NASA Contractor Report 189606

P. 347

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

Christopher J. Slominski
Mark A. Parks
Kelly R. Debure
William J. Heaphy

Computer Sciences Corporation
Hampton, Virginia

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

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

(NASA-CR-189606) ADVANCED TRANSPORT OPERATING SYSTEM (ATOPS) CONTROL DISPLAY UNIT SOFTWARE DESCRIPTION (Computer Sciences Corp.) 347 p

N92-24689

CSCL 09B

Unclas

G3/06 0086862

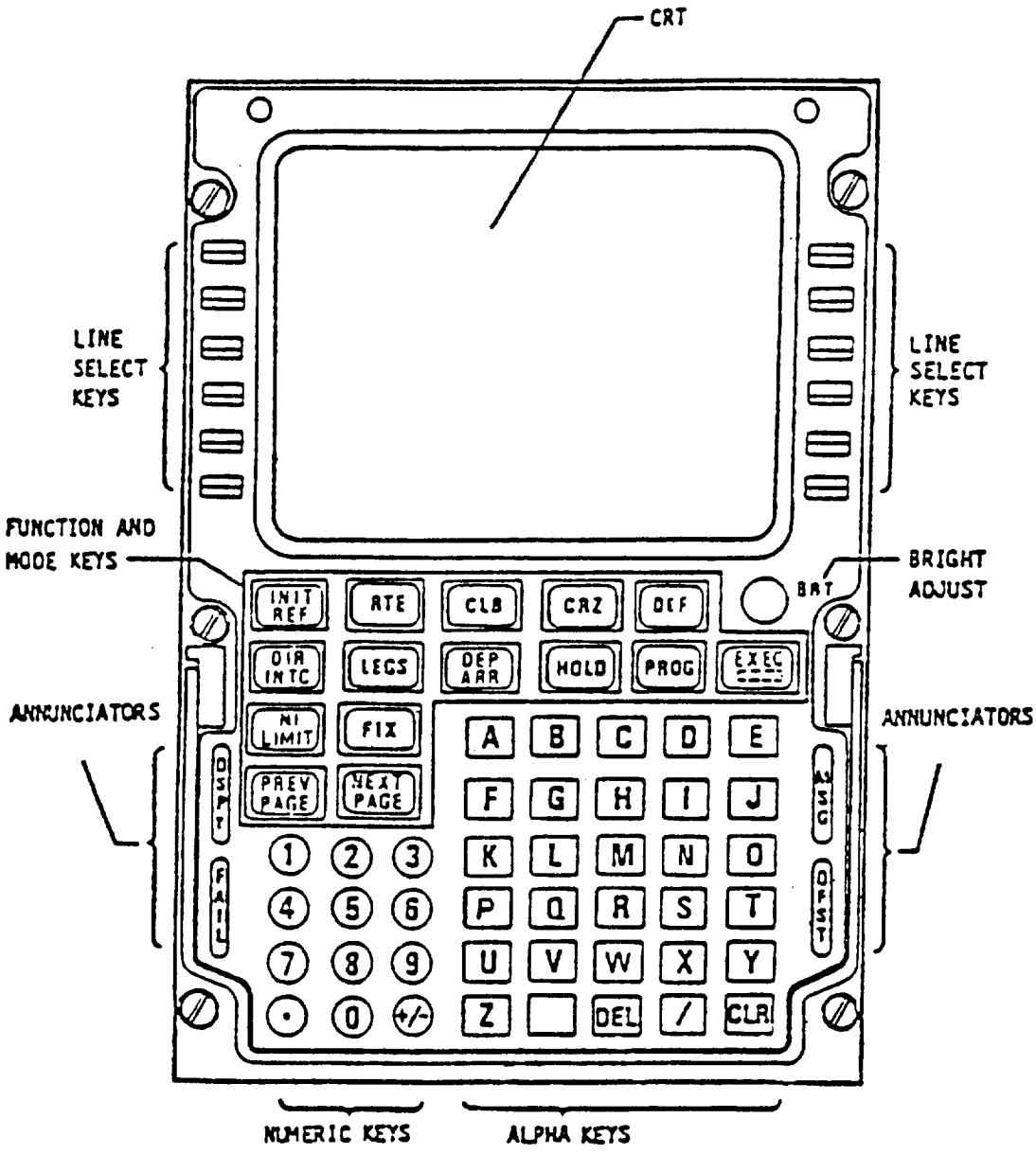
LIST OF FIGURES

Section 1.0 INTRODUCTION TO CDU SOFTWARE

The following sections of this document describe the CDU software which runs on the Flight Management/Flight Controls VAX computer on-board the TSRV. All the software, with the exception of two small modules, is built into the flight management background process SLOW. The remaining modules, CDUFST and KEYBRD, serve as the CDU's foreground interface and are built into the processes FMFAST and HDL respectively. CDU applications running in the background means that no definitive timing exists for the repetitive scheduling of CDU operations and the software may be interrupted at any point by time critical foreground software. Data structures shared by both background and foreground must be synchronized through software flags.

Two important functions of the CDU software include the management of the CDU interactive display and the flight management functions performed in assisting the flight crew in choosing and following a flight plan. Operations performed with CDU software affects the aircrafts guidance, navigation, and display software. The actual CDU hardware is a Lear-Siegler unit having 14 display lines of 24 character width. There are also 5 programmable display lights on the face of the keyboard. Besides alphanumeric data keys, there are six line select keys (LSK) on each side of the display area. See figure 1.0.

PAGE 8 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED



(figure 1.0)

PRECEDING PAGE BLANK NOT FILMED

Section 2.0 CDU INPUT DATA

Input to the CDU comes from two sources. Most CDU data is received from flight crew entries on the Lear-Siegler keyboard, however data input for the CDU software may also come from the data-link. Information on data link I/O is contained in the CDU data-link description (section 5).

Keyboard entries received by CDU software are of two types; function and buffered data. Function entries consist of one key code (one byte of data) while data entries have one to 16 bytes of ASCII data followed by a termination key code. The termination code is from either a line select key or the sampled scratch pad code, 'FF' hex. The key codes sent by the Lear-Siegler unit are non-standard character codes which must be translated into usable data for CDU software. The module KEYBRD performs the translation upon receiving the data in the I/O handler process (HDL). The alphanumeric codes are mapped into their ASCII counterparts for ease of use in the software. Figures 2.0 and 2.1 show the codes both before and after the translation process.

Once the code translation is complete the data is stored in the global input buffer, ENTRY. The first byte of ENTRY is set to the key code count. All function entries will have the count byte set to one, data entries will be from two to seventeen. Note that when the CDU applications have finished processing the keyboard input, the count byte is cleared.

Data entry is initiated by any alphanumeric keystroke. At that time the CDU will automatically clear all of line #14 (CDU data entry line), then echo the character at the start of line #14 (the CDU is now in data entry mode). During data entry mode any line #14 update sent to the CDU from the host computer will be ignored. When a function key is selected during data entry mode the function code will be immediately sent to the host, with no effect on the current scratch pad entry. The CLR function will not be passed to the host computer unless the scratch pad is inactive. When data entry is in progress, the CLR will be used by the CDU to either delete one or all characters from the scratch pad depending on duration of selection. When all characters are deleted from the scratch pad the CDU will exit data entry mode and allow line #14 updates. When data entry is completed by a LSK selection the scratch pad line is cleared and data entry mode is canceled.

Data buffering by the CDU may be disabled by the host software. When this situation arises the CDU scratch pad will be cleared and disabled. Neither direct key entry nor host software scratch pad programming will place the CDU in data entry mode. Note that line #14 of the display screen will always be available as a display line when data buffering is disabled. All key entries, including line select keys, will be sent immediately as single key function entries (count = 1). This process will continue until the host computer commands the re-enabling of data buffering.

PAGE 12 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED

Figure 2.0 CDU UNTRANSLATED KEY CODES

KEY	CODE	KEY	CODE	KEY	CODE
	00H	K	20H	O	40H
	01H	P	21H	T	41H
	02H	U	22H	Y	42H
	03H	Z	23H		43H
LSK L2	04H	F	24H	J	44H
LSK L1	05H	A	25H	E	45H
LSK R1	06H	LEGS	26H	EXEC	46H
LSK R2	07H	RTE	27H		47H
1	08H	L	28H	CLR (1)	48H
4	09H	Q	29H	PREV PAGE	49H
7	0AH	V	2AH	NI LIMIT	4AH
.	0BH	(BLANK)	2BH		4BH
LSK L4	0CH	G	2CH		4CH
LSK L3	0DH	B	2DH		4DH
LSK R3	0EH	DEP ARR	2EH		4EH
LSK R4	0FH	CLB	2FH		4FH
2	10H	M	30H		
5	11H	R	31H		
8	12H	W	32H		
0	13H	DEL	33H		
LSK L6	14H	H	34H		
LSK L5	15H	C	35H		
LSK R5	16H	HOLD	36H		
LSK R6	17H	CRZ	37H		
3	18H	N	38H		
6	19H	S	39H		
9	1AH	X	3AH		
+/-	1BH	/	3BH		
NEXT PAGE	1CH	I	3CH		
FIX	1DH	D	3DH		
DIR INTC	1EH	PROG	3EH		
INIT REF	1FH	DES	3FH	CLR (1)	C8H

NOTE (1): CLR KEY CODE FOR KEY ENGAGED
< 1/2 SEC 48H, > 1/2 SEC C8H

Figure 2.1 CDU TRANSLATED KEY CODES

HEX VALUE	KEY
00	
01-0C	line select 1-12
0D-0F	
10	INIT REF
11	DIR INTC
12	N1 LIMIT
13	RTE
14	LEGS
15	FIX
16	CLB
17	DEP ARR
18	CRZ
19	HOLD
1A	DES
1B	PROG
1C	PREV PAGE
1D	NEXT PAGE
1E	EXEC
1F	
20	(blank)
21	short CLR
22	long CLR
23	DEL
24-2C	
2D	-
2E	.
2F	/
30-39	0-9
3A-3F	
40	
41-4F	A-O
50-5A	P-Z
5B-5F	
60-FE	
FF	scratch pad terminater

Section 3.0 CDU OUTPUT DATA

The CDU display screen consists of 14 lines of 24 characters each. The top and bottom lines are referred to as the title and scratch pad lines respectively. The title line identifies the active CDU display page and the scratch pad line is alternately used as a data entry and warning display line. The lines in between are identified as line #1 through #12. Typically odd numbered lines are used as label lines where text is written in small font. The even numbered lines except #12 are normally used as data entry and display lines. Line #12 often has special control tags such as "ERASE>". The six line select keys on each side of the display correspond to label/data line pairs. For example the top LSK is positioned between lines #1 and #2.

The data transmission to the Lear-Siegler Control Display Units is a variable length byte stream consisting of character codes (00H - 7FH) and special functions (80H - FFH). The visible representation for each character code is shown in figure 4, page 26 of the Design Requirement Specification for the CDU. Only a subset of the existing symbols is used by CDU software. Figure 3.0 outlines the symbols and their hex codes used for the NASA CDU software. The minimum amount of data that can be modified in one update is one 24 character line on the display. However any number of lines may be updated at once. The CDU software sets the flag IOWAIT when a block of data is complete. CDU software remains idle until the I/O handler process transmits the data to the CDU I/O processor (CVIU) and clears the flag.

The utility procedure FMTOUT is used to build the CDU output buffer. This module inserts the special control codes into the data stream for the applications software when called with the various parameters available. The next section describes the use of FMTOUT.

The remainder of this section describes the special control codes placed in the output buffer. The sign bit of all function codes is set, therefore CVIU software parsing the transmitted data can quickly identify leading, trailing, and embedded functions. The high order nibble of a function byte is the function identifier and the low order nibble is the function qualifier. Therefore there are eight distinct CDU functions (8xH - FxH), each having 16 qualifiers.

The following pages describe each of the defined function identifiers and the effect of the various qualifiers.

PAGE 16 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

FUNCTION "8x" (1000---- binary); CLEAR LINE

This function is used to blank a line on the CDU display. The qualifier bits designate which line is to be cleared. Since there are 14 display lines on the CDU screen valid values for this function are 81H through 8EH.

The count function "Ax" is placed immediately following the clear line function to blank a number of contiguous lines of the display.

The entire screen can be cleared by the two bytes "81H, AEH".

FUNCTION "9x" (1001---- binary); UPDATE LINE

This function is used to replace all 24 characters on a CDU display line. The qualifier bits designate the line which will be updated. The count function "Ax" can follow the update function to replace a number of consecutive lines. Valid values for this function are 91H through 9EH.

Directly following the updated function, or the count function if supplied, are the ASCII character bytes used to fill the designated line(s). For example, the following 25 bytes place an ASCII zero, "0", in each character position of line number three.

93H,30H,30H,.....30H

FUNCTION "Ax" (1010---- binary); LINE COUNT

This function is used to make the clear and update functions (8x and 9x) work over a range of display lines. The count function is valid only when immediately succeeding the other two functions in the data stream.

The valid set of values for this function are A1H through AEH.

FUNCTION "Bx" (1011---- binary); sample scratch pad

This function requests immediate sampling of the CDU scratch pad. The qualifier bits are undefined for this function. When the "Bx" function is received any current data entry is terminated and sent to the host computer as if a LSK was pressed by the pilot. The scratch pad is cleared and data entry mode is disabled. The termination byte, normally the selected LSK code, will be FFH.

When no data exists on the scratch pad just the terminator code is sent just like an LSK press with no data (ie count byte = 1).

FUNCTION "Cx" (1100---- binary); SET MODE

The mode function handles several miscellaneous CDU operations. In particular there are eight mode commands (C0H - C5H, CEH, CFH) which are described below.

- C0 -

This code is the end of transmission byte which is always the last byte of the data block.

- C1 -

Mode qualifier "1" causes the CDU to be initialized. After this byte is processed the display screen is clear, all lights are off, video is standard, data entry is disabled and data buffering is enabled.

- C2 -

Sets standard video. All text written to the CDU after receiving this function will have the standard video characteristic. Note that this code may be imbedded within an ASCII text string.

- C3 -

Sets reverse video. All text written to the CDU after receiving this function will have the reverse video characteristic. Note that this code may be imbedded within an ASCII text string.

- C4 -

Disables CDU data buffering. Keystrokes will be sent immediately to the host computer as function entries.

- C5 -
Enables CDU data buffering. Data may be entered on the scratch pad by manual entry or software programming.

- CE -
Selects pilot's CDU. This function (or CF) must always be the first byte of the data sent to the CDU. This byte is always followed by the CDU "lights" byte described below.

- CF -
Selects co-pilot's CDU. This function (or CE) must always be the first byte of the data sent to the CDU. This byte is always followed by the CDU "lights" byte described below.

CDU lights byte:
This byte is always the second byte of a transmission from the host computer. The low-order 5 bits represent the desired status of the CDU lights (bit set = light on). The bits are assigned as follows.

0	FAIL
1	DSPY
2	MSG
3	OFST
4	EXEC

1) Note that when the MSG light is on, no scratch pad entry may be started by either keyboard entry or scratch pad programming with function "Dx". Any entry on the scratch pad when MSG is set on can be finished and transmitted with a LSK.

FUNCTION "Dx" (1101---- binary); SCRATCH PAD UPDATE

Function D is used to place a text string into the scratch pad as if it had been manually entered via the keyboard. The qualifier bits indicate the number of characters in the update string (offset by one; 0 means 1, F means 16). Note the string of characters immediately follows the function byte.
Valid values for this function are D0H - DFH. The three bytes given below would clear the scratch pad of any existing entry and place the text "10" into the scratch pad area. Note that the CDU will be in data entry mode after receiving a "D" function.

D1H, 31H, 30H

Figure 3.0 , CDU OUTPUT CODES

00-0F	
10-19	small font digits (0-9)
1A-1F	
20-22	standard ASCII
23	degrees F
24	degrees
25	standard ASCII
26	degrees C
27-3F	standard ASCII
40	box
41-5B	standard ASCII
5C	
5D	standard ASCII
5E	
5F	standard ASCII
60	
61-7A	small font alphabet
7B	
7C	standard ASCII
7D-FF	

Section 3.1 CREATING OUTPUT WITH FMTOUT CALLS

Background CDU software creates a block of data to refresh part or all of the CDU display screen with calls to FMTOUT. The format of the call is as follows:

```
INTEGER*2 PAD, LENGTH, CODE
BYTE STRING(*)

CALL FMTOUT(PAD, STRING, LENGTH [, CODE] )
```

Each call appends data to the current output buffer being built for transmission to the CDU. The display codes at "STRING" are added to the current line after padding with "PAD" blanks. Note that all 24 characters of a line do not need to be supplied. FMTOUT will extend all short lines with blanks anytime a short line is terminated. The optional code parameter is an integer value with several defined bit fields. CODE is used to designate the start of a new line, enable reverse video, program the scratch pad, clear the screen, send special function codes, or terminate the buffer to cause transmission.

There are predefined symbols used to create the CODE word. The individual symbols must be added together to produce the final integer parameter.

To send a literal string to FMTOUT use %REF() or a Hollerith constant.

LINE0 - LINE13	starts new line
VIDEO -	string written in reverse video
SCRTCH -	initialize scratch pad to string
EOT -	finished updating current buffer
CLS -	clear CDU screen
FCTN -	string consists of special functions

The following example code segment creates one complete update for the CDU. The change consists of new top and bottom lines on the screen. The top line will have the text "EXAMPLE PAGE" preceded by 2 blanks and followed by 10. The bottom line will have the text "HELLO THERE FRED" followed by 9 blanks. Note that FRED will be written in reverse video.

```
CHARACTER*12 TITLE
INTEGER*4 USER

TITLE = 'EXAMPLE PAGE'
USER = 'FRED'
CALL FMTOUT(2, %REF(TITLE), 12, LINE0)
CALL FMTOUT(0, %REF('HELLO THERE'), 11, LINE13)
CALL FMTOUT(1, USER, 4, VIDEO + EOT)
```

PAGE 22 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED

Section 4.0 CDU EXECUTIVE

This section contains the module descriptions for CDU executive software. The executive software performs miscellaneous functions that are independent of the currently displayed CDU page. There are five modules described in this section. The remaining executive modules are associated with the data-link portion of the CDU and are described in section 5. The majority of the CDU executive sofware, including data-link, is found on the file CDUEXC.FOR.

PAGE 24 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: CDUEXC
FILE NAME: CDUEXC.FOR
PROCESS: SLOW
CALLED BY: SLOW
CALLING SEQUENCE: CALL CDUEXC

PURPOSE:
To manage those CDU functions which are independent of the current CDU display page.

DESCRIPTION:
This module performs several miscellaneous operations for the CDU software. Since most sections of the module are unrelated, the operations are simply itemized below in the order found in CDUEXC.

- Cause transmission of the CDU initialize code on start up of the software.
- Inhibit all CDU software until the I/O handler has completed last output.
- Initialize output buffer with the predifined start byte and the CDU lights byte.
- Call MESSAGE_MGR upon detecting data-link inputs.
- Call EXEC_FCTN to handle special CDU function entries not destined for specific page manager sofware.
- Compute the barometric pressure altitude correction value and issue baro-set alert when traversing the 18,000 foot threshold.
- Call active page manager software.
- Perform auto-update of waypoints every ten seconds when required by 'POS' type waypoints. Calls UPDATE_POS.
- Manage "North-up" map display center position.
- Place appropriate error messages into CDU output buffer when problems detected by the various page managers. Errors will be placed in the buffer each time a new one is generated until the CLR key has been pressed to acknowledge the error. At that time the error message is replaced with the original scratch pad entry which caused the error. Warning messages are only sent out one time. Acknowledgement by CLR entry is not required for these. See figure 4.0 at the end of this section for error codes and their associated message.

GLOBAL REFERENCES:

VARIABLES
  ALTCOR BARSET* BARSFT* CDUCNT* CDU_INIT* CDU_MODE CTRF*
  ERCODE* IOWAIT LT_DSPY LT_EXEC LT_FAIL LT_MSG* LT_OFST
  MODCNT PGINIT* PGRQST* POSTIME* TIME TIME_VLD

ARRAYS
  ENTRY LOKWPT MESSAGE OLDPAGE*

RECORD ARRAYS
  WPT_MOD

FUNCTIONS AND SUBROUTINES
  EXEC_FCTN FMTOUT LINK_CMD MESSAGE_MGR SELECT UPDATE_POS

MODULE NAME: EXEC_FCTN
FILE NAME: CDUEXC.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL EXEC_FCTN(SAVE, ER_FLAG)

PURPOSE:
To process CDU function entries not intended for the current CDU page display module.

DESCRIPTION:
When CDUEXC receives a function entry it calls EXEC_FCTN to determine if the entry is the type handled by the executive. If not, EXEC_FCTN returns and the entry is used by the current page display module.
The types of function entries handled by EXEC_FCTN are listed below along with a brief description of the actions taken.

. CDU page selection; The function keycode is used as an index into a page ID array and placed in the page change request variable (PGRQST).

. Clear key (long or short press); If an error message is displayed it is replaced with the data entry which caused the error by a call to RECALL. The message light is also turned off. When no error message is shown this function simply blanks the bottom CDU line.

. Execute key; When the execute light is not on this function is ignored. Otherwise if there is a provisional flight plan it is made active by a call to EXECUTE. When neither condition is true the execute function is assumed to be handled by the current page display software.

. If none of the above were true and an error message is currently displayed then the entered function (LSK, PREV/LAST page, or DEL) is ignored.

. Delete key; The scratch pad line is programmed with the word "DELETE". Typically this text will be placed at a particular display line with a LSK to designate the deletion of a certain CDU item.

GLOBAL REFERENCES:

VARIABLES
ERCODE LT_EXEC PGRQST* PMODE

ARRAYS
CDUBUF* ENTRY*

FUNCTIONS AND SUBROUTINES
EXECUTE FMTOUT RECALL

MODULE NAME: RECALL
FILE NAME: CDUEXC.FOR
PROCESS: SLOW
CALLED BY: EXEC_FCTN, MESSAGE_MGR
CALLING SEQUENCE: CALL RECALL(SAVED_ENTRY)

PURPOSE:
To recall erroneous data entry.

DESCRIPTION:
When CDU entries cause error message displays the erroneous entry is saved in a buffer "SAVE". The entry is programmed back into the scratch pad by RECALL. Note that when no text exists for reprogramming (function entry error for example) the only action is to clear the bottom CDU display line.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
FMTOUT

MODULE NAME:	SELECT
FILE NAME:	SELECT.MAR
PROCESS:	SLOW
CALLED BY:	CDUEXC
CALLING SEQUENCE:	CALL SELECT(PAGE_ID)
CALLS TO:	DSP_DUMP, INITREF, IDENT, INITPOS, PFINIT, TKOFF, APPREF, NAVPG, STATPG, ROUTE, CLIMB, CRUISE, DESCENT, LEG_MGR, RTENDX, EPRLIM, PROGRESS, INTC_MGR, DEPARR, FIX_INFO, HLD_MGR, LEG_TIME, TEST, RTENDX

PURPOSE:

Call the appropriate page manager subroutine.

DESCRIPTION:

The variable "PAGE" contains the index of the current CDU display page. During each iteration of the CDU executive, the module SELECT is called to perform the corresponding call to a page manager listed in a local address table. Note that the values for "PAGE" have predefined symbolic names assigned in the file CODES.CDU.

GLOBAL REFERENCES: none

MODULE NAME: UPDATE_POS
FILE NAME: CDUEXC.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL UPDATE_POS

PURPOSE:
To update the "POS" type waypoint with current aircraft parameters.

DESCRIPTION:
A provisional flight plan may begin with a "POS" type waypoint which does not become stationary until the flight plan has been executed. The position, altitude, and speed of the pilot defined waypoint are updated every 10 seconds to the values of the aircraft. The module DEMODE is called to incorporate the changes into the provisional flight plan.

The variable POSTIME is set to the update time by CDUEXC. During every iteration, POSTIME is compared to the current aircraft time to check for 10 seconds elapsed. When this occurs the call to UPDATE_POS is made. Note that a POSTIME value of zero corresponds to no "POS" waypoint to update.

GLOBAL REFERENCES:

VARIABLES
ALTCOR GS LAT LON

RECORD ARRAYS
PPT_WPT* WPT_MOD

FUNCTIONS AND SUBROUTINES
DEMODE FIND_PPT

ERROR CODES AND MESSAGES

1) INVALID DATA ENTRY
2) CHECK AIRFIELD
3) NOT FOUND IN MEMORY
4) BUFFER OVERFLOW
5) DEAD KEY ERROR
6) ENTRY WPT NOT DEFINED
7) INVALID EXIT WPT
8) BAD RADIUS AT XXXXX
9) NO DATA
10) DEAD WAYPOINT ERROR
11) BAD DATA FORMAT
12) ENTRY OUT OF RANGE
13) INVALID DELETE
14) ILLEGAL ASSIGNMENT
15) FIX ALREADY SPECIFIED
16) NO ABEAM RADIAL

EXAMPLES

LSK can be used for neither data nor function entries	5
LSK cannot be used for data entries (function OK)	1
LSK cannot be used for function entries (data OK)	9
NEXT or PREV cannot be used	5
Programmed DELETE unacceptable	13
Allowed data entry is unacceptable because ...	
. unrecognizable data	11
. recognized data can't be used because	
data base search failure	3
below or above acceptable value range	12
specific value improper in context	14
. codes 2, 4, 6, 7, 8, 10, 15 & 16 are for specific cases	

-figure 4.1-

Section 5.0 CDU DATA-LINK

One method of input to CDU software is through the TSRV data-link. This method is used to receive clearance information sent by ground controllers. The CDU also is used for data-link outputs when composing messages and sending clearance requests to the ground station.

Two blocks of memory are allocated for data-link I/O in the global section IPLCOM. The data at these locations is transmitted between the FM/FC VAX and the data-link computer every 50 milliseconds. The input area consists of 102 bytes of memory. The first 2 bytes are labeled CDU_CMD and are used as a bit control word for commands from the data-link computer. The remaining 100 bytes (LINK_IN) may contain a text string uplinked from the ground station. The memory allocated for output is a 202 byte block. The first 2 are bytes used as a control word to be sent to the data-link computer to describe the text data stored in the remaining 200 bytes. The first word is labeled MSG_CNT and the text block is CDU_MSG. MSG_CNT uses the low byte as a character count of the data in CDU_MSG. MSG_CNT is not updated until the CDU background software has completed the entire text buffer. The high byte of MSG_CNT is used to control the use of the text buffer by the data-link computer. When composing a text message or sending the current provisional flight plan to the data-link computer this byte is zero. When the processing of a new clearance sent by the data-link computer is complete it will be set to FFH, unless an error in the uplinked clearance was detected. With a clearance error the byte will contain the character count of an error message appended to the text buffer. The total length of the text buffer is then the sum of the low and high bytes of MSG_CNT.

The CDU executive calls the module LINK_CMD each iteration of the background process to check for any data link commands in CDU_CMD. The variable CDU_MODE is set by LINK_CMD to signal MESSAGE_MGR (called by CDUEXC) to initiate message handling by the various CDU data-link modules. The remainder of this section contains module descriptions for all the CDU data-link procedures.

MODULE NAME: ADD_PLAN
FILE NAME: LINK.FOR
PROCESS: SLOW
CALLED BY: LINK_RT
CALLING SEQUENCE: CALL ADD_PLAN(WPT_NAME)

PURPOSE:
To prepare the provisional flight plan for the insertion of the new data-link clearance waypoints.

DESCRIPTION:
This procedure is called with the name of the first waypoint of the new clearance. The current flight plan is searched for a match of the input waypoint. If it is not there the old flight plan is removed and the input waypoint is made the first of the new plan. When the waypoint does exist in the current flight plan all waypoints after it are removed. Note that the waypoint may be part of a route function, in which case the procedure must make the input waypoint the new exit waypoint of the route function.

GLOBAL REFERENCES:

VARIABLES
ACTCNT ACTION*

ARRAYS
ENTRY* RTE_CNT*

RECORD ARRAYS
RTE_MOD* WPT_MOD

FUNCTIONS AND SUBROUTINES
BOUNDS FILL_RTE RTE_WPT WPT_ADDR

MODULE NAME: BEG_RTE
FILE NAME: LINK.FOR
PROCESS: SLOW
CALLED BY: LINK_PD
CALLING SEQUENCE: CALL BEG_RTE(RTE_BUFFER_INDEX)

PURPOSE:
To prepare the provisional route buffer for proceed direct assignment.

DESCRIPTION:
BEG_RTE modifies the provisional route buffer so that the entry indicated by the input parameter RTE_BUFFER_INDEX becomes the second element of the route buffer. To do this it may eliminate elements, open a new slot at the start, or simply leave the buffer alone (already #2) depending on the value of RTE_BUFFER_INDEX.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
KILL OPEN

MODULE NAME: BYTE_IN
FILE NAME: CDUEXC.FOR
PROCESS: SLOW
CALLED BY: MESSAGE_MGR
CALLING SEQUENCE: CALL BYTE_IN

PURPOSE:
To handle CDU keyboard entries during data-link message composition mode.

DESCRIPTION:
When BYTE_IN is called one CDU key code resides in the CDU input buffer, ENTRY. The action taken depends on the type of key entered. If it was a page change or execute key it is simply passed on to the current page software called later by the executive. If a line select or delete key was pressed the entry buffer is cleared and the key is ignored. All other keys affect the CDU message being composed. A short clear removes the last character from the text while a long clear clears the entire message. Any other key received is an alphanumeric which is appended to the message buffer.

GLOBAL REFERENCES:

VARIABLES
FUNC MSG_CNT

ARRAYS
CDU_MSG* ENTRY*

MODULE NAME: DELIMIT
FILE NAME: LINK.FOR
PROCESS: SLOW
CALLED BY: LINK_EA, LINK_PD, LINK_RT
CALLING SEQUENCE: CALL DELIMIT(TEXT, CNT, DONE)

PURPOSE:
To parse the data-link clearance message.

DESCRIPTION:
This procedure parses the input string TEXT searching for either a "." or ":" character which are the only valid terminaters. The string length is returned in CNT and the boolean flag DONE is returned when at the end of the clearance message (":" encountered).

GLOBAL REFERENCES: none

MODULE NAME: EXPAND_RTE
FILE NAME: LINK.FOR
PROCESS: SLOW
CALLED BY: DEMODE, LINK_CMD, REJECT
CALLING SEQUENCE: CALL EXPAND_RTE

PURPOSE:
To create an expanded data-link text buffer for the data-link display.

DESCRIPTION:
When the flight crew desires to request a clearance, the current provisional flight plan is formatted into the data-link display buffer for transmission to the data-link computer. This is performed when the initial request is received and each time the provisional flight plan is changed during clearance request mode. This procedure steps through the provisional flight plan storing data into the display buffer with calls to TEXT_OUT. The destination airfield and cruise altitude are also formatted into the buffer.

GLOBAL REFERENCES:

VARIABLES
CRZALT MSG_CNT

ARRAYS
AIRPTS CDU_MSG* RTE_CNT

RECORD ARRAYS
RTE_MOD

FUNCTIONS AND SUBROUTINES
FILL_OUT FSTRNG GET_LONG TEXT_OUT TYPE_WPT

MODULE NAME: FILL_OUT
FILE NAME: LINK.FOR
PROCESS: SLOW
CALLED BY: MESSAGE_MGR, LINK_EA, LINK_RT, TEXT_OUT, EXPAND_RTE
CALLING SEQUENCE: CALL FILL_OUT(COUNT, BUFFER)

PURPOSE:
To fill data into the data-link message buffer.

DESCRIPTION:
The data specified by the input parameters is appended to the data-link display buffer that is built when flight plan clearance information is received. A display buffer pointer is maintained to account for the append position.

GLOBAL REFERENCES:

VARIABLES
MSG_CNT*

ARRAYS
CDU_MSG

FUNCTIONS AND SUBROUTINES
LIB$MOVC3

MODULE NAME: FILL_RTE
FILE NAME: LINK.FOR
PROCESS: SLOW
CALLED BY: LINK_PD, ADD_PLAN
CALLING SEQUENCE: CALL FILL_RTE(INDEX, ADDRESS)

PURPOSE:
To make a data-link waypoint entry in the route buffer.

DESCRIPTION:
FILL_RTE is called to fill in waypoint information in the provisional route buffer at the position indicated by the input parameter INDEX. If the address of the waypoint is supplied as a non-zero value, the waypoint is simply entered into the buffer position. Its type is determined by the function WPT_TYPE. When the address parameter is zero, FILL_RTE creates a "POS" pilot defined waypoint at the current aircraft position and inserts the created waypoint data into the route buffer. The function MAKE_WPT is used to create the waypoint.

GLOBAL REFERENCES:

VARIABLES
ALTCOR GS LAT LON

RECORD ARRAYS
RTE_MOD*

FUNCTIONS AND SUBROUTINES
MAKE_WPT TYPE_WPT

MODULE NAME: LINK_CMD
FILE NAME: LINK.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL LINK_CMD

PURPOSE:
To serve as the data-link software executive.

DESCRIPTION:
LINK_CMD is called by the CDU executive (CDUEXC) every pass through the background software. It monitors the data-link control word received from the data-link computer to initiate the appropriate actions for the specific data link commands.

The bits of the data-link control word (CDU_CMD) are assigned as follows.

0 CDU message composition mode
1 Clearance information received
2 Insert clearance as provisional flight plan
3 Erase previously received clearance
4 Clear current message composition buffer
5 Send current provisional flight plan to data-link

LINK_CMD looks for a change in state of the CDU_CMD bits, performing certain operations when a bit changes from off to on and others for changes from on to off.

The Insert command from the data-link computer requires special checking in LINK_CMD. If a provisional flight plan already exists when the Insert clearance is commanded the uplinked flight plan is not placed into the guidance buffer. Instead an error message is appended to to expanded flight plan text in CDU_MSG. The software then waits for another Insert command. When the second Insert is issued and the provisional guidance buffer is finished LINK_CMD restarts the parsing of the uplink clearance. This occurs since the changes to the flight plan which were being made may alter how the clearance affects the current flight plan. When the clearance processing is complete the insertion occurs immediately without response from the data-link computer.

Note that clearance commands may occur during CDU data link output sequences; data composition or clearance requests. LINK_CMD will save the current output data to make room for the overriding clearance data. When the new clearance sequence is finished the CDU will be restored to the previous state of data-link output.

GLOBAL REFERENCES:

VARIABLES
  ACTION CDU_CMD CDU_MODE* CRZALT* LNK_CNT LNK_CRZ MSG_CNT*
  NEW_CMD PMODE POSTIME* SQUAT TIME

ARRAYS
  AIRPTS CDU_MSG LNK_ARPT LNK_RTE MSG_BYT* RTE_CNT* WX_DEF

RECORD ARRAYS
  RTE_ACT RTE_MOD

FUNCTIONS AND SUBROUTINES
  DEMODE EXPAND_RTE LIB$MOVC3

MODULE NAME: LINK_EA
FILE NAME: LINK.FOR
PROCESS: SLOW
CALLED BY: MESSAGE_MGR
CALLING SEQUENCE: CALL LINK_EA(MESSAGE, INDEX, ERR_TEXT)

PURPOSE:
To handle expected altitude clearances from the data-link.

DESCRIPTION:
The input to LINK_EA is the parameter MESSAGE. It contains all the uplinked clearance following the "EA" field found by MESSAGE_MGR. The remaining parameters are outputs. INDEX is the pointer into the data-link input buffer LINK_IN. It is updated to point to any clearance following the EA data. ERR_TEXT is filled with text information when an error is detected while parsing the EA data.

The only data used in the EA clearance is an altitude assignment. The entry is decoded by the function ALTX. The message for the data-link display is created and stored in CDU_MSG and the altitude value is saved in the global variable LNK_CRZ.

GLOBAL REFERENCES:

VARIABLES
ACTION* ERCODE LNK_CRZ*

FUNCTIONS AND SUBROUTINES
ALTX DELIMIT FILL_OUT ISTRNG LIB$MOVC3

MODULE NAME: LINK_PD
FILE NAME: LINK.FOR
PROCESS: SLOW
CALLED BY: MESSAGE_MGR
CALLING SEQUENCE: CALL LINK_PD(MESSAGE, INDEX, ERR_TEXT)

PURPOSE:
To handle Proceed Direct clearances from the data-link.

DESCRIPTION:
The input to LINK_PD is the parameter MESSAGE. It contains all the uplinked clearance following the "PD" field found by MESSAGE_MGR. The remaining parameters are outputs. INDEX is the pointer into the data-link input buffer LINK_IN. It is updated to point to any clearance following the PD data. ERR_TEXT is filled with text information when an error is detected while parsing the PD data.

When LINK_PD is called it finds the waypoint name supplied in MESSAGE in the navigation data base, AADCOM. Once identified, a search of the provisional route buffer is made to determine if the waypoint exists on the provisional flight plan. If it does, all the waypoints preceding the selected waypoint are replaced by an auto-update "POS" waypoint, the remainder of the flight plan is not altered. When the selected waypoint is not part of the provisional flight plan a provisional flight plan consisting of an auto-update "POS" waypoint and the selected waypoint become the only two provisional flight plan waypoints.

Note that the actual route buffer manipulations are performed through calls to BEG_RTE and FILL_RTE.

GLOBAL REFERENCES:

VARIABLES
ACTION* ERCODE MSG_CNT

ARRAYS
CDU_MSG* ENTRY* RTE_CNT*

RECORD ARRAYS
RTE_MOD

FUNCTIONS AND SUBROUTINES
BEG_RTE BOUNDS DELIMIT FILL_RTE GET_LONG KILL LIB$MOVC3
TEXT_OUT WPT_ADDR

MODULE NAME: LINK_RT
FILE NAME: LINK.FOR
PROCESS: SLOW
CALLED BY: MESSAGE_MGR
CALLING SEQUENCE: CALL LINK_RT(MESSAGE, INDEX, ERR_TEXT)

PURPOSE:
To handle route clearance messages from the data-link.

DESCRIPTION:
The input to LINK_RT is the parameter MESSAGE. It contains all the uplinked clearance following the "RT" field found by MESSAGE_MGR. The remaining parameters are outputs. INDEX is the pointer into the data-link input buffer LINK_IN. It is updated to point to any clearance following the RT data. ERR_TEXT is filled with text information when an error is detected while parsing the RT data.

A RT clearance consists of one or more waypoints for the aircraft flight plan. Origin and destination airfields may be supplied also. The waypoint data can appear in several forms. Including individual waypoints, airway segments, standard instrumentation departures (SID), standard terminal arrivals (STAR), approaches, and implicit runway waypoints. The different types of multiple waypoint constructs are collectively referred to as route functions.

LINK_RT starts by using the procedure DELIMIT to parse the input message. Each item in the clearance is separated and saved for later processing in a waypoint/route function list. If the first entry in the list is an origin airfield a total reclearance is made. Note that a previously entered flight plan can only be erased when the aircraft is on the ground. When the last entry is an airfield it is used as the destination airport. When no destination has been entered, manually or by data-link, the destination is assumed to be the same as the origin.

There are three situations that are identified to prepare the provisional route for the new clearance. If the origin airfield was supplied, a completely new clearance is made. This means all previous waypoints are eliminated and the "new plan" flag is set which effects the processing in the module "RT_NEW". When the clearance is a SID, STAR or APPROACH no flight plan preparartion is needed since these always replace existing pieces or come at the very beginning or end of the flight plan. Other clearances are modifications to the existing flight plan which requires a call to ADD_PLAN to prepare the provisional guidance buffers. Once the preparation phase is complete LINK_RT steps through each item of the list with a call to RT_NEW to enter the flight plan.

GLOBAL REFERENCES:

VARIABLES
  ACTION* ERCODE MSG_CNT SQUAT

ARRAYS
  AIRPTS CDU_MSG* RTE_CNT*

FUNCTIONS AND SUBROUTINES
  ADD_PLAN DELIMIT FILL_OUT LIB$MOVC3 LUARP RT_NEW TEXT_OUT

MODULE NAME: MESSAGE_MGR
FILE NAME: CDUEXC.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL MESSAGE_MGR(ENTRY_RESTORE_BUFFER)

PURPOSE:
To manage the creation of data for the CDU_MSG data-link output buffer.

DESCRIPTION:
MESSAGE_MGR uses the global index CDU_MODE to determine the action required. It is not called when CDU_MODE is set to zero.

When CDU_MODE is set to -1 a new clearance has been received. The cryptic text uplinked from ground control must be expanded into more meaningful text for display to the flight crew on the data-link display. The expanded text is stored in the data-link output buffer CDU_MSG. A new provisional flight plan is also created from the up-linked clearance. The original clearance data is saved while the called modules create the new one. After processing is complete the original is restored and the new data is saved to be available when the flight crew chooses to "INSERT" the data-link clearance into the flight plan. There are four different types of clearance messages, and one or more will be found in a new data-link clearance. They are denoted by the following 2 letter code in the input text.

RT	Route clearance
PD	Proceed direct to a position
EX	Expected arrival clearance
EA	Expected altitude

The module LINK_RT is called for both the RT and EX types. The PD and EA types are processed by calls to LINK_PD and LINK_EA respectively.

When CDU_MODE is set to 1 a sequence of events is started for the data-link text message composition on the CDU scratch pad line. On each iteration of the CDU executive one of the follwing steps is taken.

CDU_MODE = 1: The CDU scratch pad sample request is sent to the CDU. CDU_MODE is set to 2.

CDU_MODE = 2: If the sample scratch pad has arrived the sampled data is saved, the CDU is commanded into no data buffering mode, and CDU_MODE is set to 3.

CDU_MODE = 3: The CDU remains in this mode until the composition text is complete. Each key entry on the CDU is appended to the current text buffer and the last 20 chracters of the text are output to the CDU scratch pad.

CDU_MODE = 4: This mode is set by the module LINK_CMD when message composition termination is detected. MESSAGE_MGR commands the CDU back into scratch pad buffering of text and reprograms the scratch pad with any data that existed there before composition mode was started.

GLOBAL REFERENCES:

VARIABLES
CDU_MODE* ERCODE LNK_CNT* LNK_CRZ* MSG_CNT* SAVE_CNT TIME

ARRAYS
AIRPTS* CDU_MSG ENTRY LINK_IN LNK_ARPT* LNK_RTE MESSAGE
MSG_BYT* RTE_CNT* SAVE_MOD

RECORD ARRAYS
RTE_MOD

FUNCTIONS AND SUBROUTINES
BYTE_IN FILL_OUT FMTOUT GET_WORD LIB$MOVC3 LINK_EA LINK_PD
LINK_RT RECALL SHOW_MESSAGE

MODULE NAME: RT_NEW
FILE NAME: LINK.FOR
PROCESS: SLOW
CALLED BY: LINK_RT
CALLING SEQUENCE: CALL RT_NEW(NAME, LENGTH, NEW_PLAN)

PURPOSE:
To enter clearance data into the flight plan.

DESCRIPTION:
RT_NEW is called with three input parameters from the procedure LINK_RT. The first is the name of a route item such as a waypoint or airway. The length of the name is the second parameter and the third is a flag indicating whether or not the current clearance was a new flight plan.

This module identifies the type of clearance entry and calls the appropriate subroutine to store the information in the flight plan being created for the received data-link clearance. The only clearance allowed when NEW_PLAN is false is a departure/arrival type entry. These are handled with a call to MODIFY. The NEW_PLAN type entries include departure/arrivals, waypoints, and airways. The waypoint entries are placed in the flight plan with a call to WAYPOINT while other types use a call to GROUP.

GLOBAL REFERENCES:

VARIABLES
ERCODE* LINE* MSG_CNT

ARRAYS
CDU_MSG* RTE_CNT

RECORD ARRAYS
RTE_MOD

FUNCTIONS AND SUBROUTINES
GET_LONG GROUP MODIFY RTE_ID TEXT_OUT TYPE_WPT WAYPOINT WPT_ID

MODULE NAME: SHOW_MESSAGE
FILE NAME: CDUEXC.FOR
PROCESS: SLOW
CALLED BY: MESSAGE_MGR
CALLING SEQUENCE: CALL SHOW_MESSAGE

PURPOSE:
To display the composed data-link message on the CDU.

DESCRIPTION:
SHOW_MESSAGE is called when the data-link computer has placed the CDU in message composition mode. In this mode the pilot creates a message intended for the ground controllers. This module writes the text "MSG>" to the CDU scratch pad and appends the last 20 characters of the message.

GLOBAL REFERENCES:

VARIABLES
MSG_CNT

ARRAYS
CDU_MSG

FUNCTIONS AND SUBROUTINES
FMTOUT

MODULE NAME: TEXT_OUT
FILE NAME: LINK.FOR
PROCESS: SLOW
CALLED BY: LINK_PD, LINK_RT, RT_NEW, EXPAND_RTE
CALLING SEQUENCE: CALL TEXT_OUT(ADDRESS, TYPE)

PURPOSE:
To store expanded message text for data-link display.

DESCRIPTION:
TEXT_OUT is called with an address of a route buffer item and its type. The item may be a waypoint or a route function. The following list describes the text stored in the data-link display buffer for the various types of route elements.

- GRP or PPT waypoint - the waypoint name
- other waypoints - AADCOM text associated with WPT
- Airways - the word VICTOR or JET followed by the airway number
- SID/STAR/APPROACH - AADCOM text associated with the route function followed by the text APPROACH, DEPARTURE, or ARRIVAL

GLOBAL REFERENCES:

VARIABLES
MSG_CNT

ARRAYS
CDU_MSG*

FUNCTIONS AND SUBROUTINES
FILL_OUT GET_BYTE GET_LONG

Section 6.0 CDU FLIGHT PLAN OPERATIONS

The most common use of the CDU is the creation and modification of the aircraft flight plan. Over 100 procedures are dedicated to transforming the pilots flight plan entries to a database used for aircraft guidance and cockpit displays. The flight crew may examine details of the flight plan on both the CDU and the navigation display. The flight plan database is used by automatic guidance to produce aircraft control signals and by the primary flight display to drive guidance cues used in manual flight operations.

The basic element of the flight plan is the waypoint. A waypoint is a global position defined by its latitude and longitude. Waypoint positions may be defined for some real geographic location such as a VOR transmitter or may be a convenient location such as the start of the final approach leg to a runway. The following are the different types of waypoints used in the ATOPS CDU system.

GRP - Ground reference point.
NAVAID - Navigational aid transmitter; VOR, DME, TACAN.
AIRFIELDS - Airfield tower position.
PILOT WAYPOINT - Dynamically generated waypoint. Can be created as a bearing and distance from a fixed reference (including the airplane) or an absolute latitude/longitude value.

Predefined groups of waypoints are referred to as route functions. The waypoints in a route function are defined in a sequence which is used to form a connected path segment. Not all waypoints defined for a route function must be included into the flight plan. Particular entry and exit waypoints may be chosen to bound the set of waypoints actually used in the plan. The different types of route functions used in the CDU are as follows.

SID - Standard Instrument Departure for airports.
AIRWAY - Both Victor and Jet airways which are bi-directional routes defined for major air traffic.
STAR - Standard Terminal Arrival Routes to airports.
APPROACH - Approach path to a particular airfield's runway.
HOLD - Holding pattern consisting of four pilot waypoints.

PAGE 53 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

The flight plan is made up of waypoints, route functions, and route dicontinuities which are collectively referred to as route elements. Route discontinuities are gaps in the flight plan which seperate path segments. They require a position in the route and waypoint buffers as do the previously mentioned elements, however the various data fields in the buffer are zeroed.

The desired route elements are manually entered into the flight plan by use of the various clearance pages of the CDU, shown below.

ROUTE - Enter origin/destination airfields and route elements into the flight plan.
LEGS - Enter individual waypoints and their constraints into the flight plan.
DIRECT/INTERCEPT - Designate a destination waypoint to be reached by a "Direct To" segment or a fixed bearing intercept.
LEG TIME - Specify an arrival time at a particular waypoint.
ROUTE INDEX - Request airway intercept.
HOLD - Define holding pattern.
DEPARTURE/ARRIVAL - select airfield departure and arrival routes.

Any particular waypoint on the path may have up to four constraints applied to it. These are altitude, speed, arrival time, and turn radius. The waypoint positions and their constraints are the parameters which are used in the creation of the waypoint guidance buffer used by flight management and display procedures.

Section 6.1 THE FLIGHT PLAN DATABASE

There are eight data buffers used to save flight plan information. The following sections describe the form and usage of the data stored. Each buffer is part of the set of commons defined as system global sections.

PAGE 56 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

Section 6.1.1 THE NAVIGATION DATABASE (AADCOM)

AADCOM is a read-only global common containing all pre-defined aircraft navigation data. A pointer block is placed at the begining of the common to direct search routines to the various data areas within the common. The format of the pointer block is as follows.

OFFSET	LABEL	POINTER TO
0	IBPTR	longitudinal strip data
4	BULK_ID	database ID text
20	JETPTR	jet airways
24	VICPTR	victor airways
28	RTEPTR	standard route airways
32	RESPTR	restricted areas
36	ADZPTR	air defense zones
40	CDZPTR	coastal defense zones

The longitudinal strip data consists of airfields, GRPs, NAVAIDs, and obstruction points existing within two degree increments of longitude. A strip is made up of a longitude boundary pair followed by four address pointers to the previously mentioned strip data items. The format of the various strip items is shown below. Note that a zero terminater word is used to denote the end of any block of navigation data.

AIRFIELDS:

OFFSET	TYPE	DATA
0	CHAR*4	airfield name
4	CHAR*1	" " (always blank)
5		not used
6	REAL*4	control tower latitude (deg)
10	REAL*4	control tower longitude (deg)
14	INT*4	pointer to next strip airfield
18	REAL*4	main runway length (ft)
22	REAL*4	main runway true heading (deg)
26	REAL*4	local magnetic variation (deg)
30	REAL*4	elevation
34	INT*4	terminal data block pointer (not used)
36	INT*2	tower frequency (2X5 code)
38	INT*2	clearance frequency (2X5 code)
40	INT*2	ground control frequency (2X5 code)
42	INT*2	ATIS frequency (2X5 code)
44	INT*4	pointer to list of SIDs
48	INT*4	pointer to list of STARs
52	INT*4	pointer to list of APPROACHs
56	INT*4	airfield ID pointer
60	n*48	runway data blocks

PRECEDING PAGE BLANK NOT FILMED

PAGE 58 INTENTIONALLY BLANK

AIRFIELDS - STANDARD INSTRUMENT DEPARTURE (SID) &
STANDARD TERMINAL ARRIVAL ROUTE (STAR):

OFFSET	TYPE	DATA
0	CHAR*6	SID name
6	INT*4	next item pointer (used for last wpt access)
10	INT*4	SID ID pointer
14	INT*4	first waypoint pointer
18	REAL*4	first waypoint assigned altitude (ft)
22	REAL*4	first waypoint assigned speed (kt)
26	REAL*4	first waypoint assigned radius (ft)
30	REAL*4	first waypoint DME arc bearing (deg)
		.
		.
		.
(N-1)*20+14		Nth waypoint pointer
		.
		.
		.
(N-1)*20+34		zero terminater

AIRFIELDS - APPROACHES:
These are identical to SID/STAR data format except for the insertion of a runway pointer at offset ten which increases all offsets above ten by four bytes.

AIRFIELDS - RUNWAYS:

OFFSET	TYPE	DATA
0	CHAR*3	runway name
3	BYTE	not used
4	REAL*4	threshold latitude (deg)
8	REAL*4	threshold longitude (deg)
12	INT*4	outter marker pointer
16	REAL*4	MLS/ILS latitude (deg)
20	REAL*4	MLS/ILS longitude deg)
24	REAL*4	runway length (ft)
28	REAL*4	runway true heading (deg)
32	REAL*4	runway elevation (ft)
36	REAL*4	glide slope angle (deg)
40	INT*4	ILS frequency (2X5 code)
44	INT*4	missed approach path pointer (not used)

GROUND REFERENCE POINTS (GRPs):

OFFSET	TYPE	DATA
0	CHAR*5	GRP name
5	BYTE	compulsory report flag
6	REAL*4	GRP latitude (deg)
10	REAL*4	GRP longitude (deg)
14	INT*4	navaid pointer

NAVIGATIONAL AIDS (NAVAID):

OFFSET	TYPE	DATA
0	CHAR*3	navaid name
3	BYTE	bit set 0:compulsory 1:vortac 2:non-directional 3:high alt 7:always
4	INT*4	frequency (2X5 code)
6	REAL*4	navaid latitude (deg)
10	REAL*4	navaid longitude (deg)
14	REAL*4	local magnetic variation (deg)
18	REAL*4	altitude (ft)
22	INT*4	navaid ID pointer

OBSTRUCTIONS:

OFFSET	TYPE	DATA
0	BYTE	bit 7 set: obstruction, else mountain
1	CHAR*5	obstruction altitude
6	REAL*4	obstruction latitude
10	REAL*4	obstruction longitude

AIRWAYS:

OFFSET	TYPE	DATA
0	CHAR*6	airway name
6	INT*4	pointer to next airway
10	INT*4	waypoint #1 pointer
		.
		.
		.
4*(N-1)+10		waypoint #N pointer
4*(N-1)+14		zero terminator

COMPANY ROUTES:

OFFSET	TYPE	DATA
0	CHAR*6	route name
6	INT*4	pointer to next route
10	INT*4	origin airfield pointer
14	INT*4	destination airfield pointer
18	INT*4	SID pointer
22	INT*4	STAR pointer
26	INT*4	waypoint #1 pointer
		.
		.
		.
4*(N-1)+26		waypoint #N pointer
4*(N-1)+30		zero terminator

BOUNDARIES:

OFFSET	TYPE	DATA
0	INT*2	not used
2	CHAR*6	boundary #1 name
8	REAL*4	bound #1, point #1 latitude
12	REAL*4	bound #1, point #1 longitude
		.
		.
		.
8*(N-1)+8		bound #1, point #N latitude
8*(N-1)+12		bound #1, point #N longitude
8*(N-1)+16		zero terminater

(boundary #2-N; terminated with zero word)

TEXT ID BLOCK:

OFFSET	TYPE	DATA
0	BYTE	ID character count
1	CHAR*N	ID text

Section 6.1.2 WAYPOINT CONSTRAINTS (CONBUF)

The constraint buffer holds altitude, speed, and turn radius values specified for flight plan waypoints. The connection between the route and the various constraint buffer packets is the ".CPTR" node of the route buffer structures (see Section 1.5.1.5). When the route buffer element is a route function ".CPTR" will be an index of a linked list of constraint packets in the buffer. Note that waypoint constraints from the system database and cruise altitude assignments do not appear in the constraint buffer. The structure of the constraint buffers is as follows.

```
INTEGER*4 CONBUF(4, 50)  ! 50 PACKETS OF 4 LONG WORDS EACH.

CONBUF(1,I) 0-15:  RTE OFFSET TO WAYPOINT (0 FOR NON-RTE WPT)
            16-18: ALT/SPD/RAD DEFINED FLAG
            19-22: UNUSED
            23:    ACTIVE CONSTRAINT FLAG
            24-31: INDEX TO NEXT RTE CONSTRAINT (0 - NO MORE)
CONBUF(2,I) 0-31:  ALTITUDE CONSTRAINT VALUE
CONBUF(3,I) 0-31:  SPEED CONSTRAINT VALUE
CONBUF(4,I) 0-31:  RADIUS CONSTRAINT VALUE
```

Section 6.1.3 HOLDING PATTERN DATA (HLDBUF)

The common block "HOLD" is a section of memory reserved for holding pattern data created by hold page modules. The memory allocation is defined in the file HLDBUF.MAR. This file contains definitions for a four waypoint airway and four GRPs used as hold waypoints. The format of these blocks is the same as those used in AADCOM (see section 1.5.1.1). The difference between the AADCOM and HLDBUF structures is that AADCOM is predefined read-only memory and HLDBUF is a template filled in holding pattern procedures.

PRECEDING PAGE BLANK NOT FILMED

PAGE 64 INTENTIONALLY BLANK

Section 6.1.4 PILOT DEFINED WAYPOINTS (PPT_WPT)

The pilot defined waypoint buffer is used to save information for waypoints created through calls to the function MAKE_WPT. Pilot waypoints are made for runway selection, aircraft position reference, bearing/ range from reference point, and absolute position selection. The Fortran allocation is shown below.

```
STRUCTURE /PPTS/
  CHARACTER*5 NAME
  BYTE BITS
  REAL LAT, LON, ALT, SPD
  CHARACTER*16 TEXT
END STRUCTURE
RECORD /PPTS/ PPT_WPT(20)
```

The ".BITS" node of the structure is set to indicate when altitude and speed have been supplied for the pilot waypoint. Bit #0 is set for altitude definition and bit #1 is set for speed definition. The ".TEXT" node is set to the CDU command string entered by the pilot which caused the pilot waypoint creation. This text may be viewed on the NAV DATA page of the CDU.

PAGE 66 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

Section 6.1.5 THE ROUTE BUFFERS (RTE_MOD/RTE_ACT)

The route buffers are in the global common area CDUCOM. There is one for the active flight plan and one for the modified plan. Each has room for 30 route elements. The structure definition is shown below.

```
STRUCTURE /RTE/
  INTEGER*4 ADDR
  BYTE TYPE, CPTR
  UNION
    MAP
      INTEGER*2 RWY
    END MAP
    MAP
      INTEGER*2 EXIT
    END MAP
  END UNION
END STRUCTURE
RECORD /RTE/ RTE_MOD(30), RTE_ACT(30)
```

The nodes of the structure are described in the following list. Note that ".RWY" and ".EXIT" are duplicate references to the same memory location. This is because both nodes are not used for the same route element.

.ADDR	Memory address of the route element. May point to a location in AADCOM, HLDBUF, or PPT_WPT.
.TYPE	Route element type as follows. DISCONTINUITY = 0, AIRFIELD = 1, GRP = 2, NAVAID = 3, PILOT WPT = 4, HOLD PATTERN = 5, APPROACH = 6, SID = 7, STAR = 8, AIRWAY = 9
.CPTR	Constraint buffer index.
.RWY	Runway waypoint. 1:origin 2:destination
.EXIT	Route function offset to exit waypoint address.

PRECEDING PAGE BLANK NOT FILMED

PAGE 68 INTENTIONALLY BLANK

Section 6.1.6 THE WAYPOINT BUFFERS (WPT_MOD/WPT_ACT)

The waypoint buffers contain the actual waypoint data which defines the entire flight plan. WPT_ACT is used for the active flight plan while WPT_MOD has the path which is under modification. WPT_MOD is re-created each time a flight plan change is entered on the CDU. The waypoint buffer is actually an expansion of the data already existing in the route buffer. Each route element is replaced by one or more waypoints having any constraints defined by CONBUF (see section 1.5.1.2), the cruise altitude, or AADCOM predefinition. A number of waypoint buffer parameters are computed from the geometry of the waypoints taken from the expansion process. This "Path Definition" phase, performed by the procedure PATHDF, starts when the expansion process is complete. The structure template of the waypoint buffers is shown below followed by a description of each of the parameters.

```
STRUCTURE /WPTS/
  CHARACTER*5 NAME
  BYTE DMA, SOURCE, PHASE, ALTF, SPDF, RADF, FILL
  INTEGER*4 RNAV, ETA
  REAL LAT, LON, ALT, GS, TIME, CCD
  REAL ARC2, DTT, RAD, BRNG, ANGLE, ERAD, PPD
  REAL WPV(3), TCV(3), NMV(3)
  REAL IAS, TCLAT, TCLON, WSPD, WDIR, MGVR, TDEV, FPA
END STRUCTURE
RECORD /WPTS/ WPT_ACT(30), WPT_MOD(30)
```

.NAME	Waypoint name.
.DMA	1:DMA turn start, 2:DMA turn stop, else 0.
.SOURCE	Index into route buffer indicating the element the waypoint was expanded from.
.PHASE	Phase of flight; 1:climb 2:cruise 3:descent 0:undefined
.ALTF	Altitude defined flag; 0:undefined 1:explicit definition (AADCOM,PPT_WPT,CONBUF), 2:implicit defintion (cruise alt,'POS' updatable wpt). If equal 2, shown in small font on LEGS page.
.SPDF	Speed definition flag; see .ALTF
.RADF	Radius definition flag; 0:computed 1:assigned
.FILL	Keeps remaining nodes on long word boundary.
.RNAV	Radio navigation aid address pointer.
.ETA	Estimated time of arrival (seconds past midnight).
.LAT	Waypoint latitude (deg).
.LON	Waypoint longitude (deg).
.ALT	Waypoint altitude (ft).
.GS	Waypoint ground speed (kt).
.TIME	Leg time from last waypoint (seconds).
.CCD	Turn center to turn center distance (ft).

PAGE 70 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

.ARC2	One half turn arc length (ft).
.DTT	Distance from waypoint to turn tangent point (ft).
.RAD	Turn radius (ft).
.BRNG	Inbound leg bearing or DME waypoint bearing (deg).
.ANGLE	Turn angle (deg; -:left +:right)
.ERAD	Local earth radius value (ft).
.PPD	Point to point distance from last waypoint (ft).
.WPV	Earth center to waypoint unit vector.
.TCV	Earth center to turn center unit vector.
.NMV	Normal unit vector. Perpendicular to plane formed by earth center, previous, and current waypoints.
.IAS	Waypoint airspeed (not used).
.TCLAT	Turn center latitude (deg).
.TCLON	Turn center longitude (deg).
.WSPD	Local wind speed (not used).
.WDIR	Local wind direction (not used).
.MGVR	Local magnetic variation (deg).
.TDEV	Local temperature deviation (not used).
.FPA	Leg flight path angle from last waypoint.

A subset of the waypoint buffers is copied into other waypoint buffers for transmission to the Display VAX. This is done to save time since I/O time for 30 copies of waypoint data is significant. The copying of the data is performed in the background also to utilize available "fast loop" processing time. The structure of the display waypoint buffer is shown below. All the nodes match the structure described above except for ".CODES". The .DMA, .ALTF, and .SPDF data mentioned above are packed into one byte in the display waypoint buffers. Bits 0 & 1 are used for the DMA index and bits 2 & 3 are used for the altitude and speed booleans respectively.

```
STRUCTURE /DWPTS/
  INTEGER*4 ETA
  REAL LAT, LON, ALT, GS, TIME, CCD, ARC2, DTT, RAD, BRNG
  REAL ANGLE, ERAD, PPD, WPV(3)
  CHARACTER*5 NAME
  BYTE CODES
END STRUCTURE
RECORD /DWPTS/ MOD_WPTS(30), ACT_WPTS(30)
```

Section 6.2 FLIGHT PLAN DATA PROCESSING

This section covers the internal operations performed on the flight plan data buffers. The 23 modules described here are contained in four files named EXECUTE.FOR, XLAT_RTE.FOR, PATHDF.FOR, CONST.FOR. These modules use the constraint buffer, navigation database, holding pattern data, pilot waypoint buffer, and the route buffer to create a provisional waypoint buffer which, upon pilot acceptance, becomes the active waypoint buffer.

The various clearance pages of the CDU have three modes of operation; original clearance, modified clearance, and active clearance. The current mode is shown with the first three characters of the title line of clearance pages as follows.

" " - orignal clearance
"MOD" - modified clearance
"ACT" - active clearance

The original clearance mode is active by default upon starting the system or when the origin airfield is entered on the ROUTE page of the CDU. At this time RTE_MOD has the provisional flight plan and RTE_ACT is undefined. Once the original clearance is EXECuted by the pilot the mode becomes ACT. At this time RTE_MOD and RTE_ACT both contain the active flight plan. When changes are made to the active plan the CDU mode becomes MOD. RTE_ACT will contain the active flight plan being used by guidance, however the CDU shows the modified flight plan stored in RTE_MOD.

Creation of a new "MOD" waypoint buffer is started when a CDU page handler receives a flight plan input and calls the procedure DEMODE. Acceptance of the new flight plan can be automatic, depending on DEMODE parameters, or may require pilot interaction. The pilot may however choose to reject the modified route and return to the last active plan. These operations are performed by the modules EXECUTE and REJECT.

Section 6.2.1 CONSTRAINT BUFFER USAGE

The constraint buffer is used to store altitude, speed, and turn radius constraints for route waypoints. The constraints are entered manually on the CDU through the LEGS page. The format of this buffer is described in section 1.5.1.2. Seven procedures perform various operations on the constraint buffer. Module descriptions of each are provided on the following pages.

PAGE 74 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: CLEAN_CON
FILE NAME: CONST.FOR
PROCESS: SLOW
CALLED BY: EXECUTE, REJECT
CALLING SEQUENCE: CALL CLEAN_CON

PURPOSE:
To identify unused packets of the constraint buffer.

DESCRIPTION:
This module performs clean-up on the CDU constraint buffer. When a flight plan is executed or changes to the active flight plan have been rejected, both the "ACT" and "MOD" route buffers are identical. At this time "CLEAN_CON" is called to identify which of the 50 constraint buffer packets of data are actually used. All others are marked as free for future use while the used packets are flagged as active constraints.

A 50 byte array of booleans (USED) is initialized as false (not used). Each ".CPTR" pointer is followed into the linked list of constraints. As each constraint is found the "active constraint" bit is set and the corresponding USED byte is set true. When finished, the first long word of each constraint packet not denoted as used is cleared to designate it as available.

GLOBAL REFERENCES:

ARRAYS
CONBUF* CONBYT RTE_CNT

RECORD ARRAYS
RTE_MOD

FUNCTIONS AND SUBROUTINES
CLRBUF

MODULE NAME: COPY_CON
FILE NAME: CONST.FOR
PROCESS: SLOW
CALLED BY: KILL_CON, NEW_CON
CALLING SEQUENCE: NEW_INDEX = COPY_CON(OLD_INDEX)

PURPOSE:
To copy a constraint list to other free constraint buffer locations.

DESCRIPTION:
This function is called to copy a constraint list starting at the packet indicated by the OLD_INDEX input parameter. The data is copied to free packets in the constraint buffer and the index of the new list is returned as the function value.

The linked list pointers are followed and for each old constraint packet a call to FIND_EMPTY is made to get an available block. The old data is copied to the new locations and the "active constraint" bit of the new data is cleared.

GLOBAL REFERENCES:

ARRAYS
CONBUF CONBYT

FUNCTIONS AND SUBROUTINES
FIND_EMPTY LIB$MOVC3

MODULE NAME: FIND_EMPTY
FILE NAME: CONST.FOR
PROCESS: SLOW
CALLED BY: COPY_CON, HLD_POS, NEW_CON
CALLING SEQUENCE: INDEX = FIND_EMPTY()

PURPOSE:
To locate an available constraint packet.

DESCRIPTION:
This function returns the index of the first free packet in the constraint buffer. The first long word of each packet is examined until an available set is found (equal 0). If all 50 constraints are used the CDU error code is set to #4 and a return to the caller's caller is performed.

GLOBAL REFERENCES:

VARIABLES
ERCODE*

ARRAYS
CONBUF

FUNCTIONS AND SUBROUTINES
RET

MODULE NAME: GET_CON
FILE NAME: CONST.FOR
PROCESS: SLOW
CALLED BY: RTE, XLAT_RTE
CALLING SEQUENCE: GET_CON(RTE_PTR, OFFSET, WPT_PTR)

PURPOSE:
To store constraints into waypoint buffer locations.

DESCRIPTION:
This procedure is called with an index into the route buffer (RTE_PTR) of the route element containing a particular waypoint. If the route element is a route function then the database offset is also supplied. Any constraint data existing for the waypoint is fetched and copied to the waypoint buffer for the waypoint designated by the parameter list index WPT_PTR. The waypoint buffer flags .ALTF, .SPDF, and .RADF are set appropriately.

Note that when the route element is a route function, the constraint buffer contains a linked list which must be followed until a matching offset value is found or the end of the list is encountered.

GLOBAL REFERENCES:

ARRAYS
CONBUF CONBYT CONWRD

RECORD ARRAYS
RTE_MOD WPT_MOD*

MODULE NAME: KILL_CON
FILE NAME: CONST.FOR
PROCESS: SLOW
CALLED BY: NMBRS
CALLING SEQUENCE: CALL KILL_CON(WPT_PTR, CODE)

PURPOSE:
To remove one or more constraints from a waypoint.

DESCRIPTION:
This procedure is called to remove one or more constraints associated with a particular waypoint buffer waypoint. If the constraint packet is designated as an active set, a copy of the packet is made and the route buffer element is redirected to the new copy. The assignment bits (16-18 of first long word) of the constaint packet are cleared as indicated by the CODE input parameter. If the constraint packet becomes null it is removed from the linked list or the route buffer pointer (CPTR) is cleared if it were the only constraint packet.

GLOBAL REFERENCES:

ARRAYS
CONBUF CONBYT* CONWRD

RECORD ARRAYS
RTE_MOD* WPT_MOD

FUNCTIONS AND SUBROUTINES
COPY_CON RTE_WPT WPT_ADDR

MODULE NAME: NEW_CON
FILE NAME: CONST.FOR
PROCESS: SLOW
CALLED BY: NMBRS, XFER_CON
CALLING SEQUENCE: variable (see below)

PURPOSE:
To insert waypoint constraints into the constraint buffer.

DESCRIPTION:
This module is called to add a constraint to a waypoint on the flight plan. The waypoint may have other constraints already defined. Two calling sequences exist for this module, as shown below. The module P_LIST is used to determine which calling sequence was used.

```
CALL NEW_CON(WPT_PTR, V_TYPE, VALUE)
CALL NEW_CON(,V_TYPE, VALUE, RTE_PTR, RTE_OFF)
```

WPT_PTR	Index into waypoint buffer of selected waypoint.
V_TYPE	Type of constraint: 1=altitude 2=speed 3=radius
VALUE	Constraint value.
RTE_PTR	Rte buffer index of rte element owning selected wpt.
RTE_OFF	Offset into rte function for rte type waypoint.

The constraint pointer of the route element corresponding to the selected waypoint is fetched. It is used to determine if a constraint packet already exists for the waypoint. If saved constraints for the route element have the "active" bit set, a duplicate copy is made so modifications may be made. When the pointer points to a linked list, the links must be followed to determine the existence of data for a particular waypoint. When constraint data already exists, the fields of the packet are simply updated. Otherwise a new packet is created. When the waypoint is part of a route function which has other waypoint constraints, the new packet is inserted into the linked list chain.

GLOBAL REFERENCES:

ARRAYS
CONBUF* CONBYT* CONWRD*

RECORD ARRAYS
RTE_MOD* WPT_MOD

FUNCTIONS AND SUBROUTINES
COPY_CON FIND_EMPTY P_LIST RTE_WPT WPT_ADDR

MODULE NAME: XFER_CON
FILE NAME: CONST.FOR
PROCESS: SLOW
CALLED BY: MERGE, NEW_ENTRY
CALLING SEQUENCE: CALL XFER_CON(FROM, OFFSET, TO)

PURPOSE:
To transfer constraint data.

DESCRIPTION:
This procedure is called to transfer existing waypoint constraints to another waypoint. Both waypoints must be part of a route function which is indicated by the route buffer pointer FROM/TO and the route function offset, OFFSET. This procedure is used by routines which split a single route function into a repeated pair of route functions with different entry/exit points.

GLOBAL REFERENCES:

ARRAYS
CONBUF CONBYT CONWRD

RECORD ARRAYS
RTE_MOD

FUNCTIONS AND SUBROUTINES
NEW_CON

Section 6.2.2 ROUTE TRANSLATION AND PATH DEFINITION

The modified route buffer (RTE_MOD) and other basic flight plan database elements are combined to form the provisional waypoint buffer (WPT_MOD). The procedures on the file XLAT_RTE.FOR perform this task. Once the route buffer has been translated into a basic waypoint buffer, the path definition procedures contained in the file PATHDF.FOR are called to create the mathematical constructs associated with using the waypoint buffer as a guidance buffer. The translation and definition process is started by a call to the procedure DEMODE any time a flight plan modification is made on the CDU. The following pages contain the module descriptions for these routines.

MODULE NAME: CREATE_BUF
FILE NAME: EXECUTE.FOR
PROCESS: SLOW
CALLED BY: DEMODE, EXECUTE
CALLING SEQUENCE: CALL CREATE_BUF(COUNT, WPT_BUF, DSP_BUF)

PURPOSE:
To move selected portions of the waypoint buffer to the display waypoint buffer.

DESCRIPTION:
This procedure is called with a waypoint count and one of the waypoint buffers as inputs. It stores portions of the waypoint buffer into one of the display waypoint buffers as the sole output parameter.

For each waypoint sixty-eight consecutive bytes of data starting at the .ETA parameter are moved to the display buffer. Then the .DMA, .ALTF, and .SPDF data is packed into the .CODES bytes of the display buffer.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
LIB$MOVC3

MODULE NAME: DEMODE
FILE NAME: EXECUTE.FOR
PROCESS: SLOW
CALLED BY: FLT_TYPE_INP, HLD_POS, HOLD_INPUT, INTC_WPTS, LINK_CMD, MOD_ROUTE, NEWCRZ, ROUTE, TIME_IN, UPDATE_POS, WPT_DATA
CALLING SEQUENCE: CALL DEMODE(MODE_FLAG)

PURPOSE:
To initiate the creation of a new waypoint buffer.

DESCRIPTION:
This procedure is called when a change is made to the flight plan. If the change is made to the active plan the CDU demodes to the provisional plan status indicated by the text "MOD" on the header of clearance pages. DEMODE initiates the translation of the route buffer to a complete waypoint buffer. There are three modes of operation for this module; Auto execute, No execute, and No trim. The normal sequence occurs for the No execute mode. The active plan waypoints already over-flown are trimmed away and the route translation occurs always leaving the CDU in the MOD plan mode. The No Trim mode is the same as No execute but the removal of passed waypoints is not performed. If the changes made to the flight plan do not require final approval from the pilot through EXEC selection, the Auto execute mode is used.

The first test in DEMODE determines if the current CDU mode is "Active". If it is, DEMODE will enable auto execute if requested, and perform the waypoint trimming by calling TRIM_WPTS. The current destination waypoint pointer PTR2D is saved to be used later when the pilot executes the modified plan. Next consecutive route discontinuities are removed from the route buffer and the route buffer to waypoint buffer translation is performed through calls to DSC_CHECK and XLAT_RTE respectively. Once the waypoint buffer is created it is either made active, if Auto execute is enabled, or several tests are made on the new buffer. If the modified waypoint buffer WPT_MOD starts with at least two waypoints before any route discontinuity markers the EXEC light flag is set. Also if the buffer contains any route discontinuity markers the DSPLY light flag is set.

The final steps of the waypoint buffer creation process are to create the display waypoint buffer subset and expand the provisional route to data-link text description if expansion is enabled. The display buffer is created by calling CREATE_BUF and the transmission to displays is activated by clearing GDTIME.

GLOBAL REFERENCES:

VARIABLES
CDU_CMD GDTIME* LT_DSPY* LT_EXEC* MODCNT PMODE* PTR2D
SAVPTR*

ARRAYS
RTE_CNT

RECORD ARRAYS
MOD_WPTS RTE_MOD WPT_MOD

FUNCTIONS AND SUBROUTINES
CREATE_BUF DSC_CHECK EXECUTE EXPAND_RTE TRIM_WPTS XLAT_RTE

MODULE NAME: DSC_WPT
FILE NAME: XLAT_RTE.FOR
PROCESS: SLOW
CALLED BY: XLAT_RTE, RTE
CALLING SEQUENCE: CALL DSC_WPT(WPT_INDEX, RTE_INDEX)

PURPOSE:
To insert a route discontinuity into the waypoint buffer.

DESCRIPTION:
This subroutine is called to insert a discontinuity marker into the flight plan. One is inserted only if the previous waypoint entry was not also a discontinuity marker and there is room in the waypoint buffer. The discontinuity is associted with the route buffer via the source index "I".

GLOBAL REFERENCES:

RECORD ARRAYS
WPT_MOD*

MODULE NAME: FIND_CCD
FILE NAME: PATHDF.FOR
PROCESS: SLOW
CALLED BY: PATH
CALLING SEQUENCE: CCD = FIND_CCD(TO_ADJUST)

PURPOSE:
To compute the CCD parameter in the waypoint buffer.

DESCRIPTION:
The turn center to turn center distance (CCD) differs from the waypoint to waypoint distance for "Pass By" waypoints. The adjusment at each end of the leg is the tangent distance minus half the turn arc length (DTT-ARC2). The "From Waypoint" adjustment is computed in this procedure and the "To Waypoint" adjustment is passed as a parameter to avoid DMA arc entry waypoint testing.

Checks are made to assure that the leg being processed has proper geometry at both ends. When the sum of the tangent distance becomes larger than the point to point distance a "Bad Radius" situation has occured. This means a "Pass By" turn is to large for the given leg length. The offending turn radius is set to zero and the path definition waypoint index is reset to force the calling module to recompute parameters using the new turn radius.

GLOBAL REFERENCES:

VARIABLES
ERCODE*

RECORD ARRAYS
WPT_MOD

MODULE NAME: LOCAL_ERAD
FILE NAME: PATHDF.FOR
PROCESS: SLOW
CALLED BY: PATH, NEW_POS, POINTS
CALLING SEQUENCE: COMMON /PTHCOM/ LAT_FEET, LON_FEET, RAD
CALL LOCAL_ERAD(ALT, SIN_LAT, COS_LAT)

PURPOSE:
To compute local earth radius values.

DESCRIPTION:
This procedure uses the input parameter for waypoint altitude, and sine/cosine of latitude to compute local earth radius values. The computed values, returned through common PTHCOM, are the earth radius to waypoint and the number of feet per degree of both latitude and longitude.

GLOBAL REFERENCES:

VARIABLES
LAT_FEET* LON_FEET* RAD*

MODULE NAME: PATH
FILE NAME: PATHDF.FOR
PROCESS: SLOW
CALLED BY: PATHDF
CALLING SEQUENCE: CALL PATH(START_INDEX, END_INDEX)

PURPOSE:
To compute flight plan parameters required by guidance and display software.

DESCRIPTION:
This procedure computes many of the guidance buffer parameters associated with the aircraft flight plan, which are contained in the structure "WPT_MOD". Several of the structure nodes are filled by "XLAT_RTE" before calling this subroutine. The following list shows which parameters of "WPT_MOD" are computed. Some values are not computed for all waypoints, therefore these exceptions are noted.

WPT_MOD(I)".XXX"		
LAT	waypoint latitude	INBOUND DMA WPTS ONLY
LON	waypoint longitude	INBOUND DMA WPTS ONLY
PPD	point to point distance	ALL WPTS
DTT	distance to tangent	ALL WPTS
ARC2	one half arc length	ALL WPTS
ANGLE	turn angle	ALL WPTS
CCD	center to center distance	ALL WPTS
BRG	bearing	ALL EXCEPT OUTBOUND DMA
TCLAT	turn center latitude	ALL EXCEPT INBOUND DMA
TCLON	turn center longitude	ALL EXCEPT INBOUND DMA
FPA	flight path angle	ALL WPTS
TIME	delta time	ALL WPTS
ERAD	local earth radius	ALL WPTS
WPV	waypoint vector	ALL WPTS
NMV	unit normal vector	ALL WPTS
TCV	turn center vector	ALL WPTS

PATHDF computes the guidance parameters in two passes through the waypoint buffer. The exact mathematical concepts invloved with the various parameters are discussed in Appendix A.

During the first pass the waypoint vectors (WPV), normal vectors (NMV), point to point distances (PPD), and local earth radius (ERAD) values are computed for each waypoint. WPV and ERAD are computed by calls to XYZ and LOCAL_ERAD respectively. NMV is the cross product of the current and previous waypoints WPV vectors. These WPV values are also used in combination with the ERAD value to compute PPD using the arc length formula Arc_length = Radius * Angle_radians. DMA waypoint repositioning occurs also during the first pass.

When PATHDF encounters a DMA entry waypoint, new latitude and longitude values are computed from the turn center previously stored as LAT/LON. The old LAT/LON values are moved to the turn center locations (TCLAT/TCLON). The new LAT/LON values are found from the turn center position, bearing from turn center to new position, and the turn radius which are all fetched from AADCOM and stored by XLAT_RTE prior to calling PATHDF.

The remaining guidance parameters are computed during the second pass through the waypoint buffer. None of the parameters for the second pass are computed for the first waypoint. Several are not assigned for the last waypoint (ANGLE, ARC2, DTT, TCV, TCLAT, TCLON). There are three guidance buffer parameters that are handled identically for all types of waypoints. These are RAD, FPA, and TIME. For a DMA entry waypoint only the CCD value is computed. The TCV, CCD, and previous waypoints ARC2 parameters are computed when the waypoint is a DMA arc exit waypoint. For standard waypoints the BRNG, CCD, ANGLE, ARC2, DTT, TCV, TCLAT, and TCLON values are set.

When starting and ending tangent distances (DTT) for a leg are large enough to overlap, a "Bad Radius" turn exists. The module FIND_CCD makes a zero radius turn at the offending waypoint and signals PATHDF to recompute guidance parameters for the new turn radius. The waypoint with the redefined turn radius appears on the LEGS page of the CDU with an asterisk.

GLOBAL REFERENCES:

VARIABLES
  COS_LAT LAT_FEET LON_FEET RAD SIN_LAT

RECORD ARRAYS
  WPT_MOD*

FUNCTIONS AND SUBROUTINES
  FIND_CCD GRID LOCAL_ERAD MTH$ASIN MTH$ASIND MTH$ATAN2
  MTH$ATAND2 SCOSD UVC VCP VDP VMG XYZ

MODULE NAME: PATHDF
FILE NAME: PATHDF.FOR
PROCESS: SLOW
CALLED BY: XLAT_RTE
CALLING SEQUENCE: CALL PATHDF

PURPOSE:
To initiate path definition computations.

DESCRIPTION:
This subroutine is the main driver of the path definition process. Calls to "PATH" are made after delimiting groups of consecutive waypoints in the provisional waypoint structure "WPT_MOD". Usually one call to "PATH" is made for the entire set of waypoints, however multiple calls are made when route discontinuities exist in the flight plan. The effect of this is to make several disjoint path segments in one waypoint buffer.

GLOBAL REFERENCES:

VARIABLES
MODCNT

RECORD ARRAYS
WPT_MOD

FUNCTIONS AND SUBROUTINES
PATH

MODULE NAME: RTA_TIMES
FILE NAME: XLAT_RTE.FOR
PROCESS: SLOW
CALLED BY: XLAT_RTE
CALLING SEQUENCE: CALL RTA_TIMES

PURPOSE:
To set arrival times in the waypoint buffer.

DESCRIPTION:
The Requested Time of Arrival (RTA) waypoint is located in the waypoint buffer. When the waypoint no longer exists the RTA parameters are reset. Otherwise the selected RTA time is assigned to the RTA waypoint and the remaining waypoint times are set according to the stored leg time values.

GLOBAL REFERENCES:

VARIABLES
MODCNT RTA_INDX* RTA_TM RTA_WPT*

RECORD ARRAYS
WPT_MOD*

MODULE NAME: RTE
FILE NAME: XLAT_RTE.FOR
PROCESS: SLOW
CALLED BY: XLAT_RTE
CALLING SEQUENCE: CALL RTE(RTE_INDEX, WPT_INDEX)

PURPOSE:
To store route waypoints in the waypoint buffer.

DESCRIPTION:
This subroutine is called by "XLAT_RTE" when a route function is encountered while translating the route buffer. All the waypoints for a complete route function (both entry and exit waypoints defined) are placed in the WPT buffer through calls to "WPT", except for the entry waypoint. Since the route buffer defines the entry WPT as a separate route buffer element the entry waypoint will already be placed in the waypoint buffer when the route function is being processed. The only thing "RTE" does for the entry waypoint is the search of the constraint buffer mentioned below. When the route function is not an airway, the speed, altitude, and DMA turn information is fetched from the navigation database (AADCOM). The last processing of each waypoint consists of calling GET_CON to extract constraint data. Note that previously stored AADCOM values will be overwritten if higher priority constraint buffer values exist.

When a route function's entry waypoint has not been defined a discontinuity will already exist in the waypoint buffer. In this case the exit waypoint only is saved following the discontinuity. For the reverse situation a discontinuity is stored after the existing entry waypoint for the missing exit waypoint. When niether is defined one discontinuity is placed in the waypoint buffer for the entire route function.

A distinction is made between airways and other route functions (SID STAR APPROACH HOLD). Each airway waypoint is assigned a "cruise" phase of flight and the current cruise altitude, which may be overridden later during the constraint buffer fetches. For non-airway route functions, SIDs are assigned the "climb" phase of flight while others are set to "descent". The altitude, speed, and turn radius is fetched from the navigation database (AADCOM) or the hold buffer (HLDBUF). The turn radius may contain a zero which is a cue to the path definition routine to compute the value. The fetched altitude is also used as a flag. When negative, the current waypoint is a DMA turn entry waypoint and the following waypoint is a DMA turn exit waypoint. The DMA bearing and turn angle are also fetched from the database in these cases.

GLOBAL REFERENCES:

VARIABLES
  CRZALT

RECORD ARRAYS
  RTE_MOD WPT_MOD*

FUNCTIONS AND SUBROUTINES
  BOUNDS DSC_WPT GET_CON GET_LONG GET_REAL TYPE_WPT WPT

MODULE NAME: TRIM_WPTS
FILE NAME: EXECUTE.FOR
PROCESS: SLOW
CALLED BY: DEMODE, ADD_WPT, DIRECT
CALLING SEQUENCE: CALL TRIM_WPTS(NEXT_WPT_INDEX)

PURPOSE:
To remove passed waypoints from the active flight plan.

DESCRIPTION:
This procedure is called to eliminate the beginning waypoints from the route buffer which have already been passed on the active flight plan. When the deletion splits a route function a new entry waypoint is created.

The input parameter NEXT_WPT_INDEX points to the end waypoint in the waypoint buffer of the leg which is to become the first leg of the route. The route buffer element corresponding to the waypoint before the chosen waypoint is examined to determine its type. If the route element is a single waypoint, all the route elements up to and including the tested one are removed from the route buffer (RTE_MOD) by calling KILL. When a route function is encountered the procedure NEW_ENTRY is used to split the route function into the portion that is to be saved. Then the prior elements are removed by calling KILL.

GLOBAL REFERENCES:

RECORD ARRAYS
RTE_MOD WPT_MOD

FUNCTIONS AND SUBROUTINES
BOUNDS KILL NEW_ENTRY RTE_WPT WPT_ADDR

MODULE NAME: WPT
FILE NAME: XLAT_RTE.FOR
PROCESS: SLOW
CALLED BY: XLAT_RTE, RTE
CALLING SEQUENCE: CALL WPT(RTE_INDEX, WPT_INDEX, ADDR, TYPE)

PURPOSE:
To store waypoint data into the waypoint buffer.

DESCRIPTION:
This subroutine is called to place a waypoint in the next available position of the modified waypoint buffer (WPT_MOD). The following items are stored for each waypoint.

.NAME	5 character WPT name padded with blanks on the right.
.LAT	Waypoint latitude position.
.LON	Waypoint longitude position.
.RNAV	database address pointer to local navaid.
.MGVR	Local magnetic variation value.
.SOURCE	Index into rte buffer which associates a waypoint with the rte buffer data that caused its creation.

If the waypoint is a pilot defined waypoint four other items may be set.

.ALT	Altitude constraint.
.ALTF	Altitude definition flag.
.GS	Ground speed constraint.
.SPDF	Speed definition flag.

The data is fetched using the input address parameter which is a pointer into AADCOM, HLDBUF, or PPT_WPT. The TYPE parameter determines how the data is fetched. The .LAT, .LON, .MGVR, and .SOURCE paramters are set identically for all types. Other parameters are set as follows.

AIRFIELD:	4 character name / no navaid pointer (0).
GRP:	5 character name / navaid fetched from GRP block in database (ADDR+14).
NAVAID:	3 character name / navaid reference is self.
PILOT PT:	5 character name / no navaid / alt & spd data fetched from pilot buffer (PPT_WPT). flags set according to pilot waypoint type, POS, PPT, or RWY.

GLOBAL REFERENCES:

VARIABLES
  ERCODE*

RECORD ARRAYS
  WPT_MOD*

FUNCTIONS AND SUBROUTINES
  GET_BYTE GET_CHAR GET_LONG GET_REAL GET_WORD MAG_VAR

MODULE NAME: XLAT_RTE
FILE NAME: XLAT_RTE.FOR
PROCESS: SLOW
CALLED BY: DEMODE, HLD_POS
CALLING SEQUENCE: CALL XLAT_RTE

PURPOSE:
To translate the route buffer into a waypoint buffer.

DESCRIPTION:
This subroutine translates the route buffer "RTE_MOD" into the equivalent waypoint buffer "WPT_MOD". The route buffer consists of waypoints, route functions, and route discontinuity markers. The waypoint buffer contains just waypoints and discontinuity markers. First, the entire memory area reserved for WPT_MOD is cleared to initialize all parameters to zero. Once this is finished each item in RTE_MOD is examined to determine the appropriate action to perform corresponding to its type. The last step is to call PATHDF to compute flight plan parameters for each waypoint.

The conversion is done by indexing through each element of the modified route buffer (RTE_MOD). If the route element is a route function the procedure RTE is called to store the data for each waypoint contained on the route function. If the element is a discontinuity marker a call to DSC_WPT is made. The last possibility is a single waypoint element. The procedure WPT is called to store the basic waypoint data followed by a call to GET_CON to fetch any ALT/SPD/RAD constraint data. If no altitude constraint was found in the constraint data the cruise altitude, if entered, is assigned to the waypoint.

Once stepping through the route buffer is finished a few miscellaneous operations are performed. The created waypoint buffer is examined to see if takeoff and landing runways exist. If not the respective runway addresses are cleared (AIRPTS(2,1), AIRPTS(2,2)). The waypoint buffer may not end in a discontinuity marker so the buffer is checked and the discontinuity is removed if present.

Finally, the procedures PATHDF and RTA_TIMES are called to compute the remaining guidance parameters.

GLOBAL REFERENCES:

VARIABLES
CRZALT MODCNT* RTA_WPT

ARRAYS
AIRPTS* RTE_CNT

RECORD ARRAYS
RTE_MOD WPT_ACT WPT_MOD

FUNCTIONS AND SUBROUTINES
CLRBUF DSC_WPT GET_CON PATHDF RTA_TIMES RTE WPT

Section 6.2.3 EXECUTE/REJECT THE MODIFIED FLIGHT PLAN

When flight plan entries are complete, either for the original clearance or a modified active plan, the pilot must choose between executing or rejecting the provisional flight plan. The EXEC button of the CDU will be lit when execution is allowed. If pressed, the provisional plan becomes the new active flight plan. At this time the MOD buffers will be identical to their active counterparts (RTE_MOD/ WPT_MOD - RTE_ACT/WPT_ACT) and the CDU clearance pages will display "ACT" as the first part of their title line. When the clearance entries were made as modifications to an existing active flight plan the "Erase" option is given. The text "ERASE>" appears on the right hand side of the last display line of the CDU when on the ROUTE, LEGS, or TIME pages. If the pilot presses the line select key adjacent the erase prompt, the changes to the clearance are removed and the "ACT" mode is returned.

Three modules handle execution and rejection of the provisional flight plan. Their descriptions are provided on the following pages.

PAGE 100 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: EXECUTE
FILE NAME: EXECUTE.FOR
PROCESS: SLOW
CALLED BY: CDUEXC, DEMODE
CALLING SEQUENCE: CALL EXECUTE(MODE)

PURPOSE:
To activate the current provisional flight plan.

DESCRIPTION:
This procedure activates the provisional flight plan by copying the MOD route and waypoint buffers to their ACT counterparts. A number of simple steps are perfomed when execution is required. They are enumerated below in the appropriate sequence. Afterward more detailed explanation is provided for those parts requiring it.

. If a "POS" update waypoint starts the new plan, make one last update of the waypoint by calling UPDATE_POS.

. Signal guidance software (HVGUID) that the flight plan is temporarily invalid by clearing the 2D, 3D, and 4D guidance flags.

. If not called with Auto execute flag, identify next "To" waypoint and reset some phase of flight flags.

. Copy the modified route buffer (RTE_MOD) to the active buffer (RTE_ACT). If a discontinuity is encountered terminate the plan at that point.

. Copy the modified waypoint buffer (WPT_MOD) to the active (WPT_ACT). Check altitude and speed definitions at each waypoint to determine possible guidance modes.

. Process execution of holding pattern data if entered by calling HOLD_SET.

. Perform cleanup on the constraint buffer by calling CLEAN_CON.

. Enable guidance remode (SETGD = 2).

. Save new active data which may be modified on next plan changes; airfield info, cruise alt, RTA waypoint.

. Fetch destination runway information from navigation database (AADCOM).

. If not Auto execute mode reset EXEC and DSPLY lights and set the active guidance waypoint pointer (PTR2D) to the previously chosen "To" waypoint. Otherwise check if PTR2D should be set to the active hold waypoint.

. Fill in active display waypoint buffer by calling CREATE_BUF and flag display buffer transmission by clearing GDTIME.

Selecting a "To" waypoint can be complicated. If there was no previous active flight plan the second waypoint is designated. If there was an active plan several tests are made. When a current active leg is part of a holding pattern the last PTR2D index is used since waypoint trimming is not used for holding pattern changes (see DEMODE). When trimming was enabled the pointer is initialized to the second waypoint. However the second waypoint corresponds to the last active waypoint when changes to flight plan started. The airplane may cross waypoints while entries are being made. When this situation occurs the pointer is advanced along the flight plan as along as one-to-one correspondence remains in the following waypoints.

GLOBAL REFERENCES:

VARIABLES
ACTCNT ACTCRZ* ANTLAT* ANTLON* CLBCHNG* COSRH CRZALT
CRZCHNG* DESCHNG* DESCHNG1* GDTIME* GSA* GUID2D* GUID3D*
GUID4D* HLD2D* LT_DSPY* LT_EXEC* MODCNT ORGRWY* PMODE*
POSTIME* PTR2D* RTA_INDX RTA_PTR* RWYHDG RWYLAT* RWYLEN*
RWYLON* RYELEV* SAVPTR* SETGD* SINRH TST3D* TST4D*

ARRAYS
AIRPTS RTE_CNT*

RECORD ARRAYS
ACT_WPTS RTE_ACT* RTE_MOD WPT_ACT WPT_MOD

FUNCTIONS AND SUBROUTINES
CLEAN_CON CREATE_BUF EXIT GET_REAL HOLD_SET SCOSD UPDATE_POS
XLAT_RTE

MODULE NAME: HOLD_SET
FILE NAME: EXECUTE.FOR
PROCESS: SLOW
CALLED BY: EXECUTE, REJECT
CALLING SEQUENCE: CALL HOLD_SET(MODE)

PURPOSE:
To setup active holding pattern data structures.

DESCRIPTION:
This module is called upon the execution of a provisional flight plan or the rejection of modifications to an existing active plan. It determines if a holding pattern exists in the active flight plan and sets up a pointer to the hold 'FIX' waypoint and also saves its name. One other action is taken when the pattern is found, which will depend on whether the call to HOLD_SET was made from EXECUTE or REJECT. On execute, the hold pattern database set up by hold page software, is saved in local memory. On reject the saved active database is restored to replace changes that may have been made to the holding pattern.

GLOBAL REFERENCES:

VARIABLES
ACTCNT HLD_PTR* HLD_WPT*

ARRAYS
START

RECORD ARRAYS
WPT_ACT

FUNCTIONS AND SUBROUTINES
LIB$MOVC3

MODULE NAME: REJECT
FILE NAME: EXECUTE.FOR
PROCESS: SLOW
CALLED BY: LEGS, LEGS_TIME, ROUTE
CALLING SEQUENCE: CALL REJECT

PURPOSE:
To remove all changes to the last active flight plan.

DESCRIPTION:
This module is called when the pilot chooses to reject the changes made to the current active flight plan. The following information sequentially lists the steps taken to restore the active flight plan to the MOD buffers.

. Set the CDU mode to "ACT" and clear POS update waypoint flag (if one existed).

. Copy the active waypoint buffer into its MOD buffer.

. Copy the active route buffer into its MOD buffer.

. Restore holding pattern parameters by calling HOLD_SET.

. Free unused constraint buffer locations by calling CLEAN_CON.

. Restore miscellaneous flight plan variables; airfields, cruise alt, RTA waypoint.

. Turn off EXEC and DSPLY lights.

. Expand restored route buffer to data-link text buffer if enabled.

GLOBAL REFERENCES:

VARIABLES
ACTCNT ACTCRZ CDU_CMD CRZALT* LT_DSPY* LT_EXEC* MODCNT*
ORGRWY PMODE* POSTIME* RTA_INDX RTA_PTR RTA_WPT*

ARRAYS
AIRPTS* RTE_CNT*

RECORD ARRAYS
RTE_ACT RTE_MOD* WPT_ACT WPT_MOD*

FUNCTIONS AND SUBROUTINES
CLEAN_CON EXPAND_RTE HOLD_SET

Section 6.3 THE FLIGHT PLANNING PAGES

The CDU has numerous ways of generating and modifying the aircraft's flight plan. There are seven CDU pages dedicated to this purpose, not including the phase of flight or initialization pages which allow cruise altitude selection. The following sections give a brief outline of the usage for each page followed by descriptions of the software modules used. A detailed functional description of the CDU clearance pages will be provided in another document to be provided by NASA's CDU requirements designer.

PAGE 106 INTENTIONALLY BLANK

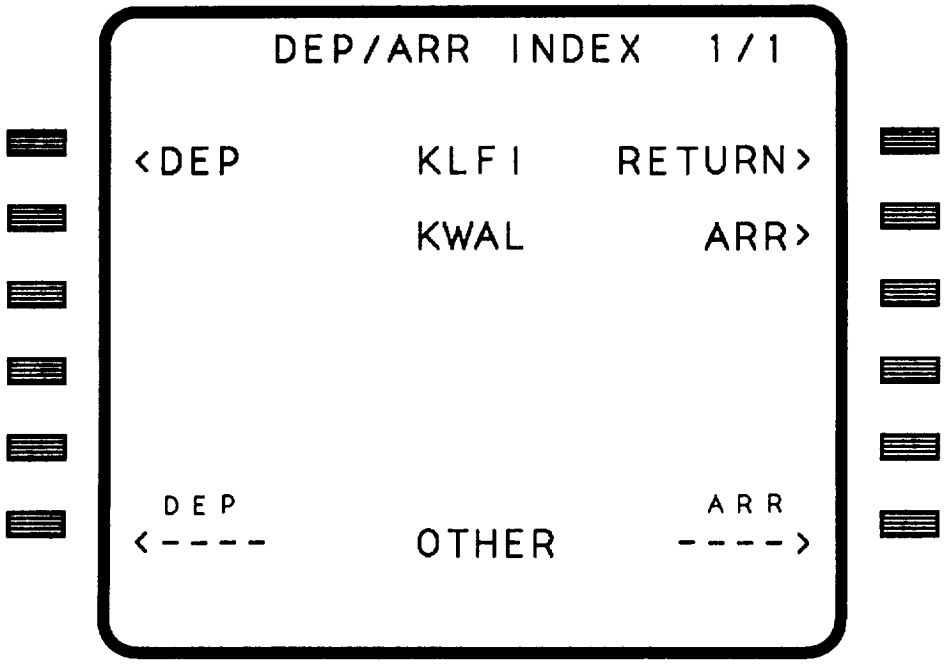
PRECEDING PAGE BLANK NOT FILMED

Section 6.3.1 THE DEPARTURE/ARRIVAL PAGE

The DEPARTURE/ARRIVAL pages provide the flight crew with departure and arrival information for the origin and destination airports or for any other airport in the navigation database (AADCOM). Also, these pages allow the flight crew to insert departure and arrival route elements into the route buffer by pressing the labeled LSKs.

The DEPARTURE/ARRIVAL INDEX page provides access to the DEPARTURE subpage and the ARRIVAL subpage, where specific information about each airport is displayed. On the DEPARTURE subpage, all SIDs and runways listed in AADCOM for the selected airport, are diplayed. On the ARRIVAL subpage, all STARs, approaches and runways for the selected airport are displayed. <SEL> and <ACT> bugs are displayed next to the route elements which are part of the current provisional or active flight plan. Refer to figures 6.1 and 6.2 on the following pages.

PAGE 108 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED

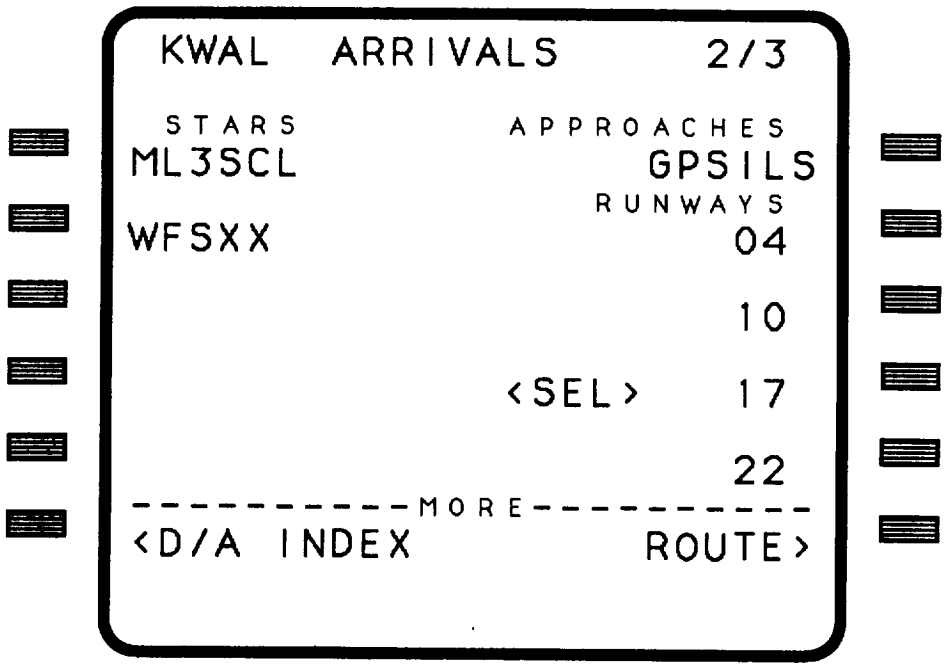


The Departure and Arrivals Index Page

(figure 6.1)

PAGE 110 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED



The Arrivals Page

(figure 6.2)

PAGE 112 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: DA_INPUT
FILE NAME: DEPARR.FOR
PROCESS: SLOW
CALLED BY: DEPARR
CALLING SEQUENCE: CALL DA_INPUT

PURPOSE:
To parse CDU data entries for the DEPARTURE or ARRIVAL page.

DESCRIPTION:
This subroutine is called when a data entry is detected while on the DEPARTURE or ARRIVAL subpage. Valid entries on this page are limited to the following:

. Requesting the DEPARR index page or ROUTE page. If there is data on the scratch pad, it is reprogrammed back onto the scratch pad for use by the requested page.

. Display NEXT or PREVious section of currently dispalyed page

. Insert a route element into the flight plan, by calling MOD_ROUTE, provided that it is not already a part of the provisional or active flight plan. This is only a valid data entry when on the DEPARTURE page and the ORIGIN airfield is shown or when on the ARRIVAL page and the origin or desination airfield is shown.

. Delete a route function from the provisional or active flight plan, by calling MOD_ROUTE.

GLOBAL REFERENCES:

VARIABLES
DST ERCODE* NUMPGS ORG PASS* PGRQST* SUBPAG*

ARRAYS
ENTRY SIDLINE

FUNCTIONS AND SUBROUTINES
DEL_IN FMTOUT MOD_ROUTE REPROG SET_SIDLINE

PAGE 114 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: DEPARR
FILE NAME: DEPARR.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL DEPARR

PURPOSE:
To serve as the DEPARTURE/ARRIVAL page executive module.

DESCRIPTION:
This subroutine is the main procedure for the DEPARTURE/ARRIVAL page software. It performs a few top-level functions beginning with first pass initialization and the computation of a few variables used by other modules. Input to the DEPARTURE/ARRIVAL page is handled by one of two modules, INDX_INPUT or DA_INPUT, depending upon which subpage of the DEPARURE/ARRIVAL page is presently active. This procedure also monitors the global variable PMODE so that in the event the execute button is pressed, the subroutine SET_SIDLINE is called to update the <SEL> and <ACT> bugs on the DEPARTURE or ARRIVAL subpages. A call to the screen update procedure, REFRESH_DA, is made every time the CDU executive calls DEPARR, with one exception. When a new subpage has been requested, the update of the CDU screen is delayed for one pass to allow time for route function information to be updated.

GLOBAL REFERENCES:

VARIABLES
PAGE PASS* PGINIT* PLAN* PMODE SUBPAG* SUBPGINIT

ARRAYS
ENTRY*

FUNCTIONS AND SUBROUTINES
DA_INPUT INDX_INPUT PAGE_COUNT REFRESH_DA SET_SIDLINE

MODULE NAME: INDX_INPUT
FILE NAME: DEPARR.FOR
PROCESS: SLOW
CALLED BY: DEPARR
CALLING SEQUENCE: CALL INDX_INPUT

PURPOSE:
To parse CDU data entries for the DEPARTURE/ARRIVAL INDEX page.

DESCRIPTION:
This subroutine is called when a data entry is detected while on the DEPARTURE/ARRIVAL INDEX page. The following list describes the requests which are valid data entries.

. display DEPARTURE information for the ORIGIN airfield

. display ARRIVAL information for the ORIGIN airfield, for emergency return.

. display ARRIVAL information for the DESTINATION airfield.

. display DEPARTURE or ARRIVAL information for the specified airfield contained in AADCOM.

Upon receipt of a valid data entry, some initialization variables are set, including the PAGE variable which is set to reflect the page number of the requested page.

GLOBAL REFERENCES:

VARIABLES
ADDR DST* ERCODE* ORG* PAGE* SUBPGINIT*

ARRAYS
AIRPTS ENTRY

FUNCTIONS AND SUBROUTINES
DEL_IN LUARP

MODULE NAME: ITEM_ADDR
FILE NAME: DEPARR.FOR
PROCESS: SLOW
CALLED BY: SET_SIDLINE
CALLING SEQUENCE: PATHADDR = ITEM_ADDR (ITEM)

PURPOSE:
To return the address in AADCOM of the first SID or the last STAR or approach in the route buffer.

DESCRIPTION:
This function searches the route buffer for the first occurrance of a route function of type ITEM. The types are identified by an integer value where, approach=5, SID=6, and STAR=7. Because the route buffer may contain more than one route function of a certain type, the function searches the buffer, from bottom to top, for the first SID or the last STAR or approach in the buffer. If searching for a STAR or approach, the search halts when the item is located. If searching for a SID, then the search continues if a SID is found to ensure that it is the first SID in the path.

GLOBAL REFERENCES:

ARRAYS
RTE_CNT

RECORD ARRAYS
RTE_MOD

MODULE NAME: ITEM_COUNT
FILE NAME: DEPARR.FOR
PROCESS: SLOW
CALLED BY: PAGE_COUNT
CALLING SEQUENCE: NUM = ITEM_COUNT(ADDR, OFFST)

PURPOSE:
To determine the number of SIDs, STARs, approaches or runways which are available at the selected airfield

DESCRIPTION:
This function searches for the end of a list of addresses of SIDs, STARs, approaches or runways. It begins the search at location ADDR, increments the search address by the value contained in OFFST, and continues until it finds a zero valued address. It returns the computed number of items in the list.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
GET_LONG

MODULE NAME: MODIFY
FILE NAME: MODIFY.FOR
PROCESS: SLOW
CALLED BY: RT_NEW, MOD_ROUTE
CALLING SEQUENCE: CALL MODIFY(ADDRESS, TYPE, CLEAR_FLG)

PURPOSE:
To place airfield selection into the route buffer.

DESCRIPTION:
This subroutine is called by the Departutes/Arrivals page of the CDU to make modifications to the flight plan. Insertions and Deletions of SIDs, STARs, Approaches, and runways may be requested. Note that insertions become replacements when the inserted type already exists.

Special processing occurs when the third parameter of the call list is set. This flag indicates the desire to return to the departure airfield after takeoff, usually for emergency situations. A waypoint is created at the current aircraft position. This waypoint is created as a 10 second update waypoint (see section 1.5.3.2). The route element selected on the DEP/ARR page is then placed in the route buffer with a call to WAYPOINT or GROUP and the remainder of the flight plan is deleted.

If a selected type is a route function a search of the route buffer is made to find an existing element of the type passed in the parameter list. If found, the element is deleted by a call to KILL and inserted by a call to GROUP.

If a match is not found the position of insertion is determined by the element type. An approach is always placed as the last element in the buffer. SIDs are first unless a takeoff runway is present, in which case they are placed after the two runway waypoints. STARs are placed at the end unless a touchdown runway is defined. They are inserted two positions before the end to account for either two runway waypoints or an approach with its entry waypoint.

Deletions of route functions are requested when the type parameter is set to "DELETE". The actual type of the route function is not needed to remove it from the buffer since the passed address is used to locate the element.

Origin and destination runways have the types "RWY1" and "RWY2". The address parameter contains a pointer to the navigation database when an insertion is requested. If the address value is zero a deletion is desired.

GLOBAL REFERENCES:

VARIABLES
ALTCOR ERCODE* GS LAT LON POSTIME* TIME

ARRAYS
  AIRPTS RTE_CNT

RECORD ARRAYS
  RTE_MOD

FUNCTIONS AND SUBROUTINES
  DEL_RTE GROUP KILL MAKE_WPT ORG_RWY REMOVE WAYPOINT

MODULE NAME: MOD_ROUTE
FILE NAME: DEPARR.FOR
PROCESS: SLOW
CALLED BY: DA_INPUT
CALLING SEQUENCE: CALL MOD_ROUTE(ITEMNAME, ADDIT)

PURPOSE:
To insert or delete route functions on the DEPARR page.

DESCRIPTION:
This subroutine searches for a route function, whose name is given by the input parameter ITEMNAME, in BULK DATA with a call to LUSID. It then calls MODIFY with the address and type of the route function to be inserted or deleted from the provisional or active route buffer. The boolean input parameter ,ADDIT, specifies whether the item is to be inserted or deleted. Finally, this routine calls DEMODE with a parameter of NOEXEC to create the new waypoint buffer and allow the pilot to execute the new path.

Note the special case where a route function has been selected on the ARRIVAL page and the ORIGIN airfield is shown. This indicates an emergency request to return to the departure airfield, and some special processing occurs. The origin airfield becomes the new destination airfield and the origin and destination airfield flags, which affect the informational display and input parsing on the DEPARR page, are updated appropriately.

GLOBAL REFERENCES:

VARIABLES
ADDR DST* ORG* PAGE PASS*

ARRAYS
AIRPTS*

FUNCTIONS AND SUBROUTINES
DEMODE LURWY LUSID MODIFY SET_SIDLINE

MODULE NAME: PAGE_COUNT
FILE NAME: DEPARR.FOR
PROCESS: SLOW
CALLED BY: DEPARR
CALLING SEQUENCE: CALL PAGE_COUNT

PURPOSE:
To compute the number of subpages required to display route function information for a selected airfield.

DESCRIPTION:
If the DEPARTURE subpage has been requested, this subroutine compares the number of SIDs and runways available in BULK DATA for the selected airfield and chooses the larger of the two to determine the number of pages required for the DEPARTURE subpage. If the ARRIVAL subpage has been requested, it compares the number of STARs with the number of approaches and runways and chooses the larger of the two to compute the necessary number of subpages. One page is required to display five lines of route functions.

GLOBAL REFERENCES:

VARIABLES
ADDR NUMAPP NUMBOTH NUMPGS* NUMSS PAGE

FUNCTIONS AND SUBROUTINES
GET_LONG ITEM_COUNT

MODULE NAME: REFRESH_DA
FILE NAME: DEPARR.FOR
PROCESS: SLOW
CALLED BY: DEPARR
CALLING SEQUENCE: CALL REFRESH_DA

PURPOSE:
To update the CDU display for the DEPARURE/ARRIVAL pages.

DESCRIPTION:
This subroutine updates the CDU display for the DEPARTURE/ ARRIVAL INDEX page, the DEPARTURES subpage or the ARRIVALS subpage with calls to FMTOUT. The entire screen is updated every eight consecutive calls to this subroutine. The value of PASS determines which particular lines are updated. During the first call of the cycle, the appropriate page title is output along with an indication of the current and last page numbers.

If the DEPARTURE/ARRIVAL INDEX page is currently active, then the name of the ORIGIN airfield is shown on line #2 along with the labels for the LSKs which provide access to the DEPARTURE and ARRIVAL subpages. If no ORIGIN airfield exists in the current provisional flight plan, then blanks are displayed in place of the airfield name. Likewise, the name of the DESTINATION airfield is displayed on line #4 along with the label for the LSK which provides access to the ARRIVAL page. Again, if no DESTINATION airfield is present in the current flight plan, then blanks are diplayed in place of the name. Lines #11 and #12 of the display contain labels for the LSKs which provide access to the DEPARTURE and ARRIVAL subpages for information on any airfield contained in AADCOM.

If either the DEPARTURE or ARRIVAL subpage is currently active then line #1 will contain headings for the lists of SIDs, STARs, approaches or runways available at the selected airfield. Route element information is displayed on lines #2, #4, #6, #8, and #10. This route element information is contained in the array, SIDLINE, which is updated by the subroutine, SET_SIDLINE. Line #11 contains a dashed line, with the label "more" if additional pages of information are available. Line #12 contains labels for the LSKs which provide access to the INDEX and ROUTE pages.

GLOBAL REFERENCES:

VARIABLES
ADDR LBL NUMAPP NUMPGS PAGE PASS* SUBPAG

ARRAYS
AIRPTS DASHES SIDLINE

FUNCTIONS AND SUBROUTINES
FMTOUT

MODULE NAME: SET_SIDLINE
FILE NAME: DEPARR.FOR
PROCESS: SLOW
CALLED BY: DEPARR, DA_INPUT, MOD_ROUTE
CALLING SEQUENCE: CALL SET_SIDLINE

PURPOSE:
To format lines of route element information for display on the DEPARTURE or ARRIVAL page of the CDU.

DESCRIPTION:
This subroutine is called to update the information in the array SIDLINE, which contains appropriate route element information for the selected airfield. This information is displayed by the subroutine REFRESH_DA. Each element of the array SIDLINE is a string of 24 characters and corresponds to a display line on the CDU. The information contained in SIDLINE is updated when either the DEPARTURE or ARRIVAL page of the CDU is initially requested, and whenever the NEXT or PREVious subpages of the DEPARTURE or ARRIVAL page are requested.

If the DEPARTURE page is currently active, then the left side of each line will contain the name of a SID and the right side of each line will contain a runway number, available in AADCOM for the selected airfield. If the selected AIRFIELD is the ORIGIN airfield then <SEL> or <ACT> bugs will be displayed next to the route elements which are part of the current provisional or active flight plans. If the current clearance mode is ACT then <ACT> bugs will be displayed, otherwise <SEL> bugs will be displayed.

If the ARRIVAL page is currently active, the left side of each line will contain available STARs and the right side will contain the names of available approaches and runways for the selected airfield. If the selected airfield is the DESTINATION airfield then <SEL> or <ACT> bugs will be appropriately displayed next to route elements which are present in the current provisional or active flight plans, again, dependent upon the current clearance mode.

Based on the value of SUBPAGE, this routine determines which portion of the route element lists should be displayed, and stores the appropriate names in the array SIDLINE. Also it sets a flag which tells the subroutine REFRESH_DA where to display the header for the list of runways for a given airfield. This is not a static position since the runway header must follow the last approach for a selected airfield.

GLOBAL REFERENCES:

VARIABLES
ADDR DST NUMAPP NUMBOTH NUMSS ORG PAGE PLAN* PMODE
RWYLABEL* SUBPAG

ARRAYS
AIRPTS SIDLINE*

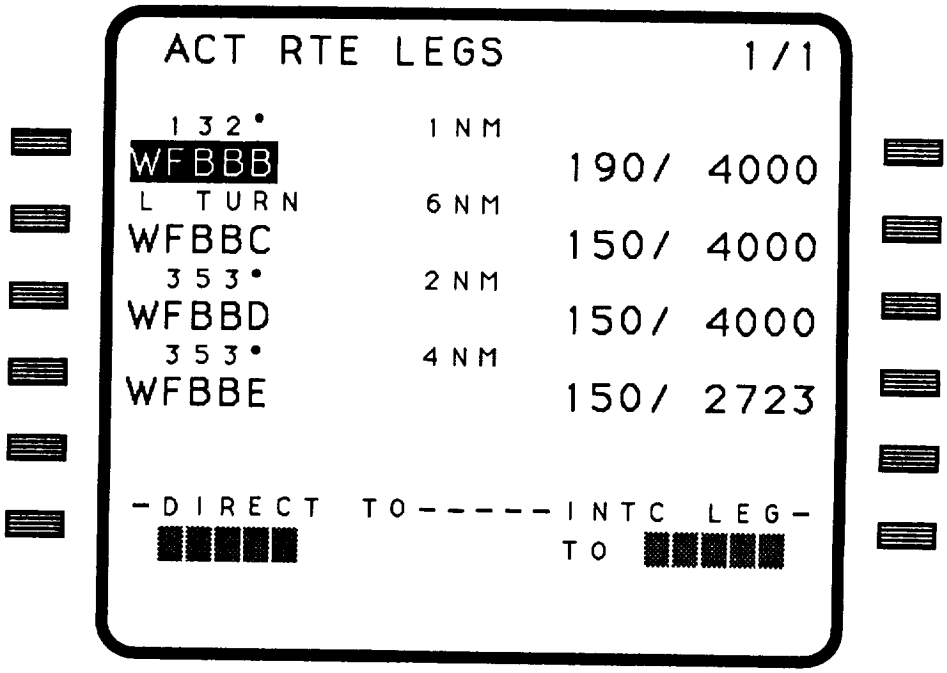
FUNCTIONS AND SUBROUTINES
GET_CHAR GET_LONG ITEM_ADDR

Section 6.3.2 THE DIRECT/INTERCEPT PAGE

This page is a variation of the LEGS page. The CDU display is the same except for "Direct To" and "Intercept Leg" prompts on the bottom. Section 6.3.4 describes the standard LEGS page and should be referenced to understand the DIR/INTC page. This section describes the three modules unique to the DIR/INTC page which are on the file INTC.FOR (Refer to figure 6.3 on the following page).

Two operations are performed on this page besides the standard LEGS page functions. The first option creates a waypoint at the airplanes present position and connects it to the selected "To" waypoint. If the chosen waypoint is part of the current flight plan the remaining waypoints along the path are kept. The second option, like the first, requires a "To" waypoint entry. Once selected the CDU screen is updated to prompt for an inbound bearing. A new waypoint is generated 100 nautical miles away from the selected waypoint to define a leg with the desired inbound bearing. Remaining flight plan waypoints are kept for this operation also.

Note that the pilot defined waypoint created at the aircraft position is updated every ten seconds to the current aircraft position until the flight plan is made active.



The Direct/Intercept Page

(figure 6.3)

PAGE 128 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: DIRECT
FILE NAME: INTC.FOR
PROCESS: SLOW
CALLED BY: LEGS
CALLING SEQUENCE: CALL DIRECT

PURPOSE:
To create a "Direct To" flight plan.

DESCRIPTION:
This subroutine is called when either a "Direct To" or "Intercept Leg" entry is made on the DIR/INTC page of the CDU. The selected waypoint's name is in the global CDU entry buffer (ENTRY). Two valid situations may occur. If the entered waypoint is found on the current flight plan, all waypoints before the chosen one are removed from the route buffer and the current aircraft position is inserted as the "From" waypoint. When the waypoint is not entered on the current flight plan a two waypoint path is generated consisting of the current position and the selected waypoint.

A "Direct To" may not be performed to a holding pattern waypoint. If attempted an error condition is flagged. The function MAKE_WPT is called to create the waypoint at the aircraft position. The new waypoint is assigned the current altitude and ground speed of the aircraft. The "From" format of the standard legs page is requested once complete.

GLOBAL REFERENCES:

VARIABLES
ALTCOR ERCODE* FROMPG* GS LAT LON MODCNT PGRQST* PMODE
POSTIME* PTR2D TIME

ARRAYS
AIRPTS ENTRY RTE_CNT*

RECORD ARRAYS
RTE_MOD* WPT_MOD

FUNCTIONS AND SUBROUTINES
BREAK MAKE_WPT OPEN PAD_NAME TRIM_WPTS WPT_ID

PAGE 130 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: INTC_MGR
FILE NAME: INTC.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL INTC_MGR

PURPOSE:
To call the LEGS executive requesting the DIR/INTC variation.

DESCRIPTION:
The only thing done by this module is to call LEGS with the DIR/INTC parameter.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
LEGS

MODULE NAME: INTERCEPT
FILE NAME: INTC.FOR
PROCESS: SLOW
CALLED BY: LEGS
CALLING SEQUENCE: CALL INTERCEPT

PURPOSE:
To create an "Intercept Leg" waypoint.

DESCRIPTION:
This subroutine is called when the final entry is made on an "Intercept Leg" creation. The inbound bearing is decoded from the CDU entry line and used to create a pilot defined waypoint 100 nautical miles from the previously selected "To" waypoint. The last "From" waypoint, generated by DIRECT after the "To" waypoint selection, is replaced by the new waypoint to form the desired inbound path leg.

GLOBAL REFERENCES:

VARIABLES
ALTCOR ERCODE* FROMPG* GS PGRQST*

ARRAYS
ENTRY

RECORD ARRAYS
RTE_MOD* WPT_MOD

FUNCTIONS AND SUBROUTINES
FLTVAL MAG_VAR MAKE_WPT NEW_POS

Section 6.3.3 THE HOLD PAGE

The HOLD page is used by the flight crew to create a holding pattern at the present airplane position or at any waypoint contained in the waypoint buffer (except DME turn waypoints). When the HOLD key of the CDU is pressed, a special variation of the LEGS page format, the LEGS-HOLD page, is displayed. If a holding fix is selected, a variable HLD_WPT is assigned the name of the holding fix and a holding pattern is set up with default parameters, by calling HOLD_INIT. The holding pattern is inserted into the provisional route buffer as well as the waypoint buffer and the HOLD page format is displayed on the CDU. The HOLD page allows the flight crew to modify the default parameters of the holding pattern and execute the new programmed route. Holding pattern parameters which may be modified include the direction of turns in the holding pattern, the holding speed and the bearing to the holding fix. The flight crew may also specify the time required to fly a straight leg of the holding pattern or the length of a straight leg in nautical miles (Refer to figure 6.4 and 6.5 on the following pages).

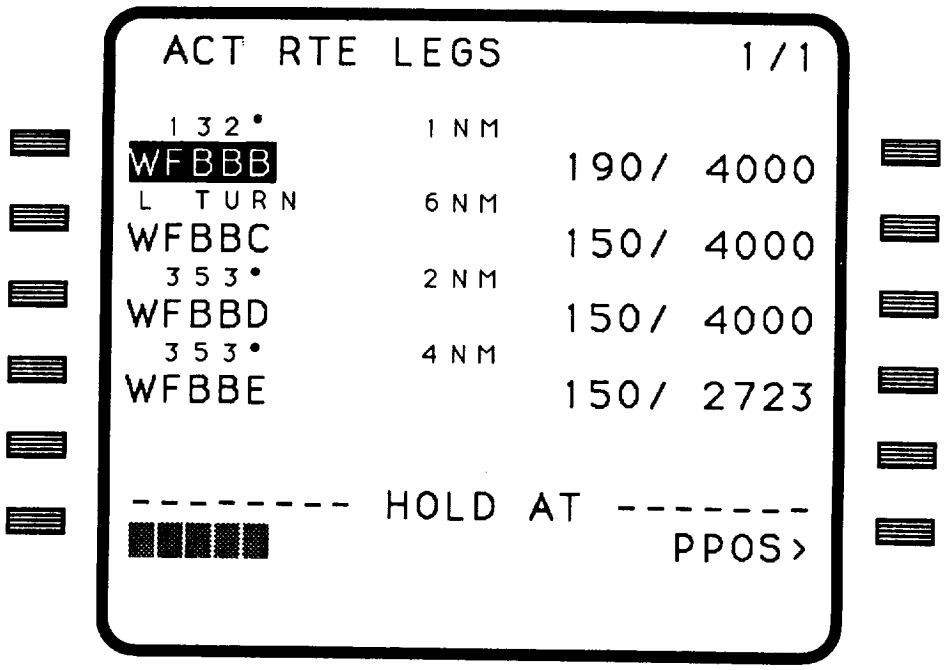
When a holding pattern is created, four points which comprise the holding pattern are defined and inserted into the provisional waypoint buffer. The names of the waypoints which make up the holding pattern are HOLD1, HOLD2, HOLD3 and HOLD4. These waypoints are inserted into the path just prior to the position of the holding fix. The holding fix and the waypoint HOLD1 have the same latitude, longitude and altitude (if one exists for the fix) and are combined with the other hold waypoints to form a path section.

When the autopilot is engaged flying a holding pattern, the airplane repeatedly flys the holding pattern until a request is made to exit. Each time the airplane passes waypoint HOLD4, a check is made in HVGUID to see if a request has been made to exit the holding pattern, indicated by EXHOLD. If no request has been made to exit the holding pattern, the active "to" waypoint pointer is set back to point to HOLD1. Otherwise, the holding fix becomes the active "to" waypoint.

It is possible to create a holding pattern in the provisional flight plan which differs from the holding pattern in the active flight plan. Additional bookkeepng is performed to provide this capability.

PAGE 134 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

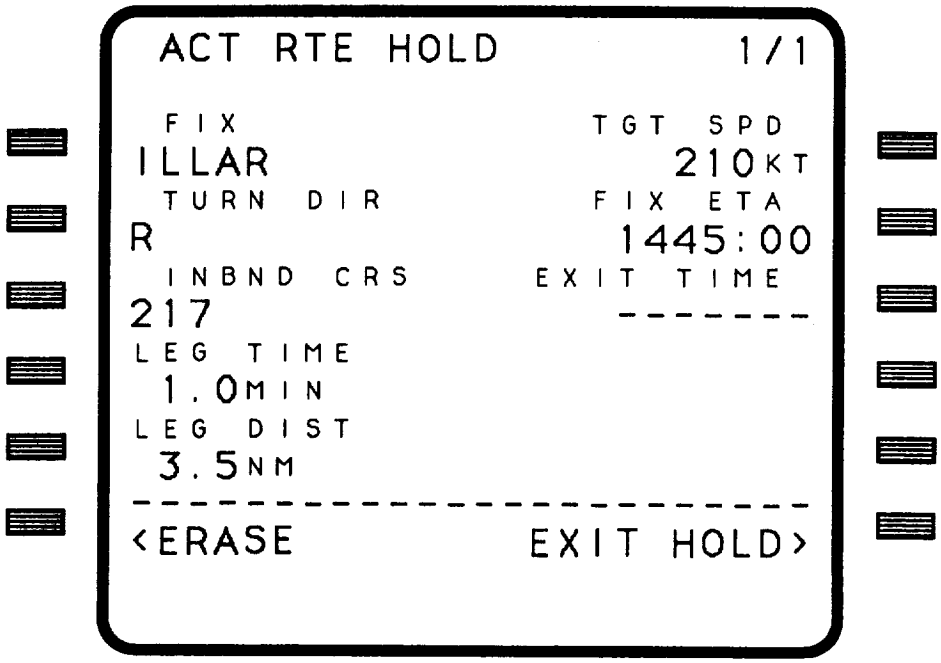


The Legs Hold Page

(figure 6.4)

PRECEDING PAGE BLANK NOT FILMED

PAGE 136 INTENTIONALLY BLANK



The Hold Page

(figure 6.5)

PAGE 138 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: GET_ETA
FILE NAME: HOLD.FOR
PROCESS: SLOW
CALLED BY: HOLD_INPUT, REFRESH_HOLD
CALLING SEQUENCE: ETA = GET_ETA

PURPOSE:
To compute the estimated time of arrival (ETA) at the holding fix.

DESCRIPTION:
This routine computes the ETA at the selected holding fix. It accumulates the distance to the holding fix by adding the distance to the next waypoint, DTOGO, to the distances along each of the legs of the path which lie between the airplane and the holding fix. It divides the accumulated sum by the current ground speed in feet per second and adds the result to the current time. The ETA is computed and displayed only when the current clearance mode is ACT, the airplane position is within the holding pattern and ground speed is greater than zero.

GLOBAL REFERENCES:

VARIABLES
DTOGO GS HLD_PTR PMODE TIME TOWPT

RECORD ARRAYS
WPT_ACT

PAGE 140 INTENTIONALLY BLANK
PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: HOLD_INIT
FILE NAME: HOLD.FOR
PROCESS: SLOW
CALLED BY: HLD_IN
CALLING SEQUENCE: CALL HLD_INIT(INDEX)

PURPOSE:
To create an initial holding pattern with default parameters.

DESCRIPTION:
This routine is called from the LEGS-HOLD page when a holding fix is selected. The input parameter is an index into the waypoint buffer designating the position of the holding fix. When this routine is called some flags are initialized and a holding pattern is created with the following defaults:

- holding pattern turns are right turns
- inbound course to fix is path bearing at fix waypoint
- hold speed is 210 kts
- if fix altitude is greater than 14000 feet, the default leg time is 1.5 minutes, otherwise it is 1 minute

This routine calls the the LENGTHS routine to compute the radius of the holding pattern turns, HLDRAD, and the lengths of the straight legs of the holding pattern, LEG_LEN. It calls POINTS to compute the positions of the four waypoints which define the holding pattern and stores the necessary values in HLDBUF.

GLOBAL REFERENCES:

VARIABLES
DELHOLD* EXHOLD* HLD_WPT* MODCNT PGINIT*

RECORD ARRAYS
WPT_MOD

FUNCTIONS AND SUBROUTINES
ANGL LENGTHS MTH$SIGN POINTS

MODULE NAME: HOLD_INPUT
FILE NAME: HOLD.FOR
PROCESS: SLOW
CALLED BY: HLDWPT
CALLING SEQUENCE: CALL HOLD_INPUT(I)

PURPOSE:
To parse CDU data entries for the main HOLD page.

DESCRIPTION:
This subroutine is called when a data entry is detected while on the HOLD page. Valid entries on this page are limited to the following:

. Deleting the holding pattern from the provisional or active flight plan.

. Echoing the ETA into the scratch pad.

. Toggling the turn direction of the holding pattern.

. Requesting exit of holding pattern.

. Entering a new holding pattern speed, leg length or leg time.

. Cancelling the deletion of or exit from a holding pattern.

Note that once a holding pattern has been executed and flown, the holding pattern remains on the MAP display, even after it has been exited. If the HOLD page of the CDU is requested, the holding pattern information is displayed and the message "EXIT HOLD PATTERN ARMED" remains on line #12 of the CDU. To enable removal of the holding pattern from the flight plan and MAP display, after the holding pattern has been exited, special input processing has been implemented. The word "DELETE" may be entered using LSK-L6 or LSK-R6. This displays the ERASE label for LSK-L6 and therefore allows the holding pattern to be erased. This required special processing in the flight plan modification code since "dead" waypoints cannot be deleted under normal conditions.

```
GLOBAL REFERENCES:

VARIABLES
  DELHOLD* ERCODE* EXHOLD* HLD2D* LT_EXEC* PMODE PTR2D
  TIMED_LEG

ARRAYS
  ENTRY

RECORD ARRAYS
  WPT_MOD

FUNCTIONS AND SUBROUTINES
  ANGL DEL_IN DEMODE FLTVAL FMTOUT FMTTIM GET_ETA KILL
  LENGTHS MTH$SIGN POINTS
```

MODULE NAME: HLD_MGR
FILE NAME: HOLD.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL HLD_MGR

PURPOSE:
To call the appropriate HOLD page module.

DESCRIPTION:
This routine determines which HOLD page format should be displayed and calls the appropriate executive module. The HOLD page format is displayed only if the variable HLD_WPT contains the name of a fix waypoint, otherwise the LEGS-HOLD page is displayed. When the HOLD format is required, the routine HLDWPT is called with the index of the holding fix. When the LEGS page is required, LEGS is called with a parameter of 2 to indicate that the HOLD variation of the LEGS format is to be displayed.

GLOBAL REFERENCES:

VARIABLES
HLD_WPT

RECORD ARRAYS
WPT_MOD

FUNCTIONS AND SUBROUTINES
HLDWPT INDX LEGS

MODULE NAME: HLDWPT
FILE NAME: HOLD.FOR
PROCESS: SLOW
CALLED BY: HLD_MGR
CALLING SEQUENCE: CALL HLD_MGR(INDEX)

PURPOSE:
To serve as the HOLD page executive module

DESCRIPTION:
This subroutine is the main procedure for the HOLD page software. It performs a few top-level functions including first pass initialization. The input parameter is an index into the waypoint buffer designating the position of the holding fix. Input to the HOLD page is handled by the module HOLD_INPUT. A call to the screen update module, REFRESH_HOLD, is made every time the HOLD page executive module, HLDWPT, is called.

GLOBAL REFERENCES:

VARIABLES
PASS* PGINIT*

ARRAYS
ENTRY*

FUNCTIONS AND SUBROUTINES
HOLD_INPUT REFRESH_HOLD

MODULE NAME: INDX
FILE NAME: HOLD.FOR
PROCESS: SLOW
CALLED BY: HLD_MGR, LENGTHS
CALLING SEQUENCE: I = INDX(NAME)

PURPOSE:
To locate the holding fix in the waypoint buffer.

DESCRIPTION:
This function is called with the five character name of the holding fix waypoint. The waypoint buffer is searched and the index into the structure is returned. A zero index is returned when the fix waypoint is not found in the waypoint buffer.

GLOBAL REFERENCES:

VARIABLES
MODCNT

RECORD ARRAYS
WPT_MOD

MODULE NAME: LENGTHS
FILE NAME: HOLD.FOR
PROCESS: SLOW
CALLED BY: HOLD_INIT, HOLD_INPUT
CALLING SEQUENCE: CALL LENGTHS(TIME, SPD, LEN, RAD)

PURPOSE:
To compute the radius of the turns and the lengths of the straight legs of the holding pattern.

DESCRIPTION:
This routine is called on creation of a holding pattern or when leg time or holding speed are modified by crew inputs to the CDU HOLD page. The input parameters are the desired time, in minutes, to fly one straight leg of the holding pattern and the requested speed in knots. The length for the straight legs of the holding pattern as well as a new turn radius for the holding pattern are computed using the following equations:

length = speed in feet per second * time in seconds

$$\text{radius} = \frac{(\text{true airspeed} + \text{wind speed})^2}{(\text{gravitational acceleration} * \tan(\text{nominal bank angle}))}$$

GLOBAL REFERENCES:

VARIABLES
HLD_WPT

RECORD ARRAYS
WPT_MOD

FUNCTIONS AND SUBROUTINES
INDX

MODULE NAME: POINTS
FILE NAME: HOLD.FOR
PROCESS: SLOW
CALLED BY: HOLD_INIT, HOLD_INPUT
CALLING SEQUENCE: CALL POINTS(I, RAD, LEGLEN, INCRS, SPD)

PURPOSE:
To create the four waypoints which define a holding pattern.

DESCRIPTION:
This routine computes the positions of the four waypoints which define a holding pattern at the selected holding fix. The input parameters are as follows:

I	- an index into the waypoint buffer designating the position of the holding fix
RAD	- the holding pattern turn radius in feet
LEGLEN	- the length in feet of one of the holding pattern straight legs
INCRS	- the bearing of the path segment preceeding the holding fix
SPD	- the desired speed for the holding pattern (knots)

For each waypoint, the latitude and longitude are computed with a call to PROJPOINT. Along with latitude and longitude, the altitude, speed, turn radius, and associated navaid must be stored in HLDBUF for use by the path definition modules. The speed and turn radius are determined by the values of the input parameters RAD and SPD. The associated navaid is the the same as that of the holding fix. If the altitude flag in the waypoint buffer is set for the holding fix, then the altitude of the hold waypoints are set equal to the altitude of the holding fix, otherwise the default altitude of 15,000 feet is used. Also, note that the holding pattern turns are defined as DME turns. The latitude and longitude define the turn center, and therefore the bearing from the turn center to the inbound waypoint must be stored, and the altitude must be negated to indicate that they are inbound waypoints. For the outbound waypoints, the turn angle must be stored. (see AADCOM format description for SID/STAR Route functions)

GLOBAL REFERENCES:

RECORD ARRAYS
HLDPTN* HLDPTS* WPT_MOD

FUNCTIONS AND SUBROUTINES
ANGL LOCAL_ERAD MTH$SIGN PROJPOINT SCOSD

MODULE NAME: PROJPOINT
FILE NAME: HOLD.FOR
PROCESS: SLOW
CALLED BY: POINTS
CALLING SEQUENCE: CALL PROJPOINT(PT1, BRG, DIST, PT2)

PURPOSE:
To compute a waypoint latitude and longitude.

DESCRIPTION:
This routine computes the latitude and longitude of a waypoint given a reference waypoint and a bearing and distance from that point. It uses LATFT and LONFT which are created by LOCAL_ERAD. The PTS structure is used so that the computed latitudes and longitudes can be stored directly into HLDBUF.

GLOBAL REFERENCES:

VARIABLES
LATFT LONFT

FUNCTIONS AND SUBROUTINES
SCOSD

MODULE NAME: REFRESH_HOLD
FILE NAME: HOLD.FOR
PROCESS: SLOW
CALLED BY: HLDWPT
CALLING SEQUENCE: CALL REFRESH_HOLD(I)

PURPOSE:
To update the CDU display for the HOLD page.

DESCRIPTION:
This subroutine updates the CDU display for the HOLD page with calls to FMTOUT. The entire screen is updated every fourteen consecutive calls to this subroutine. The value of PASS determines which particular lines are updated. During the first call of the cycle, the page title is output along with an indication of the current and last page numbers. Information about the holding pattern is displayed on lines #1 through #10. This information includes:

. the name of the holding fix
. the direction of holding pattern turns
. the target speed for the holding pattern
. the holding pattern leg time and leg distance
. the bearing of the leg which aproaches the holding fix
. the estimated time of arrival at the holding fix
. the desired holding pattern exit time

The LSK labels which are displayed on line #12 depend upon current clearance mode. If the current clearance mode is original or MOD then an ERASE label is displayed on the left side of this line, if the current clearance mode is ACT then initially the ERASE label is displayed on the left and EXIT HOLD is displayed on the right side of this line. If the LSK labelled EXIT HOLD has been pressed then the message, "EXIT HOLD PATTERN ARMED", will be displayed on line #12, and if the LSK labelled ERASE has been pressed when the current clearance mode is ACT then the message, "DELETE HOLDING PATTERN" will be displayed on line #12.

GLOBAL REFERENCES:

VARIABLES
DELHOLD EXHOLD HLD_WPT PASS* PMODE

RECORD ARRAYS
WPT_MOD

FUNCTIONS AND SUBROUTINES
ANGL FMTOUT FMTTIM FSTRNG GET_ETA MTH$SIGN TITLE

Section 6.3.4 THE LEGS PAGE

The LEGS page of the CDU allows the entry, manipulation, and application of constraints to flight plan waypoints. Four separate CDU pages actually use the LEGS page format. The various pages are listed below along with CDU access information.

LEGS - The standard legs page which is accessed by using the LEGS key or selecting the "<LEGS" option on the ROUTE INDEX page.

DIRECT/INTERCEPT - This page uses the LEGS format except for the addition of the box prompts displayed on the bottom which allow the entry of the destination waypoint. Once the waypoint is selected this page automatically transfers to the standard legs page. The DIR/INTC key is the only access to this page.

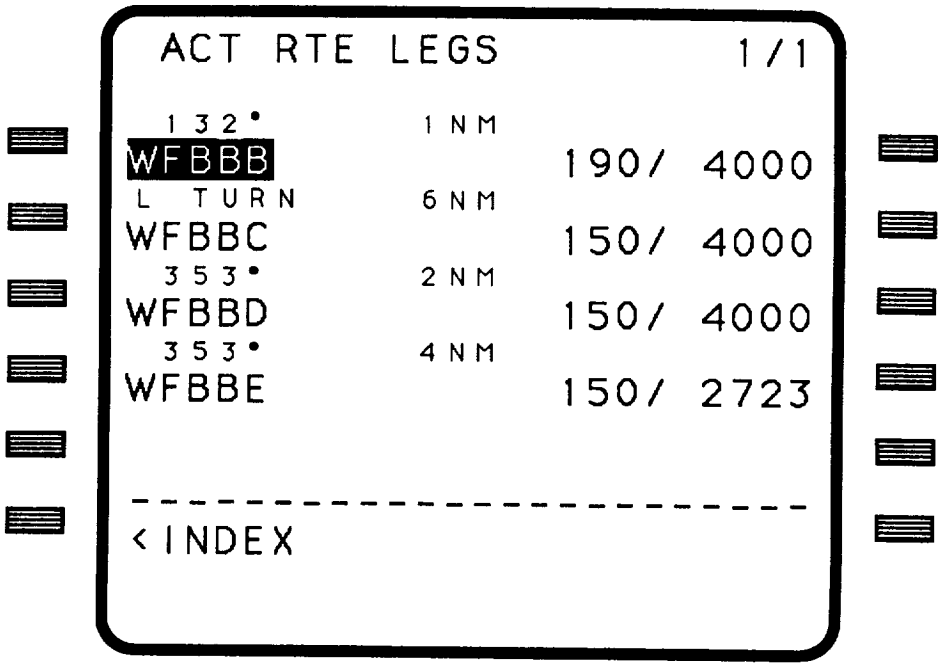
HOLD - This page uses the LEGS format only until the hold waypoint is selected, at which time the hold page uses its unique page format. The only deviation from the standard legs page is the box prompts provided for hold waypoint entry.

FROM WAYPOINT - This page is accessed by selecting the "<FROM WPT" prompt on the ROUTE INDEX page. This page differs from the standard legs page in that only the first waypoint shown. The active LEGS page starts with the "To" waypoint while the FROM page uses the "From" waypoint. The modified (MOD) LEGS page starts with the second flight plan waypoint while the FROM page uses the first. Both pages are identical in the initial clearance mode (start with #1).

Note that any LEGS page operations may be performed while on the other variations of the LEGS page. Refer to figure 6.6 for a picture of the standard LEGS page.

Individual waypoints may be entered and deleted on the LEGS page. The altitude, ground speed, and turn radius constraints associated with the waypoint may also be entered. As many "sub-pages" as necessary are maintained to cover the entire flight plan. The current and last page numbers are shown on the LEGS title line. For a detailed functional description of the LEGS page refer to the CDU requirements produced by Charlie Knox of NASA. The remaining pages of the section explain the 25 modules associated with the LEGS page. Other sections of this document must be referenced for information about the variations of the standard LEGS page.

PAGE 152 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED



The Legs Page

(figure 6.6)

PAGE 154 INTENTIONALLY BLANK

MODULE NAME: ADD_WPT
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: HLD_POS, WPT_DATA
CALLING SEQUENCE: CALL ADD_WPT(INDEX)

PURPOSE:
To insert a waypoint into the flight plan.

DESCRIPTION:
This procedure adds a waypoint to the flight plan by creating a waypoint in the route buffer. Three cases must be accounted for. The waypoint may be appended to the end of the route buffer, inserted at a route dicontinuity, or inserted between flight plan waypoints. When inserted between waypoints that are part of a route function, the route function must be split into separate parts. This procedure is also used to update the active "To" waypoint. When a waypoint which appears further along in the flight plan is entered at the current "To" waypoint, which is highlighted in reverse video, the flight plan updated to reflect the new destination waypoint. All waypoints behind the new "From" waypoint are removed from the flight plan, which must be manually activated to become the new active flight plan.

The subroutine WAYPOINT is called to actually perform the waypoint insertion. Checks are made prior to calling WAYPOINT to identify the situations mentioned above. When the waypoint is inserted within an existing route function the procedure SPLIT is called to break the route function at the selected waypoint. The insertion is then made between the two new route function pieces.

GLOBAL REFERENCES:

VARIABLES
ACTCNT ERCODE* MODCNT PMODE TOWPT

ARRAYS
AIRPTS ENTRY RTE_CNT

RECORD ARRAYS
RTE_MOD WPT_ACT WPT_MOD

FUNCTIONS AND SUBROUTINES
BOUNDS PAD_NAME RTE_WPT SPLIT TRIM_WPTS WAYPOINT WPT_ADDR
WPT_ID

PAGE 156 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: ALTX
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: FLT_TYPE_INP, LINK_EA, NMBRS, PFINP
CALLING SEQUENCE: ALT = ALTX(TEXT, COUNT)

PURPOSE:
To decode altitude entries.

DESCRIPTION:
This function evaluates an ASCII numeric string which represents an altitude. Note that values entered with three or less digits are assumed to be flight levels. Any value greater than 18,000 feet must not have non-zero tens or ones digits since it will be displayed as a flight level. The CDU error code value may be set to reflect an "OUT OF RANGE" or "BAD FORMAT" error. Out of range errors ocuur when the value is not between 0 and 40,000 feet.

GLOBAL REFERENCES:

VARIABLES
ERCODE

FUNCTIONS AND SUBROUTINES
FLTVAL

MODULE NAME: BOUNDS
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: ADD_PLAN, ADD_WPT, HLD_IN, KILL_WPT, LINK_PD, MERGE, NEXT_WPT, TRIM_WPTS, RTE
CALLING SEQUENCE: CALL BOUNDS(INDEX,IN_OFS,OUT_OFS,STEP)

PURPOSE:
To find entry/exit waypoint offsets.

DESCRIPTION:
This subroutine is called to compute the byte offsets, from the start of a route function defined in the route buffer, of the entry and exit waypoint pointers (see section 1.5.1.1 for database formats). When one of the waypoints is not defined a zero is returned as its offset. The number of bytes between consecutive waypoints in the route function is also returned. Note that the STEP may be a negative value since airways may be flown in either direction.

The parameter list for BOUNDS consists of one input and three output values. The first is the index into the route buffer of the chosen route function. The output parameters are the entry waypoint offset, exit waypoint offset, and waypoint separation respectively.

GLOBAL REFERENCES:

RECORD ARRAYS
RTE_MOD

FUNCTIONS AND SUBROUTINES
ENTRY_WPT

MODULE NAME: DSP_WPTS
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: LEGS
CALLING SEQUENCE: CALL DSP_WPTS(PAGE_ID)

PURPOSE:
To create CDU display data for the LEGS page.

DESCRIPTION:
This subroutine is called to update the CDU display screen with the information pertinent to the "LEGS" page of the CDU. The entire screen is updated every six consecutive calls to this procedure. During the first call of the cycle the title line and fixed labels are generated. On subsequent calls the waypoint information for one of the five available slots on the screen is updated.

Waypoint information is shown on CDU line pairs starting with #2/#3 and ending with #10/#11. Three different things can occupy a line pair. The lines are blanked when finished with waypoint buffer elements. A route discontinuity marker is shown for positions which correspond to a break in the flight plan. Lines that show waypoint data have the waypoint name, speed, altitude, inbound bearing, and inbound leg distance. Other information appears with the waypoint data at certain times. The "<CTR>" bug is shown on the map center waypoint during Plan mode. The waypoint radius override symbol "R" is placed on waypoints which have a manually entered turn radius. The waypoints which were assigned a zero turn radius because of bad flight plan geometery are indicated by the "*" symbol. When the displayed waypoint is the "To" waypoint of the active flight plan the name is shown in reverse video and the inbound distance is from the airplane, not the previous waypoint.

Note that the altitude and speed fields may be dashed when their respective constraints are undefined. When shown, the values may be either small or large font depending on the constraint type. The description for the module XLAT_RTE discusses constraint types.

GLOBAL REFERENCES:

VARIABLES
  CTR DTOGO FIRST_PTR LASTPG MODCNT PAGE PASS* PLANM PMODE
  TOWPT

ARRAYS
  BOXES

RECORD ARRAYS
  WPT_MOD

FUNCTIONS AND SUBROUTINES
  FMTOUT FSTRNG HLD_END INBOUND INTC_END LEG_END STEPS TITLE

MODULE NAME: HLD_END
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: DSP_WPTS
CALLING SEQUENCE: CALL HLD_END

PURPOSE:
To create CDU display labels for the LEGS page.

DESCRIPTION:
This procedure updates CDU display lines #11 and #12 for the HOLD variation of the LEGS page. The "HOLD AT" query is placed on line #11. Line #12 contains the box prompts and "PPOS>" response which may be selected using either LSK-L6 or LSK-R6.

GLOBAL REFERENCES:

ARRAYS
BOXES

FUNCTIONS AND SUBROUTINES
FMTOUT

MODULE NAME: HLD_IN
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: HLD_POS, WPT_DATA
CALLING SEQUENCE: CALL HLD_IN(WPT_INDEX)

PURPOSE:
To initiate the processing of a selected hold waypoint.

DESCRIPTION:
This procedure is called when a holding pattern is requested on the LEGS-HOLD page. The input parameter is an index into the waypoint buffer designating the selected waypoint. If the parameter is zero the waypoint buffer is searched for the name stored on the global CDU entry line, ENTRY. The procedure HOLD_INIT is called to create the holding pattern waypoints in HLDBUF. If the hold waypoint is part of a route function the route function waypoints are separated into two pieces by calling the procedure SPLIT. The holding pattern, which consists of an entry waypoint and a hold route function, is inserted before the hold waypoint in the route buffer. The last step is to automatically signal the CDU executive to perform a page change to the hold page display.

GLOBAL REFERENCES:

VARIABLES
ERCODE* HLD_WPT* MODCNT PGRQST* START

RECORD ARRAYS
RTE_MOD* WPT_MOD

FUNCTIONS AND SUBROUTINES
BOUNDS HOLD_INIT OPEN PAD_NAME RTE_WPT SPLIT WPT_ADDR

MODULE NAME: HLD_POS
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: LEGS
CALLING SEQUENCE: CALL HLD_POS

PURPOSE:
To create a holding pattern at present position.

DESCRIPTION:
This subroutine inserts a holding pattern about the airplane's present position on the flight plan. A "PPOS" entry is simulated to create the "hold waypoint" in the route buffer by calling ADD_WPT. The route discontinuity generated from the call is removed and aircraft altitude and ground speed are set up as constraints. A new waypoint buffer is created by calling XLAT_RTE which is used when HLD_IN is called to create the holding pattern at the new PPOS waypoint.

GLOBAL REFERENCES:

VARIABLES
ALTCOR ERCODE* GS PMODE STRING* TOWPT

ARRAYS
CONBUF* ENTRY*

RECORD ARRAYS
RTE_MOD* WPT_MOD

FUNCTIONS AND SUBROUTINES
ADD_WPT DEMODE FIND_EMPTY HLD_IN KILL XLAT_RTE

MODULE NAME: INBOUND
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: DSP_WPTS
CALLING SEQUENCE: CALL INBOUND(WPT_INDEX, BRG_TEXT)

PURPOSE:
To generate bearing text for LEGS display.

DESCRIPTION:
This subroutine creates ASCII text for display in the inbound bearing field of the LEGS page. The created character string will have "TURN" for outbound DMA waypoints. Other type will have a number, up to three digits, with a degree symbol. Note that the bearing saved in the waypoint buffer for DMA inbound waypoints is perpendicular to the actual inbound bearing.

GLOBAL REFERENCES:

RECORD ARRAYS
WPT_MOD

FUNCTIONS AND SUBROUTINES
ANGL FSTRNG MTH$SIGN

MODULE NAME: INTC_END
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: DSP_WPTS
CALLING SEQUENCE: CALL INTC_END

PURPOSE:
To create CDU display labels for the LEGS page.

DESCRIPTION:
This procedure updates lines #11 and #12 of the CDU display screen when the DIR/INTC version of the LEGS page is shown. Two distinct formats are used for this page depending on the status of the global flag INTCF. This happens because the DIR/INTC page requires a user response after the initial DIR/INTC selection. The normal display shows the "direct to" and "intercept leg" prompts. When the intercept leg choice is selected the lines are updated with the intercept course prompt.

GLOBAL REFERENCES:

VARIABLES
INTCF

ARRAYS
BOXES DASHES

FUNCTIONS AND SUBROUTINES
FMTOUT

MODULE NAME: KILL_WPT
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: WPT_DATA
CALLING SEQUENCE: CALL KILL_WPT(WPT_INDEX)

PURPOSE:
To remove a waypoint from the flight plan.

DESCRIPTION:
This subroutine removes a waypoint from the flight plan by modifying the route buffer. When the waypoint is not part of a route function it is simply replaced by a route discontinuity marker. Otherwise the route function which contains the waypoint must be split into two pieces that contain the preceeding and following waypoints.

In the case of the route buffer element being a single waypoint a test is made on the following route buffer element. If it is a route function the deleted waypoint was its entry waypoint. The module NEXT_WPT is called to make the next route function waypoint in sequence the new entry waypoint. The same tests are made when the exit waypoint of a route function is deleted. The waypoint may have also served as the entry waypoint of a following route function in which case the NEXT_WPT call is required. When the exit waypoint is deleted the previous waypoint on the flight plan is used as the new exit, unless it has an undefined entry waypoint. A route function with an undefined entry generates a single waypoint in the waypoint buffer (the exit), so the exit deletion creates a null route function (undefined entry and exit). If the route function has been reduced to a one waypoint route function, having the same entry and exit points, the route function is deleted and the entry waypoint remains followed by a route discontinuity. All other cases of removing a route function waypoint are handled by the procedure SPLIT.

GLOBAL REFERENCES:

VARIABLES
ERCODE* MODCNT

ARRAYS
RTE_CNT

RECORD ARRAYS
RTE_MOD* WPT_MOD

FUNCTIONS AND SUBROUTINES
BOUNDS BREAK NEXT_WPT OPEN RTE_WPT SPLIT WPT_ADDR

MODULE NAME: LEGS
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: HLD_MGR, INTC_MGR, LEG_MGR
CALLING SEQUENCE: CALL LEGS(PAGE_ID)

PURPOSE:
To serve as the LEGS page executive module.

DESCRIPTION:
This procedure is called from the various LEGS format managers to handle function and data entries, and generate the data for the CDU LEGS page display. The one input parameter identifies the calling manager module, which is used to select the minor variations in the LEGS page format.

The first time the LEGS page is called after a change from another page format, some initialization is performed. The LEGS subpage is set to one unless returning from the LEGS-TIME page. In that case the subpage remains the same as it was on the LEGS-TIME page. Other LEGS variables are set to their default values.

A number of independent operations are performed in the body of the procedure. The following is a sequential list describing the functions.

. If the CDU clearance mode has changed to active, change from "FROM" format to standard.

. Call SET_PG to set up LEGS page parameters.

. Determine if the Plan Mode LEGS format is to be used. In this mode the navigation display format is centered at the waypoint marked with the "<CTR>" bug on the CDU. This format of the LEGS page is only shown when the NAV display is in Plan mode and the standard LEGS page is being used. Note that on the first pass of Plan mode the "<CTR>" bug is set to the last selected map center waypoint (see the module description for CDUFST).

. Respond to the following function entries.
  . Advance/Backup to next subpage.
  . Advance/Backup "<CTR>" bug (Plan mode only). Calls NEWCTR.
  . Change to ROUTE INDEX page (Standard LEGS only).
  . Hold at PPOS (HOLD page only). Calls HLD_POS.
  . Reject modified flight plan (standard LEGS only).
  . Echo waypoint name to scratch pad. Calls WPNAME.
  . Echo ALT/SPD constraints to scratch pad. Calls PROG_NUM.

. Respond to data entries by calling WPT_DATA.

. Update display lines by calling DSP_WPTS.

GLOBAL REFERENCES:

VARIABLES
  CTR* DISPST ERCODE* FROMPG* INTCF* LASTPG LATCEN LONCEN
  MODCNT PAGE* PASS* PGINIT* PGRQST* PLANM PMODE

ARRAYS
  ENTRY* OLDPAGE

RECORD ARRAYS
  WPT_MOD

FUNCTIONS AND SUBROUTINES
  DSP_WPTS HLD_POS NEWCTR PROG_NUM REJECT SET_PG WPNAME
  WPT_DATA

MODULE NAME: LEG_END
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: DSP_WPTS
CALLING SEQUENCE: CALL LEG_END

PURPOSE:
To create CDU display labels for the LEGS page.

DESCRIPTION:
This subroutine is called to update lines #11 and #12 of the CDU display screen when in the standard LEGS format. The reference time of arrival and RTA waypoint name are shown in the middle of lines #11 and #12 when defined. The prompts "<INDEX" and "ERASE>" are placed on the outside of line #12 to identify the use of LSK-L6 and LSK-R6. The erase prompt is only shown during the MOD CDU clearance mode.

GLOBAL REFERENCES:

VARIABLES
PMODE RTA_INDX RTA_TM

RECORD ARRAYS
WPT_MOD

FUNCTIONS AND SUBROUTINES
FMTOUT FMTTIM

MODULE NAME: LEG_MGR
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL LEG_MGR

PURPOSE:
To call the LEGS page module with standard format.

DESCRIPTION:
Since the standard legs format is used by several pages the main LEGS procedure must be called with a parameter indicating specific format. When the executive wishes to activate the standard legs format it calls the procedure LEG_MGR which in turn calls LEGS with a parameter value of "1".

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
LEGS

MODULE NAME: NEWCTR
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: LEGS
CALLING SEQUENCE: CALL NEWCTR(STEP)

PURPOSE:
To move the "<CTR>" bug on the LEGS page.

DESCRIPTION:
The "<CTR>" bug is moved STEP increments on the display. Note that STEP may be negative to "step back" or zero to force the page computation mentioned below. The bug will wrap around the ends of the flight plan. Also another STEP is performed when the new placement is on a route discontinuity. The navigation display format map center variables are set to the position of the new "<CTR>" waypoint. The last action is the computation of the CDU LEGS page which contains the "<CTR>" waypoint. This is performed because the bug may be STEPed off the current page.

GLOBAL REFERENCES:

VARIABLES
CTR FIRST_PTR GDTIME* LATCEN* LONCEN* MODCNT PAGE*

RECORD ARRAYS
WPT_MOD

MODULE NAME: NEW_ENTRY
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: NEXT_WPT, SPLIT, TRIM_WPTS
CALLING SEQUENCE: CALL NEW_ENTRY(RTE_PTR, WPT_ADR, RTE_OFF, EXIT_OFF)

PURPOSE:
To define a new route function entry waypoint.

DESCRIPTION:
This module sets up a new route function entry waypoint. The route buffer index for the new entry waypoint is passed as RTE_PTR. The waypoint's database address and route offset are also provided from the parameter list. The last paramter in the list is the offset of the exit waypoint.

A check is made to determine if the new route entry waypoint is the same as the route exit waypoint. If so, the route function is removed to leave the entry waypoint only. Any waypoint constraints (ALT/SPD/RAD) are extracted from the constraint buffer and assigned to the new waypoint.

GLOBAL REFERENCES:

RECORD ARRAYS
RTE_MOD*

FUNCTIONS AND SUBROUTINES
KILL TYPE_WPT XFER_CON

MODULE NAME: NEXT_WPT
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: KILL_WPT
CALLING SEQUENCE: CALL NEXT_WPT(INDEX)

PURPOSE:
To modify a route function when its entry waypoint is deleted.

DESCRIPTION:
When the entry waypoint of a route function is deleted a route discontinuity is inserted before the route function and a new entry waypoint is selected. The new entry is set up by a call to NEW_ENTRY.

When the route function does not have an exit waypoint defined, its definition only creates one waypoint, the entry, in the waypoint buffer. When the entry waypoint is deleted the route function is null, having neither an entry nor an exit waypoint.

GLOBAL REFERENCES:

RECORD ARRAYS
RTE_MOD

FUNCTIONS AND SUBROUTINES
BOUNDS BREAK GET_LONG KILL NEW_ENTRY OPEN

MODULE NAME: NMBRS
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: LEGS
CALLING SEQUENCE: CALL NMBRS(WPT_INDEX)

PURPOSE:
To decode constraint data entries for the CDU LEGS page.

DESCRIPTION:
This module is called to decode the numeric constraint value input for the waypoint indicated by the input parameter WPT_INDEX.

The data may be either speed, altitude, or turn radius information. Speed and altitude values may be entered on any display line containing a waypoint name. Turn radius values may only be assigned to waypoints not used in DMA turns. The five valid entry formats are shown below. The "nnn" depicts a one or more character numeric string.

nnn/nnn	Speed/Altitude entry
nnn	Altitude entry
nnn/	Speed entry
/nnn	Altitude entry
R/nnn	Turn radius entry

To delete the manually assigned speed and altitude entries at a waypoint use the LSKs to direct the DELETE text from the scratch pad to the chosen waypoint. Entering "R/" at a particular waypoint removes a manually entered turn radius.

Note that constraints may not be assigned to holding pattern waypoints.

GLOBAL REFERENCES:

VARIABLES
ERCODE* INDAT MODCNT

ARRAYS
ENTRY

RECORD ARRAYS
WPT_MOD

FUNCTIONS AND SUBROUTINES
ALTX DEL_IN FLTVAL KILL_CON LIB$MATCHC NEW_CON

MODULE NAME: PAD_NAME
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: ADD_WPT, DIRECT, HLD_IN
CALLING SEQUENCE: NAME = PAD_NAME()

PURPOSE:
To append blanks to the entered waypoint name.

DESCRIPTION:
PAD_NAME returns a five character ASCII string which is set to the name in the CDU entry line padded with blanks on the end. If the initial data is longer than five characters the returned string is "?????".

GLOBAL REFERENCES:

VARIABLES
ECHARS

ARRAYS
ENTRY

MODULE NAME: PROG_NUM
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: LEGS
CALLING SEQUENCE: CALL PROG_NUM

PURPOSE:
To echo altitude and speed values to the CDU scratch pad.

DESCRIPTION:
This procedure is called when the LSK adjacent to a waypoint's altitude and speed values is pressed. The values are echoed to the scratch pad as if manually entered, which allows their use elsewhere. PROG_NUM calls FMTOUT to perform the actual scratch pad update after the ASCII data is encoded from the waypoint buffer.

GLOBAL REFERENCES:

VARIABLES
ERCODE* FIRST_PTR MODCNT PAGE

ARRAYS
ENTRY

RECORD ARRAYS
WPT_MOD

FUNCTIONS AND SUBROUTINES
FMTOUT ISTRNG

MODULE NAME: SET_PG
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: LEGS, LEG_TIME
CALLING SEQUENCE: CALL SET_PG(FROM_FLAG)

PURPOSE:
To set LEGS page parameters.

DESCRIPTION:
The waypoint buffer index of the waypoint shown on the first position of LEGS page #1 is set. The decision depends on the current CDU clearance mode and the "From" variation status. The chart below shows the chosen index.

CLEARANCE MODE	VALUE (regular) "To" wpt	VALUE ("From") "From" wpt
Active	2	1
Modified	1	1
Original	1	1

The number of pages required to show all the waypoints is also computed by SET_PG.

GLOBAL REFERENCES:

VARIABLES
FIRST_PTR LASTPG* MODCNT PMODE TOWPT

MODULE NAME: SPLIT
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: ADD_WPT, HLD_IN, KILL_WPT
CALLING SEQUENCE: CALL SPLIT(INDEX,IN,OUT,OFFSET,STEP,FLG)

PURPOSE:
To break a route function into two pieces.

DESCRIPTION:
This procedure is called when operations are performed on waypoints within route functions defined in the route buffer. The existing route function must be split into two pieces at the selected waypoint.

The call list to SPLIT consists of six input parameters. The first is the index into the route buffer of the selected route function. The memory offsets to the entry and exit waypoint pointers are next. The fourth parameter is the memory offset to the "split" waypoint. The number of bytes between consecutive route function waypoints is provided through the fifth parameter. Note that the waypoint step value may be negative. The last parameter is a boolean variable used to request the deletion of the "split" waypoint.

A route function is made out of the first piece of the "split" by inserting a copy of the original route function in the previous route buffer slot. The exit waypoint of the new pieces is set to the waypoint one step behind the "split" waypoint. If the new route function has the same entry and exit points the copy is not created since the already defined entry waypoint is sufficient for the first piece of the split.

If the "split" waypoint is removed, a route discontinuity replaces the waypoint. In either case a new position in the route buffer is opened to hold the entry waypoint for the second part of the "split" route function. When the second part will contain only one waypoint the new entry waypoint is all that is needed. In this case the original route function is removed from the route buffer. The module NEW_ENTRY is called to set-up the second piece.

GLOBAL REFERENCES:

RECORD ARRAYS
RTE_MOD

FUNCTIONS AND SUBROUTINES
BREAK GET_LONG NEW_ENTRY OPEN

MODULE NAME: STEPS
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: DSP_WPTS
CALLING SEQUENCE: CALL STEPS

PURPOSE:
To create CDU display labels for the LEGS page.

DESCRIPTION:
This procedure updates CDU display lines #11 and #12 for the Plan mode LEGS page. Line #11 is completely dashed. The "step up"/ "step down" prompts are placed on line #12.

GLOBAL REFERENCES:

ARRAYS
DASHES

FUNCTIONS AND SUBROUTINES
FMTOUT

MODULE NAME: WPNAME
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: LEGS, LEG_TIME
CALLING SEQUENCE: CALL WPNAME

PURPOSE:
To echo a waypoint name to the scratch pad line.

DESCRIPTION:
This procedure performs the scratch pad programmimg of selected waypoint names. When one of the waypoints shown on the LEGS or LEGS-TIME pages is selected by pressing the adjacent line select key (LSK), this subroutine is called to enter the waypoint name into the CDU scratch pad for use as an entry elsewhere.

Error messages are signaled when a line with a route discontinuity or not containg a waypoint is selected.

GLOBAL REFERENCES:

VARIABLES
ERCODE* FIRST_PTR MODCNT PAGE

ARRAYS
ENTRY

RECORD ARRAYS
WPT_MOD

FUNCTIONS AND SUBROUTINES
FMTOUT

MODULE NAME: WPT_ADDR
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: ADD_PLAN, ADD_WPT, HLD_IN, KILL_CON, KILL_WPT, LINK_PD, NEW_CON, TRIM_WPTS
CALLING SEQUENCE: ADDRESS = WPT_ADDR(WPT_NAME)

PURPOSE:
To initiate a database search for a waypoint.

DESCRIPTION:
This procedure is called with the name of a waypoint from the waypoint buffer. The waypoint must not be a HOLD waypoint. The actual search is performed by calling the procedure WPT_ID.

GLOBAL REFERENCES:

VARIABLES
ERCODE* STRING*

ARRAYS
ENTRY*

FUNCTIONS AND SUBROUTINES
RET WPT_ID

MODULE NAME: WPT_DATA
FILE NAME: LEGS.FOR
PROCESS: SLOW
CALLED BY: LEGS
CALLING SEQUENCE: CALL WPT_DATA(PAGE_ID, FROM_FLG)

PURPOSE:
To parse CDU data entries for the LEGS page.

DESCRIPTION:
This procedure is called when a data entry is detected while on the LEGS page. There are two input parameters to the module. The first is an index indicating which version of the LEGS page is active (Standard, Hold, Dir/Intc). The second parameter signals when the "From Waypoint" format is being used. The following list describes the different valid data entries.

. A request to transfer to the ROUTE INDEX page. The data on the scratch pad was not intended for the LEGS page so it is reprogrammed back into the scratch pad for use by the ROUTE INDEX page. (Standard format only)

. Create a provisional holding pattern by calling HLD_IN. (Hold format only).

. Generate a "direct to" leg by calling DIRECT. (DIR/INTC format only).

. Generate a "bearing intercept" leg by calling INTERCEPT or DIRECT depending on status of entries. (DIR/INTC format only).

. Parse constraint entries by calling NMBRS.

. Delete flight plan waypoint by calling KILL_WPT.

. Insert flight plan waypoint by calling ADD_WPT.

After any flight plan modifications which did not set an error condition, the module DEMODE is called to generate the new "MOD" waypoint buffer.

GLOBAL REFERENCES:

VARIABLES
ERCODE* FIRST_PTR INTCF* PAGE PGRQST*

ARRAYS
ENTRY

FUNCTIONS AND SUBROUTINES
ADD_WPT DEL_IN DEMODE DIRECT HLD_IN INTERCEPT KILL_WPT
NMBRS REPROG

Section 6.3.5 THE LEGS TIME PAGE

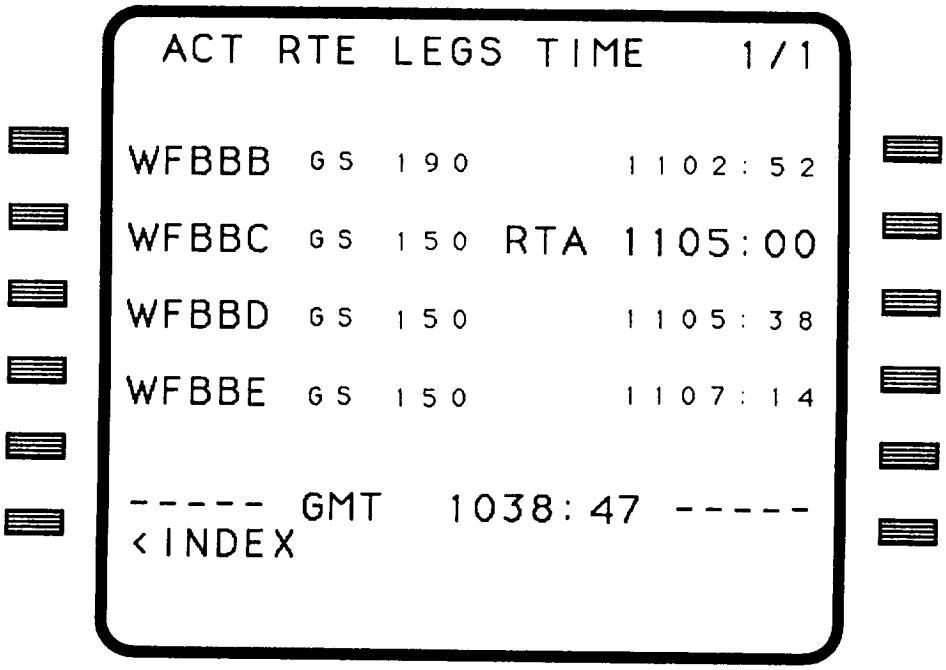
This page is used to select the Reference Time of Arrival (RTA) waypoint. The page is accessed through the ROUTE INDEX page. The flight plan waypoints appear on the left side of the display pages followed by the defined ground speed constraint and the assigned arrival time. When a RTA waypoint has not been selected the arrival time fields contain dashes. To designate a RTA waypoint a time is keyed on the scratch pad line and entered at the desired waypoint with one of the LSKs on the right hand side of the CDU display. The format for the time entry is "HHMM.SS". The ".SS" field is optional. The line containing the RTA waypoint has the "RTA" symbol placed on its line.

Note that the current time of day is always displayed on the bottom of the page for reference. Refer to figure 6.7 for the format of the LEGS TIME page.

The remainder of this section provides the descriptions of the four LEGS TIME modules which reside on the file LEG_TIME.FOR.

PAGE 184 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED



The Legs Time Page

(figure 6.7)

PAGE 186 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: DSP_TIME
FILE NAME: LEG_TIME.FOR
PROCESS: SLOW
CALLED BY: LEG_TIME
CALLING SEQUENCE: CALL DSP_TIME

PURPOSE:
To create data for the LEGS TIME CDU screen.

DESCRIPTION:
This subroutine causes the CDU display to show data pertinent to the LEGS TIME format. The screen is completely refreshed every six calls to this module. On the first call of the cycle the title line and prompt text are output. On calls #2 through #6 the five lines that show waypoint information are updated.

Data for the following items is created on the first call of the cycle. The data is moved to the CDU display buffer via calls to FMTOUT.

- Call TITLE to generate the title line.
- Encode the current time of day by calling FMTTIM.
- Place the "<INDEX" prompt to the left of line #12.
- When a modified flight plan exists place the "ERASE>" prompt on the right of line #12.

The module SET_PG defines which waypoint will appear at the top of page #1. The remaining flight plan waypoints are placed sequentially on display lines, five per page. Enough LEGS TIME pages are maintained to account for all the waypoints. The following three items may be placed on the waypoint lines of the display page.

- A blank line for slots past the last defined waypoint.
- A "RTE DSC" symbol in reverse video for route discontinuities found in the waypoint buffer.
- The waypoint name, ground speed constraint, and assigned arrival time.

GLOBAL REFERENCES:

VARIABLES
FIRST_PTR LASTPG MODCNT PAGE PASS* PMODE RTA_INDX TIME

ARRAYS
DASHES

RECORD ARRAYS
WPT_MOD

FUNCTIONS AND SUBROUTINES
FMTOUT FMTTIM FSTRNG TITLE

PAGE 188 INTENTIONALLY BLANK
PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: ECHO_TIME
FILE NAME: LEG_TIME.FOR
PROCESS: SLOW
CALLED BY: LEG_TIME
CALLING SEQUENCE: CALL ECHO_TIME

PURPOSE:
To echo selected arrival times to the CDU scratch pad.

DESCRIPTION:
This procedure is called when the arrival time at a particular waypoint is selected for insertion into the scratch pad line. The time value is encoded in place as if manually enter from the keyboard. An error code is signaled when the LSK adjacent to a route discontinuity is selected.

GLOBAL REFERENCES:

VARIABLES
ERCODE* FIRST_PTR MODCNT PAGE RTA_WPT

ARRAYS
ENTRY

RECORD ARRAYS
WPT_MOD

FUNCTIONS AND SUBROUTINES
FMTOUT FMTTIM

MODULE NAME: LEG_TIME
FILE NAME: LEG_TIME.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL LEG_TIME

PURPOSE:
To serve as the LEGS TIME page executive.

DESCRIPTION:
This procedure is the main routine for the LEGS TIME page of the CDU. When CDU keyboard entries are made either an inline action is made or the appropriate handler is called. After checking inputs the CDU screen refresh module is called.

The first time LEG_TIME is called, upon transfer from a different CDU page format, some initialization occurs. The LEGS TIME subpage is set to one, unless transfering from the LEGS page. The same subpage is used as was on the LEGS page to provide agreement between the waypoints seen when transfering between the pages.

Page and subpage change requests are handled inline by LEG_TIME. Other entries are handled by special procedures. The following list describes the types of entries and the called procedure.

- Reject modified flight plan. REJECT
- Echo waypoint name to scratch pad. WPNAME
- Echo arrival time to the scratch pad. ECHO_TIME
- Decode and process arrival time entries. TIME_IN

GLOBAL REFERENCES:

VARIABLES
ERCODE* LASTPG PAGE* PASS* PGINIT* PGRQST* PMODE

ARRAYS
ENTRY* OLDPAGE

FUNCTIONS AND SUBROUTINES
DSP_TIME ECHO_TIME REJECT SET_PG TIME_IN WPNAME

MODULE NAME: TIME_IN
FILE NAME: LEG_TIME.FOR
PROCESS: SLOW
CALLED BY: LEG_TIME
CALLING SEQUENCE: CALL TIME_IN

PURPOSE:
To decode and process arrival time entries.

DESCRIPTION:
This procedure handles data entries on the LEGS TIME page of the CDU. The normal data entry consists of an arrival time entered at a waypoint using one of the upper five LSKs on the right hand side of the display screen. "DELETE" may also be entered adjacent to the RTA waypoint to remove all arrival times from the flight plan. The only other valid data entries are actually function entries that were made when data happened to be on the scratch pad line (the two page change commands). When this occurs the data is reprogrammed to the scratch pad for use by subsequent CDU pages.

Note that when an entered time is more than a half day earlier than the current time of day, the entered value is assumed to fall into the following day.

GLOBAL REFERENCES:

VARIABLES
ERCODE* FIRST_PTR LASTPG* MODCNT PAGE PGRQST* RTA_INDX*
RTA_TM* RTA_WPT* TIME

ARRAYS
ENTRY

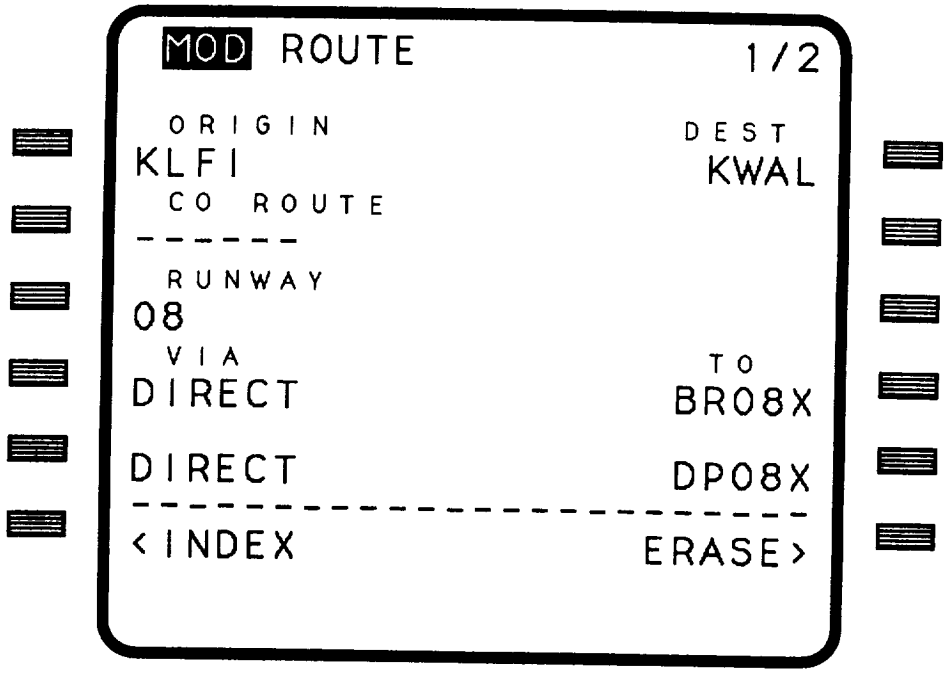
RECORD ARRAYS
WPT_MOD

FUNCTIONS AND SUBROUTINES
DEL_IN DEMODE REPROG TIMVAL

Section 6.3.6 THE ROUTE PAGE

The ROUTE page is used for the creation and modification of aircraft flight plans. An origin and destination airfield must be chosen before any flight plan information is entered. Page #1 of the ROUTE page is used to choose airfields. The takeoff runway and company route may optionally be selected on page #1 also. The remainder of page #1 and all following pages contain the route function and waypoint names comprising the flight plan. The various route elements may be entered and deleted from the ROUTE page, however no waypoint constraint data may be entered. As many route pages as needed to hold all the desired route elements will automatically be maintained. The current page and last page are always displayed on the title line in the form "<current>/<last>".

The remaining pages of this section contain pictures of a typical route page and are followed by descriptions of the 34 modules contained in the file ROUTE.FOR.

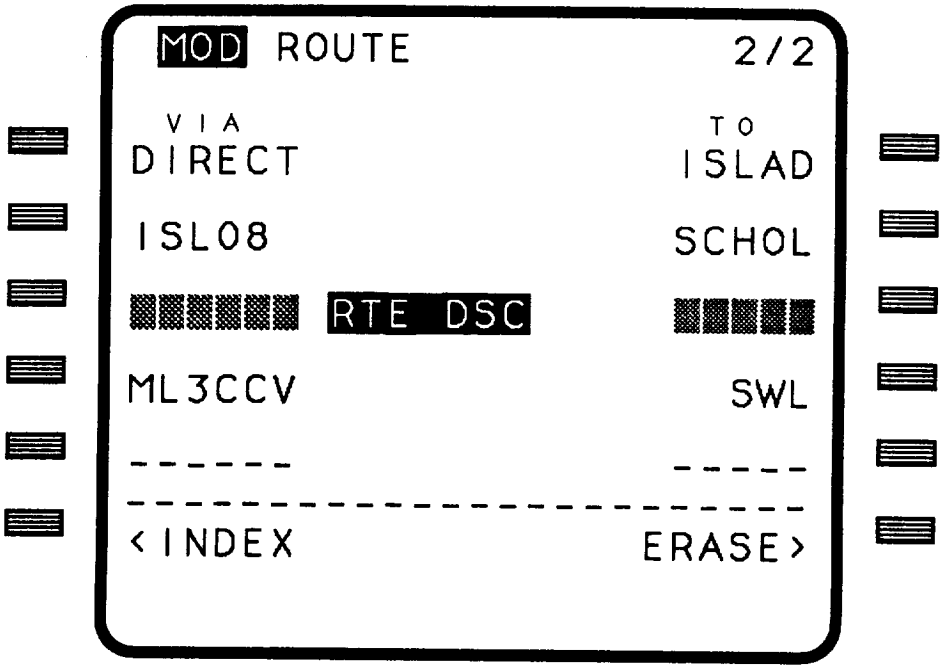


The Route Page

(figure 6.8)

PAGE 194 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED



The Route Page

(figure 6.9)

PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: ACT_EXIT
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: DATA_IN
CALLING SEQUENCE: CALL ACT_EXIT(INDEX)

PURPOSE:
To verify "dead waypoint" errors.

DESCRIPTION:
Dead waypoint errors occur when the pilot attempts to delete overflown waypoints from the ROUTE page. The module signals a dead waypoint situation when a DELETE is placed at one of the route elements. However a dead waypoint situation is flagged when the exit waypoint of a route function that contains the active "To" waypoint is deleted. This procedure checks for that special case and allows the exit waypoint deletion instead of setting the error code.

GLOBAL REFERENCES:

VARIABLES
ACTCNT ERCODE* TOWPT

RECORD ARRAYS
RTE_ACT RTE_MOD* WPT_ACT

MODULE NAME: AIRPORT
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: DATA_IN, WPT_ID
CALLING SEQUENCE: AIRPORT(COUNT, ADDRESS)

PURPOSE:
To search the database for the entered airfield.

DESCRIPTION:
The airfield name in the CDU entry buffer is used to search the navigation database (AADCOM). The address is returned when found, otherwise a zero is returned.
Two different error code values can be used when no airfield is found. When the entered text is not the proper format for an airfield the "BAD DATA FORMAT" code is returned. When not found in the database the "NOT FOUND IN MEMORY" is used. Proper airfield format is a four character name starting with the letter "K".

GLOBAL REFERENCES:

VARIABLES
ERCODE*

ARRAYS
ENTRY

FUNCTIONS AND SUBROUTINES
LUARP

MODULE NAME: BREAK
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: DATA_IN, DEL_RTE, DIRECT, GROUP, INTC_WPTS, KILL_WPT, NEXT_WPT, SPLIT
CALLING SEQUENCE: CALL BREAK(INDEX)

PURPOSE:
To create a route dicontinuity.

DESCRIPTION:
The route buffer element at route buffer location "INDEX" is made a route discontinuity. This is done by clearing the TYPE, ADDR, CPTR, and EXIT modes of that buffer location.

GLOBAL REFERENCES:

RECORD ARRAYS
RTE_MOD*

MODULE NAME: CLEAN_PPT
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: MAKE_WPT
CALLING SEQUENCE: CALL CLEAN_PPT

PURPOSE:
To search for free pilot waypoint buffer locations.

DESCRIPTION:
This procedure is called when all twenty of the pilot waypoint buffer (PPT_WPT) positions are defined. A search is made for the use of each definition in the provisional and active route buffers. Those definitions no longer used are marked as available. If no positions are found an error code is returned.

GLOBAL REFERENCES:

VARIABLES
ERCODE*

ARRAYS
RTE_CNT

RECORD ARRAYS
PPT_WPT RTE_ACT RTE_MOD

FUNCTIONS AND SUBROUTINES
GET_CHAR

MODULE NAME: COMPANY
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: DATA_IN
CALLING SEQUENCE: CALL COMPANY(NAME_LENGTH)

PURPOSE:
To insert a company route into the route buffer.

DESCRIPTION:
This procedure is called when a company route is entered on the ROUTE page of the CDU. The length parameter passed is the number of characters in the global buffer, ENTRY, which contains the company route name.

Once the company route is found in the navigation database the default origin and destination airfields are set. Runway data for these airfields is invalidated. When an origin airfield has already been entered on the CDU the default value must match or the company route entry is rejected. The company route SID and STAR pointers are tested for non-zero values. If supplied, the departure and arrival route functions are placed in the route buffer by calling GROUP. The company route's list of waypoints are inserted into the route buffer after the SID and before the STAR.

GLOBAL REFERENCES:

VARIABLES
ADDRESS ERCODE*

ARRAYS
AIRPTS* ENTRY RTE_CNT

RECORD ARRAYS
RTE_MOD*

FUNCTIONS AND SUBROUTINES
GET_LONG GROUP LURTE TYPE_WPT

MODULE NAME: DATA_IN
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: ROUTE
CALLING SEQUENCE: CALL DATA_IN

PURPOSE:
To parse route page keyboard data entries.

DESCRIPTION:
This subroutine is called to parse keyboard data entries. Function entries are handled by the main procedure ROUTE or the module ECHO. These data entries are the names of route functions, waypoints, and runways. If route page #1 is being shown there are four entries which are made at fixed positions on the screen. This includes the origin and destination airfields, origin runway, and company route. The adjacent line select keys (LSKs) are used to modify these items. All other entries are waypoints, route functions, or 'DELETE' entries which are enterd at various display lines via the LSKs.

First, DATA_IN checks for fixed position inputs. These are the origin/destination airfields, company route, and takeoff runway. When a valid origin airfield is entered, the flight plan is initialized to be empty. When the destination airfield is entered any previously existing touchdown waypoints are removed from the plan. When a company route is entered the entire flight plan is setup by a call to the module COMPANY. Valid takeoff runways elicit a call to ORG_RWY to generate the required runway waypoints.

The #6 LSK on the left hand side (LSK-L6) is used to change to the ROUTE INDEX page. Data on the scratch pad when this LSK is selected is not intended as a data entry. DATA_IN detects this situation and responds by calling REPROG to restore the data to the scratch pad, and then signals the page change request.

The processing of waypoints, route functions, and DELETE entries occurs next. Note that all route functions are always entered with left LSKs and waypoints with right LSKs. DELETE entries are made on either side. The LSK selected and the current ROUTE page number are used to identify a particular element in the route buffer. The element may already contain an entry or may be the next available spot at the end of the buffer. Note that route buffer elements corresponding to waypoints prior to the current "To" waypoint on an active flight plan may not be modified. Also the position imediately following a route function with an undefined exit waypoint may not be changed.

A left LSK is used when a route function or DELETE is entered. The DELETE entry will erase the route function and replace it with a route discontinuity marker, unless the deleted item was the last in the buffer. When a route function is entered, RTE_ID is called to search the system database for the name and to identify its type (SID, STAR, ...). Two types of route function entries are possible. If an individual route function name was entered, RTE_ID set OFFSET to zero, the route function is made part of the flight plan by calling GROUP. An intercept route entry of the form BEARING/ROUTE_NAME/EXIT_WAYPOINT is also processed by RTE_ID. In this situation the offset of the exit waypoint is returned to this procedure in the global variable OFFSET. The procedure RTE_INTC is called instead of GROUP for the insertion of the intercept waypoint and the route function waypoints.

A right LSK is used when a waypoint or DELETE entry is made. DELETE can either erase the exit waypoint of a route function or a normal waypoint. Route discontinuity markers are inserted for DELETE entries unless the erased item was the last in the buffer. When a waypoint name is entered, WPT_ID is called to search the system database and identify the waypoint type (GRP, NAVAID, ...). If found the waypoint information is inserted into the route buffer by calling WAYPOINT.

GLOBAL REFERENCES:

VARIABLES
ERCODE* PAGE PGRQST* PMODE PRMT TOWPT

ARRAYS
AIRPTS ENTRY RTE_CNT*

RECORD ARRAYS
RTE_MOD WPT_ACT

FUNCTIONS AND SUBROUTINES
ACT_EXIT AIRPORT BREAK COMPANY DEL_IN DEL_RTE EXIT GROUP
INIT_PLAN LURWY OPEN ORG_RWY REMOVE REPROG RTE_ID RTE_INTC
WAYPOINT WPT_ID

MODULE NAME: DEL_IN
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: ACTION, DATA_IN, DA_INPUT, FIX_INFO, FIX_INP, FLT_TYPE_INP, HOLD_INPUT, IDENT, INDX_INPUT, INITUP, NAV_INPUT, NMBRS, PFINP, SUBNAV_INPUT, TIME_IN, TKOFFINP, WPT_DATA
CALLING SEQUENCE: BOOLEAN=DEL_IN()

PURPOSE:
To identify DELETE entries.

DESCRIPTION:
The CDU entry line buffer ENTRY is tested for "DELETE".

GLOBAL REFERENCES:

VARIABLES
STRING

ARRAYS
ENTRY

MODULE NAME: DEL_RTE
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: DATA_IN, MODIFY
CALLING SEQUENCE: CALL DEL_RTE(INDEX)

PURPOSE:
To remove a route function from the route buffer.

DESCRIPTION:
This procedure is called with a route buffer index designating a route function which is to be removed from the route buffer. If it is the last route element in the buffer it is simply removed. Otherwise a route discontinuity is created in its place.

Note that the module XLAT_RTE handles the clearing of the destination runway address save in AIRPTS. Therefore this module does not have to determine if the deleted route function was an approach.

GLOBAL REFERENCES:

ARRAYS
RTE_CNT*

FUNCTIONS AND SUBROUTINES
BREAK

MODULE NAME: DSC_CHECK
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: DEMODE
CALLING SEQUENCE: CALL DSC_CHECK

PURPOSE:
To perform maintainance on the route buffer.

DESCRIPTION:
Each time changes are made to the route buffer DSC_CHECK is called to determine if the route buffer manipulations have produced route discontinuities which either appear at the end of the buffer or immediately following another discontinuity. Also checks are made to determine if the same waypoint is placed in consecutive buffer locations. When any of these situations occur, the excess elements are removed from the buffer.

GLOBAL REFERENCES:

ARRAYS
RTE_CNT*

RECORD ARRAYS
RTE_MOD

FUNCTIONS AND SUBROUTINES
GET_LONG KILL

MODULE NAME: ECHO
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: ROUTE
CALLING SEQUENCE: CALL ECHO

PURPOSE:
To echo route element names to the scratch pad.

DESCRIPTION:
LSK's are used on the ROUTE page to select particular route elements for use in the scratch pad. In this form the LSK is used as a function entry since no scratch pad data is entered prior to selection. The route function name is programmed into the scratch pad line as if the individual characters were manually entered.

The PROG_SCR routine is called to echo the names of origin or destination airfields, direct waypoints, exit waypoints, and route functions to the scratch pad. The type byte stored in the route buffer is used to determine how many characters to echo for the various kinds of waypoints.

GLOBAL REFERENCES:

VARIABLES
ERCODE* PAGE

ARRAYS
AIRPTS ENTRY RTE_CNT

RECORD ARRAYS
RTE_MOD

FUNCTIONS AND SUBROUTINES
GET_CHAR GET_LONG PROG_SCR TYPE_WPT

MODULE NAME: ENTRY_WPT
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: BOUNDS, GROUP, SEQUENCE
CALLING SEQUENCE: OFFSET = ENTRY_WPT(INDEX,ADDR,TYPE)

PURPOSE:
To determine validity of a route buffer entry waypoint.

DESCRIPTION:
This function is used to determine if a waypoint stored in the route buffer can be used as the entry waypoint of a route function. The offset from the start of the route function to the waypoint pointer is returned when the waypoint is valid, otherwise zero is returned. See section 1.5.1.1 for detatils of the database format for route function data.

There are three input parameters to ENTRY_WPT. The first is the index into the route buffer of the element to be tested as an entry waypoint. Note that the selected route buffer element may be either a waypoint or a route function. If it is a route function the exit waypoint is used for the validity test. When this situation occurs, two route functions share the same waypoint as a joint exit/entry. The remaining parameters are the address and type of the route function stored in the navigation database.

GLOBAL REFERENCES:

RECORD ARRAYS
RTE_MOD

FUNCTIONS AND SUBROUTINES
GET_LONG RTE_WPT

MODULE NAME: EXIT
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: DATA_IN, EXECUTE, ROUTE, UPDATE, WAYPOINT
CALLING SEQUENCE: BOOLEAN = EXIT(INDEX)

PURPOSE:
To check exit waypoint status.

DESCRIPTION:
The route element defined in the route buffer, location INDEX, is tested to determine if it is a route function with an undefined exit waypoint. If it is the function returns the boolean value FALSE. Otherwise the return value is TRUE.

GLOBAL REFERENCES:

RECORD ARRAYS
RTE_MOD

MODULE NAME: EXIT_WPT
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: GROUP
CALLING SEQUENCE: OFFSET = EXIT_WPT(INDEX,ADDR,TYPE)

PURPOSE:
To determine validity of a route buffer exit waypoint.

DESCRIPTION:
This function is used to determine if a waypoint stored in the route buffer can be used as the exit waypoint of a route function. The offset from the start of the route function to the waypoint pointer is returned when the waypoint is valid, otherwise zero is returned. See section 1.5.1.1 for detatils of the database format for route function data.

There are three input parameters to ENTRY_WPT. The first is the index into the route buffer of the element to be tested as an entry waypoint. If this index points to an unused buffer location or the element at the location is a route function, the invalid status is immediately returned. The remaining parameters are the address and type of the route function stored in the navigation database. The input data is passed on to the function RTE_WPT to check for validity as a route waypoint.

GLOBAL REFERENCES:

ARRAYS
RTE_CNT

RECORD ARRAYS
RTE_MOD

FUNCTIONS AND SUBROUTINES
RTE_WPT

MODULE NAME: FIND_PPT
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: PROCESS_GRP, UPDATE_POS, WPT_ID
CALLING SEQUENCE: INDEX = FIND_PPT(NAME)

PURPOSE:
To locate a specific pilot defined waypoint.

DESCRIPTION:
This function is called with the five character name of a pilot waypoint. The pilot waypoint buffer (PPT_WPT) is searched and the index into the structure is returned. A zero index value is returned when not found.

GLOBAL REFERENCES:

RECORD ARRAYS
PPT_WPT

MODULE NAME: FIND_RTE
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: RTE_ID
CALLING SEQUENCE: ER_CODE = FIND_RTE(NAME,CNT,ADDR,TYPE)

PURPOSE:
To locate a route function in the navigation database.

DESCRIPTION:
This function searches the system database for the route function whose name and length are supplied as the first two parameters in the calling sequence. The address and route function type are returned through the parameter list. The value returned through the function reference is a CDU error code. If the value is zero, no errors were detected.

An airway name consists of two to six letters, starting with "J" or "V". SIDs, STARs, and APPROACHES have five or six characters with no special starting letter.

Non-airway route functions are associated with particular airfields. The module LUSID is called to search AADCOM for any SID/STAR/APPROACH at the destination airfield first. Searching is complete if the item was found in the database and the type of the located route function is not a SID, since departures from the destination airfield may not be referenced. Searching continues using the origin airfield as reference. This time the only route function type considered a valid find is the SID.

GLOBAL REFERENCES:

ARRAYS
AIRPTS

FUNCTIONS AND SUBROUTINES
LUJET LUSID LUVIC

MODULE NAME: GROUP
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: COMPANY, DATA_IN, MODIFY, RT_NEW
CALLING SEQUENCE: CALL GROUP(ADDR, TYPE, INDEX, APPEND)

PURPOSE:
To insert a route function into the route buffer.

DESCRIPTION:
This procedure enters a group of waypoints (route function) into the route buffer. Processing varies for the different route function types. In append mode the new route function is simply added to the end of the route buffer. Otherwise an insertion, possibly with the creation of route discontinuities, into the buffer is performed. When insertion occurs the following route elements, including the one at the selected position, are moved ahead in the route buffer.

The parameter list in the call consists of four input parameters. The first is the database address of the entered route function. Next is the route function type code (defined in CDU.INC). The INDEX parameter points to the destination slot in the route buffer for the entered route function. The last parameter is a flag indicating when the append mode is active.

The previous route buffer element may be used as an entry waypoint into the entered route function. The function ENTRY_WPT is used to set the variable ENTRY_OFS to the offset within the route function data to the entry waypoint. If the previous element cannot be used as an entry waypoint, ENTRY_OFS is zeroed. The following chart describes how the different route function types are handled by GROUP.

APPROACHES: When an entry waypoint does not already exist in the route buffer, the first waypoint on the approach is used as a default. The last waypoint of the approach is always used as the exit waypoint, which automatically terminates the flight plan as the touchdown runway. No route discontinuities are generated from approach entries.

SID/STAR: The entry waypoint defaults to the first of the route function unless a valid entry waypoint was found in the route buffer. The following route buffer element is checked for validity as an exit waypoint for the entered route function. If it cannot be used the default exit waypoint is the last waypoint on the route function. No route discontinuities are generated from SID/STAR entries.

AIRWAYS: An entry waypoint must have been explicitly defined as the previous route element or the route function entry is rejected. When an airway is inserted (not appended) into the route buffer a route discontinuity marker is always inserted immediately following the entered route function. No default exit waypoint is used and the next route buffer element is not considered as a possible exit waypoint.

GLOBAL REFERENCES:

VARIABLES
ERCODE*

ARRAYS
AIRPTS* RTE_CNT*

RECORD ARRAYS
RTE_MOD*

FUNCTIONS AND SUBROUTINES
BREAK ENTRY_WPT EXIT_WPT GET_LONG KILL OPEN REMOVE TYPE_WPT

MODULE NAME: INIT_PLAN
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: DATA_IN
CALLING SEQUENCE: CALL INIT_PLAN

PURPOSE:
To initialize the flight plan.

DESCRIPTION:
Each time the origin airfield entry is made on the ROUTE page of the CDU the flight plan is initialized. A number of memory locations are cleared or invalidated as follows.

. Company route address is invalidated.
. CDU mode is set to initial clearance.
. 2D/3D/4D guidance modes are invalidated.
. MOD & ACT route buffer counts are zeroed.
. MOD & ACT hold waypoints are invalidated.
. Phase of flight booleans are reset.
. Cruise altitude is invalidated.
. Origin/destination airfield info is invalidated.
. The pilot waypoint buffer is cleared.
. The waypoint constraint buffer is cleared.

GLOBAL REFERENCES:

VARIABLES
CLBCHNG* CRZALT* CRZCHNG* C_ADR* DESCHNG* DESCHNG1*
GUID2D* GUID3D* GUID4D* HLD_PTR* HLD_WPT* PMODE*

ARRAYS
AIRPTS* CONBUF RTE_CNT*

RECORD ARRAYS
PPT_WPT

FUNCTIONS AND SUBROUTINES
CLRBUF

MODULE NAME: INTC_WPTS
FILE NAME: RTE_INTC.FOR
PROCESS: SLOW
CALLED BY: RTE_INTC
CALLING SEQUENCE: CALL INTC_WPTS(LAT,LON,ENTRY,RTE_ADR,
RTE_TYP,BUF_PTR,APPEND)

PURPOSE:
To store route intercept data into the route buffer.

DESCRIPTION:
This module makes the insertions into the provisional route buffer (RTE_MOD) which reflect the route intercept data generated when the "RTE INTC" option is selected on the Route page. Four elements are inserted into the route buffer.

1) A waypoint for the leg intercept point
2) The entry waypoint onto the route
3) The route function
4) A route discontinuity separating the new data from the old

The first two items passed through the calling sequence are the latitude and longitude of the intercept waypoint. The next item passed is the offset into the route function of the route entry waypoint followed by the address and type of the route function. Next is the index into the route buffer where the insertion is to be made. The last parameter is a boolean flag indicating whether the additions are either inserted within the route buffer or appended onto the end.

The intercept waypoint is created in the Pilot Defined Waypoint buffer with a call to MAKE_WPT. The route function entry waypoint is saved next, using the passed offset to fetch the address pointer to the actual waypoint. The function TYPE_WPT is used to determine what type of waypoint the entry waypoint is. The route function is stored by using the passed address and type. The global variable OFFSET already contains the required exit waypoint offset, which was found when the route intercept entry was initially parsed. In the case of the Approach type of route function, the global exit offset is ignored and the last waypoint of the Approach is used as the exit. Also for an Approach the destination runway is set and any route buffer elements past the Approach are removed. The fourth addition to the route buffer is a route discontinuity marker.

```
GLOBAL REFERENCES:

VARIABLES
  OFFSET

ARRAYS
  AIRPTS* RTE_CNT*

RECORD ARRAYS
  RTE_MOD*

FUNCTIONS AND SUBROUTINES
  BREAK GET_LONG KILL MAKE_WPT OPEN TYPE_WPT
```

MODULE NAME: INT_LEG
FILE NAME: RTE_INTC.FOR
PROCESS: SLOW
CALLED BY: RTE_INTC
CALLING SEQUENCE: DIST = INT_LEG(X1, Y1, X2, Y2, PBRG)

PURPOSE:
To compute the distance to route function intercept points.

DESCRIPTION:
This function returns the distance to the intersection point of a line through the coordinate system origin, extended at an angle "PBRG", and the line bounded by the coordinate endpoints (X1,Y1;X2,Y2). If no point is found, a "-1" is returned. The coordinate endpoints and line bearing are passed in the parameter list. Note that both lines are treated as infinte length lines extented at their defined slopes. Once the intersection of the infinite lines is found, checking is performed to determine if the intersection falls within the bounds of both lines. Note that the line defined by origin and bearing is an infinite line in the direction of the passed bearing only.

An intersection point exists somewhere on the infinite lines, as long as they do not have the same slope. The intersection coordinates are found in one of three ways; the line through the origin is vertical (infinite slope), the bounded line is vertical, or neither is vertical. The methods used for computing the X and Y intersection points are shown below.

```
SLOPE = (Y2 - Y1) / (X2 - X1)  [slope of bounded line]
Y_INT = Y1 - X1 * SLOPE  ["Y" intercept of bounded line]
TANBRG = TAN(90 - PBRG)  [adjusted to cartesian angular measure]

Origin line vertical:
  X = 0;  Y = Y_INT

Bounded line vertical:
  X = X1; Y = X1 * TANBRG

Neither vertical:
  X = Y_INT / (TANBRG - SLOPE)
  Y = TANBRG * X
```

Once the intersection point is found, it is determined whether the intersection point falls within the bounds of the actual lines. The function BETWEEN checks if the point is on the bounded line segment. Since the bearing through the origin line does not actually extend in the reverse direction, the angle to the intersect coordinates is computed and compared to the original bearing to see that they match (are not 180 degrees out of phase).

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
  ANGL MTH$ATAND2 MTH$SQRT MTH$TAND

MODULE NAME: KILL
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: BEG_RTE, DSC_CHECK, GROUP, HLD_POS, HOLD_INPUT, INTC_WPTS, LINK_PD, MERGE, MODIFY, NEW_ENTRY, NEXT_WPT, REMOVE, TRIM_WPTS
CALLING SEQUENCE: CALL KILL(INDEX, COUNT)

PURPOSE:
To remove route elements from the route buffer.

DESCRIPTION:
"COUNT" route elements are removed from the route buffer starting at the location INDEX. The created gap is removed by shifting trailing elements back over the deleted ones.

GLOBAL REFERENCES:

ARRAYS
RTE_CNT*

RECORD ARRAYS
RTE_MOD*

MODULE NAME: MAKE_WPT
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: DIRECT, FILL_RTE, INTC_WPTS, INTERCEPT, MODIFY, ORG_RWY, WAYPOINT, WPT_ID
CALLING SEQUENCE: ADDR=MAKE_WPT(LAT,LON,ALT,SPD,NAME)

PURPOSE:
To create pilot defined waypoints.

DESCRIPTION:
This function is used to create waypoints in the pilot waypoint buffer (PPT_WPT). The address of the created waypoint is returned to the caller.

The parameter list defines the characteristics of the created waypoint. The position is passed as the first two parameters (latitude & longitude). Altitude and speed are optional parameters which become default waypoint constraints. The name parameter may be either a full five character name or just the three character name prefix. When the prefix is supplied a unique two digit sequence number is appended to the name. The CDU entry line which prompted the creation of the pilot waypoint is saved in PPT_WPT also for reference by the NAV DATA page.

When the pilot waypoint buffer is full, CLEAN_PPT is called to remove waypoint definitions no longer used. If no space is found an error code is returned.

GLOBAL REFERENCES:

VARIABLES
ERCODE STRING

ARRAYS
ENTRY

RECORD ARRAYS
PPT_WPT

FUNCTIONS AND SUBROUTINES
CLEAN_PPT ISTRNG P_LIST

MODULE NAME: MERGE
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: WAYPOINT
CALLING SEQUENCE: CALL MERGE(WPT_ADR, INDEX, LAST_INDEX)

PURPOSE:
To merge sections of the route buffer.

DESCRIPTION:
This subroutine is called to determine if the last entered waypoint appears again further along on the flight plan. This is interpreted as an attempt, by the pilot, to eliminate the part of the flight plan in between. When this situation is detected, the route buffer data in between the two occurences (including one copy of the repeat waypoint) is removed from the route buffer.

The address of the entered waypoint is passed as the first call parameter. Its route buffer postion is the second parameter and the third is the route buffer position where searching should be terminated.

When the merge is performed the second copy of the waypoint is thought to be moved up to a previous position, removing elements in between. Therefore any waypoint constraints associated with the original waypoint are transfered to the new copy by calling XFER_CON.

Complications arise when the route buffer elements examined are route functions. Then the waypoints defined between, and including, the exit and entry waypoints are compared to the key waypoint. In this situation the key waypoint becomes the new entry waypoint for the tested route function.

GLOBAL REFERENCES:

RECORD ARRAYS
RTE_MOD

FUNCTIONS AND SUBROUTINES
BOUNDS KILL RTE_WPT SEQUENCE XFER_CON

MODULE NAME: NEW_POS
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: INTERCEPT, ORG_RWY, WAYPOINT, WPT_ID
CALLING SEQUENCE: CALL NEW_POS(LAT1,LON1,BRG,RNG,LAT2,LON2)

PURPOSE:
To compute a position from a defined reference position.

DESCRIPTION:
This procedure uses a reference position (LAT1/LON1), bearing (BRG), and range (RNG) from the parameter list to compute a new position which is returned through the parameter list (LAT2/LON2). The Path Definition procedure, LOCAL_ERAD, is called to find the local feet per degree values.

GLOBAL REFERENCES:

VARIABLES
LAT_FEET LON_FEET

FUNCTIONS AND SUBROUTINES
LOCAL_ERAD SCOSD

MODULE NAME: OPEN
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: BEG_RTE, DATA_IN, DIRECT, GROUP, HLD_IN, INTC_WPTS, KILL_WPT, NEXT_WPT, ORG_RWY, SPLIT, WAYPOINT
CALLING SEQUENCE: CALL OPEN(INDEX, SPACES)

PURPOSE:
To create open spaces in the route buffer.

DESCRIPTION:
A number of free positions, indicated by the SPACES parameter, are open at the route buffer location INDEX. The subsequent route elements are moved further along in the route buffer.

GLOBAL REFERENCES:

VARIABLES
ERCODE*

ARRAYS
RTE_CNT*

RECORD ARRAYS
RTE_MOD*

MODULE NAME: ORG_RWY
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: DATA_IN, MODIFY
CALLING SEQUENCE: CALL ORG_RWY(ADDRESS)

PURPOSE:
To insert an origin runway into the route buffer.

DESCRIPTION:
This procedure is called with the address of a runway in the navigation database. First any existing takeoff runway waypoints are removed from the flight plan. Two waypoints are inserted at the start of the route buffer. The first is placed at the position of the brake release point on the runway and the second is placed three nautical miles ahead at an elevation of 1005 feet above the runway (3 degree ascent).

The two waypoints are actually pilot waypoints created by calling MAKE_WPT. Their names are "BRnnx" and "DPnnx" where "nn" is the runway number and "x" is the left, right indicator "L" or "R". If it is the only runway with that direction at the airfield, the "x" character is shown as "X".

GLOBAL REFERENCES:

ARRAYS
AIRPTS*

RECORD ARRAYS
RTE_MOD*

FUNCTIONS AND SUBROUTINES
GET_CHAR GET_REAL MAKE_WPT NEW_POS OPEN REMOVE

MODULE NAME: PROG_SCR
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: ECHO
CALLING SEQUENCE: CALL PROG_SCR(STRING, LENGTH)

PURPOSE:
To program the CDU scratch pad.

DESCRIPTION:
The text passed to this module in the parameter STRING is programmed into the scratch pad by calling FMTOUT.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
FMTOUT

MODULE NAME: REMOVE
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: DATA_IN, GROUP, MODIFY, ORG_RWY, WAYPOINT
CALLING SEQUENCE: CALL REMOVE(RWY_CODE)

PURPOSE:
To remove runway waypoints from the flight plan.

DESCRIPTION:
This procedure is called to remove either takeoff or touch-down runway waypoints from the route buffer. The runway code parameter selects which type to delete. The module checks the ".RWY" field of all the waypoints in route buffer for RWY_CODE matches. The procedure KILL is called to remove the waypoints.

GLOBAL REFERENCES:

ARRAYS
RTE_CNT

RECORD ARRAYS
RTE_MOD

FUNCTIONS AND SUBROUTINES
KILL

MODULE NAME: ROUTE
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: SELECT
CALLING SEQUENCE: CALL ROUTE

PURPOSE:
To serve as the ROUTE page executive procedure.

DESCRIPTION:
This subroutine is the main procedure for the ROUTE page software. It performs a number of top-level functions beginning with first pass initialization and the computation of a few variables used by other modules. Most keyboard entries are handled by a call to DATA_IN, however simple keyboard requests such as page changes are performed inline. A call to the CDU screen update procedure (UPDATE) is made every time the CDU executive calls ROUTE. This is done to periodically update lines of the CDU screen even when no changes have been made.

Two items handled by ROUTE and used by other modules are the prompt boolean and page count. The line following the last ROUTE page entry may have dash prompts to indicate the next available entry position. The flag PRMT is set when the prompt is valid. The conditions are as follows.

. The flight plan does not end with a touchdown runway.
. The last item is either a route function complete with its exit waypoint or a single waypoint.
. origin and destination airfields are defined.

The LASTPG variable is computed as the number of ROUTE display pages required to show all the route data, including the prompt text.

GLOBAL REFERENCES:

VARIABLES
ERCODE* LASTPG PAGE* PASS* PGINIT* PGRQST* PMODE PRMT

ARRAYS
AIRPTS ENTRY* RTE_CNT

FUNCTIONS AND SUBROUTINES
DATA_IN DEMODE ECHO EXIT FMTOUT REJECT UPDATE

MODULE NAME: RTE_ID
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: DATA_IN RT_NEW
CALLING SEQUENCE: CALL RTE_ID(COUNT, ADDRESS, TYPE)

PURPOSE:
To identify a route function entry.

DESCRIPTION:
This procedure is called when a route function name is entered on the CDU. The navigation database is searched to determine the validity of the entry.

The paramter list contains one input and two output paramters. The count is the number of characters in the route function name, which is stored in the global buffer ENTRY. The route function address and type are returned to the caller. When the route function is not found in memory a zero address value is returned.

Two formats for route function entry exist. The first is the entry of a route function name only. In this situation the function FIND_RTE is called to search the database for the entered name and the result is immediately returned to the caller. The second format appears as three items, separated by slashes as follows.

BEARING/ROUTE_FUNCTION_NAME/EXIT_WAYPOINT_NAME

This format is used when a fixed intercept bearing from a waypoint, already part of the flight plan, onto a route function is desired. RTE_ID parses this entry to separate the three items. The bearing value is decoded and stored in the global variable BEARING. The route function and exit waypoint are then located in the system database and the offset of the exit is found on the route function. The route function address and the exit waypoint offset are returned to the caller through the global variables ADDRESS and OFFSET respectively.

GLOBAL REFERENCES:

VARIABLES
BEARING ERCODE* INDAT OFFSET*

ARRAYS
ENTRY

FUNCTIONS AND SUBROUTINES
FIND_RTE LIB$LOCC OTS$CVT_TU_L RTE_WPT WPT_ID

MODULE NAME: RTE_INTC
FILE NAME: RTE_INTC.FOR
PROCESS: SLOW
CALLED BY: DATA_IN
CALLING SEQUENCE: CODE=RTE_INTC(RTE_ADR,RTE_TYP,PTR,APPEND)

PURPOSE:
To perform the route function intercept computations.

DESCRIPTION:
This procedure is called when the Route Intercept option is requested on the RTE page. Each leg of the selected route is tested for an intersection with a radial drawn from the previous waypoint in the flight plan. More than one leg may be intersected so the nearest leg is used. Note that DME turn legs are never chosen.

The following list is a summary of the inputs and outputs referenced in RTE_INTC. These are accessed both as calling parameters and global variables.

INPUTS

BEARING	Entered magnetic bearing to route
OFFSET	Byte offset in database to exit waypoint
RTE_ADR	Address of route in database
RTE_TYP	Type of route
BUF_PTR	Pointer to insert position in RTE buffer
APPEND	Append to end of RTE buffer flag

OUTPUTS

RTE_MOD	Updated provisional route buffer

First the position of the reference waypoint in the flight plan is fetched. The reference waypoint is the waypoint immediately preceeding the position in the flight plan where the route function intercept was entered on the RTE page. The preceeding item in the route buffer may be another route function. In this case the exit waypoint of the route function is used as the reference waypoint.

Each leg segment on the selected route function is processed to determine if an intercept point exists. The procedure XYPOS is called to find the North and East offsets (feet) from the reference waypoint to the leg segment waypoints. The function INT_LEG is called for each pair to determine the existance of an intersection from the reference waypoint at the entered bearing. A "-1" is returned from INT_LEG when no intersection exists. Otherwise the distance from the reference waypoint to the intersection position is returned. The distance to the closest intersection is saved.

When finished with all waypoints a check is made to determine if an intersection has been found. If the leg belongs to an AIRWAY the exit waypoint is used to determine which direction to go from the intersect waypoint by selecting one of the leg endpoints. For unidirectional routes an error is flagged when the entered exit waypoint would cause the wrong direction to be made.

"NEW_POS" is called to compute the position of the intersect waypoint and INTC_WPTS inserts the appropriate route elements into the route buffer.

GLOBAL REFERENCES:

VARIABLES
BEARING OFFSET

RECORD ARRAYS
RTE_MOD

FUNCTIONS AND SUBROUTINES
GET_LONG GET_REAL INTC_WPTS INT_LEG MAG_VAR NEW_POS XYPOS

MODULE NAME: RTE_WPT
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: ADD_PLAN, ADD_WPT, ENTRY_WPT, EXIT_WPT, HLD_IN, KILL_CON, KILL_WPT, MERGE, NEW_CON, TRIM_WPTS, WAYPOINT
CALLING SEQUENCE: OFFSET = RTE_WPT(RTE_ADR, RTE_TYP, WPT_ADR)
CALLS TO:

PURPOSE:
To determine if a particular waypoint is part of a database route function.

DESCRIPTION:
RTE_WPT is called with the address and type of database route function. The address of a waypoint is also passed. The waypoint pointers in the route function body are tested to determine if the input waypoint is one of the route waypoints. If it is, the offset from the start of the route function to the waypoint pointer is returned. Otherwise a zero value is returned (See the format of AADCOM route function in section 1.5.1.1).

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
GET_LONG

MODULE NAME: SEQUENCE
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: MERGE, WAYPOINT
CALLING SEQUENCE: BOOLEAN=SEQUENCE(IN_OFFSET, INDEX)

PURPOSE:
To determine proper waypoint sequence for a route function.

DESCRIPTION:
This function returns a boolean result which informs the caller whether the entry and exit waypoints assigned to a route function are in the proper sequence. Anytime the entry and exit waypoints are identical the function returns an invalid status. Airways and company routes are bidirectional so any pair may serve as entry/exit. SID/STAR/APPROACH route functions have their waypoint order predefined by the order in the navigation database. Checks are performed to affirm that the correct order is maintained.

The parameter list consists of two input parameters. The first is the offset in the route function data block of the entry waypoint pointer. The second is the route buffer index of the designated route function. SEQUENCE may be called with the entry offset equal to zero. In this situation the entry offset is computed from the previous route buffer element.

GLOBAL REFERENCES:

RECORD ARRAYS
RTE_MOD

FUNCTIONS AND SUBROUTINES
ENTRY_WPT

MODULE NAME: SLASH
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: WPT_ID
CALLING SEQUENCE: INDEX = SLASH(START_INDEX)

PURPOSE:
Search for the "/" character.

DESCRIPTION:
The CDU entry buffer is searched starting at the character designated by the START_INDEX parameter. The function returns the index of the first slash found. A zero is returned when one is not found.

GLOBAL REFERENCES:

ARRAYS
ENTRY

MODULE NAME: TITLE
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: DSP_TIME, DSP_WPTS, REFRESH_HOLD, UPDATE
CALLING SEQUENCE: CALL TITLE(TEXT,COUNT,PAGE,LAST_PAGE)

PURPOSE:
To create the CDU page title line.

DESCRIPTION:
This procedure generates title line data for several of the CDU clearance pages. The page identification text is passed along with the current and last page numbers.

GLOBAL REFERENCES:

VARIABLES
PMODE

FUNCTIONS AND SUBROUTINES
FMTOUT

MODULE NAME: TYPE_WPT
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: COMPANY, ECHO, EXPAND_RTE, FILL_RTE, GROUP, INT_RTE, NEW_ENTRY, RT_NEW, RTE, UPDATE
CALLING SEQUENCE: TYPE = TYPE_WPT(ADDRESS)

PURPOSE:
To determine waypoint type.

DESCRIPTION:
The address of a navigation database waypoint is sent to this function. TYPE_WPT examines the data and returns to the caller the waypoint type. The logic is outlined below.

NAVAID: If fourth byte of data < 0.
AIRFIELD: If fifth byte of data is a blank.
GRP: Otherwise.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
GET_BYTE

MODULE NAME: UPDATE
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: ROUTE
CALLING SEQUENCE: CALL UPDATE

PURPOSE:
To update CDU display lines for the ROUTE page.

DESCRIPTION:
This subroutine formats lines of data for the CDU screen. The utility FMTOUT is used throughout to store data in the proper format (see section 1.2.1). Page #1 of the ROUTE page has its own unique format, while all other pages follow the same template. This module is called once per iteration of the CDU software. The calls are used in a cycle of seven steps. On each step a different section of the ROUTE page display is updated. After every seven calls to UPDATE the entire CDU screen will have been refreshed. The operations performed on each pass through this module are summarized below.

PASS #0: The title line is stored by a call to TITLE. Also the "Via" & "To" labels are sent to line #7 or line #1. Page #1 displaces the labels to line #7.

PASS #1/page #1: The origin and destination airfield data. Includes labels and airfield names. No airfield shown as dashes.

PASS #2/page #1: Company route label and name (or dashes).

PASS #3/page #1: Takeoff runway label and name (or dashes).

PASS #(4-5)/page #1 and
PASS #(1-5)/not page #1: The various route elements for the current page and pass. The following list shows what may be displayed on any route page line.

. Waypoint name under the "To" column. The label "DIRECT" will automatically appear under the "Via" column.

. Route function name under the "Via" column. The exit waypoint name, or box prompts, is shown under the "To" column.

. Route discontinuity marker. Shown with box prompts if previous line does not contain a route function with box prompts as the exit waypoint.

. Dash prompts for the first display line which does not contain any of the above. Only shown when the PRMT boolean is set.

. The line corresponding to the page/pass iteration is blanked when none of the above is shown.

PASS #6: The dash line and the "<INDEX" and "ERASE>" tags.

GLOBAL REFERENCES:

VARIABLES
C_ADR LASTPG PAGE PASS* PMODE PRMT

ARRAYS
AIRPTS BOXES DASHES RTE_CNT

RECORD ARRAYS
RTE_MOD

FUNCTIONS AND SUBROUTINES
EXIT FMTOUT GET_LONG TITLE TYPE_WPT

MODULE NAME: WAYPOINT
FILE NAME: ROUTE.FOR
PROCESS: SLOW
CALLED BY: ADD_WPT, DATA_IN, MODIFY, RT_NEW
CALLING SEQUENCE: CALL WAYPOINT(ADDR, TYPE, INDEX, APPEND)

PURPOSE:
To insert a waypoint into the route buffer.

DESCRIPTION:
This procedure enters a waypoint into the route buffer. Processing varies for the different waypoint types. In append mode the new waypoint is simply added to the end of the route buffer. Otherwise an insertion, possibly with the creation of route discontinuities, into the buffer is performed. When insertion occurs the following route elements, including the one at the selected position, are moved ahead in the route buffer.

There are four input parameters to WAYPOINT. The first is the database address of the entered waypoint. Next is the waypoint type (GRP, NAVAID, ...). The third parameter is an index into the route buffer to the selected position. The last parameter is a flag used to identify append mode.

A special situation arises when a runway number is entered as a waypoint. Two waypoints are placed into the route buffer and all following elements are deleted. The two waypoints are created pilot waypoints given the names APnnx and TDnnx. The "nn" field is the runway number and the "x" is either "L", "R", or "X" (left, right, neither). The "TD" waypoint is placed at the runway touchdown point and the "AP" waypoint is positioned three nautical miles out at an elevation of 1005 feet above the runway.

The remaining waypoint types are handled the same. First the append flag is tested. In this mode the entered waypoint is placed in the next available route buffer location. When not in append mode other tests must be made.

If the waypoint is inserted at a route discontinuity, two things can occur. If the entered waypoint is already in the route buffer further along the flight plan, the route buffer elements between the insertion position and the next occurence of the waypoint are deleted and the discontinuity is removed. Otherwise the new waypoint is inserted at the desired position, moving the following elements ahead one position, preserving the route discontinuity.

If the waypoint entry was not made to fill in an undefined route function exit waypoint, the waypoint is inserted and a test is made to determine if the new waypoint is a valid entry waypoint to a route function defined in the next location. If the next element is not a route function or the new waypoint does not qualify as an entry waypoint, a route discontinuity is inserted following the new waypoint.

When the entry was made to fill in an undefined exit waypoint, it is tested to verify its definition as part of the route function. For one-directional route functions (SID/STAR/APPROACH) a sequence check is made to ensure the exit comes after the entry waypoint. The last step is to look ahead for another occurrence of the entered waypoint. If it does the waypoint up to and including the duplicate are removed.

GLOBAL REFERENCES:

VARIABLES
  ERCODE*

ARRAYS
  AIRPTS* RTE_CNT

RECORD ARRAYS
  RTE_MOD*

FUNCTIONS AND SUBROUTINES
  BREAK EXIT GET_CHAR GET_REAL MAKE_WPT MERGE NEW_POS OPEN
  REMOVE RTE_WPT SEQUENCE

MODULE NAME:	WPT_ID
FILE NAME:	ROUTE.FOR
PROCESS:	SLOW
CALLED BY:	ADD_WPT, DATA_IN, RT_NEW, DIRECT, INITUP, WPT_ADDR
CALLING SEQUENCE:	CALL WPT_ID(COUNT, ADDRESS, TYPE)

PURPOSE:

To identify the entered waypoint name.

DESCRIPTION:

This procedure is called when the name of a waypoint is entered on the CDU. The system database (AADCOM) and the pilot defined waypoint buffer (PPT_WPT) are searched to determine validity and type of the waypoint. Note that this module may create pilot waypoints for global position entries such as absolute LAT/LON.

There is one input and three output parameters in the call list. The count value is the number of characters in the waypoint name, which is stored in the global buffer ENTRY. The address and type parameters are filled in once the waypoint has been identified. If the waypoint is not found, the address variable is returned to the caller zeroed.

The clearance pages of the CDU allow the use of global position references where waypoint names are normally entered. These are in the form LATITUDE/LONGITUDE or WAYPOINT/BEARING/RANGE. The first defines an absolute global position in terms of geographic coordinates. The second defines a position relative to a known reference point. When WPT_ID is called with a global reference as the waypoint name, the waypoint is first created in the pilot waypoint buffer. Then the address and type are returned as if the waypoint had previously existed and was found by the database search. When the relative reference form is used, the same validity checks are made for the reference waypoint as is done for standard waypoint identification.

The identification of waypoints depends on length and content of the entered waypoint name. The waypoint ID process is outlined below.

2 Character: Must be a runway number. A blank is appended and the runway lookup utility is called.

3 Character: Either a runway or a Navaid. If the first characer is a digit it is treated as a runway. Otherwise the Navaid lookup utility is called.

4 Character: Either an airfield name or "PPOS". If the entry is "PPOS" a pilot waypoint is created as mentioned above. Note that the entry "PPOS"/BEARING/RANGE is a a valid entry which creates two pilot waypoints. If not "PPOS" the airfield lookup utility is called.

5 Character: Either a GRP or a previously created pilot waypoint. First the GRP lookup utility is called. If successful the GRP data is returned. If not, the pilot waypoint lookup is invoked.

In all the above cases an error code is set when the various lookup procedures do not find the waypoint name in the database.

GLOBAL REFERENCES:

VARIABLES
ERCODE* LAT LON STRING

ARRAYS
AIRPTS ENTRY

RECORD ARRAYS
PPT_WPT

FUNCTIONS AND SUBROUTINES
AIRPORT DEGVAL FIND_PPT FLTVAL GET_REAL LUGRP LUNAVA LURWY
MAG_VAR MAKE_WPT NEW_POS SLASH

MODULE NAME: XYPOS
FILE NAME: RTE_INTC.FOR
PROCESS: SLOW
CALLED BY: RTE_INTC
CALLING SEQUENCE: CALL XYPOS(PTR,LAT,LON,X,Y)

PURPOSE:
To compute waypoint coordinate offsets.

DESCRIPTION:
This procedure is passed an address pointer of a waypoint entry within a database route. Also a reference waypoint's latitude and longitude are passed. The data for the route waypoint is located in the database and it's LAT/LON coordinates are fetched. The North and East offsets from the reference position are then calculated through a call to the utility GRID.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
GET_LONG GET_REAL GRID

Section 6.3.7 THE ROUTE INDEX PAGE

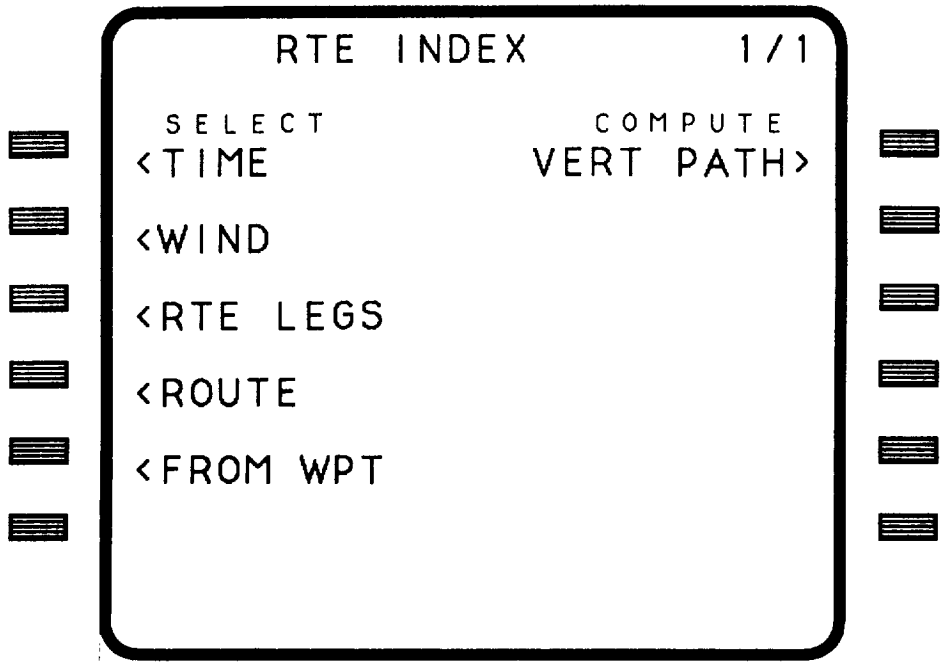
The ROUTE INDEX page is accessed via the "<INDEX" prompt which appears on the lower portion of the ROUTE, LEGS, and LEGS TIME pages. It mainly serves as an index page with prompts which may be selected to activate other pages. See figure 6.10 on the following page.

The file RTENDX.FOR contains the two subroutines for the Route Index page. The main module puts up a menu on the CDU screen and accepts pilot entries. All the left hand LSKs are used for selection of CDU pages involved with flight plan creation and modification. The upper right side LSK is used to request the vertical profile generation process on the current path (not implemented).

The two module discriptions are provided on the following pages.

PAGE 246 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED



The Route Index Page

(figure 6.10)

PAGE 248 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: PGA
FILE NAME: RTENDX.FOR
PROCESS: SLOW
CALLED BY: RTENDX
CALLING SEQUENCE: CALL PGA

PURPOSE:
A dummy module to be replaced by PGA4D algorithm when complete.

PAGE 250 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: RTENDX
FILE NAME: RTENDX.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL RTENDX

PURPOSE:
To serve as the executive module for the ROUTE INDEX page.

DESCRIPTION:
This module receives keyboard inputs and updates the CDU screen for the ROUTE INDEX page. The following chart lists the enties processed by RTENDX.

. Page change selections. If data exists, it is reprogrammed into the scratch pad. (LSK-L1 through LSK-L5)
. The PGA (dummy) module is called when requested. (LSK-R1)

A line of the display screen is updated every time before exiting this module. The screen is completely refreshed every eight calls. RTENDX has a static display so no special processing occurs for the line updates.

GLOBAL REFERENCES:

VARIABLES
ERCODE* FROMPG* PGINIT* PGRQST*

ARRAYS
ENTRY*

FUNCTIONS AND SUBROUTINES
FMTOUT PGA REPROG

VARIABLES
ERCODE* FROMPG* PGINIT* PGRQST*

Section 7.0 THE INITIALIZATION AND REFERENCE PAGES

There are a group of CDU pages refered to as the initialization and reference pages. They are used for the setting of some initial aircraft parameters, and the inspection of information pertaining to aircraft systems.

All but two of the pages are accessed through the INIT/REF index page which is selected by the INIT/REF key on the CDU keyboard. The EPR Limit and Progress pages each have their own CDU key to provide direct access to the page.

The remainder of section seven contains descriptions of the INIT/REF CDU pages and the sofware modules which manage them.

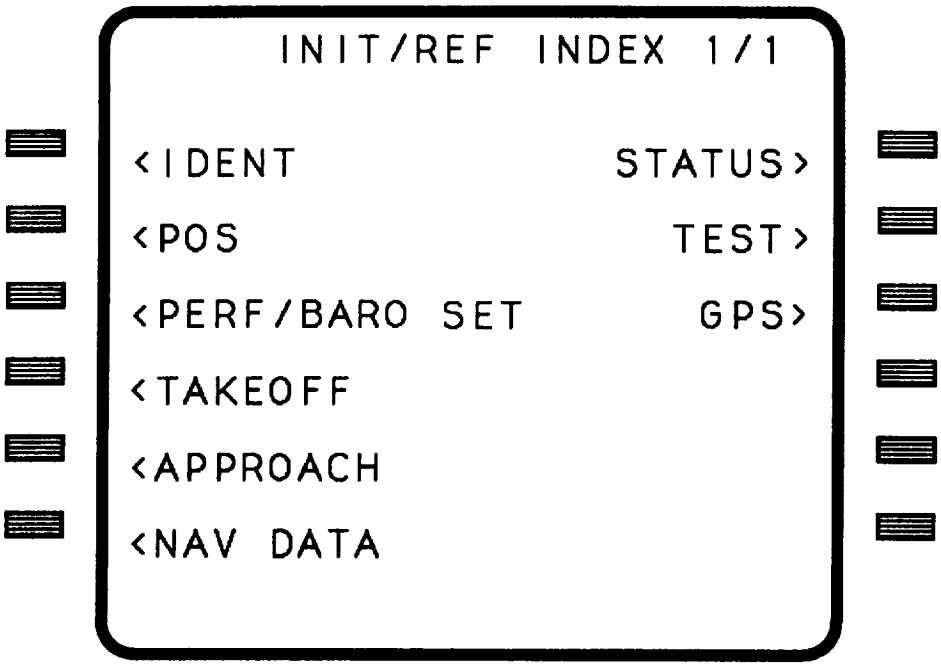
Section 7.1 THE INIT/REF INDEX PAGE

The INIT/REF page, also called the "index" page, displays seven prompts for page changes and accepts a Line Select Key (LSK) input directing a transfer to a corresponding page. This page is, effectively, a "menu" page. The diagram on the following page (figure 7.0) shows the CDU page selections available on the INIT/REF index page.

The INIT/REF code consists of a single FORTRAN module on the file INITREF.FOR.

PAGE 255 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED



The Init/Ref Index Page

(figure 7.0)

PAGE 256 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: INITREF
FILE NAME: INITREF.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL INITREF

PURPOSE:
To display a menu of CDU pages and to enable their selection by pressing the associated key.

DESCRIPTION:
The INIT/REF page, also called the INDEX page, displays seven prompts for page changes and accepts a Line Select Key (LSK) input to cause a transfer to one of those pages. The code refreshes one CDU line per pass. When there is a function key input, it is processed through a "calculated goto" structure in which the appropriate page is called or an error message displayed.

GLOBAL REFERENCES:

VARIABLES
ERCODE* PGINIT* PGRQST*

ARRAYS
ENTRY*

FUNCTIONS AND SUBROUTINES
FMTOUT REPROG

PAGE 250 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

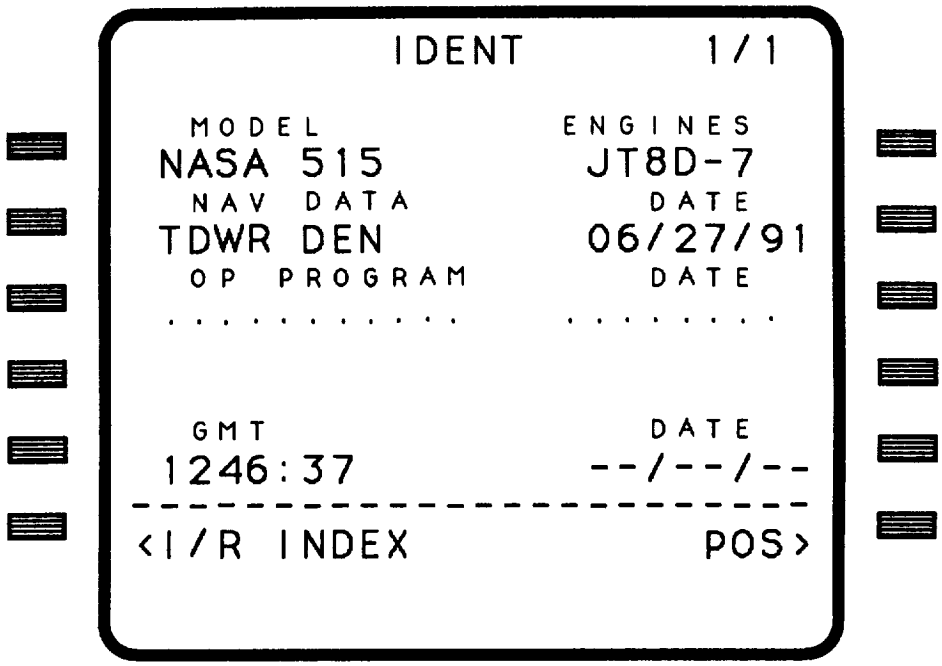
Section 7.2 THE SYSTEM IDENTIFICATION PAGE

The IDENT page is the startup CDU display; it is automatically displayed at system startup and remains active until the operator validates the time at key LSK-L5. The operator may do so by entering a time through the scratch pad or by pressing the key with a blank scratch pad. This forces use of the time from the Data Acquisition System (DAS). Date entry is optional. This display also includes two page-request choices, INDEX and POS_INIT, at key 6, left and right. Refer to figure 7.1 on the following page.

The IDENT page is coded in a single FORTRAN module.

PAGE 260 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED



The System Identification Page

(figure 7.1)

MODULE NAME: IDENT
FILE NAME: IDENT.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL IDENT

PURPOSE:
To display startup data regarding the aircraft and the software versions, and to take operator input for the time and date.

DESCRIPTION:
The IDENT page is automatically displayed at system startup and it remains active until the operator validates the system time at key LSK-L5. No other action is accepted until this is done. An actual time may be entered in the form "nnnn:nn," or the key may be pressed with a blank scratch pad to force use of the Data Acquisition System (DAS) time.

Date entry is optional and any 8-character input is acceptable. The only other options on this page are the page-request choices at key 6 (L & R) for the INDEX and POS_INIT pages.

GLOBAL REFERENCES:

VARIABLES
ERCODE HRSS MINS PGINIT* PGRQST* TIME TIME_VLD

ARRAYS
BOXES BULK_ID ENTRY*

FUNCTIONS AND SUBROUTINES
BCDTIM DEL_IN FMTOUT FMTTIM REPROG TIMVAL

PAGE 264 INTENTIONALLY BLANK

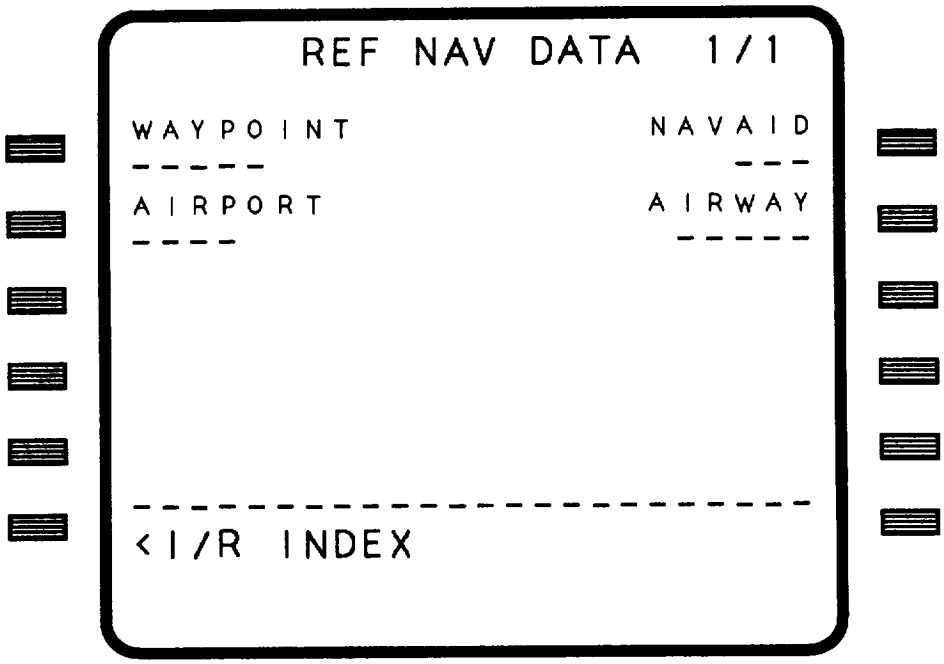
PRECEDING PAGE BLANK NOT FILMED

Section 7.3 THE REFERENCE NAVIGATION DATA PAGE

The REF NAV DATA page is used by the flight crew for identification of items such as waypoints, navaids and airports. If an item is requested via the CDU keypad, it is looked up in AADCOM and if found, the appropriate subpage is displayed. Information about the requested item is displayed on the subpage and the symbol for the requested item appears on the map display. If the item is not found in AADCOM, an informative message is displayed.

The format of the NAV DATA "root" page is shown on the follwing page (figure 7.2).

PAGE 266 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED



The Navigation Data Page

(figure 7.2)

PAGE 268 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: AIR_INPUT
FILE NAME: AIRWAY.FOR
PROCESS: SLOW
CALLED BY: NAVPG
CALLING SEQUENCE: CALL AIR_INPUT

PURPOSE:
To handle CDU entries on the airway waypoint page.

DESCRIPTION:
This module handles CDU keyboard entries while on the airway waypoint subpage of the NAV data page. The valid entries are listed below. Note that only function entries are allowed. If data is on the scratch pad when return to the "root" page is requested, it will be reprogrammed onto the scratch pad for use on the main NAV data page.

LSK-L1 through LSK-L5	Echo waypoint name to scratch pad
LSK-R1 through LSK-R5	Select map center waypoint
LSK-R6	Erase airway waypoints
NEXT	Next display page
PREV	Previous display page

Selecting a NAV format display center waypoint may be done when one of the Map displays is in "North Up" mode. The CDU display will show a "<CTR>" bug along side the chosen waypoint. The Map center selection is passed to the procedure SET_CENTER to signal the displays. When exiting from the airway waypoint display, LOKWPT is cleared to remove the airway waypoints from the Map display.

GLOBAL REFERENCES:

VARIABLES
AIR_ADR AIR_CNT AIR_LAST AIR_PG AIR_PTR* DISPST ERCODE*
PAGE* PGRQST*

ARRAYS
ENTRY LOKWPT*

FUNCTIONS AND SUBROUTINES
FMTOUT GET_LONG NAME_LEN REPROG SET_CENTER

PAGE 370 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: AIR_PAGE
FILE NAME: AIRWAY.FOR
PROCESS: SLOW
CALLED BY: NAVPG
CALLING SEQUENCE: CALL AIR_PAGE

PURPOSE:
To display the airway waypoint page.

DESCRIPTION:
This module formats the CDU display data for the airway waypoints subpage of the NAV data CDU page. The entire screen is refreshed in six consecutive passes through this procedure. On pass "0" the title line and key labels are sent out. On passes "1" through "5", the five line pairs of the display page are filled with waypoint name and LAT/LON values.

The waypoints of an airway are displayed in sets of five. The "NEXT" and "PREV" page keys may be used to view different sets of five waypoints within the airway. This module uses the global variables AIR_ADR and AIR_PG to find the waypoint information in AADCOM for the current page. The starting address of the airway is in AIR_ADR and the current airway waypoint page number is saved in AIR_PG.

The text "<CTR>" is shown on the right side of the line containing the selected Map center waypoint when one of the Map display formats is in "North Up" mode. The global variable DISPST (bit mask 2000 hex) indicates a "North Up" Map. The index within the airway of the selected Map center waypoint is obtained from the global variable AIR_PTR.

GLOBAL REFERENCES:

VARIABLES
AIR_ADR AIR_CNT AIR_LAST AIR_PG AIR_PTR DISPST

ARRAYS
DASHES

FUNCTIONS AND SUBROUTINES
FMTDEG FMTOUT GET_LONG GET_REAL NAME_LEN

MODULE NAME: CLEAR_ENTRY
FILE NAME: NAVPG.FOR
PROCESS: SLOW
CALLED BY:
CALLING SEQUENCE: CALL CLEAR_ENTRY

PURPOSE:
To fill the array OUTLINES with blanks.

DESCRIPTION:
This routine blanks all the character data in the array OUTLINES, so that all the old information can be removed from the REF NAV DATA page of the CDU display before new information is added.

GLOBAL REFERENCES:

VARIABLES
OUTLINES*

MODULE NAME: LIST_INPUT
FILE NAME: AIRWAY.FOR
PROCESS: SLOW
CALLED BY: NAVPG
CALLING SEQUENCE: CALL LIST_INPUT

PURPOSE:
To handle inputs on the airway list page.

DESCRIPTION:
This module handles CDU keyboard entries when on the airway list subpage of the NAV data page. LSK-L1 through LSK-L5 and LSK-R1 through LSK-R5 are used to echo airway names to the scratch pad. Other keys that are used are LSK-L6, LSK-R6, NEXT page, and PREV page. LSK-L6 is used to toggle between Jet and Victor airways. LSK-R6 returns to the NAV Data "root" page.

Up to ten airway names are shown per CDU airway list page. All the airway names in the current list (Victor or Jet) may be viewed ten at a time using the NEXT and PREV keys.

The airway name echoed to the scratch pad is obtained from the navigation database (AADCOM). An index into the current airway list is computed from the airway list page index and the line select key used. This index is passed to the function NAME_PTR to obtain the address of the name within AADCOM.

Switching between Jet and Victor airway lists is done by a call to SET_LIST. The index of the desired list type is passed to this procedure.

GLOBAL REFERENCES:

VARIABLES
AIR_CNT AIR_LAST AIR_PG AIR_TYPE ERCODE* PAGE* PGRQST*

ARRAYS
ENTRY

FUNCTIONS AND SUBROUTINES
FMTOUT NAME_PTR REPROG SET_LIST

MODULE NAME: LIST_PAGE
FILE NAME: AIRWAY.FOR
PROCESS: SLOW
CALLED BY: NAVPG
CALLING SEQUENCE: CALL LIST_PAGE

PURPOSE:
To display the CDU airway list page.

DESCRIPTION:
This module formats CDU display text for the airway list subpage of the NAV Data page. The entire screen is refreshed every six calls to this module. On pass "0" the title line and fixed labels are sent out. On passes "1" through "5" the available airway names for the current list type (Jet or Victor) are shown, two per line.

The global variable AIR_TYPE indicates which type list is being shown. The title line will contain the name of the current list type. Also the LSK-L6 label will contain the prompt to select the list type not currently chosen.

The variable "PASS" is used to keep track of where in the update cycle the module is. When PASS indicates one of the five lines containing airway names should be updated, an index into the airway list is computed for the two airway names which will be shown on that line. The function NAME_PTR is sent these indices to return the addresses in AADCOM of the names of the airways.

GLOBAL REFERENCES:

VARIABLES
AIR_LAST AIR_PG AIR_TYPE

ARRAYS
DASHES

FUNCTIONS AND SUBROUTINES
FMTOUT NAME_PTR

MODULE NAME: MAGV
FILE NAME: NAVPG.DOC
PROCESS: SLOW
CALLED BY: PROCESS_ARP, PROCESS_GRP, PROCESS_NAV
CALLING SEQUENCE: X = MAGV(LAT, LON)

PURPOSE:
To format the magnetic variation for the given coordinates.

DESCRIPTION:
This routine calls MAG_VAR with a latitude and longitude to compute the magnetic variation at a certain location. Then it converts the value to character data and formats it for output.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
FSTRNG MAG_VAR

MODULE NAME: NAME_LEN
FILE NAME: AIRWAY.FOR
PROCESS: SLOW
CALLED BY: AIR_INPUT AIR_PAGE
CALLING SEQUENCE: LENGTH = NAME_LEN(WPT_ADDRESS)

PURPOSE:
To find the length of a waypoint name.

DESCRIPTION:
The address of a navigation database waypoint is sent to this function. The function TYPE_WPT is then called to determine the type of the waypoint. The length is assigned from type as follows.

Navaid	3 characters
Airfied	4 characters
GRP	5 charcaters

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
TYPE_WPT

MODULE NAME: NAME_PTR
FILE NAME: AIRWAY.FOR
PROCESS: SLOW
CALLED BY: LIST_INPUT LIST_PAGE
CALLING SEQUENCE: ADDRESS = NAME_PTR(INDEX)

PURPOSE:
To find an airway name within an airway list.

DESCRIPTION:
This module returns the address of the airway name which is located in the position within the current airway list indicated by the calling parameter, INDEX. If the requested position is past the end of the airway list, a pointer to a blank character field is returned.

GLOBAL REFERENCES:

VARIABLES
AIR_ADR AIR_CNT

FUNCTIONS AND SUBROUTINES
GET_LONG

MODULE NAME: NAV_INPUT
FILE NAME: NAVPG.FOR
PROCESS: SLOW
CALLED BY: NAVPG
CALLING SEQUENCE: CALL NAV_INPUT

PURPOSE:
To parse CDU keyboard entries for the main REF NAV DATA page.

DESCRIPTION:
This subroutine is called when a keyboard entry is detected while on the main REF NAV DATA page. Valid entries on this page are limited to the following:

. Requesting the INDEX page. If there is data in the scratch pad, it is reprogrammed into the scratch pad for use by the requested page.

. Requesting the display of information about any waypoint, navaid, airway, or airport which is contained in the navigation database (AADCOM).

Upon receipt of a valid data entry, some initialization variables are set, including the PAGE variable which is set to reflect the page number of the page appropriate for the requested item type. Note that information for a waypoint, navaid or airport can be requested using either LSK-L1, LSK-L2, or LSK-R1, although the labelling of the LSKs implies that a certain LSK must be used to request information about a certain type of item. For airways information, LSK-R2 is used exclusively.

GLOBAL REFERENCES:

VARIABLES
ERCODE* PGRQST*

ARRAYS
ENTRY

FUNCTIONS AND SUBROUTINES
DEL_IN PROCESS_AIRWAY PROCESS_ARP PROCESS_GRP PROCESS_NAV
REPROG SET_LIST

MODULE NAME: NAVPG
FILE NAME: NAVPG.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL NAVPG

PURPOSE:
To serve as the REF NAV DATA page executive module.

DESCRIPTION:
This subroutine is the main procedure for the REF NAV DATA page software. It performs a few top-level functions beginning with first pass initialization and the computation of a few variables used by other modules. Input to the REF NAV DATA page is handled by the module NAV_INPUT, unless one of the subpages is active. The procedure SUBNAV_INPUT is called for all subpages except for airway information. For airways either AIR_INPUT or LIST_INPUT is called depending on the airway information shown. The screen is updated by a call to REFRESH except when on an airway subpage. Airway data updates are made by a call to either the procedure AIR_PAGE or LIST_PAGE.

GLOBAL REFERENCES:

VARIABLES
PAGE PASS* PGINIT*

ARRAYS
ENTRY* OUTLINES*

FUNCTIONS AND SUBROUTINES
AIR_INPUT AIR_PAGE CLEAR_ENTRY LIST_INPUT LIST_PAGE
NAV_INPUT REFRESH SUBNAV_INPUT

MODULE NAME: PROCESS_AIRWAY
FILE NAME: AIRWAY.FOR
PROCESS: SLOW
CALLED BY: NAV_INPUT
CALLING SEQUENCE: CALL PROCESS_AIRWAY

PURPOSE:
To handle requests for the airway waypoints page.

DESCRIPTION:
This module is called when an airway name is entered on the NAV Data page of the CDU for display of the waypoints contained within it. Global variables are set up to enable the airway waypoint display subpage. If the requested airway is not found in the database an error code is set and the subpage is not enabled.

The following list of global variables are set by this procedure when the requested airway is found in the database.

AIR_PG	Current airway waypoints subpage is initialized to the first page.
AIR_PTR	Map center waypoint is initialized to the first waypoint on the airway.
AIR_CNT	Number of waypoints on the airway is determined by stepping through the database.
AIR_LAST	Nuumber of subpages required to show all the airway waypoints, five per page.
PAGE	Set to "4" to indicate the airway waypoint page. is active from the NAV Data "root" page.
LOKWPT	The airway address and ID code are set in this array to cause the displays to place the selected airway on the Map format.

The subroutine SET_CENTER is called to set the Map center latitude and longitude from the Map center waypoint index (AIR_PTR). This is used by any Map display format in "North Up" mode.

GLOBAL REFERENCES:

VARIABLES
AIR_ADR AIR_CNT AIR_LAST* AIR_PG* AIR_PTR* ERCODE* INDAT PAGE*

ARRAYS
ENTRY LOKWPT*

FUNCTIONS AND SUBROUTINES
GET_LONG LUJET LUVIC SET_CENTER

MODULE NAME: PROCESS_ARP
FILE NAME: NAVPG.FOR
PROCESS: SLOW
CALLED BY: NAV_INPUT, SUBNAV_INPUT
CALLING SEQUENCE: CALL PROCESS_ARP

PURPOSE:
To locate an airport in the navigational database and format information for display on the REF NAV DATA page.

DESCRIPTION:
This routine is called to update the contents of the array OUTLINES, which contains appropriate informational data for display on the REF NAV DATA page. Initially, this routine looks up an airport in the navigational data base (AADCOM), with a call to LUARP. If found, it fetches informational data for that airport, performs any formatting or character conversion and stores the information in the array, OUTLINES. It stores the address of the airport and an airport type code in LOKWPT, so that the airport can be displayed on the MAP display format. Also, the variables LATCEN and LONCEN are set to the latitude and longitude of the specified airport, for the purpose of specifying the center of the MAP format. If the requested airport is not found in AADCOM an appropriate error message is displayed.

Note that prior to looking up the airport in AADCOM, the previous item address is stored off. This handles the particular case where airport information is currently displayed and a different airport is requested. If the new airport is not found, the original information remains on the CDU display. Then if a runway for the displayed airport is requested, the original airport address is available to look up the runway.

GLOBAL REFERENCES:

VARIABLES
ADDR ERCODE* LATCEN* LONCEN* NLAT NLON PAGE* PASS* TARP

ARRAYS
ENTRY LOKWPT* OUTLINES*

FUNCTIONS AND SUBROUTINES
CLEAR_ENTRY FMTDEG FMTOUT FSTRNG GET_REAL LUARP MAGV
GET_WORD LUNAVA MAGV

MODULE NAME: PROCESS_GRP
FILE NAME: NAVPG.FOR
PROCESS: SLOW
CALLED BY: NAV_INPUT, SUBNAV_INPUT
CALLING SEQUENCE: CALL PROCESS_GRP

PURPOSE:
To locate a waypoint in the navigational database and format information for display on the REF NAV DATA page.

DESCRIPTION:
This routine is called to update the contents of the array OUTLINES, which contains appropriate informational data for display on the REF NAV DATA page. Initially, this routine looks up a waypoint in the navigational database (AADCOM), with a call to LUGRP. If it is not found, it searches for the requested waypoint in the pilot defined waypoint buffer. If the waypoint is not located in either place, an appropriate error message is displayed.

If the waypoint is found in either AADCOM or the pilot defined waypoint buffer, this routine then fetches informational data for that waypoint, performs any formatting or character conversion and stores the information in the array, OUTLINES. Note that the information displayed for a pilot defined waypoint differs slightly from the information diplayed for a waypoint which is contained in the navigation database. In either case, this routine stores the address of the waypoint and a waypoint type code in LOKWPT, so that the waypoint can be displayed on the MAP display format. Also, the variables LATCEN and LONCEN are set to the latitude and longitude of the specified waypoint, for the purpose of specifying the center of the MAP format.

GLOBAL REFERENCES:

VARIABLES
ADDR ERCODE* LATCEN* LONCEN* NLAT NLON PAGE* PASS* TGRP

ARRAYS
ENTRY LOKWPT* OUTLINES*

RECORD ARRAYS
PPT_WPT

FUNCTIONS AND SUBROUTINES
CLEAR_ENTRY FIND_PPT FMTDEG FMTOUT LUGRP MAGV

MODULE NAME: PROCESS_NAV
FILE NAME: NAVPG.FOR
PROCESS: SLOW
CALLED BY: NAV_INPUT, SUBNAV_INPUT
CALLING SEQUENCE: CALL PROCESS_NAV

PURPOSE:
To locate a navaid in the navigational database and format information for display on the REF NAV DATA page.

DESCRIPTION:
This routine is called to update the contents of the array OUTLINES, which contains appropriate informational data for display on the REF NAV DATA page. Initially, this routine looks up a navaid in the navigational database (AADCOM), with a call to LUNAVA. If found, it fetches informational data for that navaid, performs any formatting or character conversion and stores the information in the array, OUTLINES. It stores the address of the navaid and a navaid type code in LOKWPT, so that the navaid can be displayed on the MAP display format. Also, the variables LATCEN and LONCEN are set to the latitude and longitude of the specified navaid, for the purpose of specifying the center of the MAP format. If the requested navaid is not found in AADCOM an appropriate error message is displayed.

GLOBAL REFERENCES:

VARIABLES
ADDR ERCODE* LATCEN* LONCEN* NLAT NLON PAGE* PASS* TNAV

ARRAYS
ENTRY LOKWPT* OUTLINES*

FUNCTIONS AND SUBROUTINES
CLEAR_ENTRY FMTDEG FMTOUT FRMFRQ FSTRNG GET_BYTE GET_REAL GET_WORD LUNAVA MAGV

MODULE NAME: PROCESS_RWY
FILE NAME: NAVPG.FOR
PROCESS: SLOW
CALLED BY: SUBNAV_INPUT
CALLING SEQUENCE: CALL PROCESS_RWY

PURPOSE:
To locate a runway in the navigational database and format information for display on the REF NAV DATA page.

DESCRIPTION:
This routine is called to update the contents of the array OUTLINES, which contains appropriate informational data for display on the REF NAV DATA page. This routine looks up a runway in the navigational database (Bulk Data), beginning the search at a specified airport address, with a call to LURWY. If found, it fetches informational data for that runway, performs any formatting or character conversion and stores the information in the array, OUTLINES. It stores the address of the runway and a runway type code in LOKWPT, so that the runway can be displayed on the MAP display format. If the requested runway is not found in AADCOM an appropriate error message is displayed.

GLOBAL REFERENCES:

VARIABLES
ADDR ERCODE* TRWY

ARRAYS
ENTRY LOKWPT* OUTLINES*

FUNCTIONS AND SUBROUTINES
FSTRNG GET_REAL LURWY

MODULE NAME: REFRESH
FILE NAME: NAVPG.FOR
PROCESS: SLOW
CALLED BY: NAVPG
CALLING SEQUENCE: CALL REFRESH

PURPOSE:
To update the CDU display for the REV NAV DATA pages.

DESCRIPTION:
This subroutine updates the CDU display for the REF NAV DATA page with calls to FMTOUT. The entire screen is updated every sixteen consecutive calls to this subroutine. The value of PASS determines which particular lines are updated. During the first call of the cycle, the page title is output along with an indication of the current and last page numbers. Information about the requested item is displayed on lines #1 through #10. Line #11 contains dashes and line #12 contains a label for the LSK which provides access to the INDEX page. If information is presently being displayed on the REF NAV DATA page, then line #12 also contains a label for the LSK which enables the erasure of the information.

GLOBAL REFERENCES:

VARIABLES
PAGE PASS*

ARRAYS
DASHES OUTLINES

FUNCTIONS AND SUBROUTINES
FMTOUT

MODULE NAME: SET_CENTER
FILE NAME: AIRWAY.FOR
PROCESS: SLOW
CALLED BY: PROCESS_AIRWAY, AIR_INPUT, NAVPG
CALLING SEQUENCE: CALL SET_CENTER

PURPOSE:
To set the Map center LAT/LON.

DESCRIPTION:
This module fetches the latitude and longitude of the airway waypoint chosen as the Map center position. The global variable AIR_PTR is the index of the waypoint within the currently displayed airway. The variables LATCEN and LONCEN are set for use by the NAV display format. The values are found by using the start address (AIR_ADR) and an offset computed from the Map center index to access the waypoint address. See the database description (AADCOM) in section 6.1.1 for the structure of airways within the database.

GLOBAL REFERENCES:

VARIABLES
AIR_ADR AIR_PTR GDTIME* LATCEN* LONCEN*

FUNCTIONS AND SUBROUTINES
GET_LONG GET_REAL

MODULE NAME: SET_LIST
FILE NAME: AIRWAY.FOR
PROCESS: SLOW
CALLED BY: NAV_INPUT LIST_INPUT
CALLING SEQUENCE: CALL SET_LIST(AIRWAY_TYPE)

PURPOSE:
To handle requests for the airway list page.

DESCRIPTION:
This procedure is called to enable the airway list page, when LSK-R2 is pressed with no data on the scratch pad, on the NAV Data page of the CDU. It is passed the type of airway list desired in the calling sequence (0: Jet airway 1: Victor airway).
The following global variables are set up by this procedure to enable the airway list page.

AIR_PG	The current airway list subpage is initialized to the first one.
AIR_TYPE	Set to the index of the requested list type (Jet or Victor).
AIR_ADR	Set to the starting address within the database of the airway list.
AIR_CNT	Set to the number of airways in the list.
AIR_LAST	Set to the number of pages required to show all the airway names in the list (10 per page).
PAGE	Set to "5" to indicate the airway list page of the NAV data page is enabled.

Refer to section 6.1.1 for the structure of airway lists in the navigation database (AADCOM).

GLOBAL REFERENCES:

VARIABLES
AIR_ADR AIR_CNT AIR_LAST* AIR_PG* AIR_TYPE* JPTR PAGE* VPTR

FUNCTIONS AND SUBROUTINES
GET_LONG

MODULE NAME: SUBNAV_INPUT
FILE NAME: NAVPG.FOR
PROCESS: SLOW
CALLED BY: NAVPG
CALLING SEQUENCE: CALL SUBNAV_INPUT

PURPOSE:
To parse CDU data entries for the navaid data subpage, the waypoint data subpage or the airport data subpage of the REF NAV DATA page.

DESCRIPTION:
This subroutine is called when a data entry is detected while on one of the three informational subpages of the REF NAV DATA page. Valid entries on this page are limited to the following:

. Requesting the INDEX page. If there is data in the scratch pad, it is reprogrammed into the scratch pad for use by the requested page.

. Requesting the display of information about any waypoint, navaid or airport which is contained in the navigation database (AADCOM). Upon receipt of a valid data entry, some initialization variables are set, including the PAGE variable which is set to reflect the page number of the page appropriate for the requested item type. Note that information may be requested for a waypoint, navaid or airport from any of the three informational subpages.

. Erasing the display of waypoint, navaid or airport information. The main REF NAV DATA page is then displayed and the item is also removed from the MAP display format.

. If information on an airport is curently displayed, then information about a particular runway may be requested.

GLOBAL REFERENCES:

VARIABLES
ERCODE* PAGE PGINIT* PGRQST*

ARRAYS
ENTRY LOKWPT*

FUNCTIONS AND SUBROUTINES
DEL_IN FMTOUT PROCESS_ARP PROCESS_GRP PROCESS_NAV PROCESS_RWY
REPROG

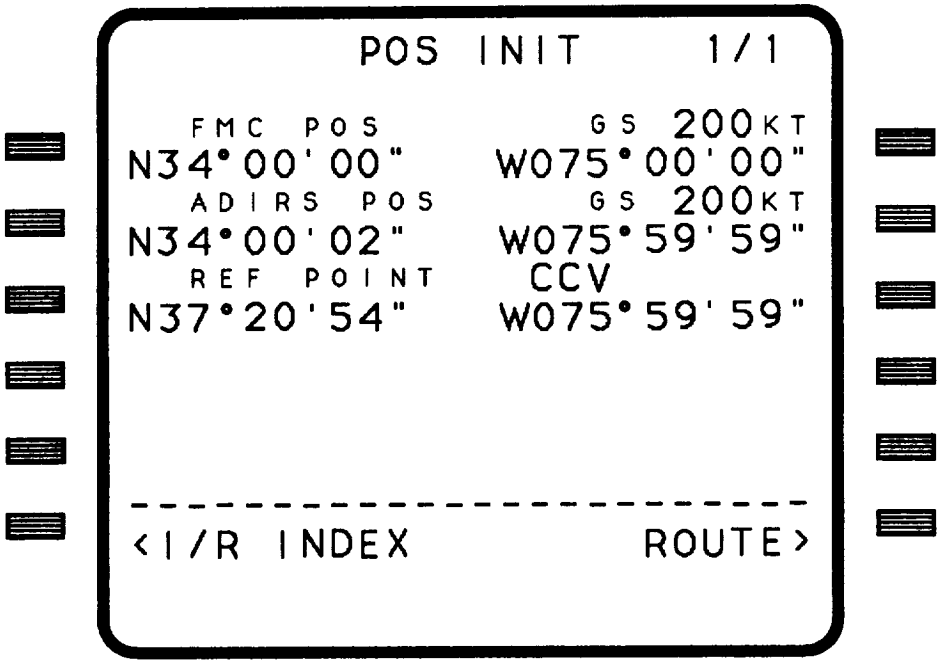
Section 7.4 THE INITIAL POSITION PAGE

The "Initial Position" page, labelled INITPOS, displays the aircraft geographical position and groundspeed and allows the operator to initialize or modify the position estimate. It also, on request, displays the location of any reference point in the database. Refer to figure 7.3 on the following page.

The INITPOS page is coded in three FORTRAN subroutines on the file INITPOS.FOR (INITPOS, INITUP, and STRIPR).

PAGE 390 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED



The Position Initialization Page

(figure 7.3)

PAGE 292 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: INITPOS
FILE NAME: INITPOS.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL INITPOS

PURPOSE:

A) Permits the operator to initialize or modify the aircraft position estimate.
B) Displays the aircraft position and groundspeed.
C) Displays the location of any reference point in the database, in response to an operator request.

DESCRIPTION:

INITPOS nominally refreshes one CDU line per iteration, under the control of a "calculated GOTO" structure. If there is user input, subroutine INITUP is called first to take care of that.

The discrete POSMOD controls the display in several respects, both in the main routine and in INITUP. It is set in INITUP when the operator inputs a valid Lat/Lon to update the computed aircraft position. While it is TRUE, the word "MOD" is displayed in reverse video on the title line of the CDU. Also, for the FMC position on line #2, the user's input is held on static display in reverse video until it is either accepted or rejected by the user. Finally, on line #12, the prompts "<ERASE" and "ACCEPT>" are displayed in place of the usual page change prompts.

Page refreshing is otherwise straightforward except that position validity is verified by checking the discrete INAVV and/or LLINIT, and the reference point status by checking LINE6_ACT. If the FMC position hasn't been initialized, it is displayed as boxes, indicating that the initialization is required. At line #5, if a reference point has not been displayed earlier, the label is output as dashes and the coordinates line is left blank.

GLOBAL REFERENCES:

VARIABLES
GS GSINS INAVV LAT LATINS LATMP LINE6_ACT LLINIT LON LONINS LONTMP PGINIT* POSMOD RFCNT

ARRAYS
DASHES ENTRY* LATADIR LATFMC LATRF LONADIR LONFMC LONRF RFID

FUNCTIONS AND SUBROUTINES
FMTDEG FMTOUT INITUP OTS$CVT_L_TI

MODULE NAME: INITUP
FILE NAME: INITPOS.FOR
PROCESS: SLOW
CALLED BY: INITPOS
CALLING SEQUENCE: CALL INITUP

PURPOSE:
To process function key and scratch pad input.

DESCRIPTION:
Subroutine INITUP processes user input which might be either a position update from the scratch pad, or a function key command. INITUP is structured as a nested IF/ELSE command. At the first level, the "IF" block processes function key inputs, and the "ELSE" block handles data input from the scratch pad. In each case, all the function keys are checked. Error messages are sent out for bad data and for invalid keys. When a position update or display is called for, FMTDEG is called to generate a latitude and longitude. Then STRIPR is called to reformat the position data and display it on the scratch pad. The flag POSMOD is set when a position update is in progress and it subsequently controls the interpretation of key input in most cases.

GLOBAL REFERENCES:

VARIABLES
ERCODE* IDDLAT* IDDLON* INAVV INDAT LAT LATINS LATMP LINE6_ACT*
LLINIT LON LONINS LONTMP PGRQST* POSMOD* RFADDR RFCNT

ARRAYS
ENTRY LATADIR LATFMC LATRF LONADIR LONFMC LONRF RFID*

FUNCTIONS AND SUBROUTINES
DEGVAL DEL_IN FMTDEG GET_REAL REPROG STRIPR WPT_ID

MODULE NAME: STRIPR
FILE NAME: INITPOS.FOR
PROCESS: SLOW
CALLED BY: INITUP
CALLING SEQUENCE: CALL STRIPR

PURPOSE:
To remove degree, minute, and second symbols from the byte strings for latitude and longitude and then to print the position information on the scratch pad as: Nddmmss/Eddmmss.

DESCRIPTION:
The digits of the latitude and longitude are read into a byte buffer, a slash is inserted between them, and FMTOUT is called to print the position on the scratch pad.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
FMTOUT

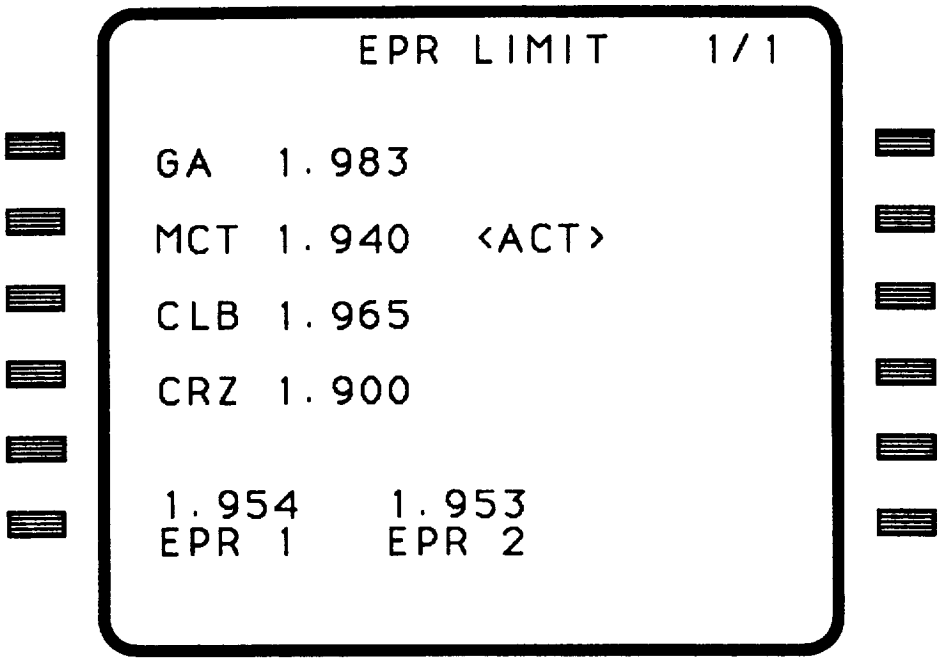
Section 7.5 THE EPR LIMIT PAGE

The engine pressure ratio (EPR) limits are calculated in subroutine EPRLMT. The EPRLIM page displays the four EPR limits, for go-around, maximum continuous thrust, climb, and cruise. It also permits the operator to change the active limit by pressing the line select key adjacent to the desired limit. The active limit is indicated by the "<ACT>" flag next to it. Additionally, the current EPR for each engine is listed at the bottom of the display. Refer to figure 7.4 on the following page.

This routine is coded in one FORTRAN subroutine titled EPRLIM.

PRECEDING PAGE BLANK NOT FILMED

PAGE 298 INTENTIONALLY BLANK



The EPR Limit Page

(figure 7.4)

PAGE 300 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: EPRLIM
FILE NAME: EPRLIM.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL EPRLIM

PURPOSE:
To print the engine pressure ratio (EPR) limits on the EPRLMT display and to permit manual selection of the EPR limit for auto-throttle operations.

DESCRIPTION:
This routine displays EPR limits; it does not compute them. That is done separately by subroutine EPRLMT.

As illustrated in figure 7.4, the EPR limits are displayed for go-around, maximum continuous thrust, climb, and cruise. Additionally, the current EPR for each engine is displayed at the bottom of the screen.

The code first checks for a function key input, accepting keys L1 through L4 only. If a valid key was pressed, the global EPRFLG is set to a number signifying the selected EPR. If no key was pressed, the routine refreshes one line of the display per pass. The flag "<ACT>" is shown adjacent to the selected EPR limit as dictated by EPRFLG.

GLOBAL REFERENCES:

VARIABLES
EC6 EPR1 EPR2 EPRFLG ERCODE* GAEPR MCLEPR MCREPR MCTEPR PGINIT*

ARRAYS
ENTRY*

FUNCTIONS AND SUBROUTINES
FMTOUT OTS$FLOAT

PAGE 302 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED

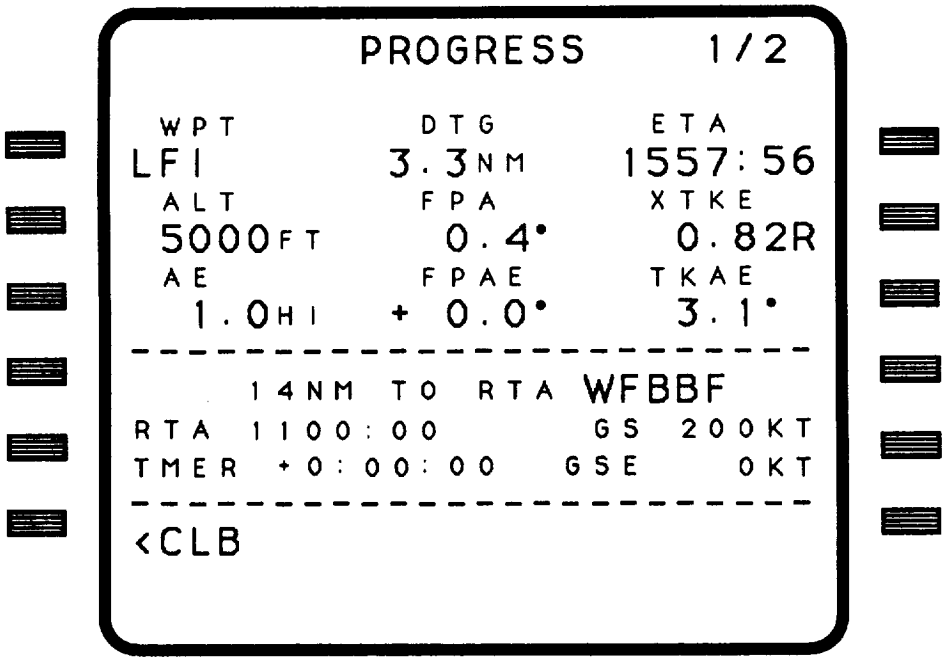
Section 7.6 THE PROGRESS PAGE

The position of the aircraft and key navigation data relative to the flight plan are presented on the two PROGRESS pages. In particular, page 1 displays data pertinent to the active waypoint and to the active 2D, 3D, or 4D flight path. Page two displays the current position estimate and the currently selected navaids. This page also allows the user to transfer the position or DME data to the scratch pad and to manually reset DME 2 or DME 3. Manual tuning mode can be toggled to automatic mode, or vice versa, by pressing the delete key followed by the appropriate line key. Both pages offer a page change to the Climb, Cruise, or Descent pages as appropriate for the current flight phase. Refer to figures 7.5 and 7.6 on the following pages.

This display is coded in two FORTRAN subroutines, PROGRESS and ACTION, in the file PROGRESS.FOR.

PAGE 304 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

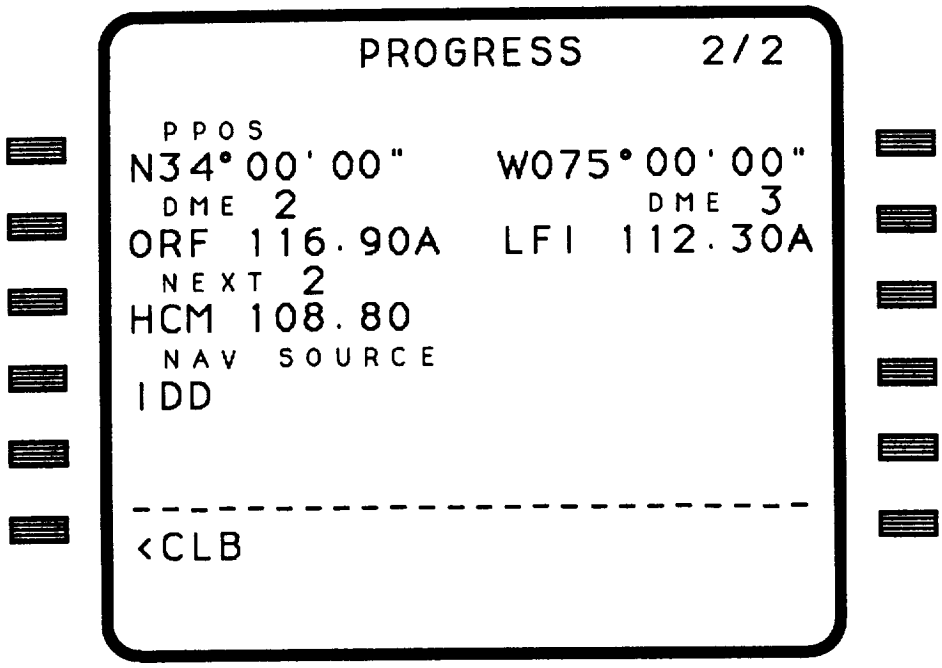


The Progress Page (#1)

(figure 7.5)

PRECEDING PAGE BLANK NOT FILMED

PAGE 306 INTENTIONALLY BLANK



The Progress Page (#2)

(figure 7.6)

PAGE 308 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: ACTION
FILE NAME: PROGRESS.FOR
PROCESS: SLOW
CALLED BY: PROGRESS
CALLING SEQUENCE: CALL ACTION(PASS,PAGE1)

PURPOSE:
To process user input for the PROGRESS page.

DESCRIPTION:
In subroutine ACTION, function key input is processed first, beginning with the functions common to both pages. This includes page requests (CLB, CRZ, DESC) and toggling between PROGRESS pages 1 and 2. Since no other function is valid for page 1, processing continues for page 2. For keys L1 and R1, the position coordinates are moved to the scratch pad. For keys L2, R2, and L3, the station identifier is moved to the scratch pad.

For data input, which is allowed only on page 2 and only for the navaid identifiers on line 4 (keys L2 & R2), three letter inputs are used to find the address of a new navaid. Otherwise, if the DELETE key was pressed and manual tuning was in effect, then auto-tuning is enabled. Any other keys or conditions result in an error message.

GLOBAL REFERENCES:

VARIABLES
ATNAV2* ATNAV3* ATUNE2* ATUNE3* ERCODE* GUID2D INAVV LAT LLINIT LON NVAD2A* VAD2B NVAD3A* PGRQST* TOWPT

ARRAYS
ENTRY*

RECORD ARRAYS
WPT_ACT

FUNCTIONS AND SUBROUTINES
DEL_IN FMTDEG FMTOUT GET_CHAR GET_WORD LUNAVA REPROG STRIPR

PAGE 310 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: PROGRESS
FILE NAME: PROGRESS.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL PROGRESS

PURPOSE:

A) To display data on the active waypoint and the active 2D, 3D, or 4D flight path.
B) To display and allow changes to the tuning of the navigation radios.
C) To determine which page change prompt to offer, and to request that change in response to key input.

DESCRIPTION:

The PROGRESS page 1 allows the flight crew to observe data pertinent to the active waypoint and to the active 2D, 3D, or 4D flight path. Page 2 displays the aircraft position estimate and the selected navaids. It also implements manual tuning of the navigation radios and an option is provided to toggle the radio tuning mode between manual and automatic.

Both pages use LSK-L5 to request transfer to the Climb, Cruise, or Descent pages. Which of the three is determined by external conditions [WPT_ACT(TOWPT-1).PHASE].

On page 1, all other line select keys (LSKs) are non-functional. On page 2, the operator may transfer the Lat/Lon or any of the three Navaid identifiers to the scratch pad. Page 2 updates are done by selecting the appropriate LSK when there is valid data on the scratch pad. The "next #2 navaid" cannot be updated by the user.

When either active Navaid is in manual tuning mode, it can be reset to auto by pressing the DELETE key followed by the appropriate LSK.

The code is structured so that on any operator interaction subroutine ACTION is called to process the input, after which subroutine PROGRESS terminates. If there is no input, then the routine computes and refreshes one CDU line per pass. Which page to update is determined by the discrete PAGE1, which, in turn, is toggled when the user presses the NEXT or PREV key on the CDU panel.

There are two "calculated goto" structures, one for each page, followed by a correspondingly labelled code block for each line on each page. The progress data and their meanings are outlined below:

PAGE 1:	Label		Source or Meaning
Line 2:	WPT	=	WPT_ACT(TOWPT).NAME
	DTG	=	DTOGO converted to nautical miles
	ETA	=	(DTOGO/GSFPS) + TIME
Line 4:	ALT	=	ALTCOR
	FPA	=	GAMMA
	XTKE	=	XTK x FTONM, "R" if XTK > 0. Blank if not at least 2D guidance.
Line 6:	AE	=	HER, "LO" if HER > 0.
	FPAE	=	PFPA - GAMMA, "+" if > 0.
	TKAE	=	TKE, "R" if > 0. Blank if not at least 2D guidance.
Lines 8-10:			
	RTA	=	Done through calls to Climb page (RTA_LN8, RTA_LN9, RTA_LN10)

NOTE: If no RTA is available, "NO RTA ASSIGNED" appears on line 8, and lines 9-11 are blank.

PAGE 2:	Label		Source or Meaning
Line 2:	PPOS	=	LAT, LON Blank if no valid position estimate.
Line 4:	DME 2		
	ID	=	NAVAD2A
	FREQ	=	ATUNE2, "A" if ATNAV2 AND DME2FQ = ATUNE2 "M" if NOT ATNAV2
	DME 3		
	ID	=	NAVAD3A
	FREQ	=	ATUNE3, "A" if ATNAV3 AND DME3FQ = ATUNE3 "M" if .NOT. ATNAV3
Line 6:	NEXT 2		
	ID	=	NAVAD2B
	FREQ	=	NAVAD2B + 4
Line 7:	NAV SRC	=	NAVTYP
LINE 12:			Both pages:
	Prompt	=	WPT_ACT(TOWPT-1).PHASE If = 1 then use <CLB If = 2 " " <CRC If = 3 " " <DES But no prompt if not at least 2D guidance.

GLOBAL REFERENCES:

VARIABLES
ACTCNT ALTCOR ATNAV2 ATNAV3 ATUNE2 ATUNE3 DME2FQ DME3FQ DTOGO
GAMMA GSFPS GUID2D GUID3D HER INAVV LAT LLINIT LON NVAD2A
NVAD2B NVAD3A PFPA PGINIT* TIME TKE TOWPT XTK

ARRAYS
DASHES ENTRY NAVTYP

RECORD ARRAYS
WPT_ACT

FUNCTIONS AND SUBROUTINES
ACTION FMTDEG FMTOUT FMTTIM FRMFRQ GET_CHAR GET_WORD
OTS$CVT_L_TI OTS$FLOAT RTA_LN10 RTA_LN8 RTA_LN9

Section 7.7 THE PERFORMANCE INITIALIZATION PAGE

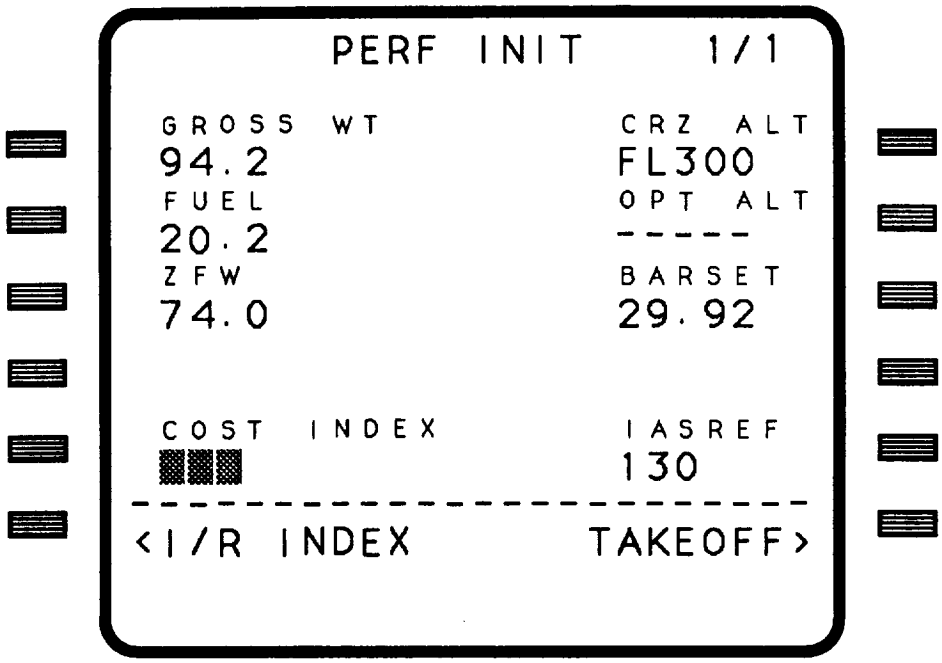
The PERFORMANCE/INITIALIZATION page of the CDU is used to initialize and display various aircraft performance parameters. This CDU page is accessed by pressing the "<PERF" prompt on the INIT/REF INDEX page and is exited by pressing the "<INDEX" prompt or "TAKEOFF>" prompt on the PERFORMANCE/INITIALIZATION page, or by selecting a different CDU page from the CDU panel. Refer to figure 7.7 on the following page.

The aircraft parameters and their associated global variables that can be modified via this page include:

PARAMETER	VARIABLE	UNITS
AIRCRAFT GROSS WEIGHT	WEIGHT	LBS*1000
AIRCRAFT ZERO FUEL WEIGHT	ZFW	LBS*1000
BARO SETTING	BARSET	IN
COST INDEX	CINDEX	-
CRUISE ALTITUDE	CRZALT	FT
INDICATED AIRSPEED REFERENCE	IASREF	KTS
OPTIMUM ALTITUDE	OPTALT	FT

The aircraft filtered total fuel quantity varaible FTFQ is also displayed on this page, but its value can not be modified.

The parameters on this page are updated once every 15 SLOW task cycles.



The Performance Initialization Page

(figure 7.7)

PAGE 316 INTENTIONALLY BLANK

PRECEDING PAGE BLANK-NOT FILMED

MODULE NAME: PFINIT
FILE NAME: PFINIT.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL PFINIT

PURPOSE:
The purpose of this CDU module is to display as well as accept modifications to various aircraft performance parameters.

DESCRIPTION:
This module is called by the CDU executive CDUEXC when the user has selected the PERFORMANCE/INITIALIZATION page. The CDU page managed by this module is accessed by pressing the "<PERF" prompt on the INIT/REF INDEX page and is exited by pressing the "<INDEX" prompt or "TAKEOFF>" prompt on the PERFORMANCE/INITIALIZATION page, or by selecting a different CDU page from the CDU panel.

If any CDU inputs are made, the PFINIT input processing routine PFINP is called.

PFINIT is called once per SLOW cycle. One CDU line is output per PFINIT call. Actual CDU output is handled via calls to the FMTOUT routine.

GLOBAL REFERENCES:

VARIABLES
BARSET CINDEX CRZALT FLKEY FTFQ GWSET IASREF OPTALT
PGINIT* WEIGHT ZFW

ARRAYS
BOXES DASHES ENTRY*

FUNCTIONS AND SUBROUTINES
CDU_SMALL FMTOUT OTS$CVT_L_TI OTS$FLOAT PFINP

PAGE 318 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: PFINP
FILE NAME: PFINIT.FOR
PROCESS: SLOW
CALLED BY: PFINIT
CALLING SEQUENCE: CALL PFINP

PURPOSE:
The purpose of this routine is to handle user inputs made from the CDU PERFORMANCE/INITIALIZATION page.

DESCRIPTION:
This module is called from the PFINIT routine whenever a user makes an input from the CDU PERFORMANCE/INITIALIZATION page. There are two basic types of CDU inputs handled by this module -- 1) Blank Scratch Pad + Line Select Key (LSK), and 2) Scratch Pad Data + Line Select Key.

If there is no data on the scratch pad when an LSK is pressed, the following applies:

LSK-L1: Since data was expected for "GROSS WT", a "NO DATA" error message will be displayed on the scratch pad.

LSK-L2: The "FUEL" parameter cannot be modified. A "DEAD KEY ERROR" message will be displayed on the scratch pad.

LSK-L3: Since data was expected for "ZFW", a "NO DATA" error message will be displayed on the scratch pad.

LSK-L4: No parameter. A "DEAD KEY ERROR" message will be displayed on the scratch pad.

LSK-L5: Since data was expected for "COST INDEX", a "NO DATA" error message will be displayed on the scratch pad.

LSK-L6: The "<INDEX" prompt was pressed and the CDU INIT/REF INDEX page becomes active.

LSK-R1: "CRZALT" was pressed. If the global Cruise Altitude variable CRZALT is > 0, the value of CRZALT is copied to the scratch pad by calling the PROG_LN routine; otherwise, a "DEAD KEY ERROR" error message will be displayed on the scratch pad.

LSK-R2: Although the "OPT ALT" parameter is shown at this LSK, it is not currently implemented. No data will be accepted. A "DEAD KEY ERROR" error message will be displayed on the scratch pad.

LSK-R3: Since data was expected for "BARSET", a "NO DATA" error message will be displayed on the scratch pad.

LSK-R4: No parameter. A "DEAD KEY ERROR" message will be displayed on the scratch pad.

LSK-R5: Since data was expected for "IASREF", a "NO DATA" error message will be displayed on the scratch pad.

LSK-R6: The "TAKEOFF>" was pressed and the CDU TAKEOFF page becomes active.

If there is data on the scratch pad prior to pressing an LSK, the following applies:

LSK-L1: A data entry was made for "GROSS WT". The data should be the aircraft gross weight in thousands of pounds. For example, if the aircraft is known to weigh 93100 pounds, the data entry should be 93.1. The zero fuel weight is computed as gross weight less the filtered total fuel quantity; however, before setting the actual global variables GRWGT and ZFW, the FUEL_LIM routine is called to verify the following:

74000 <= new GRWGT <= 102000

&

65000 <= new ZFW <= 77600.

If these tests are passed, GRWGT and ZFW are appropriately set; otherwise, their values remain unchanged and a "DATA OUT OF RANGE" message is displayed on the scratch pad.

LSK-L2: The "FUEL" parameter cannot be modified. A "DEAD KEY ERROR" message will be displayed on the scratch pad.

LSK-L3: A data entry was made for "ZFW", zero fuel weight. A new gross weight is computed as the new zero fuel weight plus the filtered total fuel quantity. The same test required when entering gross weight (see LSKL1 above) is performed with similar actions taken on the results.

LSK-L4: No parameter. A "DEAD KEY ERROR" message will be displayed on the scratch pad.

LSK-L5: The cost index to be used by the PGA-4D algorithm is entered here. It must be a number between 0 and 200 inclusively in order for the global variable CINDEX to be set. If it is not in this range, a 'DATA OUT OF RANGE' message is is displayed on the scratch pad. (note that the PGA-4D program is currently not implemented)

LSK-L6: The "<INDEX" prompt was pressed. REPROG is called to save the data before switching to the INIT/REF INDEX CDU page.

LSK-R1: CRZALT is entered at this LSK. It can be entered in a number of formats:

1) 0000 <= data <= 0999 (4 chars);
CRZALT is displayed as a number between 0 and 999.

2) 1000 <= data <= 18000;
CRZALT is displayed as entered.

3) 1 <= data <= 400 (1 to 3 chars);
CRZALT is displayed as (data * 100).

4) 18000 <= data <= 40000;
CRZALT is displayed as a flight level.
(e.g. if data = 32000, it will be displayed as FL320)

If data < 0, data > 400 (3 chars), or data > 40000, a 'DATA OUT OF RANGE' message will be displayed on the scratch pad. In the valid formats listed above, the CRZALT will be assigned to all cruise waypoints in the current active flight plan.

The CRZALT value is determined by calling the ALTX function. ALTX decodes the input data using the rules of the formats above.

LSK-R2: Although the "OPT ALT" parameter is shown at this LSK, it is not currently implemented. No data will be accepted. A "DEAD KEY ERROR" error message will be displayed on the scratch pad.

LSK-R3: The "BARSET" barometric setting is accepted here. It has a default value of 29.92.

LSK-R4: No parameter. A "DEAD KEY ERROR" message will be displayed on the scratch pad.

LSK-R5: The "IASREF" parameter is entered at this LSK. It has a default value of 130.

LSK-R6: The "TAKEOFF>" prompt was pressed. REPROG is called to save the prompt data before switching to the TAKEOFF page.

If the data on the scratch pad is DELETE and the LSK selected normally accepts data, an "INVALID DELETE" message is displayed on the scratch pad. Also note that if any entry cannot be successfully converted from character to floating point format, a "BAD DATA FORMAT" message will be output.

GLOBAL REFERENCES:

VARIABLES
BARSET* CINDEX* CRZALT ERCODE* FLKEY* FTFQ GRWGT IASREF* INDAT PGRQST* ZFW

ARRAYS
ENTRY

FUNCTIONS AND SUBROUTINES
ALTX DEL_IN FUEL_LIM NEWCRZ OTS$CVT_T_F PROG_LN REPROG

MODULE NAME: FUEL_LIM
FILE NAME: PFINIT.FOR
PROCESS: SLOW
CALLED BY: PFINP
CALLING SEQUENCE: CALL FUEL_LIM(GW, Z, GRWGT, ZFW, ERCODE)

PURPOSE:
This routine is called by PFINP to make sure that the aircraft gross weight (GW) and zero fuel weight (Z) inputs are within specific limits.

DESCRIPTION:
The tests for the GW and Z inputs are:

74000 <= GW <= 102000

&

65000 <= Z <= 77600.

If these tests are passed, the GRWGT and ZFW input parameters are set to GW and Z respectively. The global booleans GWSET and GWRESET are also set. If either of the tests fail, ERCODE is set to reflect that the data is out of range.

GLOBAL REFERENCES:

VARIABLES
GWRESET* GWSET*

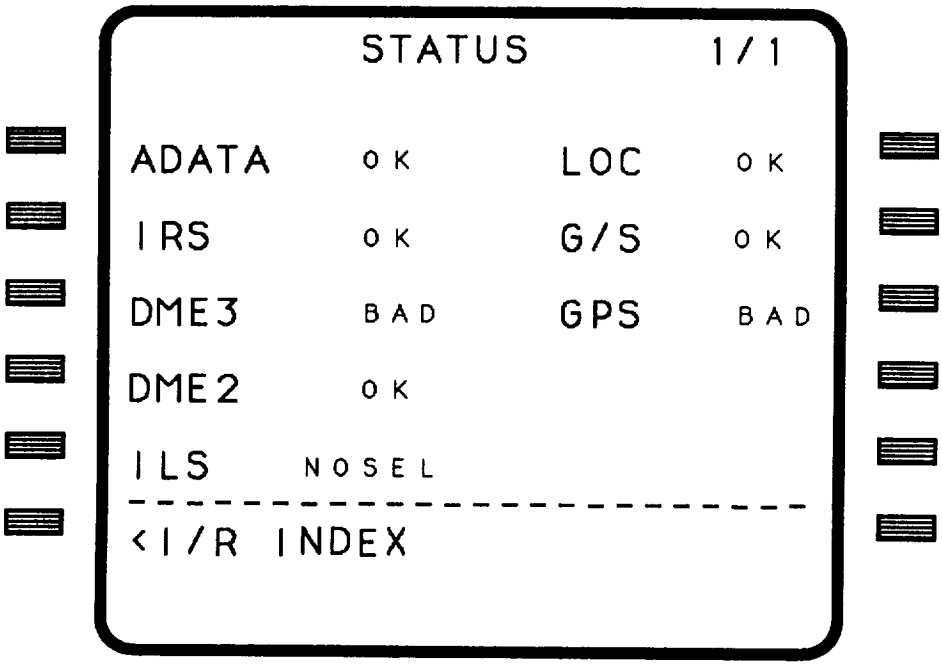
Section 7.8 THE STATUS PAGE

The STATUS page of the CDU displays the status of various onboard systems. It is accessed by pressing the "STATUS>" prompt on the INIT/REF INDEX page. It is exited by pressing the "<INDEX" prompt on the STATUS page, or by selecting a different CDU page from the CDU panel.

The status of six of the seven systems being monitored will be denoted as "OK" when functioning properly, or "BAD" when a failure has been detected. The seventh system denotes whether the localizer frequency has been selected "SEL" or not selected "NOSEL". These statuses reflect the value of associated global variables called valids. The systems and their valids are:

SYSTEM	VALID
AIR DATA SYSTEM	ADVAL
INERTIAL REFERENCE	INAVV
CROSS STATION DME	DME3VD
PATH STATION DME	DME2VD
LOCALIZER FREQUENCY SELECTED	LOCFS
LOCALIZER	LOCVLD
GLIDESLOPE	GSVLD

The STATUS page is for display purposes only. Valids can not be modified via this page. Refer to figure 7.8 on the following page.



The Status Page

(figure 7.8)

PRECEDING PAGE BLANK NOT FILMED

PAGE 326 INTENTIONALLY BLANK

MODULE NAME: STATPG
FILE NAME: STATPG.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL STATPG

PURPOSE:
The purpose of this CDU module is to display the status of various onboard systems.

DESCRIPTION:
This module is called by the CDU executive CDUEXC. The CDU page it manages is accessed by pressing the "STATUS>" prompt on the INIT/REF INDEX page. It is exited by pressing the "<INDEX" prompt on the STATUS page, or by pressing any valid function key.

When a CDU input has been detected, the STATPG input processing routine STNDRD_INP is called. The "<INDEX" prompt is the only active line select key.

STATPG is called once per SLOW cycle. One CDU line is output per STATPG call. Actual CDU output is handled via calls to the FMTOUT routine.

GLOBAL REFERENCES:

VARIABLES
ADVAL DME2VD DME3VD GPNAVV GSVLD INAVV LOCFS LOCVLD PGINIT*

ARRAYS
ENTRY

FUNCTIONS AND SUBROUTINES
FMTOUT STNDRD_INP

PAGE 328 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: STNDRD_INP
FILE NAME: STATPG.FOR
PROCESS: SLOW
CALLED BY: STATPG.FOR
CALLING SEQUENCE: CALL STNDRD_INP

PURPOSE:
The purpose of this module is to process inputs for CDU pages in which only the "<INDEX" prompt is active. If any other line select key is pressed, a "DEAD KEY ERROR" message is output. If data was entered on the scratch pad prior to selecting "<INDEX", REPROG is called to save the data before the CDU page request is processed.

VARIABLES
ERCODE* PGRQST*

ARRAYS
ENTRY

FUNCTIONS AND SUBROUTINES
REPROG

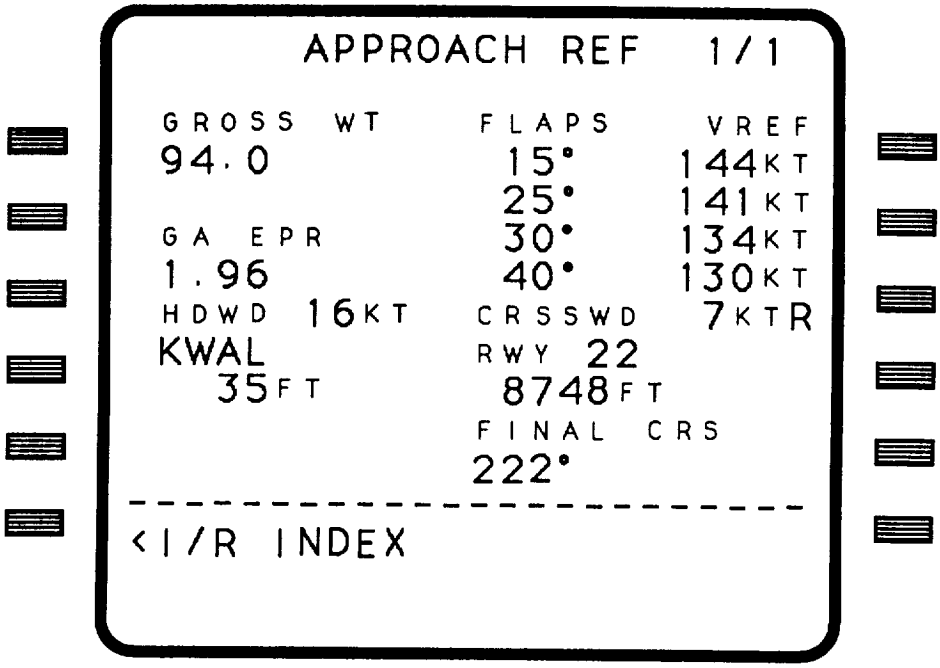
Section 7.9 THE APPROACH REFERENCE PAGE

The APPROACH REFERENCE page of the CDU displays information useful in TSRV approaches. It is accessed by pressing the "APPROACH>" prompt on the INIT/REF INDEX page. It is exited by pressing the "<INDEX" prompt on the APPROACH REFERENCE page, or by selecting a different CDU page from the CDU panel. See figure 7.9 on the following page.

The approach reference parameters and their units that are displayed on this page include:

PARAMETER	UNITS
AIRCRAFT GROSS WEIGHT	LBS*1000
REF AIRSPEEDS	KTS
GO-AROUND EPR	-
HEADWIND SPEED	KTS
CROSSWIND SPEED	KTS
CROSSWIND DIRECTION	-
DESTINATION AIRPORT NAME	-
DESTINATION RUNWAY NUMBER	-
DESTINATION RUNWAY ALTITUDE	FT
DESTINATION RUNWAY LENGTH	FT
ILS FREQUENCY	-
DESTINATION RUNWAY FINAL COURSE	DEG

The parameters are updated once every 15 SLOW task cycles. The APPROACH REFERENCE page is for display purposes only. Parameters cannot be modified via this page.



The Approach Page

(figure 7.9)

PAGE 332 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: APPREF
FILE NAME: APPREF.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL APPREF

PURPOSE:

The purpose of this CDU module is to display various approach reference parameters.

DESCRIPTION:

This module is called by the CDU executive CDUEXC. The CDU page it manages is accessed by pressing the "APPROACH>" prompt on the INIT/REF INDEX page. It is exited by pressing the "<INDEX" prompt on the APPROACH REFERENCE page, or by hitting any valid function key.

Most of the parameters displayed on this page represent global variables; however, the reference airspeeds must be computed. This is accomplished by the VREFLU routine which is called each time APPREF is called. The destination information (airport name, runway number, altitude, length, and final course direction) will not be displayed until a destination runway has been entered into the active path via the ROUTE page. Also the ILS frequency will only be displayed if an ILS is available for the selected runway.

When a CDU input has been detected, a standard input processing routine STNDRD_INP is called. The "<INDEX" prompt is the only active line select key.

APPREF is called once per SLOW cycle. One CDU line is output per APPREF call. Actual CDU output is handled via calls to the FMTOUT routine.

GLOBAL REFERENCES:

VARIABLES
ACTCNT GAEPR HDGTRU PGINIT* WD WEIGHT WS

ARRAYS
AIRPTS ENTRY

FUNCTIONS AND SUBROUTINES
FMTOUT FRMFRQ GET_CHAR GET_REAL GET_WORD OTS$CVT_L_TI
OTS$FLOAT SCOSD STNDRD_INP VREFLU

PAGE 334 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: VREFLU
FILE NAME: APPREF.FOR
PROCESS: SLOW
CALLED BY: APPREF
CALLING SEQUENCE: CALL VREFLU(GW, V15, V25, V30, V40)

PURPOSE:
This routine is called to compute VREF reference airspeeds for TSRV approaches.

DESCRIPTION:
VREF speeds are looked up in a local table and interpolated according to aircraft gross weight for flap positions of 15, 25, 30, and 40 degrees.

GLOBAL REFERENCES: none

Section 7.10 THE TAKEOFF REF PAGE

There are two CDU TAKEOFF REF pages. The first page is used to enter, compute, and display takeoff reference parameters. The second page handles the setup and execution of the Takeoff Performance Monitoring System (TOPMS). See figures 7.10 and 7.11.

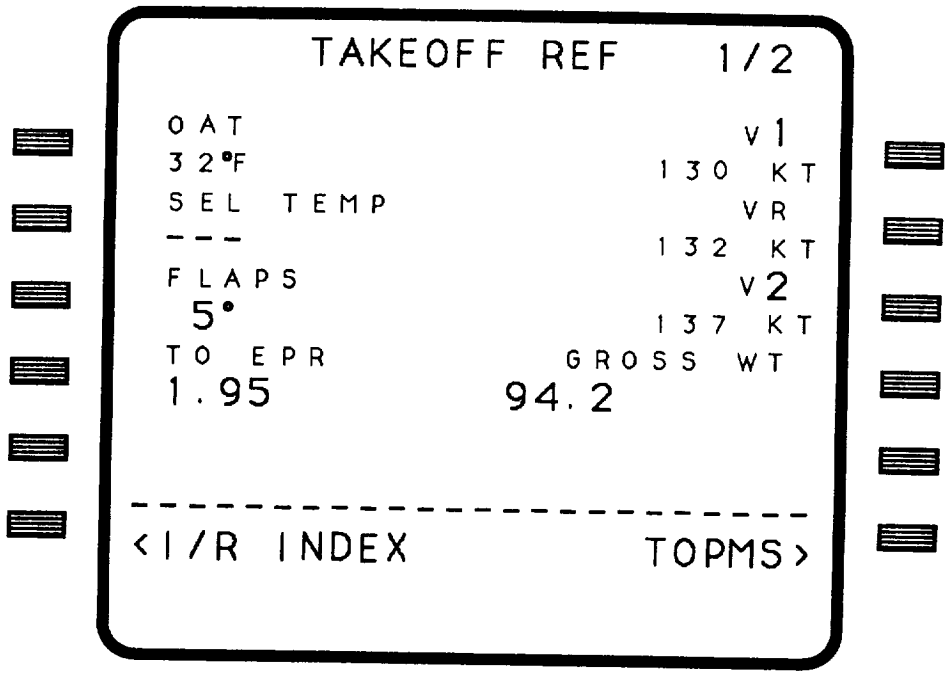
The first page is accessed by pressing the "TAKEOFF>" prompt on the PERFORMANCE/INITIALIZATION page or the "<TAKEOFF" prompt on the INIT/REF INDEX page and is exited by pressing its "<I/R INDEX" prompt or by selecting a different CDU page via the CDU function keys. The second page, or TOPMS page, is accessed by hitting the "TOPMS>" prompt on the first takeoff page and is exited by selecting a different CDU page via the CDU function keys. Also, the CDU NEXT and PREV keys can be used to toggle between the two pages.

The following lists contain the parameters, associated global variables, and units displayed on the two pages:

TAKEOFF REF PAGE 1	GLOBAL	UNITS
OUTSIDE AIR TEMPERATURE	TAT	DEG C
SELECTED OUTSIDE AIR TEMPERATURE	SOAT	DEG C
TAKEOFF FLAPS POSITION	TOFLPS	DEG
TAKEOFF EPR	TOEPR	-
AIRCRAFT GROSS WEIGHT	WEIGHT	LBS*1000
TAKEOFF REFERENCE V-SPEEDS	V1, V2, VR	KTS

TAKEOFF REF PAGE 2	GLOBAL	UNITS
AIRCRAFT CENTER OF GRAVITY	CG	-
RUNWAY FRICTION COEFFICIENT	MURWY	-
TAKEOFF WIND SPEED	TOWS	KTS
TAKEOFF WIND DIRECTION	TOWD	DEG
RUNWAY OFFSET	TOPOS	FT
RUNWAY LENGTH	TKFLEN	FT

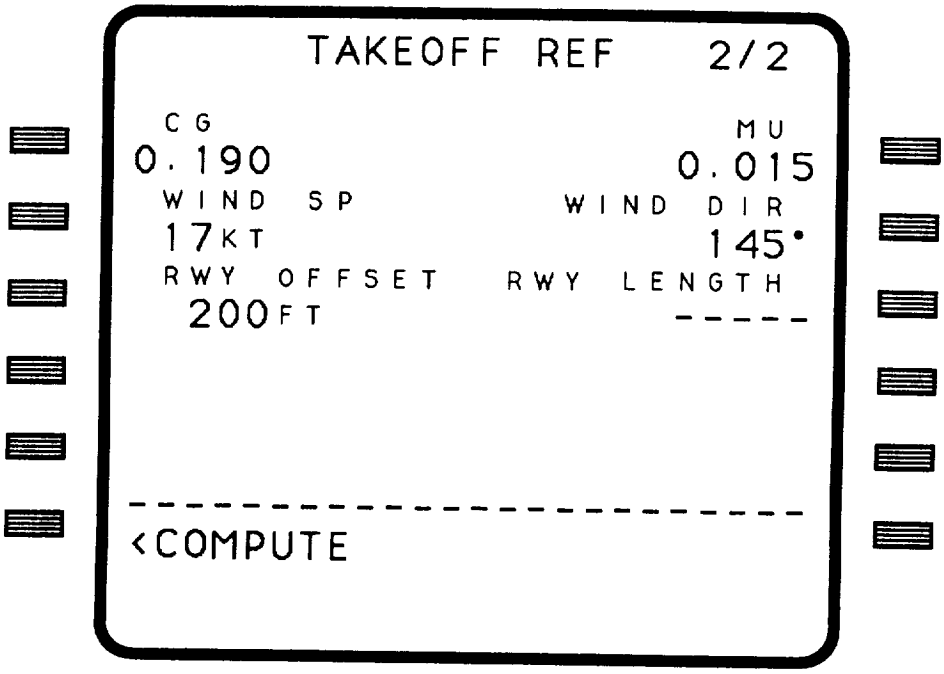
The display of parameters listed above are updated once every 15 SLOW task cycles.



The Takeoff Page (#1)

(figure 7.10)

PAGE 338 INTENTIONALLY BLANK



The Takeoff Page (#2)

(figure 7.11)

PAGE 340 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: TKOFF
FILE NAME: TKOFF.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL TKOFF

PURPOSE:
This module manages the two CDU TAKEOFF REF pages.

DESCRIPTION:
This module is called by the CDU executive CDUEXC when the user has selected the TAKEOFF REF page. There are two TAKEOFF REF pages. This module is responsible for displaying the takeoff reference and TOPMS parameters on those two pages, as well as accepting user inputs. When an input has been detected TKOFF calls TKOFFINP (see documentation below) which handles input processing.

TKOFF is called once per SLOW cycle. One CDU line is output per TKOFF call. Actual CDU output is handled via calls to the FMTOUT routine.

TAKEOFF REF PAGE 1

Various parameters relative to aircraft takeoff are displayed on this page including:

OAT: The outside air temperature in degrees Fahrenheit is always displayed and cannot be modified.

SEL TEMP: The user may enter a selected outside air temperature (SOAT) in degrees Fahrenheit which will be used in takeoff V-speed and EPR computations in place of the default, OAT. Whenever a SOAT is entered, a "DERATED TAKEOFF" message will be displayed on the CDU.

FLAPS: The takoff flaps (TOFLPS) has a default value of 5 degrees and can be modified.

TO EPR, V1, VR, V2: Takeoff EPR (TOEPR) and V-speeds are computed by the EPRTO and MANUAL routines. Aircraft gross weight must be entered on the PERFORMANCE / INITIALIZATION page before these routines can be called. The parameters are not computed every TKOFF call. EPRTO and MANUAL are only called to update the parameters when:

1) the aircraft gross weight GRWGT has been entered on the PERFORMANCE INITIALIZATION page,

PAGE 342 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

2) a SOAT is entered on this page,

3) a SOAT is deleted on this page (in which case the default OAT is used), or

4) takeoff flaps TOFLPS is entered on this page.

V-speeds can also be manually entered and will appear in small font. Manually entered V-speeds will override those computed by MANUAL; however, if one of the four items above occurs, new V-speeds will be computed by MANUAL and displayed. Computed V-speeds are displayed in large font.

GROSS WT: Aircraft gross weight is displayed, but can not be modified on this page.

For more details on these parameters, see the documentation for the TKOFFINP routine below.

TAKEOFF REF PAGE 2

This page is used to initialize and compute TOPMS parameters and enable the TOPMS display on the Navigation Display (ND). The parameters displayed on this page are:

CG: Aircraft Center of Gravity (CG) is a user-defined parameter.

WIND SP: Takeoff Wind Speed (TOWS) is a user-defined parameter.

WIND DIR: Takeoff Wind Direction (TOWD) is a user-defined parameter.

MU: Runway friction coefficient (MURWY) is a user-defined parameter with a default value of 0.015.

RWY OFFSET: Runway offset (TOPOS) is a user-defined parameter with default value of 200 feet.

RWY LENGTH: Runway length (TKFLEN) is a user-defined parameter.

For more details on inputting these parameters and setting up for a TOPMS run, see the documentation for the TKOFFINP routine below.

GLOBAL REFERENCES:

VARIABLES
ABLOFF ABROFF CG COMPFL* DISPST ENAFLG GWRESET* GWSET
HBARO MURWY NAV64K PGINIT* SOAT SOATFL TAT TKFLEN TOEPR
TOFLPS TOINDX TOPFLG TOPOS TOWD TOWS V1 V1FLAG V2 V2FLAG
VR VRFLAG WEIGHT

ARRAYS
AIRPTS BOXES DASHES ENTRY*

FUNCTIONS AND SUBROUTINES
CDU_SMALL EPRTO FMTOUT MANUAL OTS$CVT_L_TI OTS$FLOAT
TKOFFINP

MODULE NAME: TKOFFINP
FILE NAME: TKOFF.FOR
PROCESS: SLOW
CALLED BY: TKOFF
CALLING SEQUENCE: CALL TKOFFINP

PURPOSE:
TKOFFINP handles user inputs for the two CDU TAKEOFF REF pages.

DESCRIPTION:
This module is called from the TKOFF routine whenever a user makes an input from one of the CDU TAKEOFF REF pages.

TAKEOFF REF PAGE 1 INPUTS

When there is no data on the scratch pad, the only Line Select Keys (LSK) that can be legally pressed are the "<I/R INDEX" prompt (LSKL6) and "TOPMS>" prompt (LSKR6). Hitting any other LSKs will result in a "NO DATA" or "DEAD KEY ERROR" error message.

If there is data (except "DELETE") on the scratch pad when an LSK is pressed, the following applies:

LSK-L1: The outside air temperature cannot be modified. A "DEAD KEY ERROR" error message will be displayed.

LSK-L2: SOAT is entered in degrees Fahrenheit using this LSK. Its value will be converted and stored as degrees Centigrade. A "DERATED TAKEOFF" message will be displayed near the bottom of the page.

LSK-L3: Takeoff flaps TOFLPS are entered with this LSK. Only values of 1, 5, and 15 degrees will be accepted. Inputting any other value results in the output of an 'ILLEGAL ASSIGNMENT' error message.

LSK-L4: The takeoff EPR is computed by the EPRTO routine and cannot be manually modified. A "DEAD KEY ERROR" error message will be displayed.

LSK-L5: No parameter. A "DEAD KEY ERROR" error message will be displayed.

LSK-L6: REPROG is called to save the data before the CDU switches to the I/R INDEX page.

LSK-R1: V1 can be manually entered and will override the use and display of the V1 computed by the MANUAL routine. It will be displayed in small font. Before being accepted, the following test must be passed:

100kt <= new V1 <= VR.

If the test fails, the value is rejected and an "ENTRY OUT OF RANGE" error message is displayed.

LSK-R2: VR can be manually entered and will override the use and display of the VR computed by the MANUAL routine. It will be displayed in small font. Before being accepted, the following test must be passed:

V1 <= new VR <= 161kt.

If the test fails, the value is rejected and an "ENTRY OUT OF RANGE" error message is displayed.

LSK-R3: V2 can be manually entered and will override the use and display of the V2 computed by the MANUAL routine. It will be displayed in small font. Before being accepted, the following test must be passed:

VR <= new V2 <= 163kt.

If the test fails, the value is rejected and an "ENTRY OUT OF RANGE" error message is displayed.

LSK-R4: The aircraft gross weight cannot be modified on this page. A "DEAD KEY ERROR" error message will be displayed.

LSK-R5: No parameter. A "DEAD KEY ERROR" error message will be displayed.

LSK-R6: This is the "TOPMS>" prompt. Data is not allowed. A "DEAD KEY ERROR" error message will be displayed.

When TKOFFINP detects a deletion (data input is "DELETE") while this page is active, the PROC_DEL routine (see documentation below) is called to handle it. Only the SOAT and V-speeds can be deleted.

TAKEOFF REF PAGE 2 INPUTS

This page is used to setup and enable a TOPMS run. The following list contains the details for initializing the TOPMS parameters:

LSK-L1: Aircraft center of gravity CG is entered at this LSK. Its value must be greater than 0.0 and less than or equal to 0.05 to be accepted. If it is not within this range, the entry will be rejected and an "ENTRY OUT OF RANGE" message will be displayed.

LSK-L2: Takeoff wind speed TOWD is entered here. If the entry is not between 0.0 and 50.0 inclusive, it will rejected and an "ENTRY OUT OF RANGE" message will be output.

LSK-L3: Takeoff runway offset TOPOS is entered here. TOPOS has a default value of 200 feet.

LSK-R1: The runway friction coefficient MURWY has a range of 0.005 to 0.04. Any entry outside this range will result in an "ENTRY OUT OF RANGE" message.

LSK-R2: Takeoff wind direction TOWD must be a number between 0.0 and 360.0 inclusive or an 'ENTRY OUT OF RANGE' message will be output.

LSK-R3: The runway length TKFLEN can be entered at this LSK.

Pressing any of the above LSKs with no data on the scratch pad will result in a "NO DATA" error message. All other LSKs on this page are dead keys except for LSKL6 and LSKR6 which only become active when all the TOPMS parameters have been properly initialized (see next section).

No deletions are allowed on this page. Attempts at deleting any parameters will result in the output of an "INVALID DELETE" message.

For both pages, note that if any data entry cannot be successfully converted from character to floating point format, a "BAD DATA FORMAT" message will appear.

ENABLING TOPMS

Some preconditions must be met before the TOPMS computations can be performed. TOWS, TOWD, and CG must be entered, the aircraft ground speed must be less than 64 knots, and the aircraft gross weight must have been entered on the PERFORMANCE INITIALIZATION PAGE. In addition, an origin runway must have been entered on the ROUTE page. If the runway has not been entered and the other preconditions have been met, an "ENTER ORIGIN RUNWAY" message will appear on the CDU in reverse video. When the runway has been entered and the preconditions met, a "<COMPUTE" prompt will appear at LSKL6.

Pressing the "<COMPUTE" prompt causes the routine TOSTBP to compute the takeoff stabilizer position TOSTAB. The takeoff index variable TOINDX will be set to 1 allowing the TOPMS software in the Displays VAX to run. An "<ENABLE" prompt will then replace the "<COMPUTE" prompt and a "REJECT>" prompt will also appear alongside.

Pressing the "<ENABLE" prompt causes it to disappear. TOINDX will be set to 2, enabling the TOPMS display format to replace the current format on the ND.

Pressing the "REJECT>" prompt causes TOINDX to be reset to zero and the "<COMPUTE" prompt to reappear. At this time, parameters on either TAKEOFF REF pages can be altered for the next TOPMS computations, if so desired.

After completing a takeoff attempt, CDUFST reinitializes SOAT and TOINDX. When this is detected by the TKOFF module, the SOAT and V-speeds on the CDU are replaced with dashes. At this time, the takeoff parameters can be appropriately setup for the next takeoff attempt.

GLOBAL REFERENCES:

VARIABLES
CG* COMPFL* DISPST ENAFLG ERCODE* INDAT MURWY* NAV64K
PGRQST* SOAT* SOATFL* TKFLEN* TOFLPS* TOINDX* TOPFLG TOPOS*
TOSTAB TOWD* TOWS* V1 V1FLAG* V2* V2FLAG* VR VRFLAG*

ARRAYS
ENTRY

FUNCTIONS AND SUBROUTINES
DEL_IN OTS$CVT_TI_L OTS$CVT_T_F PROC_DEL REPROG TOSTBP

MODULE NAME: PROC_DEL
FILE NAME: TKOFF.FOR
PROCESS: SLOW
CALLED BY: TKOFFINP
CALLING SEQUENCE: CALL PROC_DEL(LSK)

PURPOSE:
This routine processes "DELETE" data entries.

DESCRIPTION:
This routine is called by TKOFFINP whenever a "DELETE" is attempted. The action taken is dependent upon the value of the LSK parameter and the page number. Attempting to delete a TOPMS parameter on page 2 will result in an "INVALID DELETE" error message.

On the first TAKEOFF REF page, deleting the SOAT parameter will cause TKOFF to call MANUAL to recompute the V-speeds with the OAT (TAT) as the temperature parameter. Any user-entered V-speeds will be replaced with the new computed V-speeds. Deleting the SOAT when one has not been entered will result in an "INVALID DELETE" error. Deleting a user-entered V-speed causes the TKOFF routine to call MANUAL to recompute the V-speed. The new computed V-speed will be subsequently used and displayed. Deleting a computed V-speed will result in an "INVALID DELETE" error. Also, deleting the SOAT or V-speeds will cause TKOFF to call EPRTO to recompute the takeoff EPR (TOEPR).

On page 1, it is legal to have "DELETE" on the scratch pad when switching to the I/R INDEX page via LSKL6. REPROG will be called to save the data prior to the switch. Attempts to delete any other paramters on this page will result in a "DEAD KEY" error.

GLOBAL REFERENCES:

VARIABLES
COMPFL* ERCODE* PGRQST* SOATFL* TOPFLG V1FLAG* V2FLAG*
VRFLAG*

FUNCTIONS AND SUBROUTINES
REPROG

MODULE NAME: MANUAL
FILE NAME: TKOFF.FOR
PROCESS: SLOW
CALLED BY: TKOFF
CALLING SEQUENCE: CALL MANUAL(PALT, TEMP, FLAPS, W,
V1, VR, V2)

PURPOSE:

The purpose of this routine is to compute the takeoff V-speeds V1, VR, and V2, given pressure altitude (PALT), ambient air temperature (TEMP), takeoff flaps position (FLAPS), and aircraft gross weight (W).

DESCRIPTION:

This routine is called by TKOFF whenever new computed V-speeds are required. Conditions for calling this routine are described in the description for the module TKOFF. The V-speeds are basically derived through the interpolation of a table look-up, given the values of the input parameters.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
INTRP

MODULE NAME: INTRP
FILE NAME: TKOFF.FOR
PROCESS: SLOW
CALLED BY: MANUAL
CALLING SEQUENCE: INTRP(A, X, Y)

PURPOSE:
This function returns an interpolation of the given inputs A, X, and Y.

DESCRIPTION:
The intepolation function is:

```
INTRP = FLOAT(INT(X - A * (X - Y) + 0.5))
```

GLOBAL REFERENCES: none

MODULE NAME: EPRTO
FILE NAME: TKOFF.FOR
PROCESS: SLOW
CALLED BY: TKOFF
CALLING SEQUENCE: CALL EPRTO(TEMP, PALT, ABOFF)

PURPOSE:

The purpose of this function is to return a takeoff EPR value.

DESCRIPTION:

This function is called by TKOFF. The EPR computed by this function is one recommended by the B-737 flight manual given the ambient air temperature (TEMP), pressure altitude (PALT), and airbleed status (ABOFF). It is based on the tabulation on page 3 (4B-1) of the 737-1LT flight manual dated January 5, 1970.

MODULE NAME: TOSTBP
FILE NAME: TKOFF.FOR
PROCESS: SLOW
CALLED BY: TKOFF
CALLING SEQUENCE: CALL TOSTBP(CG, SP)

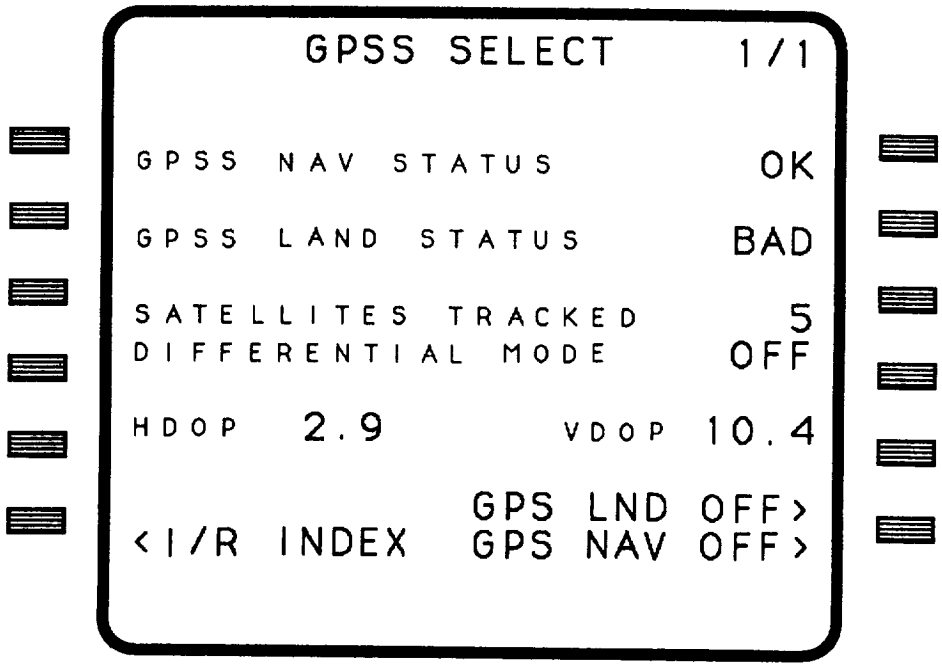
PURPOSE:
This routine computes the nominal takeoff stabilizer position.

DESCRIPTION:
This routine is called by TKOFF. The takeoff stabilizer position is computed according to the aircraft center of gravity (CG).

GLOBAL REFERENCES: none

Section 7.11 THE GPSS PAGE

This CDU page provides information about the Global Positioning Satellite System (GPSS) to the flight crew. Most of the page consists of status information, however GPS navigation may be selected or deselected on the page. See figure 7.12 on the following page.



The GPSS Page

(figure 7.12)

PAGE 356 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: GPSPG
FILE NAME: GPSPG.FOR
PROCESS: SLOW
CALLED BY: SELECT
CALLING SEQUENCE: CALL GPSPG

PURPOSE:
To generate the GPS page of the CDU.

DESCRIPTION:
This CDU page provides information about the Global Positioning Satelite System (GPSS) to the flight crew. The page is accessed via the INIT/REF index page. This module serves as the entry point to the GPS page software. CDU keyboard entries are handled directly, while output data generation is accomplished through calls to SHOW_GPS.

Only function entries are valid inputs on the GPS page. Of the 12 line select keys, only LSK-L6 and LSK-R6 are enabled. Selection and deselection of GPS navigation is made through LSK-R6. Return to the INIT/REF index page is made using LSK-L6.

Updating of the entire GPS display is performed in eight consecutive calls to GPSPG. The module SHOW_GPS is called with an index indicating the current place in the update cycle.

GLOBAL REFERENCES:

VARIABLES
ERCODE* GPLND* GPNAV PGINIT* PGRQST*

ARRAYS
ENTRY*

FUNCTIONS AND SUBROUTINES
SHOW_GPS

PAGE 358 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: SHOW_GPS
FILE NAME: GPSPG.FOR
PROCESS: SLOW
CALLED BY: GPSPG
CALLING SEQUENCE: CALL SHOW_GPS(PASS)

PURPOSE:
To display lines of text on the CDU GPSS page.

DESCRIPTION:
This procedure updates one of eight CDU display lines each time it is called. See figure 7.11.1 for the format of the GPSS page. The calling parameter contains the index (0-7) of which line to update. The following chart shows the data formatted for the various index values.

0 Title line
1 GPSS navigation status line (uses GPNAVV)
2 GPSS land status line (uses GPLNDV)
3 satellites tracked line (uses SATINVW)
4 differential mode line (uses DIFMOD)
5 VDOP and HDOP line (uses GPHDOP and GPVDOP)
6 GPS land select line (uses GPNAV)
7 GPS NAV select line (uses GPNAV)

Note that the values of HDOP and VDOP are dashed out if GPS navigation is not valid (GPNAVV).

GLOBAL REFERENCES:

VARIABLES
DIFMOD GPHDOP GPLNDV GPNAV GPNAVV GPVDOP SATINVW

FUNCTIONS AND SUBROUTINES
FMTOUT OTS$CVT_L_TU OTS$FLOAT

Section 8.0 THE PHASE OF FLIGHT PAGES

There are three CDU phase of flight pages: CLIMB, CRUISE, and DESCENT. Each page is used for displaying and altering parameters useful in its associated flight segment. See figures 8.0, 8.1, and 8.2 for typical examples of these pages.

Each page has an associated CDU key for access. Automatic switching from one flight page to another will occur under the appropriate conditions described in later sections. Exiting a page can either be accomplished by pressing its "<LEGS" page prompt or by selecting a different CDU page via the CDU panel.

Following are the parameters and if applicable, its associated global variable, and units for the pages:

CLIMB PAGE	GLOBAL	UNITS
CRUISE ALTITUDE	CRZALT	FT
OPTIMUM ALTITUDE	OPTALT	FT
TARGET SPEED	CLB_SPD	KT
TARGET MACH	CLB_MACH	--
RESTRICTED SPEED	REST_SPD	KT
RESTRICTED ALTITUDE	REST_ALT	FT
CLIMB EPR	MCLEPR	--

CRUISE PAGE	GLOBAL	UNITS
CRUISE ALTITUDE	CRZALT	FT
OPTIMUM ALTITUDE	OPTALT	FT
TARGET MACH	CRZ_MACH	--
MAX CRUISE EPR	MCREPR	--
DIST TO TOP OF DESCENT WPT	computed locally	NM
ETA TO TOP OF DESCENT WPT	computed locally	HR:SEC

DESCENT PAGE	GLOBAL	UNITS
ACTIVE WAYPOINT NAME	WPT_ACT(TOWPT).NAME	--
ACTIVE WAYPOINT ALTITUDE	WPT_ACT(TOWPT).ALT	--
TARGET SPEED OF LAST WPT	WPT_ACT(TOWPT-1).IAS	KT
TARGET SPEED OF ACTIVE WPT	WPT_ACT(TOWPT).IAS	KT
DISTANCE TO ACTIVE WAYPOINT	DTOGO	NM
END OF DESCENT WPT ALT	WPT_ACT(MODCNT).ALT	FT
AIRCRAFT ALTITUDE	ALTCOR	FT
ALTITUDE ERROR	HER	FT

All three pages will display the following information when 4D guidance is possible (GUID4D on) and an RTA exists for a waypoint along the unflown portion of the active path:

PARAMETER	GLOBAL	UNITS
DISTANCE TO RTA WAYPOINT	computed locally	NM
RTA WAYPOINT NAME	WPT_ACT(RTA_PTR).NAME	--
RTA TIME	WPT_ACT(RTA_PTR).ETA	HR:SEC
TIME ERROR	TIMERR	HR:SEC
GROUND SPEED	GS	KT
GROUND SPEED ERROR	GSE	KT

Because the CLIMB, CRUISE, and DESCENT pages contain numerous instances where I/O was identical, the code to handle the I/O was incorporated into one routine, namely FLT_TYPE. In the few instances where I/O is different from page to page, the common variable PAGENM is used by FLT_TYPE to distinguish which page is active and take the appropriate action.

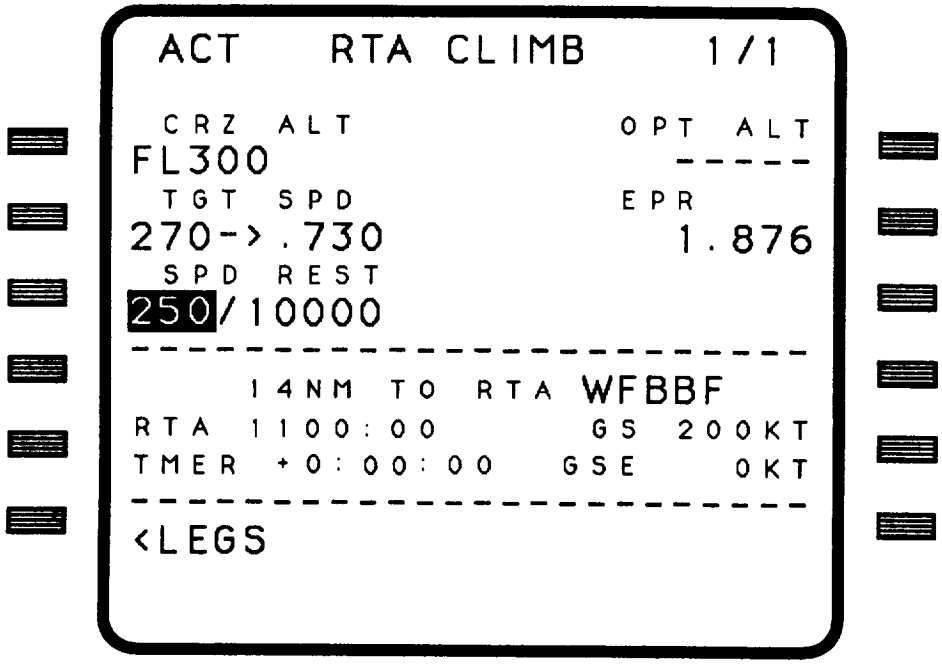
There are three main routines: CLIMB, CRUISE, and DESCENT. When called by the CDU executive, each routine sets PAGENM to a specific value and then calls FLT_TYPE.

Thus,

* if the executive calls CLIMB, CLIMB sets PAGENM to 1 and then calls FLT_TYPE

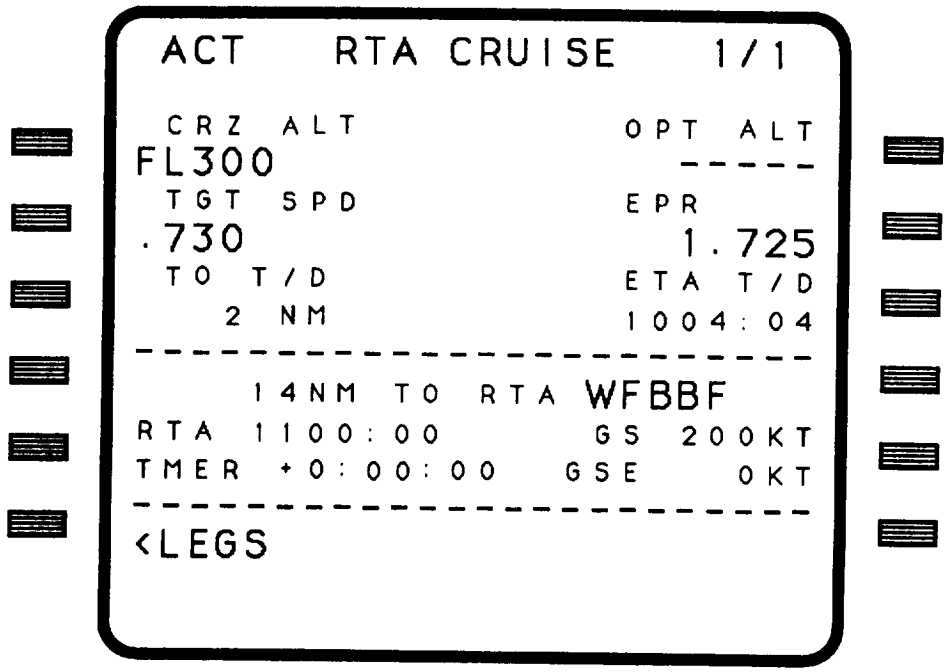
* if the executive calls CRUISE, CRUISE sets PAGENM to 2 and then calls FLT_TYPE

* if the executive calls DESCENT, DESCENT sets PAGENM to 3 and then calls FLT_TYPE



The Climb Page

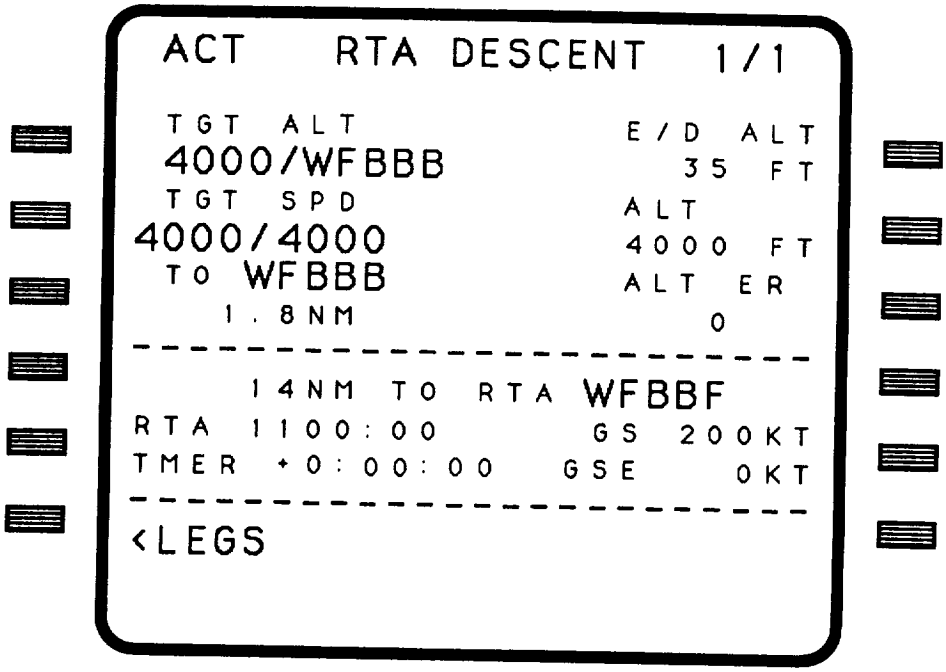
(figure 8.0)



The Cruise Page

(figure 8.1)

PAGE 364 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED



The Descent Page

(figure 8.2)

PRECEDING PAGE BLANK NOT FILMED

PAGE 366 INTENTIONALLY BLANK

MODULE NAME: CLIMB
FILE NAME: CLIMB.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL CLIMB

PURPOSE:
The CLIMB routine is responsible for setting up and calling the FLT_TYPE routine which handles all CLIMB I/O.

DESCRIPTION:
The CLIMB routine sets the common active flight page variable PAGENM to 1 and then calls FLT_TYPE. CLIMB is called once per SLOW task cycle.

GLOBAL REFERENCES:

VARIABLES
PAGENM*

FUNCTIONS AND SUBROUTINES
FLT_TYPE

PAGE 368 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: CRUISE
FILE NAME: CRUISE.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL CRUISE

PURPOSE:
The CRUISE routine is responsible for setting up and calling the FLT_TYPE routine which handles all CRUISE I/O.

DESCRIPTION:
The CRUISE routine sets the common active flight page variable PAGENM to 2 and then calls FLT_TYPE. CRUISE is called once per SLOW task cycle.

GLOBAL REFERENCES:

VARIABLES
PAGENM*

FUNCTIONS AND SUBROUTINES
FLT_TYPE

MODULE NAME: DESCENT
FILE NAME: DESCENT.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL DESCENT

PURPOSE:
The DESCENT routine is responsible for setting up and calling the FLT_TYPE routine which handles all DESCENT I/O.

DESCRIPTION:
The DESCENT routine sets the common active flight page variable PAGENM to 3 and then calls FLT_TYPE. DESCENT is called once per SLOW task cycle.

GLOBAL REFERENCES:

VARIABLES
PAGENM*

FUNCTIONS AND SUBROUTINES
FLT_TYPE

MODULE NAME: FLT_TYPE
FILE NAME: CLIMB.FOR
PROCESS: SLOW
CALLED BY: CLIMB, CRUISE, DESCENT
CALLING SEQUENCE: CALL FLT_TYPE

PURPOSE:
This routine manages the I/O for the CDU CLIMB, CRUISE, and DESCENT phase of flight pages.

DESCRIPTION:
FLT_TYPE is called by the CLIMB, CRUISE, and DESCENT CDU routines to display information relative to the current phase of flight. Most of the parameters represent global variables, but a couple are computed locally. Every time FLT_TYPE is called it calls FIND_TOD to compute the top of descent waypoint if a flight plan has been entered (GUID2D on). It also calls the SPEEDB routine to set the target speeds and compute the speedbug for the Primary Flight Display (PFD). The speedbug is also the active speed command and will be displayed in reverse video. FLT_TYPE also calls CHNG_PG to see if the conditions for an automatic phase of flight page change exist.

Many of the parameters on these pages allow for user inputs. When an input is detected, FLT_TYPE calls FLT_TYPE_INP to handle it.

The title line of each page contains the flight phase, guidance mode, and plan mode. The flight phases are of course CLIMB, CRUISE, and DESCENT. If 4D guidance is active (GUID4D on), then the title will contain "RTA"; otherwise it will contain "ECON". When the current plan mode is "active" (PMODE > 1), the string "ACT" will also be displayed. For example, if PMODE = 2, GUID4D is set, and the CLIMB page is active, the title will read "ACT RTA CLIMB".

One CDU line is output per FLT_TYPE call. Actual CDU output is handled via calls to the FMTOUT routine.

GLOBAL REFERENCES:

VARIABLES
ALTCOR CRZALT DFLKEY DTOGO FLKEY GSFPS GUID2D GUID4D HER
MCLEPR MCREPR MODCNT OPTALT PAGENM PGINIT* PMODE REST_ALT
REST_SPD REVERS RFLKEY RTA_PTR TGT_MACH TGT_SPD TIME TOWPT

ARRAYS
BOXES DASHES ENTRY*

```
RECORD ARRAYS
  WPT_ACT

FUNCTIONS AND SUBROUTINES
  CDU_SMALL CHNG_PG FIND_TOD FLT_TYPE_INP FMTOUT FMTTIM
  OTS$CVT_L_TI OTS$FLOAT PROG_LN RTA_LN10 RTA_LN8 RTA_LN9
  SPEEDB
```

MODULE NAME: FLT_TYPE_INP
FILE NAME: CLIMB.FOR
PROCESS: SLOW
CALLED BY: FLT_TYPE
CALLING SEQUENCE: CALL FLT_TYPE_INP

PURPOSE:
This routine handles all user inputs for the CLIMB, CRUISE, and DESCENT CDU pages.

DESCRIPTION:
FLT_TYPE_INP is called by FLT_TYPE whenever a user input is detected on the CDU. There are two basic types of CDU inputs handled by this module -- 1) Blank Scratch Pad + Line Select Key (LSK), and 2) Scratch Pad Data + Line Select Key.

If there is no data on the scratch pad when an LSK is pressed, the following applies:

LSK-L1: If the CLIMB or CRUISE page is active, PROG_LN is called to program the scratch pad with the value of the cruise altitude (CRZALT). If the DESCENT page is active, the 'TO' waypoint altitude (WPT_ACT(TOWPT).ALT) is put on the scratch pad.

LSK-L6: The "<LEGS" prompt was pressed and the LEGS page becomes the active CDU page.

OTHERS: If data was expected for the key, a "NO DATA" error message will be displayed; otherwise, a "DEAD KEY ERROR" error message will be output.

If there is data on the scratch pad prior to pressing an LSK, the following applies:

LSK-L1: Altitudes are entered with this key. For the CLIMB and CRUISE pages, the altitude is the cruise altitude (CRZALT). If executed, the value of CRZALT will be assigned to WPT_ACT().ALT for all cruise waypoints in the path through a call to the NEWCRZ routine. For the DESCENT page, the entered altitude will be assigned to WPT_ACT(TOWPT).ALT, WPT_ACT(TOWPT).ALTF will be set, and DEMODE is called with the AUTOEX parameter so that an automatic change to the waypoint buffer will occur. The altitudes can be entered in a number of different formats:

1) 0000 <= data <= 0999 (4 chars);
Altitude is displayed as a number between 0 and 999.

2) 1000 <= data <= 18000;
Altitude is displayed as entered.

3) 1 <= data <= 400 (1 to 3 chars);
Altitude is displayed as (data * 100).

4) 18000 <= data <= 40000;
Altitude is displayed as a flight level. (e.g. if data = 32000, it will be displayed as FL320)

If data < 0, data > 400 (3 chars), or data > 40000, a "DATA OUT OF RANGE" message will be displayed on the scratch pad.

The altitude value is determined by calling the ALTX function. ALTX decodes the input data using the format rules above.

LSK-L3: This key is used to input the restricted airspeed and altitude on the CLIMB page. It is a dead key on the CRUISE and DESCENT pages. Valid entries can be made as follows:

1) if 150 kts <= data <= 350 kts, assign the value to REST_SPD. The data in this case may be followed by a slash "/".

2) if the data is a slash "/" followed by a number, decode it as an altitude (regular or flight level format) and assign it to REST_ALT.

3) if the data is a number followed by a slash "/" and another number, decode and assign to REST_SPD and REST_ALT respectively.

When a restricted speed is being entered and 150 kts <= REST_SPD <= 350 kts is not true, the inputs will be rejected and a "DATA OUT OF RANGE" error message will be output. The same restrictions for entering an altitude described for LSKL1 apply here.

LSK-L6: REPROG is called to save the data before the LEGS page change.

OTHERS: Any other data entry results in a "DEAD KEY ERROR" error message.

GLOBAL REFERENCES:

VARIABLES
CRZALT DFLKEY* ERCODE* FLKEY* GUID2D INDAT PAGENM PGRQST* PMODE REST_ALT* REST_SPD* RFLKEY* TOWPT

ARRAYS
ENTRY

RECORD ARRAYS
WPT_ACT WPT_MOD*

FUNCTIONS AND SUBROUTINES
ALTX DEL_IN DEMODE NEWCRZ OTS$CVT_T_F PROG_LN REPROG

MODULE NAME: FIND_TOD
FILE NAME: CLIMB.FOR
PROCESS: SLOW
CALLED BY: FLT_TYPE
CALLING SEQUENCE: CALL FIND_TOD(TOD_INDX)

PURPOSE:

The purpose of this routine is to find the top of descent waypoint for the active flight path.

DESCRIPTION:

This routine is called by FLT_TYPE every time it is called. It searches through the active waypoint buffer WPT_ACT for the first waypoint with a descent phase. The position prior to this waypoint in the buffer is then assigned to the top of descent variable TOD_INDX.

GLOBAL REFERENCES:

VARIABLES
MODCNT

RECORD ARRAYS
WPT_ACT

MODULE NAME: CHNG_PG
FILE NAME: CLIMB.FOR
PROCESS: SLOW
CALLED BY: FLT_TYPE
CALLING SEQUENCE: CALL CHNG_PG(CRZALT, TOD_INDX)

PURPOSE:
This routine checks the conditions for automatic phase of flight page changes.

DESCRIPTION:
CHNG_PG is called by FLT_TYPE every pass when the plan mode is active (PMODE = 3). If a new cruise altitude has been entered on the CLIMB or CRUISE pages, the following checks are made:

1) if CLIMB or CRUISE page active and CRZALT <= ALTCOR - 200 feet, then make an automatic switch to the DESCENT page.

2) if CLIMB page is active and CRZALT < ALTCOR + 200 feet and CRZALT > ALTCOR - 200 feet, then make an automatic switch to the CRUISE page.

3) if the CRUISE page is active and CRZALT >= ALTCOR + 200 feet, then make an automatic switch to the CLIMB page.

In addition, a page change to the CRUISE page will occur when the aircraft is within 0.5 nm of the first cruise waypoint. Likewise, when the aircraft is within 0.5 nm of reaching the waypoint following the top of descent waypoint, a switch to the DESCENT page will occur.

GLOBAL REFERENCES:

VARIABLES
ACTCNT ALTCOR CRZALT CRZCHNG* DESCHNG* DTOGO PAGENM
PGRQST* TOWPT

RECORD ARRAYS
WPT_ACT

MODULE NAME: SPEEDB
FILE NAME: CLIMB.FOR
PROCESS: SLOW
CALLED BY: FLT_TYPE
CALLING SEQUENCE: CALL SPEEDB

PURPOSE:

The purpose of this routine is to set the target speed and mach and then set the speedbug for the Primary Flight Display (PFD).

DESCRIPTION:

FLT_TYPE calls SPEEDB every pass. If 2D guidance is not possible, the target speed and mach are set to zero; otherwise they are set as follows:

1) If the CLIMB page is active, the target speed and mach are set to the global climb variables CLB_SPD and CLB_MACH respectively.

2) If the CRUISE page is active, the target speed will be zero and the target mach will be the global cruise mach CRZ_MACH.

3) If #1 and #2 are not true and the active 'TO' waypoint speed flag (WPT_ACT(TOWPT).SPDF) is not zero, the target speed is set to the IAS of the current waypoint (WPT_ACT(TOWPT - 1).IAS) and the target mach is set to the 'TO' waypoint IAS (WPT_ACT(TOWPT).IAS).

Before the speedbug is set, the commanded airspeed COMIAS is computed as:

```
COMIAS = 661.5*SQRT(5.*(((1.+DELTA*(((1.+0.2*M*M)**3.5)
         -1))**(2./7.))-1.))
```

where DELTA is (1-6.87535E-06*HBARO)**5.2561 and M is the target mach.

The speed bug and active speed command (displayed in reverse video on the CDU) are set by the following conditions:

1) if HBARO < restricted altitude (REST_ALT), then SPDBUG = resticted airspeed (REST_SPD) and the active speed command will be REST_SPD.

2) if HBARO >= REST_ALT and the target airspeed <= COMIAS, then SPDBUG = target speed and the active speed command will be the target speed.

3) if neither #1 or #2 is true, then SPDBUG = COMIAS and the active speed command will be the target mach.

GLOBAL REFERENCES:

VARIABLES
CLB_MACH CLB_SPD CRZ_MACH GUID2D HBARO PAGENM REST_ALT
REST_SPD REVERS* SPDBUG* TGT_MACH TGT_SPD TOWPT

RECORD ARRAYS
WPT_ACT

FUNCTIONS AND SUBROUTINES
MTH$SQRT

MODULE NAME: PROG_LN
FILE NAME: CLIMB.FOR
PROCESS: SLOW
CALLED BY: FLT_TYPE_INP
CALLING SEQUENCE: CALL PROG_LN(VAR, LINE)

PURPOSE:
This routine outputs the string VAR to the CDU line whose value is LINE.

DESCRIPTION:
The CDU line is programmed starting with the first non-blank character in VAR through a call to FMTOUT.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
FMTOUT OTS$CVT_L_TI

MODULE NAME: RTA_LN8
FILE NAME: CLIMB.FOR
PROCESS: SLOW
CALLED BY: FLT_TYPE, PROGRESS
CALLING SEQUENCE: CALL RTA_LN8

PURPOSE:
This routine outputs line #8 of the CLIMB, CRUISE, and DESCENT pages.

DESCRIPTION:
If 4D guidance is possible (GUID4D on), RTA_LN8 computes and displays the distance in nautical miles to the RTA waypoint. The RTA waypoint name is also displayed on this line. If the RTA waypoint has already been passed, a blank line is output. If 4D guidance is not possible the line will read "NO RTA ASSIGNED". Actual CDU output is handled through calls to FMTOUT.

GLOBAL REFERENCES:

VARIABLES
DTOGO GUID4D RTA_PTR TOWPT

RECORD ARRAYS
WPT_ACT

FUNCTIONS AND SUBROUTINES
CDU_SMALL FMTOUT OTS$CVT_L_TI

MODULE NAME: RTA_LN9
FILE NAME: CLIMB.FOR
PROCESS: SLOW
CALLED BY: FLT_TYPE, PROGRESS
CALLING SEQUENCE: CALL RTA_LN9

PURPOSE:
This routine outputs line 9 of the CLIMB, CRUISE, and DESCENT pages.

DESCRIPTION:
If 4D guidance is possible (GUID4D on), RTA_LN9 displays the RTA input time (WPT_ACT(RTA_PTR).ETA) and the aircraft ground speed (GS); otherwise, a blank line is output. Actual CDU output is handled through calls to FMTOUT.

GLOBAL REFERENCES:

VARIABLES
GS GUID4D RTA_PTR

RECORD ARRAYS
WPT_ACT

FUNCTIONS AND SUBROUTINES
CDU_SMALL FMTOUT FMTTIM OTS$CVT_L_TI

MODULE NAME: RTA_LN10
FILE NAME: CLIMB.FOR
PROCESS: SLOW
CALLED BY: FLT_TYPE, PROGRESS
CALLING SEQUENCE: CALL RTA_LN10

PURPOSE:
This routine outputs line 10 of the CLIMB, CRUISE, and DESCENT pages.

DESCRIPTION:
If 4D guidance is possible (GUID4D on), RTA_LN10 displays the RTA time error (TIMERR) and the aircraft ground speed error (GSE); otherwise, a blank line is output. Actual CDU output is handled through calls to FMTOUT.

GLOBAL REFERENCES:

VARIABLES
GSE GUID4D TIMERR

FUNCTIONS AND SUBROUTINES
CDU_SMALL FMTOUT FMTTIM OTS$CVT_L_TI

Section 9.0 THE FIX PAGE

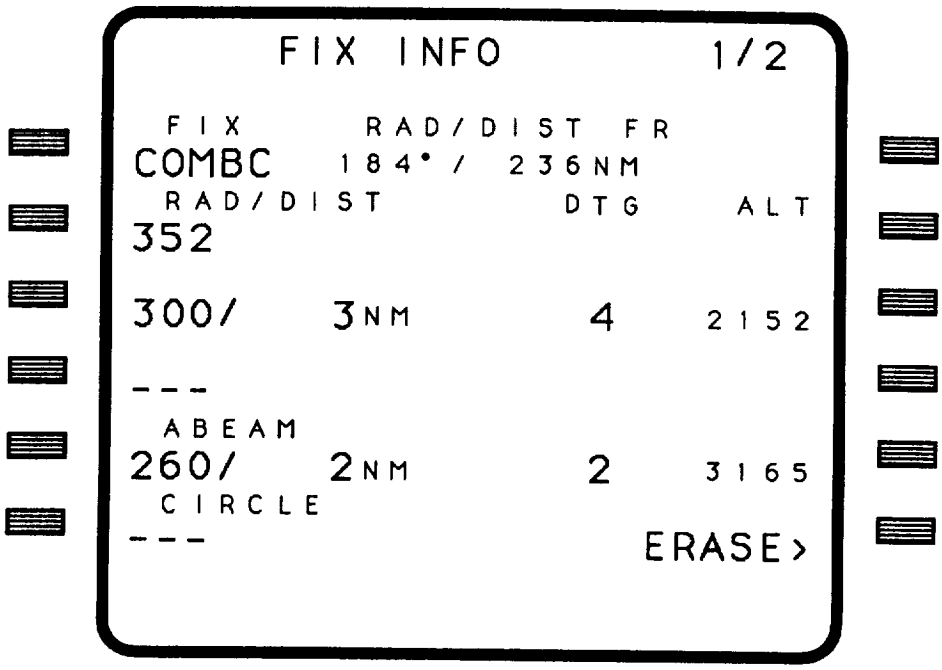
The purpose of the FIX CDU page is to provide bearing and distance information from an entered fix to the current aircraft position.

There are two fix pages with identical formats and functions. They are accessed by pressing the FIX function key and exited by selecting another CDU page via the CDU panel.

A fix may be a navaid, waypoint, or airport, as long as it exists in the aircraft's navigation database. The distance (nm) and magnetic bearing (deg) from the fix to the aircraft are computed, displayed, and continually updated once a fix has been entered. Up to three radials from each fix can be entered. The distance from the fix along the radials to the nearest intercept point on a straight leg of the unflown active path and the altitude (ft) at that intercept point are computed once at the time of the radial entry. The aircraft distance to go (nm) along the path to the intercept point is also continually updated and displayed. If a radial does not have an intercept point, these parameters are not computed and only the radial value is displayed. The frequency of the fix updates is once per 14 SLOW task cycles.

A special radial called an abeam can also be displayed. This abeam radial is one that intersects a straight leg of the unflown portion of the active path perpindicularly. If there are more than one abeam radials, the one with the shortest distance between the fix and its intercept point will be the one displayed. If no abeam radials exist for a given fix, an error message will alert the user. A reference circle can also be displayed by entering its radius (nm) on the fix page.

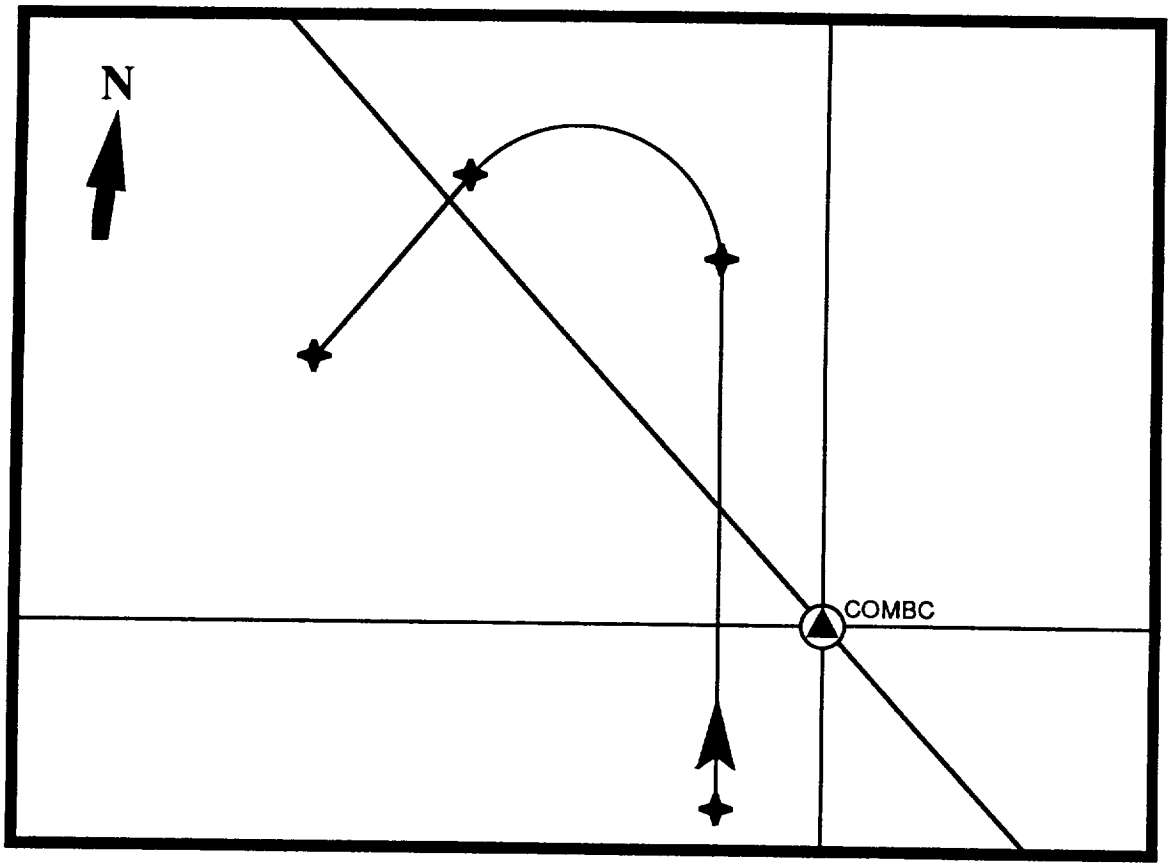
In addition to the CDU display, the Navigation Display (ND) gives the user a graphical depiction of the fix data. Fixes, radials, and reference circles will be drawn in green. Figures 9.0 and 9.1 on the following pages show a typical CDU Fix page with the corresponding Map display symbology.



The Fix Page

(figure 9.0)

PAGE 386 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED



Nav Display Fix Example

(figure 9.1)

PAGE 388 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: FIX_INFO
FILE NAME: FIX.FOR
PROCESS: SLOW
CALLED BY: CDUEXC
CALLING SEQUENCE: CALL FIX_INFO

PURPOSE:
FIX_INFO is the main FIX routine.

DESCRIPTION:
When a user has selected the CDU FIX page, the CDU executive CDUEXC calls FIX_INFO once per SLOW task cycle.
Fix information is represented in a FORTRAN structure declared in the file FIXCOM.INC. This structure contains:

FIX(PG).ADDR: AADCOM address of the fix
FIX(PG).NAME: fix name
FIX(PG).NAME_LEN: number of chars in fix name
FIX(PG).TYPE: fix type (nvd=1, wpt=2, apt=3)
FIX(PG).BRG: mag bearing from fix to aircraft (deg)
FIX(PG).CIRCLE: fix circle radius size (nm)
FIX(PG).DIST: distance from fix to aircraft (nm)
FIX(PG).LAT: fix latitude (deg)
FIX(PG).LON: fix longitude (deg)
FIX(PG).MAG: fix magnetic variation (deg)
FIX(PG).SET: boolean set when a fix has been entered
FIX(PG).RAD(I).SET: conveys a radial has been entered
FIX(PG).RAD(I).INDEX: path segment number containing intercept point (IP)
FIX(PG).RAD(I).VAL: radial value (deg)
FIX(PG).RAD(I).IP_LAT: radial IP latitude (deg)
FIX(PG).RAD(I).IP_LON: radial IP longitude (deg)
FIX(PG).RAD(I).IP_ALT: radial IP programmed alt (ft)
FIX(PG).RAD(I).IP_DIST: distance from fix to IP (nm)
FIX(PG).RAD(I).IP_DTG: aircraft distance to go along path to IP (nm)

where PG is the fix page number (1 or 2) and I is the radial number (1, 2, 3, or abeam = 4).
If an input has been made and a fix has not yet been entered on the page, the entry must be either a fix at LSKL1 or a page change request using the NEXT or PREV CDU panel keys. FIX_INP is called if a fix was entered and CH_FIX_PG is called to handle a page change. Any other inputs will result in the output of an error message.

PAGE 390 INTENTIONALLY BLANK PRECEDING PAGE BLANK NOT FILMED

After any inputs are processed, POS_INFO is called to compute the distance and magnetic bearing from the fix(es) to the current aircraft position. Next, if 2D guidance is possible (GUID2D on), the aircraft distance to go along the path to all existing radial intercept points is computed by COMP_IP_DTG. FIX_DISP is then called to set the global variables in the global section DISNAV, needed for displaying fix information on the ND.

Finally, CDU line outputs are performed. All output lines are handled in this routine except for radial output lines which are managed by calls to OUT_RAD. A different CDU line is output on each FIX_INFO call. All actual CDU output is handled through calls to FMTOUT.

Fixes are erased from the CDU and ND by pressing the "ERASE>" key or by entering "DELETE" and pressing the fix name line select key (LSKL1).

GLOBAL REFERENCES:

VARIABLES
  ERCODE* GUID2D LAT LON PG PGINIT*

ARRAYS
  BOXES DASHES ENTRY*

RECORD ARRAYS
  FIX

FUNCTIONS AND SUBROUTINES
  CDU_SMALL CH_FIX_PG COMP_IP_DTG DEL_FIX DEL_IN FIX_DISP
  FIX_INP FMTOUT OTS$CVT_L_TI OTS$FLOAT OUT_RAD POS_INFO

MODULE NAME: OUT_RAD
FILE NAME: FIX.FOR
PROCESS: SLOW
CALLED BY: FIX_INFO
CALLING SEQUENCE: CALL OUT_RAD(LINE, INDEX)

PURPOSE:

This routine is called by the main fix routine FIX_INFO to output the parameters for the four radials to the CDU.

DESCRIPTION:

If a radial has not been entered, or in the case of the computed abeam, dashes will be displayed. For entered radials (1, 2, 3), if there is no intersection of a straight leg in the path, only the radial value will be displayed. When these radials intersect a straight leg, the radial value, distance between the fix and the intercept point, distance to go along the path from the aircraft's current position to the intercept point, and the desired altitude at the intercept point will be displayed. These parameters are also displayed for the abeam radial if one has been previously computed.

The input parameter LINE is the CDU line being output and the INDEX is the radial index (1-4). All actual CDU output is handled through calls to FMTOUT.

GLOBAL REFERENCES:

VARIABLES
PG

ARRAYS
DASHES

RECORD ARRAYS
FIX

FUNCTIONS AND SUBROUTINES
CDU_SMALL FMTOUT OTS$CVT_L_TI

MODULE NAME: FIX_INP
FILE NAME: FIX.FOR
PROCESS: SLOW
CALLED BY: FIX_INFO
CALLING SEQUENCE: CALL FIX_INP(PASS)

PURPOSE:
The purpose of this routine is to determine the type of user input and call the appropriate input processing routine.

DESCRIPTION:
This routine is called by FIX_INFO any time a user input is detected. If an LSK has been pressed and there is no data on the scratch pad, FUNC_INP_FIX is called. If a deletion is attempted by the user (data is "DELETE"), DEL_FIX is called. All other data inputs will be handled by calling DATA_INP_FIX.

GLOBAL REFERENCES:

ARRAYS
ENTRY

FUNCTIONS AND SUBROUTINES
DATA_INP_FIX DEL_FIX DEL_IN FUNC_INP_FIX

MODULE NAME: FUNC_INP_FIX
FILE NAME: FIX.FOR
PROCESS: SLOW
CALLED BY: FIX_INP
CALLING SEQUENCE: CALL FUNC_INP_FIX(PASS)

PURPOSE:

The purpose of this routine is to process LSK inputs in which no data is involved.

DESCRIPTION:

When an LSK has been pressed and there is no data on the scratch pad, the following applies:

LSK-L1: If a fix name has been previously entered, it will be written to the scratch pad; otherwise, a "NO DATA" error message will be output.

LSK-L2, LSK-L3, LSK-L4: If a fix radial exists at the LSK and an intercept point was found, the latitude and longitude of the intercept point will be written to the scratch pad. If the fix radial does not have an intercept point, a "NO DATA" message is displayed. If a radial has not yet been entered, a "DEAD KEY ERROR" message is output.

LSK-L5: If 2D guidance is available (GUID2D on), then COMP_ABRAD is called to find the abeam radial. If there is no abeam radial, the message "NO ABEAM RADIAL" will be displayed. If 2D guidance is not available, the "DEAD KEY ERROR" message is output.

LSK-L6: Circle data is expected. "NO DATA" will be displayed if a fix has been previously entered and "DEAD KEY ERROR" will be output otherwise.

LSK-R6: The fix is erased from the CDU and the ND. FIX_INIT is called to reset the fix data structure.

Pressing any of the remaining LSKs will result in a "DEAD KEY ERROR" message.

GLOBAL REFERENCES:

VARIABLES
  ERCODE* GUID2D PG

ARRAYS
  ENTRY

RECORD ARRAYS
  FIX

FUNCTIONS AND SUBROUTINES
  CH_FIX_PG COMP_ABRAD FIX_INIT FMTDEG FMTOUT STRIPR

MODULE NAME: DATA_INP_FIX
FILE NAME: FIX.FOR
PROCESS: SLOW
CALLED BY: FIX_INP
CALLING SEQUENCE: CALL DATA_INP_FIX

PURPOSE:
This routine handles fix data inputs.

DESCRIPTION:
DATA_INP_FIX is called by FIX_INP whenever an attempt to input data via the LSKs is made. The following list explains what is expected for each LSK.

LSK-L1: Input should be a fix name. If the number of characters is not 3, 4, or 5, a "BAD DATA FORMAT" message is output. LUNAVA is called to look up the fix in AADCOM. If found, the .MAG, .ADDR, .NAME, .LAT, and .LON fields of the FIX structure are retrieved from AADCOM. The .TYPE is set according to the fix type (airport, navaid, or waypoint). The .NAME_LEN field is set to the number of input characters and the .SET field is set. If the fix is not found in AADCOM, a "NOT FOUND IN MEMORY" error message is output.

LSK-L2, LSK-L3, LSK-L4: Radials are entered at these LSKs. If 2D guidance is possible and the entry is a number between 0.0 and 360.0 , COMP_RAD is called to find the intercept point and compute all associated radial information.

LSK-L5: Data is not allowed for the abeam LSK. "INVALID DATA ENTRY" will be output.

LSK-L6: A circle whose radius must be between 1 and 99.9 can be entered at this LSK. The .CIRCLE field of the fix data structure is set to the input value. If the entry does not fall in this range, a "DATA OUT OF RANGE" message is output.

LSK-R6: Data is not allowed for the "ERASE>" LSK. "INVALID DATA ENTRY" will be output.

All other keys are dead keys. Also, if a fix has not yet been entered, all keys except for LSKL1 will be dead keys. Also note that if any numerical entry cannot be successfully converted from character to floating point format, a "BAD DATA FORMAT" message will be output.

```
GLOBAL REFERENCES:

VARIABLES
  ERCODE* FNAME GUID2D INDAT PG

ARRAYS
  ENTRY*

RECORD ARRAYS
  FIX*

FUNCTIONS AND SUBROUTINES
  COMP_RAD GET_LONG GET_REAL LUARP LUGRP LUNAVA OTS$CVT_T_F
```

MODULE NAME: DEL_FIX
FILE NAME: FIX.FOR
PROCESS: SLOW
CALLED BY: FIX_INFO, FIX_INP
CALLING SEQUENCE: CALL DEL_FIX

PURPOSE:
This routine handles "DELETE" fix inputs.

DESCRIPTION:
FIX_INP and FIX_INFO call DEL_FIX when a user has entered "DELETE" as data. The following applies for this type input:

LSK-L1: The fix page is reinitialized. The associated fix information drawn on the ND will also be erased. FIX_INIT is called to reset the fix data structure. The CDU will be ready for a new fix to be entered.

LSK-L2, LSK-L3, LSK-L4: The radial and its associated information is romoved from the CDU and ND and the appropriate radial fields of the fix data structure are reset.

LSK-L5: The fix circle is removed from the CDU and ND and the .CIRCLE field of the fix data structure is set to zero.

Attempting to delete one of these parameters when it does not exist results in an "INVALID DELETE" error message. Any other deletion attempt results in a "DEAD KEY ERROR" error.

GLOBAL REFERENCES:

VARIABLES
ERCODE* PG

ARRAYS
ENTRY

RECORD ARRAYS
FIX

FUNCTIONS AND SUBROUTINES
FIX_INIT

MODULE NAME: CH_FIX_PG
FILE NAME: FIX.FOR
PROCESS: SLOW
CALLED BY: FIX_INFO
CALLING SEQUENCE: CALL CH_FIX_PG(PG)

PURPOSE:
This routine handles page changes between the two fix pages.

DESCRIPTION:
When the user has pressed the PREV or NEXT CDU panel buttons, FIX_INFO calls CH_FIX_PG to set the current page variable PG so that the inactive fix page becomes active.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
FMTOUT

MODULE NAME: FIX_INIT
FILE NAME: FIX.FOR
PROCESS: SLOW
CALLED BY: FUNC_INP_FIX, DEL_FIX
CALLING SEQUENCE: CALL FIX_INIT

PURPOSE:
This routine reinitializes those variables necessary to erase a fix page.

DESCRIPTION:
When a user wishes to erase a fix from the CDU and the ND, FUNC_INP_FIX and DEL_FIX calls FIX_INIT to reset the fields of the fix structure so that the fix will no longer be displayed.

GLOBAL REFERENCES:

VARIABLES
PG

RECORD ARRAYS
FIX*

MODULE NAME: COMP_ABRAD
FILE NAME: FIX.FOR
PROCESS: SLOW
CALLED BY: FUNC_INP_FIX
CALLING SEQUENCE:

PURPOSE:
This routine attempts to find the shortest abeam radial between a fix and a straight leg segment of the active path.

DESCRIPTION:
When a user presses the "abeam" line select key (LSKL5) for a given fix, FUNC_INP_FIX calls this routine to search the remainder of the active path for a straight leg segment in which a radial drawn from the fix to the segment intersects perpendicularly.

FIND_LEG_AB is called for each straight leg segment to perform the actual computations that determine whether the segment contains an abeam radial. If the aircraft's position is currently on a straight leg and the cross track error XTK is less than 200 feet, the unflown portion of that segment is also considered. A segment is considered to be a straight leg if either of its waypoint DMA fields are zero (i.e. WPT_ACT(J).DMA = 0 or WPT_ACT(J+1).DMA = 0, where J+1 is the 'TO' waypoint of the segment). Once all of the abeams have been found, FMIN is called to determine which abeam has the shortest distance between the fix and the intercept point. It is this abeam that will be displayed on the CDU and ND. Before exiting COMP_ABRAD calls AB_IP_LL to compute the intercept point latitude and longitude and COMP_IP_DTG to compute the distance to go along the path to the intercept point. If this distance is greater than zero meaning the intercept point has not been already passed, the desired altitude at the intercept point is also computed.

If no abeams exist, the error message "NO ABEAM RADIAL" will be output to the CDU.

GLOBAL REFERENCES:

VARIABLES
ACTCNT DTOGO ERCODE* LAT LON PG TOWPT XTK

RECORD ARRAYS
FIX* WPT_ACT

FUNCTIONS AND SUBROUTINES
AB_IP_LL COMP_IP_DTG FIND_LEG_AB FMIN MTH$TAND POS_INFO
UNITVEC

MODULE NAME: FIND_LEG_AB
FILE NAME: FIX.FOR
PROCESS: SLOW
CALLED BY: COMP_ABRAD, COMP_RAD
CALLING SEQUENCE: CALL FIND_LEG_AB(FR_VEC, VEC, FLAT, FLON, FRLAT, FRLON, PTR, DIST)

PURPOSE:

The purpose of this routine is to find out if an abeam radial exists for a given straight leg path segment.

DESCRIPTION:

FIND_LEG_AB calls COMP_ANG to compute two angles whose values tell whether or not a segment contains an abeam radial for a fix. The first angle is the one which contains the leg's FROM and TO waypoints and the fix location. The vertex of this angle is the TO waypoint location. The second angle contains the same points except that the vertex is the FROM waypoint. If these two angles are both acute angles, an abeam radial must exist for the leg. If an abeam is found, the distance between the fix and the intercept point is computed. This distance must be greater than 100 feet or the abeam will be rejected.

GLOBAL REFERENCES:

RECORD ARRAYS
WPT_ACT

FUNCTIONS AND SUBROUTINES
COMP_ANG FIX_ERAD MTH$ATAN2 VDP

MODULE NAME: UNITVEC
FILE NAME: FIX.FOR
PROCESS: SLOW
CALLED BY: COMP_ABRAD, COMP_RAD
CALLING SEQUENCE: CALL UNITVEC(ULAT, ULON, UVEC)

PURPOSE:
This routine computes a unit vector (UVEC) from the earth's center to a position given by its latitude (ULAT) and longitude (ULON)

DESCRIPTION:
This routine is called when a unit vector from the earth's center to the given latitude and longitude is needed in COMP_ABRAD and COMP_RAD computations.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
MTH$COSD MTH$SIND

MODULE NAME: FMIN
FILE NAME: FIX.FOR
PROCESS: SLOW
CALLED BY: COMP_ABRAD, COMP_RAD
CALLING SEQUENCE: CALL FMIN(X, CNT)

PURPOSE:
This routine searches the array X for a minimum value greater than zero and returns its position CNT in the array.

DESCRIPTION:
A simple loop is used to find the minimum value greater than zero in the array. If all elements are less than zero, CNT is returned as zero.

GLOBAL REFERENCES: none

MODULE NAME: FIX_ERAD
FILE NAME: FIX.FOR
PROCESS: SLOW
CALLED BY: FIND_LEG_AB, COMP_RAD
CALLING SEQUENCE: CALL FIX_ERAD(ALT1, RAD1, LATFT, LONFT)

PURPOSE:
This routine computes the earth radius for a given position.

DESCRIPTION:
East/west and north/south values are computed for a given position and are then used to compute a local "feet per degree" value for latitude and longitude at that position.

GLOBAL REFERENCES:

VARIABLES
LAT

FUNCTIONS AND SUBROUTINES
MTH$COSD MTH$SIND

MODULE NAME: AB_IP_LL
FILE NAME: FIX.FOR
PROCESS: SLOW
CALLED BY: COMP_ABRAD, COMP_RAD
CALLING SEQUENCE: CALL AB_IP_LL(VECT, PTR, TMP_LAT, TMP_LON)

PURPOSE:

The purpose of this routine is to compute the latitude and longitude of an intercept point between an abeam radial and a given path segment.

DESCRIPTION:

Given the abeam input vector (VECT) and the normal vector WPT_ACT(PTR).NMV, the intercept point latitude (TMP_LAT) and longitude (TMP_LON) are computed using vector algebra.

GLOBAL REFERENCES:

RECORD ARRAYS
WPT_ACT

FUNCTIONS AND SUBROUTINES
MTH$ASIND MTH$COSD VDP

MODULE NAME: COMP_RAD
FILE NAME: FIX.FOR
PROCESS: SLOW
CALLED BY: DATA_INP_FIX
CALLING SEQUENCE: CALL COMP_RAD(VAL, INDEX)

PURPOSE:

This routine is called to find an intercept point between a radial and a straight leg segment of the unflown portion of the active path.

DESCRIPTION:

A user is able to enter up to 3 radials (other than the abeam radial) per fix. These radials have values between 0.0 and 360.0 degrees. Each time a radial is entered, this routine is called by DATA_INP_FIX to see if the radial intersects a straight leg segment of the unflown active path. Passed as inputs are the radial value in degrees (VAL) and the radial index (1, 2, or 3) INDEX.

If the current path leg is a straight segment, COMP_RAD first checks to see if there is an intercept point between the aircraft's current position and the 'TO' waypoint. FIND_LEG_RAD is called to actually do the computations needed to determine if the correct geometry for intersection exists. The remaining straight legs are then searched. Once all intercept points have been found, FMIN is called to determine which radial has the shortest distance from the fix to the intercept point. This distance, however, must be greater than 100 feet or it will not be considered.

If no intercept points were found, only the radial value will be displayed on the CDU. The radial will still be drawn on the ND.

If a minimal distance intercepting radial is found, the intercept point latitude, longitude, and altitude are computed. Also, COMP_IP_DTG is called to compute the aircraft's distance to go along the path to the intercept point. Displayed on the CDU radial line is the radial value, distance in nautical miles from the fix to the intercept point, aircraft distance to go in nautical miles to the intercept point, and the aircraft's desired altitude in feet at the intercept point.

GLOBAL REFERENCES:

VARIABLES
ACTCNT DTOGO LAT LON PG TOWPT XTK

RECORD ARRAYS
FIX* WPT_ACT

FUNCTIONS AND SUBROUTINES
AB_IP_LL COMP_IP_DTG FIND_LEG_AB FIND_LEG_RAD FIX_ERAD
FMIN MTH$COSD MTH$SIND MTH$TAND UNITVEC

MODULE NAME: FIND_LEG_RAD
FILE NAME: FIX.FOR
PROCESS: SLOW
CALLED BY: COMP_RAD
CALLING SEQUENCE: CALL FIND_LEG_RAD(FRLAT, FRLON, FLAT,
FLON, TOLAT, TOLON,
PTOPD, VAL, PTR, DIST)

PURPOSE:
The purpose of this routine is to find an intercept point between a fix radial and a straight leg segment.

DESCRIPTION:
Given input latitudes and longitudes for the 'TO' and 'FROM' waypoints of the path segment and the fix, FIND_LEG_RAD is able to determine whether or not an intercept point exists. It does this by computing two bearings: one from the fix to the FROM waypoint of the segment and the other from the fix to the 'TO' waypoint. If the radial value lies between these two bearings, an intercept point must exist and the distance between the fix and the intercept point will be computed.

GLOBAL REFERENCES:

VARIABLES
PG

RECORD ARRAYS
FIX

FUNCTIONS AND SUBROUTINES
COMP_ANG F_ANG GRID MTH$ATAND2 MTH$COSD MTH$SIND MTH$SQRT

MODULE NAME: F_ANG(X, Y, ANG)
FILE NAME: FIX.FOR
PROCESS: SLOW
CALLED BY: FIND_LEG_RAD
CALLING SEQUENCE:

PURPOSE:
This routine is called to find the angle between two given bearings that share a common vertex.

DESCRIPTION:
The bearing inputs X and Y have values between -180.0 and 180.0 degrees. The angle between the bearings will be returned in ANG as a value between 0.0 and 360.0 degrees.

MODULE NAME: COMP_ANG
FILE NAME: FIX.FOR
PROCESS: SLOW
CALLED BY: FIND_LEG_AB, FIND_LEG_RAD
CALLING SEQUENCE: CALL COMP_ANG(LT1, LN1, LT2, LN2, LT3, LN3, ANG)

PURPOSE:
This routine computes the angle between a vertex point and two other points.

DESCRIPTION:
The latitudes and longitudes of three points are passed in as parameters. The second lat/lon pair is the vertex. The computation is a matter of simple geometry. The angle is stored in ANG and will have value between 0.0 and 360.0 degrees.

GLOBAL REFERENCES:

FUNCTIONS AND SUBROUTINES
GRID MTH$ATAND2

MODULE NAME: POS_INFO
FILE NAME: FIX.FOR
PROCESS: SLOW
CALLED BY: FIX_INFO, COMP_ABRAD
CALLING SEQUENCE: CALL FIX_INFO(LT1, LN1, LT2, LN2, DIST, BRG)

PURPOSE:
The purpose of this routine is to compute the magnetic bearing (degrees) and distance (feet) between two points.

DESCRIPTION:
POS_INFO is called whenever bearing and distance are needed between two points. It uses the latitudes and longitudes of the two points to perform the computations. The bearing will be returned as a value between 0.0 and 360.0 degrees.

GLOBAL REFERENCES:

VARIABLES
PG

RECORD ARRAYS
FIX

FUNCTIONS AND SUBROUTINES
GRID MTH$ATAND2 MTH$SQRT

MODULE NAME: COMP_IP_DTG
FILE NAME: FIX.FOR
PROCESS: SLOW
CALLED BY: FIX_INFO, COMP_ABRAD, COMP_RAD
CALLING SEQUENCE: CALL COMP_IP_DTG(PG, INDEX, DTGO)

PURPOSE:

This routine is called to compute the distance along the path from the aircraft's current position to a radial's intercept point on the path.

DESCRIPTION:

COMP_IP_DTG is called every time FIX_INFO is called to compute the distance to the intercept point for the active fix (PG) and the given radial index (INDEX). It sums 1) the distance from the current position to the 'TO' waypoint, 2) the distances of the path legs up to the waypoint just before the intercept point, and 3) the distance from that waypoint to the intercept point. If the intercept point has already been passed, there is no computation to perform and only the radial value will be displayed. In this case DTGO is passed back as zero.

COMP_ABRAD and COMP_RAD also call this routine. If the distance returned is zero, these routines do not attempt to compute the desired aircraft altitude at the intercept point.

GLOBAL REFERENCES:

VARIABLES
DTOGO LAT LON TOWPT

RECORD ARRAYS
FIX* WPT_ACT

FUNCTIONS AND SUBROUTINES
GRID MTH$SQRT

MODULE NAME: FIX_DISP
FILE NAME: FIX.FOR
PROCESS: SLOW
CALLED BY: FIX_INFO
CALLING SEQUENCE: CALL FIX_DISP

PURPOSE:
This routine sets the parameters used by the displays computer to display fix information on the Navigation Display.

DESCRIPTION:
FIX_INFO calls this routine every time the executive calls FIX_INFO. The global integer array FIXWRD(2) has bit patterns that are set by FIX_DISP. FIXWRD is used by the displays computer to graphically represent the fix(es) and radials. FIXWRD(1) is used for the first fix and FIXWRD(2) is used for the second page. The bit patterns for each word are:

bit 0: set if a fix has been entered
bit 1: set if first radial has been entered
bit 2: set if second radial has been entered
bit 3: set if third radial has been entered
bit 4: set if abeam has been computed
bit 12: set if fix is a navaid
bit 13: set if fix is a waypoint
bit 14: set if fix is an airport

If the conditions for setting a bit do not exist it remains cleared. Also set for use in the displays computer and ND are the following global variables:

FIXADD: address of fix in AADCOM
FIXCIR: fix circle radius
FIXRAD(1 - 4): radial values

GLOBAL REFERENCES:

VARIABLES
FIXADD* FIXCIR* FIXRAD* FIXWRD*

RECORD ARRAYS
FIX

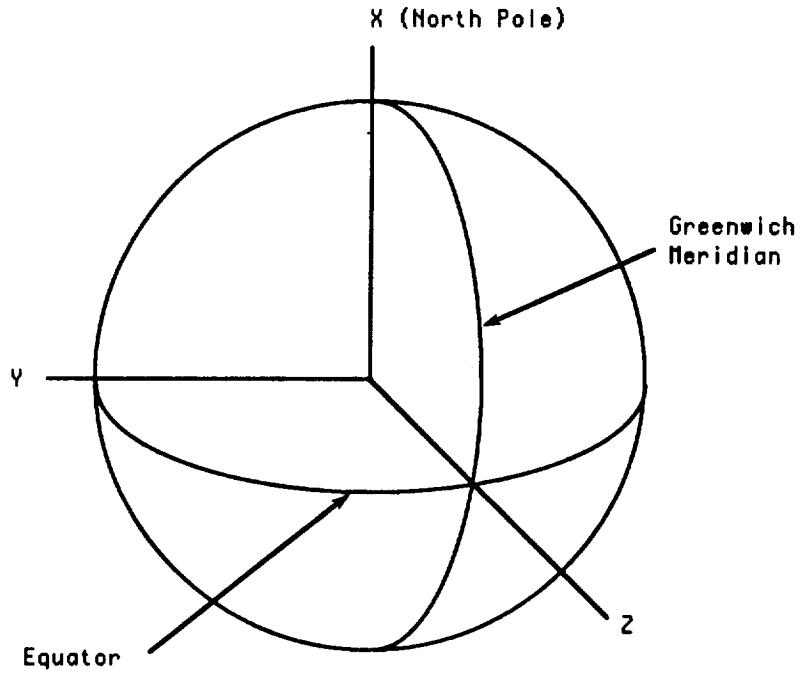
Appendix A PATH DEFINITION COMPUTATIONS

*** THE WAYPOINT UNIT VECTOR ***

The unit vector from the Earth center pointing to a global position (latitude & longitude) is defined in the guidance vector coordinate system as shown below.

$$\hat{WP} = [SIN(LAT),\ -COS(LAT)\ SIN(LON),\ COS(LAT)\ COS(LON)]$$

The elements of this vector are stored in the waypoint buffer structure element labeled WPT_MOD(i).WPV. The figure below describes the coordinate system involved in the guidance vector computations.



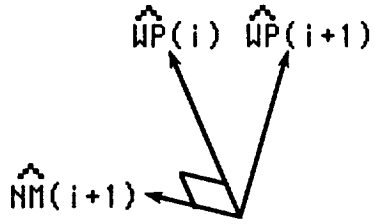
. Longitude measured in degrees east of the Greenwich meridian
. Latitude measured in degrees north of the equator

*** THE NORMAL UNIT VECTOR ***

The normal unit vector is calculated as follows.

$$\overline{NM}(i) = \hat{WP}(i-1) \times \hat{WP}(i); \quad \hat{NM}(i) = \overline{NM}(i) \; / \; |\overline{NM}(i)|$$

The elements of these vectors are stored in the waypoint buffer element labeled WPT_MOD(i).NMV. The figure below depicts the relationship between the normal unit vector and its associated waypoint unit vector pair.



*** TURN ANGLE (TA) BETWEEN WAYPOINTS ***

The normal unit vector and the waypoint unit vectors are used to compute the track angle change between waypoints by using the following equations.

$$SIN\_TA = -\hat{NM}(i) \times \hat{NM}(i+1) \cdot \hat{WP}(i)$$

$$COS\_TA = \hat{NM}(i) \cdot \hat{NM}(i+1)$$

$$TA = ARCTAN2(SIN\_TA, COS\_TA)$$

The turn angle at each waypoint, except the first and last on the flight plan or DMA turn waypoints, is stored in the waypoint structure under the label WPT_MOD(i).TA.

*** RADIUS OF TURN ***

The following list shows the three ways the turn radius at a waypoint may be defined. They are listed in the order of highest priority.

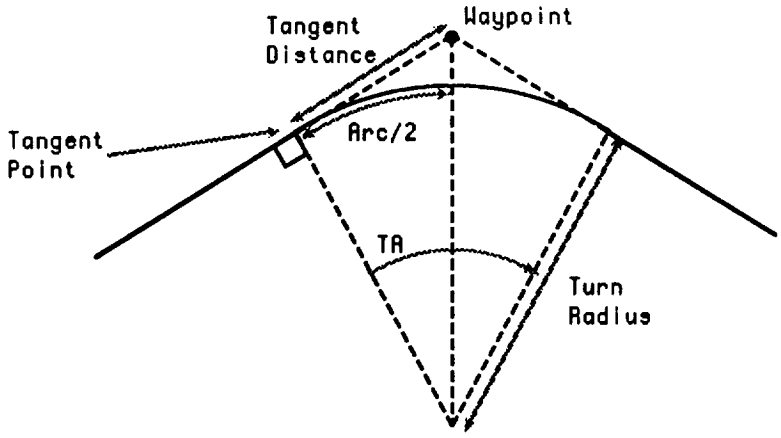
- Assigned directly from the CDU.
- Default from system database when waypoint is part of a defined route function.
- When a ground speed is defined at the waypoint, the equation R = GS ** 2 / (G * TAN_15_DEG).
- When the waypoint altitude is assigned, and greater than 15000 feet, the radius is set to 50000 feet.
- Radius is set to 15000 feet.

*** TURN ARC LENGTH ***

The arc length calculation is performed for all waypoints except DMA arc waypoints. The value computed and stored is actually one half the turn arc length because of the way it is used by other software. The following formula is used in the calculation.

$$HALF\_ARC = TURN\_RAD * TURN\_ANG\_RADIANS / 2$$

The computed value is stored in the waypoint buffer element WPT_MOD(i).ARC2.



*** THE DISTANCE TO THE TANGENT POINT (DTT) ***

The distance between a waypoint and either tangent point is computed as follows.

$$DTT = TURN\_ANG\ [SIN(TA) / (1 + COS(TA))]$$

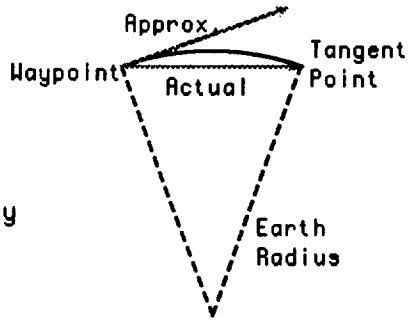
Note that SIN(TA) / (1 + COS(TA)) is identical to TAN(TA / 2). The first form is used because the sine and cosine both already exist from the turn angle calculation. The tangent distance is stored in the waypoint buffer element WPT_MOD(i).DTT.

*** THE TURN CENTER VECTOR (TCV) ***

The first step in finding the turn center vector is to compute the waypoint to tangent point vector (TNGT). This vector is approximated by the following equation.

$$\overline{TNGT} = DTT\ [\hat{WP} \times \hat{NM}]$$

The figure to the right depicts the nature of the approximation. Note that the difference shown is highly exaggerated to emphasize the point. In real conditions the arc distance to the tangent point is extremely small compared to the earth's radius.



Next the vector fom the earth's center to the tangent point (TP) is computed as follows.

$$\overline{TP} = EARTH\_RAD\ \hat{\overline{WP}} + \overline{TNGT}$$

The turn center vector calculations proceed as shown below. Note that the "+/-" shown means add in a left turn and subract in a right turn.

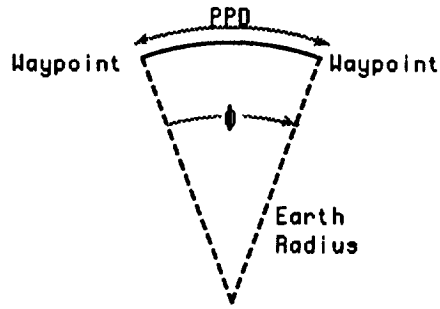
$$\overline{TCV} = \overline{TP} +/- (TURN\_RAD\ \hat{\overline{NM}});\quad \hat{\overline{TCV}} = \overline{TCV} / |\overline{TCV}|$$

The turn center vector is saved as WPT_MOD(i).TCV

*** THE DISTANCE BETWEEN WAYPOINTS (PPD) ***

The distance between waypoints, "point to point" distance, is found by computing the angle (radians) between successive waypoint vectors. The equations are shown below. The computed value is saved as WPT_MOD(i).PPD.

$$SIN\_\phi = |WP(i-1) \times \hat{WP}(i)|$$
$$PPD = EARTH\_RAD\ ARCSIN(SIN\_\phi)$$

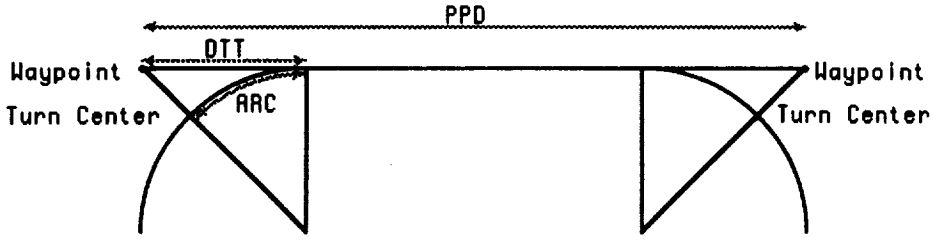


*** CENTER OF TURN LEG DISTANCE (CCD) ***

The along path distance between waypoints, turn center to turn center, is computed from the point to point distance by adjusting each end of the point to point line. The adjustment is performed as follows.

$$CCD(i) = PPD(i) - DTT(i) + ARC(i) - DTT(i-1) + ARC(i-1)$$

Note that no adjustment is performed for leg ends having a DMA turn waypoint since no DTT or ARC distances are defined. The resulting path distance is saved as WPT_MOD(i).CCD.



*** THE FLIGHT PATH ANGLE (FPA) ***

The flight path gradient between successive waypoints is approximated from the along path distance between waypoints and the change in assigned altitudes. The computed value is actually the tangent of the flight path angle, however it will approximate the desired number for the small flight path angles typically on the flight plan.

$$FPA\_RADIANS = (ALT(i) - ALT(i-1)) / CCD(i)$$

The value is converted to degrees and stored in WPT_MOD(i).FPA.

*** THE PATH LEG TIME ***

The time required to fly a leg of the flight plan is computed when ground speeds (GS) are defined for both leg end waypoints.

$$TIME(i) = 2\ CCD(i) / (GS(i) + GS(i-1))$$

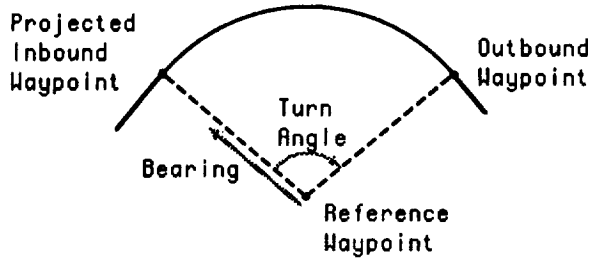
Note that the ground speeds must be converted to feet per second before use (stored as knots) since the path distance (CCD) is defined in feet. The resulting value of leg time is saved in WPT_MOD(i).TIME.

*** DMA ARC COMPUTATIONS ***

The DMA arc reference waypoint position, projection bearing and turn angle are predefined in the navigation database. The position of the projected waypoint and along path distance are computed as follows.

$$LAT = LAT\_REF + TURN\_RAD\ COS(BRNG) / DLATFT$$
$$LON = LON\_REF + TURN\_RAD\ SIN(BRNG) / DLONFT$$
$$CCD = TURN\_ANGLE\_RADIANS\ TURN\_RAD$$

Note that DLATFT and DLONFT are the number of degrees per foot in latitude and longitude respectively, defined for the locality of the current turn center.



REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 1992	Contractor Report

4. TITLE AND SUBTITLE Advanced Transport Operating System (ATOPS) Control Display Unit Software Description

5. FUNDING NUMBERS
C NAS1-19038
WU 505-64-13-11

6. AUTHOR(S)
Christopher J. Slominski
William J. Heaphy
Mark A. Parks
Kelly R. Debure

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Computer Sciences Corporation
3217 N. Armistead Avenue
Hampton, Virginia 23666-1379

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665-5225

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA CR-189606

11. SUPPLEMENTARY NOTES
Langley Technical Monitor: Dr. James R. Schiess (COTR)
Robert A. Kudlinski

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited

Subject Category 06

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This document describes the software created for the Lear-Siegler Control Display Units (CDUs) used for the Advanced Transport Operating Systems (ATOPS) project on the Transport Systems Research Vehicle (TSRV). The software delivery of April 1991, known as the "baseline system", is the one described in this document. Throughout this publication, module descriptions are presented in a standardized format which contains module purpose, calling sequence, detailed description and global references. The global reference section includes subroutines, functions and common variables referenced by a particular module.

The system described supports the Research Flight Deck (RFD) of the TSRV. The RFD contains two Lear-Siegler CDUs, one for the pilot and copilot, which are used for flight management purposes. Operations performed with the CDU affects the aircraft's guidance, navigation, and display software.

14. SUBJECT TERMS			15. NUMBER OF PAGES
Aircraft Navigation Systems	Flight Management Computer	Flight Management Systems	420
Glass Cockpit	Control Display Unit	Area Navigation Software	16. PRICE CODE
Flight Planning	Electronic Flight Instrumentation System		A18

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	