performed here. The time box average acceleration along each path segment, the time box velocity, and the distance from the box to the next waypoint are computed.

**PROCEDURE AAT:**

Turn calculations in AAT are performed when the time integration procedure IAVS is called and the TURN1 flag is on.

When the mid-turn flag is set (TEND1) and the box is at the end of a turn (AMG1 >= MAGTA1), the turn flags are cleared and the flight director speed command procedure (CDG) is called.

When the box is halfway through the turn, the second half turn flag (TEND1) is set. The distance made good, which is the length of the path between the path segment just completed by the time box and the reference waypoint, is updated. The 4D reference pointer is updated to point to the next waypoint. If the turn is not dme-arc a new box acceleration (SDCC), time box box velocity (SDCC), and time box to next waypoint distance (SC) are computed. The flight director speed command procedure (CDG) is called.

**PROCEDURE CDG: (FLIGHT DIRECTOR SPEED COMMAND)**

The flight director speed command is an auto throttle command. It is the acceleration required to minimize the along track distance, velocity, and acceleration between the airplane and the time box. The separation distance from the box to the airplane (SEPR) is used to compute the speed command (SCMD). To prevent the airplane from closing in on the box faster than the dynamics of the airplane allow, the closure rate limit is one-tenth the ground speed associated with the abeam point. The rate is multiplied by the time path velocity error gain in order to size it properly to limit the closure rate.
A condition that might cause a stall is identified by the flight director speed command "too slow" tests. The conditions that might cause the airplane to stall are:

1. The synthesized angle of attack (ALFSYN) is greater than 10 degrees.
2. The airplane is flying too slow to maintain enough lift to fly.

If either of these conditions are met the TOSLOW flag is set and the airplane will not use the speed command in the computations of the auto throttle command.

GLOBAL INPUTS: ACTIVE, ALFSYN, BINTIME, CAS, GSFPS, GUID4D, IASREF, KTOFPS, PTR4D, PTR4D1, SDC, TITIME, TURN1, WPPT

GLOBAL OUTPUTS: ADMG, AMG1, DMG, DMG1, GUID4D, PSTIAS, PTH4DN, PTR4DN, PTR4D1, SC, SCMD, SDCC, SEPR, TEND1, TOSLOW, WPPT
9.0 DATA RECORDING/AQUISITION
DATA RECORDING OVERVIEW

There are five data recording programs which provide the capability to record selected data on tape, paper, and strip charts. DSTDAT, task built with DSTAR, contains a list of data to be recorded on tape through the Data Acquisition System (DAS). It also contains a group of alternate tables which contain data lists to be plotted on the strip charts. DSTAR is the only interactive data recording program. It permits the experimenter to modify the data and parameter lists from DSTDAT and also to set up "snap" tables to print certain groups of data on the onboard line-printer. Finally, it processes the lists and tables and stores addresses and scale factors for DAS.

The SNAP program works with tables generated through DSTAR. When a user specified condition is encountered, SNAP records the associated set of data. Subsequently, in near real-time, SNPOUT formats the data and prints it out on the line-printer.

DASOT takes the data specified by the DAS lists from DSTAR, formats it, and stores it for the DAS, which has a set of 150 words reserved in SIR memory for this purpose. The strip chart data are also included in this data set and are routed to the strip charts by the DAS.

Output from SNAP and the strip charts is, of course, available in flight. Data stored on the DAS tape are available for a "quick look" a day or so following the flight. They are also available over the long term for more thorough data reduction and analysis.
MODULE NAME: DASOT (Data Acquisition System Output)

PURPOSE: To collect and format for output specified aircraft sensor and performance data.

LANGUAGE: MACRO-11

TASK: FAST

CALLING SEQUENCE: CALL DASOT

 CALLED BY: FAST

CALLS TO: None

DESCRIPTION:

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

RSWADR: USAGE

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

ADDRESS= The address of some discrete, such as MLSVAL, which will, when TRUE, cause the secondary set of alternate tables to be used. (RECWD2)

RECWD1/RECWD2: BIT MAP

BITS 2,1,0: Value 0-7, Use Alt Tables 0 - 7 for Strip Blk 1.

BITS 5,4,3: " " " " " " 2.

BITS 7,6 : " 0-3, " " " 8 - 11 " " 3.

E.G.: For a normal configuration of tables 0, 1 and 8, RECWD1 would be set to 000010 octal. For a secondary configuration of tables 4, 5, and 9, RECWD2 would be set to 000154 octal.

Data output is performed starting at label CONT. Because of timing limitations, this code is skipped for any cycle in which the strip chart output configuration is changed. In this case, only the code beginning at label N0IT is executed. The initial logic determines which block of code to use.

At label CONT, the status return bits are packed as follow:

SELWD1: GSDEV, HDOTR, FWHL, HRAD, HDD, R, P, Q

297
Each entry rates 2 bits in the select word. The values may be 0, 1, or 2 which signify channel A, B, or C, respectively.

Next, the boolean discretes from the global MLSFC are divided and stored in the global words DMLFC and MLSOV. The discretes addressed by the local table DISLST are shifted into the global word DISOUT. The MLS/IND position error is calculated and output in the global words LATDIF and LONDIF. Finally, the entire DASPAR list is processed with each entry being fetched, scaled and stored in the global buffer DASBF. The length of the buffer is reported in NDAS and DASOT terminates.

GLOBAL INPUTS: ALTPAR, BCFLAG, FLADM, FLRM, IDDLAT, IDDLON, LAT, LON, MLSFC, NAVFLG, NAV64K, PATHND, RECWD, RECWD1, RECWD2, RSWAOR, TOSLOW

GLOBAL OUTPUTS: DASBF, DASPAR, DISOUT, DMLFC, LATDIF, LONDIF, MLSOV, NDAS, RECWD
TASK NAME: DSTAR (DAS/SNAP Table Access Routine)

PURPOSE: A utility to interactively build or modify parameter lists for use by SNAP and DASOT.

LANGUAGE: MACRO-11

INVOKED BY: A. FAST (On cold start and fresh boot)
B. User (Manually)

CALLING SEQUENCE: A. FAST: CALL DSTAR
B. Manually: Put system in Hold mode. RUN DSTAR

TASK SIZE: 6720 Words.

TASK PRIORITY: A. AUTO mode: 180
B. Manual mode: 50

TASKS INVOKED: None

RESCOMS USED: IOCOM, NAVCOM, SYMTAB

DESCRIPTION:
DSTAR fills or modifies the SNAP and DASLST tables which are used to select, format and print a prescribed set of SNAP data and to select, scale, and output a prescribed set of data to the Data Acquisition System (DAS) for recording and/or for display on the strip charts. (See SNAP and DASOT.)

There are five SNAP tables, each containing 20 words. The first five words constitute the SNAP criteria: the key word address, threshold, type of key, and window size (range). The 15 remaining entries are the addresses of the data items to be printed by the SNAP. The SNAP tables are filled directly as the data are input.

There are up to 150 entries in the DAS list, each consisting of a name, address, and scale factor. Nominally, this list is maintained in an external file, DSTDAT, which is task built with DSTAR. Within DSTDAT, the ASCII names of the variables are kept in the global block DNAME. The addresses and scale factors are kept in the global block DDATA. DSTDAT may be edited externally or modified interactively at run time. Entries may be changed, and/or new ones added up to the 150 entry limit. Entries 1 through 24 are reserved for strip chart parameters and are organized in three blocks of eight entries each. Each of these blocks corresponds with one of the 12 alternate tables which may be read into the block by subroutine DASOT. These alternate tables are maintained in the lower section of DSTDAT, each under its own global name (TABLO, TABL1, etc.) and each following the format of the primary list. Tables zero through seven relate to blocks one or two. Tables eight through 11 relate to block three and are limited to discrete data recording. The contents of DDATA and the parameters from the alternate tables are written to the global DASPAR and ALTPAR tables, respectively, during the DAS dump routine.
Therefore, after a system build, DSTAR must be run in order to fill these tables with whatever has been preset.

DSTAR has two modes of operation: manual and automatic. It is called by FAST on a cold start (COLDST) or on a fresh boot. For a cold start, DSTAR will already have been run and the various tables loaded. In this case only, the automatic mode is implemented. All I/O to the CRT and the printer is disabled and the dump routines are called to transfer all the tables. This mode is totally transparent to the user.

DSTAR is interactive and generally self-explanatory. The date must be entered on the first run after a system boot, after which a carriage return will suffice. The user is given the option to select SNAP or DAS processing, and in the latter case a sub-option to process the 12 alternate tables. Once in, the user may create, modify, delete, or add to the selected table. With DAS processing, whichever course is selected will, upon completion, return to the SNAP/DAS option. On the SNAP side, this will happen only when a SNAP table has been deleted -- all other activities end with a normal exit from the program. Therefore, if both SNAP and DAS processing are to be done, the DAS processing should be done first. See Figure 9-1.

FIGURE 9-1: FUNCTIONAL OVERVIEW.
When a name is requested, the input may be up to 10 characters and it may include a parenthetical index. E.G.: POSHAT(2). If the name matches one in SYMTAB, then the compool offset and data type will be calculated automatically, otherwise, they will be requested. If necessary to use different names to facilitate data reduction, or to name an array element, etc., the user may concoct a unique name and then enter the desired compool and offset information.

With SNAP processing, consecutive elements of a vector or an array may be specified by entering a "repeat" factor after the first element number, as follows: "POSHAT(I*3)". This will cause elements one, two and three of POSHAT to be entered into the SNAP table. The data entered for element one will be copied over for elements two and three.

There are two normal ways to exit DSTAR: by selecting a dump through one of the option menus, or by entering CTRL-Z at any time after the date entry. In either case, dump activation is identical. If a dump flag is set, then the corresponding tables are dumped to the printer. The DAS and alternate table dump flags are initialized "ON" because these tables may have preset data which must be transferred to the DASPAR AND ALTPAR tables for DASOT. The SNAP tables can only be loaded interactively, at which time the flag will be set. Consequently, if the CTRL-Z exit is taken immediately after date entry, only the DAS and alternate tables will be dumped. The case by case results of dump activation are shown below in Figure 9-2.

Error checking is generally limited to identifying and avoiding gross errors, usually in input. Error messages are not displayed, but the last prompt or series of prompts is repeated. The user may correct erroneous entries through the modify options. However, if the error is in the SNAP key variable parameters, it will be necessary to re-enter the entire table.
**FIGURE 9-2:**

**DUMP TO PRINTER: VARIATIONS**

<table>
<thead>
<tr>
<th>WHERE</th>
<th># TYPE</th>
<th>TABLES DUMPED</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITIATED</td>
<td>INITIATION</td>
<td>DASLST</td>
</tr>
<tr>
<td>DATE ENTRY</td>
<td>CTRL-Z</td>
<td>*</td>
</tr>
<tr>
<td>IN DAS MOD</td>
<td>CTRL-Z</td>
<td>X</td>
</tr>
<tr>
<td>&quot;</td>
<td>&quot;O&quot;</td>
<td>X</td>
</tr>
<tr>
<td>IN ALT MOD</td>
<td>CTRL-Z</td>
<td>*</td>
</tr>
<tr>
<td>&quot;</td>
<td>&quot;Y&quot;</td>
<td>*</td>
</tr>
<tr>
<td>IN SNAP MOD</td>
<td>CTRL-Z</td>
<td>*</td>
</tr>
<tr>
<td>&quot;</td>
<td>&quot;O&quot;</td>
<td>*</td>
</tr>
<tr>
<td>ANYWHERE</td>
<td>CTRL-Z</td>
<td>*</td>
</tr>
</tbody>
</table>

* The DASLST and Alternate Tables will be dumped on the first call to DSTAR. Thereafter, a set of tables will be dumped only if the modification routine for that set was entered, causing the dump flag to be set.

**GLOBAL OUTPUTS:**
- ALTPAR, DASPAR, DDATA, DNAME, DSTDAT, SCRT1 thru SCRT5,
- SDATA, TABLO thru TABL11

**MODULES CONTAINED:** DSTDAT, LABEL

**VIRTUAL MEMORY MAP:**

<table>
<thead>
<tr>
<th>BLK.</th>
<th>Offset</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>001334</td>
<td>030620</td>
<td>12688</td>
</tr>
<tr>
<td>001374</td>
<td>020702</td>
<td>08642</td>
</tr>
<tr>
<td>022236</td>
<td>007540</td>
<td>03936</td>
</tr>
<tr>
<td>031776</td>
<td>000156</td>
<td>00110</td>
</tr>
<tr>
<td>001374</td>
<td>020702</td>
<td>08642</td>
</tr>
<tr>
<td>022236</td>
<td>007540</td>
<td>03936</td>
</tr>
<tr>
<td>031776</td>
<td>000156</td>
<td>00110</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BCKCOM:</th>
<th>Offset</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>160000</td>
<td>00252</td>
<td>00170</td>
</tr>
<tr>
<td>140000</td>
<td>013200</td>
<td>05760</td>
</tr>
<tr>
<td>160252</td>
<td>000766</td>
<td>00502</td>
</tr>
<tr>
<td>153200</td>
<td>004410</td>
<td>02312</td>
</tr>
<tr>
<td>161240</td>
<td>001252</td>
<td>00682</td>
</tr>
<tr>
<td>162512</td>
<td>005034</td>
<td>02588</td>
</tr>
<tr>
<td>167546</td>
<td>003640</td>
<td>01952</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RECOM:</th>
<th>Offset</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>167546</td>
<td>003640</td>
<td>01952</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SYMTAB:</th>
<th>Offset</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>120000</td>
<td>012457</td>
<td>05423</td>
</tr>
</tbody>
</table>

*
MODULE NAME: DSTDAT

PURPOSE: To provide a list of data parameters for recording.

TASK: DSTAR

LANGUAGE: MACRO-11

CALLS TO: REAL & INTEG (internally)

CALLED BY: None (Components are addressed by their global names: DNAME, DDATA, TABLE0, TABLE1, etc.)

CALLING SEQUENCE: N/A

DESCRIPTION:

DSTDAT is the component of the DSTAR task which contains the default list of data from the Flight Controls/Flight Management computer to be recorded by the Data Acquisition System (DAS). It contains up to 150 names and corresponding scale factors. Of these, the first 24 specify the output to the strip chart recorders. There is also a group of 12 alternate tables with eight entries each. Three of these may be selected during flight to be read into the strip chart block entries one through twenty-four.

The first section of DSTDAT, labeled DNAME, consists of up to 150 ASCII names. These fields are used only for identification purposes. No calculations are based on them and the names need not be otherwise defined. The fields must include exactly 10 characters, including blanks. However, the fields may not include a right-leaning slash since this symbol terminates an ASCII declaration. An example follows:

```
.ASCII /MLSSVCRNG /
```

Two MACRO functions, REAL and INTEG, are included to convert the data declarations to floating point or integer format at assembly time. This is the only "functional" code in DSTDAT.

The next section, labeled DDATA, contains the data declarations which correspond to the names in the DNAME section. Each line entry consists of a function call to INTEG or REAL, the global name of the data, a scale factor, and a comment section. For example:

```
REAL IDDXTK,10. ; 83 IDDXTK FPS2
INTEG FOG,2048. ; 84 FOG (DME3) PACKED DISCRETE
```

The first two fields are self-explanatory.
The third field, the scale factor, is a decimal number which determines how the data will appear on a strip chart, either on the aircraft or in the post-flight data reduction phase. Starting from a value of zero at the centerline of the chart, the scale is the number at which the needle will be at the edge of the chart. When this limit is exceeded, the recorder "wraps-around"; the needle jumps to one side or the other and continues reflecting relative changes in the data. Thus, aircraft altitude, scaled at 500.0, would wrap quickly as the aircraft climbs or descends, but it would give a good record of small changes from level flight. Scaled at its maximum range, say 40,000 feet, the resolution of the altitude plot would be very poor.

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

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

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

The 100 scale factor for ZHATF will produce good resolution for small changes. At 100 * (2**11) the scale for ZHATC, 204800 feet, is slightly more than the maximum range of ZHAT (32 miles) and it provides a 5 bit overlap in the 16 bit words.

Boolean data items are usually scaled at 2048, which equates to full displacement of the needle.

The comment section of a DDATA line also requires attention because some of the fields are parameters for one of the data reduction programs (CALDAS). The line length may not exceed 72 columns. The last word on the line specifies the unit of measurement. This field may not exceed 10 characters and it may not contain embedded blanks, commas, or slashes. Left-leaning slashes and underlines are acceptable. Any type of unit may be specified, but discretes must be indicated by the word "discrete". In the case of a "packed discrete", then those two words must be present. Abbreviations are not acceptable.
The alternate tables are short versions of DNAME and DDATA. Each table contains 8 ASCII name lines, followed by 8 global names with scale factors. Tables 0 through 7 are for real or integer data. Tables 8 through 11 may contain only booleans, defined as integers (INTEG). In these tables, comments are optional.

GLOBAL INPUTS: None

GLOBAL OUTPUTS: None
NAME: SNAP (SNAPSHOT recording routine)

PURPOSE: To record snapshot values of user specified variables
according to user defined parameters.

LANGUAGE: MACRO-11

CALLED BY: FAST

CALLING SEQUENCE: CALL SNAP

CALLS TO: None

DESCRIPTION:

The SNAP routine records single-event values, called snapshots,
for selected variables and stores them in the SDATA table in NAVCOM
for subsequent output to the line printer by the SNPOUT routine.
There are 5 snapshot criteria tables (SCRIT_), each of 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 a snapshot occurs. (See the SCRIT1 comments
in the RECOM listing.)

Snapshot variables and parameters are specified by the user
through the task DSTAR.

On the first run after a system boot the local START flag will
have been set. SNAP clears the flag and sets the appropriate index
(0 - 4) in the SADR field of each SCRIT_ table. SNAP then checks
the SNAP reset flag, SRST, and, if set, clears the store pointer
(SPTR), the read pointer (RPTR), and SRST. All three are globals
defined in RECOM.

SNAP then processes each of the 5 SCRIT tables in turn, or until
it finds an empty key variable field (SVARA). If the key variable
is an integer or a floating point value, it is processed at the local
labels INTG or FLTP. If not, it is assumed to be a boolean and is
compared with the threshold SVART in the SCRIT table. If it
matches, the snapshot is processed at the local label FLG.

At the label INTG, if the type of comparison specified in
SCRIT (STYPE) is "=" or "", the value is checked for approximate
equality with the threshold by incorporating the range from
SCRIT (SVARS). If within the window, the snapshot is
processed at label FLG. If the type of comparison is a "<" or
">", the key and threshold are checked for that relationship
and, if the condition is met, the snapshot is processed at the
label FLG.

At label FLTP, floating point key variables are processed
like integers except that floating point operations are
forced to compute 16 bit values by setting a 16 bit floating
point operand (eg., the threshold) in the second word of
the floating point instruction.

306
At the label FLG, if bit 15 of SCRIT(STYPE) is set, then that snapshot has already been done and control passes to the local label NXT2. If not done, the "done" flag is set, the title index is moved to SDATA, followed by the values of up to 15 variables listed in the address table SCRIT(SADR). Processing stops at 15 or at the first empty address field. Unused buffer space is cleared, the storage pointer (SPTR) is incremented (Modulo-4), and, at NXT2, the next snapshot is initiated or SNAP terminates.

GLOBAL INPUTS: SCRIT1, SCRIT2, SCRIT3, SCRIT4, SCRIT5, SRST

GLOBAL OUTPUTS: SCRIT1, SCRIT2, SCRIT3, SCRIT4, SCRIT5, RPTR, SPTR, SRST, SDATA
NAME: SNPOUT (SNaPshot OUTput Routine)

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

LANGUAGE: MACRO-II

TASK: SLOW

CALLED BY: SLOW

CALLING SEQUENCE: CALL SNPOUT

CALLS TO: None

DESCRIPTION:
SNPOUT prints out SNAP data whenever new snapshots have been added to the SNAP buffer (SDATA). The global counter SPTR is set by the SNAP routine when new SNAP data is stored. The global counter RPTR is incremented in SNPOUT when it is printed. If the two counters do not agree, then one or more SNAP lists remain to be printed. Both counters are Modulo 4. SNPOUT prints one SNAP list per call.

SNPOUT first increments the read counter RPTR and then formats a header line with the SNAP number and the system time, storing these in the output buffer. It then takes one entry at a time from the table SDATA, checks the form (floating point, boolean or integer), performs the necessary conversions, and stores the ASCII value in the buffer. It repeats this for 5 entries per line, for 3 lines, or until the data location in SDATA is empty. It prints the output buffer and returns.

GLOBAL INPUTS: HRSS, MINS, RPTR, SCRIT1, SCRIT2, SCRIT3, SCRIT4, SCRIT5, SDATA

GLOBAL OUTPUTS: IOACT, RPTR
ADVANCED TRANSPORT OPERATING SYSTEM SOFTWARE UPGRADE

FLIGHT MANAGEMENT / FLIGHT CONTROLS
SOFTWARE DESCRIPTION

Appendix A
Flowcharts

Appendix B
Digital Systems Diagrams
<table>
<thead>
<tr>
<th>ALPHABETIC LIST OF FLOWCHARTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCPRC ..................................................... A-2</td>
</tr>
<tr>
<td>ATCMOD ..................................................... A-4</td>
</tr>
<tr>
<td>BLOW ...................................................... A-41</td>
</tr>
<tr>
<td>CLBIS ..................................................... A-42</td>
</tr>
<tr>
<td>CTLCK ..................................................... A-43</td>
</tr>
<tr>
<td>DASOT ..................................................... A-44</td>
</tr>
<tr>
<td>DECODE .................................................... A-46</td>
</tr>
<tr>
<td>DINUSE .................................................... A-62</td>
</tr>
<tr>
<td>DISFD ..................................................... A-64</td>
</tr>
<tr>
<td>DSTAR ..................................................... A-67</td>
</tr>
<tr>
<td>EPRLMT ................................................... A-102</td>
</tr>
<tr>
<td>ERAD ..................................................... A-108</td>
</tr>
<tr>
<td>FAST ..................................................... A-109</td>
</tr>
<tr>
<td>FDSTR .................................................... A-117</td>
</tr>
<tr>
<td>FLTEM .................................................... A-125</td>
</tr>
<tr>
<td>FLTHDL .................................................... A-127</td>
</tr>
<tr>
<td>FLTMOD .................................................... A-132</td>
</tr>
<tr>
<td>FMTMG .................................................... A-151</td>
</tr>
<tr>
<td>FPEXCP ................................................... A-152</td>
</tr>
<tr>
<td>F2CMP .................................................... A-154</td>
</tr>
<tr>
<td>GETDAT ................................................... A-159</td>
</tr>
<tr>
<td>HNAVSL ................................................... A-160</td>
</tr>
<tr>
<td>HVGUID ................................................... A-180</td>
</tr>
<tr>
<td>ILSRC .................................................... A-194</td>
</tr>
<tr>
<td>INIMOD ................................................... A-195</td>
</tr>
<tr>
<td>LOKMOD ................................................... A-205</td>
</tr>
<tr>
<td>MLOG ..................................................... A-237</td>
</tr>
<tr>
<td>MLSEX ................................................... A-247</td>
</tr>
<tr>
<td>MSPLGC ................................................... A-267</td>
</tr>
<tr>
<td>MSPRO ................................................... A-276</td>
</tr>
<tr>
<td>NCDKEY ................................................... A-278</td>
</tr>
<tr>
<td>NCDUEX ................................................... A-282</td>
</tr>
<tr>
<td>NCDUID ................................................... A-285</td>
</tr>
<tr>
<td>NVDMOD ................................................... A-287</td>
</tr>
<tr>
<td>PANEL .................................................... A-304</td>
</tr>
<tr>
<td>PATHDF ................................................... A-307</td>
</tr>
<tr>
<td>PRFLT .................................................... A-318</td>
</tr>
<tr>
<td>RDALT .................................................... A-320</td>
</tr>
<tr>
<td>RGYRO .................................................... A-321</td>
</tr>
<tr>
<td>SELMOD .................................................... A-323</td>
</tr>
<tr>
<td>SINUSE .................................................... A-338</td>
</tr>
<tr>
<td>SLOW ..................................................... A-341</td>
</tr>
<tr>
<td>SNAP ..................................................... A-345</td>
</tr>
<tr>
<td>SNPOUT ................................................... A-350</td>
</tr>
<tr>
<td>SRPTA .................................................... A-353</td>
</tr>
<tr>
<td>SRVCK .................................................... A-357</td>
</tr>
<tr>
<td>TGUID .................................................... A-363</td>
</tr>
<tr>
<td>TUNCK .................................................... A-370</td>
</tr>
<tr>
<td>TUNDM2 ................................................... A-371</td>
</tr>
<tr>
<td>TUNXTK ................................................... A-383</td>
</tr>
</tbody>
</table>
ACCPRC

MCONF 00070 ?

MCONF 0020 ?

SQRT

AMAG = SQRT (ATKACC^2 + XTKACC^2)

COMPUTE LOWER RESOLUTION VALUES OF
ATKACC AND XTKACC BASED ON AMAG AND A HYPOTHETICAL
STEP SIZE

A

MCONF 0040 ?

ACCSTP

COMPUTE ATKACC AS
A FUNCTION OF PULSES/SEC

ACCSTP

COMPUTE XTKACC AS
A FUNCTION OF PULSES/SEC
SCOSD
COMPUTE SIN & COS OF DFTANG

MLSC ?

SCOSD
COMPUTE SIN & COS OF ROLL

SCOSD
COMPUTE SIN & COS OF PITCH

COMPUTE 2 X 3 LAMBDA MATRIX

\{\text{ATKACC}\} = [\text{LMB}]\{\text{BMACC}\}

RETURN
NOTE: Path definition functions are requested by ATC MOD. When they are completed, the NAV interface flag (NGIDON) is set so the busy message can be cleared and NCDU functions can continue.

NOTE: All functions except REJ cause AREJ1 to be cleared. AREJ1 is a flag that is used to keep track of 2 consecutive REJ function presses.
NOTE: Reclearance (clearance while an active path exists) may only be initiated by the WPT or SID function keys.
NOTE: The NCDU display for the ATC page consists of a header line, 6 lines of user entries and the bottom message line. The user entry lines are scrolled upward, with the most recent user entry always appearing on the sixth line. The ATC page remembers 19 lines of user entry, so scrolling can be done by using the UP/DOWN function keys. A pointer is used to mark the offset in the user entry buffer to designate which user entry is the current sixth line of the 6 line display window. The bottom line is used to display prompts and error messages.
NOTE: ATCMOD stores waypoints in the provisional guidance buffer (PGBUF) and keeps track of the next insertion position with the pointers PPTRZD (byte offset) and PWPSTF (wpn number). Since many waypoints can be stored with one entry (STAR etc.), the last pointer positions must be saved in case of the last user entry being rejected.

NOTE: If the user entry filled much of the input line, line skip is set so that description entries such as ALT will appear on the line below the long entry.
RESET PCEBUF PTRS TO MAKE
PRESENT POSITION THE FIRST
WPT. THIS ENTRY MAY INITIATE
RECESSION SO CLEAR START
OF RECESSION EXIT

WPTSTR
STORE WPT, USE
FLAG TO INHIBIT
ACTION CALL TO
RECESSION

STORE CURRENT
MAP ALT & E
IN PLAIN FOR
POS WPT

FLAG FIRST WPT
AS POS AND
NOTE. IF WPT
MUST BE RE-DONE
AT EXECUTION

RECESSION
UNDERWAY
Y
N
MAKE A/D
ACTIVE POS-
ITION THE RE-
CLEARANCE
POSITION

5B
AWSTR
10B
ALREC1

ORIGINAL PAGE IS
OF POOR QUALITY.
SEE NOTE ON PAGE # 12

ORIGINAL PAGE IS OF POOR QUALITY
SEARCH THE UPTS FROM THE 'SID' ENTRY MADE PRIOR TO THIS WPT ENTRY FOR A MATCH.

WPT FOUND

Y

TERMINATE 'SID' AT THIS WPT BY BULKING UP POINTERS IN PGEUF

N

(REGULAR WPT ENTRY)

5A

AWSTR

5C

ATCTR M

ATCMOD
(PAGE 9 OF 37)
NOTE: When waypoints are being entered, after reclearance was started, each new waypoint is checked if it rejoins the active path.

NOTE: When a route end wpt was entered, the section RTEND jumps to I1A under reclearance. The RTE end wpt will already be included in PSBUF at this point.

NOTE: Copy the active wpts, starting at the rejoind wpt, to the provisional buffer at the current entry position.
NOTE: Routes are a series of associated waypoints in bulk data. They contain many consecutive waypoints covering a large distance. Any subset of consecutive waypoints may be extracted and used as part of a flight path.

Route entries are made in 3 steps. First the name of a waypoint on the route is entered to designate where on the route to start. Next the route name is entered and finally the terminating waypoint name. The first entry is a normal waypoint entry, which saves the address of the waypoint in WPTENT. The route entry finds the offset of the first entry and stores it in RTESTR and also stores the route address in RTENTR to flag the next waypoint entry as a route terminator. The last entry (see page #8) causes the actual storing of the route waypoints in the provisional guidance buffer.

All waypoints between the two waypoint entries are stored starting at the first entry and moving either forward or backward in BULK DATA to reach the last entered waypoint.
SID ENTRY

GET ORIGIN RUNWAY, RESET
PERF PIES 50 THEE WILL
BE THE FIRST WPTS AND
ALL ENTRIES WILL BE
MADE UNDER ORIGIN CLEAR-
ANCE CONDITIONS.

SAVE PERF PIES FOR RED-
VICE OF THIS ENTRY

FOR EACH
WPT

STORE WPT IN
GUID. BUFFER

WPTSTR
STORE WPT
FEATURES FROM
BULK DATA

STORE SPEED, ALTITUDE,
SEEDUS OF TMG, AND BEAR-
ING FROM SID DEFINITION.
IF APPLICABLE, SET GTS:
IAS REF, DME, ARC, FAD
INHIBIT. IF DOE SET
REFERENCE LENTION AS
WPT LAT/LON, SET
INDEX FOR SID, STAR
OR MAP WPT

SET NO-DISPLAY
WPT AS
1ST - SID
LAST - STAR

DO NOT RECLEARANCE
OR
NO NEW PTA OR
PTA POSITION NOT SET

SET NEW
PTA POSITIONS

ATCTRM
ORIGINAL PAGE IS OF POOR QUALITY
FLIGHT LEVEL ENTRY

ISA

ATCODS1

SUBROUTINE DOES NOT RETURN ON ERROR

SET-UP FOR CASCADING FILE

ATCODFS2

STORE FILE AT LAST WKT

IF NOT ON IDL WKT OF RTE ENTRY, SET FILE "BIT"

FORMAT ATC SCROLL LINE

5E

ATCSEN

ORIGINAL PAGE IS OF POOR QUALITY
19A

ATCDS1

DATA ENTRIES ALLOWED?

SUBROUTINE DOES NOT RETURN ON ERROR

SET-UP FOR CASCADED ALT

ATCDS2

STORE ALT AT LAST WPT

IF NOT ON 160
WPT OF FTE
ENTRY, CLEAR
ALT BIT

FORMAT ALT SCROLL LINE

5E

ATCSEN

ORIGINAL PAGE IS
OF POOR QUALITY
ATCDSQ

FIRST GFT OF A REF/ANY

Y

SET NO STORE FLAG

N

STORE INPUT VALUE INTO REG0F

CLEAR NO STORE FLAG

RETURN
ATCXQ

FROM MODE
N

RETURN

DEC MIST WPTS
Y

INSELH

RESET 'HOLD' AND 'TO' WPT ON OTHER ACFT PAGE

PLACE LINK-UP MESSAGE ON ATC PAGE IF NEW PATH DOES NOT LINK TO OLD PATH OR IF INTELLIGENCE ON OTHER ACFT
REQUEST PATH EFFECT FROM PATH OF
START FRAME CYCLE OF RWY OR TO INS,
SET TO 'WPT TO WPT WPT' B' WPT
CLEAR PROVISIONAL MODE.

ORIGINAL CLEARANCE
Y

ATCXQ.2

15 WPT PTA = CURRENT TIME

ORIGINAL PAGE IS OF POOR QUALITY
CLEAR CUE MESSAGE FTR AND ACTIVE TO PROVIDE BIAS

YES

WHEN A "POS" IS USED TO START A PATH, THE CURRENT VALUES ARE USED FOR THE NEW POSITION. NO CHANGES TO THESE VALUES TO THE IP POSITION AT THE TIME OF THE "EXEC"
NOTE: This procedure is called when the pilot chooses the current position (POS) along the path as a HOLD waypoint. The active path is cooled back (FROM wpt to end of path) to the provisional buffer in PROVMD and the pilot defined waypoint is inserted between the 'FROM' and 'TO' waypoints by WPTSTR.
PROVOG

CLEAR ATC ORIG-AL CLEARANCE VARIABLES

31A

CLEAR ATC RECLEARANCE VARIABLES

RETURN

ORIGINAL PAGE IS OF POOR QUALITY

A-34

ATCMOD
(PAGE 31 OF 37)
**Diagram Description:**

- **Start Node:** PRIMAP

- **Branches:**
  - **Yes:**
    - **Decision Node:** Rwy entered and map at this runway?
    - **Branches:**
      - **Yes:**
        - **Decision Node:**
          - **Note:** Called at start of final leg to runway
      - **No:** RETURN

- **Branches:**
  - **No:**
    - **Decision Node:**
      - **Action:** Reset ATC variables, format map input to ATC input buffer.
    - **Action:** Set up FTG to map input for storing in PBUF.
    - **Action:** Start provisional mode

- **Flow:**
  - **Node:** 13C
  - **Node:** 5101

**Notes:**
- Original page is of poor quality.

**References:**
- A-35
- ACMOD
  - Page 32 of 37
RETIME

Y

PASS OF NECK CHECK

N

RESET CALL CONDITION

(REFER TO OK. GUIDANCE)

AND NOT NEW PTA

Y

SET PTA TIME AND POSITION TO CLEAR
EJECTION. FILE NEW PTA

N

RETURN

RETURN

ORIGINAL PAGE IS OF POOR QUALITY
WPTST2

STORE CASCADeD ALT.OR.FIL AND GS

RETURN

RWYDAT

FILL IN THE FOLLOWING COMPOL VARIABLES, FOR THE CURRENT RUNWAY, FROM BULK DATA:

RWYLAT, RWYLEN, ANTLAT, ANTLON, RWYHDG, RYLEVEL, RWYLEN, SINRH, COSRH, GSA

RETURN
BLOW

HRADE < 5 FT?

N

\[ X = \sinh \cdot \text{TASGS} \cdot \text{VE} \]
\[ Y = \cosh \cdot \text{TASGS} \cdot \text{VN} \]

ATN2D

WD = ATAN(X,Y)

SQRT

WS = \sqrt{X^2 + Y^2}

RETURN

SET WIND SPEED = 0
DASOT

CHECK ALT. STRIP CHART CONFIGURATION REQUIREMENT

NORMAL ?

ALTS:

SAME SET OF ALT

ALREADY DONE ?

CONT:

BUILD SELWDL, 2, 3

FETCH NEW SETUP PATTERN

DOIT:

BUILD POINTERS: SELECTED ALT. TABLE, AND DASPAR, BLK1.

BLK1:

MOVE ALTERNATE TABLE TO BLOCK 1.

BLK2:

MOVE ALTERNATE TABLE TO BLOCK 2.

BLK3:

MOVE ALTERNATE TABLE TO BLOCK 3.

A

D

(OUT)
BUILD DMLFC & DISOUT

BUILD AND STORE: LATDIF, LONDIF.

DASLP.

FETCH DAS DATA ADDRESS.

EMPTY?

N

F/P.?

Y

FETCH DATA, SCALE AND STORE

F/P.?

N

Y

FETCH DATA, SCALE AND STORE

ANY MORE?

N

ANY MORE?

Y

RETURN

OUT:

CALCULATE AND RECORD BUFFER SIZE

PAGE 2 OF 2

A-45
SAVE DECODED TYPE, LATITUDE AND LONGITUDE

SAVE 5 CHARACTER INPUT NAME

STORE DATABASE FUNCTION TYPE TO ENABLE USE OF ACTIVE FACE FURTHER

RETURN
NAVIAID WAYPOINT

4A

LUNAV
SEARCH BULK DATA

FOUND

Y

SAVE ADDRESS IN DECMUX AND DECKAV

DECNV
FETCH MAGNETIC VARIATION

2A

N

ERROR #3

RETURN

NOTE: FOR A NAVIAID, THE ASSOCIATED NAVIAID (DECMUX) IS ITSELF
PILOT DEPICTED WHICH:

7A

LUWAY SEARCH PILOT BUFFER

FOUND

(NOT IN REALITY)

BREAK #3

RETURN

DECODER: ADDRESS OF PPT
DECODER = MACRO ADDRESS FOR THIS PPT

DECODER

FETCH MAGNETIC VARIATION

2A
POSITION - CREATE NEW WAYPOINT

LAT/LOL/MAGVAR = CURRENT AIRCRAFT VALUES
DECNAV = ADDRESS OF CURRENT PATH STATION

WPINVT
CREATE NEW WPT

QC
OF POOR QUALITY
ORIGINAL PAGE IS

RETURN

NOT IN MEMORY)

DATA

Search

Search

RETURN

END OF

END OF

RETURN

END OF
NOTE: Select page #1, option #5 uses this code for WPT/BEW entry of the radial symbol. The KOT may not be a POS for that option however.

LAT/lon/magvar = current aircraft values
decNav = address of current path station

decode bearing and range, then format the values to the ndc input buffer. Note only the bearing is handled if select option = 5 is being processed.

(error # 1)

format error

(error # 1)

ORIGINAL PAGE IS
OF POOR QUALITY

A-58

DECODE
(PAGE 13 OF 16)
CLEAR ALL DISCRETE SENSOR IN USE BITS

IN PRENG, FFDE, OR MANEL?

SET DISCRETE SENSOR IN USE FOR LANDS, AUTOS, VCWSS, AND ACWSS

MLSMOD?

LANDR?

DLPSI > 90?

SET DISCRETE IN USE FOR LOCFS

VBS?

SET DISCRETE IN USE FOR GSVLD

NXT2

NXT3

NXT4

ORIGINAL PAGE IS OF POOR QUALITY.
DISFD

GET SERIAL BUS WRD & DISCRETES

CLR DME VALID BIT (BIT 1)

LOCFS ?

Y

CLEAR LOCALIZER VALID FLAG

BUS ERROR ?

Y

SET ERROR FLAG (ISBV)

N

A, B, C DONE ?

Y

FETCH SINUS3

N

SINGLE THREAD ?

Y

SET VALID BITS

N

DEBOUNCE ROUTINE. (SEE TEXT)
BUILD OPTION MESSAGE

ENTER OPTION:
0 = CHANGE WHOLE TABLE
1 to nn = CHANGE ENTRY nn
n+1 = ADD ENTRY

ATDCI
CONVERT REPLY TO INTEGER

REDO WHOLE TABLE
Y
N

ADD ENTRY
Y
N

LOCATE ENTRY TO CHANGE

SMCHG

ITEM
FETCH AND STORE NEW ENTRY.

S NM1:

PAGE 3 OF 34
ITEM
FETCH/STORE ADDRESSES AND NAMES

MORE?
Y
N
CLEAR REMAINING OLD ENTRIES

SNAP PROCESSING COMPLETE
DSPIRC

DIRM
ALTERNATES
OR
DAS TBLS?

ALT/DAS

P (ALTMOD)

ITEMS ARE IN THE DASLIST.
YOUR OPTIONS ARE:
0 DUMP DASLIST TBLS TO PRTN
1 XX MODIFY ENTRY
(N+1) ADD AN ENTRY TO DAS LIST.

DUMP?

NEW ENTRY?

DIRM
SET ENTRY
NUMBER
IN MSG.

DIRM
ADD
HOW
MANY?

ATDCI
CONVERT
REPLY TO
INTEGER

DIRM
"CHANGE
ENTRY TO"

R (OSMOD)
ATDCI

GET ASCII
DECIMAL
CHARACTER

NUMBER?

> 9

CHANGE
ASCII TO
INTEGER

MULT. BY 10.
-REPOSITION
PRIOR
DIGITS

ADD IN
THIS
DIGIT

DONE

SET CARRY BIT
FOR ERROR
FLAG.

PAGE 16 OF 34

RETURN

B1

B1
ATDEC

SET UP RCIS$ PARAMS

RCIS$: ASCII TO REAL

ERROR

Y

N

OUTPUT SINGLE PRECISION DECIMAL

RESET STACK AND MODES

RETURN
GADRS

DIRM
GET COMPOOL NUMBER

ERROR ?

Y

GET COMPOOL BASE ADDRESS

D1

DIRM
GET OFFSET

ATINT
CONVERT TO INTEGER

ERROR ?

D1

COMPOOL BASE PLUS OFFSET = ADDRESS

RETURN

PAGE 19 OF 34
OTDEC

CLEAR OUTPUT BUFFER

DIVIDE NUMBER BY 10.

MAKE ASCII MOVE DIGIT TO BUFFER

2/3RD DIGIT

RETURN

PAGE 20 OF 34
GET NEXT BYTE

ERROR ?

SAVE REPEAT COUNT

ROOM IN TABLE ?

RESET COUNTERS

SET CARRY BIT (ERROR FLAG)
GTIM

FETCH HOUR (BCD) FROM BF COMPOOL

1ST CHAR, MAKE ASCII AND STORE IN MSG

2ND CHAR, MAKE ASCII, STORE DIGIT PLUS COLON.

FETCH MINUTES

SPLIT BCD NUMBERS, MAKE ASCII AND STORE

D ERR
PRINT HEADER

RETURN
DIVIDE BY 10
- RESTORE PREVIOUS VALUE

VALUE MOD 10
= 1 DIGIT OUT

CONVERT TO ASCII AND STORE

MORE?

Y

PUT "E" IN BUFFER, EXPONENT FIELD

SET "-0" IN BUFFER

SET "+0" IN BUFFER

N

EXponent +/−

↑

↓

G4

PAGE 28 OF 34
EPRLMT

ALT\(F = ALT/14.5499.6\)
\(STAT_{\text{pres}} = BARSETO\)
\(((11.9 - ALT\(F\) 12.6) ALT\(F\) - 5.258) ALT\(F\) + 1)\)

BLEED ON?

Y

BASE VALUE:
\(MX_{\text{EPRT}} = 1.94 - 0.00628 TAT\)

ALT > 30000?

N

Y

ALT\(F = (ALT - 30000)/5000)\)
\(MX_{\text{EPRT}} = MX_{\text{EPRT}} - ALT\(F\) (TAT 0.00037 + 0.025)\)

MX\(_{\text{EPRT}} > 2.35\)?

Y

LIMIT: MX\(_{\text{EPRT}} = 2.35\)

N

EPRLMT
(PAGE 1 OF 6)
BASE VALUE:
MXEPRP = 3.514
- (-0.0535 STAT-PRES)

MXEPRP > MXEPRP?

Y

MCLEPR = MXEPRP

N

MCLEPR = MXEPRP

MCLEPR, MCREPR = MXEPRP

ALT > 1500?

Y

N

ALT < 30,000?

Y

N

ALT < 30,000?

Y

N

G

B
EPR LMT
(PAGE 4 OF 6)
\[ E \]

**Decision 1:**
- **TAT \( \leq -20 \)?**
  - **Y**
    - **MXEPRT** = \( 1.8918 - 0.00591 \) TAT
  - **N**

**Decision 2:**
- **TAT > -20 \( \)?**
  - **Y**
    - **MXEPRT** = \( 1.843 - 0.00714 \) TAT
  - **N**

**Decision 3:**
- **MXEPRT \( \geq \) MXEPRT \( ? \)**
  - **Y**
    - **MCERP** = **MXERP**
  - **N**

**Assignment:**
- **MCERP** = **MXERP**

**Flow:**
- **MCERP** = **MCERP**

**Final Decision:**
- **EPRLMT**

*(Page 5 of 6)*
START ERAD

SIM AIRPLANE ENG?

INS NAVIGATION VALID

MAGVAR = INS DIGITAL INPUT (MVINS)

COMP LOCAL N RADIUS OF CURVATURE

COMP LOCAL E RADIUS OF CURVATURE

RM = ALTCOR FTONM + RM
RN = (ALTCOR FTONM +RN) cos(LAT)
RADFT = RM ' NMTFT
DLATFT = RADFT DTOR
DLONFT = RN ' NMTFT DTOR

END ERAD
FAST

SET UP FLOATING POINT EXCEPTION AST

FIRST PASS?

Y

PERFORM INITIALIZATION OF SELECTED VARIABLES

ACTIVATE OSDAR TO ESTABLISH DEFAULT DATA RECORDING

ACTIVATE BACKGROUND TASK - SLOW

DECLARE SIGNIFICANT EVENT CLEAR EVENT FLAG 33

WAIT FOR EVENT FLAG 33 DETECTED START OR 500 USEC LOOP

LABEL 6 SET?

SNAP REQUESTED

SNIPSET

FAST

PAGE 1 OF 8
2A

ABORT FLAG SET?

Y

STOP SLOW AND FAST TASKS
RETURN

N

TOGGLE 100 MSEC LOOP FLAG, UPDATE TIMER

24 HAS EXCEEDED

Y

RESET TIME TO ZERO

N

DISFD

PERFORM DISCRETE FAILURE DETECTION

IOFL1

INPUT AND FORMAT 50 MSEC DATA

MSPLGC

PERFORM MODE SELECT PANEL LOGIC

LABFLG AND MISVAR

Y

COMPUTE X Y Z POSITION FOR PLOTTER

N

RUN MODE?

Y

MLSEX

N

3A

FAST

PAGE 2 OF 8

A-110
ENTRY TIME - ASTTIN1 ≤ 10.

Y

Dump floating point and general purpose registers to console

N

FPEXCP

RESET STACK POINTER

ASTTIM = BIVTIME

ASTX$
CHECK IF AIRCRAFT IS PARKED ON THE GROUND

FDSTR

REPLACE PMSWIT WITH MSWIT FROM LAST FRAME

MSWIT IS A MODE SWITCH THAT REFLECTS WHETHER THE SYSTEM IS IN FLIGHT, PREFLIGHT OR MAINTENANCE MODE.

SQUAT AND NOT WHEELSPEIN?

MSWIT = 0 (FORCE FLIGHT MODE)

CREATE NEW MSWIT FORM SYSTEM TEST PANEL MODE SWITCH INPUT

CALCULATE ADDRESS WHERE NEXT FAILURE IS TO BE STORED IN THE FAILURE DATA TABLE

THE FAILURE DATA TABLE IS AREA RESERVED FOR HOLDING UP TO 32 SENSOR FAILURES UNTIL THEY ARE REQUESTED FOR DISPLAY BY THE OPERATOR.

FAILURE DETECT

PERFORM FAILURE DETECTION PROCEDURE ON INDIVIDUAL 10 MS SENSOR

LAST 10 MS SENSOR CHECKED?

N

1A

FDSTR

A-117
FAILURE DETECT
PERFORM FAILURE DETECTION PROCEDURE ON INDIVIDUAL 50 MS ANALOG SENSOR

LAST 50 MS ANALOG SENSOR CHECKED?

IN PREFLIGHT?

FAILURE DETECT
PERFORM FAILURE DETECTION PROCEDURE ON INDIVIDUAL 50 MS DIGITAL INS SENSOR

LAST 50 MS INS SENSOR CHECKED?

2A
FAILURE DETECT
PERFORM FAILURE DETECTION PROCEDURE ON INDIVIDUAL DISCRETE SENSOR

LAST DISCRETE SENSOR CHECKED?

TRUE
AIRSPEED VALID?

Y

DSTOR
STORE FAILURE INFORMATION IN FAILURE DATA TABLE

TURN ON FAILURE READ LAMP

IN
PRE-ENGAGE?

N

INSERT INTO FAILURE DATA TABLE "SYSTEM RESET" MESSAGE FLAG

CLEAR SEIU AND DATAC LINK FAIL FLAG

3A

FDSTR
(PAGE 3 OF 8)
3A

(1) ANY SEIU COMMUNICATION FAILURES ALREADY LOGGED?

(2) IS WRAPAROUND FROM SIU THE SAME AS THAT SENT LAST FRAME?

(3) THIRD TIME IN A ROW AN ERROR WAS FOUND?

SET SEIU 2ND FAIL FLAG

SEIU POWER FAIL ALREADY RECORDED?

SEIU POWER FAIL DETECTED 3 TIMES?

SET SEIU 2ND FAIL FLAG

DSTOR
STORE "SYSTEM WDT" MESSAGE IN FAILURE DATA TABLE

SET SEIU 2ND FAIL FLAG

DSTOR
STORE POWER FAIL ERROR IN FAILURE DATA TABLE AND TURN ON FAILURE READ LAMP

4A
(1) SEIU A-D CONVERTER FAILED VALUE THREE TIMES IN A ROW?

STORE A-D CONVERTER FAIL MESSAGE IN FAILURE DATA TABLE

TURN ON FAILURE READ LAMP

(2) ANY RFDIU COMMUNICATION FAILURES LOGGED?

BUY

(3) IS WRAPAROUND FROM RFDIU SAME AS THAT SENT LAST FRAME?

THIRD TIME IN A ROW AN ERROR WAS FOUND?

STORE "RFDIU WDT" MESSAGE IN FAILURE DATA TABLE

TURN ON FAILURE READ LAMP

5A
5A

RFDIU POWER FAIL ALREADY LOGGED?

RFDIU POWER FAIL DETECTED THREE TIMES?

STORE POWER FAIL ERROR IN FAILURE DATA TABLE ON FAILURE READ LAMP

DATA LINK TEST INHIBIT FLAG SET?

DATA LINK ERROR FOUND 3 TIMES?

STORE "SSFDF-DATAC LINK" ERROR IN FAILURE DATA TABLE

TURN ON FAILURE READ LAMP AND SET SSDFD2 (2ND FAIL FLAG)

RETURN
THIS PROCESS IS USED IN-LINE AT FOUR POINTS IN FDSR TO CHECK SENSOR STATUS AND RECORD FAILURES IN THE FAILURE DATA TABLE.

FAILURE DETECT

DOES STATUS WORD HAVE A VALID SELECTION INDEX?

SELECTION INDEX FAILURE ALREADY DETECTED?

SET SENSOR 2ND FAIL

INDICATE SENSOR 2ND FAIL AND SET FAIL STORED FLAG

ANY SENSOR FAILURES?

2ND FAIL ALREADY STORED IN FAILURE DATA TABLE?

SENSOR 2ND FAIL?

1ST FAIL ALREADY STORED IN FAILURE DATA TABLE?

CHANNEL A FAILED?

CHANNEL B FAILED?

CHANNEL C FAILED?

INDICATE CH. A FAIL AND SET FAIL STORED FLAG

INDICATE CH. B FAIL AND SET FAIL STORED FLAG

INDICATE CH. C FAIL AND SET FAIL STORED FLAG

FDB

FDB

FDSTR

(PAGE 7 OF 8)
RECORD INDICATED FAILURE IN FAILURE DATA TABLE USING DSTOR

TURN ON FAILURE READ LAMP

END OF PROCESS
FLITEM

GET SYSTEM
TIME AND
FORMATTED IN
MESSAGE BUFFER

DO:
FOREACH OF 6 ERROR
ACUMULATORS:
MS1OM, MS5OM, MS10TP, ES1ERR, ES1ERR, ES1ERR

* EXCEEDED ERROR THRESHOLD
FOR THIS MINUTE?

N
Y

FORMAT
OUTPUT ERROR
MESSAGE

DONE
N
Y

RESET ALL
ERROR COUNTERS

RETURN
FORMAT

PUT ERROR IDENTIFIER IN MESSAGE BUFFER

FORMAT %O OF ERRORS OVER 1 MINUTE SCAN

WRITE OUT TO CHOSEN LINE ON SYSTEM CRT

RETURN
FLTHDL

SET UP POWER FAILURE AND ABORT AST'S ASSIGN UMR'S

INITIALIZE FRAME COUNTER AND FLAGS

SET DEVICE CHARACTERISTICS FOR TT10

POWER FAIL RECOVERY ENTRY POINT

EIU CONNECTED?

INITIALIZE EVENT FLAGS AND ERROR COUNTERS

DISPLAY 'EIU, SIU DATAC DISCONNECTED'

DELAY FOR 30 SECONDS

INIT

INITIALIZE KW11-P AND UP LINK INTER-rupt SERVICE ROUTINES

WAIT FOR EVENT FLAG SET BY 10 MSEC INTERRUPT

DEALLOCATE UMR'S

OUTPUT 'HANDLER TERMINATED' ON CONSOLE

EXIT FLTHDL

ABORT REQUESTED?

1A

2A

FLTHDL
DO CASE (M FRAME)
M FRAME = 0-4

2A

EIU/SIU ERRORS?

INCkaMENT EIU/SIU ERROR COUNTER

INCREMENT 10 MSEC FRAME COUNTER AND M FRAME

50 MSEC ATTN BIT SET?

INCkMENT MISSED 10 MSEC INTERRUPT COUNTER

SET FRAME INDEX TO ZERO

COMPARE M FRAME TO 5?

INCREASENt MISSED 50 MSEC INTERRUPT COUNTER

SET M FRAME = 5 WHICH FORCES CASE 2 SELECT

FLTHDL
(PAGE 2 OF 5)
CASE 0

OUT10M
FORMAT RUDCMD, AILCMD, DECMD FOR OUTPUT

OUTPUT 10 MSEC AND 50 MSEC DATA

DAS DATA READY ?

OUTPUT DAS DATA

INPUT EIU MAJOR FRAME DATA (368 WORDS)

IN10M
FORMAT Q,P,R, HDD + BMACC

CASE 1,2,3

OUT10M
FORMAT RUDCMD, AILCMD, DECMD FOR OUTPUT

OUTPUT 10 MSEC DATA

IN10M
FORMAT Q,P,R, HDD + BMACC

CASE 4

OUT10M
FORMAT RUDCMD, AILCMD, DECMD FOR OUTPUT

OUTPUT 10 MSEC DATA

I/P LINK INTERRUPT RECEIVED ?

ERROR DETECTED ?

A

B

C

I/P LINK READY ?

Y

COMPUTE NO. WORDS TO XFER USING I/P LINK

INITIATE I/P LINK XFER

INPUT 10 MSEC DATA

IN10M
FORMAT Q,P,R, HDD + BMACC

A

B

C

INCREMENT MISSED INTERRUPT COUNTER

A-129

FLTHDL
(PAGE 3 OF 5)
IPISR

PERFORM ERROR CHECKING

DETERMINE WHICH MACHINE BEING USED (PDP-11/70 OR PDP-11/84) AND PROCESS ACCORDINGLY

DETERMINE WHICH DATA TO TRANSFER - DISNAV + ACTIVE OR PGBUF ONLY

SET UP WORD COUNT FOR BURST MODE TRANSFER

ISSUE BURST MODE WRITE COMMAND TO TRANSFER DATA TO NORDEN #1

RETURN
* FIRST PUSh of
FLIGHT PLAN PAGE
(MOD PAGE = 0)

SET
MOD PAGE = 1

PROVISIONAL
MODE? (PROV)

YES
CLEAR
HOME PAGE
FLAG
(FMHPG,F)

NO

SET
FLIGHT PLAN PAGE
WP, PTR. (FPPWP) =
PTRON

SET
HOME PAGE FLAG
(FMHPG)

EMPRPG

FLTMOD
(PAGE 2 OF 19)
CLEAR HOME PAGE FLAG (FMHNPFG)

TEST UP & DOWN ROLL LIMIT
IF DOWN ROLL LIMIT EXCEEDED
DISABLE FURTHER ROLL BY
STORING THE NEGATED WP7 (L-20) IN FLT. PLAN PAGE WP7 PTE. (FPWNP).
IF UP LIMIT EXCEEDED STORE
POINTER TO LAST WAYPOINT
IN FLT. PLAN PAGE WP7 PTE. (FPWNP)
CLEAR 96 WORD NCGW PAGE BUFFER AND SAVE TURN-UPAR POINTS IN PREFERENCE TO SPECIFIED ENDS.

ROLL DOWN WPT?

STORE LAT/LOT OF CENTER WPT. ON NCGW PAGE AS MAP CENTER LAT/LOT.

STORE ORIGIN AND DESTINATION AIRPORT NAMES.

STORE PAGE HEADING FLT. PLAN. # 1 OR FLT. PLAN # 2 DEPENDING ON MARINE.

SET FLT. PLAN PAGE WORD ADDER AND PAGE LINE ADDER.

COMPUTE AND STORE PAGE LINE PTA LEMPT REFORMAL, OF PREVIOUS WPT.

CALCULATE & STORE DATA FOR LINES 2 & 3 (AL discretion. G.S., PTA, FPA).

ORIGINAL PAGE IS OF POOR QUALITY.
A

INCREMENT FLT. PLAN PAGE PTR. (FPWA0R) AND LINE ADDRESS PTR. TO START OF LINE 4 ON PAGE SET PAGE LINE POINTER (EP0TR) TO EPPWP

WPCAL

CALCULATE & STORE WAYPOINT DATA FOR LINES 4 & 5

INCREMENT FLT. PLAN PAGE PTR. (FPWA0R) AND LINE ADDRESS PTR. TO START OF LINE 6 ON PAGE SET PAGE LINE PTR (EP0TR) TO WAYPOINT FOLLOWING EPPWP

WPCAL

CALCULATE & STORE WAYPOINT DATA FOR LINES 6 & 7

IF ACTIVE PATH DOES NOT TERMINATE IN A STAR OR RUNWAY, OUTPUT "LINK GAP" MESSAGE.

B

ORIGINAL PAGE IS OF POOR QUALITY

FLTMOD
(PAGE 5 OF 19)
IF TURN RADIUS RESULTS IN PATH DISCONTINUITY
MOVE "BAD RADIUS"
CUE MESSAGE TO
DISPLAY.
IF CUE MESSAGE
NEEDS TO BE OUTPUT
AS INDICATED BY AN
ADDRESS IN ATCLNG-
OUTPUT CUE MESSAGE.

SET NEWPAGE
(NEWPAGE)FLNAM
CLEAR BUFFER
SEND TCHP
SYSTEM

RETURN
WPCAL

IF THE STARTING ADDRESS POINTER (LMPTR) POINTS BEFORE THE START OF BUFFER OR PAST THE END OF BUFFER RETURN

CTDOS
CONVERT WAYPOINT NAME FROM 7 TO 8 BIT CASE

IF PROVISIONAL WAYPOINT BIT IS SET PLACE A '?' IN COLUMN 1. IF WAYPOINT IS THE 'TO' WAYPOINT PLACE A '✓' IN COLUMN 1 ELSE BLANK COLUMN 1

STORE CONVERTED WAYPOINT NAME

ORIGINAL PAGE IS OF POOR QUALITY

PAGE 1

A-138
IF ALTITUDE IS NOT ZERO AND 'FLIGHT LEVEL' IS SELECTED FORMAT 3 CHARACTER ALTITUDE (IN HUNDREDTHS) PRECEDED BY 'FL'. ELSE FORMAT 5 CHARACTER ALTITUDE (IN THOUSANDS).

MOVE "PTA" TO LOCATION Indicated BY FMLND.

PLANNED TIME OF ARRIVAL ED?

YES

NO

FORMAT PLANNED TIME OF ARRIVAL

WPC4

ORIGINAL PAGE IS OF POOR QUALITY

FLTMOD (PAGE 8 OF 19)
MOVE "REMD" ON LINE #1 PAGE #2

CLEAR ACCUMULATORS FOR REMAINING DISTANCE (CREMID) REMAINING TIME (CREMT)
COMPUTE POINTER FROM CENTER TO CENTER DISTANCE & STORE IN LEGDAD & LEDDAD
CLEAR LAST WAYPOINT FLAG (IFPLSTW)

PATH TERMINATED? YES WPC38

NO

ACCUMULATED REMAINING TIME =
LEG TIME CONVERTED TO MINUTES +
ACCUMULATED REMAINING TIME
SET LAST WAYPOINT FLAG
ACCUMULATED REMAINING DISTANCE =
LEG DISTANCE CONVERTED TO NAUTICAL MILES +
ACCUMULATED REMAINING DISTANCE

INCREMENTS WAYPOINT NAME TO NEXT WAYPOINT

ORIGINAL PAGE IS OF POOR QUALITY

A-140.
DETERMINE IF THE TURN IS A LEFT OR A RIGHT TURN AND MOVE 'TURN L' OR 'TURN R' TO SCREEN

FORMAT
IF IT IS NOT THE END OF BUFFER DETERMINE SIGN OF FLIGHT PATH ANGLE (GAMMA) IF +ve (NOT ZERO) MOVE 'U' -ve MOVE 'L' ROUND GAMMA & STORE WITH DEG("`) SYMBOL

MOVE "POS" TO LOCATION INDICATED BY FMLAD

FORMAT
VELOCITY FORMAT AND STORE VELOCITY IF VELOCITY # 0 OR IF IT IS THE FIRST OR LAST WAYPOINT

RETURN

A-143
FMWPT
CALCULATE WAYPOINT POINTER

IF FLT. LEVEL (AS INDICATED BY LEVEL FLAG, = FLWPT)
SET F.L. BIT IN PWNAM1
ELSE
RESET F.L. BIT IN PWNAM1

STORE ALT/F.L. DATA IN GUID. BUFFER

FMSSM
COMPUTE BEGINNING ADDR. OF WAYPOINT

END OF BUFFER?

YES
FMREP

NO
FMAL
STORE FLIGHT LEVEL OR ALTITUDE IN CASKAL DEPENDING ON INDICATION IN FLWPT

FMREMOTE 14

FMGOS

CLEAR CASKAL GROUND SPEED (CASING)

PROVISION MODE OR LAST WPX ASIGNED

SET PTATIME: PLANNED TIME OF ARRIVAL WPX: WAYPOINT NUMBER

F 17

ORIGINAL PAGE IS OF POOR QUALITY
FMWPT
CALCULATE WAYPOINT POINTERS

END OF BUFFER?

STORE PLANNED TIME OF ARRIVAL 'PTA' IN PTATH.
CALCULATE THE POSITION # OF THE WAYPOINT (OF PLANNED TIME OF ARRIVAL) AND STORE IN PTAPX.

IF PROVISIONAL MODE FLAG IS SET (SETUD) THEN CALCULATE PTA'S AND PLACE IN ACTIVE FLIGHT PLAN ELSE SET FLAG TO EXECUTE PATH DEFINITION

RETURN
A demand message is one that is to be output immediately without any request by the operator.

ICO is a local subroutine used to convert an integer number into ASCII for display.
EXP$OT

CONVERT MACHINE CODE
FLOATING POINT VALUE IN X0
TO ASCII FORMAT STARTING AT ADDRESS IN X4
FORMAT
S. NNNNNNNN
S = +/-
N = NUMFX DIGIT

RETURN

OCT$OT

CONVERT MACHINE INTEGER TO ASCII OCTAL CODE
(6 DIGIT FIELD)
INPUT: X0 (VALUE)
OUTPUT: X4 (ADDRESS)

RETURN

ORIGINAL PAGE IS OF POOR QUALITY
IN PREFLIGHT?

MLS VALID?

BODY MOUNTED ACCELEROMETER USED?

FAILURE?

PITCH ROLL, IATTV?

FLAG MLS 2ND FAIL

(1)* PITCH, HDD, XTKACC, VN, VE, ATKACC, ROLL, GSINS, INAVV, IATTV?

THE FORMAT FOR THE DECISION SHOWN HERE IS USED THRUOUT THIS CHART. IT IMPLIES THAT THE INCLUDED SENSORS ARE TESTED FOR 2ND FAILURES AND IF ANY ARE FOUND, THE FAILURE BRANCH IS CHOSEN.
(1) DCOL, DWML, FLAP, SPL2, SPR7, PEDAL, ATRIM, DRPOS?

SET ALTNV IF 2ND FAIL ON CALTV OR HBC OR HBF AND NOT MLSMOD

(1) FAILURE

PRE-ENGAGE?

IN MANUAL ELECTRIC?

Q, P, CAS, ROLL?

CASV SET?

AFCSV SET?

EXIT

2A

F2CMP

(PAGE 2 OF 5)
SET AUTOThROTTLE FAIL 2 IF NOT TRUE:
ATE AND (ATKACC STATUS OK IF MLSMOD
AND (MCNF AND #10))) AND (((TAS < 150)
AND (CAS STATUS OK) AND CASV) OR
((TAS >= 150) AND TASV))

SET MODE FAIL FOR CURRENT
FLIGHT MODE

SET AUTOThROTTLE FAIL2

Y

F2CMP
(PAGE 4 OF 5)
CHECK FOR AND LOG ANY MODE FAILURES USING FMTMG.

CRSET ?

CLEAR ALL LOGGED MODE FAILS

ERSET ?

CLEAR SENSOR FAILURE STORED TABLE, SEIU ERROR COUNTERS, AND TURN OFF MODE FAIL AND STATUS ALERT LAMP

RETURN
GETDAT

ADD THE INPUT OFFSET TO THE INPUT ADDRESS

MOVE A WORD FROM THE INPUT LOCATION TO THE OUTPUT LOCATION

ALL WORDS MOVED?

NO

YES

RETURN
HNAVSL

RESET FAIL FLAGS F0, F2, F3, DMEER, AND VORER

ASSIGN LOCAL COPIES OF GLOBAL NAVIGATION VARIABLES

ASSIGN SINE AND COSINE OF A/C LAT. AND LON.

TUNPATH

TUNXTK

COMPUTE COMPONENTS OF VECTOR FROM EARTH CENTER TO A/C

RADCAL PROCESSING

A
ADVANCED TRANSPORT OPERATING SYSTEM SOFTWARE UPGRADE

FLIGHT MANAGEMENT / FLIGHT CONTROLS
SOFTWARE DESCRIPTION

Appendix A
Flowcharts

Appendix B
Digital Systems Diagrams
SET APPROPRIATE DMEER BITS UNDER CROSS STATION ERROR CONDITIONS.

SET DELTA GAIN (DELGN) ACCORDING TO FLIGHT SYSTEM MODE.

BULK DATA STATION POINTER NVAD3A = 0?

CALCULATE DISTANCE TO CROSS STATION AND ASSIGN TO CDME3.

CDME3 > 327.67 NM?

SET DMEER

CDME3 = 0?

SET DMEER

COMPUTE DELTA RANGE (DRANGE) BETWEEN CALCULATED AND MEASURED DISTANCES TO CROSS STATION.

PROCEED TO STATION CDME3 SET TO ZERO.
B

DRANGE 
>= 5NM 
? 

Y 

SET DMEER 

N 

N 

C

COMPUTE NORTH AND 
EAST ERROR COMPONENTS 
FOR CROSS STATION 

ALITUDE 
>= GROUND 
RANGE TO 
STATION 
? 

Y 

SET F0 

N 

N
UPDATE F0

DMEER \(= 0\)?

- Y: RESET 15 SEC TIMER TO ZERO
  - FLSTA3\$(2) = OFF

- N: TIMER AT 0?
  - Y: SET TIMER EQUAL TO BINTME
  - N: TIMER EXPIRED?
    - Y: ROLL < 15 DEG?
      - N: D
      - Y: FLSTA3\$(2) = ON
    - N: D
SET PTHSTA AND CLEAR DMEER OF PREVIOUS ERRORS

SET DMEER AND VORER BITS UNDER PATH STATION ERROR CONDITIONS

BULK DATA STATION POINTER NVAD2A = 0 ?

CALCULATED DISTANCE CDME2 AND BEARING CVOR2 TO STATION SET TO ZERO

CRBSC
CALCULATE DISTANCE AND BEARING TO PATH STATION AND ASSIGN TO CDME2 AND CVOR2 RESPECTIVELY

CDME2 > 327.67 NM ?

SET DMEER AND VORER CDME2 = 0

CDME2 >= 150 NM ?

SET VORER

CDME2 = 0 ?

SET DMEER AND VORER

PROCESS STATION #2 DATA

HNAVSL (PAGE 5 OF 20)
COMPUTE DELTA RANGE (DRANGE) BETWEEN CALCULATED AND MEASURED DISTANCES TO PATH STATION

DRANGE >= 5NM ?
  Y  SET DMEER AND VORER
  N

COMPUTE NORTH AND EAST ERROR COMPONENTS FOR PATH STATION

ALTITUDE >= GROUND RANGE TO STATION ?
  Y  SET F2 AND F3
  N

VOR/LD = FALSE ?
  Y  SET VORER
  N

COMPUTE DELTA BEARING (DELBR) BETWEEN CALCULATED AND MEASURED BEARINGS TO PATH STATION

DELBR >= 30 DEG ?
  Y  SET VORER
  N

F

G
DELBR * GROUND RANGE MAGNITUDE >= 5NM?

Y SET VORER

N COMPUTE WEIGHTING FOR PATH STATION

UPDATE F2 AND F3

DMEER = 0 ?

Y RESET 15 SEC TIMER TO ZERO

N SET RETUN2 = 0 AND FLSTA2$2(2) = OFF

TIMER AT 0 ?

Y SET TIMER TO EQUAL BINTME

N ROLL < 15 DEG ?

Y FLSTA2$2(2) = ON

N
F5 = FALSE

ILSZON ?
Y
N

DESTINATION RUNWAY NOT ENTERED YET ?
Y
N

DELTA LAT AND LON TO ANTENNA >= 1.4 DEG ?
Y
N

F5 = TRUE

SET TANGENT OF THE GLIDE SLOPE ANGLE (TANGSA)

ILS UPDATE CALCULATIONS
CALCULATE DISTANCE (RNGLS) AND BEARING (BRGLS) TO ILS SHACK

ILSZON OR F5?

CRBC

SET ILSTMP = FALSE

0.1648 NM < RNGLS < 10 NM?

BRGLS < 60 DEG?

SET ILSTMP = TRUE AND F5 = FALSE

LOCFS AND LOCVLD AND SLOCDV < 1.9 DEG?

F4 = OFF

F4 = OFF

A-168

HNAVSL
(PAGE 9 OF 20)
I F5=TRUE I

COMPUTE SECOND ESTIMATE OF NORTH AND EAST ERROR COMPONENTS FOR ILS SHACK USING GLIDESLOPE DATA

F4 = OFF

F4, F5 = OFF

SET F0 AND F2

F4 = OFF
I SET F3
AND
ZERO THE DELTA POSITION VARIABLES DPN AND DPE

UPDATE POSITION AND VELOCITY DELTAS FOR USE IN FAST LOOP NAVIGATION UPDATES

F0 <> 0 AND F2 <> 0 AND F4 = FALSE

SET F3 AND ZERO THE DELTA POSITION VARIABLES DPN AND DPE

5 MIN PASSED WITHOUT AN UPDATE TO DPN OR DPE

FLADM OR FLRM

ZERO DELTA VELOCITY VARIABLES DVN AND DVE

RETURN
UPDATE DPN AND DPE ACCORDING TO NAVIGATION MODE

SET POSITION AND VELOCITY GAINS (K1P AND K2P) USED IN FAST LOOP UPDATING

SET NAVIGATION MODE NCDU MESSAGE VARIABLES NAVTYPE AND MNAVTY

UPDATE GLOBAL FAIL FLAGS F0G, F2G, AND F3G

RETURN
NAV64K = FALSE?

Y

PTRSTA = PTR2D

NXTPS

N

PTRPS = O?

Y

NVAD2A = PTRPS

SET F2

TUNEPS

TICHEX

SET FLSTA2$(4) = ON

RETURN

N

AUTO MODE (STATION #2)
Q

PTR2D <-> PTRSTA?
Y

PTRSTA = PTR2D

NXTPS

COP < DTG AND PSFAIL = FALSE?
Y

TICHEX

RETURN

N
NXTPS

GUID2D ?

Y

COMPUTE CENTER TO CENTER DISTANCE (COP) OF CURRENT WAYPOINT

PTRPS = 0

SET PTRPS TO ADDRESS OF STATION DEFINED BY THE NEXT WAYPOINT

RETURN

TUNEPS

NVAD2A <> 0?

Y

FETCH FREQUENCY OF STATION TO BE TUNED FROM BULK DATA

FLSTA2 = 0

RETURN
RETMSG

GS > 140 KNOTS AND RETUN2 = O ?

Y

RETUN2 = -1

N

RETURN

TICHEX

ATUNE2 = DME2FQ ?

Y

CLEAR PSTMR

FLSTA$2(3) = OFF

N

FLSTA$2(3) = ON

PSTMR = 0 ?

Y

PSTMR = B INTME

N

RETURN

4 SECONDS PASSED ?

Y

PSTMR = 0

N

PSTMR = 0
I COMPUTE DELTA LAT AND LON BETWEEN STATION AND A/C

I COMPUTE SINES AND COSINES OF STATION LAT AND LON

I COMPUTE COMPONENTS OF VECTOR FROM EARTH CENTER TO STATION

I COMPUTE SLANT RANGE

I COMPUTE GROUND RANGE

I COMPUTE BEARING TO STATION

RETURN
LIMIT WPTR (1) TO 2 OR MORE
HVGUID

COLDST ?

TURN OFF FIRST PASS FLAG (HVGP1) AND ALCFLG CLEAR FWDPTh

END OF PATH? (PATHND)

GUID2D AND NAV64K?

LIMIT WPTR (1) TO 2 OR MORE SET PTR4D + PTR2D TO WPTR (1)

CLEAR 1ST PASS FLAG (HVG1) AND PATHND

SET ADDRESS POINTER FOR POPTCVEC AND UADR

ALCBA,XTK, TKE AND LATSTR = 0

GD3D INITIALIZE 3D VARIABLES & DISTANCE TO MIDTURN

HVGP1 = 0 ?

CALCULATE NOMINAL BANK ANGLE (NOMBA)

TURN ?

LEGSW

NCU VALID (NCUVAL) = ON

RETURN

HVGUID

(A-180)
LEGSW

INITIALIZE TURN DISCRETES AND DTG

INITIALIZE DTOGO; SET FIRST PASS ON (HVGP1)

FIRST PASS?

Y

N

HVG2

GUID3D = ON?

N

Y

GD3D

HVG6

RETURN
IF THE AIRPLANE IS OFF THE PATH (RIGHT OR LEFT) SUCH THAT XTK ≠ 1000 FT.
IS > THE RADIUS OF TURN, FORCE THE TURN COMPLETION BY SETTING AMG = MAGTA

A-184

HVGUID
(PAGE 5 OF 14)
```
DTG1

DTOGO = AMG * RTN (PTR2D)

PWDMA (PTR2D) DME?
  Y  N

DTOGO = DTOGO + PWCCD (PTR2D + 1)

TEND?
  Y  N

DTOGO = DTOGO + PWAO2 (PTR2D) + PWCCD (PTR2D + 1)

DTOGO = DTOGO + PWCCD (PTR2D + 1)

TRALCBA

RETURN
```
HVGUID
(PAGE 8 OF 14)
3A

COMPUTE CROSS CUT LIMIT (PSCRATCH)

COMPUTE LATSTR AND DTOTL

1ST HALF OF DME ARC?

N

Y

TURN OR TEND?

Y

BTOTL = DTOTL + PWCCD (PTR4D+1)

COMPUTE SDC

N

COMPUTE SDC

GUID3D ON?

N

GD3D

HVG6

RETURN
GD3D

INITIALIZE 3D GUIDANCE VARIABLES

PTR3D = PTR4D

COMPUTE HDIS

RETURN
CASE NCDSFC-618

0
PUT "ORIGIN" MESSAGE ON NCDU LINE 8

1
DISPLAY "DESTIN" ON NCDU LINE 8

2
DISPLAY "GMT" ON NCDU LINE 8

3
DISPLAY "BARGET" ON NCDU LINE 8

4
DISPLAY "IAGLID" ON NCDU LINE 8

4B

UNLOCK KEYBOARD FOR DATA ENTRY

10C

INIMOD
(PAGE 4 OF 10)
INITIALIZE
LOOK UP PAGE VARIABLES

STORE ORIGIN AIRPORT NAME
RESET DESTIN VECTOR

GET BSET0 VALUE AND OUTPUT ON NCDU

Y

NAVFLG ON?

N

GET LAT/LON FROM IAS AND STORE IDDLAT/IDDLON

CLEAR SELECT MODE

CLEAR GUIDANCE BUFFERS

PUT 6 IN NCDERR FLAG

AIRPORT FOUND?

N

INITIALIZE SELECT PAGE VARIABLES

INSELH

CLEAR NAV DATA PAGE VARIABLES AND AUTOTUNE FLAGS

IDA

LOOK FOR ORIGIN AIRPORT NAME IN BULK DATA
INMID2

LUARP

LOOK FOR DESTINATION AIRPORT IN BULK DATA

AIRPORT FOUND?

STORE DESTINATION AIRPORT NAME, CLEAR RUNWAY POINTER

Y

N

PUT & IN NCERR FLAG

3C

10A

INIMOD
(PAGE 6 OF 10)
INID3

BLANK ENTRY?

N

Y

BCDTIM

GET TIME SUPPLIED
BY THE DATA ACQUISITION SYSTEM

TIMBIN

CONVERT ASCII TIME
TO FLOATING POINT
SECONDS

PUT TIME
IN "BINTIME"

IOA

INIMOD
(PAGE 7 OF 10)
INID4

ASCBIN
CONVERT INPUT ASCII
BARSET TO BINARY

NEGATIVE
VALUE? Y

VALUE
CHANGED >
[1.5] N

IF VALUE CHANGES
FROM 29.92 BY
MORE THAN 1.5
THEN SET ERROR
FLAG

DISPLAY
VALUE ON
LINE 4

10A

PUT 2 IN
NLDERR
(FORMAT ERROR)

3C
INMOD

(PAGE 9 OF 10)
LOKMOD

PAGE #1
ACTIVE

LOOKP1
FORMAT PAGE
#1 DISPLAY
FOR ICDU

LOOKP2
FORMAT PAGE
#2 DISPLAY FOR
1600X16

RETURN

ORIGINAL PAGE IS
OF POOR QUALITY
LOOKP1

DO:
FOR EACH BOOLEAN:
  VORLOC - LOCFS - DA3BV3
  DN4AVD - GE2VD - C4LY
  IDAVV

* CURRENT STATE OF BOOLEAN IS SAME AS IT WAS LAST TIME PAGE WAS UPDATED

Y

N

STORE "OK" OR "FAIL" OR LOOK-UP TABLE FOR DISPLAY

DONE

Y

N

FIRST ALL ON IS SAME OR ON OR MORE BOOLEANS CHANGED STATE

N

Y

MOVEXEC

MOVE NEXT PAGE TO FULL DISPLAY BUFFER FOR OUTPUT

RETURN

ORIGINAL PAGE IS OF POOR QUALITY
CLEAR 1ST PASS
FLAG (LUMODE, BIT 24)

PG2
ACTIVE
(LOKPGN = 2)

YES

NO

LOOK UP WPT
ALREADY ENTERED (LOKWP2)

YES

NO

RESTORE WPT NAME TO LOKWPT,
LOKWP+1

PG 2
ACTIVE
(LOKPGN = 2)

YES

NO

LOAD POINTER TO LOOK UP PAGE BUFFER

LU2I

LUZOUT

LU2OTR
SRCLRP
CLEAR NCDU PAGE

LOKLN1
FORMAT AND STORE LINE 1 (HEADING) OF PG 2 BUFFER

STORE REMAINDER OF SET UP PAGE IN BUFFER

SET NEW PAGE FLAG (NUPAGE)

REQUEST MFD UPDATE

RETURN
LU2INT

INITIALIZE

PG 2

CLEAR 1ST CELL
OF LOOK UP
BUFFER (LOKBUF)

STORE "WPT"
IN LINE 2
BUFFER (LPL2)

STORE WPT
ADDRESS IN
LOKWPT

LNMST2

STORE NAME
ON PAGE

FORMLL

FORMAT WPT
LAT AND STORE

FORMLL

FORMAT WPT
LON AND STORE

D

A-211
YES

IS

LOOK UP

WPT AN

ARPT

YES

LU2NAV

NO

STORE "a" IN

LOKNPT+1

LU2WVR

LOOK UP MAG
VAR AND STORE
IN L.U. PG
BUFFER

LU2WEL

LOOK UP

ELEVATION AND
STORE IN

L.U. PG

LU2WFO

LOOK UP

FREQUENCY AND
STORE IN

L.U. PG

LU2WPR

LU2WPR

A-212
LU2NAV

IS WPT A NAVAID

YES

STORE "1" IN LOKWPT+1

INCREMENT WPT ADDR

POINTER TO NAVAID FREQ

LU2WFQ

LOOK UP FREQ AND STORE IN L.U. PG BUFFER

INCREMENT FREQ POINTER TO BECOME MAGVAR POINTER

LU2WVR

LOOK UP MAG VAR AND STORE IN L.U. PG BUFFER

INCREMENT POINTER TO BECOME ELEVATION POINTER

LU2WEL

LOOK UP ELEVATION AND STORE IN L.U. PG BUFFER

LU2WPR
LU2WPR

STORE POINTER TO LOOK UP WPT IN LOKWP2

LU2OUT

LU2WFQ

STORE "FREQ" IN LINE 6 BUFFER

LOAD POINTER TO FREQ

FRMFRQ

FORMAT FREQ AND STORE IN LINE 6 BUFFER

RETURN
LU2WVR

STORE "VAR" IN LINE 5 BUFFER

MAG VAR OF WPT

LOAD "W" FOR WEST VARIATION

LOAD "E" FOR EAST VARIATION

STORE IN LINE 5 BUFFER

STORE DEGREE MARK ("\(^{0}\)"") IN LINE 5 BUFFER

LOAD MAG VAR OF WPT

FORMAT

FORMAT AND STORE IN LINE 5 BUFFER

RETURN

A-216
STORE "EL" IN LINE 5 BUFFER

LOAD ELEVATION OF NAVAID OR AIRPORT

FORMAT
FORMATT AND STORE IN LINE 5 BUFFER

RETURN
LOAD INPUTED RUNWAY NUMBER (NCMESG)

LURWY: LOOK UP RUNWAY

ERROR:

NO

STORE POINTER TO RUNWAY DATA IN DECADR

LU2INT

INITIALIZE FOR PAGE 2

STORE RUNWAY DATA POINTER IN LOKBUF

STORE "4" IN LOKWPT+1

E
STORE "RWY" IN LINE 2 BUFFER

STORE "5" IN CHAR COUNT (LN2CHR)

LOOK UP MAP CENTER POSITION

LOAD OUTER MARKER NAME

LOAD RUNWAY LENGTH

FORMAT FOR RUNWAY LENGTH AND STORE IN LINE 2 BUFFER

STORE "HDG 0M" IN LINE 4 BUFFER
LOAD HEADING, CONVERT TO DEG, MAGNETIC

FORMAT
FORMAT HEADING AND STORE LINE 4

STORE "EL" IN LINE 4 BUFFER

LOAD RUNWAY ELEVATION

FORMAT
FORMAT ELEVATION AND STORE LINE 4

STORE "-" IN LINE 5 BUFFER

LOAD GLIDE SLOPE CONVERT TO DEGREES

FORMAT
FORMAT GLIDE SLOPE STORE IN LINE 5 BUFFER

STORE "ILS" IN LINE 5 BUFFER

LOAD ILS FREQUENCY

FORMAT FREQUENCY AND STORE LINE 5

RETURN
LU2RT

LU2INT

INITIALIZE PAGE 2

LOAD "RTE"

STORE IN LINE 2 BUFFER

LU2RNM

STORE RTE/AYW NAME IN LINE 2 BUFFER AND INITIALIZE POINTERS

SET POINTERS TO LOOK UP BUFFER (LOKBUF), ROUTE BUFFER (LRTBFA)
BULK DATA
SET WPT COUNTER AND LOOP COUNTER

LU2A1

SET LU2TYP = 1 FOR RTE/AWY LOOK UP

LOAD 1ST/NEXT WPT POINTER
STORE IN LOOK UP BUFFER & ROUTE BUFFER

NO WPT POINTER = 0

INCREMENT ALL POINTERS

BUFFERS FULL

NO

YES

G

A-221
STORE NUMBER OF WPT'S IN LWPTMX

COMPUTE POINTER TO THE LOCATION IN L.U. RTE BUFFER FOR WPT ON TOP LINE OF PAGE AND STORE IN LRTEBFP

SET LOOP COUNTER FOR 5 WAYPOINTS

SET LINE POINTER TO TOP LINE

SET FIRST WPT FLAG

LOAD WPT INDICATED BY L.U. RTE BUFFER POINTER (LRTEBFP)

PAST THE LAST WPT (PTR = 0)

NO

LOAD LINE POINTER

LNSTR

STORE WPT NAME IN PAGE BUFFER

YES 1ST WPT

NO
CLEAR FIRST WPT FLAG

SET LN2CHR = 26 TO INDICATE ROUTE

STORE MAP CENTER WPT NAME

INCREMENT L.U. RTE BUFFER PTR.

INCREMENT LINE POINTER TO NEXT LINE

5 LINES DONE

END OF WPT LIST IN BUFFER

LU2OUT 5

LAWY1 18

LKEYUP 26
STORt "STAR"

LU2STR

LU2INT

INITIALIZE PAGE #2

LU2RNM

LOOK UP NAME STORE IN LINE 2 BUFF

STORE POINTER TO FIRST WPT

STORE LOOK UP BUFFER POINTER AND L..U. ROUTE BUFFER POINTER

SET LOOP COUNTER FOR 20 WPTS TO LOKBUF/RTEBUF

STORE ZERO IN WPT COUNTER

LSIDU

SET LU2TYP = -1 TO INDICATE SID/STAR

LOAD FIRST/NEXT WPT

J
STORE PTR LOOK UP BUFFER AND L.U. ROUTE BUFFER

END OF SID/STAR YES → LSIDUD-2

NO

LOAD ALTITUDE AND GROUND SPEED ASSIGNED AT WPT AND STORE IN L.U. ROUTE BUF

UPDATE POINTERS TO NEXT WPT POSITIONS

RTE BUFF FULL (20 WPT'S) NO → LSIDI

YES

STORE WPT COUNT IN LWPTMX

COMPUTE POINTER TO FIRST WORD ON NCDU PAGE AND STORE IN LRTBFP

SET TO LOOP FOR SWATS ON SCREEN SET FIRST WPT FLAG = TRUE

END OF BUFFER NO → LSID2

YES → LU2OUT

NO

K

A-225
LOAD POINTER TO LINE 3 BUFFER

LSIDST
FORMAT AND STORE RTE DATA

1ST WRT

YES
CLEAR FLAG

SET LN2CHR TO INDICATE RTE

LNMS2
LOOK UP MAP CTR

INCREMENT POINTER TO NEXT LINE

5TH LINE PROCESSED

NO
LSID2

YES

DOES PAGE CONT

NO
LU2OUT

YES
LKEYUP
LU2EXEC

L.U. PG 2

LOAD LEFT PARENTHESIS SYMBOL (ASCLPN)

PUT BRACKETS AROUND NAME

STORE WPT LAT/LON AS MAP CENTER

STORE "2" IN LOKCEN AND "0" IN CENWPT

IS THERE A L.U. WPT POSIT

STORE WPT POSIT IN CENWPT

LU2OUT
LOKMOD 24/32

LU2UP 4

LOAD "1"

ADD WPT # IN RTE/AWY/STID/STAR (LWPTNO)

= 0

NO

YES

LOAD "1"

STORE IN LWPTNO

LWPTNO > LWPTMX

NO

YES

STORE LWPTMX IN LWPTNO

CLEAR DISPLAY LINES 1-THEY 8

SID/STAR WHAT TYPE LOOK UP WPT

LSIDUD 21

AWY

LAWYUD 18

LUZOUT 5
LKEYUP 19

PUT "KEY-UP" MESSAGE ON LINE BP

RETURN

LNAME2 28

MOVE NAME IN RCOLSC TO RACE AREA POINTED TO BY WCHR

NOTE: CLEAR NAVAID BIT IF NEEDED

RETURN
**LNMSTR**

**MOVE NAME PRINTED TO BY 'ADDRVEC' TO ADDRESS IN 'LN PTR'.**

**NOTE: IF VALID CLEAR NAMEBIT AND MOVE 3 CHARACTERS ELSE MOVE 5 CHARACTERS**

**RETURN**

*ORIGINAL PAGE IS OF POOR QUALITY*
LNMST2

SAVE LAT/LON OF THIS WPT

STORE "?" OR "E" IN NAME SCRATCH ARROW NAME FIELD

PLACE NAME IN BETWEEN DELIMITERS

LNAME2
STORE BUFFER ON PAGE

RETURN
LU2INT

COMCLR:
CLEAR NCDU PAGE

LOKLNI
STORE L.U. PG LINE 1

CLEAR L.U. WPT NO (LWPTNO) AND L.U. TYPE (LU2TYP)

STORE "2" IN ACTIVE L.U. PG NO (LOKPGN)

CLEAR PARAMETERS:
LRTED
LRTEBF
LOKWPT, LOKWPT+1
LOKBUF

RETURN
LOKMOD 31/32

LREJ2

PAGE 2
ACTIVE

NO

YES

LOAD "?"

LNAME2
STORE NAME
AND STATE
SYMBOL

RETURN

A-235

LU2RNM

STORE DECODED
ADDRESS IN
LRREAD

STORE "1" IN
L.U. WPT
NUMBER (LWPTNO)

MOVE NAME
INTO PAGE

RETURN
CLEAR PARAMETERS:
- LOKPGN
- LOKCEN
- CENWPT
- LOKBUF
- LOKWPT

RETURN.

STORE LINE 1 TEXT " - LOOK UP #2"
IN NCDU PAGE BUFFER AND LOOK UP PAGE BUFFER

RETURN

STORE "3"
IN NCDERR

RETURN
Y INITIALIZE
HDCF FILTER
N
INITIALIZE MLS SWITCHES AND RUNWAY PARAMETERS
MLSM = TRUE

CALCULATE HTDZ
H = HTDZ

MLOG 3 : 10
MODEX = 1
PRENG

DELAY > 0 ?
Y
E
N
DELAY = DELAY - 1 MODEX = 1 (PRENG)
F
Y
N
Y
N
Y
N
Y
N
Y
N
Y
N
Y
N
Y
N
Y
N
Y
N

ACWSS
GAS
AUTOS
AUTOE
CLR AUTO FAIL
DISPLAYED FLAG
DETNT COL & WHL IN DETENT (DZNE, 1.0)
IN DETENT
FAIL2 AUTO
MODEX = 6 (AUTO) VCWSE = F

GAE = TRUE AUTOE = FALSE
AUTOE = F VCWSS = T

A-241
LANDR = TRUE

DLSPI < 90.?

T

LANDR = TRUE

F

LOCA = (MLSMOD + LOCVLD)
- LOCE

GSARM = (MLSMOD + GSVLD)
- GSEG

LAND = TRUE

H < 150. + DECRB?

F

MODEX = 7
- LAND

T

LOCE • ONCRS • GSEG • GSTRK?

N

Y

LAND = TRUE

N

MODEX = 6
- AUTO

MODEX = 10
- RLOUT

MODEX = 9
- FLARE

FLARE?

Y

MODEX = 8
- DECRB

N

LANDA = LAND

A-243
Decision Logic:
(1) MLSM \cdot (|AZ| < 2.5)
(2) LOCVLD \cdot (|LOCDEV| < 2.5)
(3) MLSM \cdot (|EL1 - 3| < 0.8)
(4) GSVLD \cdot (|GSDEV| < 0.75)
MLSEX

MLSC

FPF = TRUE
MLSVAL = FALSE

MLSC = FALSE

MCONF$1

FPF

UNPK

UNPACK 5's OF MCONF

VGS-FLG = MCONF$6

RSCON

SET UP ANTENNA CONSTANTS

A

FPF

T

B

F

A-247
Loop for $k$ (3 or 4) signals: $(R, A_2, E_{11}, E_{12})$

- If $CF\text{RUN} > 120$
  - If $Y$
    - Compute OUTLIER_ERR and OUTLIER_VALID
    - $MLS_{LSV} = MLS_{SV}$
  - If $N$
    - $MLS_{LSV} = MLS_{SV}$
    - $MLS_{LSV} = MLS_{SV} = OUTLIER_V$ 

- If $MLS_{LSV}$
  - $T$
    - Update HISTORY $f$ COUNTER
      - If $MLS_{VC} < 115$
        - $T$
          - $PF_{CMC} = T$
        - $F$
          - $PF_{CMC} = T$
    - $F$
      - $MLS_{VC} = 58$
        - $T$
          - $PF_{IC} = T$
        - $F$
          - $IC_{PF} = T$
      - $MLS_{VC} = 1$
        - $T$
          - $IC_{PF} = T$
        - $F$

C

Original page is of poor quality

(Continued on page 5 of 20)
ALL VARIABLES
INDEXED BY
LOOP COUNTER
(I = 1 TO K)

ORIGINAL PAGE IS
OF POOR QUALITY

END LOOP FOR 'K' SIGNALS

A-252 MLSEX
(PAGE 6 OF 20)
ORIGINAL PAGE IS OF POOR QUALITY
\[ S_{\text{HAT}} = \text{MLSS}_p + \alpha \text{PHAT} + \text{TEMP} \]
\[ \text{MLSS}_P = \text{MLSS}\_\text{HAT} + \beta \text{PHAT} + \text{TEMP} \]
\[ \text{MLSS}_p = S_{\text{HAT}} + \text{MLSS}\_\text{HAT} \times \text{DELTAT} \]

\[ \text{ALL VARIABLES INDEXED BY LOOP COUNTER (I) EXCEPT: MISVAL, TEMP} \]

\[ \text{ORIGNAL PAGE IS OF POOR QUALITY} \]

RETURN

(PFILT 2 OF 2)

MLSER
(PAGE 13 OF 20)
XFORM

SCOSD
compute
\sin, \cos (\Delta \psi)

SCOSO
compute
\sin, \cos (\phi)

ECOSL
compute
\sin, \cos (\tau)

compute lambda data
matrix = \textbf{f}(\psi, \phi)
(see code)

\begin{align*}
\text{ANT_VEC} &= \text{LMB} \times \text{ANT_POS} \\
\text{KAP_TRK_FLG} &= F
\end{align*}

\begin{align*}
\text{xyz} &= \text{xyz} + \text{LMB} \times \text{THOUSE + VGAM + VELL} \\
\text{xyz Ek} &= \text{PHISH} - (v) \xi
\end{align*}

RETURN

MLSEX
(PAGE 14 OF 20)
MLS RANGE $\geq$ AZIMUTH VALID?

$PF_{IC} < L$

$PF_{IC} < 2$

SCOSD

COMPUTE

$\sin, \cos(A^2)$

$R_{A2} = \hat{R} + X_{DME} \cos(A^2) - Y_{DME} \sin(A^2)$

$Y_A = -R_{A2} \sin(A^2)$

$X_A = \sqrt{R_{A2}^2 - Y_A^2 - Z_A^2}$

ALTREF

$> 0$

$A$

$Z_A = HRAD - HTDC - ANZVEC*3$

$B$

NOTE: $\sin(-A^2)$ IS USED IN CODE IMPLEMENTATION

A-261

(MLS/1 OF 2)

MLSEX

(PAGE 15 OF 20)
ALIGNED =

ELX = XA - XEL1
ELY = YA - YEL1

ELX = XA - XEL2
ELY = YA - YEL2

TAN
COMPUTE
TAN (EL1)

Z1 = TAN (EL1) \sqrt{{ELX^2 + ELY^2}} + EEL1G

TAN
COMPUTE
TAN (EL2)

Z2 = TAN (EL2) \sqrt{{ELX^2 + ELY^2}} + EEL2G

\[ \text{POS CG} = [\text{TAN Vec}] + \begin{bmatrix} \text{XG} \\ \text{YG} \end{bmatrix} \]

RETURN

ORIGINAL PAGE IS OF POOR QUALITY

(XYZIN 2 OF 2)

A-262 (PAGE 16 OF 20)
\[ S_{TK} = \frac{\sin \text{WINS} - \cos \text{WINS}}{\sin \text{WINS}} \]
\[ C_{TK} = -\frac{\sin \text{WINS} - \cos \text{WINS}}{\sin \text{WINS}} \]
\[ \begin{bmatrix} \text{ACC} \\ \text{HDD} \end{bmatrix} = \begin{bmatrix} \text{ACC} \\ \text{HDD} \end{bmatrix} = \begin{bmatrix} \text{ACC} \\ \text{HDD} \end{bmatrix} \]

ORIGINAL PAGE IS OF POOR QUALITY.
PRINV

\[
\begin{bmatrix}
    x_{AH} \\
    y_{AH} \\
    z_{AH}
\end{bmatrix} = \begin{bmatrix}
    \cos \theta \\
    \sin \theta
\end{bmatrix} \begin{bmatrix}
    x_{IVR} \\
    y_{IVR} \\
    z_{IVR}
\end{bmatrix}
\]

\[E_{LX} = x_{AH} - x_{LX}\]
\[E_{LY} = y_{AH} - y_{LY}\]
\[R_{PRED} = \sqrt{E_{LX}^2 + E_{LY}^2 + z_{AH}^2}\]

\[E_{LX} = \sqrt{x_{AH}^2 - z_{AH}^2}\]
\[E_{LY} = y_{AH} \times (-y_{AH}, z_{AH})\]

\[E_{LX} = \sqrt{x_{AH}^2 - z_{AH}^2}\]
\[E_{LY} = y_{AH} \times (-y_{AH}, z_{AH})\]
\[E_{LZ} = z_{AH} - z_{LZ}\]
\[TEMP = \sqrt{E_{LX}^2 + E_{LY}^2}\]
\[ELZ_{PRED} = \text{ATAN}(E_{LZ}/TEMP)\]

\[ELZ_{F} \quad F\]
\[T\]

\[E_{LX} = x_{AH} - x_{ELZ}\]
\[E_{LY} = y_{AH} - y_{ELZ}\]
\[E_{LZ} = z_{AH} - z_{ELZ}\]
\[TEMP = \sqrt{E_{LY}^2 + E_{LY}^2}\]
\[ELZ_{PRED} = \text{ATAN}(E_{LZ}/TEMP)\]

RETURN

ORIGINAL PAGE IS OF POOR QUALITY

A-266
KNOBER is a local subroutine which increments or decrements the input parameter based on the knob input. It also sets a Boolean corresponding to the knob turned. These Booleans are the following:

RAKNOB - Altitude
RFKNOB - Flight path angle
RTKNOB - Track angle
RCKNOB - Airspeed

LIMIT ALTSUM BETWEEN 0 AND 50,000 FT.

LIMIT FPASUM BETWEEN ±10.0 DEG.
KNOBER
CALCULATE
NEW VALUES FOR
FPASUM

CORRECT TKASUM
VALUE TO WITHIN
0 AND 360 DEG.

KNOBER
CALCULATE
NEW VALUES FOR
IASSUM

LIMIT IASSUM
TO WITHIN
IASREF AND
400

MSP 4

MSP 100

MSP 6

MSP 100
**BCWSA**

**CLEAR BUTTON**
**INPUT DISCRETE WORD**

MSP 100

**PSTTKA = NOT (TKSEL) AND (RTKNOB OR (NOT(D2D) AND PSTTKA))**

**HORARM = GUID2D AND AUTOE AND NOT (LOCE) AND ((NOT (HORARM) AND D2D) OR (NOT (DTKSEL) AND NOT (D2D) AND HORARM))**

**HORPTh = HORARM AND NOT (BCFLAG)**

**TKSEL = AUTOE AND NOT (LOCE) AND NOT (HORPTh)**

**PSTALT = NOT (ALTARM) AND (RAKNOB OR (PSTALT AND NOT (D3D OR DFPSEL)))**

**PSTFPA = NOT (FPASEL) AND (RFKNOB OR (PSTFPA AND NOT (D3D OR DALSEL)))**

MSP 107

**MSPLGC 6/9**
ALTSEL = ALTARM AND (((ABS(DELALT < 1200') AND ALTSEL) OR ((ABS (DELALT < 152') OR ((DELALT >= 0) AND (DELALT < 20' HDCF)) OR ((DELALT < 0) AND (DELALT > 20' HDCF))))

FPASEL = AUTOE AND NOT (GSE) AND NOT (VERPTH OR ALTSEL)

PSTIAS = NOT (IASSEL) AND (RCKNOB OR (NOT (D4D) AND PSTIAS))

TIMARM = AUTOE AND VERARM AND NOT (TO SLOW OR ATDC) AND ((NOT (TIMARM) AND D4D) OR (TIMARM AND NOT (D4D OR DIASEL)))

TIMPTH = TIMARM AND VERPTH AND (NOT TIMPTH OR (TIMPTH AND ATE))

IASSEL = NOT (ATDC) AND (((IASSEL AND NOT (DIASEL OR TIMPTH)) OR (NOT (IASSEL) AND DIASEL) OR (NOT (TIMPTH) AND PTIMPT AND NOT (IASSEL) AND NOT (DIASEL))))

SOUND AURAL MODE REVERSION WARNING IF THERE HAS BEEN AN AUTOMATIC REVERSION TO ANY OF THE FOLLOWING MODES: TKSEL, FPASEL, IASEL, CWS, OR DISENGAGED

ZERO MM
I CLEAR THE FOLLOWING MOMENTARIES:
DASEL, DFPSL, DTKSEL, DIASEL, D2D, D3D, D4D

DACWS ← VCWS OR ACWS OR AUTOS OR GAS

SET THE PAST BOOLEANS FOR NEXT FRAME:
PTKSEL ← TKSEL
PPFSEL ← FPASEL
PIASEL ← IASSEL
PTIMPT ← TIMPTH
PAUTOE ← AUTOE

RETURN
TEST THE FOLLOWING BOOLEANS AND TURN ON THE CORRESPONDING LAMP
IF SET:
PSITKA - PRESELECT TRACK
TKSEL - TRACK ANGLE SELECT
PSITFPA - PRESELECT FLIGHT PATH ANGLE
FPASEL - FLIGHT PATH ANGLE SELECT
PSITALT - PRESELECT ALTITUDE HOLD
ALTSEL - ALTITUDE HOLD SELECT
ALTARM - ALTITUDE HOLD ARMED
PSITIAS - PRESELECT AIRSPEED
ATE AND IASSEL - AIRSPEED SELECT
GUID4D AND TIMPTH - TIME PATH ENGAGED
GUID4D AND TIMARM - TIME PATH ARMED
NOT GUID4D AND TIMPTH - SPEED MODE ENGAGED
NOT GUID4D AND TIMARM - SPEED MODE ARMS

VERPTH?

VERARM?

HER INCREASING?

BLINK VERARM LAMP

TURN ON VERARM LAMP

TURN ON VERPTH LAMP
HORPHT ?
Y

HORARM ?
N

DTOGO INCREASING ?
Y

BLINK HORARM LAMP

TURN ON HORARM LAMP

TURN ON HORPHT LAMP

SCALE AND FORMAT USING BTOBCD THE FOLLOWING SYMBOLS. LOAD EACH FOR OUTPUT TO ITS CORRECT DISPLAY WINDOW.

ALTSUM ---> ALTITUDE DISPLAY
FPASUM ---> FLIGHT PATH ANGLE DISPLAY
TKASUM - MAGVAR ---> TRACK ANGLE DISPLAY
IASSUM ---> AIR SPEED DISPLAY

* BTOBCD CONVERTS A FLOATING POINT NUMBER INTO BINARY CODED DECIMAL.
NCDK 3

COMPUTE NCDU 50 MILLISECOND VALUES.
SEE NCDKEY DESCRIPTION FOR INFORMATION ON THESE
VARIABLES. ALSO SEE
NCDUSO DESCRIPTION FOR
THE VARIOUS ALERT
RECEIvES.

RETURN

KEYERR

Last function entry was erroneous,
reset entry variables and display
one of the following messages on
line #5 of the NCDU.

FORMAT ERROR
FORMAT ERROR PRESS #
NOT IN MEMORY
NOT IN MEMORY CHECK ORIG
NOT IN MEMORY CHECK DEST
NOT IN MEMORY PRESS #
INVALID FUNCTION
INVALID FUNCTION PRESS #
WAYPOINT BUFFER FULL
KEY WAYPOINT ON ANY/RTX
KEY WAYPOINT ON PATH
INVALID THIS WAYPOINT
KEY AIRPORT ON INIT PAGE
KEY RUNWAY ON PATH
BAD RADIUS-xxxx (NAME OF WPT)

RETURN

A-280

NCDKEY
(PAGE 3 OF 4)
FUNCTION_CODE

ACCESS ONE OF THE FOLLOWING INPUT FUNCTIONS:
- UP/ALCH/ATE
- STAFF/AP/FL/6S
- DTA/INIT/FL/SEL/ACT
- NAV/LOOK/EXEC/UP-DOWN
- CLEAR/DEBUG/NOTE

SEE NCDKEY DESCRIPTION OF THESE FUNCTIONS

RETURN
NCDUEX

PROCESS THE STATUS OF LOOK-UP PAGE #1 DISCRETES.

The following discrete must be computed first: VORLOC

The following valids are used to compute the boolean STATUS: VORLOC, LOCS, DME2O, DME3O, GSVD, CALV, INAV.

If any of the discretes is off (FAILED) then STATUS is set true. Otherwise STATUS is false. STATUS is checked by NDCKEY to issue the "LOOK-UP STATUS" message.

NCDU IN TEST MODE

Y

NCDU IN MANUAL MODE

Y

TSTMCB TSTMCB TSTMCB TSTMCB

NCDU TEST PATTERN

Y

NCDU IN AUTOMATIC MODE

Y

2A

3A

* ONE OF 2 PATH DEFINITION FUNCTION IS REQUESTED

NCDU IN AUTOMATIC MODE

Y

NCDU DEBUG PAGE

Y

*
CALL ACTIVE HCDS PAGE ROUTINE

DECODE ENTRY REQUESTED

Y

RETURN

DECODE WAYPOINT ENTRIES

N

CALL ACTIVE HCDS PAGE ROUTINE

SELMOD
Select Page

ATCMOD
ATC Page

FLTMOD
Flight Page

INIMOD
Init Page

LOKMOD
Look-up Page

NVDMOD
Nav data Page

HOLD ABOUT POSITION SELECTED

Y

POSET INSERT UST ON PATH

N

ORIGINAL PAGE IS OF POOR QUALITY
* Guided and IAS BIT of 'from' WPT set and PRVMAP not yet called for current path

```
3A
```

```
* X
```

```
Y
```

```
PRVMAP
```

```
BUILD Prov. MISSED APPROACH PAT
```

```
RETURN
```
During the 10 frame (1/5 second) cycle, new NCDU outputs are inhibited.
On frame #2 the hardware transmit discrete (NCDUMP) is cleared
On frame #10 the cycle active flag is cleared (present)

PUT NCDU in DATA ENTRY mode
CANCEL DATA ENTRY, put out alert message if one needed
START OUTPUT CYCLE (PASS) AND ENABLE HARDWARE TRANSMISSION (NCDUMP)

RETURN
A

USE DECNUM VALUE NAV1 y2

OBTAIN ADDRESS OF ROUTINE TO BE PROCESSED.

DECLAT
LINE 1
LATITUDE

DECLON
LINE 2
LONGITUDE

NAVNO3

LINE 3: NAVAD5
LINE 4: NAVAD2
LINE 5: NEXT NAV1 3

NAVNO2

INPUT BETWEEN 1-5

NO NAVNO 4

USE INPUT #2 TABLE (NAV1 Y3) TO OBTAIN POINTER TO CUR MESSAGE. SEND CUR MESSAGE

LOCK KEYPAD. (SET KEYPAD) SET NEWPAGE-FLAG. (NUPAGE) CLEAR INPUT FROM NCOL. (NCOLPC)

RETURN
Format time error after limiting it to 59 m and 59 sec. Insert + or - ve sign after formatting depending on time error sign.

3D Guidance

Blank altitude error

Format altitude error after limiting it to 32000 ft. Insert + or - ve sign after formatting depending on altitude error sign.

A/P or time box complete path?

4D guidance possible

Blank time to capture

Format time to capture after limiting it to 59 m and 59 sec.

Original page is of poor quality.

NVDMOD
(PAGE 6 OF 17)
IF OFFSET OPTION CHOSEN LOAD CHARACTERS "OFF" ELSE BLANK OUT CHARACTERS TO BE SENT TO DISPLAY
(PAGE 1 LINE 8)

HOLD MODE

STORE "HOLD" ON PAGE 1 LINE 8

MOVE TO THE "HOLD" AREA ON LINE 8

FETCH & STORE POINTERS TO "FROM" & "TO" WAYPOINTS
IF HOLD MODE NOT SELECTED MOVE "FROM AND TO" WAYPOINT TO LINE # 8 ELSE MOVE ONLY "TO" WAYPOINT TO LINE # 8

NAVIGATE

FORMAT NAV MESSAGE ONTO LINE # 8 OF PAGES 1, 2 AND 3.

RETURN

NAVIGATION

(PAGE 10 OF 17)
FORMAT ROUNDED DIFFERENCE OF TRUE HEADING & MAGNETIC VARIATION (HGTTRU - MAGVAR)

FORMAT ROUNDED TRUE HEADING (HGTTRU)

FORMAT ROUNDED WIND DIRECTION

FORMAT WIND SPEED

FORMAT DRIFT ANGLE, UPDATE 'L' OR 'R' CHARACTER DEPENDING ON SIGN OF DRIFT ANG

FORMAT ALONG TRACK ACCELERATION (VGSDDT) WHICH HAS BEEN CONVERTED FROM KTS TO FPS, INSERT +VE OR -VE SIGN DEPENDING ON SIGN OF VGSDDT
AIDCAL

NAV AID

Pointed to

CLEAR LINE ON PAGE 3 BUFFER

RETURN

FETCH NAV AID NAME
CONVERT TO EBCDIC
MOVE TO NAV DATA PAGE #3

RETURN

FORMAT AND STORE FREQUENCY ON NAV DATA PAGE #3
The need for a "SYSTEM RESET" message is indicated by a boolean true in the null word (word 8) of the eight words stored for each message in the failure data table.
PD2

NEXT WAYPOINT ONE ARC TYPE

2B N

SRVEC COMPUTE TURN CENTER VECTOR

STORE AT CURRENT WAYPOINT AND WAYPOINT + 1

SRRTN COMPUTE RADIUS OF TURN

COMPUTE LAT(IN), LONG(IN)
LAT(OUT), LONG(OUT)

CLEAR FOLLOWING VARIABLES:
PWTA, PWA02, PWOTT, PWPPD

CLEAR VARIABLES FOR OUTBOUND WAYPOINT

2A
NOTE:
At this point the buffers are set such that the total DME arc is attached to the in waypoint. The out waypoint contains only the end of arc lat, long and the CCD for elapsed time calculations.
Case ArcSkip+1

1. PD3A: Calculate TA, AO2, DT1, and TC
2. PD5A: Calculate Gamma and Delta Time
3. PD4: Calculate PPD and CED
4. PD3A: Calculate TA, AO2, DT1 and TC

Else:
ArcSkip = 3

PD5A: Calculate Gamma and Delta Time

ArcSkip > 0

Yes:
ArcSkip = ArcSkip - 1

No:

NXWP = NXWP + 1
CWPV = CWPV + 1

Increment Scalar Buffer Pointer and Vector Buffer Pointer

RETURN

Path DF
(PAGE 5 OF 11)
PD3A

VECTOR BUFFER POINTER > 1

N

Y

COMPUTE AD2(I), HALF ARC, DTT(I)
TANG TO WPT TA(I), TURN ANGLE

COMPUTE TC2(I)

PD4
CALCULATE PPD AND CCD

RETURN

A-312
PATHDF
(PAGE 6 OF 11)
PD4

COMPUTE PPD(x)
POINT TO POINT
DISTANCE

TANGENT
OVERLAY

Y

SET BAD
RADIUS MSG FOR
OUTPUT ON NCDU

N

COMPUTE

CCD(x) = PPD(x)
-DDT(x) +
AO2(x)

N

DME ARC

Y

CCD(x) = CCD(x)
-DDT(x) + AO2(x)

PD5A
CALCULATE
GAMMA AND
DELTA TIME

RETURN

PATHDF
(PAGE 7 OF 11)
PDSA

COMPUTE GAMA
(FLIGHT PATH ANGLE)

CORRCE ELSAED TIME
BETWEEN WAYPOINTS
(DT(2) - DT(1)) / 
(ABS(DT(2) - DT(1)))

LASTALT

ALTGSX

CHECK ALTITUDE
AND GROUND SPEED
AT EACH WAYPOINT
AND SET PATH INCOM-
COMPLETE MESSAGE IF
A ZERO VALUE FOUND,
DISABLE GP AND HD
GUIDANCE

RETURN

ORIGINAL PAGE IS
OF POOR QUALITY

PATHDF
(PAGE 8 OF 11)
PDEND

LASTPASS = ON
CLEAR TA(i), A02(i)
DTT(i) FOR LAST WAYPOINT

NXWP > 2

SET ALTGSF

PD4
CALCULATE PPD AND CCD

RETURN
CALL SREV (LAT, LON, ASSIGN (WPT))

E:G

ARE LAT, LON, ASSIGN (WPT)

ACKNOWLEDGES TO THIS ROUTING

GET SIN, COS

GET LON, COS

SREV

GOTO

RETURN
\( \tan 15^\circ \times \text{nominal acceleration of gravity} \)

\( (8.62097522) \)
RGYRO

YES

MAN A

SHUT IT OR

TEST COMPLETE?

NO

EXACTLY
ONE SECOND

SINCE START?

YES

NO

SET SECOND
FAIL FOR

α (PITCH RATE)

SET SECOND
FAIL FOR

ρ (ROLL RATE)

SET SECOND
FAIL FOR

ρ (YAW RATE)

REMOVE RATE
GYRO STIMULUS

EXACTLY
TWO SECONDS

SINCE START?

YES

NO

| IQ | \leq 0.7 |

EXIT

1 A

RGYRO

(PAGE 1 OF 2)

A-321

ORIGINAL PAGE IS
OF POOR QUALITY
EXIT

1A

Is \(|p| \leq 0.7\) ?

No : SET SECOND FAIL FOR ROLL RATE

Is \(|r| \leq 0.7\) ?

No : SET SECOND FAIL FOR YAW RATE

RETURN

RGYRO
(PAGE 2 OF 2)
CLEAR VNAV OPTIONS AND ERASE 'SELECTED' FROM SCREEN

INVALID FUNCTION PRESENT

FUNCTION 1, 2 OR 3

SET NEW VNAV AND PLACE 'SELECTED' ON FREQUENCY

QA
7A

WPT ENTERED A "POS"?

Y

REQUEST INSERTION OF CURRENT POSITION AS WPT IN GUIDANCE BUFFER

N

LUGUID SEARCH BUFFER FOR WPT

(NEGATIVE MATCH)

FOUND

N

ERROR 11

RETURN

Y

7B

PORT WPT NAME ON
LINE 41 PRECEDED BY
"?" AND FOLLOWED BY "/".

APPROX "L-CRLF" ON
NCU LINE 48 AND SET
UP SO SELECT FUNCTION
WILL BE "1" AFTER RE-
SPONSE HAS BEEN ENTER-
TED. UNLOCK NCU KEY-
BOARD FOR TEXT ENTRY.

7B

ORIGINAL PAGE IS
OF POOR QUALITY
* WAITING FOR OPTION #1 BEARING ENTRY

* N

Y

DEFAULT TO RIGHT HAND HOLDING PATTERN AND USE THE PATH HEADING AT THE HOLD WAYPOINT AS THE BEARING FOR THE PATTERN

10B

SM4

CLEAR SELECT PAGE LINE ACTIVE FLAGS

2A

SM1
*PHASE 1: HOLD WP NAME JUST ENTERED
PHASE 2: HOLD DIRECTION/BEARING JUST ENTERED

10A

PHASE 1

Y

V

NULL ENTRY

N

QA

Y

ASCBIN

DECODE

INPUT

VALUE

CAUSE DECODE
TO LOOK-UP
WP IN BULK
DATA, SET
SMQGI TO FOR-
CE "L-ADER"
PROMPT NEXT
PASS

RETURN

N

FORMAT

FORMAT

OFFLINE ON;

NCOD LINE

10C

PROMPT WITH

"EXEC OR RET"
ON NCOD LINE

10B

FORMAT

OFFLINE ON;

NCOD LINE

2B

SMLB2

ORIGINAL PAGE IS
IN POOR QUALITY
FLOW CHART:

1. ASCBIN
   - Decode input offset

2. If legal number is not valid, go to error number 2 and return.

3. Store offset and format direction & bias to NMR display buffer.
   - Clear selected boolean until "EXEC PRESSED".

4. Return to initial state.

(ORIGINAL PAGE IS OF POOR QUALITY)
IN RT: FIND WF - 1 + 2
RADIAL SPLASH
REMOV - BEARING SYMBOL
FILL NWK DISPLAY WITH WF NAME AND BEARING

12A

LUGUID
SEARCH FOR WF ON PATH

12B

FOUNO

ERROR : 11
RETURN

SAVE 'TO' WF INDEX AND GUIDANCE AND POSSIBLE FLAGS FOR RE-MOVE
AFTER NEW 'TO' WF IS EXECUTED. DISPLAY WF NAME PRECEDED BY '?'
ON LINE 6 OF NCDU

10C
To Play ALREADY

SET a SECOND
UPDATE FLAG/ ENABLE NUMERIC
FUNCTIONS/ CUE
:"CHECK AIRbled"

1 SECOND
SINCE LAST UPDATE

RETURN

FORMAT AIRbled (ON/OFF)
FORMAT ACTEDR, DCLFR, AND LCEFR LIMIT VALUES
TO DISPLAY ALONG WITH WHICH IS CURRENTLY
SELECTED. FORMAT ENGINE PRESSURE RATIOS
TO DISPLAY.

2A
ALWAYS SET SENSOR IN USE FOR: HDD, CAS, PITCH, ROLL, MACH, HBARO, TAS

INAV 2ND FAIL?

SET SENSOR IN USE FOR: XTKACC, ATKACC, VNINS, VEINS, THDG, INSLAT, INSLON, INSMV

PRE-ENGAGE?

SET SENSOR IN USE FOR: FLAP, SPL2, SPR7, DRPOS

MANUAL-ELECTRIC?

SET SENSOR IN USE FOR: Q, P

FORWARD FLIGHT DECK OWS?

SET SENSOR IN USE FOR: FWHL AND FCOL

SET SENSOR IN USE FOR: DCOL, DWHL, PEDEL, ATRIM

1A

SINUSE

(PAGE 1 OF 3)
1A

MLSMOD ?

Y

FLARE ?

Y

VBS ?

N

SET SENSOR IN USE FOR GSDEV

LBS ?

N

SET SENSOR IN USE FOR LOCDEV

CXRSP AND NOT LOCE ?

Y

SET SENSOR IN USE FOR XTVEL

NEXT

NEXT 3
N
I SET SENSOR
IN USE FOR HDOTB

I s_s_.so_, I
USE FOR GSINS

SET SENSOR IN
USE FOR HRAD

PLACE SENSOR IN USE INFORMATION
WITH CRSET AND INSST INTO FOUR
WORDS SINUS0,SINUS1,SINUS2
SINUS3 FOR USE BY SSFD

RETURN
SLOW

SET UP FLOATING POINT EXCEPTION AST

LABELG

Y

BREAKPOINT REQUESTED

N

BREAKPOINT IN ROOT SEGMENT

N

MAP

N

SNIPSET

2A

NOTE: SLOW WILL BE ABORTED VIA AN 'IDT' EXECUTION IF AN ERROR IS DETECTED IN THE MRTRC EXECUTIVE REQUEST

MAP OVERLAY SEGMENT IN WHICH BREAKPOINT IS TO BE INSERTED

SET BREAKPOINT INSTRUCTION
Determine #10 mSec Frames Used Last Pass

Simulated A/P On?

N

MAP

SMINIT

NCDUX

HNAVSL

ERAD

Y

Map Appropriate Overlay Segment for Simulated A/P Initialization

Initialize A/P

Call NCDU Executive

Perform Slow Loop Navigation

Perform Earth's Radius Computations

Page 2 of 4
PERFORM WIND SPEED/DIRECTION COMPUTATIONS

COMPUTE ENGINE PRESSURE RATIO LIMITS

CALL ERROR MONITOR ONCE PER MINUTE

MAP APPROPRIATE SEGMENT FOR ERROR MONITORING

PERFORM ERROR REPORTING

I/O ACTIVE?

YES

MAP APPROPRIATE SEGMENT FOR SNAPSHOT

NO

SNAPSHOT DATA TO PRINT?

NO

YES

PAGE 3 OF 4
SNPUT
PRINT SNAPSHOT DATA

SYSTEM
TEST PANEL
MESSAGE?

Y

MAP
MAP APPROPRIATE SEGMENT
FOR TEST PANEL

GMSG
SEND MESSAGE TO PANEL

PAGE 4 OF 4

A-344
SNPOUT

INCREMENT & REPLACE RPTR (modulo 4)

CALCULATE SELECTED SNAP DATA ADDRESS

FETCH SNAP NUMBER FOR HEADER MSG

UNPACK TIME? STORE IN HDR MSG.

FETCH SNAP CRITERIA BLOCK (SCRIT)

SET COUNTER AT 5 ENTRIES PER LINE

F/P DATA? Y

INSERT SPACES FOR SHORT OUTPUT

CONVERT DATA TO ASCII FOR OUTPUT

PAGE 1 OF 3

A-350
FETCH FLOATING POINT DATA

+/−?

+−

PUSH "−"

LIMIT INTEGER PORTION TO 5 DIGITS

OVERFLOW?

Y

PRECEDE OUTPUT WITH "??"

1st DIGIT?

Y

POP SIGN

E 3

PAGE 2 OF 3
SET PTAFLG OFF
SET GUIDANCE 4D P0SSIBLE (PD64D = -1)

4B

I ≤ PWNUM

Y

INITIALIZE TIME BOX POINTER TO 2ND WPT

POINT TO NEXT WPT (I = I + 1)

N

GINTIME < PTA(I)

STARTING AT 2ND WPT
FIND FIRST PTA > CURRENT TIME AND STORE IT IN GINTIME FOR PTA, FOR TO WPT FOR TIME BOX

PTAxFG

ON

N

PTAxFG = OFF
PTAPOS = PWNUM
GINTIME = PTA(PWNUM)

SET VALID TIME FLAG
TERMINATE LOOP (I = PWNUM + 1)

Y

VALTIME ON

TOTIME = 0
TBOX WPT PTR = 1

RETURN

SRPTA 4/4
SRVCK

YES

IS MOTION INHIBITED OR TEST CANCELLED?

LESS THAN 3 SEC SINCE START TEST?

NO

EXACTLY 3 SEC SINCE START TEST?

LATAX
EXECUTE LATERAL AXIS TEST

ELEV and H2STB
EXECUTE LONGITUDINAL TEST

APEUD
EXECUTE RUDDER SERVO TEST

ALL SERVO TESTS COMPLETE?

NO

RECTIFY GROUND TEST SERVO BIAS USING CLRIS

REMOVE GROUND TEST SERVO BIAS USING CLRIS

RETURN

SRVCK
(PAGE 1 OF 6)

A-357

ORIgINAL PAGE IS
OF POOR QUALITY
HZSTB

TEST COMPLETE?

YES

TRIM RATE TEST COMPLETE?

YES

NO

FIRST PASS?

NO

SAVE CURRENT STAB POSITION

STAB POSITION 2.4 < 0.7?

NO

COMMAND NOSE DOWN

COMMAND NOSE UP

COMMAND

NO

223 SEG 2.11

YES

\[ \Delta \text{STAB} = \text{CURRENT \ STAB} - \text{FIRST \ FLAP} \]

FLAP POSITION 22.61?

YES

1A

1B

1C

1D

A-360

ORIGINAL PAGE IS
OF POOR QUALITY
1A

\[ \text{DISTRAIL} = 0.6 \pm 0.1^\circ \]

1B

\[ \text{DISTRAIL} = 3.6 \pm 0.7^\circ \]

1C

LOG "ASRP FAIL"
MESSAGE WSHS DSTOR

1D

RE-CENTER STABILIZER TO 4.5 UNITS

RETURN
FIRST PASS FLAG ON INDICATES INITIALIZATION HAS BEEN COMPLETED

TAHID

FIRST PASS FLAG (TGH) ON

IAVS

RETURN

SEP. DISTANCE (SEP) =
0 FIRST PASS Flag (TGH) = OFF

RETURN

SET TURN (TURN) AND MID-TURN (MID) OFF.
DISTANCE MADE GOOD (DGM) = 0

IF TIME BOX MOV PTRA = 2,
SET = 2

INITIALIZE 4D ARRAY VARIABLES
THRE, PHYTHRE, TASKPTR, TOWPTR

COMPUTE DISTANCE MADE GOOD
FOR TIME BOX AND A/P DEPENDING
ON WHETHER A/I IS AHEAD OF
A/P OR VICE VERSA

SET TIME GUIDANCE FIRST
PASS FLAG ON (TGH)

COMPUTE INITIAL VALUES FOR TIME
BOX ACCELERATION (SDDC), VELOCITY
(SDVC), POSITION
(SDVC)

TGHID
TURN INITIALIZATION

N

"TO" WITH DIR ACROSS HORIZ?

Y

INCREMENT VELOCITY, ACCELERATION, VELOCITY, POSITION

SET MAGNITUDE OF TURN ANGLE (NANOG), AMB = 0, TURNON SET RADIUS OF TURN

AAT

RETURN

TGUID
PAGE 5 OF 7

A-367
FLIGHT DIRECTOR
SPEED COMMAND

SET DISTANCE TO GO FOR TBOX
(DT001) = POSITION (52)

TD* WPT DM*E ARC INBOUND?

RECOMPUTE DT001

SET SEPARATION DISTANCE (SEPR) BETWEEN AJP AND TBOX DEPENDING ON WHETHER OR NOT "TO" WPT IS DM*E ARC INBOUND

COMPUTE SCMD (SPEED COMMAND)
SET TOSLOW OFF

IF SCMD ≤ 0 AND ANGLE OF ATTACK E16º OR CAS < CASREF SET TOSLOW ON

TOSLOW ON

SET SCMD BASED ON CAS LIMIT OR MAIN LIMIT

RETURN

*SEE TGUID DSV
PAGE 8-35

TGUID
PAGE 7 OF 7
ZONELM = 40M
RZNCTR = 0

FETCH FIRST LONGITUDINAL STRIP INDEX IN BULK DATA

END OF STRIPS IN BULK DATA?

RETURN

A/C LON WITHIN STRIP BOUNDS?

BUMP TO NEXT STRIP INDEX

TUN6A
SAVE ZONE POINTER RZNPTR

SET RTNSCR TO CURRENT POINTER ADDRESS

SET SEARCH IN PROGRESS FLAG RINPFL

SET STATION COUNTER RSTCTR = 1

TUN7
CLEAR BAD, SAVE STATION ADDRESS, AND FETCH STATION LAT/LON

COMPUTE ΔLAT, ΔLON BETWEEN STATION AND A/C

COMPUTE RANGE TO STATION FROM A/C

RANGE < ZONELM?

SET BAD

BAD SET

TUN8

STLOOP
STLOOP

STATION IN RTNPRV ARRAY. (PAST USAGE)?

ENTER STATION INTO RTNPRV USING RSTCTR AS THE INDEX.

SET NVAD2A EQUAL TO STATION ADDRESS

INCREMENT RSTCTR

RSTCTR = 5?

RETURN
TUN8A

CLEAR ZONE COUNTER RZNCTR

RESET RTNSCR TO START OF ORIGINAL STRIP

INCREMENT ZONELM BY 40 MILES

ZONELM > 200 M ?

Y

SET ZONELM = 40M, CLEAR RINPFL, AND SET RSTCTR = 1

RETURN

N
BUMP TO NEW STRIP INDEX (5)

BEYOND BEGINNING OF BUFFER?

ASSIGN ADDRESS OF NEW NAVAID STRIP TO RTNSCR

RETURN
ASSIGN ADDRESS OF NEW NAVAID STRIP TO RTNSCR

RETURN
END OF INDEX BLOCK?

YES

SET ZONE COUNTER RZNCTR = 3

NO

ASSIGN ADDRESS OF NEW NAVAID STRIP TO RTNSCR

RETURN

TUN11
TUNEXS

FETCH CROSS STATION FREQUENCY AND ASSIGN TO ATUNE 3

CLEAR FLSTA 3 FLAG

RETURN
RESET ZONE COUNTER

SET TUNSCR TO BEGINNING OF FIRST STRIP

EXTEND ZONE RANGE LIMIT BY 40 MILES

LIMIT ≥ MAX RANGE?

RESET STATION COUNTER AND ZONE LIMIT

RETURN
TUN 12

DME 2 GOOD?

Y

TUNCK

GOOD GEOMETRY AND RANGE?

Y

T2STA

N

SET FLSTA 3 FLAG

RETURN

N

T2STA
FETCH LONGITUDINAL ZONE BOUNDARY

BOUNDARY ≠ 0?
  Y
  RETURN

INSIDE LEFT BOUNDARY?
  Y
  INSIDE RIGHT BOUNDARY?
    Y
    FETCH NAVAID POINTER FOR NEW STRIP
    SET STATION SEARCH IN PROGRESS FLAG CINPFL
    INITIALIZE STATION COUNTER
    TUN7
  N
  GO TO NEXT STRIP

N
TUN6
ASSEMBLE PARAMETERS FOR STATION GEOMETRY CHECK

TUNCK

GOOD GEOMETRY?

SAME AS STATION #2?

SAME AS STATION #3?

HAS STATION BEEN SELECTED BEFORE?

ASSIGN NVAD3A AND CLEAR OR INCREMENT CROSS STATION COUNTER

TUN8

T2STA
TUN10

CALCULATE NEW STATION POINTER FOR STRIP 5

BEYOND BEGINNING OF BUFFER?

Y

N

TUN8A

ASSIGN NEW NAVAID STRIP ADDRESS TO TUNSCR

RETURN
CALCULATE NEW STATION POINTER FOR STRIP 4

BEYOND START OF BUFFER?

Y

TUN8A

N

ASSIGN NEW NAVAID STRIP ADDRESS TO TUNSCR

RETURN
SET ZONE COUNTER = 3

END OF INDEX BLOCK?

Y

N

TUN11A

TUN11

ASSIGN NEW NAVAID STRIP ADDRESS TO TUNSCR

RETURN
<table>
<thead>
<tr>
<th>Digital Systems Diagrams</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATHCL</td>
</tr>
<tr>
<td>CMPF</td>
</tr>
<tr>
<td>DSPOT</td>
</tr>
<tr>
<td>ELEV</td>
</tr>
<tr>
<td>FRCWS</td>
</tr>
<tr>
<td>HNAVFS</td>
</tr>
<tr>
<td>HNAVVB</td>
</tr>
<tr>
<td>HNAVML</td>
</tr>
<tr>
<td>HVG6</td>
</tr>
<tr>
<td>HXFCL</td>
</tr>
<tr>
<td>LATRL</td>
</tr>
<tr>
<td>MLOG</td>
</tr>
<tr>
<td>MLSEX</td>
</tr>
<tr>
<td>NCFM</td>
</tr>
<tr>
<td>PAFD</td>
</tr>
<tr>
<td>PAL</td>
</tr>
<tr>
<td>PFFD</td>
</tr>
<tr>
<td>PVPC</td>
</tr>
<tr>
<td>RCOM</td>
</tr>
<tr>
<td>STABT</td>
</tr>
<tr>
<td>TGUID</td>
</tr>
<tr>
<td>TRALCBA</td>
</tr>
<tr>
<td>VTFCL</td>
</tr>
</tbody>
</table>
CMPF PROCEDURE
COMPLEMENTARY FILTER
ELEV P PROCEDURE

DEPOS ELEVATOR POSITION
(Nose Down) Deg.

ELEV PROCEDURE
ELEV P PROCEDE
MODE DETERMINATION AND INITIALIZATION:

INS MODE (INAVV) | AIR DATA OR RADIO MODE (FLYFLG + INAVV) | SIMULATED AIRPLANE (FLYFLG)

<table>
<thead>
<tr>
<th>ATKACC</th>
<th>ATKACC • MCONF(15)</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>XTKACC</td>
<td>XTKACC • MCONF(15)</td>
<td>IDXTK COMPUTED BY NAVIG</td>
</tr>
<tr>
<td>THDG</td>
<td>MAGVAR + MAGHOG</td>
<td>SMUDG</td>
</tr>
<tr>
<td>VEINS</td>
<td>(TASV • SIN(HDSTRO)) TASV ELSE = 0</td>
<td>CAS • SIN(HDSTRO)</td>
</tr>
<tr>
<td>VINS</td>
<td>(TASV • COS(HDSTRO)) TASV ELSE = 0</td>
<td>CAS • COS(HDSTRO)</td>
</tr>
<tr>
<td>OFF</td>
<td>TASV</td>
<td>OFF</td>
</tr>
<tr>
<td>OFF</td>
<td>TASV</td>
<td>OFF</td>
</tr>
<tr>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
</tr>
</tbody>
</table>

W MCONF(15): EXPERIMENTAL SWITCH
SET = BODY MOUNTED ACCELEROMETER IN USE FOR ATKACC/XTKACC
MLSVD = MLSVAL RUNM
MLSMD = FLYFLG MLSLI MLSVD (DESTINS(4)/O) (ANTLAT/O)
LATCHED ONCE SET
CLEARED BY FLYFLG + MLSLI = MLSVAL
NAVFLG = COLOST (GS > 4)
LIMIT = ON, IF NAVFLG
OFF, IF COLDST
LIMIT OTHERWISE

PROCEDURE HNAVFS
**PROCEDURE HNAVFS (HNAVFS)**

**UPDATE CALCULATIONS.** Following signal selection process, HNAV is called unconditionally. Procedure HAV is next called if MLSM0 is true.
LATITUDE UPDATE CALCULATIONS:

PROCEDURE HNAVFB (HNAVFS)
LONGITUDE UPDATE CALCULATIONS:

PROCEDURE HNAV (HNAVFS)

GROUND SPEED UPDATE CALCULATIONS:
VERTICAL UPDATE CALCULATIONS:

PROCEDURE HNAVB (HNAVFS)

PROCEDURE NAVFLG (HNAVFS)

PAGE 4 OF 4
(HNAVFS PAGE 3 OF 11)
PROCEDURE HNAVML (HNAVFS)

(HNAVML IS CALLED AFTER HNAVB, IF MLSMOD IS TRUE)
(HNAVFS PAGE 6 OF 11)
PROCEDURE HNAVML (HNAVFS)

PAGE 2 OF 4

(HNAVFS PAGE 7 OF 11)
PROCEDURE HNAVVML (HNAVFS)

PAGE 3 OF 4
(HNAVFS PAGE 8 OF 11)
OUTPUT SELECTION: SELECT HNAVML OR HNAVB OUTPUTS AS SOURCES AND PERFORM FINAL CALCULATIONS

PROCEDURE HNAVFS
MLOG PROCEDURE
(LAND COMPUTATIONS)
PAGE 1 OF 2
MLOG PROCEDURE
(LAND COMPUTATIONS)
PAGE 2 OF 2
HORIZONTAL MODE CALCULATIONS:

PROCEDURE NCFM
PVPC PROCEDURE

- INTEGRATE AT AUTO - INIT - HOLD
- HOLD AT TRIM - (SIGN(EVPC) × SIGN(INTS))
- IC TO ZERO ON FIRST PASS
AUTOSTAB
TRIM POT (+ NOSE UP)
DEG

AUTO STAB
TRIM POT
(+ NOSE UP)
DEG

1.2 SECOND TIME DELAY

TRIM = 1

TRIMT

TRIMD

TRIMT: TRIMD TRUE CAUSES NOSE DOWN

TRIMD

YES

ASTP < 0

TRIM < 1

TRIMD = 0

HRAD (+= ABOVE RUNWAY)
FT

LINEAR RADIO ALTITUDE

H < 10 FT DETECTOR

GRD

STABT PROCEDURE

1. GRD + MANEL + PRENS

2. GRD IS SET ON SQUAT, CLEARED ON HRAD > IO
Once true, FLARE is to be latched (reset at change)

On first pass, IC = 'DEP' \& F is QFBI \& DECSL

On first pass, IC = 0

VTFCL SUBROUTINE
VARABLE TAU FLARE
This document describes the Flight Management/Flight Controls (FM/FC) software for the Norden #2 (PDP-11/70M) computer installed on the NASA 737 airplane. The software computes the navigation position estimates, guidance commands, and those commands to be issued to the control surfaces to direct the airplane in flight based on the modes selected on the Advanced Guidance Control System (AGCS) mode panel, and the flight path selected via the Navigation Control/Display Unit (NCDU).
PREPARATION OF THE REPORT DOCUMENTATION PAGE

The last page of a report facing the third cover is the Report Documentation Page, RDP. Information presented on this page is used in announcing and cataloging reports as well as preparing the cover and title page. Thus it is important that the information be correct. Instructions for filling in each block of the form are as follows:


Block 2. Government Accession No. Leave blank.

Block 3. Recipient’s Catalog No. Reserved for use by each report recipient.

Block 4. Title and Subtitle. Typed in caps and lower case with dash or period separating subtitle from title.

Block 5. Report Date. Approximate month and year the report will be published.


Block 7. Author(s). Provide full names exactly as they are to appear on the title page. If applicable, the word editor should follow a name.

Block 8. Performing Organization Report No. NASA installation report control number and, if desired, the non-NASA performing organization report control number.

Block 9. Performing Organization Name and Address. Provide affiliation (NASA program office, NASA installation, or contractor name) of authors.

Block 10. Work Unit No. Provide Research and Technology Objectives and Plans (RTOP) number.

Block 11. Contract or Grant No. Provide when applicable.


Block 13. Type of Report and Period Covered. NASA formal report series; for Contractor Report also list type (interim, final) and period covered when applicable.


Block 15. Supplementary Notes. Information not included elsewhere: affiliation of authors if additional space is required for block 9, notice of work sponsored by another agency, monitor of contract, information about supplements (film, data tapes, etc.), meeting site and date for presented papers, journal to which an article has been submitted, note of a report made from a thesis, appendix by author other than shown in block 7.

Block 16. Abstract. The abstract should be informative rather than descriptive and should state the objectives of the investigation, the methods employed (e.g., simulation, experiment, or remote sensing), the results obtained, and the conclusions reached.

Block 17. Key Words. Identifying words or phrases to be used in cataloging the report.

Block 18. Distribution Statement. Indicate whether report is available to public or not. If not to be controlled, use "Unclassified-Unlimited." If controlled availability is required, list the category approved on the Document Availability Authorization Form (see NHB 2200.2, Form FF427). Also specify subject category (see "Table of Contents" in a current issue of STAR), in which report is to be distributed.


Block 21. No. of Pages. Count front matter pages beginning with iii, text pages including internal blank pages, and the RDP, but not the title page or the back of the title page.

Block 22. Price Code. If block 18 shows "Unclassified-Unlimited," provide the NTIS price code (see "NTIS Price Schedules" in a current issue of STAR) and at the bottom of the form add either "For sale by the National Technical Information Service, Springfield, VA 22161-2171" or "For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402-0001," whichever is appropriate.
Teamwork Reasoning
and
Multi-Satellite Missions

NASA Agreement
Investigating Teamwork Among Spacecraft
Number
NAG 2-1489
Period of Performance: 7/1/01 - 4/30/02

Stacy C. Marsella
Information Sciences Institute
University of Southern California
Abstract

NASA is rapidly moving towards the use of spatially distributed multiple satellites operating in near Earth orbit and Deep Space. Effective operation of such multi-satellite constellations raises many key research issues. In particular, the satellites will be required to cooperate with each other as a team that must achieve common objectives with a high degree of autonomy from ground based operations. The multi-agent research community has made considerable progress in investigating the challenges of realizing such teamwork. In this report, we discuss some of the teamwork issues that will be faced by multi-satellite operations. The basis of the discussion is a particular proposed mission, the Magnetospheric MultiScale mission to explore Earth’s magnetosphere. We describe this mission and then consider how multi-agent technologies might be applied in the design and operation of these missions. We consider the potential benefits of these technologies as well as the research challenges that will be raised in applying them to NASA multi-satellite missions. We conclude with some recommendations for future work.
# Contents

1 INTRODUCTION ............................................................................................................................................. 4

2 MAGNETOSPHERIC MULTISCALE .................................................................................................................... 8
   2.1 MMS AND TEAMWORK ............................................................................................................................ 10
   2.2 MMS, FORMATION FLYING TESTBED AND DISTRIBUTED SATELLITE SIMULATION ............................. 11

3 SUPERVISORY CONTROL .................................................................................................................................. 12
   3.1 PLANNING AND SCHEDULING ..................................................................................................................... 12

4 TEAMWORK .................................................................................................................................................. 14
   4.1 TEAMCORE AND MMS .................................................................................................................................. 15

5 ANALYSIS AND SYNTHESIS OF TEAMWORK ............................................................................................. 17
   5.1 TECHNICAL DETAILS .................................................................................................................................. 18
   5.2 AN MTDP - MMS ANALYSIS EXAMPLE ......................................................................................................... 19
   5.3 MODELING REAL WORLD CONSTRAINTS .................................................................................................... 20
   5.4 SYNTHESIS AND RE-SYNTHESIS POTENTIAL .......................................................................................... 21
   5.5 EFFECTIVE POLICY DERIVATION ALGORITHMS ..................................................................................... 21

6 AN ASIDE: DATA-DRIVEN ANALYSIS ........................................................................................................... 23
   6.1 OVERVIEW OF ISAAC ................................................................................................................................. 24

7 ADJUSTABLE AUTONOMY ............................................................................................................................... 27
   7.1 TRANSFER OF CONTROL STRATEGIES ....................................................................................................... 28
   7.2 IMPLICATIONS OF STRATEGIES ................................................................................................................. 29
   7.3 MMS AND AA ............................................................................................................................................. 29

8 INTEGRATION .................................................................................................................................................. 30

9 RECOMMENDATIONS ..................................................................................................................................... 32

10 ACKNOWLEDGEMENTS .................................................................................................................................. 34

11 BIBLIOGRAPHY .............................................................................................................................................. 35
1 Introduction

NASA is rapidly moving towards the use of spatially distributed multiple satellites operating in near Earth orbit and Deep Space. The satellites will be required to cooperate with each other as a *team* that must achieve common objectives with a high degree of autonomy from ground based operations. Such satellite teams will be able to perform spatially separated, synchronized observations that are currently not feasible in single satellite missions. This will enable or improve multi-point observations of large scale phenomenon, co-observation of single phenomenon and interferometry. Autonomous operations will reduce the need for ground based support that would otherwise be prohibitively expensive in such missions. However, the underlying control systems necessary to enable such missions will raise many new challenges in autonomous, multi-platform operations.

In particular, a critical requirement for these satellite constellations is that they must act coherently as a coordinated, often autonomous team, and to do so even in the face of unanticipated events. This ability to operate as an autonomous team will need to be satisfied in many of the multi-satellite missions being planned. Therefore, it is important to understand this requirement, elucidate the research challenges it presents and consider approaches to satisfying it.

For example, consider the Magnetospheric Multiscale (MMS) mission. The mission involves 5 satellites flying in various formation configurations while making coordinated, simultaneous observations of the three-dimensional structure of the magnetosphere. MMS's observation plan has a projected 2 year life span involving multiple phases with different orbits and formation scales. The satellite "constellation" will face and need to respond in a timely fashion to hard to predict and unexpected events such as solar flare observation opportunities or equipment failures. The constellation will likely have to address most of these events without human operator intervention; there will be limited and delayed communication with earth-based human operators.

If an observation event occurs, the constellation may need to make a coordinated decision concerning the onset of observations and which sensor to use, decisions which in turn may be impacted by the status of each craft's sensor equipment. To realize this coordination, the satellites will need to communicate with each other. An effective policy for that communication is clearly a key requirement for the success of the mission. MMS also raises key issues about coordination between the constellation and ground-based operations. For example, in the face of unexpected events, the satellites must balance the need to react coherently in a timely fashion against the need for human oversight at critical junctures. At times, it may be best for the constellation to make an autonomous decision as how to proceed. At other times it may be best to seek human operator intervention. If that intervention does not come in a timely fashion, the constellation may still need to make an autonomous decision. An effective policy for such adjustable autonomy will be critical to the long-term survivability and success of the mission.
Of course, the question of how to achieve the necessary coordination between these craft and the adjustable autonomy with ground operations are requirements that are not unique to MMS or even multi-satellite operations in general. The multi-agent research community has been investigating these issues and has made considerable progress in addressing them. Various general approaches to coordinated teamwork and adjustable autonomy have been proposed, have been implemented in a variety of domains and have demonstrated considerable robustness. These approaches, for example, lay out prescriptions for when teammates should communicate and what they should communicate in order to achieve effective coordination on a team task such as a multi-point observation of an event. The design of multi-satellite missions will likely benefit greatly from this research. At the same time, a multi-satellite constellation will face difficult challenges that raise research questions which are not only at the frontiers of multi-agent research but will likely push that frontier forward.

One of these research challenges concerns the complexity of the process by which the satellites come to some coordinated decision and the quality of the resulting decision. For example, we might consider how MMS decides to make a joint observation with some sensor and whether it is the best decision they could make. Any approach will require certain communications, which consume power and time, and result in a specific decision that is, more or less, the optimal decision given the situation, the time it took to make the decision, etc. Furthermore, the MMS craft must make decisions in the context of a mission that has a 2 year lifespan, therefore the optimal decision for a specific observation event, for example, may be far from optimal in the context of subsequent tasks that must be performed. Indeed the very concept of optimal must take into account that the tasks the mission faces cannot be a priori specified with certainty, given the opportunistic nature of the observations, unexpected equipment failures, etc.

To address this challenge, it is useful to know certain baselines, such as what is the optimal decision for the team to make in any given situation and what is the complexity of finding that decision. However, to date insufficient progress has been made in precisely characterizing what constitutes an optimal decision and understanding the complexity of finding such optimal decisions. Given the lack of such baselines, it is not surprising that the various practical approaches to making teamwork decisions that have been proposed by the research community have also not been comparatively analyzed in terms of their optimality or complexity. Thus the optimality/complexity tradeoffs of proposed approaches cannot be determined, making it difficult to evaluate alternative approaches. Indeed, the optimal policy for a particular domain or application is typically unknown. This lack of progress in evaluating alternative approaches to central problems in teamwork is particularly worrisome in high cost, critical applications such as satellite constellations.

A second, closely related, research question concerns the limited resources any multi-satellite mission faces. Only limited progress has been made by the multi-agent research in explicitly modeling the real world constraints that are fundamental to the success of a satellite mission. For example, communication is in general a cornerstone of effective teamwork and will likely be key to maintaining MMS satellite coordination. However, communication has a cost. It can consume considerable power, can impact certain kinds of data collection and can delay other actions if for example one member of team communicates and waits for a response from other teammates. Similar real world issues arise in the case of adjustable autonomy. Traditionally, the adjustable autonomy issue has been framed as a one-shot decision to either make an autonomous decision
or pass control to a human (e.g., ground controllers). However, the decision to pass control may lead to costly delays which ideally should be factored into the decision to transfer control. But in the real world the length of the delay is typically indeterminate, drawing into question the advisability of making such a one-shot decision.

However, recent advances in formal models of teamwork and adjustable autonomy have begun to address these challenges. For example, work in casting teamwork into a formal framework, what we call an MTDP (multi-agent team decision problem), provides a tool to address a range of analyses critical to fielding teams in real world applications. Using the MTDP framework, the complexity of deriving optimal teamwork policies across various classes of problem domains can be determined. The framework also provides a means of contrasting the optimality of alternative approaches to key teamwork issues like role replacement. Finally, the framework also allows us to empirically analyze a specific problem domain or application of interest. To that end, a suite of domain independent algorithms has been developed that allow a problem domain to be cast into the MTDP framework. This allows the empirical comparison of alternative teamwork approaches in that domain. Derivation of the optimal policy for the problem domain serves not only as the basis of comparison but also can inform the design of more practical policies. Most recently, progress is being made in addressing how real world operating constraints like power consumption can be modeled in this framework.

Another critical research question concerns integration. Clearly, these teamwork and autonomy decisions cannot be made independently from the rest of the operational decisions being made on the craft. But the question of how they integrate is yet another research question.

In this report, our goal is to illuminate several basic issues in the application of multi-agent research to multi-satellite missions. We discuss the need to develop robust and effective coordination prescriptions for multi-satellite teamwork. Rather than mission-by-mission ad hoc approaches to coordination, we focus on a general approach to teamwork that will be both more robust in a particular mission while also building across mission teamwork infrastructure. We also stress the need for analysis and suggest an approach to assessing the quality of alternative prescriptions, based on MTDPs, that allows both formal and empirical evaluation. We illustrate how the approach could be applied to MMS and discuss how it could be extended to provide a faithful rendering of difficult resource limits that such missions will operate under. In addition, we discuss alternatives to realizing the teamwork reasoning and how teamwork and autonomy is integrated into a craft's overall software architecture.

The discussion of these issues begins in Section 2, by describing the MMS mission and pointing out some of the technical challenges it raises for teamwork and adjustable autonomy. But of course, teamwork and autonomy reasoning are just one part of the constellation's operation, which must include various flying, observation, communication and maintenance tasks over the duration of the mission. So we briefly introduce the supervisory control software that manages and schedules these tasks. In particular, we discuss one approach to the design of this supervisory software in order to facilitate later discussions. We then discuss in Section 4 the issue of realizing robust teamwork, as the problem is approached by the STEAM architecture. Section 5, Analysis and Synthesis of Teamwork, presents one of the central proposals of this report, the use of formal models for analyzing teamwork. Section 6 presents some prior work in teamwork analysis. Sections 7 and 8 discuss in turn adjustable autonomy and the integration of
teamwork reasoning with the supervisory control software. Finally, Section 9, Recommendations, suggests several directions for the research and also potential collaborations with NASA.
2 Magnetospheric Multiscale

The Magnetospheric Multiscale (MMS) mission is being designed to investigate the processes of magnetic reconnection, charged particle acceleration and turbulence in the Earth’s magnetosphere. The study is concerned with the dynamic and spatial structure of these processes and thus it can not feasibly be undertaken by a single craft. A multi-satellite mission design is being developed that uses identical spacecraft capable of flying in formation and making the simultaneous, coordinated observations required. The 5 satellites of MMS will fly in a hexahedral formation near apogee, comprising two tetrahedral with three of the satellites in a plane with the fourth satellite above and a fifth below that plane. See Figure 1. An alternative design will have four craft defining a tetradron with the fifth craft (potentially) placed within that tetrahedron. See Figure 2. The formation will at times elongate into a string of pearls, depending on where it is in the orbit and the temporal/spatial goals of the observations. Whereas observations that could separate spatial and temporal characteristics of observed phenomenon could be done by two craft, the ability to resolve these characteristics are significantly improved by 5 craft. In order to capture data from different regions of the magnetosphere, there are multiple phases to the mission with different orbits and different inter-satellite distances. Specifically, Phase 3 and Phase 4 will involve more distant observations, including magnetotail studies at up to 120 RE. Depending on the phase of the mission, the spacing between satellites will range for from tens of kilometers to tens of thousands of kilometers, with separations sometimes increasing or decreasing over orbital phase. The MMS mission has an operational duration of 2 years.

Figure 1. MMS Spacecraft in hexahedral configuration.
Each craft has memory on board to record data which must be transferred to ground stations at appropriate times. As of early 2002, the craft design proposed a sensor system that has two data rates, high and low, which give them different resolution observations. At interesting events, such as a solar flare, the craft should go into high data rate to get the greatest amount/resolution of data. However, some highly desired events happen quickly enough that they can be missed, at least partially. The memory also fills quickly at high data rates. Plus the buffers on the craft may have different amounts of free memory at any time, making it more or less feasible for them to go into high data rate. Not all craft need to be at the same rate during an observation, surprisingly, but the more the better. Also, Phase 3 and 4 of the mission will require the DSN 34-meter dish. Since the downlink of data is sensitive to distance and ground station cost can be prohibitive, the craft will be required to store weeks of data until their orbit brings them close enough for high speed downlinks.

Additionally, the MMS satellites will carry a range of instruments, including plasma instrumentation, energetic particle detector, electric field/plasma wave instruments and magnetometer. For various reasons, a craft's instruments may not be operable simultaneously. For example, they may share electronics. The operation of the sensors will also need to be coordinated between craft.

Figure 2. MMS Spacecraft, five tetrahedral configuration.
Finally, the formation is not designed to dynamically reconfigure - the reconfiguration is preplanned depending on where they are in orbit and which phase of the mission it is. Apparently, it is too expensive to consider dynamic reconfiguration - it costs too much in fuel (and consequently liftoff weight). This in principle limits the kinds of coordination tasks that need to be addressed, but does not eliminate the need for coordination. Because of this limitation, however, this document will not address in great detail possible relations between teamwork reasoning or adjustable autonomy and low-level control algorithms that will be used to maintain the crafts' formation. Rather the focus will in large measure be on the science operations.

### 2.1 MMS and Teamwork.

A decision to make an observation potentially faces various tradeoffs with respect to the interestingness of the observation, the feasibility of any particular satellite going into high rate given its free memory, the quality of the observations that results or when the next downlink is feasible. Additional factors may arise that affect the high/low data rate decision. It is also not clear when to turn back to low data rate - presumably because it is not clear when the observation of an event should end. Closely related is the possibility of foregone future observations due to too full memory prior to any downlink. Or for that matter some satellite could run out of memory mid-observation. The state/precision of the constellation's formation will also impact observation quality and arguably should be factored into the high and low data rate decision.

Related to this observation decision, there are also interesting coordination issues and tradeoffs to be considered. The current proposed coordination approach is an alarm system. A satellite individually detects interesting events and signals others that it spot the event and is going into high data rate, other satellites should in turn signal that they are going into high data rate, assuming they have the buffer space. This is "what's my state" coordination technique that appears to preclude the possibility of coordinating the high/low data rate decision as a team decision which arguably might be a better approach — since the individual decision may need to take into account the state of the team such as the other satellites state of memory, value of only part of the formation going into high rate, the teams current formation, the possibility of false alerts, whether all craft's sensors are working, power levels in the various craft, etc.

Moreover, the high/low data rate decision is clearly just one decision to coordinate. For example, which instruments will be activated to perform an observation clearly must be coordinated between craft. Again, there may be many factors that could impact this decision and might argue that a coordinated, team decision is preferable. For example, if one or more craft has an instrument failure, then this might argue for changing the observation to other instruments. Since useful, but degraded, observations of spatial/temporal characteristics can possibly be made by even two craft, the appropriate decision may not be obvious.

There also is another planned mission, solar sentinel, that will be closer to the sun that could coordinate with MMS. Specifically, it could be used as early warning sensors for interesting events.
Finally autonomy is critical here. Ground links in general are expensive, especially when, in Phase 3 and 4, DSN is required. Communication would thus drive up costs astronomically and will be relatively infrequent. The uplink of data is designed to be quite contained, one design for the mission specifies that commanding for the instruments will be 100 bytes per day per craft.

2.2 **MMS, Formation Flying Testbed and Distributed Satellite Simulation.**

The MMS mission has been chosen as the first mission design to be explored within the Formation Flying TestBed (FFTB) being developed at Goddard. FFTB is specifically designed to evaluate the low-level distributed control algorithm (DCA) and hardware involved in realizing the low-level formation maintenance and station-keeping necessary for a mission like MMS. However, the FFTB is also becoming the kernel of a distributed satellite simulation system (DSS) that will bring software and hardware together within a distributed system that will allow the simulation of an entire mission. Since the FFTB and DSS presents special opportunities for evaluating the constellations control software, we briefly describe these components here and raise certain implications of their design for the teamwork research.

The Formation Flying Testbed (FFTB) at NASA GSFC is a modular, hybrid dynamic simulation facility being developed as a platform for the evaluation of guidance, navigation, and control of formation flying clusters and constellations of satellites. The FFTB is being developed to support both hardware and software development for a wide range of missions involving distributed spacecraft operations.

The FFTB has several features of special note here. It is being designed to realize very high fidelity simulations of a constellations formation flying that will provide a strong test for software design. It is a hybrid simulation system that can employ a blend of hardware of software components. The use of hardware within the simulation system can constrain the simulation to run in real time. However, the FFTB design is modular. Software modules can be swapped in for the hardware modules, which would allow faster than real time simulation but at the cost of some loss in the fidelity of the simulation.

Most critically, FFTB is the core of an evolving distributed simulation system/environment (DSS) for satellite constellations. DSS could potentially support the simulation of all aspects of a mission, including the multiple sensors, absolute and relative position determination and control, in all (attitude and orbit) degrees of freedom, information management, high-level supervisory control as well as the underlying physical phenomenon the constellation is designed to observe.

This implementation is therefore an ideal framework for exploring and evaluating alternative approaches to the high-level supervisory control of the craft and its coordination with other craft in the constellation and ground control. The supervisory control has the general functions of validating the data in the navigation system, switching the modes of operation of the vehicle based on either events or schedules, and interfacing the on-board functions with ground functions. Thus teamwork reasoning will need to play some integrated role in supervisory control.
3 Supervisory Control

The main focus of this paper is the adjustable autonomy and teamwork decision-making that arise in missions like MMS. However, these capabilities are realized within the context of each craft's supervisory control software that overall decides what tasks are performed and when they are performed. Thus the relation between the teamwork reasoning and supervisory control is a central issue. For example, one issue that will arise in later discussions concerns the tradeoffs between realizing teamwork reasoning as a separate module versus a tighter integration.

In order to help set the context for that subsequent discussion, we introduce here an example of a general purpose architecture for supervisory control of remote craft that has been proposed by NASA Ames. We leave out many architectural specifics such as the relation of supervisory control to the distributed control algorithms used for formation maintenance.

3.1 Planning and Scheduling

NASA Ames's Intelligent Deployable Execution Agents (IDEA) [14] framework for planning and scheduling, which is a continuation of the work begun for the Remote Agent. IDEA has four main components; (1) a plan database which represents all possible plans that are consistent with the current set of instantiated constraints, (2) a domain model which defines the operational constraints for the craft, (3) a set of planners that generate plans in the plan database and (4) the plan runner which performs execution. Figure 1, borrowed from a NASA report, depicts IDEA.

The IDEA has many interesting capabilities but for our subsequent discussions, two features are most relevant. IDEA allows for multiple planners with different planning time responses, some of which may be more deliberative while others may be more reactive or scripted. The plan database provides a uniform representation for these planners and the execution of the resulting (partial) plan. Within the plan database, it is possible to represent not only partial plans for execution tasks but also flexibly represent planning tasks and reason about the scheduling constraints on those planning tasks. For example, IDEA could schedule a planning task, based on other operational constraints (e.g., whether the cpu is available). IDEA could also choose between alternative planning strategies based on scheduling constraints or modify other mission tasks to ensure time for planning (e.g., go into a wait loop).
Figure 3. The IDEA Framework (from NASA report)
4 Teamwork

Although there is an increasing demand for multi-agent systems that enable a team of agents to work together, getting the team to perform well in a dynamic environment remains a difficult challenge. It is particularly difficult to ensure robust and flexible performance in the face of unexpected events. Individual agents may fail and there may also be coordination breakdowns, due to agent's not having a shared mental model. Building a system may require a potentially large number of special purpose coordination plans to cover all the low-level coordination details. If the underlying system tasks change or new agents are added, new coordination plans will be needed.

Considerable progress has been made over the years in developing and implementing practical models of teamwork that address these design challenges. Theoretical work on teamwork [Cohen & Levesque, Grosz, etc] laid the solid basis for implemented systems, such as STEAM [Jair], a general model of teamwork that explicitly reasons about commitments in teamwork. STEAM demonstrated the real-world utility of explicit reasoning about teamwork commitments for designing robust organizations of agents that coordinate amongst themselves. In a STEAM system, each team member has general purpose teamwork reasoning skills as well as an explicit model of the team plan and its commitments to teammates. Thus each teammate knows that it is in a team and it has commitments to achieve team goals. Plus they possess rules for achieving the coordination required by those commitments in the face of unforeseen events. So, for example, if a teammate sees another teammate fail in a key task, it will reason about whether to warn teammates.

In particular, STEAM contains maintenance-and-repair rules that enable team members to monitor the impact of failing teammates and suggest recovery for such failures (e.g., by substitution of a failing team member with another). It also contains coherence-preserving rules which enable team members to supply each other key information to maintain coherence within a team, and communication-selectivity rules that help agents limit their communication using decision-theoretic reasoning. For example, one coherency preserving rule is that teammates need to know when a team task is achievable. Therefore, if an agent observes that a team task is achievable, this rule comes into play and the agent will decide to communicate the information, based on the communication-selectivity rules.

These rules realize general teamwork reasoning and therefore apply across any team task. They are as well practical in the sense that they take into account tradeoffs. In particular, the communication-selectivity rules take into account the criticality of the task, the cost of the communication and the likelihood that teammates already know. Our experience in a host of difficult domains is that this combination of general teamwork reasoning skills, explicit team plans and decision-theoretic reasoning about tradeoffs is robust. It's robustness follows from the emphasis on giving general teamwork reasoning skills to each teammate. The underlying assumption is that the world is "open", that the unexpected event can happen in the world. The designer of the team cannot pre-plan for every such event but rather must design general
methods for teamwork reasoning about failures that allow the teamwork to maintain a coordinated, effective response.

STEAM's successful applications of teamwork to multi-agent systems lead to the Teamcore architecture. The key hypothesis behind Teamcore is that teamwork among agents can enhance robust execution even among heterogeneous agents in an open environment. The Teamcore architecture enables teamwork among agents with no coordination capabilities, and it establishes and automates consistent teamwork among agents with some coordination capabilities, by providing each agent with a proxy capable of general teamwork reasoning. At the heart of each Teamcore proxy is the STEAM teamwork model, which provides the set of rules that enable heterogeneous agents to act as responsible team members. The power of the resulting architecture stems from these built-in teamwork capabilities that provide the required robustness and flexibility in agent integration, without requiring modification of the agents themselves.

The Teamcore/STEAM framework has been successfully applied in several different domains. STEAM's original application was in the battlefield simulation environment where it was successfully used to build a team of synthetic helicopter pilots that participated in DARPA's synthetic theater of war (STOW'97) exercise, a large scale exercise involving virtual and real entities, including human pilots. STEAM was later reused in RoboCup Soccer, where it led to top performing teams in International RoboCup Soccer tournaments. STEAM is at the heart of the Teamcore proxies, which now enable distributed heterogeneous agents to be integrated in teams. Teamcore has been applied to bring together agents developed by different developers in DARPA's COABS program; these agents had no teamwork capabilities to begin with, but Teamcore allowed their smooth integration. Finally, Teamcore is also being used in the "Electric Elves" project, a deployed agent system at USC/ISI, which has been running 24/7 since June, 2000. This system provides Teamcore proxies for individual researchers and students at USC/ISI, as well as proxies for a variety of schedulers, matchmakers, information agents. The resulting team of 15-20 agents helps to reschedule meetings, decide presenters for our research meetings, track people and even order our meals.

4.1 TEAMCORE and MMS

It is useful to consider how we might apply Teamcore to MMS. Consider the previously discussed observation coordination example. To realize coordinated observations, observations would be defined as a team task, which would be achievable, for instance, when a solar flare happened. If a satellite now observed a solar flare, the coherency-preserving and communication rules would lead it to communicate to its teammate satellites that observation was now achievable (i.e., should be jointly executed). This would lead them to turn on high data rate as a team.

We can also consider somewhat more ambitious, speculative scenarios based on general teamwork reasoning and team reformation. For instance, assume MMS is in operation when some other mission is launched, enabling in some way better or earlier sensing of interesting events. For example, Solar Sentinel would be such a mission. To exploit this new capability, the already in-flight MMS craft, in principle, would not have to be modified (which would be risky and costly to do via command uplink). The new craft would just be added as a member of the MMS observation team, using the same Teamcore reasoning and observation team task. When it
sensed an event, it would inform its teammates, the original MMS team. In practice, of course, this flexibility presumes that the new craft has some communication channel with the MMS craft, a network in some sense. Although such a network may not be currently feasible, if it were one can envision such plug and play teams of heterogenous satellites helping each other on their missions by dynamically taking on new roles in each other’s tasks.

Let’s also consider the case of failures. Assume some planned action by the supervisory control software, such as an attitude adjustment, suffers a failure of some kind. If the failure impacts a team task such as an observation, then the agent will signal its Teamcore proxy teamwork layer that it cannot perform its role in the team task as planned. The proxy will communicate with the other satellites in the team which will attempt to adjust their plans. If they cannot, they will in turn communicate failure on the team task that will in turn lead to a coordinated response to the initial failure.
5 Analysis and Synthesis of Teamwork

Based on systems like Teamcore/STEAM, multi-agent systems have moved out of the research lab into a wide range of applications areas. But of course, multi-satellite control is a highly critical application, where seemingly minor control decisions can have drastic consequences when made incorrectly. To meet the challenge of such a bold application, multi-agent research will need to provide high-performing, robust designs that performs such control as optimally as feasible given the inherent uncertainty of the domain. Unfortunately, in practice, research on implemented systems has often fallen short in assessing the optimality of their proposed approaches with respect to mission-level performance criteria.

To address this shortcoming, researchers have increasingly resorted to decision-theoretic models as a framework in which to formulate and evaluate multi-agent designs. Given some group of agents, the problem of deriving separate policies for them that maximize some joint reward (i.e., performance metric) can be modeled as a decentralized partially observable Markov decision process (DEC-POMDP). In particular, the DEC-POMDP model is a generalization of a POMDP to the case where there are multiple, distributed agents basing their actions on their separate observations. POMDP is in turn a generalization of a single agent Markov decision process, or MDP, whereby the agent makes decisions based on only partial observations of the state.

The Com-MTDP model is a closely related framework that extends DEC-POMDP by explicitly modeling communication. R-COM-MTDP in turn extends Com-MTDP to enable explicit reasoning about Team Formation and Re-Formation.

These MTDP frameworks allow a variety of key issues to be posed and answered. Of particular interest here, these frameworks allow us to formulate what constitutes an optimal policy for a multi-agent system and in principle to derive that policy.

For example, the COMmunicative Multiagent Team Decision Problem (COM-MTDP) provides a general-purpose language for representing the interactions among intelligent agents sharing a complex environment. The COM-MTDP can capture the different capabilities of the various agents in the world to perform actions and send messages. The model can represent the uncertainty in the occurrence of events, in the ability of the agents to observe such events, and in the effects of those events on the state of the world. The model also uses a reward function to quantify fine-grained preferences over various states of the world. The overall model provides a decision-theoretic basis for examining and evaluating possible courses of action and communication for the agents so as to maximize the expected reward in the face of their environment's ubiquitous uncertainty.

In a COM-MTDP, the behavior of the team is modeled as a joint policy that determines each agent's action based on its observations. There is also a reward function that assigns a value for an agent performing that action in the current state of the world. This framework allows us to determine what is the expected utility of any policy and in principle derive the optimal policy for a team of agents. It may also be a robust policy but only if the probabilistic models have done a faithful rendering of what could happen in the world. Note that in this analysis framework the
individual agent knows nothing about being in a team. Knowledge about being in a team is not explicitly being modeled. Rather, a central planner derives a joint policy and each agent only has its part of the policy which tells it what to do next based on its current beliefs. This is quite different from the STEAM teamwork reasoning where each teammate knows it is in a team and can reason individually and as a team about how to best maintain team coordination in pursuit of team goals.

5.1 Technical Details

The COMmunicative Multiagent Team Decision Problem (COM-MTDP) model subsumes previous distributed models in control theory, decision-theoretic planning, multiagent systems, and game theory. An instantiated COM-MTDP model represents a team of selfless agents who intend to perform some joint task. This COM-MTDP is specified as a tuple, <S,A,O,B,R>.

S is a set of world states which describes the state of the overall system at a particular point in time. For example, the state of a typical COM_MTD model would capture the status of the agents (e.g., satellites) themselves, including their positions, their available power, their communication queue, etc. The state would also represent the current environment, external to the agents themselves (e.g., position of other satellites or observation targets).

A_i is the set of control decisions that each agent i can make to change itself or its environment, implicitly defining a set of combined system actions, A. The actions of an individual agent/satellite, for example, may include choice of sensor, choice of orientation, choice of power consumption (perhaps selecting between high- and low-quality sensing), and potentially even a choice to do no sensing at all (e.g., to maximize power conservation).

The state of the world evolves in stages that represents the progression of the system over time. For nontrivial domains, the state transitions are non-deterministic and depend on the actions selected by the agents in the interval. The non-determinism inherent in these transitions is quantified by specifying transitions as a probabilistic distribution. The transition probability function can represent the non-deterministic effects of each agent's choice of action.

O_i is a set of observations that each agent, i, can experience of its world, implicitly defining a combined observation. O_i may include elements corresponding to indirect evidence of the state (e.g., sensor readings) and actions of other agents (e.g., movement of other satellites or robots). The observations that a particular agent receives are non-deterministic (e.g., due to sensor noise), and this non-determinism is quantified with a set of observation functions. Each such observation function defines a distribution over possible observations that an agent can make. Each observation function represent the noise model of a node's sensors, so that we can determine the relative likelihood of the various possible sensor readings for that node, conditioned on the real state of the system and its environment.

C_i is a set of possible messages for each agent, i, implicitly defining a set of combined communications. An agent may communicate messages to its teammates.
Each agent forms a belief state based on its observations seen and messages received through time, where $B_i$ circumscribes the set of possible belief states for the agent. The agents update their belief states at two distinct points within each decision epoch: once upon receiving observation (producing the pre-communication belief state) and again upon receiving the other agents' messages (producing the post-communication belief state). The distinction allows us to differentiate between the belief state used by the agents in selecting their communication actions and the more "up-to-date" belief state used in selecting their domain-level actions.

An agent's belief state forms the basis of its decision-making in selecting both domain-level actions and communication. This decision-making is summarized by mappings from belief states into actions and messages, using a domain-level policy that maps an agent's belief state to an action and a communication-level policy.

A common reward function $R$ is central to the notion of teamwork in this model. This function represents the performance metric by which the system's overall performance is evaluated. The reward function represents the team's joint preferences over states, the cost of domain-level actions and the cost of communicative acts (e.g., communication channels may have associated cost).

### 5.2 An MTDP - MMS analysis example

The COM-MTDP work was originally envisioned as a framework for analyzing teamwork strategies but increasingly we have begun to explore its use in synthesizing teamwork strategies. However, let's first exemplify its use in analysis.

MTDP can be applied to represent the MMS spacecraft's data acquisition discussed earlier. To do this, we would represent each of the spacecraft as an agent, with state features representing the status of each spacecraft. We could also potentially represent a spacecraft's power limitations and consumption within the MTDP model's state space and transition probability. In particular, for each spacecraft, there would be a corresponding state feature representing its available power. The transition probability function would model the dynamics of this available power as a stochastic process, with the change in available power as a function of the spacecraft's choice of action (e.g., data transmission accelerates the rate of power consumption). We can use similar state features to represent the position, orientation, amount of data recorded for each spacecraft, as well as a similar transition probability function to represent the dynamics of each. Such state-based representations have proven successful in modeling distributed systems, and we have had similar success ourselves in applying them to multiagent systems.

There would be additional state features to represent the state of the magnetosphere around them. These features would capture the presence/absence of the various phenomena of interest to the mission. The transition probability function would capture the stochastic evolution of the magnetosphere state, perhaps by incorporating existing models (e.g., MHD models). An agent's observation function would provide a probabilistic model of its corresponding spacecraft's
sensors in relation to the state of the surrounding magnetosphere.

Each agent would have a choice of recording or not recording data. The MTDP reward function represents the relative value of its choice after taking into consideration the magnetosphere state. In other words, recording data will have a high value when phenomena of interest are present in the current state. The magnitude of the value will correspond to the relative value of the present phenomena. When an agent decides to record data, the transition probability function will represent the change in the spacecraft's state (i.e., it will have less memory left for recording data).

Given such an MTDP model, we can evaluate data acquisition procedures by encoding them as agent policies. In other words, each agent's policy would represent its corresponding spacecraft's decision process in deciding when to record data, based on its sensor readings. We can then use MTDP algorithms to simulate the behavior of these policies over the possible magnetosphere events. By evaluating the reward earned by the agents over these possible events, weighed against their likelihood, we can derive an expected reward of the policies selected, which in turn allows us to characterize the various performance tradeoffs. We can manipulate the MTDP reward function to isolate the dimensions of interest for each such tradeoff. For instance, if we wish to quantify the ability of an acquisition procedure to avoid running out of power, we can define a reward function that has value 1 in a state where a spacecraft has no available power and 0 in all other states. We can then use our evaluation algorithm to compute the expected reward earned by the nodes, which, with this reward function, will exactly measure the probability that a spacecraft runs out of power. We can make similar reward function definitions that allow our evaluation algorithm to compute expected amount of data recorded, amount of data transmitted, expected number of interesting phenomena missed, etc. We can combine reward functions over different dimensions into a single reward function to consider the two dimensions simultaneously and thus quantify the tradeoffs between them. Furthermore, by replacing the expectation in these algorithms with minimization and maximization, we can compute best- and worst-case statistics as well.

This provides a potential basis for selecting between various candidate data acquisition policies. The MTDP model can also potentially provide feedback into the design process underlying data acquisition. A system designer can consider the output of our evaluation algorithms (i.e., the separate predictions and the tradeoffs between them) when choosing among various candidate data-acquisition procedures. These performance predictions will provide the algorithm designers with concrete performance profiles of their algorithms' performance under realistic conditions. The designers can then take these profiles (e.g., too many messages, low probability of success) and use them to make informed improvements to the means by which they achieve data acquisition. This will help our research but in addition provide useful, practical information and software tools (the MTDP analysis framework in particular) for developing these missions.

5.3 Modeling Real World Constraints

Formal models of distributed systems have typically neglected to model real world resource limits. In contrast, one of the features of the previous example use of MTDP was the proposal to
model power consumption dynamics. This represents current research in which we are investigating how various real world resource limits such as power consumption can be modeled as first class entities. Since one of the difficult challenges faced by many NASA missions and MMS in particular is the tight resource constraints they operate under, this added capability will clearly have special relevance for using the MTDP framework for NASA mission analyses.

5.4 Synthesis and Re-Synthesis potential.

As commented earlier, the MTDP work was originally envisioned as a framework for analyzing teamwork algorithms. Our experiences to date have also revealed an extremely interesting potential for synthesis. For example, we can use the MTDP work to derive an optimal policy for some team mission by simply simulating all possible policies out to some bounded point in the simulation and picking the best one. This optimal policy is of course only optimal under the assumptions about the world built into the probabilistic models used in the simulation. And it is not a tractable simulation to perform in general. Nevertheless, it does provide a benchmark against which to measure the optimality of alternative teamwork reasoning approaches such as the TEAMCORE work mentioned above. When we have done this kind of benchmarking, we found that the MTDP may generate optimal policies that were entirely unexpected. For example, the optimal policy might replace "failed" teammates before they fail - in essence employing a redundancy approach in high-risk situations. The optimal policy might flexibly decide to replace or not replace based on the expected utility. Finally in some cases it might choose to abandon the mission. None of these capabilities were built into the experiments by the designers - they were discovered by deriving and then inspecting the optimal policy.

This discovery suggests a third approach to building agent teams – the iterative combined approach. Here the domain is modeled probabilistically, the optimal policy is derived and this policy is analyzed to suggest possible improvements to the more general-purpose teamwork reasoning strategies such as employed in TEAMCORE. In other words, by examining the optimal policy (which may be infeasible with real-world resource constraints), we could identify deviations made by our more practical TEAMCORE architecture. We can then modify the architecture to be more in line with the ideal behavior specified by the optimal policy, and thus minimize the suboptimality that we achieve in practice.

5.5 Effective policy derivation algorithms

COM-MTDP and decentralized POMDPs clearly show considerable promise for multi-agent research as well as the application of that research. One key step to using these formalisms is the derivation of the policies. However effective algorithms for deriving policies for decentralized POMDPS is ongoing research. Significant progress has been achieved in efficient single-agent POMDP policy generation algorithms (refs, Monahan, etc). However, it is unlikely such research can be directly carried over to the decentralized case. Finding an optimal policies for
decentralized POMDPs is NEXP-complete and therefore provably does not admit a polynomial time algorithm (Bernstein, Zilberstein and Immerman). In contrast, solving a POMDP is PSPACE-complete (Papadimitriou and Tsitsiklis). As Bernstein et al. note (ref), this suggests a fundamental difference in the nature of the problems. Since the reward function is a joint one, the decentralized problem can not be treated as one of separate POMDPs in which individual policies can be generated for individual agents. (For any one action of one agent, there may be many different rewards possible, based on the actions that other agents may take.)

In our own work, we have developed several policy derivation algorithms. Among these is an exact algorithm that generates optimal policies via a full search of the space of policies. This exact algorithm is of course expensive to compute which limits its applicability to problems for which there is sufficient time to offline pre-compute such an exact solution or some way of decomposing the problem a priori. Therefore, we have also developed approximate algorithms. For example, one approach is to search the space of policies incrementally. This algorithm iterates through the agents, finding an optimal policy for each agent assuming the policies of the other agents are fixed. The algorithm terminates when no improvements to the joint reward is achieved, thus achieving a local optimum similar to a Nash Equilibrium.

This question of effective algorithms will likely be of special relevance to the application of these formalisms to MMS. Given its projected mission duration of two years, a brute force search for the optimal policy would not be feasible. However, although the resource constraints of such missions will complicate our representation, they may actually simplify such algorithms by restricting the search space of implementable policies. For example, the optimal policy for many COM-MTDP problems requires that the agents remember all of their observations throughout their lifetime and then choose different actions based on all possible such observation sequences. Spacecraft with the limited memory resources cannot store such a policy, let alone execute it. The number of possible policies that are executable is much smaller than the number of unrestricted policies, which suggests that finding optimal policies subject to the mission resource constraints may be feasible through novel COM-MTDP synthesis algorithms.
6 An aside: Data-driven analysis

The COM-MTDP work provides an approach to analyzing team performance. A key requirement for the analysis is the probabilistic models of the domain and task, for example the state transition probabilities and the observation function. This begs the question of where these models come from.

In the case of MMS, these models could be derived directly or indirectly from the models of the magnetosphere, of the low-level flight control, etc. that are part of the Formation Flying Test Bed (FFTB) and Distributed Satellite Simulation (DSS) mentioned earlier which are being developed at Goddard. For example, an indirect derivation would rely on the simulation of these models within the DSS that could be sampled to derive estimates of the probabilistic models needed for COM-MTDP. By combining the COM-MTDP framework and the DSS simulation, the overall approach to the analysis would be more driven by the data in the simulations. More generally, we envision such combinations of analytical analysis and simulation to be a particularly fruitful research path.

This optimism stems from our prior experiences in researching data-driven approaches to analysis that used simulation data to derive models that were subsequently used for teamwork analysis. In particular, such an approach was used by the ISAAC teamwork analysis tool (ref). ISAAC performs post-hoc, off-line analysis of teams using agent-behavior traces derived from the team’s performance in the domain or simulation of the domain. This analysis is performed using data mining and inductive learning techniques to derive models of the team’s performance in the domain. Using data from the agents’ external behavior traces, ISAAC is able to analyze a team with very little in the way of pre-existing models of the domain or the team’s internals.

In fact, ISAAC develops multiple models of a team. To fully understand team performance, multiple levels of analysis are critical. One must understand individual agent behavior at critical junctures, how agents interact with each other at critical junctures as well as the overall trends and consequences of team behavior throughout the life of a mission. Thus ISAAC is similarly capable of analyzing from multiple perspectives and multiple levels of granularity. To support such analyses, ISAAC derives multiple models of team behavior, each covering a different level of granularity. More specifically, ISAAC relies on three heterogeneous models that analyze events at three separate levels of granularity: an individual agent action, agent interactions, and overall team behavior. These models are automatically acquired using different methods (inductive learning and pattern matching) — indeed, with multiple models, the method of acquisition can be tailored to the model being acquired.

Yet, team analysts such as ISAAC must not only be experts in team analysis, they must also be experts in conveying this information to humans. The constraint of multiple models has strong implications for the type of presentation as well. Analysis of an agent action can show the action and highlight features of that action that played a prominent role in its success or failure, but a similar presentation would be incongruous for a global analysis, since no single action would suffice. Global analysis requires a more comprehensive explanation that ties together seemingly
unconnected aspects and trends of team behavior. ISAAC uses a natural language summary to explain the team’s overall performance, using its multimedia viewer to show examples where appropriate. The content for the summary is chosen based on ISAAC’s analysis of key factors determining the outcome of the engagement.

Additionally, ISAAC presents alternative courses of action to improve a team using a technique called ‘perturbation analysis’. A key feature of perturbation analysis is that it finds actions within the agents’ skill set, such that recommendations are plausible. In particular, this analysis mines data from actions that the team has already performed.

ISAAC has been applied to all of the teams from several RoboCup tournaments in a fully automated fashion. This analysis has revealed many interesting results including surprising weaknesses of the leading teams in both the RoboCup '97 and RoboCup '98 tournaments and provided natural language summaries at RoboCup '99. ISAAC was also awarded the ‘Scientific Challenge Award’ at the RoboCup '99 international tournament. ISAAC is available on the web at http://coach.isi.edu and has been used remotely by teams preparing for these competitions.

While ISAAC is currently applied in RoboCup, ISAAC’s techniques are intended to apply in other team domains such as agent-teams in satellite constellations. For example, ISAAC could produce a similar analysis for the DSS simulation system and use similar presentation techniques as well. Indeed, we believe that the COM-MTDP analysis work could be incorporated into a ISAAC-like tool for the DSS system.

6.1 OVERVIEW OF ISAAC

(Perhaps delete this section)

ISAAC uses a two-tiered approach to the team analysis problem. The first step is acquiring models that will compactly describe team behavior, providing a basis for analyzing the behavior of the team. As mentioned earlier, this involves using multiple models at different levels of granularity to capture various aspects of team performance. The second step is to make efficient use of these models in analyzing the team and presenting this analysis to the user. An overview of the entire process is shown in Figure 4.

![Figure 3. ISAAC analysis process.](image-url)
Input to all models comes in the form of data traces of agent behaviors. In the current implementation of ISAAC, these traces have been uploaded from users around the world through the Internet.

As shown in figure 4, acquiring the models involves a mix of data mining and inductive learning but is specific to the granularity of analysis being modeled. Analysis of an individual agent action (individual agent key event model) uses the C5.0 decision tree inductive learning algorithm, an extension to C4.5, to create rules of success or failure [ref]. For analysis of agent interactions (multiple agent key interaction model), pre-defined patterns are matched to find prevalent patterns of success. To develop rules of team successes or failures (global team model), game level statistics are mined from all available previous games and again inductive learning is used to determine reasons for success and failure.

Utilizing the models involves catering the presentation to the granularity of analysis to maximize human understandability. ISAAC uses different presentation techniques in each situation. For the individual agent key event model, the rules and the cases they govern are displayed to the user who is free to make the final determination about the validity of the analysis. By themselves, the features that compose a rule provide implicit advice for improving the team. To further elucidate, a multimedia viewer is used to show cases matching the rule, allowing the user to better understand the situation and to validate the rules (See figure 5). A perturbation analysis is then performed to recommend changes to the team by changing the rule condition by condition and mining cases of success and failure for this perturbed rule. The cases of this analysis are also displayed in the multimedia viewer, enabling the user to verify or refute the analysis.

For the multiple agent key interaction model, patterns of agent actions are analyzed similar to the individual agent actions. A perturbation analysis is also performed here, to find patterns that are similar to successful patterns but were unsuccessful. Both successful patterns and these ‘near misses’ are displayed to the user as implicit advice. This model makes no recommendations, but does allow the user to scrutinize these cases.

The global team model requires a different method of presentation. For the analysis of overall team performance, the current engagement is matched against previous rules, and if there are any matches, ISAAC concludes that the reasons given by the rule were the determining factors in the result of the engagement. A natural language summary of the engagement is generated using this rule for content selection and sentence planning. ISAAC makes use of the multimedia display here as well, linking text in the summary to corresponding selected highlights.
Figure 5: Multimedia viewer highlighting key features
7 Adjustable Autonomy

One of the interesting issues raised by MMS is how the team of satellites interact with ground control. Recall that at various times, the constellation is quite distant from Earth and requires the DSN to communicate (and then only when the orbit takes them close enough high speed data links). This makes communication more costly and harder to schedule. The planned daily uplink of command data has been estimated in one report (1999) to be 100 bytes. Clearly, the MMS satellite will need to exhibit considerable autonomy but nevertheless it is not hard to imagine that system anomalies may occur that require human intervention.

Increasing interest in applications where humans must act as part of agent teams, has led to a burgeoning of research in adjustable autonomy, i.e., in agents that dynamically adjust their own level of autonomy. Essentially, for effective task performance, an agent may act with full autonomy or with reduced autonomy --- harnessing human knowledge or skills when needed, but without overly burdening the humans. The results of this research are both practically important and theoretically significant.

The need for agent teamwork and coordination in a multi-satellite mission leads to critical and novel challenges in adjustable autonomy --- challenges not addressed in previous work, given that it has mostly focused on individual agents' interactions with individual humans. For instance, consider one of the central problems in adjustable autonomy: when should an agent transfer decision-making control to a human (or vice versa). The presence of agent teams adds a novel challenge of avoiding team miscoordination during such transfer.

To get a more concrete sense of the Adjustable Autonomy Issues here, consider a simple example. If the MMS constellation's formation deteriorates beyond some safe bound, the side-effects of making the adjustment may make it undesirable to leave it to the low-level distributed control algorithm (DCA) to make adjustments. There may be more than one way for the individual satellites to adjust with different fuel requirements across satellites, while the satellites may differ in amount of fuel they have. One of the satellites may have a persistent but not detected/diagnosed anomaly in its attitude control that is leading to the formation degradation, which should be factored into the decision-making. The necessary adjustments may also subsequently impact the transformations of the orbits over time, which are part of the planned mission phase transitions. Finally, these factors are happening in some part of the orbit/mission that makes communication with ground more or less feasible in some amount of time.

Clearly, if a single agent were to transfer control for this decision to the human user involved, and the human fails to respond, the agent may end up mis-coordinating with its teammates who may need to act urgently. Yet, given the risks in the decision, acting autonomously may be problematic as well. Clearly, the adjustable autonomy in this context applies to the entire team of agents rather than any individual spacecraft. Further, if the decision is to transfer control, the team could not expect to wait indefinitely for a response from a human operator.

Clearly, the need for real-time response, the serious potential costs of errors, and the inability of the human to directly monitor the state of the different spacecraft add to the complexity of the
adjustable autonomy problem. In addressing such challenges, on-going work in adjustable autonomy will play a critical role.

For example, one approach to avoid team miscoordination due to transfer of control decisions is for an agent to take into account the cost of potential mis-coordination with teammates before transferring decision-making control. For example, if a satellite is having persistent difficulty maintaining formation, one response might be to ask ground control what to do and go into a wait loop waiting for a response. But such a response needs to take into account the miscoordination consequences before it decided to transfer the control decision to ground. This would avoid rigidly committing to a transfer of control decision and allow the craft to continual reevaluating the situation, reversing control and taking autonomous action when needed. This suggests that transfer of control must be more strategic.

7.1 Transfer of Control Strategies

Previous approaches to transfer-of-control were quite too rigid, employing one-shot transfers-of-control that can result in unacceptable coordination failures. Furthermore, the previous approaches ignore potential costs (e.g., from delays) to an agent's team due to such transfers of control.

To remedy such problems, more recent work (ref to Scerri et al) emphasizes the notion of a transfer-of-control strategy. A transfer-of-control strategy consists of a conditional sequence of two types of actions: (i) actions to transfer decision-making control (e.g., from the agent to the user or vice versa) and (ii) actions to change an agent's pre-specified coordination constraints with team members, aimed at minimizing mis-coordination costs. An agent executes such a strategy by performing the actions in sequence, transferring control to the specified entity and changing coordination as required, until some point in time when the entity currently in control exercises that control and makes the decision. When the agent transfers decision-making control to an entity, it may stipulate a limit on the time that it will wait for a response from that entity.

Since the outcome of a transfer-of-control action is uncertain and some potential outcomes are undesirable, an agent needs to carefully consider the potential consequences of its actions and plan for the various contingencies that might arise. Moreover, the agent needs to consider sequences of transfer-of-control actions to properly deal with a single decision. Considering multi-step strategies can allow an agent to attempt to exploit decision making sources that might be too risky to exploit without the possibility of retaking control. For example, control could be transferred to a very capable but not always available decision maker then taken back if the decision was not made before serious miscoordination occurred. More complex strategies, possibly including several changes in coordination constraints, can provide even more opportunity for obtaining high quality input.
7.2 Implications of Strategies

The goal for a transfer of control strategy is for high quality individual decisions to be made with minimal disruption to the coordination of the team. Clearly however there are dependencies. Transfer of control actions, whether they are one-shot or strategies, take time. Further the decision to use a particular transfer of control strategies may not be independent from the other task facing the team and individual craft. This clearly factors in to the question of how adjustable autonomy is realized within the overall software architecture and in particular its relation to supervisory control — a question we return to later.

Of course, one approach to deriving good transfer of control strategies is to conjoin decision-making about adjustable autonomy with the other planning and scheduling decisions. For example, one can operationalize transfer of control strategies via Markov decision processes (MDPs) which select the optimal strategy given an uncertain environment and costs to individuals and teams. Scerri et al. have also developed a general reward function and state representation for such an MDP, to facilitate application of the approach to different domains.

7.3 MMS and AA

Currently, it is not clear to what extent adjustable autonomy will play a major role in MMS. MMS is being planned with an apparent high degree of autonomy. However, it is interesting to note that the costs in time and money of any transfer of control to human operators on the ground will vary over the course of an orbit as well as the phase of the mission. For example, in phases 3 and 4 of the mission, as noted earlier, MMS will be quite distant at times and require scheduling time on the DSN for communication. This would make any interaction with ground more costly and more time consuming. The implication of this is that if Adjustable Autonomy becomes part of the mission design, the transfer of control strategies will be quite different over the course of the mission.
8 Integration

Until now, we have only briefly touched on how the teamwork reasoning and adjustable autonomy reasoning could be folded into each craft's supervisory control procedures. However, the discussions of the underlying decision-making and communication involved in teamwork and adjustable autonomy made it clear that these processes take time. For that reason, they may interact with the scheduling of other tasks. For example, the decision to turn on a sensor could be made autonomously by a craft, negotiated with other craft, transferred to ground or decided by executing some transfer of control strategy. Each of these strategies will have some kind of temporal footprint with potential tradeoffs on whether the conjoined sensing acting succeeds, whether other mission critical tasks are delayed, which tasks need to be performed, how the power levels are impacted and how much the data buffer is filled. The tradeoffs in principle might work both ways. Thus, the teamwork and autonomy decision-making processes may impact the scheduling decisions made by the supervisory control and conversely the scheduling decisions may impact which teamwork strategy is preferred. And overall solution quality may, in fact likely will, depend on the teamwork, autonomy and supervisory control decisions.

This argues for a tight integration of these teamwork and supervisory procedures, for an integration that makes teamwork decisions part of the supervisor's planning, scheduling and execution. Of course, this need for tight, uniform integration is precisely the kind of need that architectures like IDEA, specifically its plan database, are supposed to address. IDEA gives planning decisions first class status in its plan database and it could likewise incorporate teamwork and adjustable autonomy decisions. Thus, one model of a general software architecture for multi-satellite missions like MMS is to integrate the teamwork and adjustable autonomy reasoning into the rest of the decision-making. The constraints that the alternative decision choices impose on each other can then be explicitly reasoned about. For example, in such a system, the planning/scheduling decides which transfer of control strategy to use in concert with decisions being made about other tasks.

An alternative is to treat these decision-making processes as separate modules. Indeed this is often the norm in the design of multi-agent teams. Teamcore in particular is an architecture designed around the assumption that teamwork reasoning can be a distinct module or wrapper around the rest of the agent's individual task reasoning. This approach has many benefits. The separation has no doubt played a key role in the advances made in multi-agent teamwork theory. More pragmatically, the separation provides a strong decomposition that greatly simplifies the software engineering task. It also allows existing agent designs to be wrapped. It would likely work well in many multi-satellite missions. But it does, by design, enforce a separation between individual task reasoning and team task reasoning. If the tradeoffs between these tasks are inconsequential, then there needs to be someway to make those tradeoffs explicit in the interactions between decision modules. For example, supervisory control might communicate to the teamwork reasoning various time windows available to make a decision along with its estimate of their impact on solution quality. The teamwork reasoning module would make a decision on the appropriate coordination strategy based on this information and its own
estimates. One might also imagine some form of iterative communication between a single
craft’s modules, or even negotiation, to come to a joint decision.

A third alternative is arguably more radical. We mention it here only since we earlier discussed
decentralized POMDPs as a framework of analysis. This naturally raises the question of why not
consider them for synthesis. In this approach, there is no flexible supervisory control and no
teamwork reasoning module. Rather a decentralized policy is derived for all the craft. Each
craft’s software simply implements that policy that drives their behavior based on the history of
their observations. The individual craft sense, communicate, make attitude adjustments, uplink
and downlink because their individual policies informed them to perform these actions. We do
not envision this approach being feasible for anything but perhaps the shorter, simpler missions.
Given the complexity of generating Dec-Pomdp algorithms, it may not be feasible to derive the
policy for longer, more complex missions in the first place. Further, the probabilistic models for
state transitions, observations, etc. are not known with sufficient accuracy to entrust mission
success to them. The policies themselves may be too large to store on board. Arguably most
important is the fact that there are alternative approaches with well-demonstrated track records.
IDEA is the follow-on to Remote Agent which has mission experience. STEAM has been used in
many applications where it is has demonstrated its robustness and has even evaluated in several
domains within the COM-MTDP framework where it has demonstrated that it can provide a
cheaper-to-compute good approximation to optimal performance.
9 Recommendations

It is a difficult challenge to design a team of agents that can coherently and efficiently pursue common goals in dynamic, uncertain environments. Indeed, the magnitude of the challenge is often underestimated. However considerable progress has been made by the multi-agent research community in understanding this challenge, designing teamwork algorithms and implementing agent teams. Clearly, this research could play an important role in facilitating the development of NASA multi-satellite missions. As has been noted throughout this paper, the application of this research to NASA missions like MMS raises several issues and opportunities. In this conclusion, we summarize these issues and make suggestions for future directions.

NASA is embarking on a wide range of ambitious multi-satellite mission designs. By establishing FFTB and DSS, NASA has already recognized and acted on the pressing need for systematic evaluation and experimentation of any distributed spacecraft system. This presents a clear opportunity for NASA and the multi-agent research community to collaborate. In particular, models of teamwork reasoning could be part of this experimentation. Without such models, key questions about satellite coordination and performance will remain unanswered. Incorporating a teamwork module would be relatively straightforward. Indeed, there are no technological barriers to incorporating Teamcore into DSS since Teamcore, like FFTB and the DSS system, is designed to be a modular component.

In particular, the formal MTDP work can and should play a key role in analyzing designs for distributed satellite missions. These formal frameworks will likely have a major impact on multi-agent research. For example, the best-case, worst-case and average case analyses they support will be a critical part of any real-world, high-cost application of multi-agent systems. In terms of NASA missions, the formal analyses could be performed, rapidly, outside of DSS, resulting in tested and improved teamwork prescriptions that would then be tested inside of DSS. Alternatively, a hybrid approach might be feasible where some of the probabilistic functions of the MTDP framework are realized by software modules that are part of the DSS.

Note, as discussed earlier, we envision that the main role for MTDP to be in the analysis of algorithms or informing the design of new algorithms, as opposed to synthesis of MTDP policies as a replacement for existing algorithms.

As a first step towards applying this MTDP framework to the problem of designing better satellite teams, we would propose to cast an example NASA satellite constellation problem, specifically MMS, into the MTDP framework. This will allow us to evaluate alternative approaches to role replacement and adjustable autonomy eventually and contrast them with optimal policies. We also envision that an ISAAC-like tool that incorporates the MTDP framework could be readily incorporated into the DSS environment. To fully exploit the potential of the MTDP work, research is needed to develop efficient algorithms for finding approximately optimal policies.

As NASA embarks on developing multi-satellite missions, we believe it is important to explore
general approaches to teamwork reasoning and analysis from the start. We believe this is true even in early multi-satellite missions that may seemingly require minimal teamwork coordination. For example, it may seem that a mission like MMS is simple enough that it does not require general architectures for teamwork or extensive analysis of alternative coordination schemes. However, ad hoc coordination schemes that address specific coordination tasks as special cases are too brittle. This conclusion has come to the multi-agent community through hard-earned experience. Quite simply, human designers cannot think of every way coordination can break down, so there is always another special case rule to add. Further, it ends up being more time consuming and costly to come up with the host of ad hoc rules. Finally, by incorporating general teamwork reasoning and analysis early on, these initial missions could lay critical groundwork that could be exploited in later more ambitious missions.
10 Acknowledgements

I would like to thank NASA Ames researcher, Dr. Kanna Rajan, for his key help and support of this project. I would also like to thank NASA Goddard researchers Dr. Jesse Leitner, Dr. Russell Carpenter and Dr. John Bristow for their time and input. Dr. David Pynadath, provided critical input to this report. In addition, my colleagues at ISI, Dr. Milind Tambe, Dr. Paul Scerri and Ranjit Nair for their suggestions and critiques.
11 Bibliography


Nicola Muscettola, Gregory A. Dorais, Chuck Fry, Richard Levinson, and Christian Plaunt, QSS A Unified Approach to Model-Based Planning and Execution

J. Russell Carpenter, David C. Foltu, and David A. Quinn. Integration of Decentralized Linear-Quadratic-Gaussian Control into GSFC'S Universal 3-D AUtonomous Formation Flying Alogorithm, AIAA-99-4269.


Steve Chien, Rob Sherwood, Michael Burl, Russell Knight, Gregg Rabideau, Barbara Engelhardt, Ashley Davies, Paul Zetocha, Ross Wainwright, Pete Klupar, Pat Cappelaere, Derek Surka, Brian Williams, Ronald Greeley, Victor Baker, James Doan. The Techsat-21 Autonomous Sciencecraft Constellation.


Ranjit Nair, Milind Tambe, Stacy Marsella and David Pyndath: "Model for Team Formation for Reformation in Multiagent System", AAAI Workshop on Coalition Formation, 2002.


Ranjit Nair, Milind Tambe, Stacy Marsella and David Pynadath: "Formalizing Team Formation and Reformation in Multiagent Systems", AAMAS Workshop on Teamwork and Coalition Formation, 2002.


Tambe, M. 1997 Towards Flexible Teamwork Journal of Artificial Intelligence Research, Volume 7, Pages 83-124
NASA is rapidly moving towards the use of spatially distributed multiple satellites operating in near Earth orbit and Deep Space. The satellites will be required to cooperate with each other as a team that must achieve common objectives with a high degree of autonomy from ground based operations. Such satellite teams will be able to perform spatially separated, synchronized observations that are currently not feasible in single satellite missions. Autonomous operations will reduce the need for ground-based support that would otherwise be prohibitively expensive in such missions. However, the underlying control systems necessary to enable such missions will raise many new challenges in autonomous, multi-platform operations.

In particular, a critical requirement for these satellite constellations is that they must act coherently as a coordinated, at times autonomous team, even in the face of unanticipated events such as observation opportunities or equipment failures. Further, the satellites will need to take actions that will not only impact the constellation’s current tasks but may also impact subsequent tasks, an issue that is particularly relevant given the often long duration of some missions and the limited power and fuel resources available to each satellite. Overall, the ability to operate as a team will need to be satisfied in many of the multi-satellite missions being planned. Therefore, it is important to understand this requirement, elucidate the research challenges it presents and consider approaches to satisfying it.

The multi-agent research community has made considerable progress in investigating the challenges of realizing such teamwork. In the full report, we discuss some of the teamwork issues that will be faced by multi-satellite operations. In particular, we discuss the Magnetospheric Multiscale mission (MMS) to explore Earth’s magnetosphere. We describe this mission and then consider how multi-agent technologies might be applied to improve the design and operation of such missions.

Specifically, the report illuminates several basic issues. It discusses the need to develop robust and effective coordination techniques for multi-satellite teamwork. Rather than mission-by-mission ad hoc approaches to coordination, we focus on a general approach to teamwork that will be both more robust in a particular mission while also building across mission, teamwork-
technology infrastructure. We also stress the need for analysis and suggest a formal approach to assessing the quality of alternative coordination techniques, based on the MTDP (Multi-agent Team Decision Problem) framework that allows both formal and empirical evaluation. We illustrate how this approach could be applied to MMS’s science operations and discuss how it could be extended to provide a faithful rendering of difficult resource limits that such missions will operate under. In addition, we discuss alternatives to realizing the teamwork reasoning and how teamwork and autonomy is integrated into a craft’s overall software architecture.

MTDP provides a tool to address a range of analyses critical to fielding teams in real world applications. Using the MTDP framework, the complexity of deriving optimal teamwork policies across various classes of problem domains can be determined. The framework also provides a means of contrasting the optimality of alternative approaches to key teamwork issues like role replacement and communication. Finally, the framework allows us to empirically analyze a specific problem domain or application of interest. To that end, a suite of domain independent algorithms has been developed in prior work that allows a problem domain to be cast into the MTDP framework. This allows the empirical comparison of alternative teamwork approaches in that domain. Derivation of the optimal policy for the problem domain serves not only as the basis of comparison but also can inform the design of more practical policies. Most recently, progress is being made in addressing how real world operating constraints like power consumption can be modeled in this framework.

But of course, teamwork and autonomy reasoning are just one part of the multi-satellite team’s operation, which must include various flying, observation, communication and maintenance tasks over the duration of the mission. So, the report also discusses the supervisory control software that manages and schedules these tasks. In particular, we discuss one approach to the design of this supervisory software and the integration of teamwork reasoning within this supervisory control software.

The report makes several recommendations for the future of the research and also potential collaborations with NASA. In particular, it is suggests that the formal MTDP work could play a key role in analyzing designs for distributed satellite missions. MTDP formalisms could be used in the analysis of algorithms or informing the design of new algorithms. For example, the best-case, worst-case and average case analyses that the MTDP models support could be of critical assistance in the design and development of any real-world, high-cost application of multi-agent systems. In terms of NASA missions, the formal analyses could be performed entirely within the MTDP framework, resulting in tested and improved teamwork prescriptions. Alternatively, the MTDP framework could be realized by software modules that are incorporated into ongoing NASA Goddard work in distributed satellite simulation.

As a first step towards applying this MTDP framework to the problem of designing better satellite teams, we propose to cast an example NASA satellite constellation problem, specifically MMS, into the MTDP framework. This will allow us to evaluate alternative approaches to teamwork and adjustable autonomy as well as contrast them with optimal policies.

As NASA embarks on developing multi-satellite missions, we believe it is important to explore general approaches to teamwork reasoning and analysis from the start. We believe this is true even in early multi-satellite missions that may seemingly require minimal teamwork
coordination. For example, it may seem that early missions will be simple enough that they will not require general architectures for teamwork or extensive analysis of alternative coordination schemes. However, ad hoc coordination schemes that address specific coordination tasks as special cases are too brittle. This conclusion has come to the multi-agent community through hard-earned experience. Quite simply, human designers cannot think of every way coordination can break down, so there is always another special case rule to add. Further, it ends up being more time consuming and costly to come up with the host of ad hoc rules. Finally, by incorporating general teamwork reasoning and analysis early on, these early multi-satellite missions could lay critical groundwork that could be exploited in later even more ambitious missions.