AD-A056 842 ROCKWELL INTERNATIONAL CEDAR RAPIDS IA COLLINS AVION--ETC F/G 1/5
3D/4D AREA NAVIGATION SYSTEM DESIGN, DEVELOPMENT AND IMPLEMENTA--ETC(U)
JUN 77 J M BRUCKNER, F B BENSON DOT-FA72WA-3123
UNCLASSIFIED FAA-RD-77-79-VOL-2 NL

1 OF 3
AD
A056842

LEVEL III

A056605

2

SC

Report No. FAA-RD-77-79, II

AD A056842

3D/4D AREA NAVIGATION SYSTEM DESIGN, DEVELOPMENT, AND IMPLEMENTATION

Volume II Support Data and Program Listings

J.M.H. Bruckner, F.B. Benson,
L.J. McCormick, T.G. Sharpe
Collins Avionics ~~Group~~ Div.
Rockwell International
Cedar Rapids, Iowa 52402

JUNE 1977
Final Report

AD No.____
DDC FILE COPY

Document is available to the U.S. public through
the National Technical Information Service,
Springfield, Virginia 22161.

D D C
RECEIVED
JUL 31 1978
D

Prepared for

U.S. DEPARTMENT OF TRANSPORTATION
FEDERAL AVIATION ADMINISTRATION
Systems Research & Development Service
Washington, D.C. 20590

NOTICE

⑩ Juergen M.H./Bruckner, F.B./Benson,
Larry J./McCormick, T.G. Sharpe
Larry H. Hogle

⑫ 244p.

Technical Report Documentation Page

1. Report No. ⑱ ⑲ FAA-RD-77-79-~~II~~-VOL-2	2. Government Accession No.	3. Recipient's Catalog No. ⑪ Jun 77
4. Title and Subtitle ⑥ 3D/4D AREA NAVIGATION SYSTEM DESIGN, DEVELOPMENT AND IMPLEMENTATION, Volume II. Support Data and Program Listings.		5. Report Date July, 1977
		6. Performing Organization Code
7. Author(s) Juergen M. H. Bruckner, Larry H. Hogle, Larry J. McCormick, and Bob J. Rudig		8. Performing Organization Report No.
9. Performing Organization Name and Address Rockwell International Collins Avionics Group 400 Collins Road Cedar Rapids, Iowa 52402		10. Work Unit No. (TRAIS)
		11. Contract or Grant No. ⑮ DOT-FA72WA-3123
12. Sponsoring Agency Name and Address Department of Transportation Federal Aviation Administration Systems Research and Development Service Washington, DC 20590		13. Type of Report and Period Covered ⑨ Final Report
		14. Sponsoring Agency Code ARD-333
15. Supplementary Notes		

16. Abstract

This volume presents the detailed support data for the 4D time control and localizer software definitions. Program listings for the final design are also presented.

Volume I, Main Text, contains 99 pages.

17. Key Words Area Navigation Time Control ILS Capture/Tracking Algorithms		18. Distribution Statement Document is available to the US public through the National Technical Information Service, Springfield, Virginia 22161	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 232	22. Price

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

392 654

Table of Contents

Table of Contents

List of Illustrations

List of Tables

Section 1
INTRODUCTION AND SUMMARY

1.1 Summary of Volume II

This volume describes the detailed design and final implementation of the software for the 3D/4D Base and ILS and 2D/Time Control Systems. The systems were constructed by modifying the software structure used for the ANS-70A Area Navigation System.

A description of the 3D/4D software effort requires some knowledge of the ANS-70A software structure. A brief functional description of the ANS-70A software is given in the following paragraphs.

Sections 2 and 3 describe the software modifications used to effect the BASE and 2D/Time Control Systems respectively. Section 4 presents the RNAV to ILS interface design procedure. Sections 5 and 6 present the development of the lateral and longitudinal control laws respectively. Program listings are found in the appendices.

1.2 Overview of ANS-70A Software System

The ANS-70A software system has several different priority levels. The highest level of software functions have been assigned to processor channels. Each channel is allocated processing time on a sequential basis. Channels of particular interest for the present treatment are the Navigation and CDU Application Channels. Major functions assigned to these channels are illustrated in Figures 1-1 and 1-2.

The classification of major functions for the Navigation Channel remains unchanged for the 3D/4D systems. Secondary functions have been added to the software modules that support the major tasks of Aircraft Systems Coupler (ASC) Input, Lateral Navigation, and ASC Output. Navigation tasks that are unique to the 3D/4D Base and 2D Plus Time Control systems are described in Sections 2.0 and 3.0 respectively.

the CDU Channel. In addition to providing software for the CDU interface, this channel supervises tasks related to navigation support and Flight Plan Management.

Background support for the navigation function is provided by routines called the Radio Selection Program, Kalman Filter, and the Flight Plan Monitor. Each of these programs is called on a cyclic basis by the CDU Main Program. The first two routines are used without modification for the 3D/4D systems.

The Flight Plan Monitor serves as an interface between the Navigation software and the Flight Plan Editor. A primary task for the Flight Plan Monitor consists of retrieving flight plan parameters required by the Lateral and Vertical Navigation routines. Another major task consists of supervising flight plan maintenance that is required at waypoint passage. 3D/4D modifications to the Flight Plan Monitor are given in later sections of this volume.

Figure 1-3 illustrates software interfaces that are required for Flight Plan Management. Initial verification of flight plan revisions entered via the CDU are performed by the CDU Software. If a given revision fails initial tests, processing of the entry is discontinued and error annunciation is given on the CDU scratchpad. Valid revisions are recorded in the Flight Event Table by calling the Flight Plan Editor. Flight Plan parameters required by the Navigation software are then updated by the Flight Plan Monitor.

In addition to recording flight plan revisions, the Flight Plan Editor performs all geometry computations that are required as internal flight plan parameters (e.g., distance and course information). The impact of Base Offsets on the Flight Plan Editor is treated in Section 2.2.

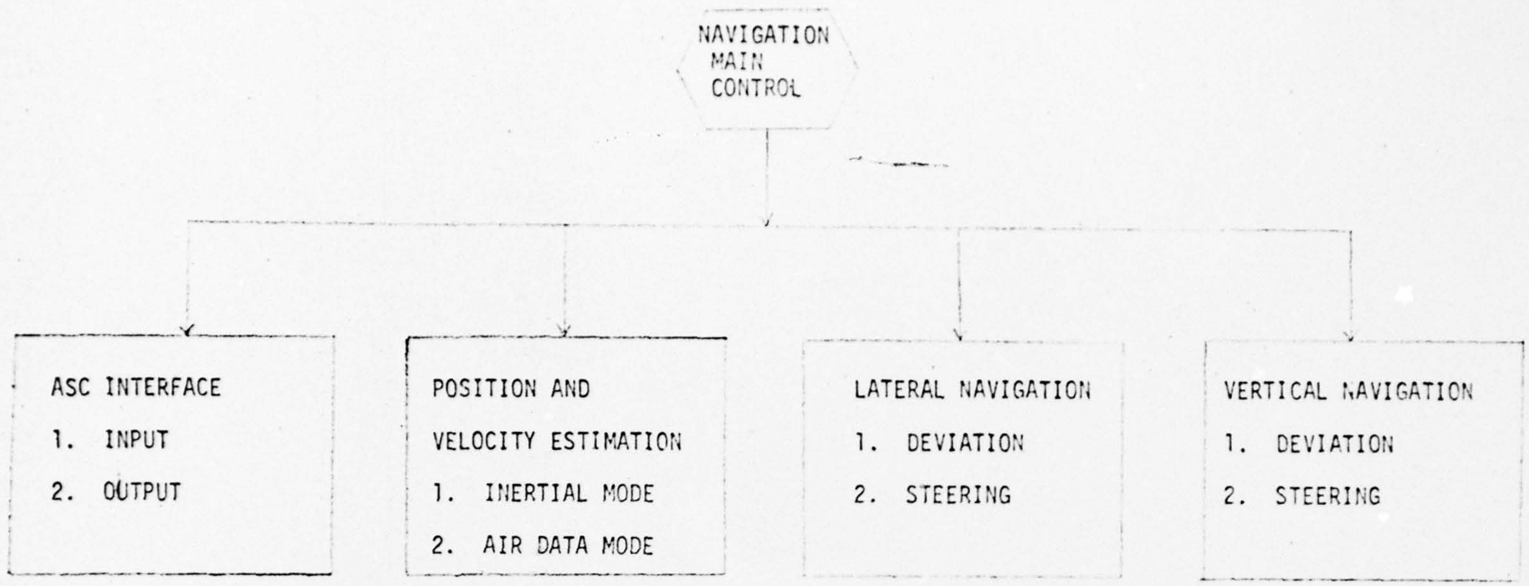


FIGURE 1-1

Major ANS-70A Functions for the Navigation Channel

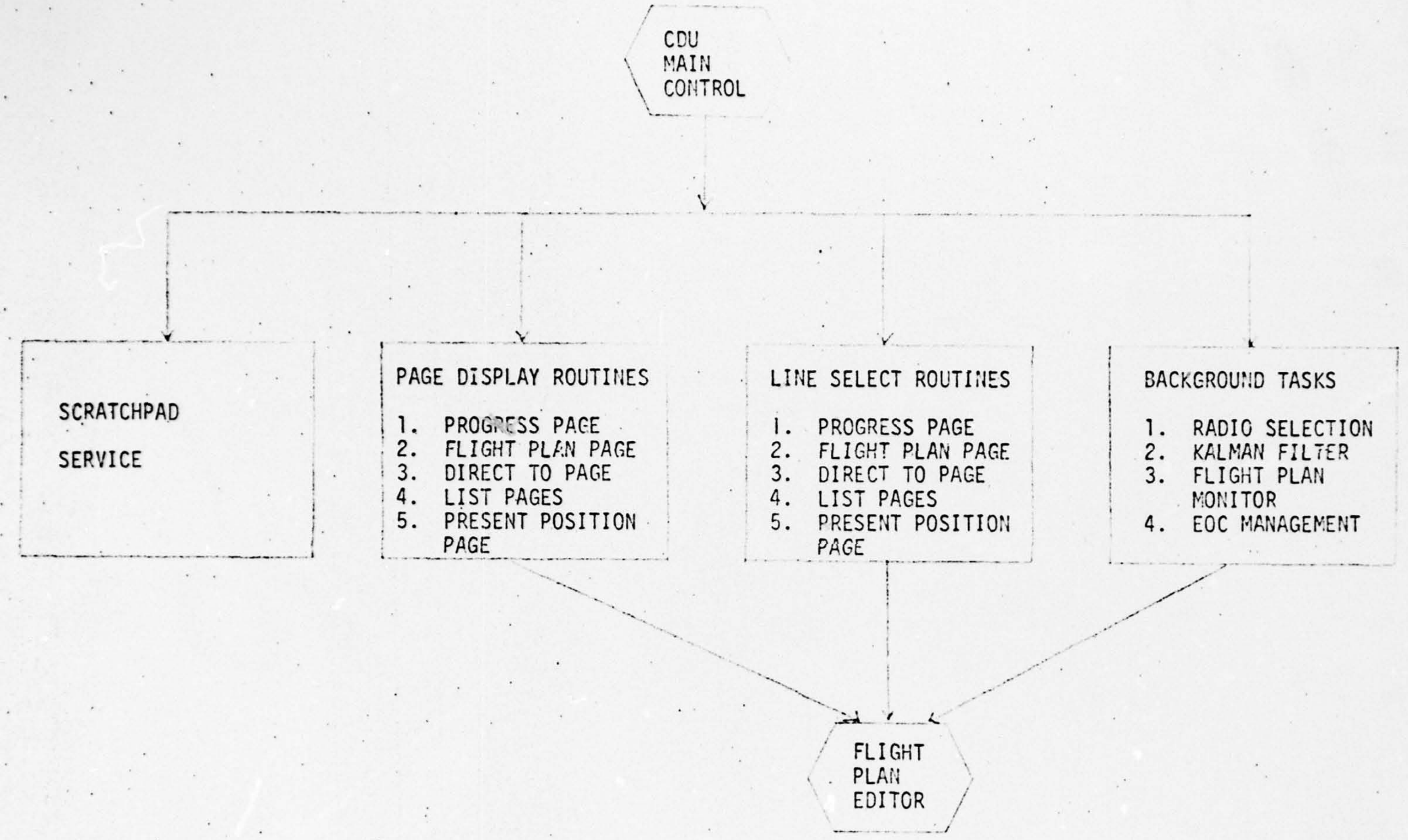


FIGURE 1-2

Major ANS-70A Functions for the CDU Channel

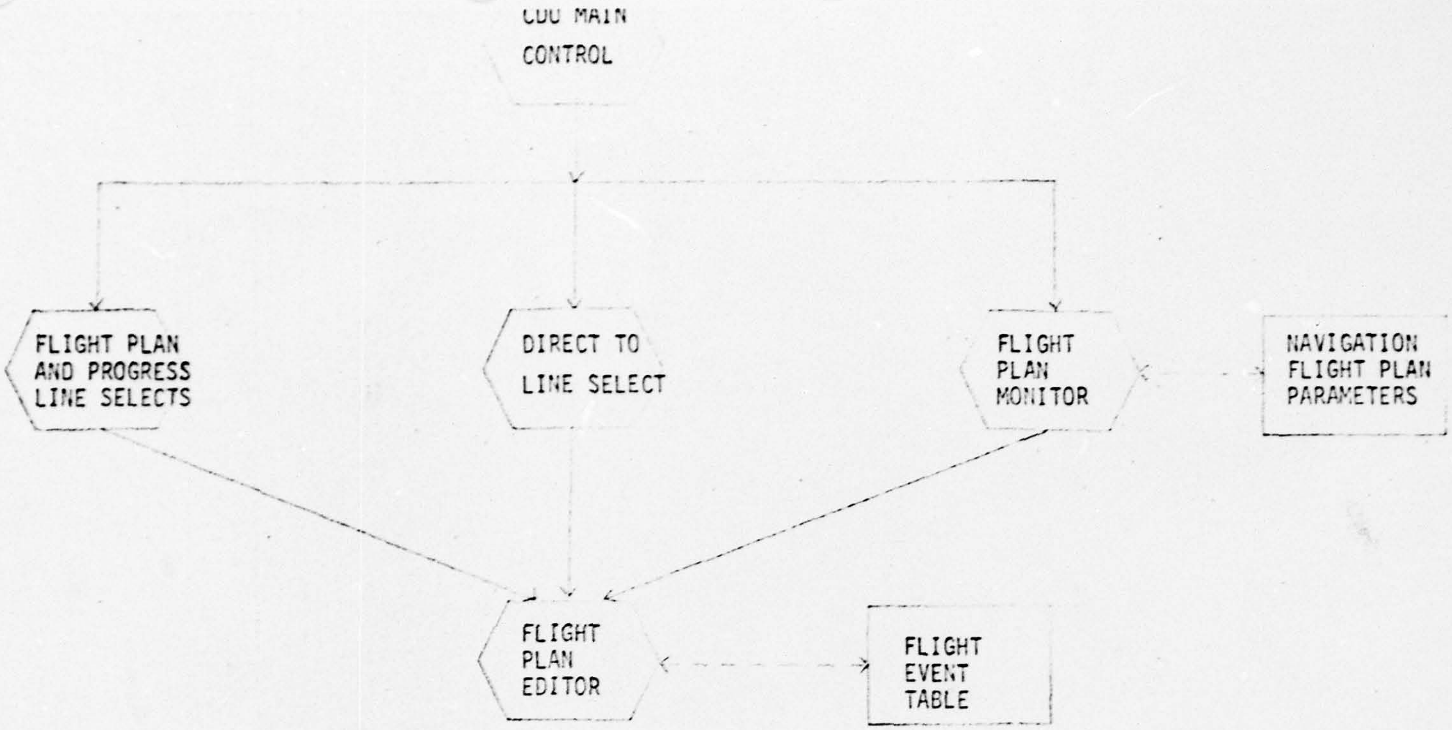


FIGURE 1-3

ANS-70A Software Interfaces for Flight Plan Management

A major goal for the software design stage was to construct a software package that minimized the impact of 3D/4D operational requirements on the existing ANS-70A Software Interfaces. In addition to ensuring system integrity, additional goals consisted of ensuring flexibility for additional changes and minimizing core requirements to the extent that software overlays could be avoided.

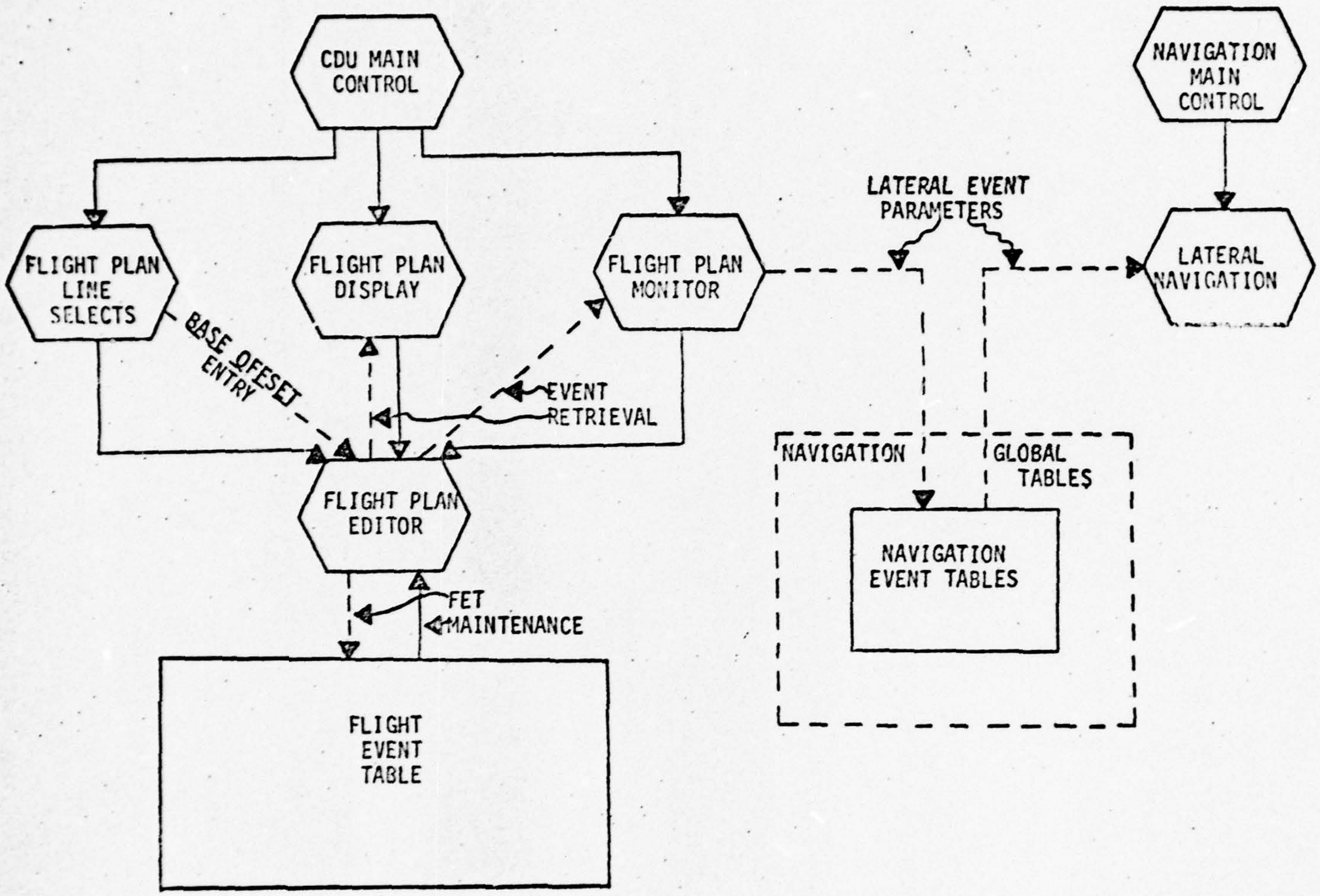
A core reduction effort was required before the 3D/4D tasks could be added to the ANS-70A configuration. Approximately 500 words of core were freed for 3D/4D usage by deleting software for Holding Patterns, the Inertial mode of navigation, Heading and Armed modes, and Commanded Vertical Angle. An additional 1500 words were gained by reducing the Data Base to 1000 words.

Once the core reduction effort was completed, the following major software tasks were implemented:

1) External Data Files
2) Base Offsets
3) Speed Page
4) Speed Profile Algorithm
5) Time Control Algorithm
6) Navigation Modifications
7) ASC Modifications

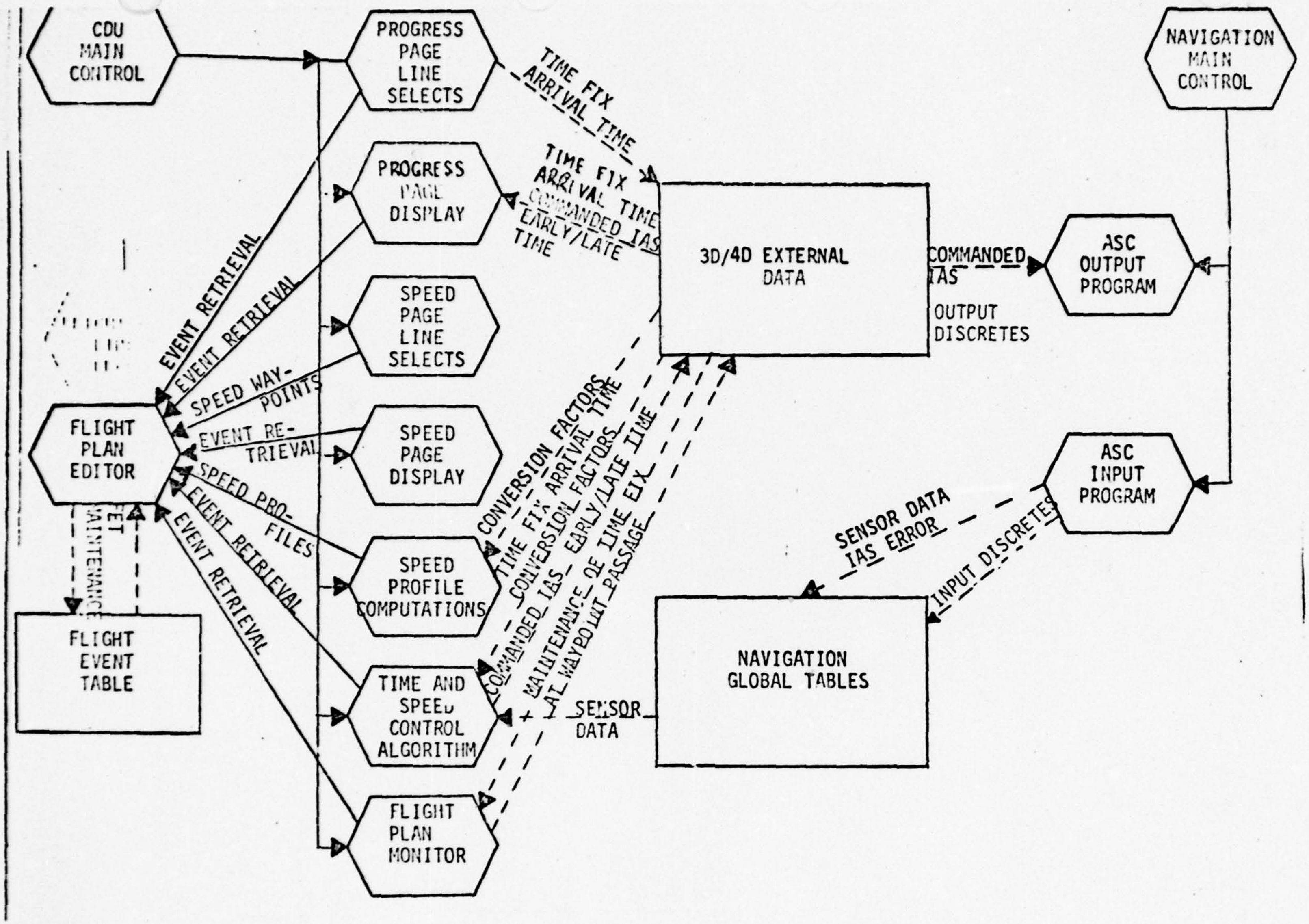
Each of these items is explained in detail by sections 2.1 through 2.6.

Software and data interfaces required for the major software tasks are illustrated in Figures 2.1 through 2.3. Routines for Base Offsets, the Speed Page, the Speed Profile Algorithm, and the Time Control Algorithm are supervised by the CDU Main Control Program. The ASC interface and navigation computations are controlled by the Navigation Main Control Program.



SOFTWARE INTERFACES FOR BASE OFFSETS

FIGURE 2-1



SOFTWARE INTERFACES FOR TIME CONTROL

FIGURE 2-2

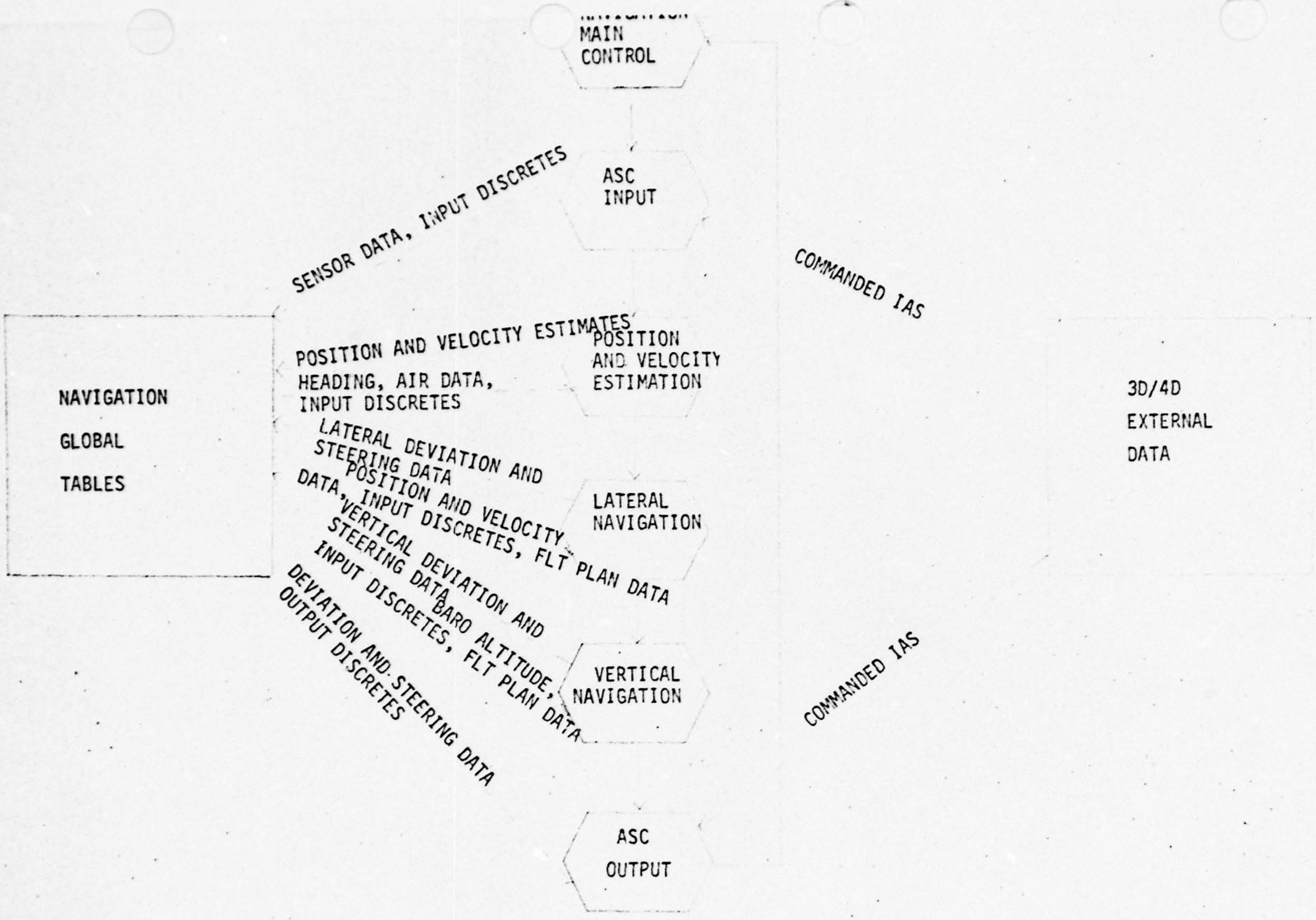


FIGURE 2-3

Software Interfaces for Navigation and ASC Interfaces

2.1 External Data Files

Program descriptions given in later sections of this volume reference data parameters whose data declarations are external to the given programs. The purpose of this section is to provide data descriptions for this external data set.

Data parameters required by more than one program have been classified by function into three basic categories. Separate compilation units have been assigned to each class of parameters for core allocation purposes. Each unit has been given a corresponding INSERT file which provides data declarations. Programs that require data declarations for a given function include an INSERT statement for the appropriate file.

The three basic EXTERNAL data files used for the Base System are 1) 3D/4D External Data File, 2) Flight Plan Editor File, and 3) the Navigation Global Tables File. A description of the data parameters assigned to each file is given below.

2.1.1 3D/4D External Data

The majority of EXTERNAL variables that are unique to the 3D/4D Task II effort are defined by the 3D/4D External Data Insert and 3D/4D External Data Preset files. Data definitions for this set of variables are given below.

Symbolic Name	Asssigned Value	Description
VALID	0	Used to assign a valid status to 3D/4D Integer parameters that are treated as boolean data.
INVALID	1	Used to assign an invalid status to the same set of Integer parameters described for the VALID synonym.

The following list of synonyms are used for VCOMIF.

Synonym Name	Assigned Value	IAS Data Display On Progress Page	Description
IASVALID	15	Value of IAS that is maintained in VCOMI.	A valid commanded IAS has been calculated by the Time and Speed Control algorithm.
NODATA	31	----	No IAS has been entered on the Speed Page.
IVGMT	14	GMT?	Either GMT has not been entered on the Present Position Page, or a power interrupt of duration exceeding two seconds has invalidated GMT.
IVTAS	13	TAS?	The Air Data Sensor input for TAS is invalid.
IVALT	11	ALT?	The Altimeter Sensor input for Baro Altitude is invalid.
IVCOCH	7	CRS!	A Teardrop Procedure is required to capture at least one of the legs used for Time Control computations.

Integer Declarations

Symbolic Name	Initial Value	Description
IASMSK	"FFFFFF00"	Mask used in "AND" operations for the removal of the IASF component.
TFIX	0	ASCII identifier of the waypoint assigned as the Time-Fix.
TCOM	0	Commanded time of arrival for the Time-Fix. TCOM is recorded in GMT units (i.e., as a fraction of 1200 hours).
TFIXF	Invalid	Validity flag for TFIX.
VCOMIF	Invalid	Validity flag for VCOMI and VCOMT.
TCOMF	Invalid	Validity flag for TCOM.
TIMERF	Invalid	Validity flag for TIMERR.
COMPRF	INVALID	Validity flag that is assigned a VALID status by the Flight Plan Monitor whenever a flight plan revision has occurred. SPDPRF, the Speed Profile Routine, assigns an INVALID status to COMPRF whenever speed profile computations are performed.

Symbolic Name	Initial Value	Description
TCONF	INVALID	Validity flag that is assigned a VALID status by the speed profile routine whenever speed profiles are updated. TCONF is assigned an INVALID status whenever Time Control computations are performed.

Real Declarations

Symbolic Name	Initial Value	Description
MINR	.000000001	Minimum magnitude that REAL numbers may be assigned on the U-1108.
VCOMI	.050	Commanded IAS displayed on the Flight Progress Page and IAS instrument. VCOMI is a fraction of 1000 kts.
VCOMT	*.050*	*Commanded TAS calculated by the Speed* and Time Control Algorithm.
MINRATE	.038566097	Value of 40 kts/min Speed Gradient. MINRATE = 1/{ (40 KT/MIN)/(1000 KT LAT VEL BASE) * 60 MIN/HR * 10.8039624 HRS LAT TIME BASE }
EARLY	.0	Expected arrival time error for the Time-Fix. EARLY is expressed in Lateral Navigation Time Units. The Early Late Display on the Flight Progress Page is based on the value of EARLY. A positive value indicates a late status.
TIMERR	.0	Magnitude of EARLY. TIMERR is used by the CDU software for display of EARLY/LATE data on the Progress Page.

2.1.2 Flight Plan Editor Files

Lateral and vertical flight plan parameters maintained by the Flight Plan Editor are illustrated in Figures 2-4 and 2-5. The IASR, IASI, IASF, POFSI, and POFSIC components are the only parameters that are unique to the 3D/4D systems. Descriptions for this set of parameters are given below.

The component structure for IAS parameters is given as follows:

	0 23	24 31
IASI/IASR		IASF

The IASF component is used to record the status of the IAS entry. Synonyms used for this parameter are given below.

Synonym Name	Assigned Value	Description
PIAS	7	Pilot-Defined IAS.
PRQUOTE	11	Profile IAS Displayed on the Speed Page as ".
PRDOWN	13	Profile IAS Displayed on the Speed Page as ↓.
PRUP	14	Profile IAS Displayed on the Speed Page as ↑.
NOIAS	15	No IAS Specified.

The first 24 bits of the entry are used to record the actual IAS magnitude as a REAL-valued fraction of 1000 kts. Although real arithmetic operations are required for the 3D/4D Speed Control and Profile Algorithms, it is also necessary to provide the capability of integer masking operations for the IASF component. Thus, the user is provided with both REAL (IASR) and INTEGER (IASI) component declarations for the IAS parameter.

WORD NUMBER	LABEL	MEANING
-3	VERS	Current Flight Plan Version (VERSFE)
-2	ABEVNO	Absolute Event Number for Event
-1	CNTRWORD	Control Word
0	WYPTID	Waypoint ID
1	WYPTLAT	Waypoint Latitude
2	WYPTLONG	Waypoint Longitude
3	WYPTVAR	Magnetic Variation (Not Supplied by the Editor)
4	POFS/POFSF	Parallel Offset (POFSFE)
5	AOFS	Along Track Offset and Status
6	DFTO	Distance from "TO" LE
7	APOF	Along Path Offset (Always = AOFS)
8	CRSI	Course In
9	CRSO	Course Out
10	AWYID	Airway ID
11	IASR/IASI IASF	Indicated Airspeed Word and Flag

FIGURE 2-4

Lateral Event Parameters Maintained by the

Flight Plan Editor

WORD NUMBER	LABEL	MEANING
-3	VERS	Current Flight Plan Version (VERSFE)
-2	ABEVNO	Absolute Event Number for Event
-1	CNTRWORD	Control Word
0	WYPTID	Waypoint ID
1	WYPTLAT	Waypoint Latitude
2	WYPTLONG	Waypoint Longitude
3	WYPTVAR	Magnetic Variation (Not Supplied by Editor)
4	POFS	Parallel Offset (POFSFE)
5	AOFS	Along Track Offset
6	DFTO	Distance from "TO" LE
7	APOF	Along Path Offset (Always = AOFS)
8	VPAI	Vertical Path Angle In
9	VPAO	Vertical Path Angle Out
10	ALTD	Altitude at VE

FIGURE 2.5

Vertical Event Parameters Maintained by the

Flight Plan Editor

The POFS component is used to record both parallel and base offsets. This component is defined as follows:

POFS/POFSI POFSF

Base and Parallel offsets are treated as Real-valued fractions of π times the radius of earth in nautical miles. The component declarations for POFS, POFSI, and POFSF are similar in structure to IASR, IASI, and IASF, respectively. Synonyms applicable to the POFSF component are given below. Figure 2-6 illustrates the Base Leg waypoints identified by the POFSF synonyms.

POFSF Synonyms

Symbolic Name	Assigned Value	Description
PARALLEL	7	Identifies Parallel Offset Entry. The synonym "PARALLEL" serves a dual purpose. The Flight Plan Editor monitors the POFSF component for one-to-one replace edits to determine the Insertion/Deletion of Parallel Offsets for the Flight Plan. Procedures that use the Flight Plan Editor "GET" routines test the POFSF component for a value of "PARALLEL" to determine the existence of a Parallel Offset.
BASEOFST	8	Identifies the POFSI component as a base offset. This value is used to inform the Flight Plan Editor that the Flight Plan Revision is for a Base Offset.

Symbolic Name	Assigned Value	Description
STRT90	13	Identifies initial waypoint associated with a base offset that provides a 90 degree capture with the localizer leg.
END90	11	Identifies final waypoint associated with a base offset that provides a 90 degree localizer capture.
STRTBASE	12	Identifies initial waypoint associated with a base offset that provides a shallow capture of the localizer leg.
ENDBASE	10	Identifies waypoint whose CRSI component defines the Base Leg for a Base Offset defined for a shallow localizer capture angle.
INTERCEP	9	Identifies waypoint whose CRSI and CRSO components define the intercept point for localizer capture.
NOPOFS	15	No offset entry has been made.
CANCEL	14	Identifies an edit that cancels an existing offset.

POFSE is used to record the value of an existing Parallel or Base Offset.

PFSTAT is used to record the present status of offsets. Synonyms PARALLEL, BASEOFST, and NOPOFS are used to indicate offset status.

With the exception of the Flight Plan Editor, POFSFE and PFSTAT are treated as read-only parameters. Any requirement to change the offset parameters is accomplished via a one-to-one replace edit with the desired change specified via the POFS component.

2.1.3 Navigation Global Tables

External parameters required for navigation computations are stored in the Navigation Global Tables. A single version of this file is used for all three Task II systems. The remaining portion of this section describes 3D/4D modifications to the Navigation Global Tables.

The ANS-70A version of the ADHD table has been expanded to include sensor inputs for Indicated Airspeed Error, Localizer Deviation, Glideslope Deviation, and HSI Course Input. A description of the ADHD components used for the 3D/4D systems is given below.

Symbolic Name	Component Position	Description	Validity Flag
LAT	0	Aircraft Latitude as maintained by the Lateral Filter.	N/A
LONG	1	Aircraft Longitude as maintained by the Lateral Filter.	N/A
PSIM	2	Magnetic Heading	PSMF
BALT	3	Barometric Altitude	BALF
TAS	4	True Air Speed	TASF
VRT	5	Vertical Rate (not input for the 3D/4D system)	N/A
PALT	6	Pressure Altitude (not input for the 3D/4D system)	N/A
DELV	7	Indicated Air Speed Error	DELVF
LOC	8	Localizer Deviation	LOCF
GS	9	Glideslope Deviation	GSF
HSICRS	10	HSI Course Input	HSICRSF

ANS-70A Lateral Deviation computations always reference the parent waypoint of the Lateral To-Event. Similar computations for the 3D/4D system use the leg intersection of the path actually being flown as the frame of reference. The following set of parameters is used for this task:

Symbolic Name	Description
TOLEG	Array used to record the coordinates of the leg intersection defined for the Lateral To-Event.
BFLEG	Work area used to record bearing from the leg intersection to the aircraft.
BTLEG	Work area used to record bearing from the aircraft to the leg intersection.
DTLEG	Distance between the aircraft and the oncoming leg intersection.

Parameters used to record ASC input discretes that are unique to the 3D/4D systems are given below.

Symbolic Name	Description
APENG	Status of the Auto-Pilot-Engaged discrete. APENG is assigned a TRUE status whenever the Area Nav system is coupled to the Auto-Pilot.
RNAVENG	Status of the RNAV-Engaged discrete. RNAVENG is assigned a TRUE status whenever the RNAV system provides lateral navigation and possibly lateral steering.
VNAVENG	Status of the VNAV-Engaged discrete. VNAVENG is assigned a TRUE status whenever the RNAV system provides vertical navigation and possibly vertical steering.

2.2 Base Offsets

Figure 2-6 illustrates the software interfaces that have been implemented for the Base Offset function. The key interface is provided by the Flight Plan Editor Utility Module. A description of this module will follow a brief summary of the roles played by the CDU Line-Select and Flight Plan Monitor modules.

Classification of a CDU scratch-pad entry as a candidate for a Base Offset is performed by the CDU software. Base Offset entries are permitted for waypoints that appear on the Flight Plan, Progress, or Speed pages. Valid candidates are passed onto the Flight Plan Editor Utility Module for additional processing. "ERROR" is annunciated on the scratch-pad if either of the following conditions invalidates the edit: range tests are failed, or the existing flight plan already includes a parallel offset.

Automatic cancellation of an existing Base Offset occurs following passage of the last waypoint associated with the given offset. This task is performed by the Flight Plan Monitor. The Base Offset is cancelled whenever waypoint passage occurs for the waypoint whose POFSF component has an assigned value of INTERCEP or END90.

The Flight Plan Editor Utility Module provides an interface between the CDU software and a modified version of the ANS-70A Flight Plan Editor. All flight plan revisions entered by the pilot are routed through this module. Most of the major software tasks required for the Base Offset function are provided by this module. The remaining portion of the section provides a description of tasks that have been implemented for the Flight Plan Editor.

A primary task consists of verifying geometry requirements for Base Offset entries. Figure 2-7 illustrates the basic geometries for Base Offsets.

Geometry requirements for the Two-Waypoint configuration are given as follows:

(1) $X = ||\mathrm{COCH}(W_{I-1})|-90°| \leq 2°$

and $|\mathrm{POFS}(\mathrm{WKPTR})| * (\pi X/180°) \leq 0.2$ NM

(2) $X = ||[\sum_{J=I-1}^{I} \mathrm{COCH}(W_J)|-180°| \leq 2°$

and $|\mathrm{POFS}(\mathrm{WKPTR})| * (\pi X/180°) \leq 0.2$ NM

and $Y = ||\mathrm{COCH}(W_I)|-90°| \leq 2°$

and $|\mathrm{POFS}(\mathrm{WKPTR})| * (\pi Y/180°) \leq 0.2$ NM

Geometry tests that must be performed for the Three-Waypoint configuration are given as:

(1) $X = ||\mathrm{COCH}(W_{I-1})|-90°| \leq 2°$

and $|\mathrm{POFS}(\mathrm{WKPTR})| * (\pi X/180°) \leq 0.2$ NM

(2) $X = ||[\sum_{J=I}^{I+1} \mathrm{COCH}(W_J)]|-90°| \leq 2°$

and $|\mathrm{POFS}(\mathrm{WKPTR})| * (\pi X/180°) \leq 0.2$ NM

and $Y = ||[\sum_{J=I-1}^{I+1} \mathrm{COCH}(W_J)]|-180°| \leq 2°$

and $|\mathrm{POFS}(\mathrm{WKPTR})| * (\pi Y/180) \leq 0.2$ NM

This set of tests is intended to provide protection against absurd offset geometries. If the geometry is found to be valid, POFS and POFSF components are updated for the offset waypoints and recorded in the Flight Event Table. If the geometry requirements are not met, control is returned to the CDU software. In either case, a return argument is used to indicate the status of the Base Offset entry.

A second major task consists of enforcing a set of operational rules that have been imposed on all flight plan revisions. The purpose of this set of rules is to insure that flight plan revisions do not invalidate the geometry for an existing Base Offset.

Special operational rules are required for DIRECT TO entries. As with the ANS-70A system, an existing Parallel Offset is cancelled whenever a valid DIRECT TO revision is made. A Base Offset, however, is cancelled only if the resulting flight plan revision affects the Base Offset waypoint configuration. A summary of the logic used to cancel Base Offsets for DIRECT TO entries is given in Table 2-1.

Operational rules that are implemented for remaining flight plan revisions are given as follows:

1) A Base Offset entry is to be inhibited whenever the current flight plan includes a Parallel Offset. The converse is also true, i.e., a Parallel Offset entry is not allowed for a flight plan that already includes a Base Offset.
2) The magnitude of an existing Base Offset may be revised at any time. A Parallel Offset may be overwritten at any time by entering another Parallel Offset into the flight plan.
3) No flight plan revision is allowed that would change the geometry of an existing Base Offset configuration.

Table 2-2 provides a summary of the logic that has been implemented for the above set of rules. PFSTAT, which appears in the table, is a parameter which is used to record current offset status. PFSTAT has three possible states: BASEOFST, PARALLEL, or NOPOFS. POFSF is an input parameter that is used to identify a flight plan revision. POFSF is assigned one of 4 possible states by the calling program: BASEOFST, PARALLEL, NOPOFS, or CANCEL. A description of this set of data parameters is given in Section 2.1.2.

The Utility Module provides four routines that are called by the CDU software. The ADD4D and REPLY4D routines are used for all flight plan revisions that occur on the Flight Plan, Speed, and Progress pages. The ADDDIR and REPDIR routines are called for DIRECT TO edits. Program listings for this set of routines are given in Appendix A.

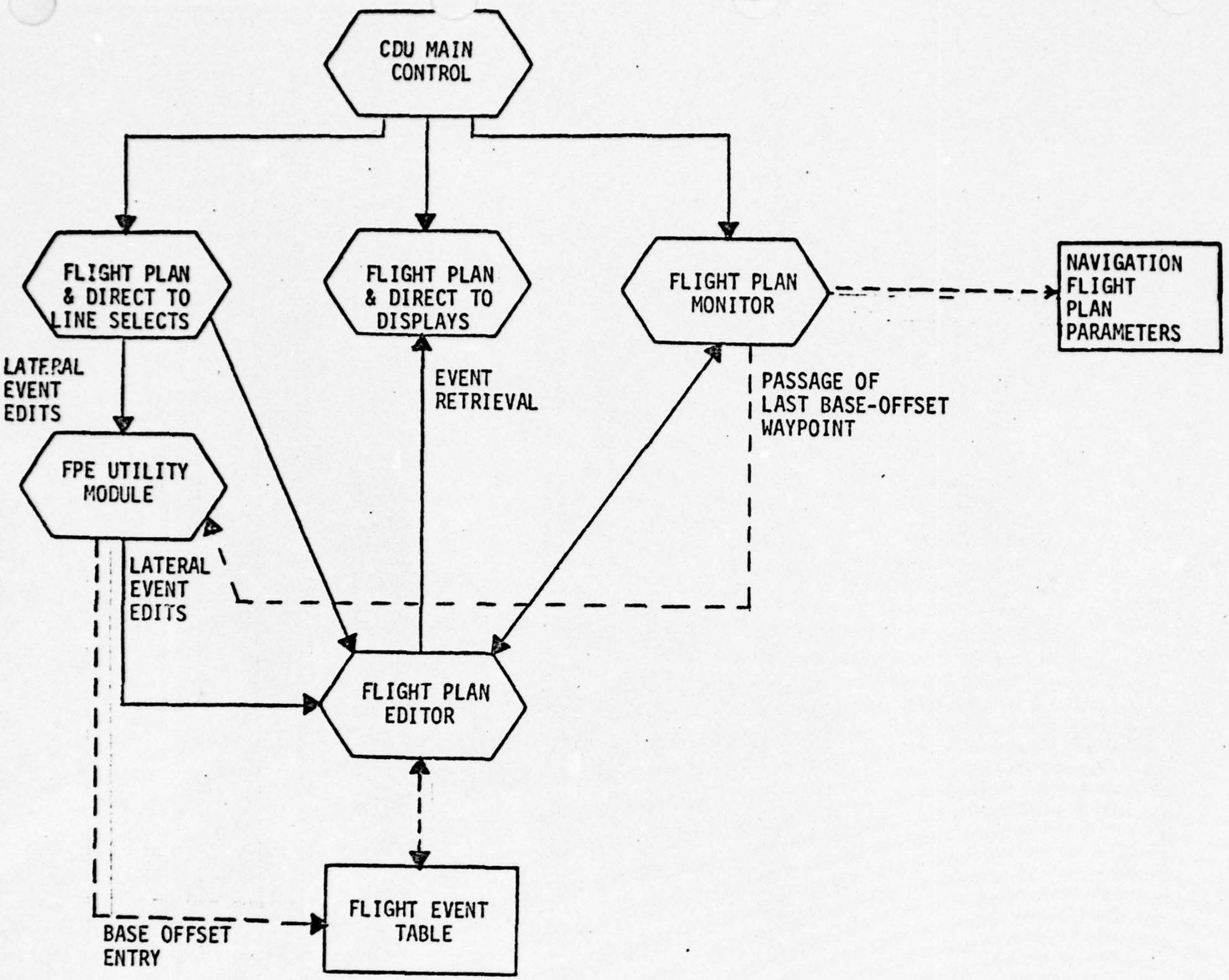


FIGURE 2-6
Software Interfaces for Base Offsets

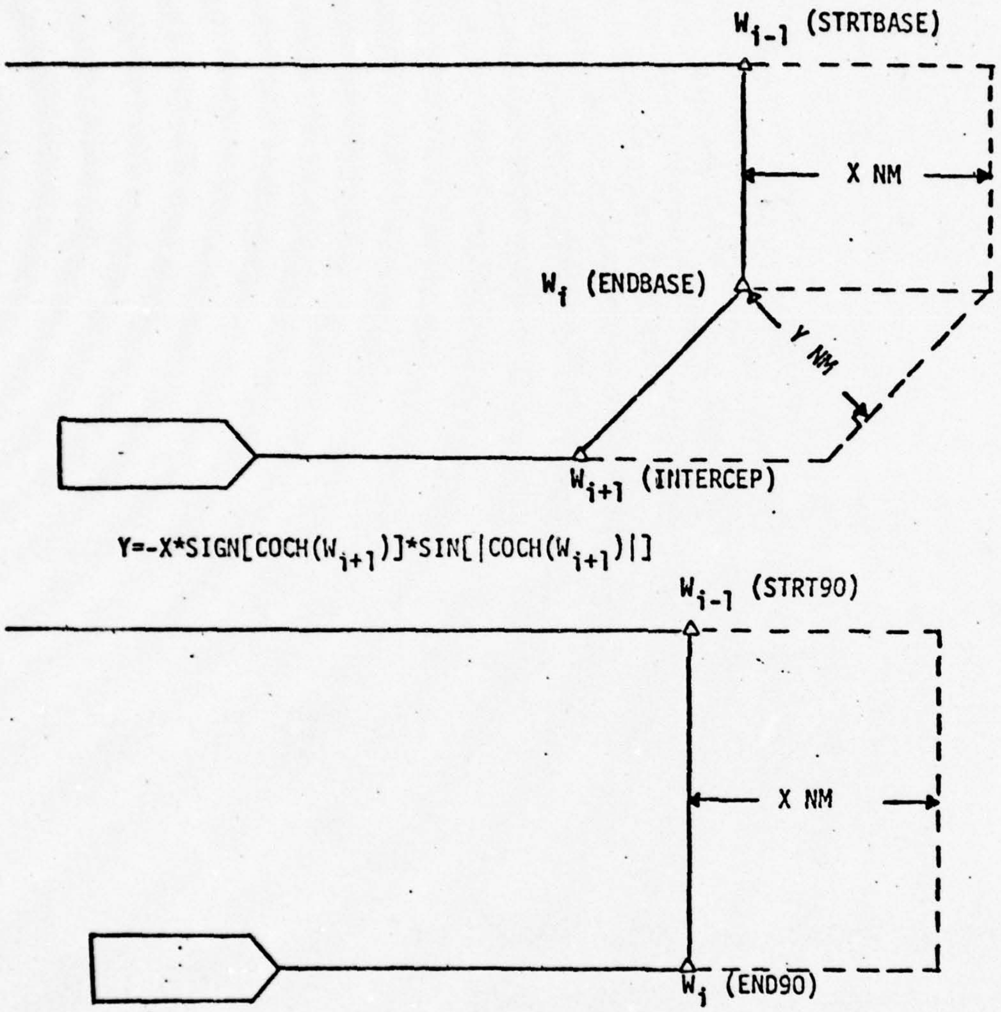


FIGURE 2-7

Basic Geometries for Base Offsets

Type of DIRECT TO	Requirements for cancelling an existing Base Offset
DIRECT TO the TO-Waypoint	The TO-Waypoint must be a Base Offset waypoint.
DIRECT TO the Scratchpad entry	The TO-Waypoint must be a Base Offset waypoint.
DIRECT TO a waypoint in the flight plan other than the TO-Waypoint	The flight plan must contain a Base Offset waypoint that is entered either at or before the "DIRECT TO waypoint."

TABLE 2-1

Operational rules used to cancel Base Offsets

for DIRECT TO entries

NP = NOPOFS
B = BASEOFST
P = PARALLEL
C = CANCEL

	DECISION LOGIC													
CONDITIONS	PFSTAT (CURRENT OFFSET CONFIG.)	NP	NP	NP	NP	B	B	B	B	B	P	P		E
	POFSF (DESIRED OFFSET CONFIG.)	NP	B	B	P	NP	NP	B	B	P	B	P	C	L
	EDIT TOUCHES BASE OFFSET					N	Y							S
	GEOMETRY CHECKS ARE SATISFIED		N	Y				N	Y					E
ACTIONS	CANCEL OFFSETS								X			X	X	
	ENTER NEW OFFSET			X	X				X			X		
	PASS EDIT TO FPE	X			X	X						X		
	MARK BASE OFFSET WAYPOINTS			X					X					
	EDIT DISALLOWED		X				X	X						X
	CDU PREVENTS OCCURENCE									X	X			

TABLE 2-2
Summary of Logic used to Implement Operational Rules for Flight Plan Revisions

2.3 Speed Profile Algorithm

Software interfaces required for implementation of the Speed Profile Algorithm are shown in Figure 2-8. An IAS entry on the Speed Page is recorded as a flight plan parameter in the Flight Event Table. A description of the IAS parameter was given in Section 2.1.2.

Speed profile computations are implemented within the CDU channel by a routine called SPDPRF. Computations are performed whenever a flight plan revisions occurs.

Special logic is required for the From-Waypoint. This waypoint is always treated as a speed-control waypoint whenever the flight plan contains an IAS entry. This is done to ensure that a speed profile presently being flown will not be changed at waypoint passage.

The task description for SPDPRF is given below. A diagram illustrating the use of program parameters is given in Figure 2-9. A program listing of SPDPRF is given in Appendix B.

Task Specification for SPDPRF

Procedure Type

Integer Procedure SPDPRF

Module

3D/4D Time and Speed Algorithms

Task Summary

Speed profile computations are performed for all lateral events that are not considered to be speed-control waypoints. (Speed-Control waypoints are waypoints for which speed has been entered via the CDU.) The lateral FROM-event is treated as a speed-control waypoint to ensure that a speed profile presently being flown will not change at waypoint passage. Results of profile computations are recorded in the IASR/IASI/IASF component structure for lateral events.

Notation

W(J)	Used to denote the Jth lateral event
IASR(J)	Used to denote the value of the IASR component for W(J). Similar notation is used for other components defined by the Flight Plan Editor.

Internal Data Structures

Integer Data

FRST	Absolute Lateral Event Number Index for the initial speed-control waypoint of a given profile.
LAST	Absolute Lateral Event Number Index for the final speed-control waypoint of a given profile.
FRST.STATUS	Used to record the validity of FRST for a given profile.
LAST.STATUS	Used to record the validity of LAST for a given profile.
J	Absolute Event number of the Jth lateral event.

Real Data

PROFDST	The along-track distance between W(FRST) and W(LAST).
DVEL	The velocity change between W(J-1) and W(LAST).
DJ	The along-track distance between W(J-1) and W(J).
DELDIST	The along-track distance between W(J) and W(LAST).
T1	The time required to fly from W(J-1) to W(LAST) at a rate of 40 knots/minute.
AVEV	The average velocity required for a linear speed profile between W(J-1) and W(LAST).
T2	The time required to fly from W(J-1) to W(LAST) assuming a constant velocity of AVEV.
DPP	The distance required to realize a velocity change of DVEL at the rate of greater than or equal to 40 knots/minute.

Calling Sequence

SPDPRF()

Return Values

Speeds assigned to intermediate profile points are recorded in the Flight Plan Editor's FET structure.

Task Description

```
IF a flight plan revision has occurred since the last speed profile computation
THEN
BEGIN     ... SPDPRF loop.
    Starting with the FROM-waypoint, scan the flight plan for a speed entry.
    IF no speed entry has been found
    THEN
    BEGIN     No Speed Entry loop.
        Update bookkeeping to indicate that another attempt to compute speed
        profiles is not required for the present flightplan (this is intended
        to assist CDU keyboard response time).
```

```
     RETURN to calling program.
END  No Speed Entry loop.
Assign the first speed to the FROM-waypoint.
Tag the FROM-waypoint as FRST.
Assign VALID to FRST.STATUS.
WHILE FRST.STATUS EQL VALID DO
BEGIN     ...FRST Valid loop
     Assign INVALID to LAST.STATUS
     Starting with W(FRST+1) scan the flight plan for a speed-control
     waypoint.
     IF the speed-control waypoint is found
     THEN
          BEGIN     ...LAST Valid loop
               Assign VALID to LAST.STATUS.
               Tag the speed-control waypoint as LAST.
               Determine PROFDST, the along-track distance between W(FRST)
               and W(LAST).
               IF LAST NEQ (FRST+1)      ...Note that speed profile computations,
               i.e., the Profile loop, will be bypassed if successive lateral
               events are defined as speed-control waypoints.
          THEN
          BEGIN     ...Profile loop.
               IF the speed for W(FRST) and W(LAST) are euqal
               THEN
               BEGIN     ...Same.Speed loop.
                    FOR J=(FRST+1) STEP 1 UNTIL LAST DO
                    BEGIN     ...Assign Same Speed loop
                         Assign the speed for W(FRST) to W(J).
```

```
        $IASR(J)=SQRT[V_0^2+(V_F^2-V_0^2)*\frac{D}{DPP}]$

            Where:

                D=DJ
                $V_0$=IASR(J-1)
                $V_F$=IASR(LAST)
                DPP=DJ+DELDIST

        Assign either PRUP or PRDOWN to the IASF component
        of W(J) for CDU display purposes.

END         ...Linear loop

ELSE        ...At this point it has been determined that an
            average speed rate less than 40 kts/min is required
            to fly from W(J-1) to W(LAST).

            IF an average rate of less than 40 kts/min is also
            required to fly from W(J) to W(LAST), assign
            IASR(J-1) to IASR(J).

BEGIN       ...40 Kts/Min loop

        Determine DPP, the distance required to realize a
        velocity change of DVEL at the rate of 40 Kts/Min,
        i.e.,
        DPP=T1*AVEV

        IF the distance, DELDIST, between W(J) and W(LAST)
        is greater than or equal to DPP

        THEN

        BEGIN       ...No Change loop

            Assign IASR(J-1) to IASR(J) and
            PRQUOTE to IASR(J).

        END         ...No Change loop

        ELSE

        BEGIN       ...Min Rate Zone loop

            Calculate a velocity for W(J) that is based on
            a 40 kt/min velocity rate between W(J-1) and W(J),
            i.e.,

            $IASR(J)=SQRT[V_0^2+(V_F^2-V_0^2)*\frac{D}{DPP}]$
```

```
                                Where
                                    $V_0$=IASR(J-1)
                                    $V_F$=IASR(LAST)
                                    D=DPP-DELDIST
                            Assign either PRUP or PRDOWN to IASR(J-1).
                        END         ...Min Rate Zone loop
                    END         ...Min Rate Zone loop
                END         ...Velocity Change loop
            END         ...Profile loop
            Define parameters for next cycle of FRST.VALUE loop:
            FRST=LAST.
            FRST.STATUS=LAST.STATUS.
            LAST.STATUS=INVALID
        END         ...LAST Valid loop
    END         ...FRST Valid loop
    Update bookkeeping to indicate that speed profile computations have been performed for the existing flight plan.
END         ...SPDPRF loop
RETURN to calling program.
```

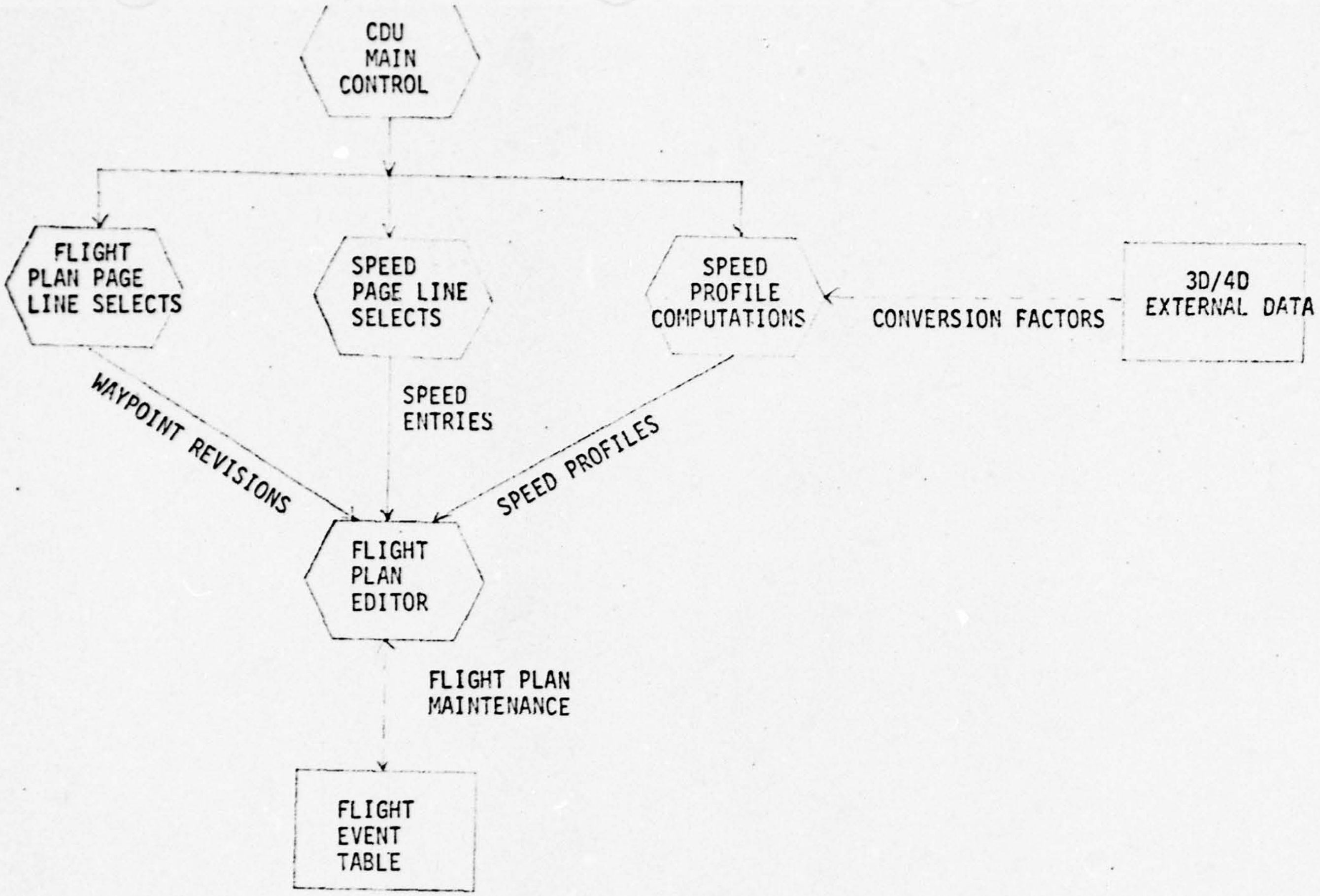


FIGURE 2-8

Software Interfaces for Speed Profiles

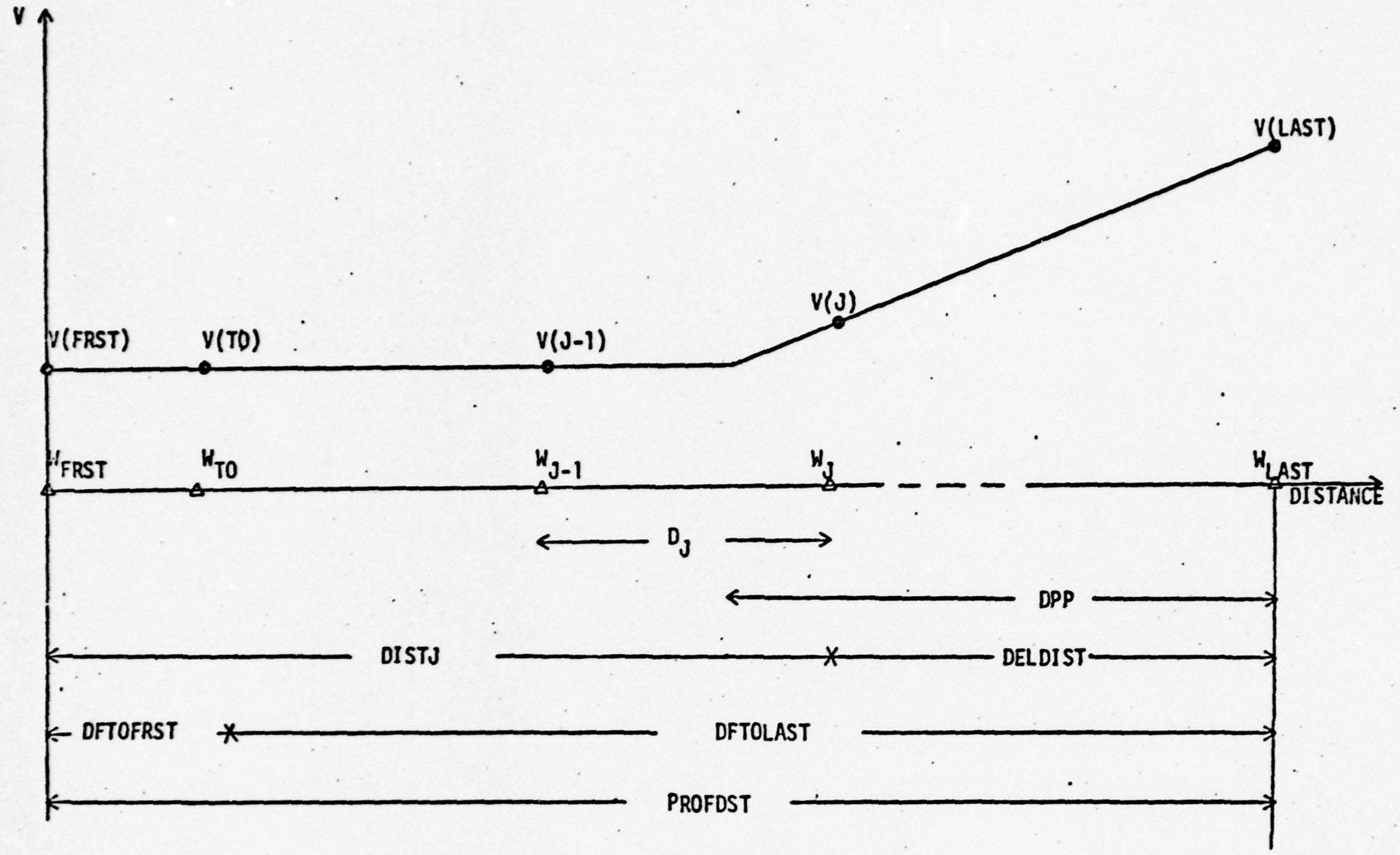


FIGURE 2-9

Description of Program Parameters used to implement the Speed Profile Algorithm

2.4 Time Control Algorithm

Figure 2-8 illustrates the basic software interfaces required for maintenance of the Time Control parameters. Data declarations for the Time Control Parameters were given in Section 2.1.1. With the exception of aircraft sensor data, all parameters are maintained by procedures that are called by the CDU Main Control procedure. The Time Fix and Commanded Arrival Time are CDU entries made on the Flight Progress Page. The same page is used to display the software computations for Commanded IAS and Arrival Time Error (Early/Late Indication). The Flight Plan Monitor monitors waypoint passage of the Time-Fix to determine the necessity of invalidating algorithm computations.

2.4.1 CDU Display of Time Control Parameters

The last two line pairs of the Flight Progress Page are used for the display of Time Control parameters. Data declarations for this set of parameters are given in Section 2.1.1. Line pair #5 provides information for the Time-Fix, TFIX, Commanded Arrival Time, TCOM, and Commanded Air Speed, VCOMI. The validity flags TFIXF and TCOMF are monitored to determine the need to display blank data fields for TFIX and TCOM, respectively. The value stored in VCOMI is displayed for the Commanded Air Speed data field whenever VCOMIF has an assigned value of IASVALID.

Line pair #6 is used for the Early/Late and Navigation Downmode displays. The latter function is identical to that defined for the ANS-70A system. Values for the Early/Late display are recorded in TIMERR and EARLY. The EARLY label is displayed whenever EARLY is valid and negative; LATE is displayed for a valid positive value. The value recorded in TIMERR is used for display of the Early/Late data line. No display is required for the Early/Late label and data lines whenever the validity flag TIMERF is assigned an INVALID status.

2.4.2 CDU Line-Select Operations

Line pair #5 for the Flight Progress Page provides the pilot with the capability of designating a waypoint as the Time-Fix and a Commanded Arrival Time for this waypoint. The two data fields used to enter this information may be entered (or revised) separately or simultaneously.

Rules governing line-select entries are as follows:

TIME Entries:

1) The same range checks used for GMT entries on the Present Position Page apply to the Time Entry.
2) The value of the Time Entry is recorded in TCOM and the associated validity flag, TCOMF, is assigned a VALID status.

WAYPOINT Entries:

1) The waypoint must be stored in the Flight Plan prior to the WAYPOINT entry on the Flight Progress Page.
2) A valid entry is recorded in TFIX. TFIXF is assigned a VALID status to indicate that a Time-Fix has been designated.

A single scratch pad entry of a minus (-) sign is used to invalidate both fields. An INVALID status is assigned to both TFIXF and TCOMF whenever this entry is made.

2.4.3 Functional Description of Time Control Algorithm

The description of the Time Control Algorithm uses the following notation:

i	Subscript used to represent the i^{TH} waypoint. The following values of i will be of special significance: $i = 0$ For the FROM-WAYPOINT $= 1$ For the TO-WAYPOINT $= T$ For the waypoint designated as the time-fix.
W_i	The i^{TH} waypoint.
CRS_i	The course into W_i as given by the flight plan.

$WATK_i$	The along-track component of wind for W is $WATK_i$ = -(NWDN* COS(CRS_i) + EWND * SIN(CRS_i)), where NWND and EWND represent present lateral filter estimates for north and east wind components, respectively. $WATK_1$ denotes the present along-track component of wind.
$\overline{VNOM}$	The nominal value for present commanded TAS. $\overline{VNOM}$ is a function of $\overline{V}_0$, $\overline{V}_1$, the time estimated to fly the along-path distance between W_0 and W_1, and the actual distance from the aircraft to the TO waypoint.
VCOM	The present commanded IAS of the aircraft. VCOM represents the commanded IAS displayed on the Flight Progress Page.
$\overline{VCOM}$	The present commanded TAS of the aircraft.
$\overline{VACT}$	Air data sensor input for present aircraft TAS.
VACT	Present aircraft IAS as given by the speed and time control algorithm.
Vi	The IAS (Indicated Air Speed) associated with Wi.
$\overline{V}i$	The TAS (True Air Speed) associated with Wi.
$\overline{TTGCOM}_i$	The commanded (actual) time required to reach Wi. In particular, the commanded time required to reach the desired time fix, $\overline{TTGCOM}_T$, is given by $\overline{TTGCOM}_T$=(Time commanded to arrive at the time fix) - (Present Time).
$\overline{TTGNOM}_i$	The nominal time required to reach Wi. $\overline{TTGNOM}_i$ is a function of IAS entries and flight path geometry.
D_i	The leg distance between W_{i-1} and Wi. D_1 is defined as the actual distance from the aircraft to the To-Waypoint.
D_i'	That portion of D_i that is to be flown at a constant speed.
D_i''	That portion of D_i that requires a velocity change.

An important concept of the Task II Speed Control Philosophy is the utilization of minimum 40 knot/minute speed change profiles. Pilot reaction to speed changes is not required until a minimum speed change of 40 knots/minute is needed to reach a desired airspeed. This means that the speed profile for any given leg may be presented as one of the two situations depicted in Figure 10.

Ignoring wind, let K denote the proportionality factor that represents the proportion of nominal speed that is required to arrive at the time fix on schedule (K=1 if no change in the nominal profile is required to meet the desired schedule).

Then $\overline{VCOM} = K*\overline{VNOM}$ where K is to be determined from the relation:

$$K = \overline{TTGNOM_T}/\overline{TTGCOM_T}$$

As noted before, $\overline{TTGCOM_T}$ is merely the difference between the time (GMT) commanded to arrive at W_T and the present time (Present value of GMT). $\overline{TTGNOM_T}$ is given by summing the nominal times required to fly all legs until W_T is reached, i.e.,

$$\overline{TTGNOM_T} = \frac{MTG'}{\overline{V}_0+WATK_1} + \frac{MTG''}{(\overline{V}_0+\overline{V}_1)/2+WATK_1}$$

$$+ \sum_{i=2}^{T} \left(\frac{D_i'}{\overline{V}_{i-1}+WATK_{i-1}} + \frac{D_i''}{(\overline{V}_{i-1}+\overline{V}_i)/2+WATK_{i-1}}\right).$$

In the above expression, MTG=MTG'+MGT" is the actual distance from the aircraft to the To-Waypoint. MTG' and MTG" are analogous to D_i' and D_i'', respectively.

Having found K, the value of the commanded TAS for the present leg is given by $\overline{VCOM}=K*\overline{VNOM}$. The best available along-track component of wind for the present leg is given by using the Lateral Filter's Estimates for north and east wind components. This dynamic computation for along-track wind cannot be proportionately modified for the remaining portion of the present leg. At best, it may only be used to directly modify the dynamic computations for $\overline{VCOM}$ and $\overline{VNOM}$. Thus,

$$\overline{VCOM} + W = K*(\overline{VNOM} + WATK_1) \text{ or}$$

$\overline{VCOM} = K*\overline{VNOM} + WATK_1*(K-1)$, where $WATK_1$ represents the present component of along-track wind.

$\overline{\mathrm{VCOM}}$ represents the present commanded true air speed that is necessary to arrive at W_T on time. The corresponding commanded IAS is computed by the relationship

$$\mathrm{VCOM} = (1-K_1 h) * \overline{\mathrm{VCOM}}$$

The value used for K_1 is the value relating present IAS to present TAS and is computed as follows. The present aircraft IAS (VACT) is not directly available to the computer but can be computed from the IAS command (VCOM) and IAS airspeed error (ΔV) signals which are available from the max allowable airspeed instrument, i.e.,

$$\mathrm{VACT} = \mathrm{VCOM} + \Delta\ \mathrm{V}.$$

Present values for aircraft TAS ($\overline{\mathrm{VACT}}$) and altitude (h) are resident in the RNAV computer memory. Using the relationship

$$\mathrm{VACT} = \overline{\mathrm{VACT}}\ (1 - K_1 h),$$

the altitude proportionality constant, K_1, may then be determined by combining the above equations, i.e.,

$$(1-K_1 h) = (\mathrm{VCOM} + \Delta \mathrm{V})/\overline{\mathrm{VACT}}$$

$$\text{and } K_1 = (\overline{\mathrm{VACT}}-\mathrm{VCOM}-\Delta \mathrm{V})/(h*\overline{\mathrm{VACT}})$$

With h in feet, K_1 should vary from 0.00001 to 0.000015. Because of sensitivity problems as h becomes small, K_1 is to be limited to the above values in the IAS to TAS speed conversions. In the TAS to IAS conversion regarding the commanded velocity the $(1-K_1 h)$ ratio can be used directly, avoiding any sensitivity problem.

The Commanded Speed displayed on the Flight Progress Page is then given by:

$$\mathrm{VCOM} = (1-K_1 * h) * \overline{\mathrm{VCOM}}$$

The final portion of the algorithm deals with the early/late computation. This computation is based upon the expected time error at the time-fix that would result by flying the present leg at the actual rather than commanded TAS.

As before, let:

$\overline{VCOM}$	Represent the present value for commanded TAS.
$\overline{TTGCOM}_T$	Represent the commanded time required to reach W_T.
$\overline{VACT}$	Represent the present air data sensor input for TAS.

Also, let:

D_T	Represent the modified along-track distance from the aircraft to W_T. The modification consists of using actual distance from the aircraft to the To-waypoint for the present leg.
$\overline{TTGACT}_T$	Represent the time required to traverse D_T assuming a TAS of $\overline{VACT}$.

Since the Distance, D_T, must be flown to arrive at W_T using either velocity, it follows that:

$$D_T = (\overline{VACT} + WATK_1) * \overline{TTGACT}_T$$

$$= (\overline{VCOM} + WATK_1) * \overline{TTGCOM}_T$$

The expected time error, $\overline{TTGAC\ }_T-\overline{TTGCOM}_T$ is then given by:

$$(\overline{TTGACT}_T-\overline{TTGCOM}_T) = \overline{TTGCOMT}_T * \left[\frac{(\overline{VCOM} + WATK_1)}{(\overline{VACT} + WATK_1)} -1\right]$$

where a positive error is to be displayed as a LATE indication.

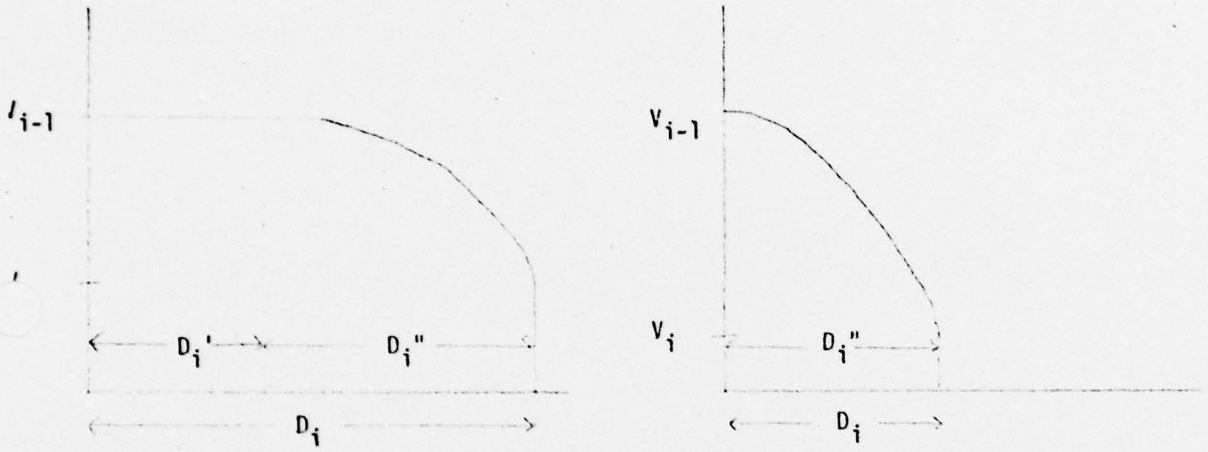


FIGURE 2-10

Speed Profiles

$D_i"=0$ whenever $V_{i-1}=V_i$ and $D_i'=0$ whenever a speed change in excess of 40 KTS/Min is required for the entire leg distance D_i.

2.4.4 Software Implementation of the Algorithm

Algorithm computations are performed by procedure TIMCNTRL. Although control is given to this procedure every cycle by the CDU Main Control Program, computations are only attempted whenever either 1) ten seconds have elapsed since the last algorithm computation, or 2) a Flight Plan revision has occurred since the last execution of the algorithm.

Since TIMCNTRL is a procedure called within the CDU channel, it is possible for sensor data maintained by the Navigation channel to change during any given execution of the algorithm. To ensure that the same set of parameters is used for a given algorithm computation, it is necessary to take a snapshot of the sensor data as part of the initialization logic for the algorithm.

A task description for TIMCNTRL is given below. A program listing of TIMCNTRL is given in Appendix C.

Task Specification
For TIMCNTRL

Procedure Type

Procedure TIMCNTRL( )

Module

4D Time and Speed Algorithms

Task Summary

The Time and Speed Control Algorithm is based upon speed and time errors with respect to a waypoint (the Time-Fix) for which an arrival time has been assigned. The value calculated for VCOMI, the Commanded Velocity displayed on the Flight Progress Page, represents the present indicated air speed that should be flown to arrive at the Time-Fix on schedule. The computation for TIMERR provides the best estimate of the total time error that is expected for the desired arrival time. Computations for VCOMI are provided whenever the Present Flight Plan, exclusive of the From-event, contains at least one speed waypoint. The computation for TIMERR is made whenever the validity flag WFIXF indicates that a Time-Fix has been defined for the system.

Notation

W(J)	Used to denote the Jth Lateral Event.
IASR(J)	Used to denote the value of the IASR component for W(J). Similar notation is used for other components defined by the Flight Plan Editor.

Internal Data Structure

Synonyms

INITEVNO	Initial Absolute Event Number required for retrieval of event data from the Flight Plan Editor.

Integer Data

J	Event Number for the Jth Event.
CNTRL	Control parameter for the main loop.
VTIMF	Flag used to control computations for the case where a Time-Fix has not been entered but VCOMI is to be calculated for the To-waypoint.

Real Data

PREVIASR	Temporary storage used to record IASR(J-1).
PREVDFTO	Temporary storage used to record DFTO(J-1).
TTGACT	Present estimate of actual time required to reach the Time-Fix.
TTGCOM	Commanded time to reach the Time-Fix.
COCHPREV	Temporary storage used to record CRSO(J-1)-CRSI(J-1).
HZ	Altitude used for IAS to TAS conversion.
LEGDST	Along-track distance from W(J-1) to W(J).
WATK	The along-track component of wind applied to W(J).
WATKTO	The along-track component of wind applied to the leg presently being flown.
KALT	The altitude proportionality constant used for conversions between IAS and TAS.
TASPREV	Temporary storage used to record computed TAS for W(J-1).
TASJ	Temporary storage used to record computed TAS for W(J).
DV	The speed change between W(J-1) and W(J).
T1	The time required to fly from W(J-1) to W(J) at a rate of 40 kts/min.

AVEV	The average ground velocity required for a linear speed profile between W(J-1) and W(LAST).
T2	The time required to fly from W(J-1) to W(J) assuming a constant velocity of AVEV.
DPP	The portion of the leg between W(J-1) and W(J) that requires a speed change.
DELT	The expected time required to fly from W(J-1) to W(J).
DP	The portion of the leg between W(J-1) and W(J) that should be flown at the velocity assigned to W(J-1).
GMTCONV	Conversion factor used to convert GMT to navigation time units.
VGNOM	Nominal ground velocity calculated for the present leg.
KSCALE	Scaling factor used to protect against overflow.

Calling Sequence

TIMCNTRL( )

Return Values

VCOMI	Commanded IAS for the present leg that is to be displayed on the Flight Progress Page and IAS instrument.
VCOMIF	Validity flag for VCOMI.
TIMERR	Expected time error for arrival at the Time-Fix.
TIMERF	Validity flag for TIMERR.

Task Description

```
IF 10 seconds have not elapsed since the last execution of the Time Control Algorithm
    AND a Flight Plan revision has not occurred since the last cycle
    THEN RETURN to calling program.
Update own clock reading.
Update own version of Flight Plan Revision number
Assign INVALID status to TIMERF, the validity flag for arrival time error.
Comment:  assign an initial validity status to VCOMIF, the validity flag for
          commanded IAS.
IF VCOMIF=
    IF invalid GMT THEN IVGMT
    ELSE IF invalid aircraft true air speed THEN IVTAS
    ELSE IF invalid aircraft altitude THEN IVALT
    ELSE NODATA) NEQ NODATA
```

```
HEN RETURN to calling program.

F the Lateral From-event is invalid

    OR no speed is defined for this event

    THEN RETURN to the calling program.  Note that VCOMIF will have a value of
        NODATA and TIMERF will have an INVALID STATUS.

        Comment:  Keep a snapshot of sensor data, i.e.,

        ACTAS = Aircraft True Air Speed
        ACALT = Aircraft Altitude
        ACIASERR = IAS Instrument Error
        ACNWND = North Component of Wind
        ACEWND = East Component of Wind

ssign IASR component of the Lateral From-Event to PREVIASR.

ssign DFTO(FROM) to PREVDFTO.

nitialize event number index by assigning INITEVNO to J.

nitialize nominal time-to-go, TTGNOM, to 0.0.

nitialize COCHPREV  to value of CRSO(FROM)-CRSO(FROM).

ssign an initial status of VALID to VTIMF to ensure computations are performed
or TO-waypoint.

HILE the Jth event is not the Time-Fix.

     AND the (J=J+1)th event is valid.
     AND VTIMF EQL VALID.

 GIN      ... Calculate TTG block.

    IF event J is a Vertical event

    THEN

    BEGIN     ...Vertical Event block

         IF event J is the Vertical TO-event

         THEN assign value of ACALT to HZ.

         ELSE assign ALTD(J) to HZ.

    END       ...Vertical Event block

    ELSE

    IF event J is a Lateral event      ...Note that End-Of-Continuity events
    are bypassed.
```

```
THEN

BEGIN     ...Lateral Event block

     Calculate LEGDST, the distance from W(J-1) to W(J), i.e.,

          LEGDST=DFTO(J)-PREVDFTO.

          Update PREVDFTO to value of DFTO(J).

          Calculate WATK, the along-track component of wind, i.e.,

          WATK=-(ACNWND*COS(CRSI(J))+
                 ACEWND*SIN(CRSI(J))).

          Calculate KALT, the altitude proprotionality constant, i.e.,

          $KALT=\frac{[ACTAS-VCOMI-ACIASERR]}{[HZ*ACTAS]}$

          Limit KALT to the interval (.00001, .000015)

     Convert IAS for W(J-1) to TAS, i.e.,

          $TASPREV=\frac{PREVIASR}{(1.0-HZ*KALT)}$ .

     Convert IAS for W(J) to TAS, i.e.,

          $TASJ=\frac{IASR(J)}{(1.0-HZ*KALT)}$ .

     Calculate DV, the speed gradient between W(J-1) and W(J), i.e.,

          DV=TASJ-TASPREV .

     Calculate T1, the time required to fly from W(J-1) to W(J) at a rate
     of 40 kts/min.

          T1=|DV*MINRATE| .

     Calculate AVEV, the average ground velocity required for a linear
     speed profile between W(J-1) and W(J), i.e.,

          AVEV=0.5(TASPREV+TASJ)+WATK

     Calculate T2, the time required to fly from W(J-1) to W(J) assuming a
     constant velocity of AVEV, i.e.,

          T2=|LEGDST/AVEV|

     IF the velocity change, DV, between W(J-1) and W(J) requires a
     velocity rate greater than or equal to 40 kts/min (i.e., if $T2 \leq T1$)

     THEN

     BEGIN     ...Linear Profile Block

          Assign T2 to DELT, the time required to fly the leg from W(J-1)
          to W(J).
```

```
    Assign LEGDST to DPP, the portion of the leg into W(J) that
    requires a speed change.
END       ...Linear Profile Block
ELSE      ...Part of the leg into W(J) should be flown at the velocity
          for W(J-1), the remaining portion at a velocity rate of 40
          kts/min. Calculate time and distance parameters for the
          portion requiring a speed change.
BEGIN     ...40 kts/min Block
    Assign T1 to DELT.
    Assign T1*AVEV to DPP.
END       ...40 kts/min Block
IF W(J) is the Lateral To-Event
THEN
BEGIN     ...To-Event Block
    Assign IASVALID to VCOMIF to indicate VCOMI computation has been
    performed.
    Assign value of TFIXF to VTIMF.
    Save the along-track wind component for the Lateral To-Event,
    WATK, in WATKTO.
    Save value of KALT in KALTTO.
    IF the velocity change, DV, between W(J-1) and W(J) requires
    a velocity rate that exceeds 40 kts/min (i.e., if $T2 \le T1$)
    OR the distance from the aircraft to the Lateral To-Event is
    within that portion of the leg that requires a velocity change
    (i.e., if $MTG \le DPP$)
    THEN
    BEGIN     ...To-Event Velocity Change Block
        Calculate VGNOM, the present nominal aircraft velocity (GS-
        referenced at this point) that must be flown to arrive at
        the Time-Fix on schedule, i.e.,
        $V_0$=TASPREV+WATK
        $V_F$=TASJ+WATK
        VGNOM=SQRT[$V_0^2+(V_F^2-V_0^2)*\frac{MTG}{DPP}$]
```

```
                    Calculate DELT, the time required to fly from aircraft
                    present position to W(J) if the aircraft is on schedule,
                    i.e.,

                    IF DMTG > DPP
                    THEN DELT = T2 + (DMTG > DPP)/ VGNOM
                    ELSE
                        $DELT=\frac{2*MTG}{(VGNOM+TASJ+WATK)}$

                    END       ...To-Event Velocity Change Block

                    ELSE      ...Velocity change is not required for present
                              portion of leg into the Lateral To-Event.

                    BEGIN     ...To-Event constant velocity block.

                        Assign velocity for W(J-1) to VGNOM, i.e.,

                        VGNCM=TASPREV+WATK

        BEGIN     ...Excessive Course Change block.

            Tag commanded IAS display as invalid by assigning IVCOCH to
            VCOMIF.

            RETURN to calling program.  Note that TIMERF is returned with
            an INVALID status and VCOMIF indicates that "CRS"! will appear
            for the commanded IAS display on the Progress Page.

        END       ...Excessive Course Change block.

        ELSE

        BEGIN     ...Leg Capture Compensation block.

            Compensate for aircraft turn made for leg capture at W(J-1), i.e.,

                $DELT=\frac{TASPREV*[|COCHPREV|-2*TAN(\frac{|COCHPREV|}{2})]}{g*TAN(25°)}$

                    where g=Acceleration due to gravity.

            Increment TTGNOM by DELT.

        END       ...Leg Capture Compensation block.

    END       ...Beyond To-Event block.

    Update COCHPREV to value of

        COCHPREV=CRSO(J)-CRSI(J)

D       ...Lateral Event block.

D       ...Calcttg Block
```

```
Calculate      1)  K, the proportion of nominal speed that is required to arrive at
                   the Time-Fix on schedule, and
               2)  TTGCOM, the commanded Time-To-Go for arrival at the Time-Fix, i.e.,

IF TIMERF = (TCOMF .V. TFIXF) NEQ VALID
        ...Computations for TIMERR, K, and TTGCOM are dependent upon status of pilot
           entries for Time-Fix and Commanded Arrival Time

THEN

     BEGIN   ...TTGCOM Invalid Block

          K = KSCALE     ...Reversionary Mode.  Set K = (1) * (Scale Factor).

          KVAL = VALID  ...Indicate K valid for later computation of VCOMI.

     END  ...TTGCOM Invalid Block

ELSE

     BEGIN   ...TTGCOM Valid Block

          Update Commanded Arrival Time, I.E.,

               TTGCOM = (TCOM-GMTS)/GMTCONV

          Determine validity of computation for K.  K is invalid when TTGCOM approaches
          zero, I.E.,

          IF TTGNOM*KSCALE  >  |TTGCOM|

          THEN Assign an INVALID to KVAL

          ELSE

               BEGIN

                    K = (TTGNOM*KSCALE)/TTGCOM

                    KVAL = VALID

               END

     END  ...TTGCOM Valid Block.

IF VCOMIF EQL IASVALID   ...Computations for VCOMI and TIMERR only valid if To-Event
                            has a defined IAS.

THEN

     BEGIN     ...VCOMIF Valid Block

          IF KVAL EQL VALID    ...Test for validity of K before updating VCOMI.

          THEN

               BEGIN    ...Update VCOMI Block
```

```
                Calculate Commanded TAS, I.E.,
                    VCOMT = (K*VGNOM)/KSCALE-WATKTO.
                Convert VCOMT to IAS for display purposes, I.E.,
                    $VCOMI = \frac{1.0-KALTTO*ACALT}{KALTSCALE} *VCOMT$
                Limit VCOMI to upper limit of IAS Instrument.
            END  ...Update VCOMI Block
        Calculate the expected time error for arrival at the Time-Fix, I.E.,
        IF TIMERF EQL VALID
        THEN
            BEGIN     ...Update TIMERR
                EARLY = (TTGNOM*VGNOM)/(ACTAS+WATKTO)-TTGCOM.
                TIMERR = $|EARLY|$.
            END  ...Update TIMERR
    END  ...VCOMIF Valid Block
RETURN to Calling Program
```

2.5 ASC Software Modifications

This section describes the 3D/4D modifications to the ANS-70A version of the Aircraft Systems Coupler (ASC) software. One version of this software has been constructed for use by all three of the Task II systems. In addition to changes implemented for the Base and 2D Plus Time Control Systems, this section will describe ASC software modifications that have been added for the 3D/4D Localizer/Glideslope System.

Table 2-3 lists the sensor data inputs that require additional processing by the ASC Input software. The first three items are required for the Localizer/Glideslope System. Each input has an associated high-level flag which is used to indicate current hardware status for the given sensor input. This status serves as one of the criteria used to determine validity of the sensor input.

Due to radio receiver limitations, localizer frequency is input in the same sensor slot normally used for the secondary VOR frequency. In addition to processing localizer frequency, this requires the ASC software to suppress system usage of the secondary navaid during the localizer mode of navigation. Use of this navaid is inhibited by setting a flag that is normally used to indicate a frequency mismatch.

A software switch whose value is specified for the Link Edit Computer run is used to determine which of two sets of computations are required for processing of IAS Airspeed Error. Computations for the G-I instrument are linear for the range of speeds that have been defined for the instrument. A Least-Squares approximation on the data furnished for the C-880 instrument has resulted in the use of a fourth-degree polynomial to determine IAS error.

The sensor word reserved for the HSI Input for Selected Course is processed every cycle by the ASC Input Program. The 2D Plus Time Control System version of the Flight Plan Monitor is the only program for the three systems that monitors this processed data. The net result is that the ASC software is allowed to process the input for all three Task II systems.

Current status of the "Engaged" discretes is recorded for system usage. The Lateral Navigation program suppresses waypoint passage whenever the Area Navigation System is disengaged from the auto-pilot. The other two discretes are used by the Localizer/Glideslope System.

Table 2-4 provides a summary of sensor data outputs that require additional processing by the ASC Output Program. IAS Command is the only output that is required for all three systems. As with the input for IAS error, conversions for the C-880 instrument are straight-forward. A fourth-degree polynomial is used to convert IAS in knots to degrees for the C-880 instrument.

Remaining outputs are required for the Localizer/Glideslope system. Deviation displays are sent to the HSI and ADI instruments. The ASC Output Program monitors internal flags to determine which navigation modes are controlling the instrument displays.

The Localizer Enable discrete is used to activate tuning the VOR/LOC receiver to localizer frequencies. Once again, internal flags are used to determine the setting of this discrete.

The "Arm" and "Capture/Track" discretes are used to control displays on the Mode Annunciator that has been implemented for Localizer/Glideslope usage. Internal flags are monitored each cycle to determine the settings of these discretes.

One additional feature has been added to the ASC Software that is unique to the 3D/4D systems. The Air Data Computer input for True Air Speed is flagged as invalid whenever TAS falls below 150 knots. 3D/4D usage in terminal area operations will require Time Control and Localizer/Glideslope modes at slow speeds. For this reason, a scheme has been implemented that uses IAS to approximate TAS whenever the latter speed reaches its limit of 150 knots.

INPUT SENSOR DATA

Localizer Frequency + High-Level Flag

Localizer Deviation + High-Level Flag

Glideslope Deviation + High-Level Flag

IAS Airspeed Error + High-Level Flag

HSI Input for Selected Course

Auto-Pilot Engaged Discrete

RNAV-Engaged Discrete

VNAV-Engaged Discrete

TABLE 2-3

Sensor Data Inputs that have been added for the 3D/4D systems

OUTPUT SENSOR DATA

Commanded Localizer Frequency + Localizer Enable discrete

Localizer Deviation

Glideslope Deviation

Localizer Track Angle Error

IAS Command

Localizer Arm Discrete

Localizer Capture/Track Discrete

Glideslope Arm Discrete

Glideslope Capture/Track Discrete

Flight Director Computer Flag

TABLE 2-4

Sensor Data Outputs that have been added for the 3D/4D systems

2.6 Navigation Software Modifications

All of the ANS-70A Navigation software programs were modified for 3D/4D usage. Functional modifications were only implemented for the Lateral Navigation program. Remaining changes were the result of the core reduction effort discussed in Section 2.0. Tasks implemented for the Lateral Navigation software consisted of modifying the ANS-70A criterions for waypoint passage and the Distance-To-Go computation.

The 3D/4D geometry for waypoint passage is illustrated in Figure 2-11. Geometry is shown for the case where the flight plan includes an offset, POFS. Passage of the TO-waypoint occurs whenever both of the following conditions are true:

1) The Lateral-Distance-Along-Track (LDAT) to the offset waypoint defined for the TO-waypoint is less than the calculated Leg-Switch-Distance (SWD).

   When flying legs without offsets, the offset waypoint is defined to be coincidental with the parent waypoint of the TO-event.

2) The bisector defined at this same leg intersection has been crossed.

Referring to figure 2-11, the bisector is given by

$$\mathrm{BISECTOR} = \mathrm{CRSO} - 1/2\,(\mathrm{CRSO}-\mathrm{CRSI}) + \pi/2\ \mathrm{SIGN}\,(\mathrm{CRSO}-\mathrm{CRSI})$$

The bisector is passed once the expressions (BFLEG-BISECTOR) and (CRSI + π - BISECTOR) have opposite signs.

Geometry for the Distance-To-Go computation is shown in Figure 2-11. The ANS-70A system always uses the parent waypoint for the Lateral To-Event as a reference point for this computation. This is considered to be undesirable for Time Control procedures when flying offset legs. The 3D/4D system uses the offset waypoint as a reference point for the Distance-To-Go computation.

The ANS-70A system records the coordinates of the parent waypoint in the Flight Plan Table. This was left unchanged for the 3D/4D systems. Coordinates of the offset waypoint are calculated by the Flight Plan Monitor whenever revisions are made to the flight plan.

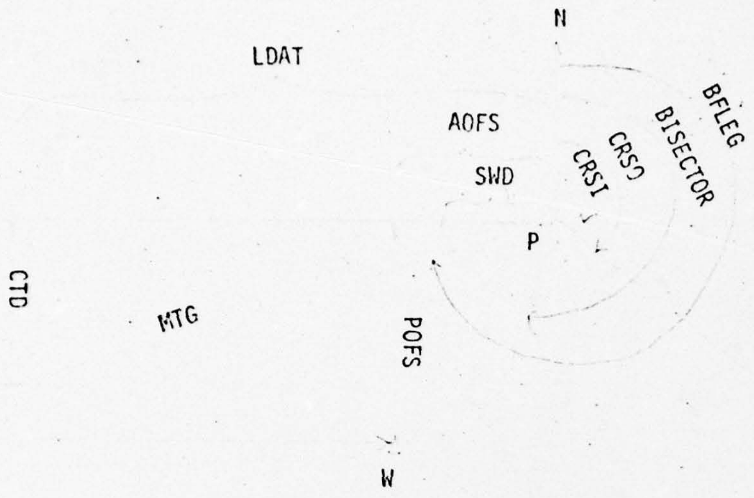


FIGURE 2-11

Geometry for Waypoint Passage

PARAMETER	DESCRIPTION
W	Parent Waypoint of Lateral To-Event
P	Offset Waypoint of Lateral To-Event
SWD	Switching Distance
LDATA	Lateral-Distance-Along-Track to the Offset Waypoint
CTD	Cross-Track Distance
MTG	Miles-To-Go from Aircraft to the Offset Waypoint
POFS	Offset (Base or Parallel) that has been defined for W
AOFS	Along-Path-Offset that must be flown beyond W to reach P
CRSI	Course Into P as determined by Flight Plan
CRSO	Course Out of P as determined by Flight Plan
BISECTOR	Bisector constructed at P
BFLEG	Course (True) from P to the Aircraft

The 2D Plus Time Control System provides the Task II capability of simulating Time Control for non automatic RNAV systems. Major software tasks for this system include the following items:

1) Implementation of CDU software for the revised Progress page. Software implementation for this task is described in Section 3.2.

In addition to the Progress page, CDU software allows pilot access to the Flight Plan and Present Position pages. The Flight Plan page is required for data assemblies. The Present Position page is to be used for pilot updates of aircraft present position.

2) Maintenance of RNAV waypoint parameters in an array structure that is maintained by the CDU software. The Flight Plan Editor is not used for maintenance of the Flight Plan Table. The pilot has the capability of specifying waypoint parameters for ten waypoints at any given time. Ten 3-word arrays are maintained to record entries for waypoint identifier, bearing, and distance. Two additional arrays are maintained as CDU work areas. A description of the array components used to identify the waypoint parameters is given in Section 3.1.

3) Use of HSI analog input for Desired Course. A description of the software implementation for this task is given in Section 3.3.

4) Implementation of a Time Control Algorithm that is unique to the 2D Plus Time Control System. All Time Control computations reference the leg presently being flown. A software description of this algorithm is given in Section 3.4.

An additional task consists of suppressing the ANS-70A computations for leg switching and waypoint passage. This has been achieved by constructing a version of the Lateral Navigation program that is unique to this System. This program was constructed by modifying the Base System version of Lateral Navigation. The only other change to this module consisted of implementing a 20-second Waypoint Alert criterion to replace the 15-second criterion used for the other systems.

.1 External Data Structure

Data parameters unique to the 2D Plus Time Control System have been assigned to a separate module. Data definition for this set of parameters are given below. With the exception of FP.EDIT and NLTOCHG, all parameters are completely maintained by the CDU software.

DATA TYPE	SYMBOLIC NAME	DESCRIPTION
Integer	Display.Wypt	Waypoint Number currently being displayed.
Integer	Use.Wypt	Waypoint Number for waypoint currently being used for navigation.
Integer	FP.Edit	Validity flag that is assigned a VALID status by the CDU software whenever a CDU edit has been made that affects the current navigation leg. FP.EDIT is assigned an INVALID status by the Flight Plan Monitor after the Navigation Event Table (NLTO array) is updated to reflect the edit.
Integer	NLTOCHG	Validity flag that is assigned a VALID status by the Flight Plan Monitor whenever an input (via either the CDU or the HSI Course knob) has been processed that affects the current navigation leg. NLTOCHG is assigned an invalid status by the Speed and Time Control Procedure after the Time and Speed Control computations are updated.

DATA TYPE	SYMBOLIC NAME	DESCRIPTION
Integer	OFFSTF	Validity flag for OFFSET. OFFSTF = valid whenever a lateral offset is commanded, invalid otherwise.
Real	OFFSET	Lateral Offset currently being flown.
Real Array	ARR0, ARR1 ARR9, ARRA, ARRB	Waypoint arrays used to record Ident, Bearing, and Distance information for the IRNAV waypoints.
Pointer Array	PTR.ARRAY	Array of 12 pointers used to reference the waypoint arrays.
Integer Component	WYPT.IDENT	Component used to record ASCII identifier Waypoint Number. Assigned a component position of 0.
Real Component	WYPT.BRG	Component used to record True Bearing from navaid to waypoint. Assigned a component position of 1.
Real Component	WYPT.DIST	Component used to record distance from navaid to waypoint. Assigned a component position of 2.

3.2 Functional Description of Revised Progress Page

Display formats for the Progress page are illustrated in Figures 3-1 and 3-2. A functional description of data fields for the Progress page is given below.

I) [STATION]

This field provides the pilot with a means of specifying the navaid that is to be used for navigation and tuning purposes. Allowable entries consist of three-character station identifiers that are stored in the data base. The scratchpad message "NOT STORED" is annunciated for attempts to enter invalid identifiers.

II) [MODE]

This field provides a CDU display of the current navigation mode. Current usage of VOR, DME, and Air Data/Heading information for navigation purposes is indicated by displaying V, D, and A respectively. The absence of a given mode is indicated by displaying a dash (-) for the corresponding letter. A downgrade of mode will result in the annunciation of the data field. The annunciation may be cancelled via a line-select entry of a single minus sign (-).

III) [OFST]

Parallel offsets are commanded by means of the OFST field. The initial character of an offset entry must be either an "L" or "R." The magnitude portion of the entry is allowed to have a range of 0.0 to 50.0 (NM). Thus, an entry of L5.1 would be used to indicate a left offset of 5.1 NM. An existing offset is cancelled via a LO (L followed by a zero) or RO entry.

IV) [WIND]

The data field for Wind serves as a display field only. Bearing information is displayed as magnetic bearing from the station. Dashes are displayed whenever Air Data or Heading Information is invalid.

V) [WPT/BRG/DIST]

The third pair of CDU lines has been reserved for waypoint information. Waypoint parameters consist of Identifier, Bearing (Magnetic) from navaid, and distance from navaid. Allowable entries consist of editing all three fields with one entry or revising any field with a single edit.

A total of 10 waypoints may be designated at any given time. Digits 0 through 9 are used to designate waypoint identifiers. Display of a waypoint other than the USE-waypoint is indicated by blinking the identifier field. A waypoint may be deleted via an entry of a single minus sign for Line-Pair No. 3. Waypoint deletion will result in a display of dashes for the BRG and DIST data fields. Navigation computations reference the navaid presently being used whenever the USE-waypoint is deleted. Deletion of a waypoint will result in the software assignment of 0.0 to the Bearing and Distance fields associated with the given waypoint.

A description of allowable edits for the WPT/BRG/DIST fields is given below:

Entry for Waypoint Identifier:

Acceptable entries consist of a single scratch pad digit (0 through 9)

Entry for Bearing:

Entries must be preceded by a single slash (/).

Range: 0.0 to 360.0 degrees
Increment: 0.1 degrees
Examples: /5.1
/05.1
/005.1

Entry for Distance:

Entries must be preceded by two slashes (//).

Range: 0.0 to 200.0 NM

Increment: 0.1 NM
Examples: //2.9
//02.9
// 002.9

VI) TTW

The Time-to-Waypoint field serves as a display field only. The value displayed represents the current estimate of time required to reach the waypoint specified by the Use-waypoint, HSI input for desired course, and current parallel offset.

Range: .0 to 99.9 minutes (values exceeding 99.9 minutes will be displayed as 99.9 minutes).

Resolution: 0.1 minutes

VII) DIST

This field is used to display Distance to the present To-waypoint. Edits are not allowed.

Range: .0 to 999.9 NM

Resolution: 0.1 NM

VIII) GS

This field is used to display the Present estimate for Ground Speed.

Range: 0 to 999 knots

Resolution: 1 knot

IX) [DTA]

Desired Time of Arrival (GMT units) is to be entered for this field. DTA may be revised or deleted at any time. DTA deletion is accomplished by an entry of a single minus sign. Software is required to invalidate an existing DTA entry upon selection of a new USE waypoint, or revision to any of the WYPT, BRG, or DIST fields of the USE-Waypoint.

Range: .0 to 2359.9 where modulo arithmetic of 60.0 minutes is applied to the last three digits.

Resolution: 0.1 minute

X) GMT

A GMT entry must be preceded by a slash mark (/). Rules for range, resolution, and display format are the same as those for the DTA field.

XI) IAS CMD

This field is used to display Commanded IAS as calculated by the Time and Speed Control Algorithm. Line-Select entries are not allowed.

Line Pair 6 is used for the Early/Late Display. Display formats for the label and data fields are identical to those used for the Base System.

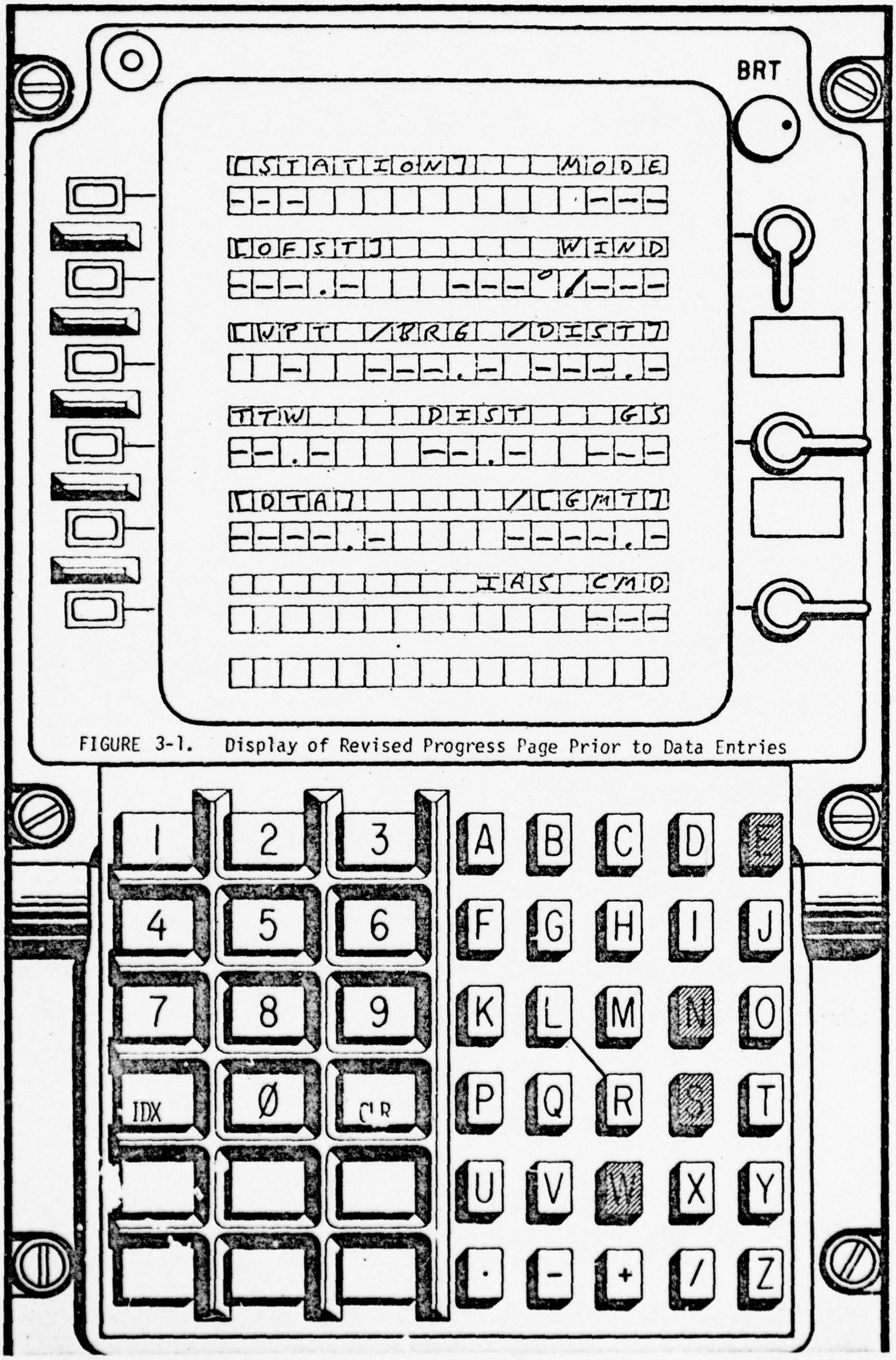
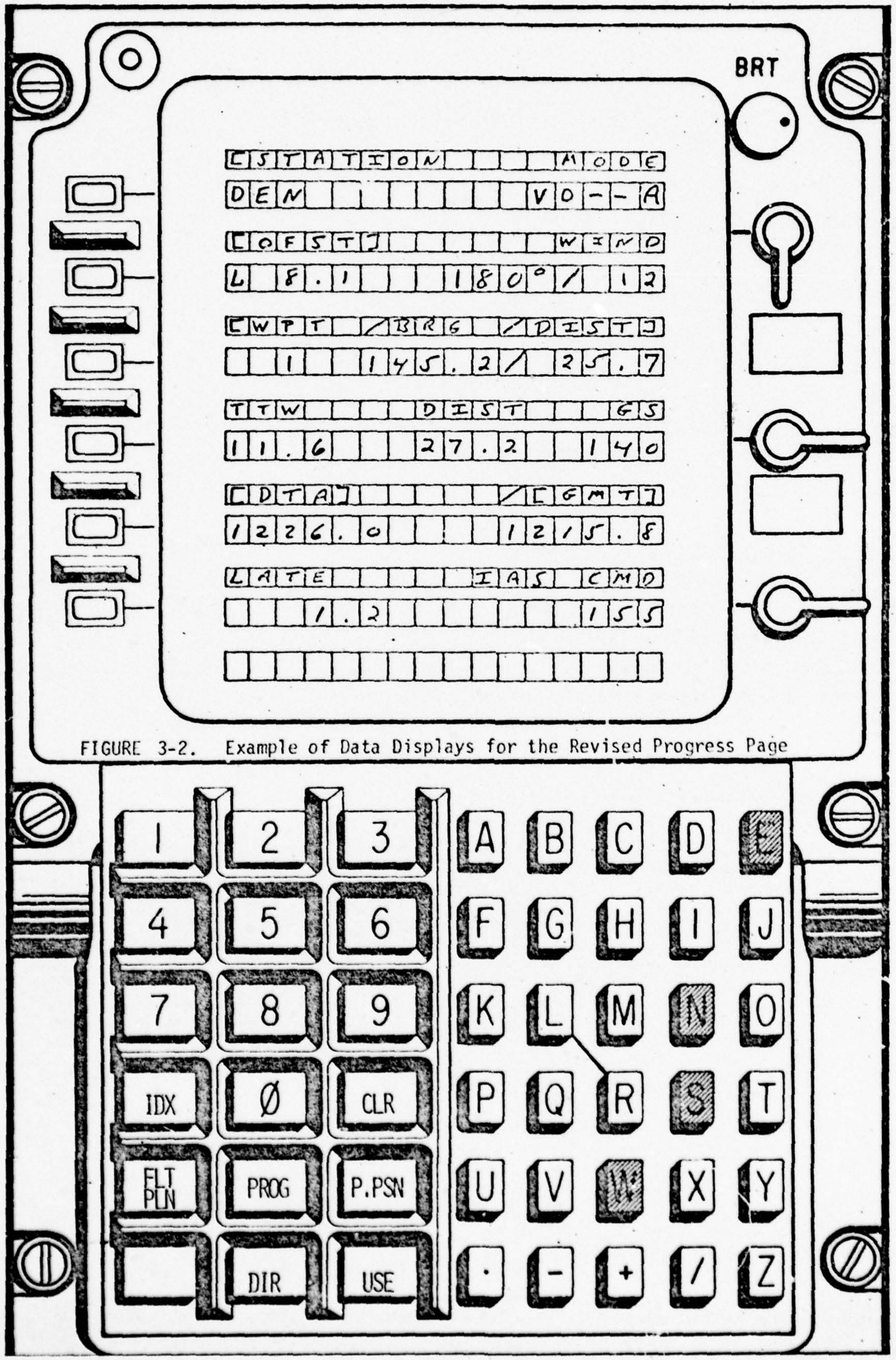


FIGURE 3-1. Display of Revised Progress Page Prior to Data Entries



```
[STATION          MODE
DEN              VO--A
[OFST]            WIND
L 8.1      180°/    12
[WPT  /BRG   /DIST]
   1   145.2/  25.7
TTW      DIST       GS
11.6     27.2      140
[DTA]         /[GMT]
1226.0         1215.8
LATE       IAS   CMD
    1.2            155
```

FIGURE 3-2. Example of Data Displays for the Revised Progress Page

3.3 HSI Input for Selected Course

Figure 3-3 illustrates software interfaces for programs that require access to the HSI Input for Selected Course. The final objective is to pass the Selected Course input to the Lateral Navigation program as a course command for the present leg.

Initial processing of the input is performed by the ASC Input Program. Converted data is recorded in the ADHD data structure. The Flight Plan Monitor uses the Course input to determine the coordinates of the TO-waypoint. The Selected Course Input is then passed to the Navigation Channel as the flight plan parameter for Course-Into-the-Waypoint (CRSI Component).

Geometry for computation of the TO-Waypoint is given in Figure 3-4.

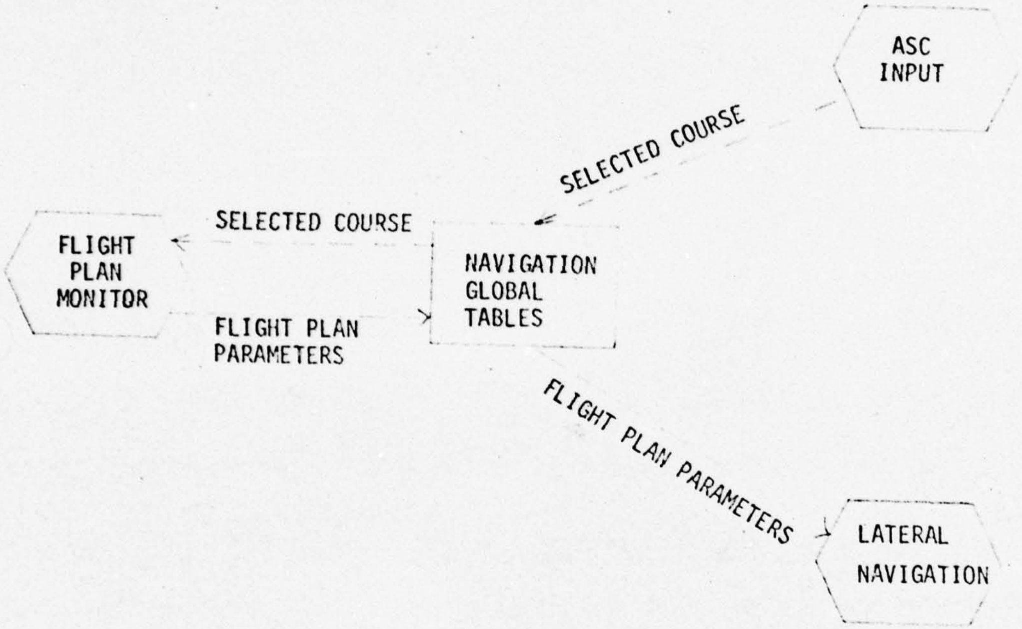


FIGURE 3-3

Software Interfaces used to implement HSI Input for Selected Course

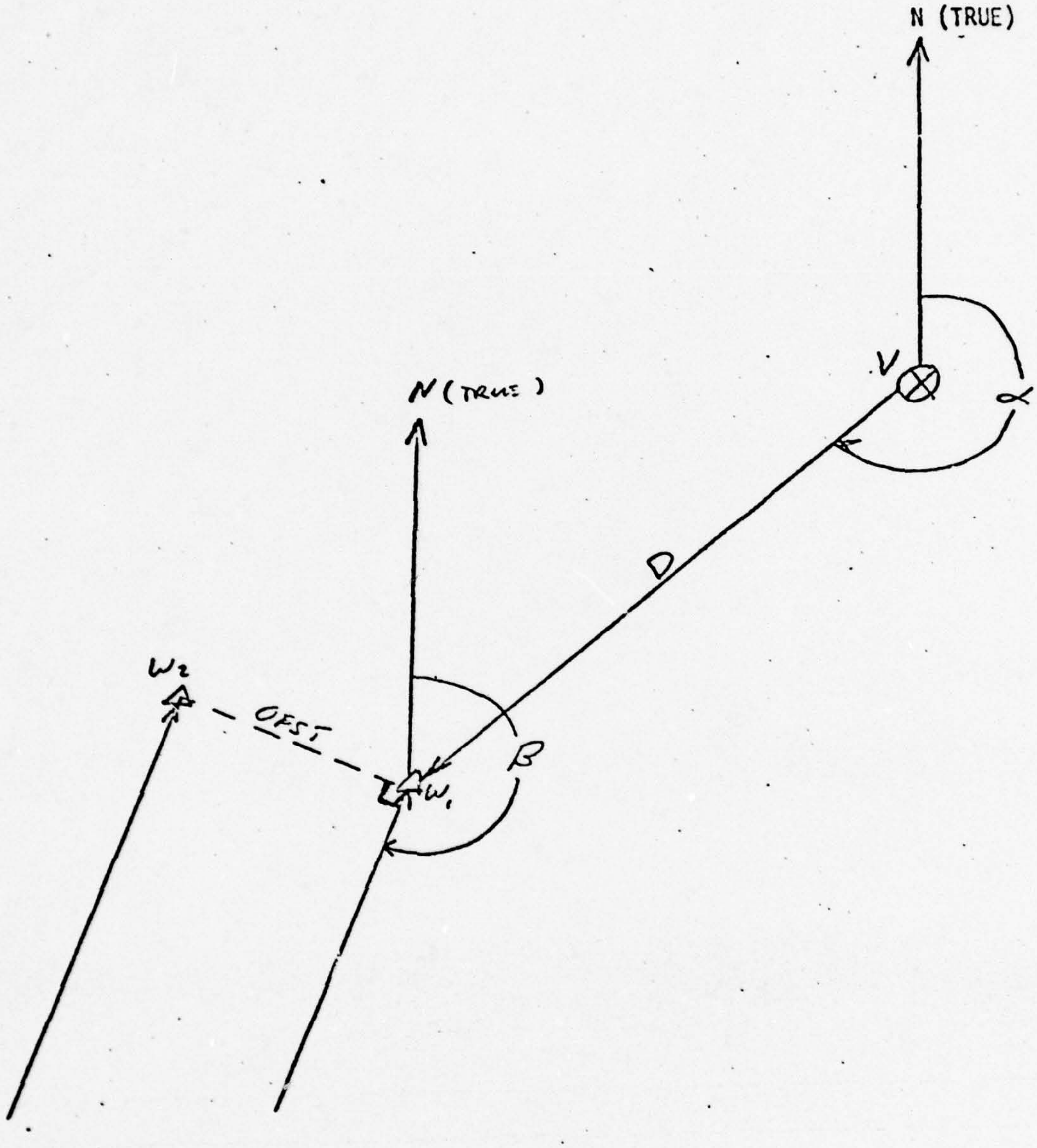


FIGURE 3-4

Geometry for Computation of Waypoint Referenced for Navigation Purposes

V: Navaid specified on Progress page
α: Bearing specified as Waypoint Parameter on Progress page
D: Distance specified as Waypoint Parameter on Progress page
W_1: Coordinates of Use Waypoint
B: HSI Input for Desired Course
OFST: Parallel Offset as specified on Progress page
W_2: Coordinates of Waypoint referenced for Navigation

3.4 Time Control Algorithm

The 2D Plus Time Control System computations for Time Control are only applicable to the leg presently being flown. Maximum use is made of existing data declarations for the Base System. Software interfaces are similar to those defined for the Base System, i.e., control is given to a procedure called TIMCNTRL every cycle by the CDU Main Control Program. Procedure TIMCNTRL will monitor the system clock and status of flight plan revisions to determine the necessity of executing the Time Control Algorithm for the given cycle. A task description for TIMCNTRL is given below.

A program listing of the TIMCNTRL routine is given in Appendix D.

Task Description for the 2D Plus Time Control System
Time Control Algorithm

Task Specification for 2D
System TIMCNTRL Procedure

Procedure Type

Procedure TIMCNTRL ( )

Module

2D Plus Time Control System Version of Time and Speed Control

Task Summary

Time Control computations for the system are only valid for the present leg. Computations are performed whenever either 1) ten seconds have elapsed since the last execution of the Time Control Algorithm, or 2) a Flight Plan Revision has occurred since the last cycle. The latter condition is determined by monitoring NLTOCHG for a value of VALID.

Computations required for the display of Time Control parameters are recorded in VCOMI, VCOMIF, EARLY, TIMERR, and TIMERF. Declarations for this set of parameters are defined by the Base System insert file DATAZ6I1.

Internal Data Structure

Synonym Declarations:

CYCLE.TIME	Minimum time allowed between Time-Control computations. CYCLE.TIME is to be assigned a value of 10 seconds, i.e.,

$$\text{CYCLE.TIME} = \frac{(10\ \text{sec})\ (1000000\ \text{us/sec})}{(409.6\ \text{us/count})}$$

Integer Declarations:

SYMBOLIC NAME	DESCRIPTION
TALCLOCK	Own record of last time that the Time Control Computations were executed.
GMTI	Own copy of GMTS expressed in terms of GMT units (12 hours).

Real Declarations:

SYMBOLIC NAME	DESCRIPTION
ACTAS	Own snapshot of Aircraft True Air Speed.
ACALT	Own snapshot of Aircraft Baro Altitude.
ACIASERR	Own snapshot of IAS error as recorded by the IAS Instrument.
ACNWND	Own snapshot of the North Component for Wind.
ACEWND	Own snapshot of the East Component for Wind.
DESIRED.CRS	Own snapshot of the HSI course input.
WATK	Along-track component of wind for the vector whose origin and direction are given as present aircraft coordinates and DESIRED.CRS, respectively.
KALT	Altitude Proportionality factor required for the conversion from TAS to IAS.
KALTSCALE	Scale Factor required to prevent overflow for the TAS to IAS computation.
GMTR	Own copy of GMTS expressed in terms of Navigation Time Units.
GMTCONV	Conversion factor used to convert GMT to Navigation Time Units. GMTCONV = 10.8039624/12.
TTGCOM	Commanded Time-to-Go to arrive at the To-Waypoint.

Data Declarations External to Procedure TIMCNTRL;

Synonym Declarations assigned to VCOMIF:

SYMBOLIC NAME	ASSIGNED VALUE	INSERT FILE	DESCRIPTION
IASVALID	15	DATAZ6I1	A valid commanded IAS has been calculated by TIMCNTRL.
IVGMT	14	DATAZ6I1	Either 1) GMT has not been entered via the CDU, or 2) a power interrupt of duration exceeding two seconds has invalidated GMT.
IVTAS	13	DATAZ6I1	The Air Data Sensor input for TAS is invalid.
IVALT	11	DATAZ6I1	The Altimeter sensor input for Baro Altitude is invalid.

Integer Declarations:

SYMBOLIC NAME	INSERT FILE	DESCRIPTION
TCOM	DATAZ6I1	Commanded Arrival Time for the To-Waypoint.
TCOMF	DATAZ6I1	Validity flag (full word) for TCOM.
VCOMIF	DATAZ6I1	Validity flag (full word) for VCOMI.
TIMERF	DATAZ6I1	Validity flag (full word) for TIMEFRAand EARLY.
GMTS		Present value of GMT in seconds.
DMTGF	NTABN6I1	Validity flag (Byte) for DMTG.
DTTGF	NTABN6I1	Validity flag (Byte) for DTTG.
TASF	NTABN6I1	Validity flag (Byte) for TAS. TASF is declared as a component for pointer ADHD.
BALF	NTABN6I1	Validity flag (Byte) for Baro Altitude. BALF is declared as a component for pointer ADHD.

Real Declarations:

SYMBOLIC NAME	INSERT FILE	DESCRIPTION
VCOMI	DATAZ6I1	Commanded IAS displayed on the CDU and IAS instrument. VCOMI is a fraction of 1000 kts.
VCOMT	DATAZ6I1	Commanded TAS calculated by TIMCNTRL.
EARLY	DATAZ6I1	Early/Late indicator calculated by TIMCNTRL and used to control the EARLY and Late displays on the CDU. A positive value is used to indicate a late status.
TIMERR	DATAZ6I1	Expected arrival time error for the To-Waypoint. TIMERR is expressed in lateral navigation time units.
MINR	DATAZ6I1	Minimum magnitude that REAL numbers may be assigned on the U-1108.
MAX64		Maximum magnitude that Real numbers may be assigned on the U-1108.
DMTG	NTABN6I1	Distance between the aircraft and the To-Waypoint.
DTTG	NTABN6I1	Estimated time required to arrive at the To-waypoint.
TAS	NTABN6I1	Sensor input for True Air Speed. TAS is declared as a component for the ADHD structure.
BALT	NTABN6I1	Sensor input for Baro Altitude. BALT is declared as a component for the ADHD structure.
DELV	NTABN6I1	IAS Instrument Error. DELV is declared as a component for the ADHD structure.
HSICRS	NTABN6I1	Desired course as input via the HSI. HSICRS is declared as a component for the ADHD structure.
NWND	NTABN6I1	North Component of Wind.
EWND	NTABN6I1	East Component of Wind.

Task Description

```
IF    Ten seconds have elapsed since the last execution of the Time Control
      Algorithm

OR    A flight plan revision has occurred since the last cycle (i.e., NLTOCHG
      EQL VALID)

THEN BEGIN    ... Computations Required

      Update own clock reading.

      Assign an INVALID status to TIMERF, the validity flag for arrival time error.

      Record the status of VCOMI i.e.,

           VCOMIF = IF invalid GMT THEN IVGMT
                    ELSE IF invalid aircraft true air speed THEN IVTAS
                         ELSE IF invalid aircraft altitude THEN IVALT
                              ELSE IASVALID

      IF VCOMIF has a status of IASVALID

      THEN BEGIN ... VCOMI VALID

           After yielding channel time, record a snapshot of sensor data needed
           for own computations, i.e.,

                 ACTAS =          Aircraft True Air Speed,
                 ACALT =          Aircraft Baro Altitude,
                 ACIASERR =       IAS Instrument Error,
                 ACNWND =         North Component of Wind,
                 ACEWND =         East Component of Wind,
                 DESIRED.CRS =    HSI Course Input.

           Calculate the along-track component of wind for the course presently
           input via the HSI, i.e.,

                WATK = -[ACNWND*NCOS(DESIRED.CRS)
                       +ACEWND*NSIN(DESIRED.CRS)].

           Calculate KALT, the Altitude Proportionality constant, i.e.,

                KALT = RLMTF[ACTAS-VCOMI-ACIASERR)*KALTSCALE,ACALT*ACTAS-MINR]
                       ...Limit function used to protect against overflow in
                       KALT computation.

           KALT = KALT/(ACALT*ACTAS).

           Limit KALT to the interval (.10, .15), i.e.,

                KALT = RMAXF[RMINF(KALT,.15),.1].

           IF a commanded arrival time has been entered via the CDU,
```

```
     THEN BEGIN   ... Valid Arrival Time

          Calculate VCOMT, the commanded True Air Speed, i.e.,

               VCOMT = RLMTF(DMTG,DTTG-MINR)/DTTG-WATK.

          Convert Commanded TAS to Commanded IAS, i.e.,

               VCOMI = [MAX64-(KALT*ACALT)/KALTSCALE]*VCOMT.

          Limit VCOMI to upper limit of IAS instrument (300 kts for G-I,
                410 kts for C-880).

          Determine commanded Time-To-Go, i.e.,

               GMTI=TCOM-GMTS  ...Integer arithmetic required as GMTS is
                               declared an Integer.

               TTGCOM=GMTR/GMTCONV ...Convert to Lateral Nav Time Units.

               EARLY = TTGCOM-DTTG.  ...Sign of EARLY provides EARLY/LATE
                                     indication for CDU Display.

               TIMERR = ABS (EARLY)  ...Record magnitude of arrival time
                                     error for CDU display.

          END  ...Valid Arrival Time.

     END  ...VCOMI VALID.

Assign an INVALID status to NLTOCHG to indicate computations are not required
next cycle.

END ...Computations Required.
```

Section 4
RNAV/ILS INTERFACE DESIGN

4.1 Introduction

The ILS program is managed by a main or executive routine. This routine is an interface between the ILS program and the navigation program. This section describes operations which transition the boundary between the ILS executive program and the 3D/4D main program. A description of the executive program is contained in Appendix E.

4.2 Operation

4.2.1 Mode Selection

The overall view of the ILS mode is achieved by selecting the ILS page. This is done by pushing the ILS line select key with the INDEX page on the CDU. After pushing this key, the ILS page will appear on the CDU. Refer to Figure 1 for a diagram of the ILS CDU page. If the localizer programs are not in core, the ILS page will indicate dashed lines under the STATUS label. At any time the parameters ILS FREQ, TD WPT (touchdown waypoint), RWY CRS (runway course), LENGTH (runway length) can be entered for data storage to be used for the ILS computations. Allowable ranges for these parameters are as follows:

a. Runway courses	0-360 degrees	error message on scratchpad if not within these bounds -- will not accept further commands until error message removed.
b. Runway length	5-16 thousand feet	error message if not within these bounds -- defaults to 9,000 feet
c. ILS frequency	a valid ILD frequency	error message if non-ILS frequency -- cannot accept further commands
d. Touchdown waypoint	A waypoint already entered in the flight plan	error message if cannot be found -- cannot accept other commands until message has been cleared.

The ILS program tape will be loaded when the ILS page is "up" and when mode is selected for the first time. The three modes available are LOC, APPR AUTO, and RNAV. LOC is an ILS mode using only localizer guidance. APPR AUTO is an ILS mode using localizer and glideslope guidance. The operation of the ILS modes will be discussed in a later paragraph.

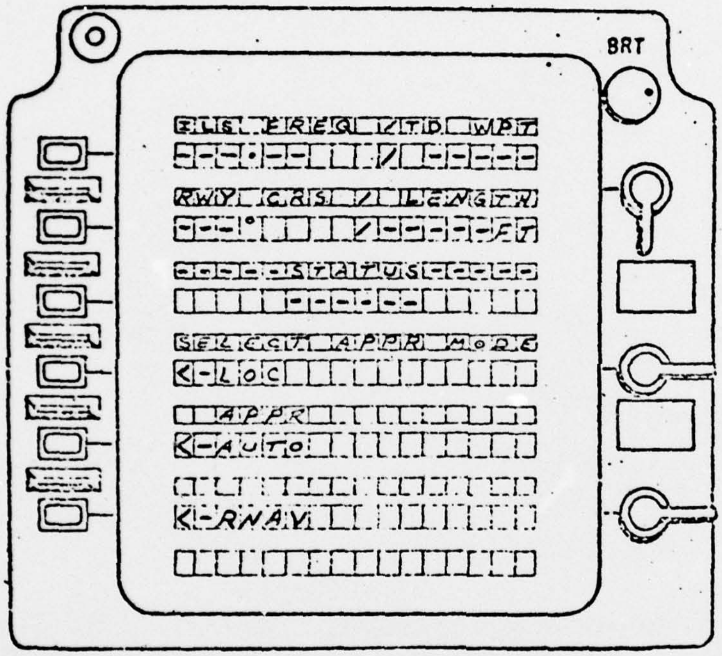


Figure 4-1. ILS Data Page Prior to Data Entry

When the program is loaded, a message will indicate that a "DATA SEARCH" is in effect. The ILS computations will not be accessed until the program is completely loaded. The mode selected will be checked for validities and if the mode is not generated, an INVALID message will be displayed near the mode selector (on the same line). If at any time the mode becomes valid, INVALID will be extinguished. If the mode is again selected, LOC or APPR AUTO will be displayed under STATUS and it will become engaged. If during the engagement of LOC a capture has not been effected and APPR AUTO is selected, and if the validities are checked and approved for APPR AUTO, then APPR AUTO will replace LOC under STATUS. If the validities do not check, INVALID will be displayed by APPR AUTO mode. Once the INVALID extinguishes, if it does, the mode must be reselected before it will take. Before any attempt to use either mode, the INVALID will not appear. In other words, the INVALID can occur only after a request has been checked and refused. Notice that the refusal of APPR AUTO does not discontinue the present use of LOC. The program tape will be loaded if R-NAV is selected. The program tape will also be loaded with the selection of the LOC or APPR AUTO modes. When R-NAV is used to load the program tape, R-NAV remains as the STATUS or selected mode.

To discontinue the use of ILS modes, R-NAV is selected. If AP is disengaged, R-NAV will appear under STATUS and R-NAV will be engaged and a message will appear in the scratchpad: PUSH TO RELOAD R-NAV DATA. Upon the second selection of R-NAV, the R-NAV data base will be overlaid, removing the ILS program. DATA SEARCH will be displayed on the scratchpad while the transition is occurring. ILS will not be used or serviced in any manner during the transition. If localizer is in progress and the autopilot is engaged, R-NAV will not become engaged with the selections of R-NAV. A message will appear on the scratchpad indicating that "DISENGAGE A/P BEFORE SELECTING R-NAV." The re-loading of the R-NAV data will also be prevented until A/P disengage.

4.2.2 Annunciators

An ILS ANNUNCIATOR panel will be mounted on the glareshield. The annunciator panel will display messages as shown in Figure 2. LAT NAV will be annunciated when the autopilot is engaged (APENG) or the flight director is engaged (FDENG), the R-NAV system is engaged and the system is not ignoring conventional R-NAV steering outputs (i.e., not during localizer capture/tracking). LOC ARM is annunciated during localizer arm. LOC is annunciated during localizer capture.

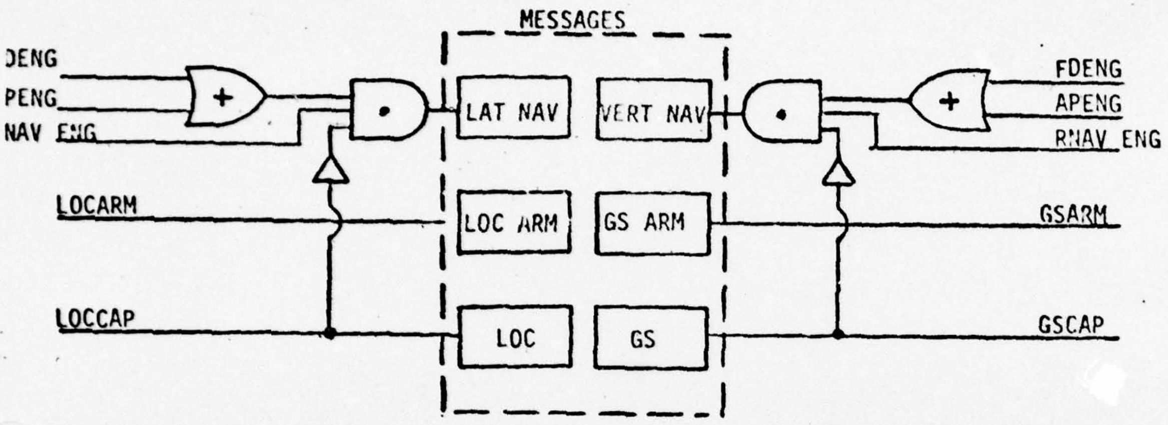


Figure 4-2. RNAV/ILS Annunciator Panel Messages

Similar logic holds for V-NAV. VNAV is annunciated when the autopilot is engaged, (APENG) or the flight director is engaged (FDENG), the VNAV system is engaged and the system is not in glideslope capture/tracking. G/S ARM is annunciated during glideslope arm. G/S is annunciated during glideslope capture.

4.2.3 Mode Description

LOC (LOCALIZER) MODE

Selection of the LOC mode will automatically inhibit the glideslope computations. Only a localizer capture and/or track will then result. If the capture conditions or logics have not been satisfied and the processor is in a valid R-NAV submode it will continue to fly the submode until the capture conditions are satisfied. If capture conditions are satisfied and exceeded (track conditions exist) the computations will automatically compute track computations. At the time of LOC mode selection, all validities or parameters monitored must be valid to engage the mode. Once engaged, loss of any validity will cause the computer flag output to drive the computer flag in the ADI.

Signals monitored: LOC SUPER FLAG, LOC FREQ, COMPASS.

Annunciator sequencing: LAT NAV, LAT NAV and LOC ARM, LOC

APPR AUTO (APPROACH AUTOMATIC) MODE

APPR AUTO mode is much like LOC mode except automatic glideslope captures can be effected. The computer will sense and switch both localizer and glideslope computations according to the level of the logics. That is, if localizer and glideslope capture conditions are not satisfied, the submodes will continue to be used. Upon proper detection of the necessary conditions to effect captures, the captures will occur automatically. If the capture conditions have been satisfied by either or both channels, either or both will start computing the track computations. Glideslope capture is inhibited from occurring before LOC capture.

Signals monitored: ALL OF THOSE IN 4.2.3 and G/S SUPER FLAG, ALTITUDE.

Annunciator sequencing: LAT NAV AND VERT NAV, LAT NAV AND VERT NAV and G/S ARM and LOC ARM, LOC and G/S.

4.2.4 Display Control

The ILS maneuver will be performed using an ADI and HSI much as is done with existing conventional systems. Steering commands for the flight director will be supplied to the flight director computer for display on the ADI. Deviation indicators will be driven by the 3D/4D ILS computations during the ILS approach.

The HSI will be controlled whenever a valid LOC ARM is annunciated or latched. Control over the course pointer, deviation indicators and distance readout will be effected by the 3D/4D main program at that time.

4.3 ILS EXECUTIVE PROGRAM OPERATION

The ILS executive program interfaces the ILS approach computations with the overall 3D/4D main computational program. The executive program converts data passed to it from the main 3D/4D program for use in the glideslope and localizer computational routines. In the ILS executive program decisions are also made on validities and mode selection logic. Basically the ILS executive program provides overall administration to the computational routines. The operation as described in the previous sections of this paper are partly accomplished by the ILS executive program. The other part is accomplished by the 3D/4D main program. Figure 4-3 is the flow diagram for the ILS executive program. AED coding of ILS executive program is included in Appendix E. From Figure 4-3, when the ILS page has been selected and the ILS program has been loaded from tape, the ILS executive program is serviced by transferring data to the ILS executive program. This is followed by a preset (initial conditions operation).

From data passed from the 3D/4D program and ILS page, the distance to touch-

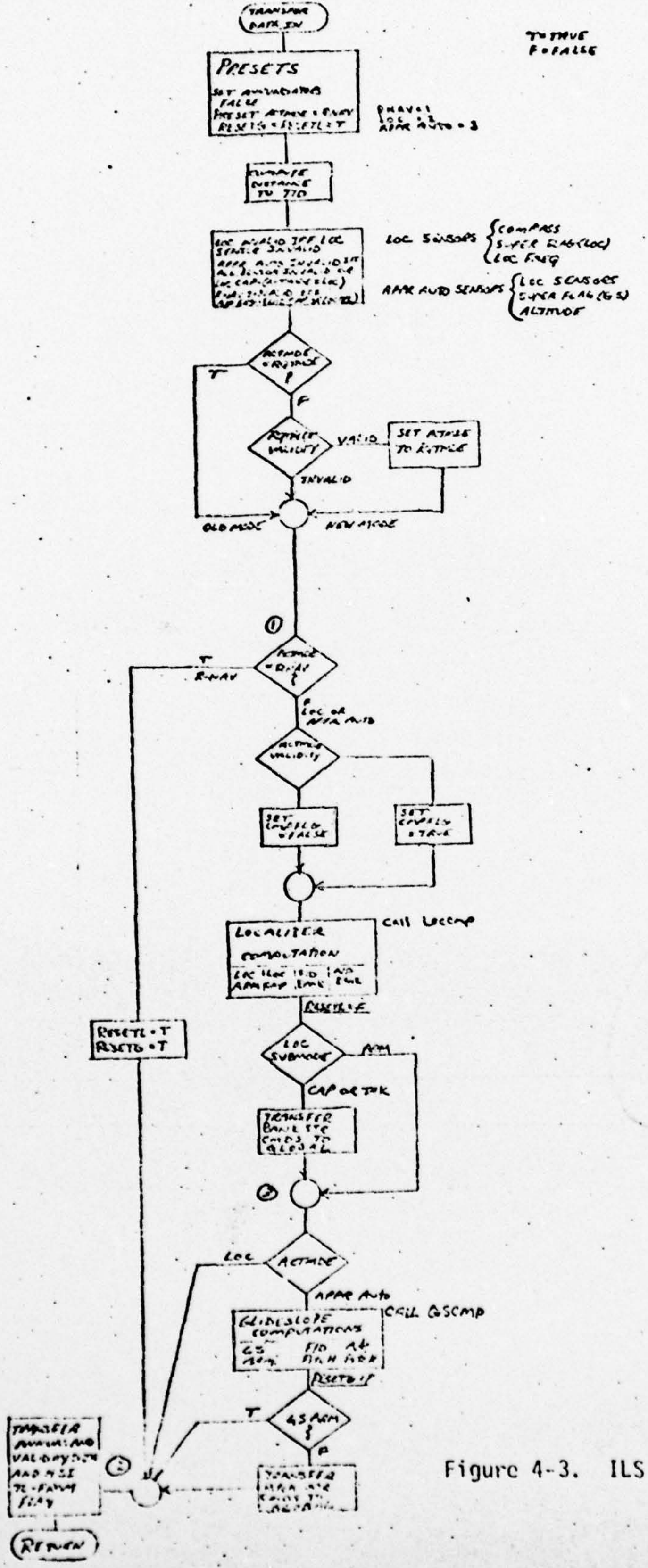


Figure 4-3. ILS Executive Program Flow

down is now computed. Next in sequence, the mode validities are update to establish the validity of each mode serviced by the ILS executive program.

At this point, the present requested mode is compared to the mode now considered active by the ILS executive program. If the requested and active mode are the same, program flow is routed to point 1.

If the requested mode is not the same as the active mode, the validity of the requested mode is checked. If the requested mode validity is true (mode valid) then the active mode is set to the requested mode. If the requested mode is false (mode checked invalid), the previous active mode is retained (requested mode denied).

Program flow now proceeds to point 1. If the active mode is R-NAV, program flow is routed directly to point 2. R-NAV pitch and roll steering data is transferred back to the 3D/4D main program unaltered. Notice also before control is transferred back to the 3D/4D main program, annunciator and mode validity updates are transferred back to the 3D/4D main program. This is true every computational cycle.

The parameter RESETL and RESETG was set true when program flow was routed control from point 1 directly to point 2. These parameters are always maintained true until after the first time either the localizer computation or glideslope computation is executed. The resets will be further treated below.

At point 1, should the active mode be either localizer (LOC) or approach auto (APPR AUTO), then the next action is to check the validity of the active mode. The computer flag is next set depending upon whether the active mode validity is true or false. If the validity is true (mode determined valid), the computer flag is set true (not displayed). If the validity is false, the computer flag is set false (the computer flag on the ADI is displayed).

At this point the localizer computations (localizer control laws for determining roll steering commands) are serviced. When control returns to the ILS executive routine RESETL is set false. RESETL has been used to establish that the localizer computations have been serviced for the first time. This is used as a key for setting initial conditions. Notice that RESETL will now be false unless R-NAV mode is re-established.

From information passed to the executive program from the localizer computation, the ILS main program passes bank steering commands to the 3D/4D main program if the localizer mode has been determined to be in a capture or track submode. If in the arm submode, bank steering commands are not passed to the 3D/4D main program. Control is essentially achieved therefore by the ILS main program overwriting the R-NAV steering commands. Notice that no overwriting will occur if the arm submode is the present status. R-NAV continues to be flown during the arm submode.

After passing through the localizer routines, a decision is made based upon whether the active mode is localizer or approach auto. If localizer mode is the active mode program flow proceeds directly to point 2. Otherwise the glideslope computations are serviced. The operation of the ILS executive program in regard to the glideslope computations is almost identical to that described above for the localizer computation. RESETG is used in an identical manner to the use described for RESETL above.

Once point 2 has been reached independent of the route, annunciator and validity data is transferred to the 3D/4D main program. The HSI to-from flag information is also passed at this point. The ILS executive program is now exited and program flow moves to the 3D/4D main program.

The flow graph of Figure 4-3 describes one computational cycle of the ILS executive program. The ILS computation is processed five times a second and its throughput time is in the order of milliseconds.

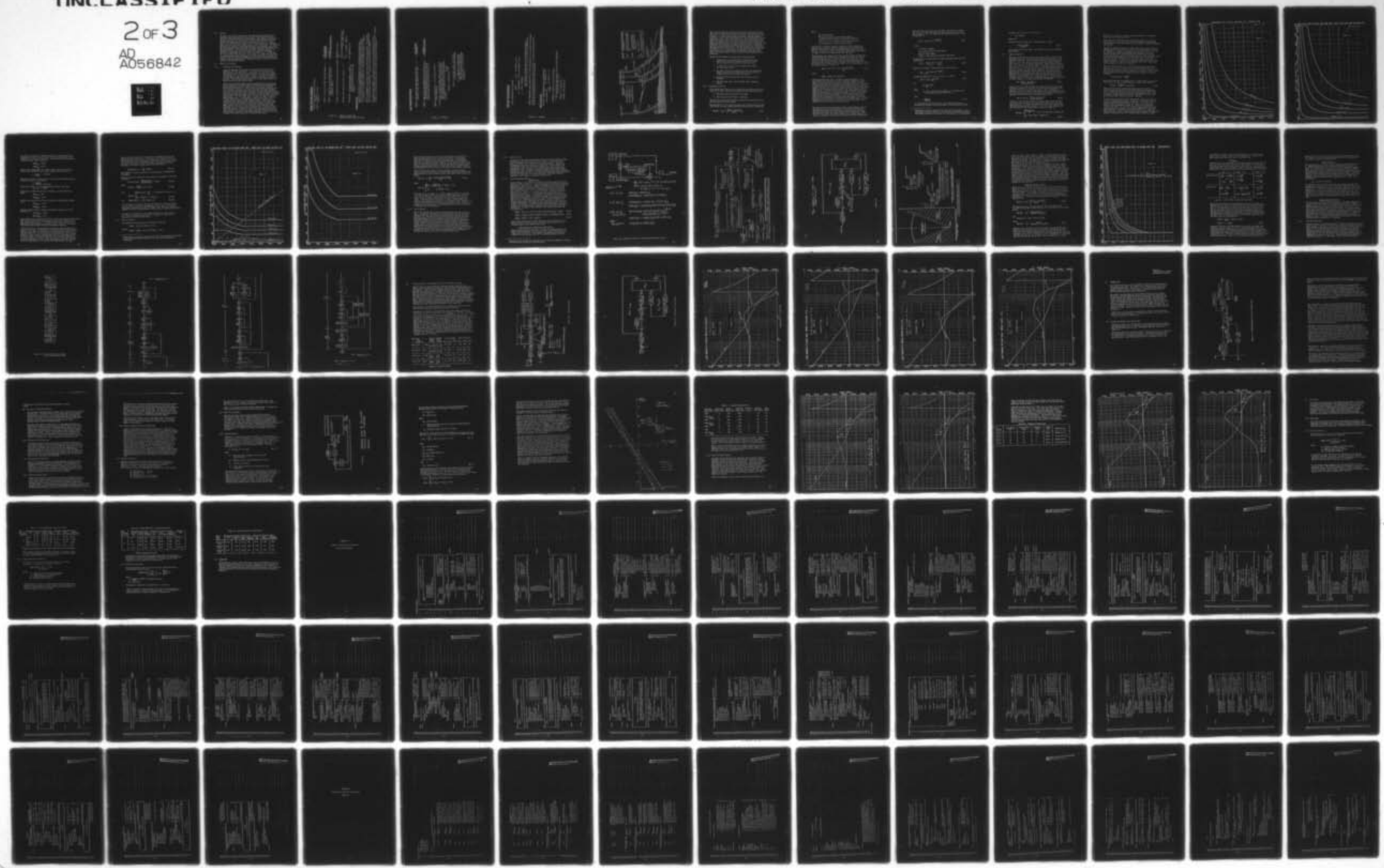
Section 5
LATERAL ILS CONTROL LAW DESCRIPTION AND ANALYSIS

5.1 Introduction

There has been interest for some time in using area navigation system derived data to aid in localizer captures. Reference 1 describes one preliminary Collins' study that investigated this technique. Localizer capture algorithms in general suffer from two important disadvantages. Range information is not generally available, hence the capture law is usually optimized for a given range and performance degradation occurs on either side of the optimum range. Reference 2 describes an attempt to overcome some of this deficiency by estimating range from beam and air data. This approach showed promise but if an area navigation system were available, range information would be inherently present. The second difficulty in localizer captures is that the localizer beam provides angular deviation from a reference centerline. Hence as one moves closer to the localizer antenna, the "beamwidth" of the localizer beam in distance units (e.g. ft.) grows smaller. Thus the available space to execute a turn onto final course once the beam is intercepted shrinks rapidly. As an example, the turn radius associated with 30° of bank at an airspeed of 200 ft/sec implies that for a 90 degree beam intercept, centerline overshoot will occur at ranges shorter than 7.6 nm assuming capture starts (with instantaneous 30° of bank) at 200 μA deviation. At higher airspeeds, the minimum range without overshoot will increase. An area navigation system offers potential alleviation of this situation by permitting the initiation of captures prior to beam intercept, thus avoiding the geometry limits described above. The R-NAV aided approach also has problems, however. The inherent errors in the R-NAV estimates of position and velocity are such that if care is not exercised, a capture initiated on R-NAV data may turn short missing the beam entirely or such a capture may turn late resulting in gross overshoots. Reference 1 described two approaches to overcoming thse problems. The first approach involved never turning to a lesser course cut than 45° before intercepting the beam. This worked well in terms of localizer overshoot but resulted in two distinct "up-down" bank commands or a "double bank" characteristic which was considered by some to be objectionable. The second approach studied in Reference 1 was to always initiate the R-NAV aided capture with 30° of bank command, but to start fading the bank command almost immediately so that hopefully a smooth blend with beam based bank commands at localizer intercept resulted. This approach had a more desirable bank characteristic but appeared to pose formidable difficulties in properly programming the bank command fade in general. The approach to be discussed in this section avoids the double bank characteristic, uses an adaptable initial bank command (i.e. not always 30 degrees), and provides a solution to the bank command fade problem in the case where the system is tending to turn short of the beam.

AD-A056 842   ROCKWELL INTERNATIONAL CEDAR RAPIDS IA COLLINS AVION--ETC F/G 1/5
3D/4D AREA NAVIGATION SYSTEM DESIGN, DEVELOPMENT AND IMPLEMENTA--ETC(U)
JUN 77 J M BRUCKNER, F B BENSON   DOT-FA72WA-3123
UNCLASSIFIED   FAA-RD-77-79-VOL-2   NL

2 OF 3
AD
A056842



5.2 Summary

An R-NAV aided capture algorithm has been developed which provides good localizer capture performance in the presence of nominal R-NAV errors. The system uses R-NAV aiding only when needed. When geometry constraints permit, the system uses primarily beam data and bank information relying on the R-NAV system only for range as a primary piece of data. Track angle error from the R-NAV system is used in this mode but in a way that makes the system relatively insensitive to errors in this data. The technique developed here assumes an integrated R-NAV/ILS flight computer, but could be readily adapted to separate R-NAV and flight computers with minimal interface requirements. This report does not detail simulation results, but design goals of maximum localizer overshoot less than 30 μA and bank activity in established track less than 1° rms with 2.5 μA, 1σ beam noise (τ = 4 sec.) and R-NAV errors of $\pm.2$ nm in CTD, $\pm2°$ in TAE, and ±40 ft/sec in ground speed have been demonstrated in hybrid simulation at ranges from 5 nm to 25 nm from localizer antenna and course cuts from 0 to 90 degrees. Simulation results are detailed in Reference 4. Table 5-1 provides a concise summary of the developed system.

5.3 Details of the Study

5.3.1 Framing of the Strategy

Figure 5-1 illustrates three potential situations that may occur when performing an R-NAV aided capture. For convenience they will be referred to as the "turn short", "good data", and "turn long" cases. For captures with trip point within the beam, as shown in the fourth case of Figure 5-1. no R-NAV aiding is needed. Also shown in Figure 5-1 are the required bank traces for each case if a successful capture is to occur. The following argument serves to validate the selection of these three R-NAV aided cases as basic.

If one is to start the capture based on R-NAV data, he has no choice but to establish a trip point algorithm based on area navigation data and consequently must design a system to be tolerant of trip point shifts due to R-NAV errors. Since capture trip points in general must be a function of velocity, this means not only cross track distance errors but also velocity errors will affect the trip point computation. Assuming a constant bank capture algorithm, the three possible situations that may exist once the capture has begun are shown in Figure 5-1 as discussed above. The "turn long" case is critical in terms of amount of bank command to use. If the system is turning long then a "reserve" of bank command must be available (once beam data is present) to increase the bank command to the required amount. Hence, since the "turn long" case may be present, the nominal bank in an R-NAV aided capture should not be the bank command limit (normally about 30°). Note, however, that this is not an absolute constraint but rather a consequence of the strategy being considered here. One could always force the system to "turn short" (see below) and thus avoid the "turn long" case entirely, but the strategy adopted here seems more natural to the author. In summary, the approach adopted is to leave some room for bank to increase in case the "turn long" case occurs.

CAPTURE TRIP POINT COMPUTATIONS

CAPTRP = Cross Track Distance Trip Point

$$\text{CAPTRP} = \begin{pmatrix}\text{CAPNAV}\\ \text{or}\\ \text{CAPILS}\end{pmatrix} + \text{CTD.ADJ}$$

CAPNAV = Trip Point for RNAV initiated capture - - - $= \dfrac{V_g^2(1-\cos(\text{TAE}))}{g\tan(\text{CAP.BNK})}$

CAPILS = Trip Point for ILS initiated capture - - - $= \dfrac{(\text{CTR.BCI})^2}{(g\tan(\text{CAP.BNK}))(1+\cos(\text{TAE}))}$
(CTRBCI is cross track rate from the beam data filter)

CTD.ADJ = Compensation for bank rate command limit (RCL) $= \dfrac{\text{CAP.BNK}}{\text{RCL}} * V_g \sin(\text{TAE})$

CAP.BNK = Bank (deg) to be used in capture - - - $= \begin{cases} -\text{MAX.CAP.BNK} * \dfrac{\text{TAE}^\circ}{90} \text{..if magnitude of result is} \geq 5^\circ \\ -5. * \text{sign(TAE)..otherwise} \end{cases}$

MAX.CAP.BNK = Programming parameter on CAP.BNK, - - - $= 8^\circ + \dfrac{17}{13} * \text{RNGLOC} \leq 25^\circ$
Corresponds to CAP.BNK at TAE = 90°.
RNGLOC is distance (nm) to localizer antenna.

CAPTURE INHIBITS

Capture is inhibited at computed (based on R-NAV data) deviations greater than nominally 7° to guard against decreasing localizer receiver output at wide angles. This computation is sensitive to ANAV errors and could compute a value as low as 4° or as high as 10° with typical A-NAV errors. Applicable software variable is "DEVFLG".

NAVCAP is inhibited if the range is such that R-NAV aiding is not needed. This prevents early trip (NAVCAP) in those cases where CTD errors from the ANAV would initiate (unnecessarily) capture outside the beam edge. Applicable software variable is "ILSLOK".

Localizer deviation (LOCDEV) is rate command limited (limit is programmed with TAE and airspeed) to protect against sudden transients which could cause false captures as well as command transients. Applicable software variable is MXDLDV ("maximum delta in deviation").

Table 5-1. Summary of RNAV Aided Localizer Capture and Track Laws

BANK COMMAND DURING CAPTURE

BCMDNC - Bank command "NAV CAP" based on R-NAV data (CTDNAV is cross track distance from R-NAV) - - - - - $= \frac{Vg^2(1-\cos(TAE))}{g\ CTDNAV}$

BCMDBC - Bank command "beam capture" based on ILS data (CTRBC = CTRBCI*COS(TAE)+(Vg sin TAE)*(1-cos TAE) where CTRBCI and CTDBC are cross track rate and distance respectively from the beam data filter. - - - $= \frac{(CTRBC)^2}{g\ CTDBC(1+\cos(TAE))}$

BCMDNC fades to BCMDBC with a 1 sec exponential fade at beam interception.

If "turn short" is detected BCMDNC fades by the algorithm below until beam intercept. Turn short test is |TAE|< .95*TAEBE where TAEBE is the magnitude of the predicted track angle error at beam intercept (see text)

$$BCMDNC(t) = BCMDNC(t_S)*e^{-\left[\frac{(t-t_S)}{\tau_B}\right]}$$

where t_S is the time at which "turn short" is detected and the fade is started;

τ_B is a computed time constant (see text) that fixes the permissable additional heading change in NAVCAP to insure against turning parallel to the beam.

Table 5-1 Continued

TRACK TRIP POINT COMPUTATIONS

Track Computations are used when

$\text{LOCDEV} < 30\ \mu\text{A}$ for $\text{RNGLOC} \leq 13$ nm

or $\text{LOCDEV} * \text{RNGLOC} < .091$ nm for $\text{RNGLOC} > 13$ nm

or $|\text{TAE}| < .9 * \text{TAETRK}$

where TAETRK is the magnitude of the predicted track angle error at track intercept (i.e. 30 μA or .091 nm as appropriate)

Capture Computations fade exponentially to track computations with a 5 second time constant.

The TAE test is to prevent turning parallel to the beam in extreme cases and should not normally be the determining factor in initiating Track.

TRACK COMPUTATIONS

$$\text{BNK.CMD} = \text{RNGPGM}(\text{KCTDTRK} * \text{CTDBC} + \text{KCTRTRK} * \text{CTRBC})$$

$$\text{where RNGPGM} = \begin{cases} 1 & \text{for RNGLOC} \leq 7 \text{ nm} \\ \dfrac{7}{\text{RNGLOC}} & \text{for RNGLOC} > 7 \text{ nm} \end{cases}$$

and RNGLOC is distance to localizer antenna.

CTDBC and CTRBC are cross track distance and rate respectively from the beam data filter.

KCTDTRK = (3.2 deg/70 ft.)

KCTRTRK = (35 deg/70 fps).

Table 5-1 Continued

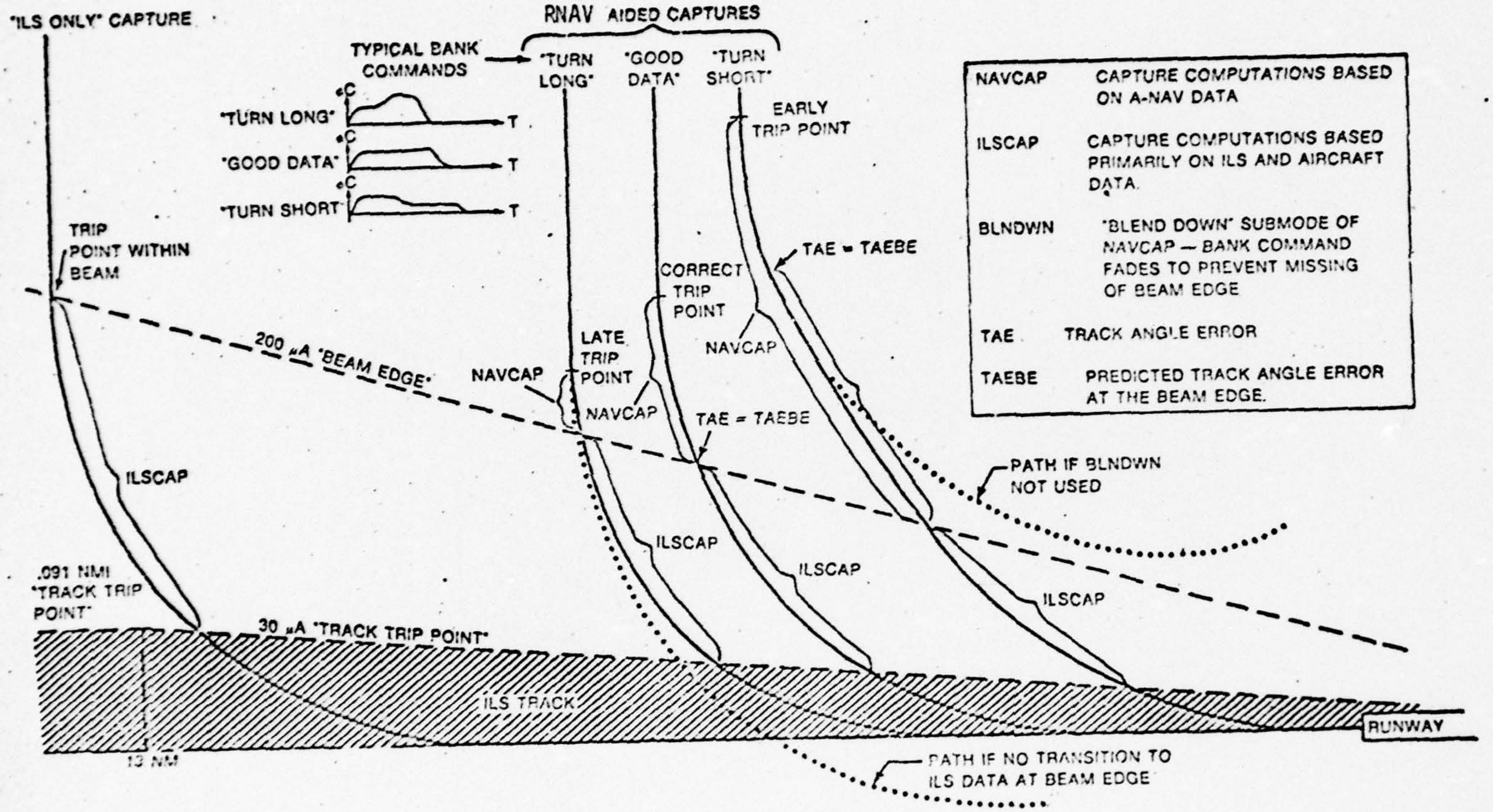


Figure 5-1. Area Nav Aided Localizer Capture.

The other disconcerting situation that can occur is the "turn short" case. Here one must be very careful because the beam could be missed entirely (i.e. beam intercept might not occur). To guard against this situation, some means must be present to detect when the system has turned far enough. Recall that Reference 1 took the approach of never letting the system turn to less than a 45° course cut relative to the beam. The approach here is similar in that it restricts course cut relative to the beam, but is a new development in that it conditions the permissable minimum course cut based on the state of the system and provides a precise bank command fade algorithm which is a function of the computed minimum course cut. This will be discussed in more detail in Section 5.3.2. For the moment, we note that in the "turn short" case a "blend down" algorithm must be provided to prevent turning parallel to the beam and to smoothly transition bank command to beam based data.

Several design problems are posed by the discussion above:

1. A capture trip point algorithm is needed which keeps nominal bank in capture significantly less than the bank command limit to handle the "turn long" case.
2. A "blend down" algorithm must be provided to handle the "turn short" case.
3. Good data filtering is needed to provide noise suppression without excessive lag in order that the real system will closely approximate the theoretical system.
4. Smooth transition between modes and data bases must be provided.
5. The individual pieces must be melded into a composite system.

5.3.2 "Blend-Down" Algorithm

As discussed above, the need for a blend-down algorithm arises in the "turn short" case. There are two aspects of the blend-down problem.

1. When should the blend-down be initiated?
2. How should the blend-down be programmed?

The first of these questions requires addressing the problem of "when has the system turned far enough?"

From Reference 2, if the system is executing a constant bank turn onto final course, the bank command (BNK-CMD) should obey the relationship

$$\text{BNK.CMD} = -\tan^{-1}\left(\frac{Vg^2 * (1 - \text{COS(TAE)})}{g * \text{CTD}}\right) \qquad (5.1)$$

where

Vg is ground speed

TAE is track angle error or angle of ground velocity vector relative to the runway centerline

CTD is cross track distance to localizer centerline

g is gravitational constant (32.17 ft/sec^2)

and consistent units are assumed. Equation 5.1 holds through the entire capture maneuver; and if all assumptions are satisfied, TAE and CTD change in such a way that BNK.CMD remains constant, assuming no wind. If there is a wind, the bank must change during the capture in order to maintain a circular track over the ground.

In particular, for a capture initiated outside the beam on R-NAV data and neglecting wind for the moment, Equation 5.1 should hold at the instant of beam intercept. Now at beam intercept, since one knows range and localizer deviation (LOC.DEV), he can compute cross track distance at the beam edge (CTDBE). Further, since Vg and BNK.CMD are known, one can compute the track angle error at the beam edge (TAEBE) by solving Equation 5.1 for TAE as

$$\text{TAEBE} = \cos^{-1}\left[1. - \frac{g*\tan(-\text{BNK.CMD})*\text{CTDBE}}{Vg^2}\right] \qquad (5.2)$$

where

$$\text{CTDBE} = \text{Range} * \text{Sin}\ (\text{LOC.DEV})$$

with appropriate units used. If wind is present Equation 5.2 will still hold at any particular instant in the turn. However, in simulation, certain combinations of errors caused BNK.CMD to converge in such a way that the TAEBE computation converged toward zero. Since this is highly undesirable it was judged best to compute TAEBE once only at capture initiation and accept any resulting anomalies due to wind, which should not be too serious since wind will still be factored correctly into command. Equation 5.2 is basically only used to define the trip point for transition to the blend-down submode. That is, one computes TAEBE at the initiation of capture and as the capture proceeds, he continually tests TAE against TAEBE. If the condition TAE = TAEBE occurs prior to encounter of the beam edge, the system is presumably turning too fast and fading of the bank command should begin. If beam encounter occurs prior to TAE = TAEBE, then the system should fade immediately to bank command based on beam data.

The second aspect of the blerd-down problem is the question of how to fade the bank. Assuming an exponential fade on bank command, the following development shows that the time constant of the fade can be computed such that for a fade beginning at the point TAE = TAEBE, the system will never turn further than to TAE = k*TAEBE where $o < k < 1$. That is, a parameter k can be specified such that under no conditions will the system turn parallel to the beam. For example, if TAEBE = 30° and k = .5, then the system will always intersect the beam at TAE ≥ k*TAEBE = 15°. The expression for faded bank command is given in Equation 5.3

$$\phi_c(t) = \phi_c(to) \exp\left[\frac{-(t-to)}{\tau}\right] \qquad (5.3)$$

where

t is time in seconds

t_o is time at which the fade begins

$\phi_c(t)$ is bank command

$\phi_c(to)$ is bank command just before fade starts

If the maneuver is coordinated and neglecting lags between bank and bank command

$$\dot{\psi}(t) = \frac{g \tan \phi_c(t)}{Uo} \approx \frac{g\ \phi_c(t)}{Uo} \qquad (5.4)$$

where Uo is airspeed. Substituting from equation (5.3)

$$\dot{\psi}(t) = \frac{g}{Uo}\ \phi_c(to) \exp\left[\frac{-t-to}{\tau}\right] \qquad (5.5)$$

Integrating equation 5.5 from t = to to t = ∞ yields

$$\psi(\infty) = \psi(to) + \frac{g\tau}{Uo}\ \phi_c(to) \qquad (5.6)$$

or

$$\Delta\psi = \frac{g\tau}{Uo}\ \phi_c(to)$$

where

$\Delta\psi = \psi(\infty) - \psi(to)$ is the change in ψ that would occur if the fade continued indefinitely.

Thus

$$\tau = \frac{Uo\ \Delta\psi}{g\phi_c(to)} \qquad (5.7)$$

is a relation for the time constant of the fade which permits us to specify the maximum permissible change in heading during the fade*.

---

*Admittedly a number of approximations are made here, notably $\phi \approx \tan\ \phi$ and neglecting of bank loop lags, but inclusion of these details would be somewhat gruesome and would not alter the essence of the development.

An appropriate choice for $\Delta\psi$ seems to be

$$\Delta\psi = k * TAEBE$$

where $o<k<1$.

Substituting this choice for $\Delta\psi$ into Equation 5.7 yields

$$\tau = \frac{Uo\ (k * TAEBE)}{g\ \phi(to)} \tag{5.8}$$

as an expression for the fader time constant to be used in the blend-down submode.

5.3 Capture Trip Point

The basic requirement for capture trip point is that it be such that reasonable banks are used in capture and reasonable overshoots occur under all conditions. Generally bank command is restricted to less than 30° in magnitude and it is highly desirable to keep overshoots less than $30\mu A (\approx .4°)$ of localizer deviation. Other factors may also enter. For example, if no R-NAV system is available, clearly the trip point must be within the beam resulting in geometry limits for certain close-in captures. Pilots often have very definite opinions about how the bank should behave under given conditions. It should also be apparent that the conditions above involve the form of the capture bank command at least implicitly. That is, the capture trip point is obviously not independent of the bank to be used in capture. if one assumes a circular capture (i.e. constant bank), it is interesting to solve Equation 5.1 for CTD to illustrate the CTD required to execute a circular capture as a function of Vg, TAE, and BNK.CMD

$$CTD = \frac{Vg^2(1 - \cos (TAE)\ )}{g \tan\ (BNK.CMD)} \tag{5.9}$$

Equation 5.9 illustrates the generally applicable point that capture trip point expressed in CTD terms is a function of ground speed, TAE, and bank command to be used in the capture. Since it appeared feasible to specify a desired bank during capture, the approach decided on in this study was to specify the capture bank (CAP.BNK) and then use Equation 5.9 to determine the trip point, specifically

$$\text{CTD Trip Point} = \frac{Vg^2 * (1 - \cos\ (TAE))}{g \tan\ (CAP.BNK)} \tag{5.10}$$

Further, it was felt that CAP.BNK should be a function of TAE, that is at high angle course cuts more bank should be used than for lower angle course cuts. Further, to prevent very small bank captures, the restriction was added that $|CAP.BNK| \geq 5°$ in all cases. Summarizing this relationship, the capture bank in degrees is given by

$$CAP.BNK = \begin{cases} \dfrac{-MAX.CAP.BNK}{90°} * TAE \ldots \text{(if magnitude of result } \geq 5°\text{)} \\ -5 * \text{Sign } (TAE) \ldots \text{(otherwise)} \end{cases} \tag{5.11}$$

where TAE is in degrees. Before choosing MAX.CAP.BNK it is important to consider the following factors.

Recall from the earlier discussion of the "turn long" case that one must leave room for the bank to increase once the beam is encountered. The question is, "How much increase space is needed?" Figure 5-2 provides some insight.

Figure 5-2, based on Equation 5.1, shows (assuming a circular capture) CAP.BNK versus CTD with TAE as a parameter for a ground speed of 200 fps. For example, if a capture is begun at .5 nmi and TAE is 90°, then 22.3° of bank must be applied instantaneously and held through the turn. Consider, however, the situation when CTD from RNAV is in error by .2 nmi such that the capture in reality begins at .3 nmi instead of .5 nmi. The required instantaneous bank for this case is 34.3° to execute a no overshoot capture. Obviously, with a 30° bank command limit, this cannot be achieved. Further, the inclusion of rate command limits and inherent system lags will make this situation worse (i.e., bank cannot step instantaneously from 0° to CAP.BNK).

Two things are important here. First, the trip point should be such that .2 nmi errors do not cause the required CAP.BNK to exceed 30°. Secondly, the rate command limit turns out to be rather significant and the trip point should be moved to account for its effect. This is most simply done by noting that the approximate time required to build to CAP.BNK is

$$\text{Time-to-CAP.BNK} = \frac{\text{CAP.BNK}}{\text{RCL}}$$

where RCL is the bank rate command limit. Further the CTD traversed in this interval of time is (neglecting any turning in this interval)

$$\text{CTD.ADJ} = \frac{\text{CAP.BNK}}{\text{RCL}} * \text{Vg}\ \text{Sin(TAE)} \qquad (5.12)$$

Hence the trip point as computed from Equation (5.10) should be incremented by Equation (5.12) with CAP.BNK determined from Equation (5.11). Figure 5-3 represents Figure 5-2 corrected by equation 5.12 to account for the time to build to the required capture bank. Given the value of CAP.BNK and TAE, therefore, the CAPTURE trip point may be read from Figure 5-3 for the case Vg = 200 fps.

The only item as yet unresolved is MAX.CAP.BNK., the maximum bank to be used in capture which as shown in Equation 5.11 becomes one of the significant programming parameters on CAP.BNK. In the event one is doing an RNAV aided capture and the "turn long" case arises, the required "room to increase" on bank once the beam is encountered is a function of range. To see this, consider captures beginning at the same CTD and course cut but at varying range (RNAV data is used until beam intercept). Intuitively, the further out one is in range, the more beam width in feet is available after beam intercept to correct the problem or conversely less time is spent using an erroneous command.

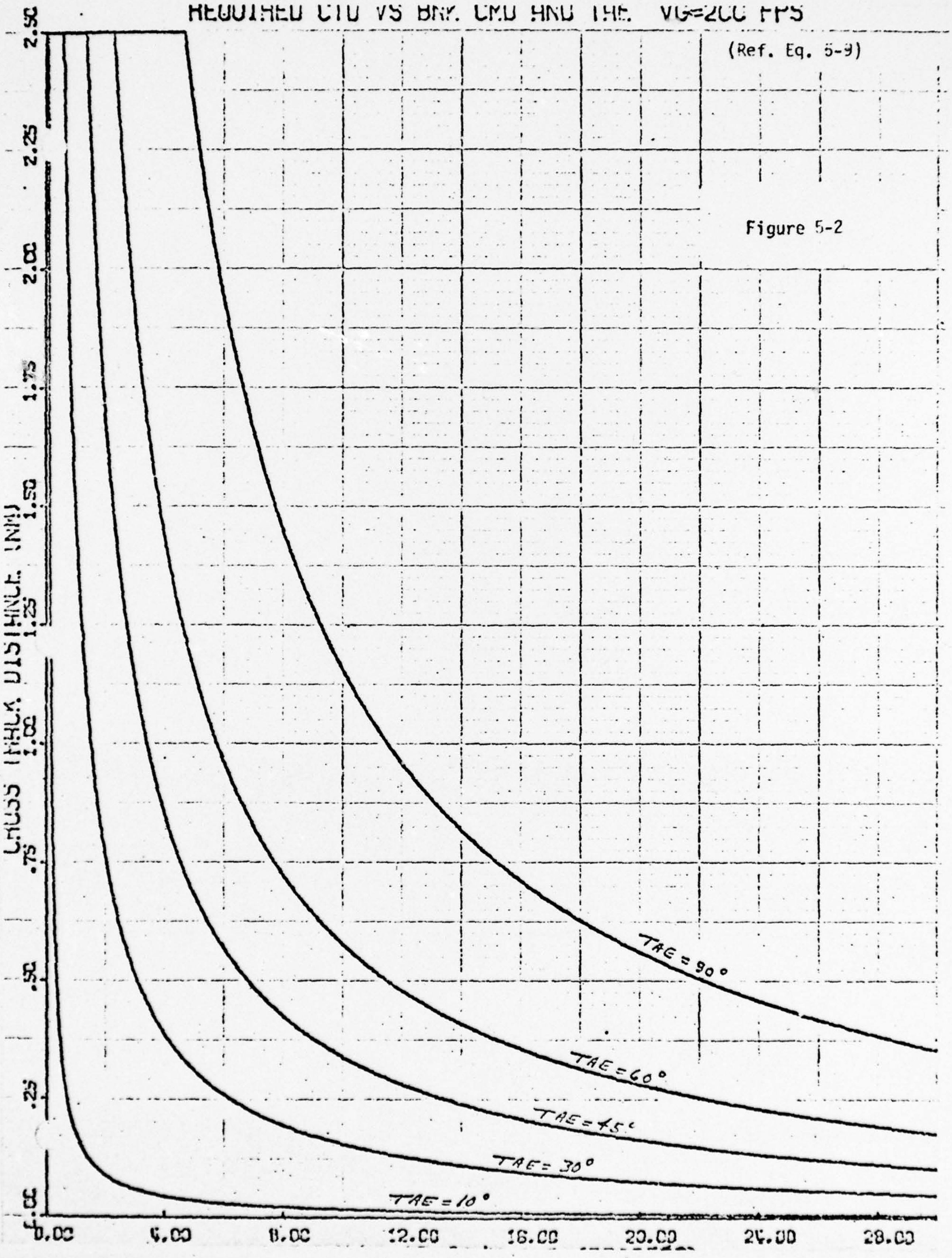
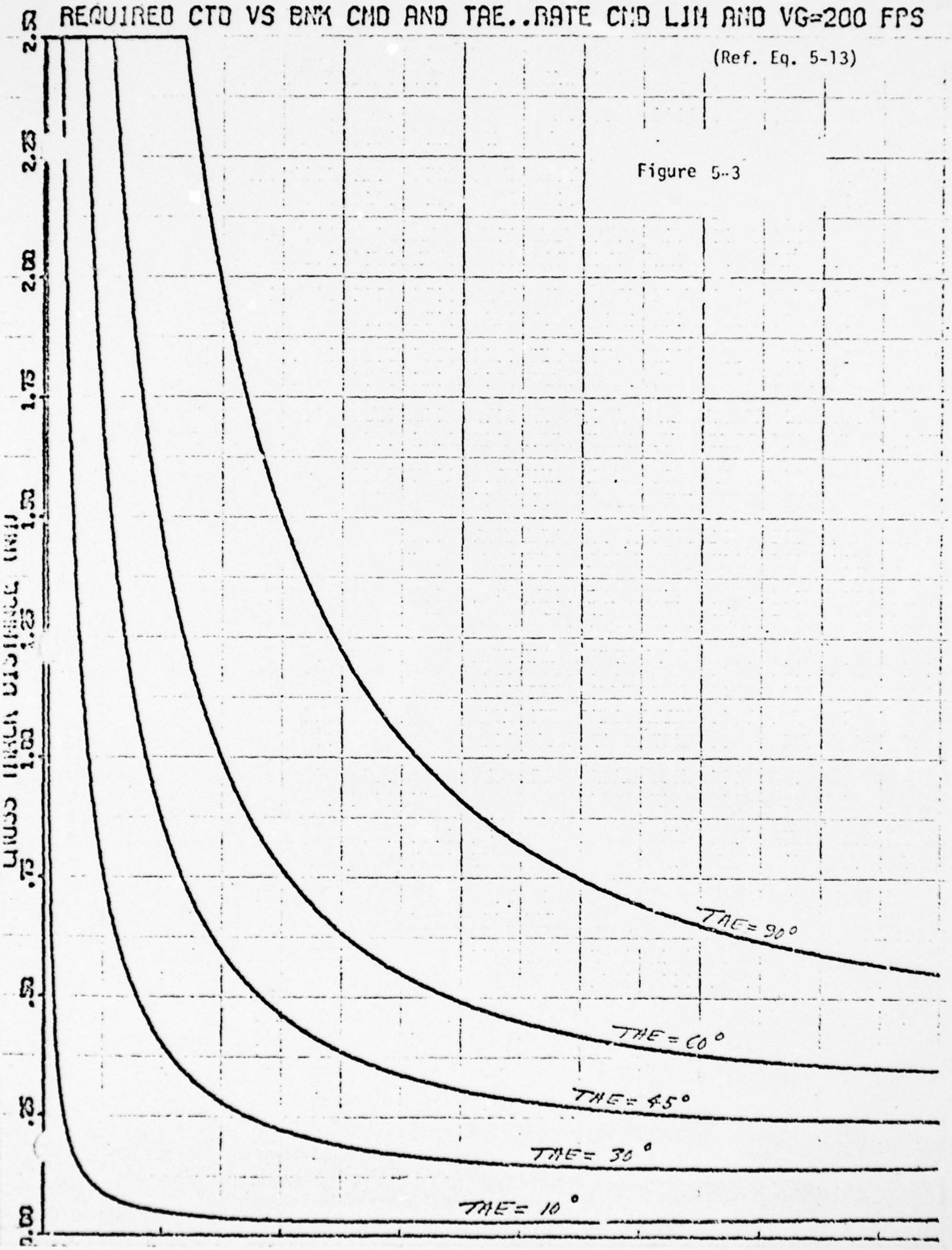

REQUIRED CTD VS BRK. CMD AND TAE. VG=200 FPS
(Ref. Eq. 5-9)
CROSS TRACK DISTANCE (NM)
2.50
2.25
2.00
1.75
1.50
1.25
1.00
.75
.50
.25
0.00
0.00
4.00
8.00
12.00
16.00
20.00
24.00
28.00
TAE = 90°
TAE = 60°
TAE = 45°
TAE = 30°
TAE = 10°

Figure 5-2



(Ref. Eq. 5-13)

Figure 5-3

For example, consider 90° captures at 5 nm and 8 nm respectively at a groundspeed of 200 fps and using -20° of bank in the capture. From Equation 5.2, if there are no errors, the respective track angle errors at the beam edge (TAEBE) should be

$$\mathrm{TAEBE}_{8nm} = 70.25°$$

$$\mathrm{TAEBE}_{5nm} = 54.11°$$

where a 200 μA "beam edge" is assumed. Now in either case at 20° of constant bank the rate of turn is also constant and has the value

$$\dot{\psi} = \frac{g \tan\phi}{Vg} = -3.35°/\mathrm{sec}$$

Now if the trip point in each case is .2 nm late, the capture starts τ_D seconds late where τ_D is given by

$$\tau_D = \frac{.2\ \mathrm{nm}}{200\ \mathrm{fps}} = 6.076\ \mathrm{sec.}$$

Therefore in each case $\Delta\tau$ of heading change would be lost where

$$\Delta\psi = \tau_D * \dot{\psi} = 20.35°$$

Hence the real TAEBE in each case is (adding $\Delta\psi$ to the TAEBE values above)

$$\mathrm{TAEBE}_{8nm} = 90°$$

$$\mathrm{TAEBE}_{5nm} = 74.46°$$

Further, the respective cross track distances at the beam edge (CTDBE) are

$$\mathrm{CTDBE}_{8nm} = .37\ \mathrm{nm}$$

$$\mathrm{CTDBE}_{5nm} = .23\ \mathrm{nm}$$

Thus the required bank to complete the capture in each case is (from Equation 5.1)

$$\mathrm{CAP.BNK}_{8nm} = -28.9°$$

$$\mathrm{CAP.BNK}_{5nm} = -33.1°$$

Notice that for the capture at 8 nm with .2 nm of error in CTD, (28.9°-20.0°)= 8.9° of bank increase was required while at 5 nm (33.1°-20.0°)=13.1° of bank increase was required. This example supports the contention that MAX.CAP.BNK should decrease as range decreases. Further, it is apparent and desirable that R-NAV aiding be used only when necessary.

If the trip point is within the beam, then capture will not begin until beam data is present. If MAX.BNK.CAP is made sufficiently large at longer ranges, it will force the capture to be within the beam in those cases (see Equation 5.10). It was decided to make MAX.CAP.BNK = 25° at longer ranges to insure localizer based captures there and to decrease the value at shorter ranges. From the example above, at 5 nm a bank increase of 13.1° was required in the "turn long" case while at 8 nm a bank increase of 8.9° was required. Further, it can be shown that for a 90° intercept at 200 fps, MAX.CAP.BNK = 25° implies R-NAV aided captures at ranges shorter than 13 nm range; hence this seems a reasonable point to start programming MAX.CAP.BNK. down. Simulation results supported the 5 nm conclusions above and indicated that slightly more increase was required for the case of a 20% ground velocity error in the slow direction. For this reason, provision of about 15° possible increase at 5 nm seemed advisable. The resulting range program for MAX.CAP.BNK is

$$\text{MAX.CAP.BNK} = 8. + \frac{17.}{13.} * \text{RNGLOC} \tag{3.13}$$

where RNGLOC is range to localizer in nm and the units on MAX.CAP.BNK are degrees.

Collecting the above discussion, the trip point computation is performed as follows

$$\text{CTD Trip Point} = \frac{Vg^2(1-\text{COSTAE})}{g\ \tan(\text{CAP.BNK})} + \text{CTD.ADJ} \tag{3.13a}$$

where

$$\text{CTD.ADJ} = \frac{\text{CAP.BNK}}{\text{RCL}} * Vg\ \sin(\text{TAE}) \tag{3.13b}$$

and

$$\text{CAP.BNK} = \begin{cases} -\text{MAX.CAP.BNK} * \dfrac{\text{TAE}}{90} \ \ldots \text{ if magnitude of result is } \geq 5° \\ -5 * \text{SIGN(TAE)} \ \ldots \text{ otherwise} \end{cases} \tag{3.13c}$$

and

$$\text{MAX.CAP.BNK} = 8. + \frac{17}{13} * \text{RNGLOC} \leq 25° \tag{3.13d}$$

CTD trip point as a function of range and course cut is shown in Figures 5-4 for a speed of 200 fps. Superimposed are lines for 200 μA (ILS CAP "beam-edge") localizer deviation and "track trip". These are significant boundaries for they delineate NAVCAP, ILSCAP, and track regions respectively.*

An alternate presentation of trip point information is shown in Figure 5-5. Here, for constant course cuts of 90 degrees, CTD trip point is plotted versus ground speed and range.

5.3.4 Track Trip Point

The transition from capture to track normally occurs when

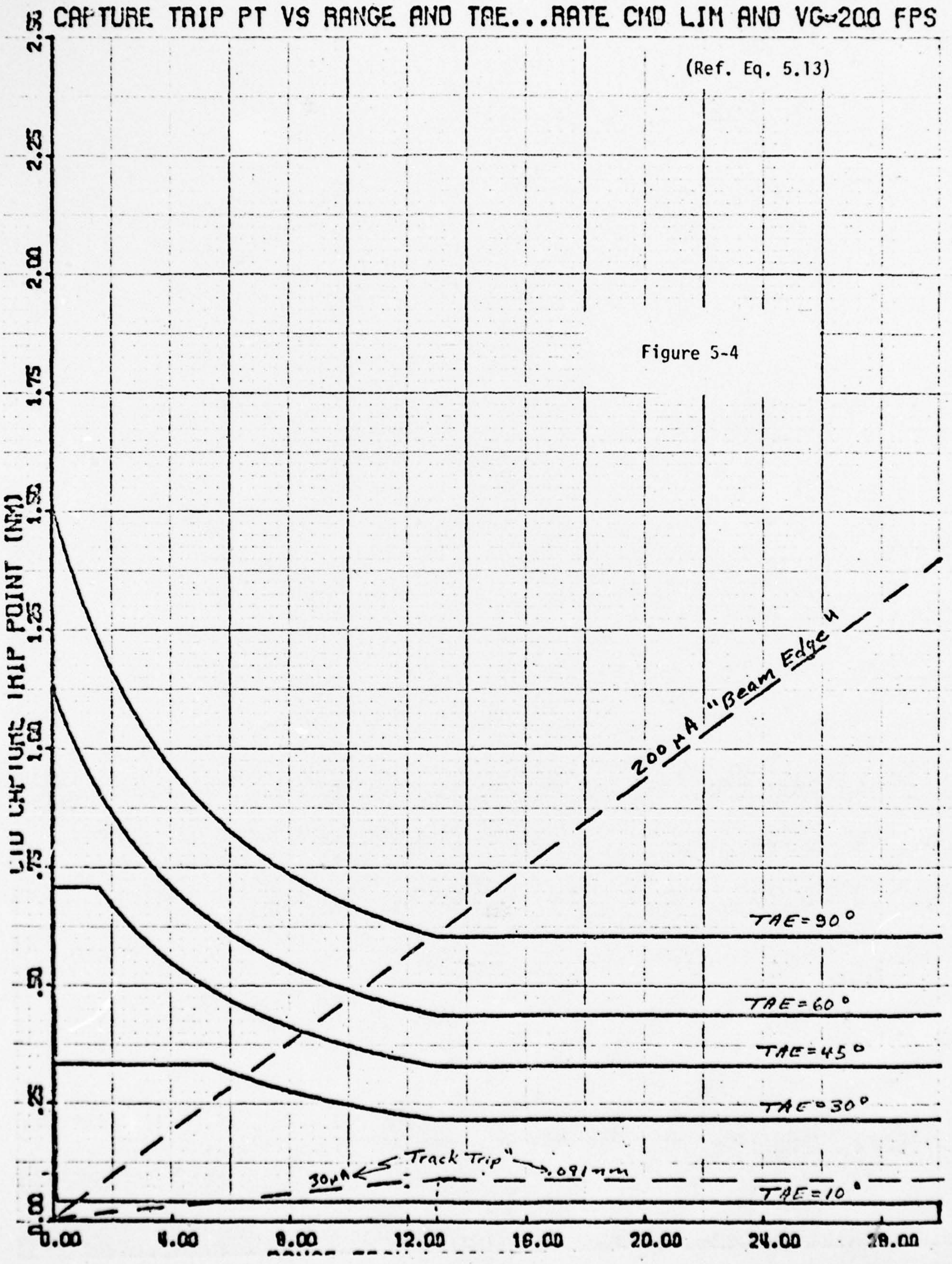
$$\text{LOCDEV} < 30\ \mu\text{A for RNGLOC} \leq 13 \text{ nm}$$

and for

$$\text{LOCDEV} * \text{RNGLOC} < .091 \text{ nm for RNGLOC} > 13 \text{ nm}$$

---

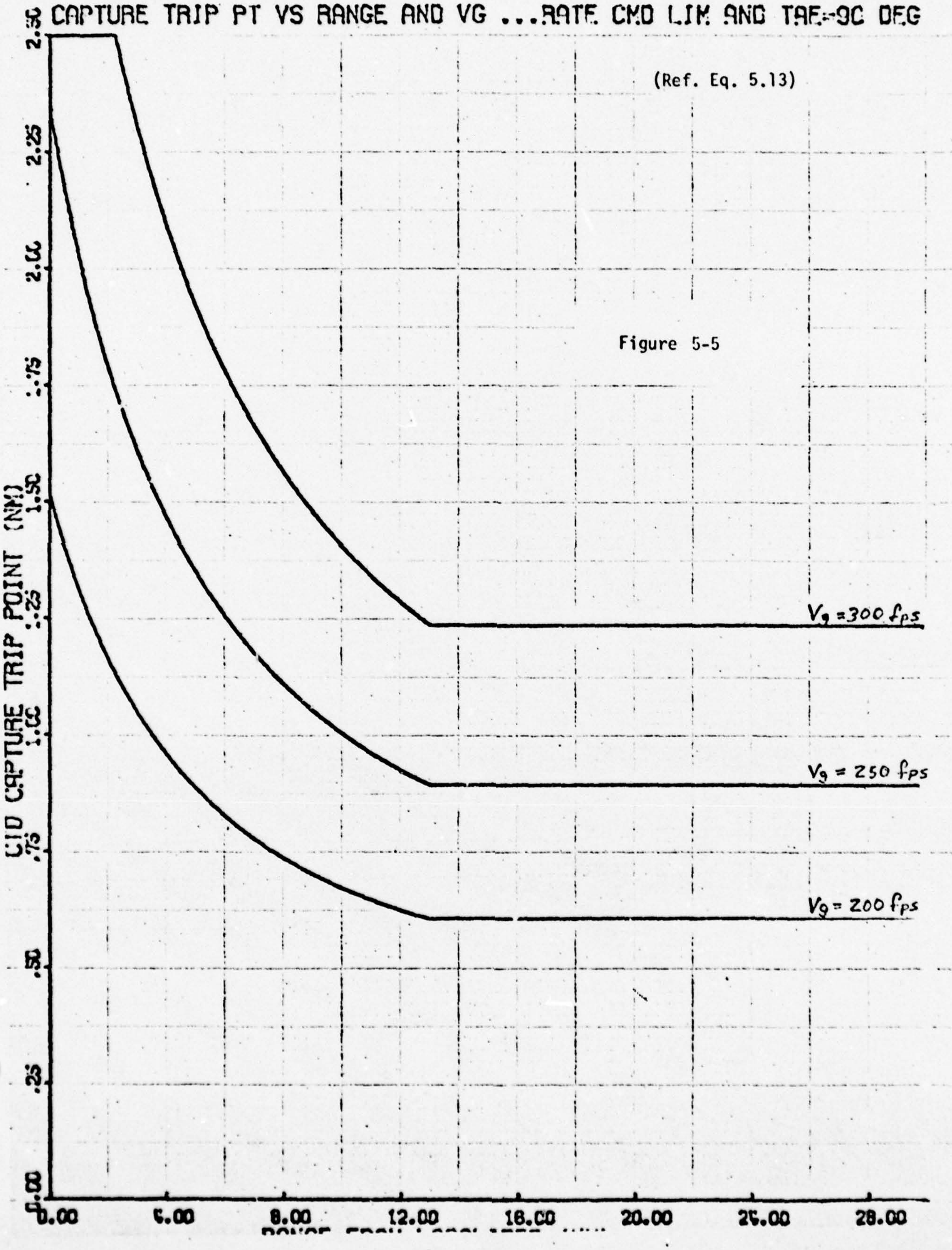
*If the capture trip point lies within the "Track Trip" boundary, capture mode is deleted and the system transitions directly from heading mode to track mode.



CAPTURE TRIP PT VS RANGE AND TAE...RATE CMD LIM AND VG=200 FPS

(Ref. Eq. 5.13)

Figure 5-4



(Ref. Eq. 5.13)

Figure 5-5

The transition from angular to linear CTD criterion for transition to track prevents going to track at high values of TAE at long ranges where 30 μA represents significant linear deviation. The advantage here lies in maintaining capture mode and hence the circular nature of the capture as long as possible. However, in the event of severe crosswinds or unusually bad initialization the possibility of turning parallel to the beam during ILSCAP exists. To provide a final safeguard against turning parallel the following override trip to track feature is provided.

Since in ILSCAP the system is still performing a circular capture a slight derivative of Equation 5.2 for the TAEBE (track angle error at the beam edge) can be used to compute TAETRK (track angle error at track trip point).

$$\text{TAE TRK} = \cos^{-1}\left[1 - \frac{g \star \tan(-\text{BNKCMD}) \star \text{CTDTRK}}{Vs^2}\right] \tag{5.14}$$

where

$$\text{CTD TRK} = \begin{cases} \left(30\mu A \star \dfrac{\text{RNGLOC}}{(75)(57.3)}\right) & \text{nm, RNGLOC} \leq 13 \text{ nm} \\ .091 & \text{nm, RNGLOC} > 13 \text{ nm} \end{cases}$$

The system tests for ($|\text{TAE}| < .9 \star \text{TAETRK}$), computing TAETRK via Equation 5.14; and if this condition occurs during ILSCAP. generates an override trip into the TRK mode. The flattening of the TAETRK curves at long ranges is due to the fixed CTD track trip point at ranges greater than 13 nmi. The flattening at short ranges is due to a bottom limit of 7° imposed on the computation (actually 7.73° on TAETRK so that $.9 \star \text{TAETRK} \geq 7°$) to ensure that this final test never drops below that value. That is, the system is constrained under all circumstances to go to track from ILSCAP if TAE drops less than 7 degrees.

5.3.5 Data Filtering

All of the foregoing discussion has neglected noise and lags on sensor data. In order for the conclusions and predictions above to be accurate, the data filtering, particularly on localizer data, must not introduce excessive lag. The often used system of about one second lag on radio and a complementary filter using radio and bank or heading to derive beam rate introduces lags that significantly compromise the performance of the system described here based on simulation results. For this reason a second order data filter for radio was developed which employs a bank input to provide complementing for both beam rate and position. With this filter, better performance in capture resulted and bank activity in track due to beam noise was very significantly decreased. This filter is described in detail in Reference 3 and performance is documented in Reference 4, hence it will not be discussed further here.

5.3.6 Mode Transition

As implied in the preceeding discussion, there are several distinct modes in the system that have evolved from this study. It is assumed that there is a heading select mode which is controlling the aircraft to a fixed heading (or course) prior to initiation of capture. Capture may be initiated either on R-NAV data or on beam data. In the former case, a blend-down submode may occur, and eventual transition to beam based capture must occur. Finally, a track mode assumes control once capture is completed. The transitions from mode to mode must be handled smoothly with no switching transients. The general mode switching smoothing technique is shown in Figure 5-6. The fade is basically a digitized exponential fade with 1 to 5 second time constants being typical.

5.3.7 Final System Description

Capture Modes

A block diagram of the total R-NAV aided ILS localizer capture and track system is shown in Figure 5-7. It should be noted that there are basically two potential capture modes. Capture computations are based on R-NAV data (NAVCAP) or based on ILS data (ILSCAP). Further there is a blend-down submode (BLNDWN) of NAVCAP which is provided to fade the bank command down in the "turn short" case. If R-NAV aiding is needed. the NAVCAP mode will occur first, transitioning to ILSCAP when localizer deviation becomes less than 200 μA. If during NAVCAP the system turns too far [determined by continually comparing TAE with predicted TAE at the beam edge (TAEBE) during NAVCAP], the BLNDWN submode is automatically selected and fades the bank command according to the fade algorithm discussed in Section 5.3.2. ILSCAP can occur in two ways. First, if NAVCAP is in progress and localizer deviation decreases below 200 μA, then ILSCAP is selected immediately. The other case occurs when the capture trip point is such that localizer deviation at that point will be less than 200 μA. In this case no R-NAV aiding is needed and the ILSCAP mode is selected immediately. In terms of logic equations, the capture conditions may be expressed as

$$\mathrm{NAVCAP} = [(\mathrm{LOC\ DEV} > 200\,\mu\mathrm{A}) \bullet (\mathrm{CTDNAV} < \mathrm{CAPTRP})] \cdot (\mathrm{RNGLOC} < \mathrm{ILSLOK}) \qquad (5.15)$$

$$\mathrm{ILSCAP} = [(\mathrm{LOCDEV} \leq 200\,\mu\mathrm{A}) \bullet (\mathrm{NAVCAP} + (\mathrm{CTDILS} < \mathrm{CAPTRIP}))] \cdot (\mathrm{DEVFLG}) \qquad (5.16)$$

$$\mathrm{BLNDWN} = \mathrm{NAVCAP}\colon (|\mathrm{TAE}| \leq |\mathrm{TAEBE}|) \qquad (5.17)$$

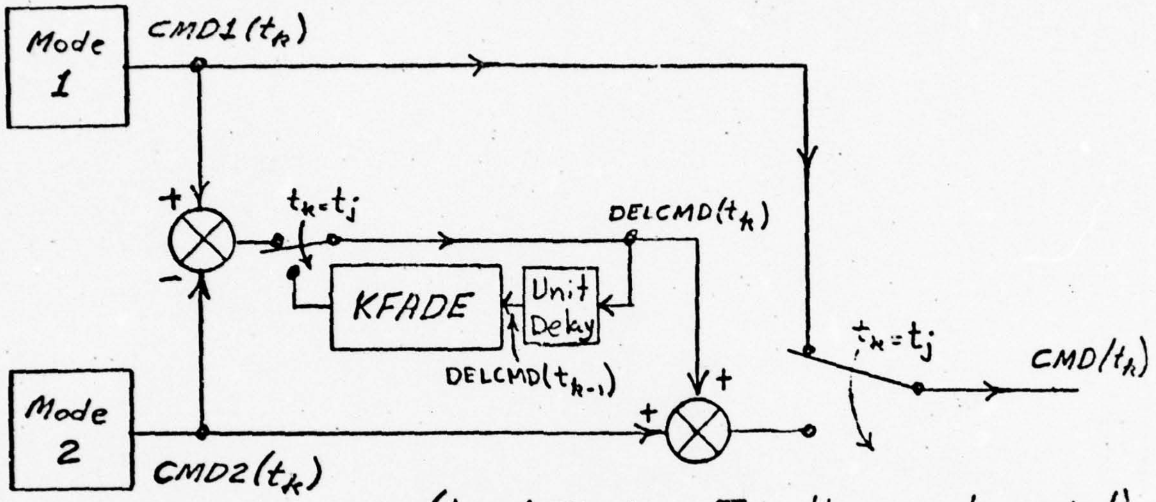
where ILSLOK and DEVFLG are added protections against R-NAV trip at excessive range due to CTDNAV errors and false ILSCAP trips due to decreasing LOCDEV at wide angle* respectively.

False Trip Guards and Deviation Rate Limit

Both of these protections are shown in Figure 5-8. DEVFLG basically inhibits localizer computations at greater than a nominal deviation of 7° from the beam centerline. Seven degrees was chosen so that even with R-NAV errors the computed value will lie reliably between 4° and 10°.

---

*Localizer signal strength in a +10 degree sector should be adequate to saturate the receiver but may drop off at wider angles.



($t_k = kT$ where T is the sample period)

(Mode changes from Mode 1
to Mode 2 at time $t_k = t_j$)

$$KFADE = e^{-T/\tau}$$

1. For $t_k < t_j$ $\quad CMD(t_k) = CMD1(t_k)$

   $DELCMD(t_k) = CMD1(t_k) - CMD2(t_k)$

2. At $t_k = t_j$ $\quad DELCMD(t_j) = CMD1(t_j) - CMD2(t_j)$

   $CMD(t_j) = CMD2(t_j) + DELCMD(t_j) = CMD1(t_j)$

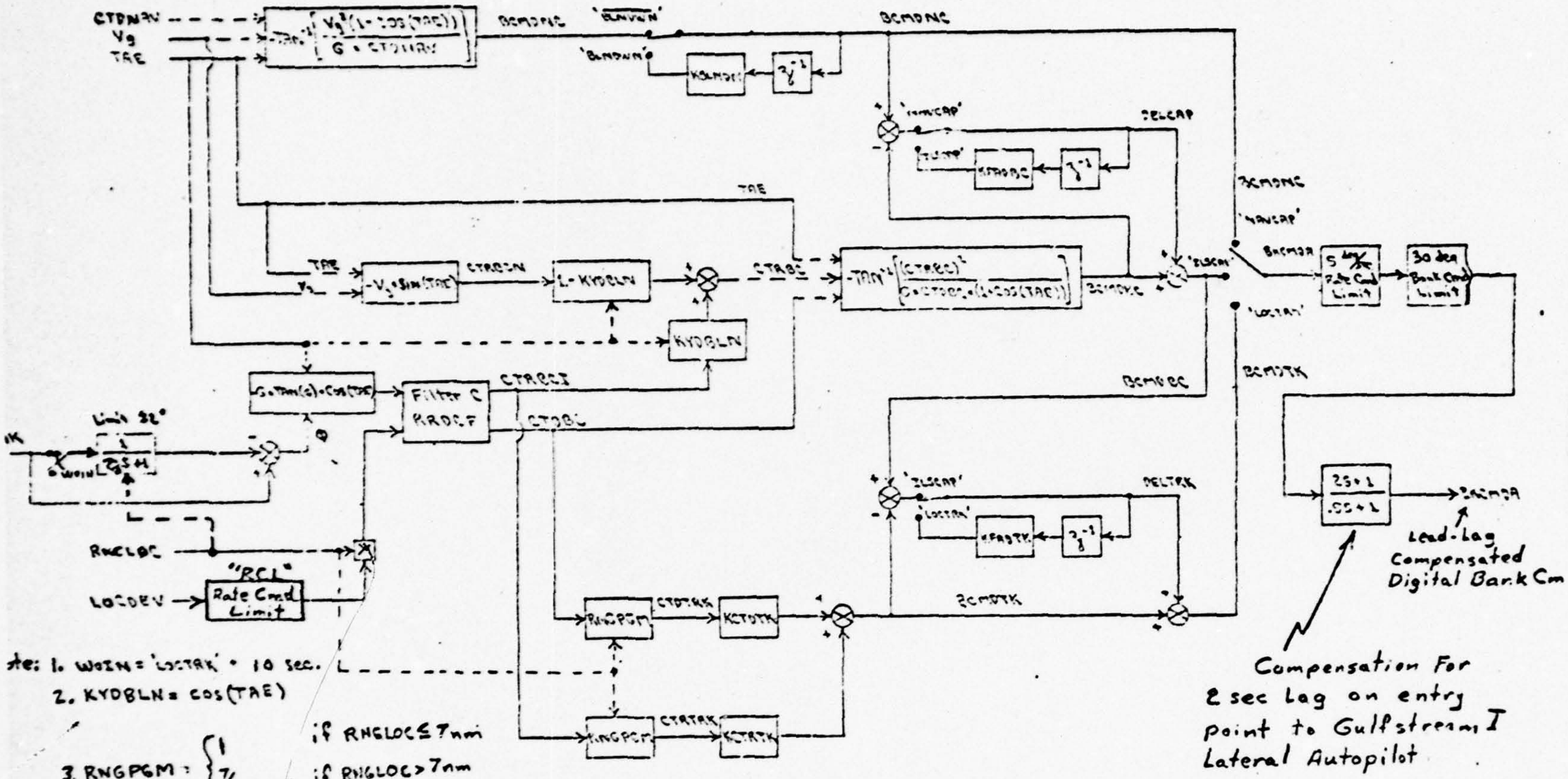
3. For $t_k > t_j$ $\quad DELCMD(t_k) = DELCMD(t_j) * (KFADE)^{(k-j)}$

   $t_k - t_j = (k-j)T$ $\quad = DELCMD(t_j) * e^{-\frac{(k-j)T}{\tau}}$

   $CMD(t_k) = CMD2(t_k) + DELCMD(t_k)$

4. For $(t_k - t_j) >> \tau$ $\quad CMD(t_k) \approx CMD2(t_k)$

Figure 5-6. Smoothing Technique For Mode Switching (Digital System)



Lead-Lag Compensated Digital Bank Cm

Compensation For 2 sec Lag on entry point to Gulfstream I Lateral Autopilot

te: 1. WOIN = 'LOCTAK' • 10 sec.
2. KYDBLN = COS(TAE)

3. RNGPGM = $\begin{cases} 1 & \text{if RNGLOC} \le 7\text{nm} \\ 7/\text{RNGLOC} & \text{if RNGLOC} > 7\text{nm} \end{cases}$

4. $\tau_\phi = 22.5 + 1.5 * \text{RNGLOC}$
5. RCL ≥ $9 \frac{\mu A}{sec}$ (See Fig 13)

Figure 5-7a 3D/4D Lateral Axis Control Laws

30/40 Gulfstream I Inner Loops
(SP20 Autopilot)

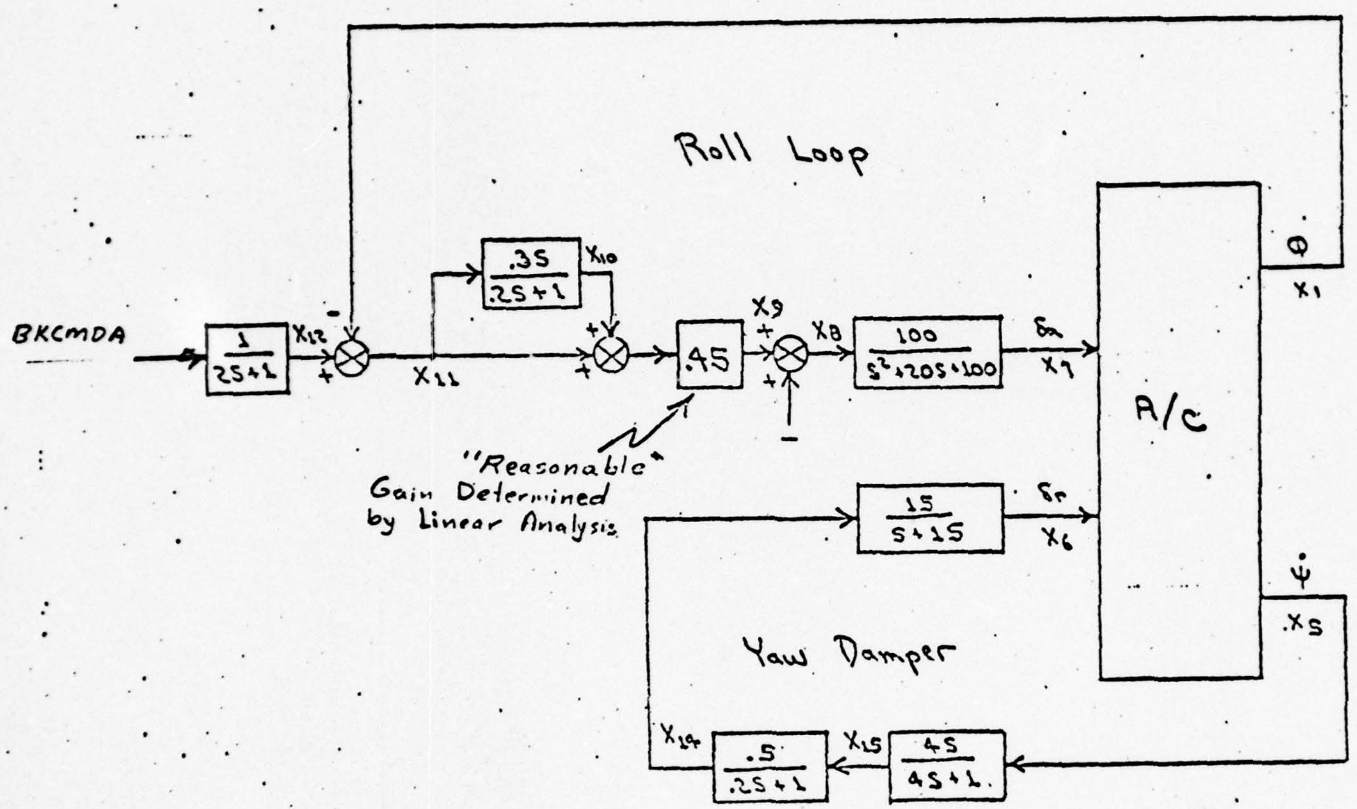
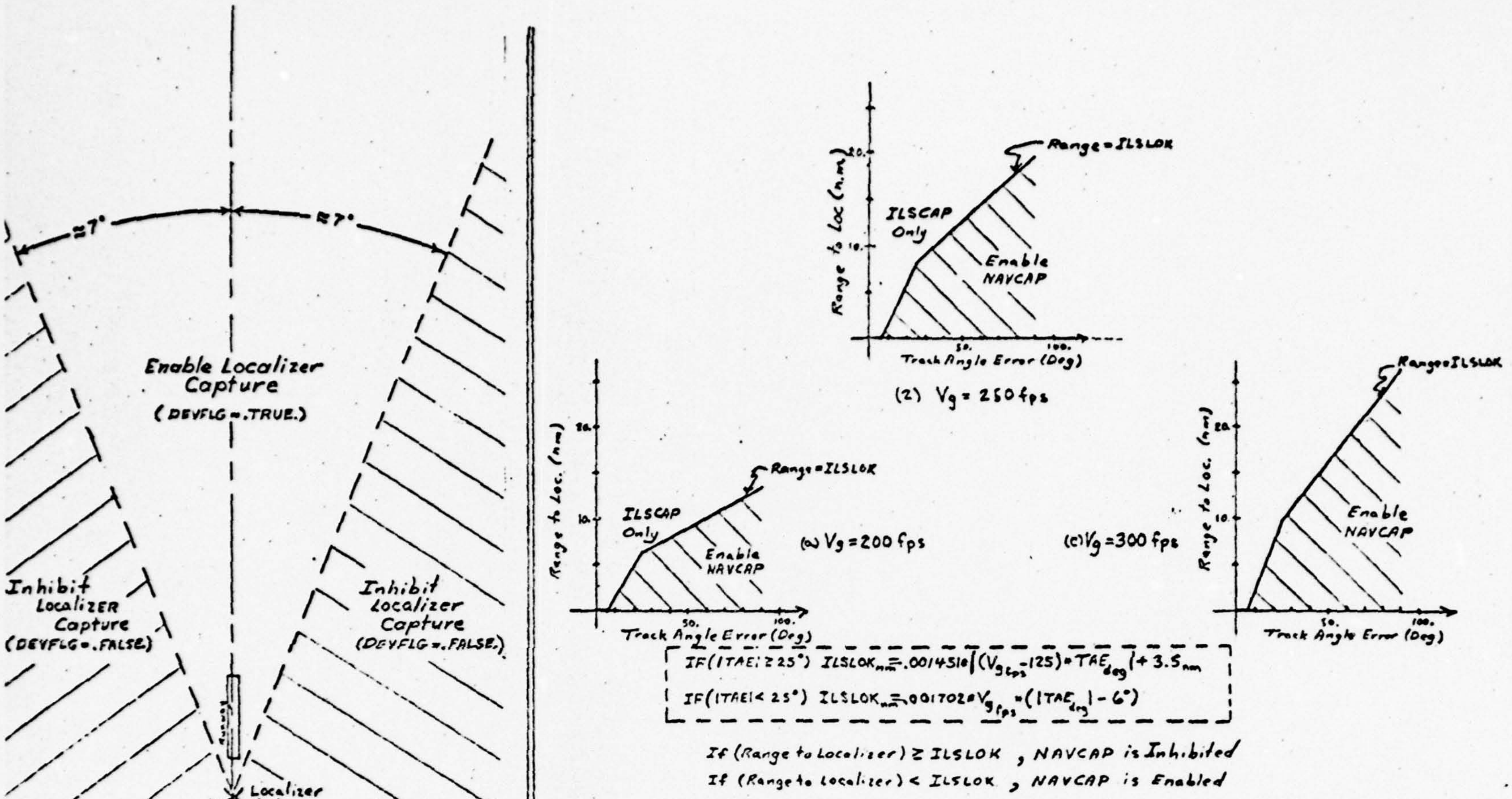


Figure 5-7b



$$\text{IF}(|TAE| \geq 25^\circ)\quad ILSLOK_{nm} = .0014516\,\left|(V_{g_{fps}} - 125) * TAE_{deg}\right| + 3.5_{nm}$$

$$\text{IF}(|TAE| < 25^\circ)\quad ILSLOK_{nm} = .0017020 * V_{g_{fps}} * (|TAE_{deg}| - 6^\circ)$$

If (Range to Localizer) ≥ ILSLOK , NAVCAP is Inhibited

If (Range to Localizer) < ILSLOK , NAVCAP is Enabled

(a) Guard Against Decreasing Localizer Deviation At Wide Angles

(b) Guard Against A-NAV Aiding At Excessive Range Due to A-NAV Cross-Track Distance Errors

Figure 5-8. Capture Trip Point Guards

TGS 5/14/75

ILSLOK basically inhibits NAVCAP if RNGLOC and TAE and VG data indicate that a non-R-NAV aided capture can be made. The only protection really is against early capture due to R-NAV CTD errors in regions where RNAV aiding is not needed. The equation for ILSLOK was generated by tabulating the ranges at which various TAE capture trip points intercept the beam edge (200 μA) at 200, 250, and 300 fps and then curve fitting to this data. It should be noted that if LOCDEV >200μA, then CAPTRP is based on R-NAV data, but if LOCDEV ≤ 200μA, it is based on beam data. *This feature is necessary to prevent* R-NAV errors from corrupting the trip point of an ILS data only capture. To protect against decreasing LOCDEV at wide angles, DEVFLG, as discussed above, is used and to prevent false trips due to momentary interference as well as to guard against sudden transients a rate limit on LOCDEV is included. This rate limit is programmed with TAE and speed but lower limited at 9μA/sec under all conditions. Figure 5-9 shows the programming on the rate limit which is basically determined by dividing the cross track rate of the aircraft by range and then increasing this value by 30% to provide a conservative estimate of deviation rate.

Fade From NAVCAP to ILSCAP

Notice that *during NAVCAP a fader circuit* like the one of Figure 5-6 is prepared to fade from NAVCAP to ILSCAP when the need arises. Similarly during ILSCAP a companion fader stands ready to blend to the track mode. The bank command in either NAVCAP or ILSCAP is the proper one to generate a circular capture and both are essentially the same as Equation 5.1 although the form used in ILSCAP is modified to more appropriately use beam based data. This form is easily derived from Equation 5.1 which is repeated below

$$\text{BNK.CMD} = -\tan^{-1}\left(\frac{Vg^2 * (1 - \text{COS(TAE)})}{g * \text{CTD}}\right) \qquad (5.18)$$

Multiplying numerator and denominator of the arctan argument by (1 + COS TAE) and substituting $\text{SIN}^2\text{(TAE)}$ for $1\text{-COS}^2\text{(TAE)}$ in the numerator yields

$$\text{BNK.CMD} = -\tan^{-1} \frac{(Vg * \text{Sin(TAE)})^2}{g * \text{CTD}(1 + \text{COS(TAE)})}$$

But

$Vg * \text{SIN(TAE)} = \text{Cross Track Rate (CTR)}$

Hence

$$\text{BNK.CMD} = -\tan^{-1} \frac{(\text{CTR})^2}{g * (\text{CTD}) * (1 + \text{COS(TAE)})} \qquad (5.19)$$

Equation (5.19) uses CTR as well as CTD explicitly and can thus make effective use of the beam data filter output. Groundspeed estimate errors will not influence Equation (5.18) directly and TAE errors will not have a large effect, particularly as the system approaches transition to track since as TAE → 0, COS(TAE)→1. and (1 + COS(TAE))→ 2. Thus, this term does not have the effect that it does in Equation (5.13), where

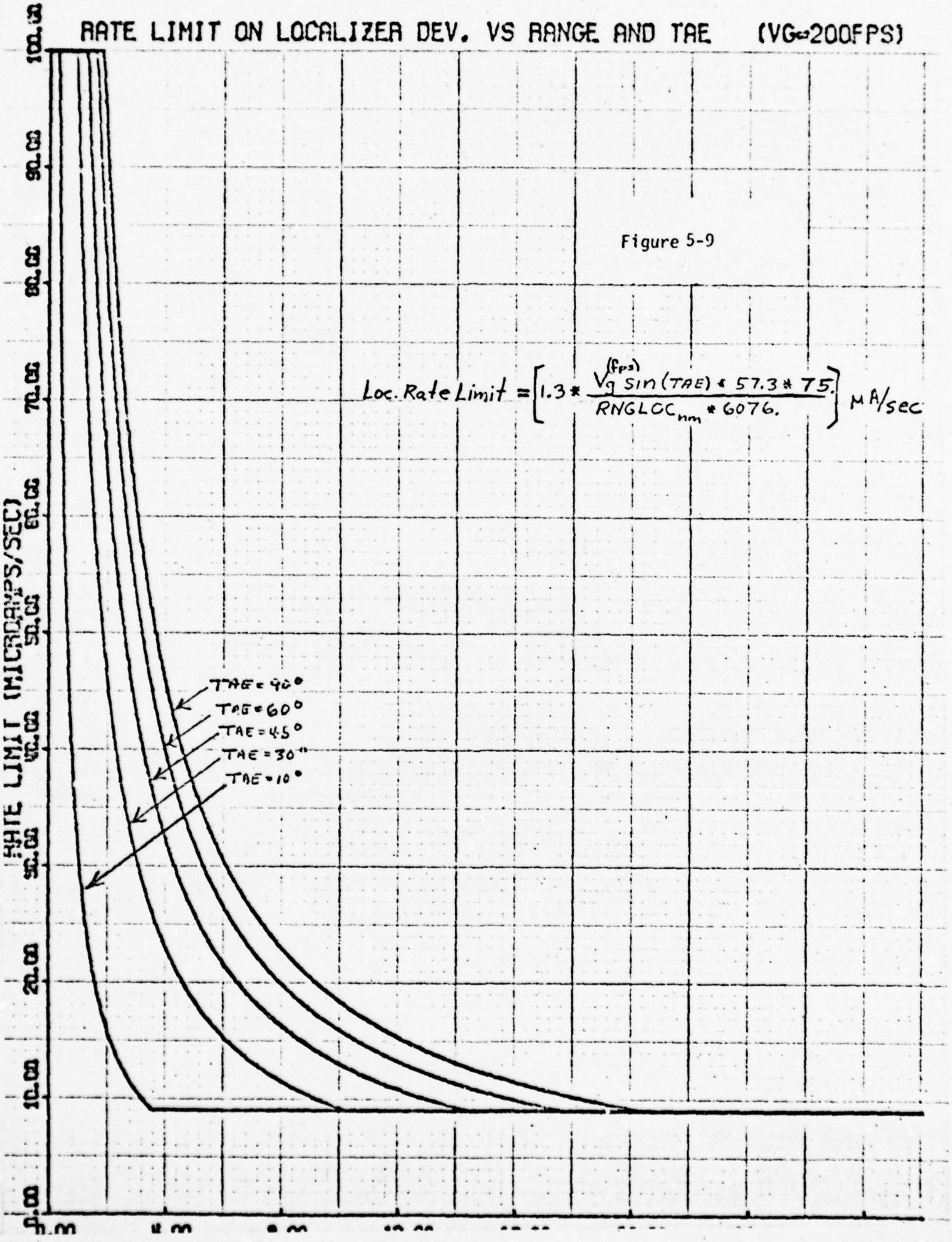

RATE LIMIT ON LOCALIZER DEV. VS RANGE AND TAE (VG=200FPS)

Figure 5-9

$$\text{Loc. Rate Limit} = \left[1.3 * \frac{V_g^{(fps)} \sin(TAE) * 57.3 * 75.}{RNGLOC_{nm} * 6076.}\right] \text{MA/sec}$$

$(1-\mathrm{COS}(TAE)) \rightarrow 0$ as $TAE \rightarrow 0$. The form of Equation 5.19 is no better than that of 5.18 in NAVCAP, however, since in NAVCAP, CTR = -Vg*Sin (TAE) which simply would bring back TAE in a different form.

Track Mode

The track mode is a conventional proportional combination of position and rate relative to the beam center to form bank command. The trip point for going to track mode was discussed in Section 5.3.4. Track gains are given in Table 5-2 in several different forms for ease of correlation with other work. A stability analysis of the track mode appears in Section 5.4.

	Range $<$ 7 nm	Range $\geq$ 7 nm	Range = 8 nm
Gain on Position	$\left(\frac{3.2}{70} = .046\right)\frac{deg}{ft}$ or $(\ 279.5\ )\ \frac{deg}{nm}$	$(.45)\ \frac{deg}{\mu A}$ or $(13.6)\frac{deg}{dot}$	$(.45)\ \frac{deg}{\mu A}$ or $(13.6)\frac{deg}{dot}$
Gain on Rate	$\left(\frac{35}{70} = .5\right)\frac{deg}{fps}$ or $(.844)\frac{deg}{kt}$	$\left(.5 * \frac{7}{RNGLOC}\right)\ \frac{deg}{fps}$ or $\left(.844 * \frac{7}{RNGLOC}\right)\frac{deg}{kt}$	$(.438\ \frac{deg}{fps}$ or $(.739\ \frac{deg}{kt}$

Table 5-2. Rate & Position Gains for Track Mode

Notice that the output of the data processing filter is position and position rate to provide the requisite data to the capture computations. This is also the appropriate form for the track mode at close ranges. However, at longer ranges the effects of beam noise, particularly excessive bank activity, require that gains be softened, usually such that the track law becomes angularly based rather than position based. The RNGPGM block of Figure 5-7 may be seen to provide this feature since at ranges beyond 7 nm (neglecting the filter for the moment since its parameters are not range dependent),

$$CTDTRK = \frac{7}{RNGLOC} * RNGLOC * LOCDEV$$

$$CTDTRK = 7 * LOCDEV \qquad (5.20)$$

Equation 5.20 shows that CTDTRK beyond 7 nm is really in angular units with the 7 being the required factor for blending properly with the linearized portion. Alternately, the combination of RNGPGM programming downstream of the filter and RNGLOC programming of LOCDEV upstream of the filter provides linearization of LOCDEV out to 7 nm and a constant gain beyond that range. The rate is treated similarly to take care of the radio contribution to rate. There is no real necessity to program the bank contribution to rate and in fact stability would be aided by not doing so. However, in order to maintain system simplicity, no separate provision was made for bank contributed rate.

Data Processing

The data processing portion of the block diagram consists primarily of a double complementing filter (DCF) as described in Reference 3. Since in this application roll or bank is used for generating acceleration, it is termed the roll radio double complementing filter (RRDCF). The proportional combination of RRDCF rate and R-NAV based rate using the function KYDBLN provides slightly improved rate for capture. The basic idea here is that although bank provides information about acceleration orthogonal to the aircraft fuselage reference line, the RRDCF filter requires acceleration relative to the beam centerline. Hence bank is processed as

$$\ddot{y}_{\phi} = -g \tan(\phi) \cos(TAE)$$

At high course cuts, therefore, bank does not provide very good information about acceleration relative to the beam. The necessity of complementing RRDCF filter rate against R-NAV rate in capture is not firmly established. R-NAV rate is used to initialize RRDCF and low frequency updates by radio rate might then provide sufficiently good rate. The function used for KYDBLN is cos(TAE) so that as the system turns onto the final course, the influence of R-NAV rate is removed. A washout on bank is used during track to prevent beam tracking standoff due to bank bias.

Output Command Processing

The final aspect of the system block diagram is the output command processing shown at the right side of Figure 5-7a. Basically a 5 degree per second rate command limit and a 30 degree command limit are imposed. The lead-lag compensation is necessary for the specific Gulfstream I autopilot interface provided for in this application. The system interface to the Sperry SP40 autopilot interposes a 2 second lag between the available heading error port and the bank command summing point. Figure 11b shows the equivalent block diagram of the SP40 autopilot. It should be noted that this is a somewhat sluggish autopilot having particularly large servo lags. This factor necessitates running somewhat lower outer loop gains than one might prefer in approach.

5.3.8 Software Organization

The basic software organization required to implement the system described above is shown in Figure 5-10. The localizer capture and track computation are a submode to the ILS main program which interfaces with the R-NAV main software. The chosen approach is to process the data in one software block, determine the proper mode in another, and then perform control computations based on the determined mode. Finally, the output control processing does the rate and amplitude command limiting and the lead-lag compensation as shown in Figure 5-7a. A more detailed block diagram is provided in Figure 5-11. The program listing is given in Appendix F.

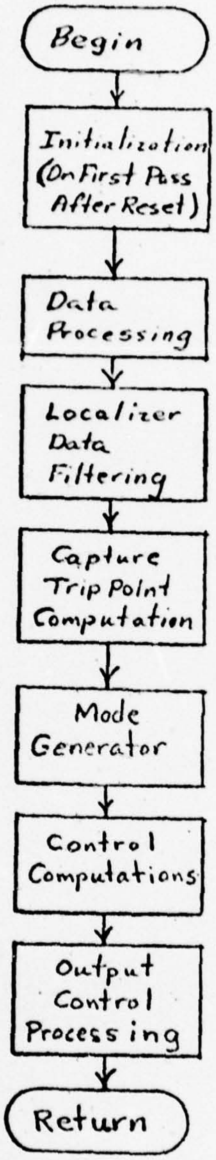
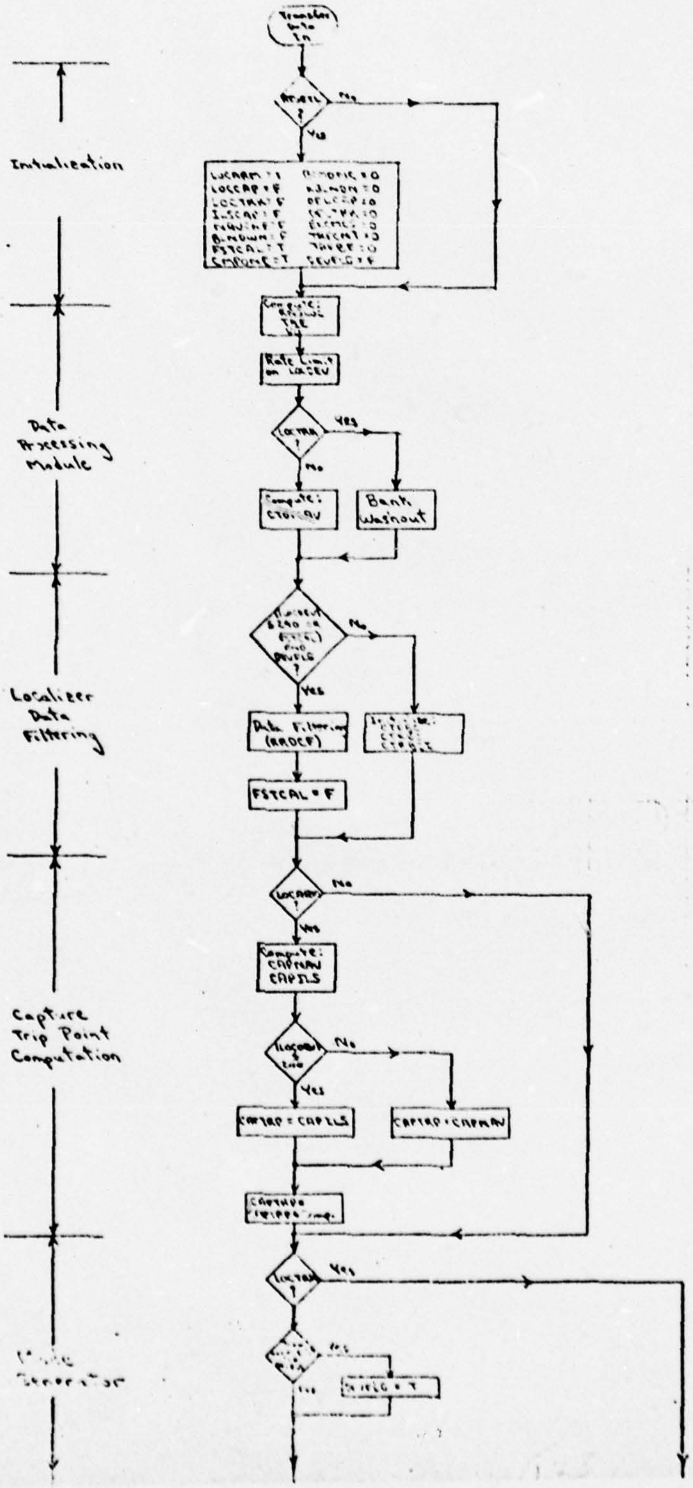


Figure 5-10. Basic Software Organization
Lateral Computational Module

Figure 5-11. Detailed Lateral Flow Chart
Section 1 out of 3



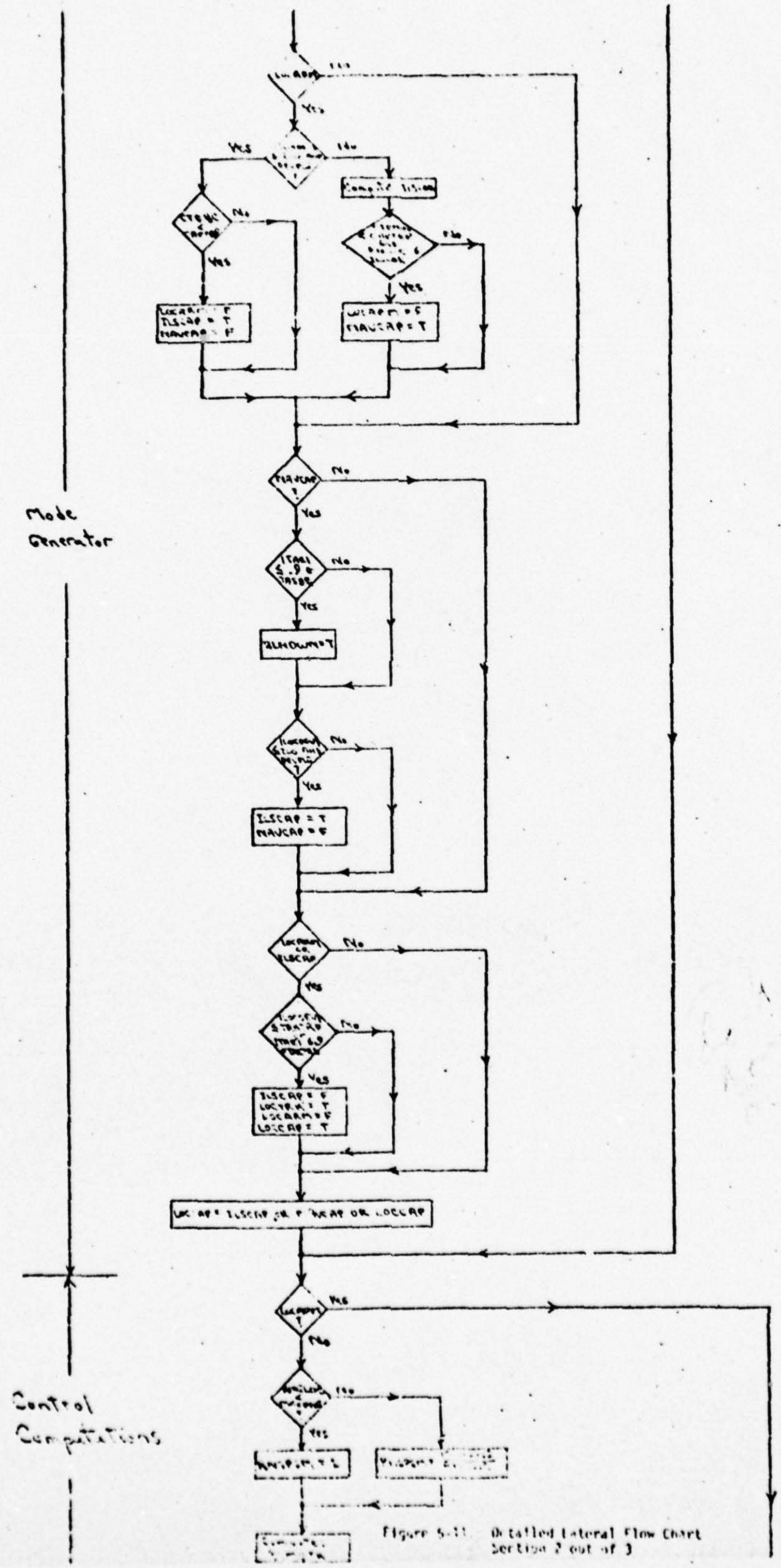


Figure 5-11. Detailed Lateral Flow Chart
Section 2 out of 3

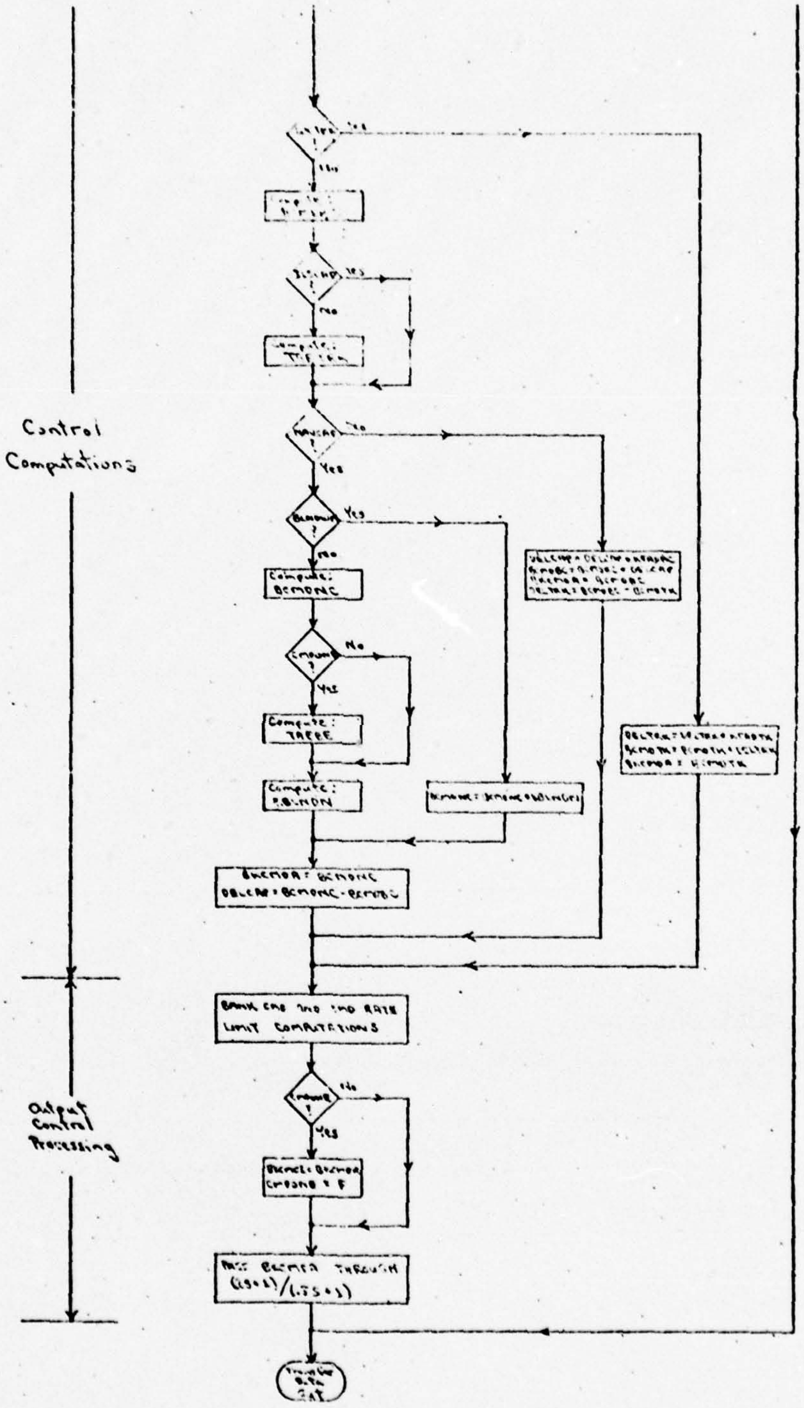


Figure 5-11. Detailed Lateral Flow Chart
Section 3 out of 3

3D/4D Localizer Module

Flow Diagram

5.4 Stability Analysis of the 3D/4D Lateral ILS Localizer Track Laws

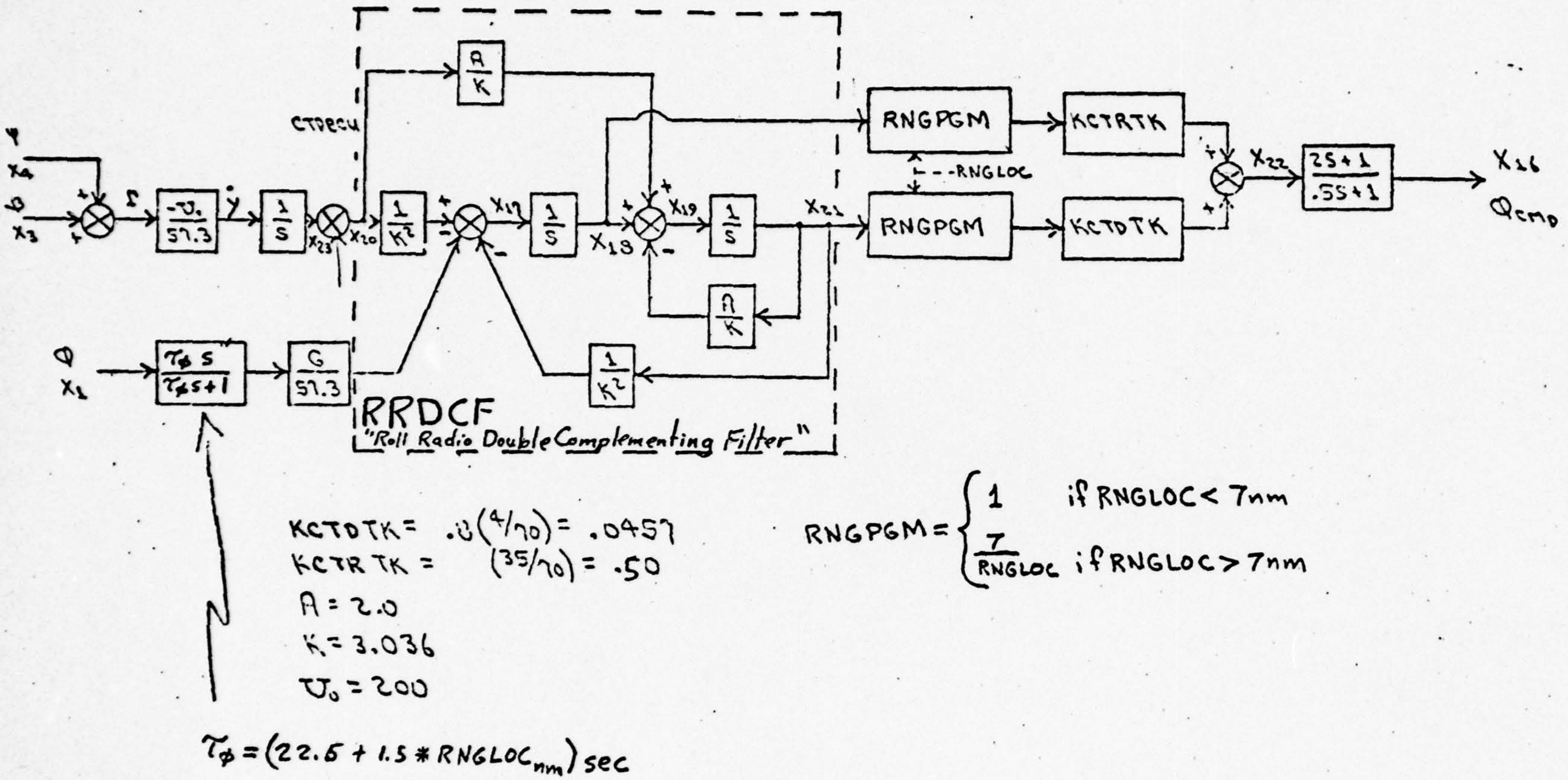
Figure 5-12 shows a block diagram of the track control laws from sensor inputs to bank command while Figure 5-13 shows the block diagram for autopilot and aircraft. (Note the rather sluggish aileron servo.) From Reference 3 the pole-zero parameter A in the RRDCF filter (see Figure 5-12) was set at 2 with the bandwidth parameter K fading from 3.036 (the capture value) to 10 once track was well established. Stability and PSD data was therefore computed for both extreme values of K recognizing that the results for K=3.036 will be applicable early in track while those for K=10.0 apply for well established track. Loops were broken at radio and at bank command. The loop at radio plots must be interpreted carefully particularly for K=10, because the complementing effect of the bank feed through to position (see Reference 3) permits somewhat lower than conventional bandwidths in the radio loop.

The system was linearized (RNGPGM) only to 7 nm from the localizer antenna since linearizing further produced excessive bank activity.

At long ranges, therefore, the bandwidth of the system both at radio and bank command decreases considerably ($-20 \log \left(\frac{7\text{nm}}{25\text{nm}}\right) = -11.06$ db). Stability was checked carefully, therefore, at the assumed extreme range of 25 nm. Figures 5-14 to 5-17 show gain/phase plots for loops broken at radio and bank command for K=3.036 and K=10 respectively, all at 25 nm. Note on Figures 5.14 and 5-15 the phase dip below 180° at low frequency. This dip is due to the bank washout shown in Figure 5-12. The magnitude of the dip is greater for shorter than for longer washout time constants. For this reason a 60 second washout is used at 25 nm programmed down to 30 seconds at 50 nm. The long washout at near range is undesirable due to the long charging time required to remove bank biases which are more troublesome at near than at far ranges. Bank biases, if not washed out, result in both position and rate biases out of RRDCF (see Reference 3). Table 5-3 tabulates the stability margin and crossover data from Figures 5-14 through 5-17. Also shown in Table 5-3 is the analogous data for 5 nm range.

Range	W/O Time Const.	K	RNGPGM* KCTDTK deg/ft	RNGPGM* KCTRTK deg/fps	Loop at Radio			Loop at Bank Cmd		
					Wc.o.	P.M.	HF G.M.	Wc.o.	P.M.	G.M.
25 nm	60	3.036	$\frac{7}{25}\left(\frac{3.2}{70}\right)$	$\frac{7}{25}\left(\frac{35}{70}\right)$	.092	25°	11	.101	38°	24
25 nm	60	10.0	$\frac{7}{25}\left(\frac{3.2}{70}\right)$	$\frac{7}{25}\left(\frac{35}{70}\right)$	.070	20°	7	.092	42°	24
≤ 7 nm	30	3.036	$\left(\frac{3.2}{70}\right)$	$\left(\frac{35}{70}\right)$	.155	31°	8	.26	57°	13
≤ 7 nm	30	10.0	$\left(\frac{3.2}{70}\right)$	$\left(\frac{35}{70}\right)$	.09	24°	10	.295	60°	13

Table 5-3. Stability Margins



$KCTDTK = .8(4/70) = .0457$

$KCTRTK = (35/70) = .50$

$A = 2.0$

$K = 3.036$

$V_0 = 200$

$$RNGPGM = \begin{cases} 1 & \text{if } RNGLOC < 7nm \\ \frac{7}{RNGLOC} & \text{if } RNGLOC > 7nm \end{cases}$$

$\tau_\phi = (22.5 + 1.5 * RNGLOC_{nm})$ sec

Figure 5-12. 3D.4D Track Laws

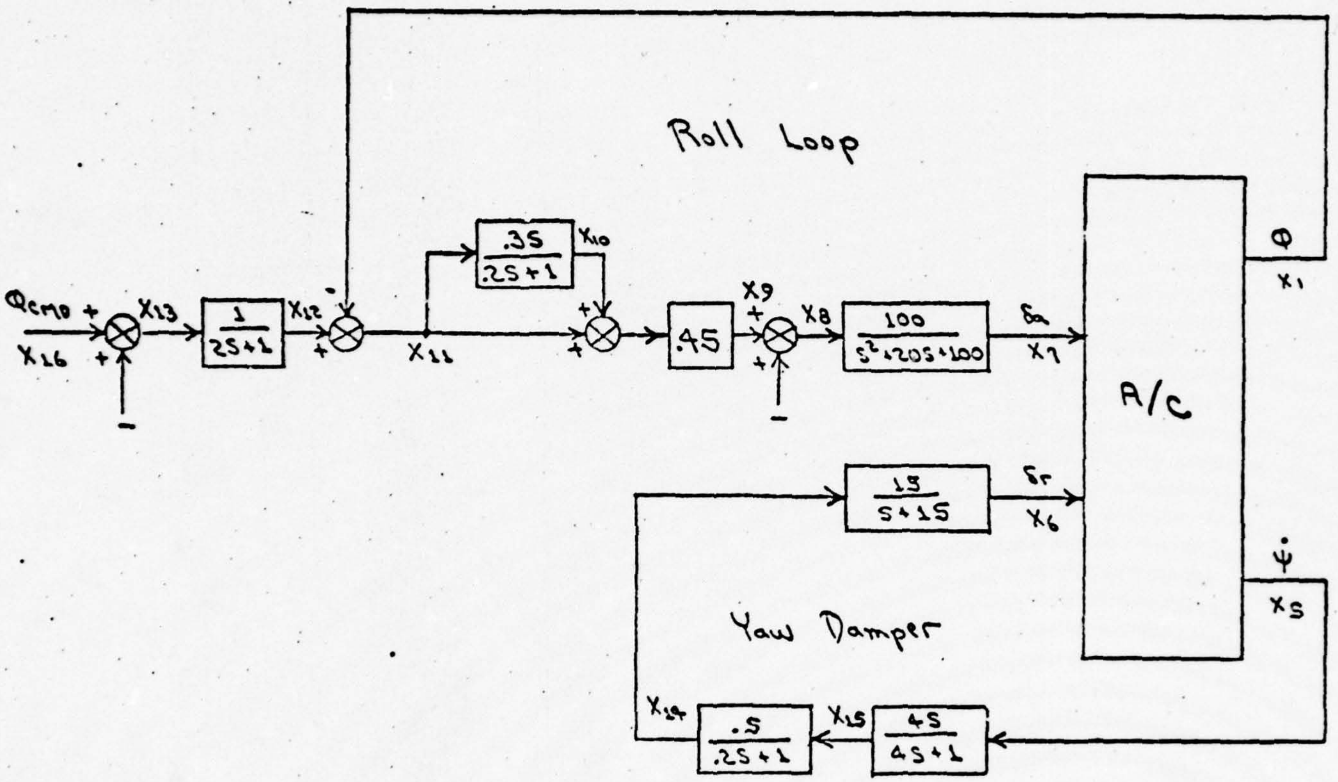
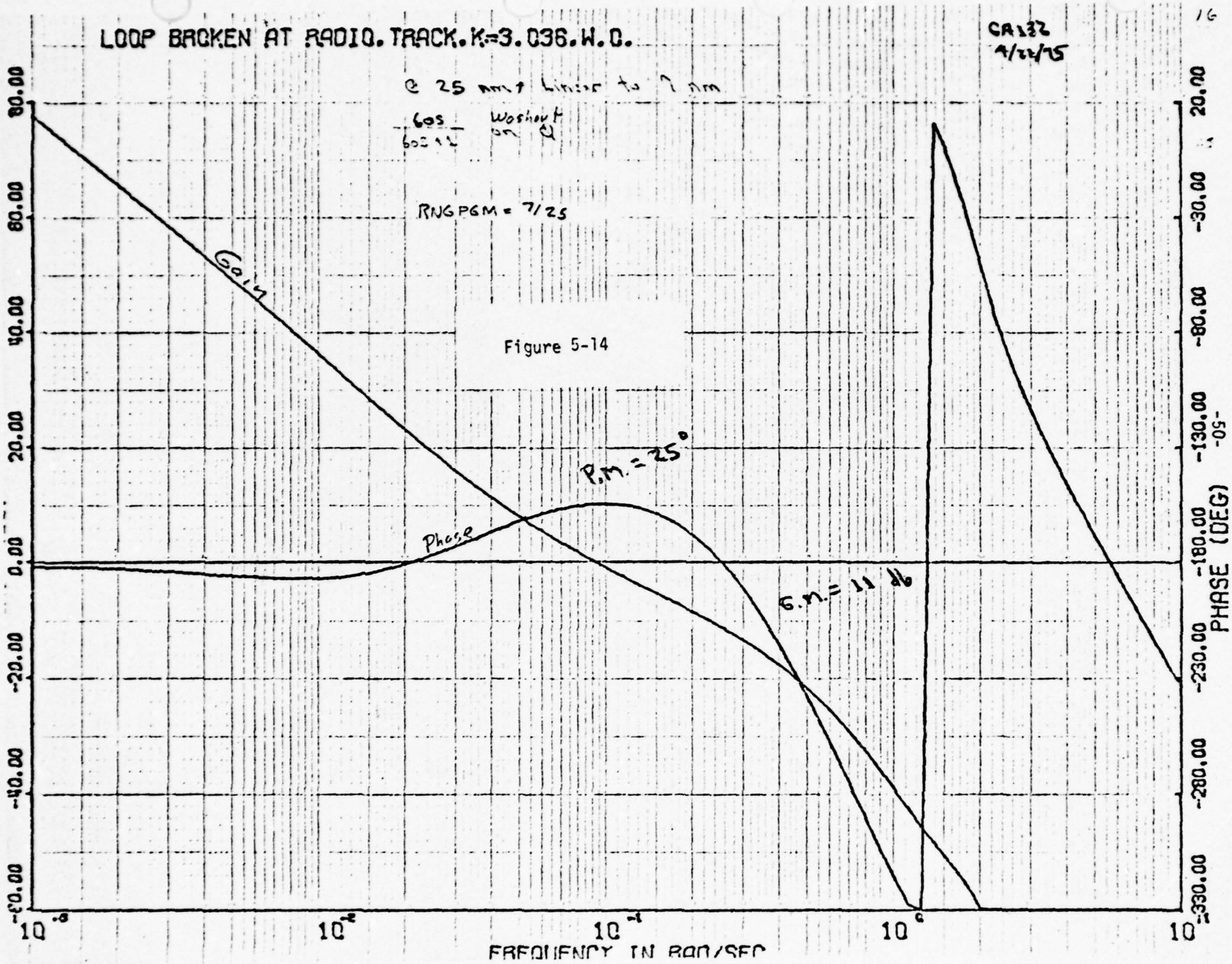


Figure 5-13. Gulfstream I Inner Loops

16

CA132
4/22/75


LOOP BROKEN AT RADIO. TRACK. K=3.036. W.O.

@ 25 nmi Linear to 7 nm

$\frac{60s}{60s+1}$ Washout on Q

RNGPGM = 7/25

Figure 5-14

(15)

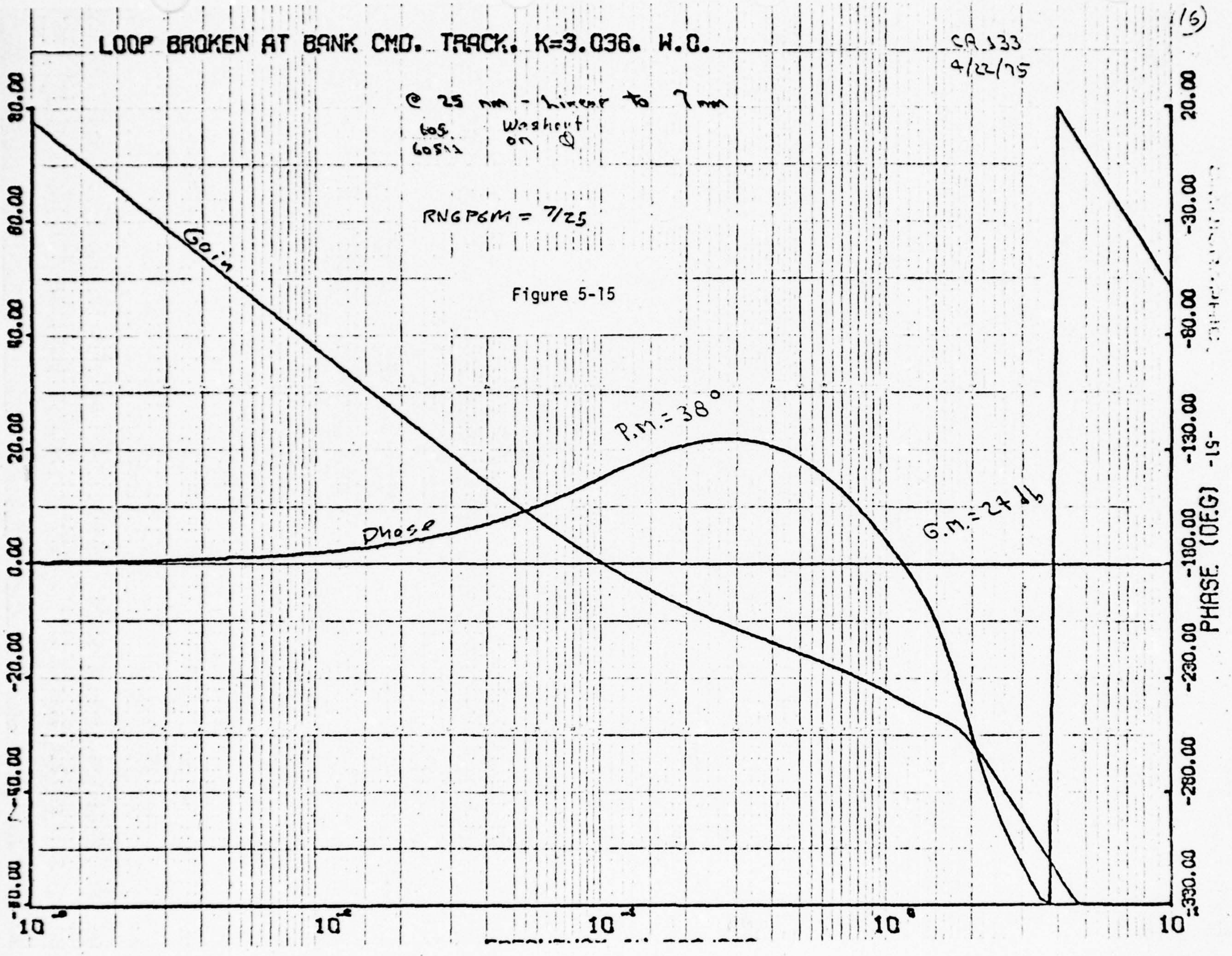


Figure 5-15

(17)

LOOP BROKEN AT RADIO. TRACK. K=10. . W.O.

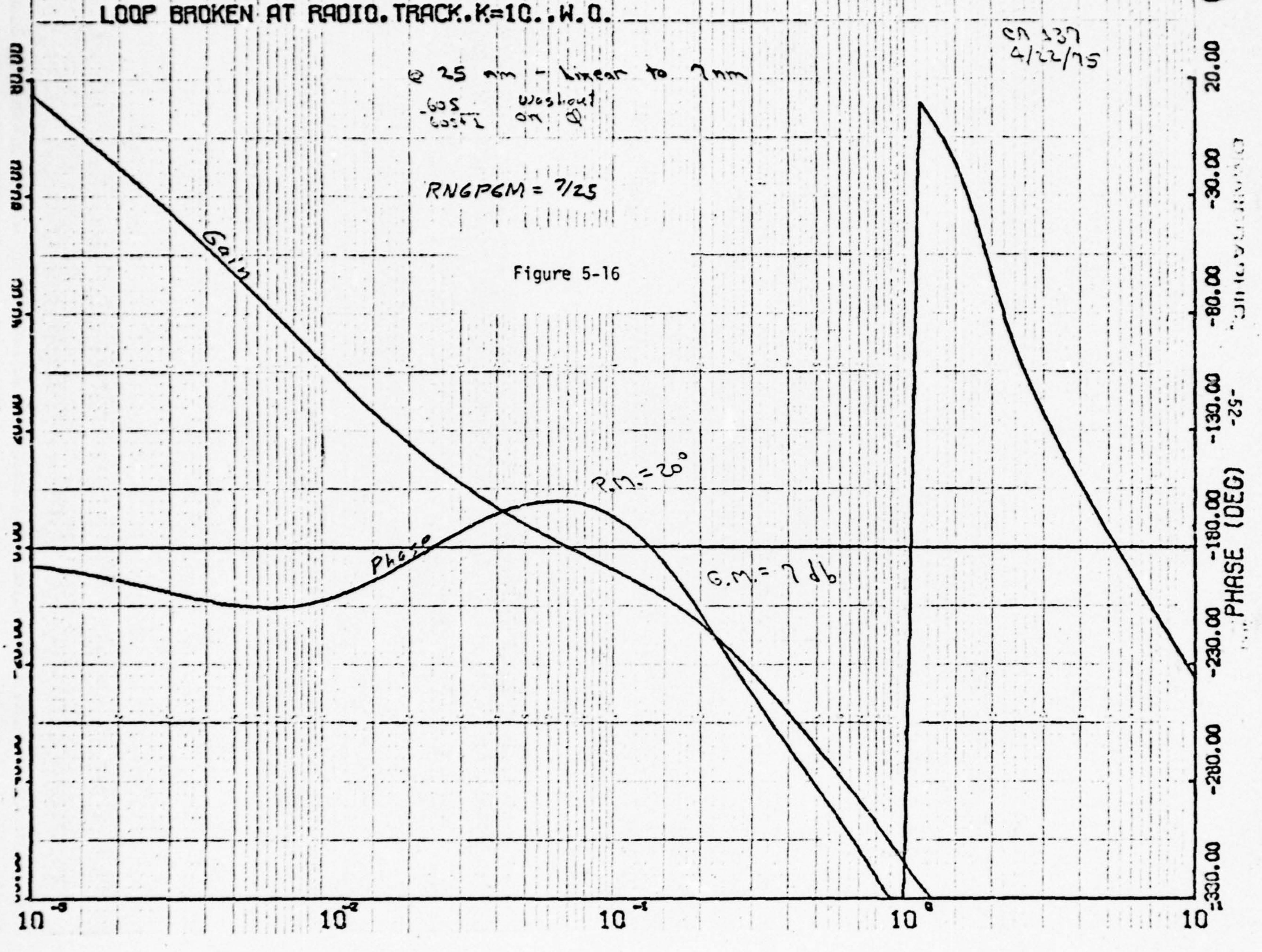
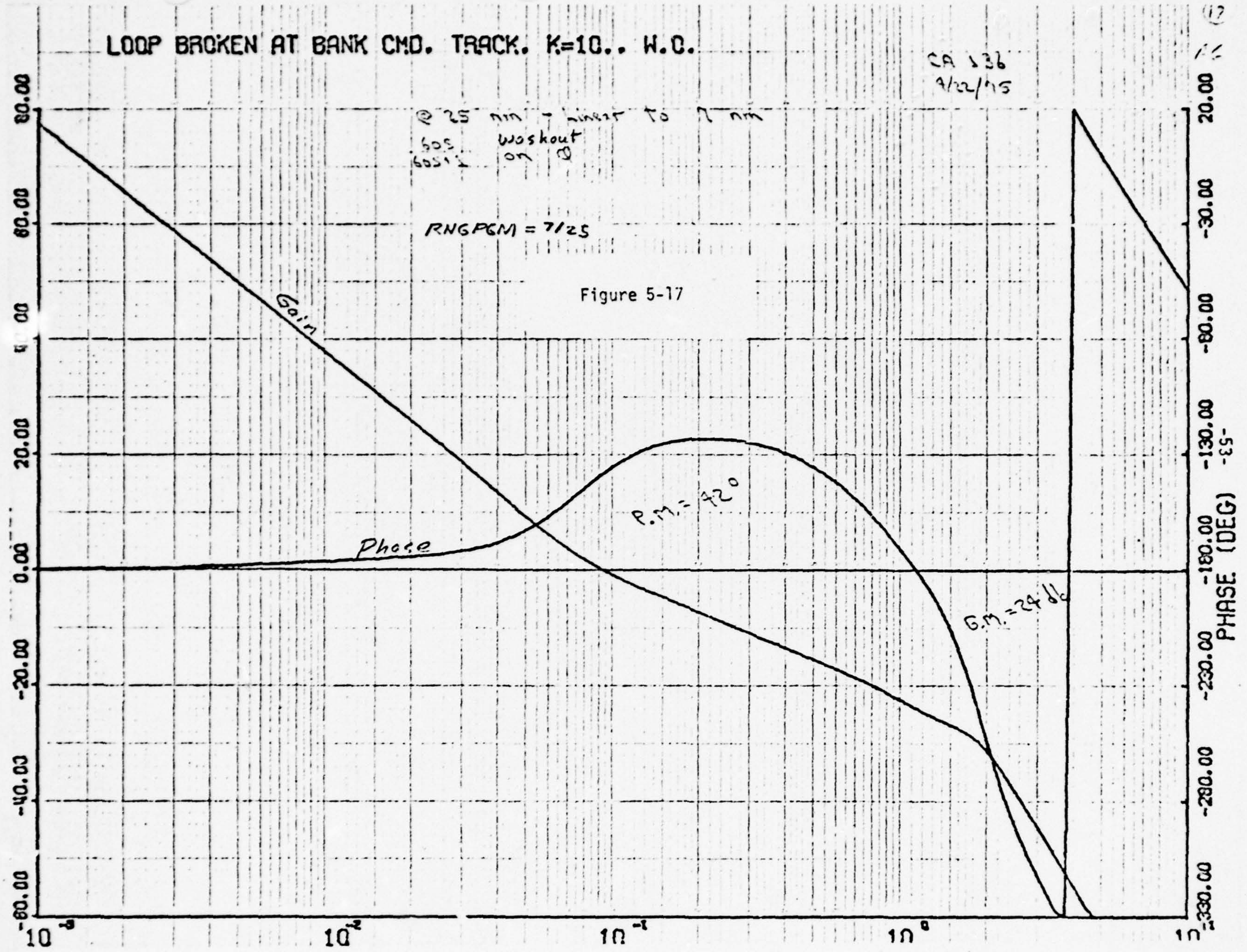


Figure 5-16



LOOP BROKEN AT BANK CMD. TRACK. K=10.. W.O.

CA 136
9/22/75

@ 25 nm - linear to 7 nm
60s washout
60s'2 on Ø

RNGPGM = 7/25

Figure 5-17

Section 6
LONGITUDINAL ILS CONTROL LAW DESIGN AND ANALYSIS

6.1 INTRODUCTION

The glideslope portion of the ILS approach control laws are presented by this report. The control laws provide for a smooth, transientless capture of the glideslope beam. The control laws will automatically adapt for varying airspeed, wind and beam angles. In addition, both above and below the beam intercepts can be accommodated.

The control laws developed rely upon derived vertical rate to enhance the adaptive capture capabilities and to supplement basic autopilot and flight director path dampening and gust alleviation. The autopilot and flight director both accept pitch commands from the glideslope control laws in the 3D/4D processor to an altitude hold port. Once attitude is closed in a glideslope control loop, it has been found that vertical rate is of some benefit for gust alleviation, but it cannot be used to its fullest advantage. The control laws, therefore, have some advantages over conventional pitch implemented controls while exhibiting similar tracking performance.

A description of the glideslope computations, stability and performance analyses are contained herein. The glideslope control laws digital program listing is contained in Appendix G.

6.2 GLIDESLOPE OPERATION AND DESCRIPTION

Glideslope control laws are provided for vertical navigation of an RNAV/ ILS approach capability. An adaptive capture capability has been designed into the glideslope control laws. The control laws are illustrated in Figure 6-1.

Glideslope operation is performed with the 3D/4D system just as is done with any conventional flight guidance computer. Basically glideslope capture is initiated from a flight path which has been set up to intercept the glideslope beam. Most often the glideslope is intercepted from a zero

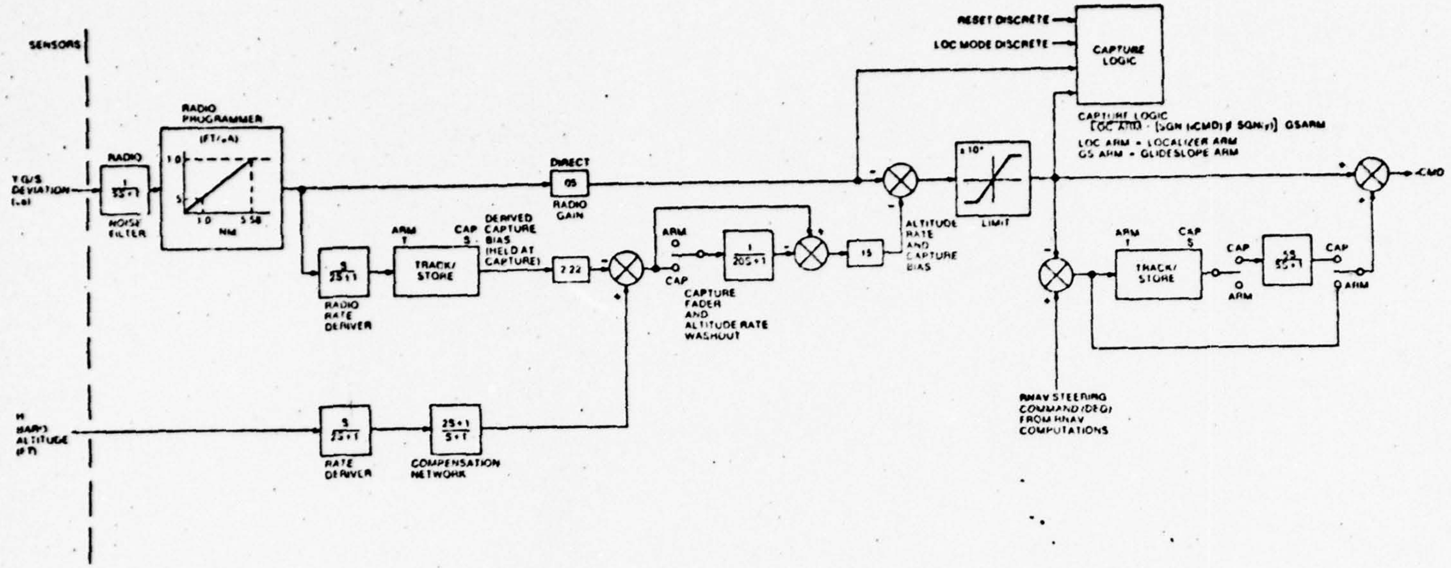


Figure 6-1 Glideslope Capture and track Control Laws

flight path angle or altitude hold maneuver. Unlike some glideslope operations the control laws in the 3D/4D system will also allow the user to capture from above.

Operationally, glideslope capture and tracking can be accomplished in the approach auto mode (APPR AUTO) of the ILS portion of the 3D/4D system. Once an intercept has been established so that the aircraft will intercept the glideslope beam, capture is completely automatic. The system compares beam and vertical rates with beam deviation and polarity to establish a capture trip point. Figure 6-1 also includes the capture logic which dictates when capture occurs. When pitch command is sensed to go through zero in the correct direction as determined by comparing its polarity with that of beam deviation, capture is initiated.

Since the capture is a function of both beam closure rate and vertical rate, capture is a function of ground speed and flight path angle. Ground speed in turn makes the capture a function of aircraft speed and wind. If a capture is initiated from altitude hold, the capture trip point will vary as a function of how fast the glideslope beam is approached. The trip point will move further from the glideslope beam center as the closure rate increases no matter what reason. Similarly, if the aircraft has a vertical rate established, capture will be adjusted to compensate for the vertical rate. As an example, when capturing from above, capture will be initiated further from the glideslope for a greater vertical rate or flight path intercept. Of course, beam rate alone can also sense this higher closure rate, but the addition of vertical rate has proved to be more accurate in adjusting the trip poin especially in above the beam captures.

Once capture has been initiated, the beam rate is held by a track/store operation. Also notice that at the instant of capture a fader or washout is initiated on both stored beam rate and vertical rate. The action is such that at the instant of capture, pitch command is zero, and the aircraft proceeds toward the glideslope beam. Deviation therefore decreases while the rate signals (which are being slowly faded out) provide a nose up or nose down pitch command, whichever is appropriate to transition the aircraft to the glideslope path. The fader washes out all of the beam rate and initial vertical rate as the aircraft acquires the beam centerline. In this manner the control laws smoothly and transientlessly transition the aircraft from its intercept flight path to the beam center.

After capture, tracking is provided by glideslope deviation and washed out vertical rate. The capture bias provided by the stored beam rate is completely removed. Vertical rate is washed out to prevent the vertical rate or descent rate from causing a deviation standoff in the control laws.

The control laws developed for the 3D/4D use vertical rate to good advantage as was explained in the introduction. It should also be pointed out that for autopilot operation, vertical rate is required to dampen the deviation path. Pitch attitude used for deviation path dampening in the autopilot is washed out (by the 5 sec forward integration) faster than can normally be done to provide stability with stiff glideslope tracking.

A breakdown of the glideslope operation and details is given below:

6.2.1 Overview of Glideslope Operation

The glideslope computation may be entered via a transition from the LOC or the RNAV mode. The RNAV computation will request the ILS main program to perform this transition. If all system validities are acceptable, the ILS main program will activate the localizer and glideslope computations. Each time a transition is requested, the ILS main program will initialize the glideslope computation by setting the logical variable RESETG true for one program cycle.

As long as the localizer computation is in the LOCARM condition, the RNAV steering commands are used to control both the lateral and longitudinal axes. The transition to LOCCAP terminates RNAV control of the lateral steering commands only. ILS control of longitudinal steering command will occur when the glideslope capture mode is entered (GSCAP) subsequent to LOCCAP. However, deactivation of the ILS computation, after LOCCAP, can only be achieved by disengaging the autopilot and reselecting the RNAV mode. Re-entry of the ILS computation is possible and will be treated as an initial request by the ILS main program.

6.2.2 Glideslope Arm (GSARM) Logic

The glideslope control law will be set to the GSARM mode whenever the logic signal RESETG from the ILS main program is true. The computation will remain in this mode as long as the localizer computation is in the arm condition (LOCARM). While in this mode, the computation will estimate the distance to and the approach rate of the glideslope beam. Following LOCCAP that information will be used to select an appropriate point to transition into the capture mode (GSCAP). The transition will occur at the first opportunity <u>following</u> the termination of LOCARM and prior to the intercept of the glideslope.

6.2.3 Glideslope Arm Computation

While in the arm mode, external steering commands are passed directly through the computation without modification. The radio deviation is passed through a gain programmer to estimate linear distance to the glideslope. A rate deriver estimates the approach rate of the glideslope and these two signals are combined with derived barometric altitude rate to generate a pitch command. That command signal is routed to the capture logic rather than the steering command.

6.2.4 Glideslope Capture Logic

Assuming the localizer computation is not in the arm mode and the aircraft is on a course which will at some future time intersect the glideslope, radio deviation will establish the major portion of the pitch command and the rate signal will contribute a portion in opposition to the radio signal. As the glideslope is approached the closure rate will cancel and then exceed the radio signal. When this occurs the pitch command will initially be zero and then of a polarity which directs a flight path away from the glideslope beam. The capture detector monitors the localizer computation status, the polarity of radio deviation and that of the pitch command. When the condition described above exists, the glideslope computation is switched to the capture mode. Once this mode is enabled the computations are latched in that mode. The GSARM mode can only be reinitiated by the RESETG command generated in the ILS main program. While in the GSCAP mode the VOR/DME/baro altitude based RNAV steering command will be overwritten by the computed pitch command.

Derived barometric rate is used in the capture logic to sense glideslope intercepts from other than a zero flight path. This enables captures to be made from all practical intercept angles, including above the beam captures.

6.2.5 Glideslope Capture and Track Computations

Figure 6-1 also identifies the capture computations. There are two major functions performed by this control law: (1) generate a pitch command which will provide good capture and tracking of the glideslope beam and (2) smoothly transition from external (RNAV) control of the steering command. The first task is accomplished by mixing radio deviation with barometric altitude rate to obtain a pitch command. The radio deviation is gain programmed, and altitude rate is washed out to remove its steady state value. Derived radio rate is held to the value it has when the capture mode is enabled, and since this signal enters pitch command via the same path as altitude rate, it will be smoothly removed by the same washout used to eliminate the steady state value of altitude rate. Notice that the washout is utilized only after capture. The value of the external steering command, at the time of transition to GSCAP, is added to the computed pitch command via a washout. Since pitch command is effectively zero at that time, overwriting the steering command will not cause an unacceptable transient while the previously coupled command is faded out.

6.3. Control Law Analysis

Control law analysis was performed on the system shown in Figure 6-1. Included in the figure is a model of the autopilot. The autopilot model was derived from time response data taken on the NAFEC Gulfstream I aircraft. The gains and time constants were selected to match the following data parameters for a step command:

a) Time to peak: .42 sec.
b) Settling time: .92 sec.
c) Overshoot: 30%
d) Forward Gain: 4 deg δe/deg θ
e) Integration Rate: 1.5° δe/sec.

The only modification was a 4:1 reduction in forward gain. The probable cause of this change is an invalid elevator sensitivity in the aerodynamic data available at Collins.

Figure 6-2 illustrates the manual throttle control loop. The model has shown good correlation with manual pilot performance.

6.3.1 Radio Gain Programmer

The gain programmer converts radio deviation (214 μa/deg) to linear feet of cross track distance (.496 ft/μa-nm) out to 5.58 nm from the end of the runway. Beyond this point cross track distance is fixed at 2.8 ft/μa. To extend the range of the programmer beyond this point would invariably result in unacceptable performance due to beam noise. Assuming a shallow 2.5 degree glideslope, the programmer will be effective to an altitude of 1,480 ft. Consequently, normal captures will be performed within the linear range of the programmer. The programmer is a function of distance to touchdown rather than altitude.

6.3.2 Capture Detection

Although it is certainly possible to initiate a capture outside the linear range of the gain programmer, the effect of the non-linearity on the capture detector is marginal and the following discussion will therefore be presented assuming linear programmer range. Referring to figure 6-1, an internal pitch command is formed when the glideslope computation is in the arm condition (GSARM). The value of that signal will be

$$\theta = K_{\dot{h}} (K_{\Delta\dot{h}} \Delta\dot{h} + \dot{h}) + K_h \Delta h \qquad \text{Eq. 6-1}$$

where

θ = degrees of pitch command (+ directs a pitch down attitude)

$\Delta\dot{h}$ = approach rate of the glideslope beam (ft/sec)

$\dot{h}$ = ascent rate (ft/sec)

Δh = cross track distance (ft), positive above the glideslope

Any flight path intersecting or almost paralleling the glideslope will result in the rate terms having a polarity opposite to that of Δh. When the rate term equals or exceeds the position term, the glideslope capture mode will be enabled (GSCAP). Two conditions exist which will result in a missed capture: (1) transversal of the capture window while LOCARM is active, and (2) a flight path paralleling the glideslope which never established sufficient proximity and/or closure rate to intersect the capture window.

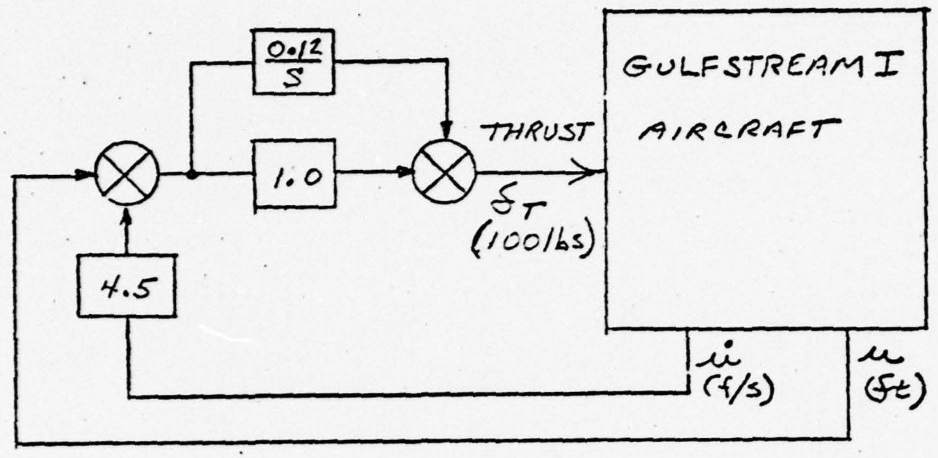


Figure 6-2. AIRSPEED LOOP TO SIMULATE MANUAL PILOT AIRSPEED CONTROL

The glideslope capture window may be more clearly defined by expressing the rate terms as functions of flight path angle:

$$\dot{h} = U_o \sin (a)$$

$$\Delta\dot{h} = U_o \sin (a-a_g)$$

where

U_o = true airspeed

a = flight path angle measured as positive counterclockwise from horizontal

a_g = glideslope angle (nominally -2.5 deg.)

Substituting those expressions into equation 6-1 and setting θ=0 yields the crosstrack distance at capture as a function of flight path angle:

$$\Delta h_c = - \frac{K_{\dot{h}}}{K_h} U_o \; [K_{\Delta\dot{h}} \sin (a-a_g) + \sin a] \qquad \text{Eq. 6-2}$$

where

$K_{\dot{h}}$ = .15 deg/(ft/sec)

K_h = .05 deg/ft

$K_{\Delta\dot{h}}$ = 2.22 (ft/sec)/(ft/sec)

U_o = 200 ft/sec

a_g = -2.5 deg.

then

$$\Delta h = -(33.8a + 58.2) \qquad \text{Eq. 6-3}$$

Equation 6-3 assumes the flight path angle relative to both the glideslope and the horizontal reference line is small enough to allow the substitution of sin(x)=x. The mode logic tests the polarity of equation 6-1 against that of Δh, i.e. if (sgn x) $\neq$ (sgnΔh) then capture is initiated. Therefore, the capture window is defined as:

$$0 \geq \Delta h \geq - \frac{K_{\dot{h}}}{K_h} U_o \; [K_{\Delta\dot{h}} \sin (a-a_g) + \sin a]$$

$$0 \leq \Delta h \leq - \frac{K_{\dot{h}}}{K_h} U_o \; [K_{\Delta\dot{h}} \sin (a-a_g) + \sin a]$$

Figure 6-3 graphically presents the glideslope capture window obtained via equation 6-2. Capture will occur when (1) the aircraft is below the beam and the flight path angle is on or above the line described by equation 6-2, or (2) the aircraft is above the beam and the flight path angle is below the line described by equation 6-2.

Five capture boundary lines are shown to illustrate the effect of longitudinal wind and varying glideslope angles.

Clearly, an RNAV approach paralleling the glideslope must, at some point, close to within 58 feet of the glideslope. Since RNAV errors of 150 feet are not out of the question, there is a possibility of missing the glideslope if capture is attempted with R-NAV assist. However, random variations in the beam as well as the aircraft position will normally result in a capture. A more probable cause of missed capture would be a close-in approach which resulted in a flight path angle in the -1.0 to -.2 degree range. Here, the capture window can become quite small. Similar arguments apply to captures from below the beam. Although it would be possible to modify the capture laws to encompass parallel RNAV approach configurations additional development will be required.

Since the capture window is actually determined by measured position and rate information, the capture point will be moved to compensate for along track and cross track wind conditions as well as variations in glideslope angle. Simulation data included in this report demonstrate the use of capture point movement to maintain a relatively smooth performance during the capture maneuver. Estimates of distance to and closure rate of the beam in equation 6-3 compensate for environmental variations but cannot distinguish between ascending, descending, or constant altitude capture conditions. The altitude rate term does provide this information and is used to extend the capture point thereby suppressing excessive vertical rates. Examination of simulated captures from above the beam shows the complete absence of overshoot under these conditions.

Table 6-1 summarizes data for simulated approaches. The intercept point is defined as the beam altitude at the intersection of the glideslope and the flight path prior to capture. The approaches were simulated to evaluate system performance capturing and tracking the glideslope.

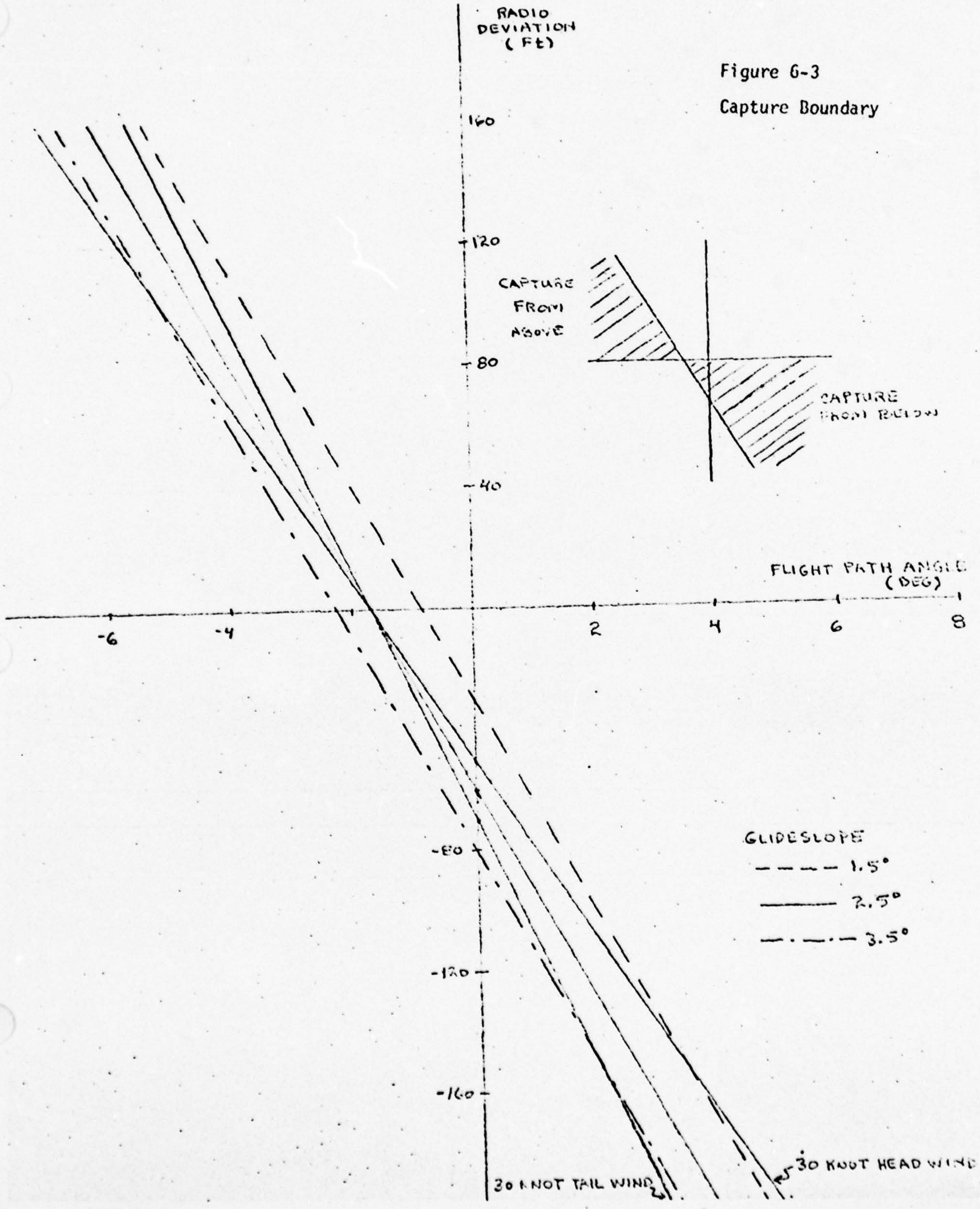


Figure 6-3

Capture Boundary

Table 6-1. Simulated Capture Data

Intercept Point (ft)	Flight Path Angle (deg)	Capture Point (ft)	Predicted Value (ft)	Overshoot (ft)	Settling Time (sec)	Strip Chart
1000	0	-50	-58.2	--	8	1A
1000 (From above)	-5	+125	+132	--	45	1B
1300	0	-50	-58.2	--	10	1C
1300 (From above)	-5	+125	+132	--	45	1D
1500	0	-54	>- 58.2	6	20	1E
1800	0	-58	>- 58.2	12	28	2C
1800 (From above)	-5	+90	<132	--	55	2D

As would be expected the data confirms anticipated results. Along with the previously mentioned adaptive behavior, the gain programmer is seen to reduce the capture window as distance to touchdown increases. This is the only non-linear effect in the capture maneuver.

Analog computer simulation results are somewhat different than the predicted values due to a dead zone in the analog capture logic. The predicted values were attained when simulations were performed using the digital version of the control laws.

6.3.3 Stability Margin Analysis

To analyze the stability of the glideslope control laws of Figure 6-1 frequency response or Bode plots were obtained. Reference to Autothrottles in this discussion is to Figure 6-2. Figure 6-4 is an open loop bode plot of the radio or deviation loop. This plot indicates the stability of the aircraft when tracking the glideslope beam. Autothrottles were also used for Figure 6-4. The aircraft used to obtain this plot was the Gulfstream I approach flight condition (200 ft/sec airspeed). From Figure 6-4 it can be seen that the bandwidth is sufficient to achieve good Category II tracking performance. The stability margins as summarized in Table 6-2 indicate good stability for the glideslope maneuver.

Figure 6-5 illustrates the outer loop or deviation loop broken as in

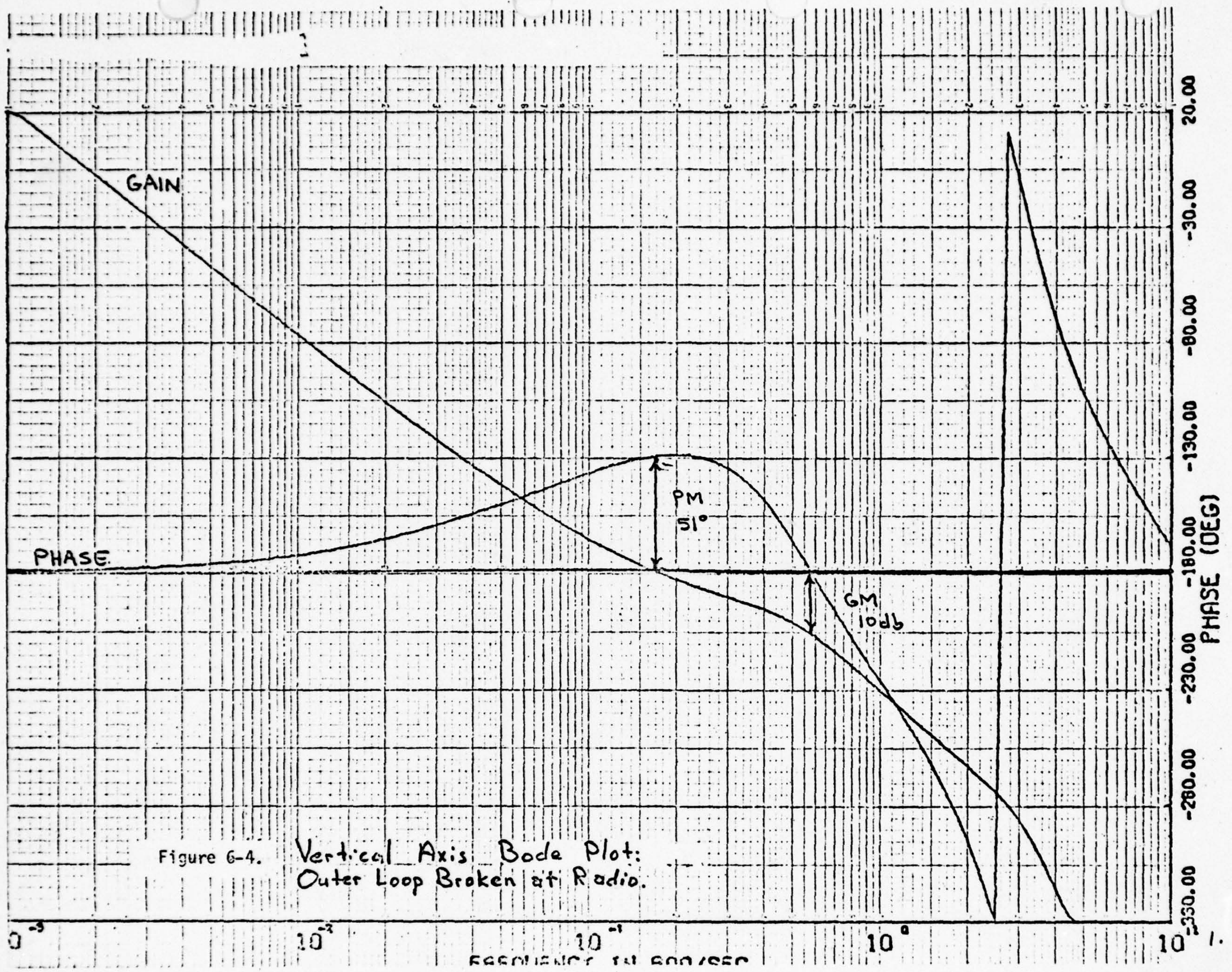


Figure 6-4. Vertical Axis Bode Plot: Outer Loop Broken at Radio.

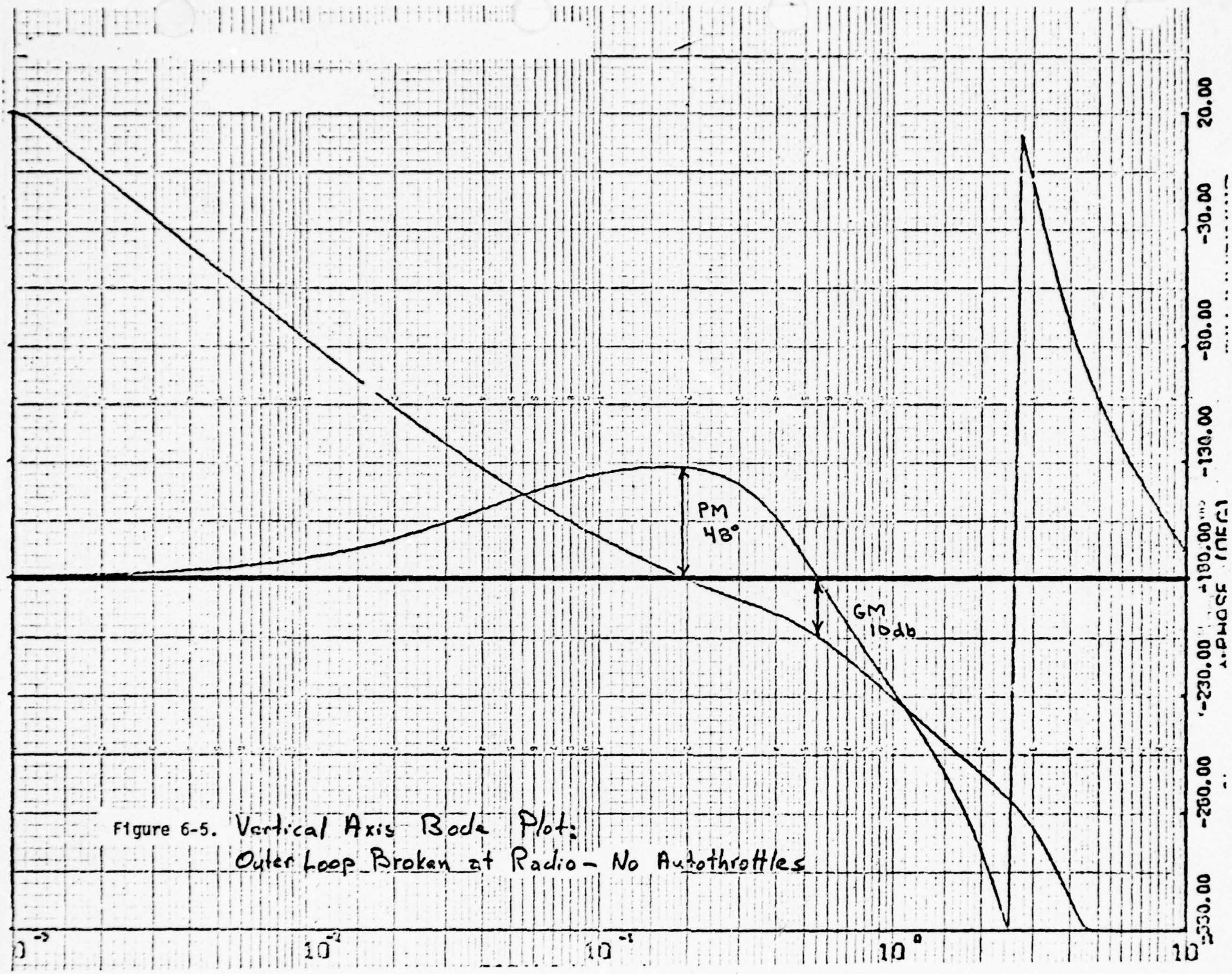


Figure 6-5. Vertical Axis Bode Plot:
Outer Loop Broken at Radio – No Autothrottles

Figure 6-4, but the autothrottles were removed. As can be seen by comparing the two figures, airspeed control does not greatly effect glideslope tracking stability.

Figure 6-6 is a bode plot of the control laws of Figure 6-1 broken in the inner loop or at the servo. All outer loops are closed. This plot is indicative of the inner loop or elevator activity and stability while the system is tracking the glideslope. The bandwidth and stability indicated by Figure 6-6 is typical of good Category II glideslope computations. The stability margins and bandwidth are summarized in Table 6-2. Figure 6-7 is the same as Figure 6-6 except for the lack of autothrottles. As can be seen airspeed control does not effect the inner loop stability or bandwidth to any great extent.

Table 6-2. Frequency Response Data

FIGURE	GAIN MARGIN (db)	PHASE MARGIN (deg)	BANDWIDTH (rod)	OPENED LOOP	CONFIGURATION
6-4	10	51	.175	Radio	Autothrottles
6-5	10	48	.2	Radio	No autothrottles
6-6	10	75	.7	Servo	Autothrottles
6-7	10	75	.7	Servo	No autothrottles

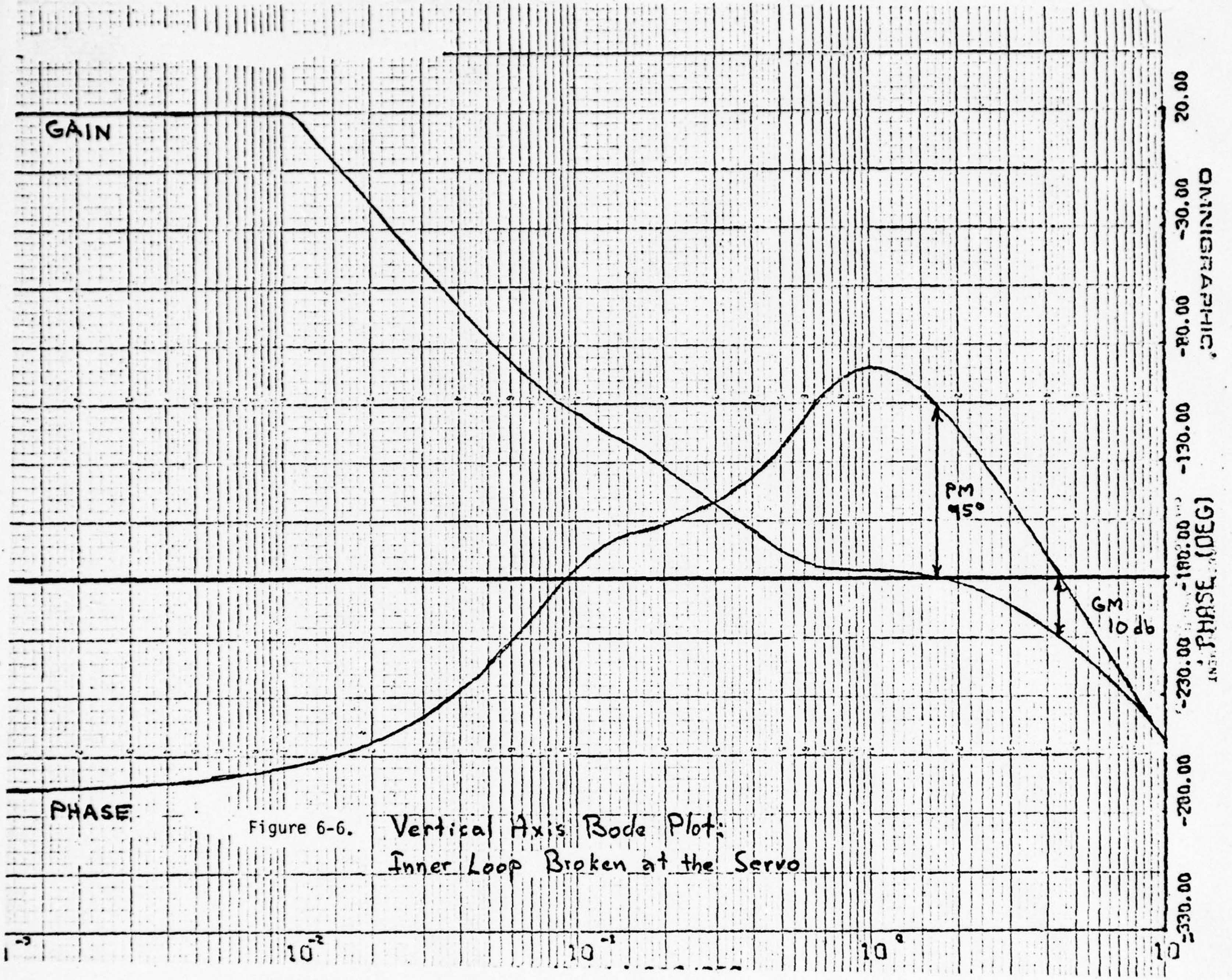


Figure 6-6. Vertical Axis Bode Plot; Inner Loop Broken at the Servo

Figure 6-7. Vertical Axis Bode Plot: Inner Loop Broken at the Servo — No Autothrottles

6.4 Performance

To evaluate the performance of the glideslope computations, the system was simulated on the EAI 680. The system of Figure 6-1 was used with a simulation of the Gulfstream I aircraft (approach flight condition). To simulate manual airspeed control, the autothrottle system shown in Figure 6-2 was used. The autothrottle system was adjusted to give airspeed control similar to what is known to be typical of manual operation. Strip chart recordings were taken using radio noise, vertical wind turbulence, longitudinal wind turbulence and radio steps.

6.4.1 Radio Steps

Strip chart recordings were taken for step changes in glideslope. These recordings illustrated the stability of the glideslope computations when tracking the glideslope beam. The glideslope tracking performance had sufficient stability and bandwidth. The responses were smooth and well dampened.

6.4.2 Vertical Turbulence

The following represents the 2σ vertical turbulence transfer function (ref. 5) used for the performance simulation.

$$\underline{\text{white noise}}\ \boxed{\frac{K}{1+(L/V)S}}\ \underline{\quad \alpha g \quad}$$

where

S = complex variable of Laplacian operator
V = approach speed (200 ft/sec)
L = vertical scale length (30 ft.)
αg = 1.3 deg RMS (2σ level)

It should be strongly pointed out that the RMS level of vertical turbulence represents a 2σ value. In other words, this level of turbulence would be expected on only 5% of the approaches if a random selection were made over a total every day operation.

For the assumed flight condition this model corresponds to .15 sec. correlated noise, and RMS output value of 1.3 degrees was used in the simulation effort. At an airspeed of 200 ft/sec this is equvalent to 5.6 ft/sec (RMS) vertical gusts. System response to wind gusts at fixed distances from the runway is summarized in Table 6-3.

Table 6-3. System Deviations to Vertical Gusts

Gust Strength (deg) RMS	Distance (nm)	Airspeed (f/s) RMS	Pitch (deg) RMS	Pitch Command RMS	Deviation (ft) RMS	Altitude Rate (f/s) RMS	Servo Command (deg) RMS
1.3	7.46	.12	.183	.233	4	1.75	.24
	6	.12	.267	.267	6.67	2.167	.316
	4.54*	.12	.2	.267	4	1.67	.267
	5.7-2.2*	.12	.2	.3	4.53	2.	.3

*typical of performance in the linear range of the radio programmer

The performance exhibited indicates no problem. Performance capable of Cat. II performance has been achieved. Servo or control activity as can be seen from Table 6-3 is satisfactory.

6.4.3 Longitudinal Gust Performance

As was the case for vertical gusts, Reference 5 provided the following 2σ longitudinal gust transfer function:

$$\underline{\text{white noise}}\boxed{\dfrac{K}{1+(L/V)S}}\underline{\;\;ug\;\;} \quad .$$

where

S = complex variable of Laplacian operator
L = longitudinal scale length (600 ft)
V = approach speed (200 f/sec)
µg = 6 ft/sec RMS (2σ level)

An RMS output of 6 ft/sec was employed during simulated approaches. Again data runs were made at fixed distances inside, outside, and at altitudes corresponding to the knee of the radio gain programmer. Table 6-4 summarizes the data recoded.

Table 6-4. *System Deviations to Longitudinal Gusts*

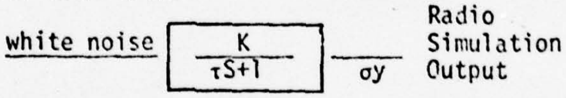
Gust Strength (f/s)RMS	Airspeed (ft/sec) RMS	Pitch (deg) RMS	Pitch Command (deg)RMS	Deviation (ft) RMS	Altitude Rate (f/s)RMS	Servo Command (deg)RMS	Distance (nm)
6.	1.5	.266	.88	5.33	2.0	.267	4.36
	1.5	.317	.567	12.0	2.83	.383	6.0
	1.67	.383	.767	13.3	3.75	.333	7.64
	1.625	.3	.567	7.65	2.75	.333	3.2-.377

*typical approach condition

From Table 6-4 it can be seen that the system tracks the glideslope beam quite well during the longitudinal turbulence. Cat. II performance is achieved. Servo or control activity is indicative of that known acceptable from past experience on other systems.

6.4.4 Radio Noise Performance

The following noise model was used to estimate system performance in the presence of beam noise:



where

S = complex variable of Laplacian operator
τ = 4 sec.
σy = 10 ua RMS

This model is considered a low quality Cat. I facility.

Table 6-5 summarizes the performance from some of the recordings for various distances. Table 6-5 illustrates that Cat. II performance is achieved and servo or control activity is not excessive.

Table 6-5. System Deviations to Radio Noise

Radio Noise (ua)	Distance (nm)	Airspeed (ft/sec) RMS	Pitch (deg) RMS	Pitch Command (deg)RMS	Deviation (ft) RMS	Altitude Rate (ft/sec)	Servo Command (deg)RMS
(12.5 ft) 3.35	7.46	.12	.18	.35	8.0	1.25	.167
(10.65 ft) 3.55	6.0	.117	.25	.434	6.67	1.5	.217
(7.85 ft) 3.45	4.54	.117	.183	.33	7.35	1.25	.125

6.5 CONCLUSION

Simulation and analysis results indicate category II performance can be expected from these glideslope control laws. In addition, these control laws have been shown to be capable of acceptable performance when glideslope captures are initiated from other than an altitude hold mode. This capability may be considered a first step to area navigation aided vertical approach maneuvers.

APPENDIX A

Program Listing of the Flight Plan Editor Utility Module

```
4D.WORKFILE.FPEDF6AX
        ...
        ***************************************************************
        *                                                             *
        *               ANS-70A FLIGHT PLAN AND EDITOR                *
        *                                                             *
        *   THE FLIGHT PLAN IS COMPOSED OF 4-WORD ITEMS OF THE FOLLOWING  *
        *   TYPES                                                     *
        *                                                             *
        *       (1) LATERAL EVENTS                                    *
        *       (2) VERTICAL EVENTS                                   *
        *       (3) HOLD EVENTS                                       *
        *       (4) ORIGIN EVENTS                                     *
        *       (5) DESTINATION EVENTS                                *
        *       (6) END-OF-CONTINUITY EVENTS                          *
        *                                                             *
        ***************************************************************
                                                                     //
              BEGIN                           ... ***** GLOBAL SPAN *******//
        .INSERT          AEDLIB;              ... AED LIBRARY            //
        .INSERT          FPEDF6I1;            ... FLIGHT PLAN EDITOR FILE  //
        .INSERT          MATHPACK ;           ... MATH PACK INSERT FILE      //
        .INSERT         FPEDF6I2;             ... INTERNAL FPE INSERTS     //
        ... ******************** BOOLEAN DECLARATIONS ***********************//
              BOOLEAN
                        PPSNST;               ... PRES POS UPDATE FLAG      //
              PRESET PPSNST=FALSE;
        ... ************************ ARRAYS ********************************//
              INTEGER ARRAY
                        EVMASK(6),                                                0000004;
                        ARAYFE(2),            ... LLTBD ANSWER READ.        //
                        FET(FET.SIZE),        ... FLIGHT EVENT TABLE        //
                        RWS(RWS.SIZE);        ... REF WAYPOINT STACK        //
              PRESET EVMASK =
                        0,                    ... ARRAY OFFSET              //
                        DMSK,                 ... DESTINATION EVENT MASK     //
                        OMSK,                 ... ORIGIN MASK                //
                        CMSK,                 ... CONTINUITY MASK            //
                        HMSK,                 ... HOLD MASK                  //
                        VMSK,                 ... VERTICAL MASK              //
                        LMSK ;                ... LATERAL MASK               //
        ... ******************** POINTER DECLARATIONS ***********************//
              POINTER
                        EOC.PTR,              ... END OF CONTINUITY POINTER //
                        STRT.RWS,             ... RWS START POINTER          //
                        STOP.RWS,             ... RWS STOP POINTER           //
                        ANSBD,                ... POINTER TO LLTBD ANSWER   //
                        FETAB,                ... 1ST WORD OF FET POINTER   //
                        FESTRT,               ... POINTER TO FET 1ST ITEM   //
                        FETEND,               ... POINTER TO FET LAST ITEM //
                        ABS.1;                ... POINTER TO FET ITEM 1     //
              PRESET ANSBD = ARAYFE;
        ...
        ******************** INTEGER DECLARATIONS ***********************.*
                                                                     //
          INTEGER
                  CEVENT;                     ... CONTROL WORD FOR EOC EVENT//
```

```
        PRESET CEVENT="03FFFFFF";
   ... ******************* PROCEDURE DECLARATIONS ***********************//
           POINTER PROCEDURE
                     RWSSRCH;                  ... RWS SEARCH ROUTINE        //
           PROCEDURE
                     BCLEAR,                   ... CLEAR ALL OFFSETS        //
                     ZEROF,                    ... ZERO FILL ROUTINE        //
                     OGDEINIT,                 ... ORG-DEST UNEED            //
                     COPYRTN;                  ... RWS COPY RTN              //
           REAL PROCEDURE
                     SIGN;                     ... SIGN OF REAL NUMBER      //
   ... ********************** EXTERNAL DECLARATIONS *********************//
           EXTERNAL
                     BCLEAR,
                     SIGN,
                     HPSNST,
                     CEVENT,
                     EOC.PTR,
                     RWSSRCH,
                     COPYRTN,                                                              00000094
                     STRT.RWS,
                     ZEROF,
                     STOP.RWS,
                     FETAB,
                     FESTRT,
                     FETEND,
                     ABS.1,
                     FET,
                     RWS;
           SYNONYMS  COPYRTN = CPYRTN;                                                     0000010A
           BEGIN                               ... ***** PROCEDURES *****   //
   ...
   *************************************************************************
   *                                                                       *
   *                               REPLACFE                                *
   *                                                                       *
   *     VALUE = REPLACFE(STRTVNO,STOPVNO,NOITEMS,PROC,WKPTR)              *
   *                                                                       *
   *          THIS IS AN INTEGER PROC USED AS THE FET REPLACE ROUTINE.     *
   *          ITEMS FROM EVENT NUM STRTVNO THRU EVENT NUM STOPVNO ARE      *
   *          DELETED. ANY NEW ITEMS ARE THEN INSERTED WITH ADDSTRNG.      *
   *                                                                       *
   *                                                                       *
   *                                                                       *
   *                                                                       *
   *************************************************************************
   *                                                                      //
   DEFINE INTEGER PROCEDURE REPLACFE(STRTVNO,STOPVNO,NOITEMS,PROC,WKPTR)
           WHERE INTEGER STRTVNO,STOPVNO,NOITEMS;
                 POINTER PROC;
                  POINTER WKPTR
        TOBE BEGIN                             ... ***** PROC REPLACFE *****//
         OWN POINTER STRT;
         OWN POINTER STOP;
         OWN POINTER PNTR;
         OWN INTEGER SVEVNSF;
         OWN INTEGER ITEM.NO;
```

```
                                       ... PROGRAM START                //
    ITEM.NO=NOITEMS;                   ... LOAD OWN REGISTER            //
    IF SVEVRSF=VERSFE NEQ VERS(WKPTR)   ... CHECK VERSION NUMBER//
    THEN BEGIN                         ... ***** VERSION MISMATCH***//
         REPLACFE=NOVER;                ... REPORT NO VERSION     //
         GOTO RETURN;
         END;                          ... ***** VERSION MISMATCH***//
    VERSFE=0;                          ... INHIBIT ACCESS TO FET       //
    STRT = CONVFE(STRTVNO);           ... MAKE START OF DELETE PTR //
    STOP = CONVFE(STOPVNO);           ... MAKE STOP OF DELETE PTR  //
    IF RWS.CHCK(NOITEMS) LES ZERO ... CHECK RWS FOR ROOM          //
    THEN GOTO NO.ROOM;                 ... CAN'T CHANGE RWS OVERFLOW//
                                       ... A ONE-FOR-ONE REPLACE IS
                                           A SPECIAL CASE WHERE ONE
                                           LATERAL EVENT IS
                                           OVERWRITTEN BY ANOTHER. A
                                           ONE-FOR-ONE REPLACE
                                           REQUIRES THE FOLLOWING
                                             1. EVENTS MUST BE
                                                LATERAL EVENTS
                                             2. ONE ITEM MUST BE
                                                DELETED
                                             3. ONE ITEM MUST BE
                                                ADDED
                                             4. ID'S MUST MATCH         //
    IF TYPEFF(STRT) EQL LTYP           ... MUST BE LATERAL EVENT      //
    THEN BEGIN                         ... ***** LATERAL EVENT *****//
         IF WO(RWSPTR(STRT)) EQL      ... ID'S MUST MATCH             //
            WYPTID(WKPTR)
         AND STRT EQL STOP              ... DELETE 1 ITEM            //
         AND NOITEMS EQL 1              ... ADD 1 ITEM               //
         THEN BEGIN                    ... ***** ONE-FOR-ONE REPLACE//
              INSERT(WKPTR,STRT);      ... OVERWRITE LATERAL EVENT  //
              ITEM.NO=ZERO;            ... FIX FOR AUDSTRNG            //
              GOTO GOODRTN;            ... GO FIXUP FET AND RETURN  //
              END;                     ... ***** ONE-FOR-ONE REPLACE//
         END;                          ... ***** LATERAL EVENT *****//
                                       ... THE FLIGHT PLAN EDITOR
                                           MUST INSURE THAT
                                           SUFFICIENT ROOM EXISTS IN
                                           THE FET FOR THE NEW ITEMS
                                           PRIOR TO MAKING ANY EDITS
                                           TO THE FET                  //
    IF STOP - STRT + ITM.SIZE          ... ROOM MADE BY DELETE PLUS //
       + FESTRT - FETAB                ... UNUSED FET IS COMPARED     //
       LES NOITEMS * ITM.SIZE          ... WITH MAX SIZE OF INSERT  //
    THEN BEGIN                         ... ***** NO ROOM ******       //
NO.ROOM:
         VERSFE = SVEVRSF;             ... REPLACE OLD VERSION NUM   // 00000282
         REPLACFE = NORUM;             ... REPORT NO ROOM             // 00000283
         GOTO RETURN;
         END;                          ... ***** NO ROOM ******       //
                                       ... DELETE ALL ITEMS FROM
                                           STRTVNO TO STOPVNO
                                           INCLUSIVE. DELETE CONSISTS
                                           OF OVERWRITING SPECIFIED
                                           ITEMS.                      //
```

```
                                              ... NEW ITEMS ARE ADDED USING
                                                  THE ROUTINE ADDSTRNG
                                                  RATHER THAN ADDATFE SO
                                                  THAT VERSFE CAN BE
                                                  INDEPENDENTLY CONTROLLED
                                                  BY BOTH REPLACFE AND
                                                  ADDATFE                    //
               PNTR=CMPRSFE(STRT,STOP);       ... OVERWRITE REPLACED EVENTS //
     GOODRTN:
               ADDSTRNG(PNTR,ITEM,NO,PROC,WKPTR);  ... ADD NEW EVENTS       //
                                              ... FIXFET WILL VALIDATE ALL
                                                  RWS ITEMS, CALCULATE DFTO
                                                  FOR ALL ITEMS AND
                                                  CALCULATE FLIGHT PARAMS
                                                  IN NEIGHBORHOOD OF NEW
                                                  ITEMS                      //
               FIXFET(STOP,NOITEMS);          ... FIXUP FET AND RWS          //
               VERSFE=SVEVRSF+1;              ... BUMP VERSION NUMBER        //
               REPLACFE = GDRTN;              ... REPORT GOOD RETURN         //
               GOTO RETURN;
               END;                           ... ***** PROC REPLACFE *****//
     ...
     ***************************************************************************
     *                                                                         *
     *                             ADDATFE                                     *
     *                                                                         *
     *    VALUE = ADDATFE(ABSEVNO,NOITEMS,PROC,WKPTR)                          *
     *                                                                         *
     *         THIS IS AN INTEGER PROCEDURE USED AS THE FET INSERTION          *
     *         ROUTINE. IT WILL INSERT ITEMS IMMEDIATELY ABOVE ABSEVNO         *
     *         (PUSH-UP INSERT) UNTIL EITHER NOITEMS OF ITEMS HAS BEEN         *
     *         INSERTED OR PROC RETURNS A NULL. PROC RETURNS A POINTER         *
     *         TO A ROUTINE THAT GENERATES THE NEXT ITEM TO INSERT AND         *
     *         PLACES IT IN A WORK AREA POINTED TO BY WKPTR.                   *
     *                                                                         *
     ***************************************************************************
     *                                                                        //
     DEFINE INTEGER PROCEDURE ADDATFE(ABSEVNO,NOITEMS,PROC,WKPTR)
               WHERE
                    INTEGER ABSEVNO, NOITEMS;
                    POINTER PROC;
                    POINTER WKPTR
               TOBE
               BEGIN                          ... ***** PROC ADDATFE ******//
               OWN POINTER EDIT,PTR;
               OWN INTEGER SVEVRSF;
                                              ... THE FLIGHT PLAN EDITOR
                                                  MUST INSURE THAT THE
                                                  VERSION NUMBER BEING
                                                  INPUT AGREES WITH THE
                                                  CURRENT FET VERSION
                                                  NUMBER - VERSFE            //
               IF SVEVRSF=VERSFE NEQ VERS(WKPTR)  ... CHECK VERSION NUMS   //
               THEN BEGIN                         ... ***** MISMATCH *****//
                    ADDATFE=NOVER;                ... REPORT NO VERSION    //
                    GOTO RETURN;
                    END;                          ... ***** MISMATCH *****//
```

```
                                      ... THE FLIGHT PLAN EDITOR
                                          MUST INSURE THAT THERE IS
                                          ADEQUATE ROOM FOR THE
                                          ITEMS TO BE ADDED PRIOR
                                          TO CHANGING THE FET      //
      IF RWS.CHCK(NOITEMS) LES ZERO             ... CHECK RWS      //
      OR (FESTRT-FETAB) LES NOITEMS*ITM.SIZE  ... AND FET          //
      THEN BEGIN                      ... ***** NO ROOM ******     //
           ADDATFE=NOROM;             ... REPORT NO ROOM           //
           GOTO RETURN;
           END;                       ... ***** NO ROOM ******     //
      IF ABSEVNO LEQ ZERO             ... INSURE LEGIT EVENT NUMBER//
      THEN BEGIN                      ... ***** BAD EVENT NUM *****//
           ADDATFE=NOEVT;             ... REPORT NO SUCH EVENT     //
           GOTO RETURN;
           END;                       ... ***** BAD EVENT NUM *****//
      VERSFE = 0;                     ... INHIBIT FET              //
      EDIT.PTR = CONVFE(ABSEVNO);   ... CONSTRUCT ITEM POINTER   //
      IF TYPEFE(EDIT.PTR) EQL OTYP
      THEN BEGIN                      ... ***** OVERWRITE ORIGIN ***//
           EDIT.PTR=CMPRSFE(EDIT.PTR,EDIT.PTR);     ... DELETE ORG //
           PPSNST=TRUE;               ... SHOW P.PSN INVALID       //
           END;                       ... ***** OVERWRITE ORIGIN ***//
      IF LCMSK .A. EVMASK(ECWTYP(WKPTR)) ... ADD LE & EOC BEFORE //
         EQL ZERO                      ... AT-PNTR, ADD OTHERS //
      THEN EDIT.PTR=EDIT.PTR+ITM.SIZE;   ... AFTER AT-PNTR         //
                                      ... NEW ITEMS ARE ADDED USING
                                          THE ROUTINE ADDSTRNG
                                          RATHER THAN DIRECTLY SO
                                          THAT VERSFE CAN BE
                                          CONTROLLED INDEPENDENTLY
                                          AT ADDATFE AND REPLACFE  //
      ADDSTRNG(EDIT.PTR,NOITEMS,PROC,WKPTR);  ... ADD ITEMS.       //
                                      ... FIXFET WILL VALIDATE ALL
                                          RWS ITEMS, CALCULATE DFTO
                                          FOR ALL ITEMS AND CALC
                                          FLIGHT PARAMS IN
                                          NEIGHBORHOOD OF NEW ITEMS//
      FIXFET(EDIT.PTR-ITM.SIZE,NOITEMS); ... FIX FET & RWS         //
      VERSFE=SVEVRSF+1;               ... BUMP VERSION NUMBER      //
      ADDATFE=GDRTN;                  ... REPORT GOOD RETURN       //
      GOTO RETURN;
      END;                            ... ***** PROC ADDATFE ******//
...
**********************************************************************
*                                                                    *
*                               ADDSTRNG                             *
*                                                                    *
*     ADDSTRNG(AT.PNTR,NOITEMS,PROC,WKPTR)                           * 00000352
*                                                                    *
*          THIS NON-VALUED PROCEDURE IS USED TO INSERT A STRING OF   *
*          FET ITEMS AT A PRE-SELECTED ITEM IN THE FET. PROC IS A    *
*          PROCEDURE THAT SUPPLIES A POINTER TO THE PROCEDURE THAT   *
*          FILLS THE WORK AREA POINTED TO BY WKPTR.                  *
*                                                                    *
**********************************************************************
*                                                                    //
```

```
DEFINE PROCEDURE ADDSTRNG(AT.PNTR,NOITEMS,PROC,WKPTR)
          WHERE POINTER AT.PNTR;
                INTEGER NOITEMS;
                POINTER PROCEDURE PROC;
                POINTER WKPTR TOBE
          BEGIN                                     ... ***** PROC ADDSTRNG *****//
          OWN POINTER FETCH;
          OWN POINTER PNT1;
          OWN POINTER PNT2;
          OWN POINTER PNTR;
          OWN POINTER FRMLEV;
          OWN POINTER FRMVLV;
          OWN POINTER ADD.PTR;
          OWN INTEGER ITEMS;
         OWN INTEGER TYPE;
          PNT1 $=$ PNTR;
          PNT2 $=$ FRMLEV;
          FETCH $=$ FRMVLV;
                                                    ... PROGRAM START             //
          FETCH=LOC PROC;                           ... GET POINTER TO FETCH PROC//
          ADD.PTR=AT.PNTR;                          ... LOAD OWN REGISTER         //
          ITEMS=NOITEMS;                            ... ITEM COUNT                //
NEXT.ITM:
          IF FETCH NEQ NULL AND                     ... EITHER PROC RETURNING     //
                ITEMS NEQ 0                         ... NULL OR 0 ITEMS = DONE.   // 00000389
          THEN BEGIN                                ... ***** ITEM TO PROCESS ***//
                ITEMS=ITEMS-1;                      ... DECR ITEM COUNT.          //
                FETCH=PDOIT(FETCH,WKPTR); ... FILL WORK AREA. GET PTR.//
                                                    ... USUAL CASE OF ADDSTRNG
                                                        OPENING ITEMS BY PUSHING
                                                        THE TOP OF THE FET UPWARD
                                                        TO OPEN AN ITEM. THE
                                                        ROUTINE INSERT THEN
                                                        BUILDS AND INSERTS THE
                                                        ITEM. AN RWS ITEM IS ALSO
                                                        OPENED IF NECESSARY. THIS
                                                        PROCESS WORKS ONE ITEM AT
                                                        A TIME BECAUSE IT IS NOT
                                                        KNOWN HOW MANY ITEMS ARE
                                                        INVOLVED (NOITEMS IS A
                                                        MAX NUMBER)               //
                PNT1=FESTRT-ITM.SIZE;               ... TO POINTER                //
                PNT2=FESTRT;                        ... FROM POINTER              //
                FESTRT=PNT1;                        ... UPDATE START OF FET ITEMS//
OPEN.ITM:
                IF PNT2 EQL ADD.PTR                 ... PUSH FET UP FM "AT" ITEM //
                THEN BEGIN                          ... ***** INSERT ITEM ****** //
                     INSERT(WKPTR,PNT1);            ... PNT1 POINTS AT OPEN SPACE//
                     GOTO NEXT.ITM;                 ... GET NEXT ITEM             //
                     END;                           ... ***** INSERT ITEM ****** //
                WO(PNT1)=WO(PNT2);                  ... TAKE FROM. PUT TO.        //
                PNT1=PNT1+1;                        ... BUMP TO POINTER.          //
                PNT2=PNT2+1;                        ... BUMP FROM POINTER         //
                GOTO OPEN.ITM;                      ... CONTINUE TO OPEN AN ITEM //
                END;                                ... ***** ITEM TO PROCESS ***//
                                                    ... THE NEXT SECTION CLEANS
                                                        UP THE FET. THE 'FROM'
```

```
                                                 AND 'TO' EVENTS ARE
                                                 DEFINED. THE START OF THE
                                                 FET IS FOUND. EVENT
                                                 NUMBER 1 IS FOUND         //
          PNTR=FESTRT;                       ... FIRST, MAKE TOEVE & FRMEV//
LOOP:                                        ... INVALID ON ALL ITEMS.     //
          W1(PNTR)=W1(PNTR) .V. 6;                                           00000474
          PNTR=PNTR+ITM.SIZE;
          IF PNTR LES FETEND
          THEN GOTO LOOP;
          FRMLEV=FRMVLV=NULL;                ... INIT POINTERS             //
                                             ... FIND 'FROM' EVENTS        //
          PNTR=FESTRT;                       ... POINT AT OLD START OF FET//
LOOP1:
          IF W1(PNTR) .A. "8" EQL ZERO       ... CHECK FOR HISTORY FLAG    // 00000489
          THEN BEGIN                         ... **** PROCESS HISTORY ***//
               IF TYPEFE(PNTR) EQL LTYP      ... IF EVENT IS LATERAL EVENT// 00000493
               THEN BEGIN                    ... **** DEFINE 'FROM' LF **//
                    FRMLEV=PNTR;             ... LAST DELETED LE FM FESTRT//
                    GOTO CONTINUE;
                    END;                     ... **** DEFINE 'FROM' LF **//
               IF TYPEFE(PNTR) EQL VTYP      ... IF ITEM IS VE             // 0000049A
               THEN BEGIN                    ... **** DEFINE 'FROM' VF **//
                    W1(PNTR)=W1(PNTR) .A. "FFFFFFFB"; ... FRMLEV=VALID//00000500
                    VPAIFE(PNTR)=0.0;        ... IN CASE ONLY ONE VE.      //
                    FRMVLV = FRMLEV;         ... VE ALWAYS FOLLOWS A LF    //
                    END;                     ... **** DEFINE 'FROM' VF **//
CONTINUE:
               PNTR=PNTR+ITM.SIZE;           ... BUMP ITEM POINTER         //
               GOTO LOOP1;                   ... GO TRY NEXT ITEM          //
               END;                          ... **** PROCESS HISTORY ***//
          ABS.1 = PNTR;                      ... EVENT 1 IS 1ST ACTIVE EVT//
          FESTRT=PNTR;                       ... IN CASE NO HISTORY YET    //
          IF FRMLEV NEQ NULL                 ... IF THERE IS ANY HISTORY,  //
          THEN BEGIN                         ... **** DETERMINE FESTRT **//
               C.BITS(FRMLEV)=C.BITS(FRMLEV) .A. "FB"; ... FRMLV=VALD//
               FESTRT = FRMLEV;              ... FET INCLUDES FROM-LE      //
               IF FRMVLV NEQ NULL            ... IF A 'FROM' VE EXISTS.    //
               THEN FESTRT = FRMVLV;         ... ITS LE PTR IS LEQ FRMLEV//
               END;                          ... **** DETERMINE FESTRT **//
          IF TYPEFE(FESTRT) EQL DTYP         ... CALL ORG/DEST IF NO EVTS //
          THEN ADDFINIT();
          IF PNTR=SRCHFET(ABS.1-ITM.SIZE,OCLHMSK) NEQ NULL  ... EVT 1 //
          THEN BEGIN                         ... **** SET 'TO' FLAGS ****//
               C.BITS(PNTR)=C.BITS(PNTR) .A. "FD"; ... TOEVE=VALID    //
               IF PNTR=SRCHFET(ABS.1-ITM.SIZE,VMSK) NEQ NULL  ... VE  //
               THEN C.BITS(PNTR)=C.BITS(PNTR) .A. "FD"; ... TO=VALID //
               END;                          ... **** SET 'TO' FLAGS ****//
          GOTO RETURN;                       ... EXIT ADDSTRNG             //
          END;                               ... **** PROC ADDSTRNG *****//
...
*****************************************************************************
*                                RWS.CHCK                                   *
*     VALUE=RWS.CHCK(NOITEMS)                                               *
*                                                                           *
*          THIS ROUTINE SEARCHES THE RWS COUNTING EMPTY ITEMS. IT THEN      *
*          COMPARES THE RESULT WITH NOITEMS. IF NOITEMS IS GREATER,         *
```

```
*           A NEGATIVE VALUE IS RETURNED.                              *
*                                                                      *
************************************************************************
*                                                                     //
CEFINE INTEGER PROCEDURE RWS.CHCK(NOITEMS)
            WHERE INTEGER NOITEMS TOBE
            BEGIN                               ... ****** PROC RWS.CHCK ****//
           OWN INTEGER CNTR;
            OWN POINTER PNTR;
                                                ... PROGRAM START          //
            CNTR=ZERO;
            FOR PNTR=STRT.RWS STEP 3            ... COUNT EMPTY ITEMS IN RWS.//
            UNTIL STOP.RWS
            DO    IF WO(PNTR) EQL ZERO          ... ZERO FOR ID MEANS EMPTY  //
                  THEN CNTR=CNTR+1;
            RWS.CHCK=CNTR-NOITEMS;              ... RETURN NEG IF NO ROOM    //
            GOTO RETURN;
            END;                                ... ****** PROC RWS.CHCK ****//
...
************************************************************************
*                               SRCHFET                                *
*     POINTER = SRCHFET(STRT.AFT,TYPMASK)                              *
*                                                                      *
*                THIS ROUTINE SEARCHES THE FLIGHT PLAN FOR THE FIRST   *
*                EVENT OF A TYPE SPECIFIED BY TYPMASK. A POINTER TO    *
*                THIS ITEM IS RETURNED. A NULL INDICATES NOT FOUND.    *
*           SEARCH BEGINS AT ITEM FOLLOWING STRT.AFT.                  *
*                                                                      *
************************************************************************
*                                                                     //
DEFINE POINTER RECURSIVE PROCEDURE SRCHFET(STRT.AFT,TYPMASK)
            WHERE POINTER STRT.AFT;
                  INTEGER TYPMASK TOBE
            BEGIN                               ... ***** PROC SRCHFET ******//
            OWN POINTER PNTR;
            PNTR=STRT.AFT;                      ... LOAD OWN REG             //
            IF PNTR LES FESTRT                  ... DON'T SEARCH BEFORE START//
            THEN PNTR=FESTRT-ITM.SIZE;
LOOP:
            PNTR=PNTR+ITM.SIZE;                 ... BUMP ITEM POINTER        //
            IF PNTR GRT FETEND                  ... CHECK FET LIMIT.         //
            THEN BEGIN                          ... EXIT                     //
                  SRCHFET = NULL;               ... NOT FOUND                //
                  GOTO RETURN;                  ... RETURN                   //
                  END;
            IF TYPMASK .A. EVMASK(TYPEFE(PNTR))
               EQL ZERO                         ... CHECK TYPMASK WITH LOOKUP//
            THEN GOTO LOOP;                     ... NO HIT                   //
            SRCHFET = PNTR;                     ... RETURN POINTER TO HIT    //
            GOTO RETURN;                        ... RETURN                   //
            END;                                ... ***** PROC SRCHFET ******//
...
************************************************************************
*                                                                      *
*                               CONVERT                                *
*     VALUE = CONVEL(EVENTNO)                                          *
*                                                                      *
```

```
*          THIS ROUTINE CONVERTS AN ABSOLUTE EVENT NUMBER TO A              *
*          FLIGHT EVENT TABLE ITEM POINTER. THIS IS COMPLICATED             *
*          BY THE FACT THAT THE ABSOLUTE EVENT NUMBERING SCHEME             *
*          IS SUCH THAT EVENTS ARE NUMBERED ..,-2,-1,+1,+2,..               *
*          THE ABSENCE OF A ZERO EVENT REQUIRES SPECIAL HANDLING.           *
*                                                                           *
*****************************************************************************
*                                                                          //
DEFINE POINTER RECURSIVE PROCEDURE CONVFE(EVENTNO)
           WHERE INTEGER EVENTNO TOBE
           BEGIN                                 ... ***** PROC CONVERT ******//
           OWN POINTER PNTRTMP;
                                                 ... EVENT NUM TIMES 4 PLUS
                                                     ADDR OF ITEM 1 IS OK IF
                                                     EVENT NUM WAS NEG, BUT IS
                                                     ONE ITEM OFF IF POS.      //
           PNTRTMP=ABS.1+(EVENTNO*ITM.SIZE);
           IF EVENTNO GEQ 0                      ... POS OR NEG?               // 00000606
           THEN PNTRTMP=PNTRTMP-ITM.SIZE;    ... POS. BACK UP 1 ITEM           //
           CONVFE = PNTRTMP;
           GOTO RETURN;                          ... RETURN                    //
           END;                                  ... ***** PROC CONVERT ******//
...
*****************************************************************************
*                                  AOFSFE                                   *
*     AOFS = AOFSFE(LE.PTR)                                                 *
*                                                                           *
*          THIS ROUTINE CALCULATES AOFS GIVEN A POINTER TO A LATERAL        *
*          EVENT. IT USES THE FOLLOWING CALCS:                              *
*                                                                           *
*                    IF ABS(COCH(I)) > 160 DEG                              *
*                                AND                                        *
*                       SIGN COCH(I) = SIGN POFS                            *
*                                OR                                         *
*                    IF ABS(COCH(I)) < 160 DEG                              *
*                    THEN AOFS = -POFS*TAN(COCH(I)/2)                       *
*                                                                           *
*                    IF ABS(COCH(I)) > 160 DEG                              *
*                                AND                                        *
*                       SIGN COCH(I) NEQ SIGN POFS                          *
*                    THEN AOFS(I) = 0.0                                     *
*****************************************************************************
*                                                                          //
DEFINE REAL RECURSIVE PROCEDURE AOFSFE(LE.PTR)
           WHERE POINTER LE.PTR TOBE
           BEGIN                                 ... ***** PROC AOFSFE *****  //
           OWN POINTER PNTR;
           OWN REAL HALF;
           OWN REAL ANG;
                                                 ...
                                                 START OF EXECUTABLE CODE
                                                                               //
           PNTR = LE.PTR;                        ... LOAD OWN REGISTER         //
           HALF=(RI(PNTR)-RO(PNTR))*0.5;     ... HALF=CRSO-CRSI/2              // 00000564
           AOFSFE = 0.0;                         ... PRESET TO SEVERE ANGLE    //
           ANG = ABS(HALF);                      ... ABSOLUTE MAG OF HALF      //
                                                 ... IF AIRCRAFT IS ON OUTSIDE
```

```
                                                OF A SEVERE COURSE CHANGE
                                                (GRT 160 DEG) FORCE AOFS
                                                TO ZERO MAKING TURN BEGIN
                                                WHEN AIRCRAFT EVEN WITH
                                                WAYPOINT.                //
              IF PFSTAT EQL PARALLEL AND
              ((ANG GRT DEG80 AND
              HALF * POFSFE GEQ 0.0)
              OR ANG LES DEG80)
              THEN AOFSFE=-POFSFE*NSIN(HALF)/NCOS(HALF);
              GOTO RETURN;
              END;                              ... ***** PROC AOFSFE *****  //
        ...
        *****************************************************************
        *                              INSERT                           *
        *     INSERT(FRM.PTR,TO.PTR)                                    *
        *                                                               *
        *     THIS ROUTINE USES THE INFO IN THE WORK AREA POINTED AT    *
        *     BY FRM.PTR TO BUILD AN FET ITEM OF THE APPROPRIATE TYPE   *
        *     AND PLACE IT AT THE ITEM POINTED AT WITH TO.PTR.          *
        *                                                               *
        *****************************************************************
        *                                                                //
        DEFINE PROCEDURE INSERT(FRM.PTR,TO.PTR)
        WHERE POINTER FRM.PTR, TO.PTR
        TUBE      BEGIN                         ... ***** PROC INSERT ****** //
              OWN POINTER FROM;
              OWN POINTER TT;
              OWN INTEGER TYPE;
              OWN INTEGER SAVE;
              OWN POINTER PNTR;
              OWN INTEGER TEMP;

              FROM = FRM.PTR;                   ... POINT AT WORK AREA       //
              TT = TO.PTR;                      ... POINT AT EMPTY ITEM      //
              TYPE = EGWTYP(FROM);              ... GET EVENT TYPE           //
              TYPEFF(TT) = TYPE;                ... SET FIELD IN NEW ITEM    //
              GOTO TYPE.SW(TYPE - 2);           ... JUMP THRU ITEM TYPE TABLE//
        SWITCH TYPE.SW =  STOR.CW, HE.TYPE, VE.TYPE, LE.TYPE;                    00000695
                                                ... HOLD EVENT TYPE ITEM     //
        HE.TYPE:
                                                ... LATERAL EVENT TYPE ITEM  //
        LE.TYPE:
              IF PNTR=RWSSRCH(WYPTID(FROM)) EQL NULL   ... ID IN RWS?        //
              THEN BEGIN                        ... ***** NEW RWS ITEM ******//
                   PNTR=RWSSRCH(0);             ... FIND EMPTY ITEM IN RWS   //
                   BEFPTR(PNTR) = ZERO;         ... KILL BD POINTER IN RWS   //
                   END;                         ... ***** NEW RWS ITEM ******//
              SAVE=BEFPTR(PNTR);                ... RWS POINTER MUST BE SAVED//
              CPYRTR(PNTR,FROM);                ... COPY NEW RWS ITEM        //
              CTLWRD(PNTR)=                     ... TRANSFER ELCENT, ELAT    //
                   (CNTRWORD(FROM) .RS. 17) .A. "70";     ... AND ELCUG.//
              RO(TT)=CRSI(FROM);                ... STORE CRSI FOR CRS EDIT. //
              SPEED(TT)=IASI(FROM);             ... STORE IAS WORD           //
              IF TEMP=ROFSF(FROM) EQL PARALLEL    ... CDU USES PARALLEL TO //
              THEN TEMP=NCPOFS;                   ... INSERT P-OFFSET,     //
              OFTFG(TT)=TEMP;                     ... STORE BYTE 2 OF SPEED//
```

```
                REFPTR(PNTR)=SAVE;              ... RESTORE RWS POINTER      //
                RWSFE(TT)=(PNTR-STRT.RWS+3)/3;    ... CALC RWS ITEM NUMBER//
        VE.TYPE:
                W2(TT)=W1D(FROM);               ... ALTD AND AWYID GO IN WRD2//
        STOR.CW:
                C.BITS(TT)=
                (CNTRWORD(FROM) .RS. 3) .V. 6;    ... TOEVF=FRMEV=INVALID   // 0000065A
                IF TYPE NEQ LTYP
                THEN IBYTES(TT)=AUX.CTL(FROM);  ... STORE EXTRA CTL BITS   // 00000689
                GOTO RETURN;
                END;                            ... ***** PROC INSERT ****** //
        ...
        *****************************************************************
        *                           CMPRSFE(STRT,STOP)                  *
        *                                                               *
        *     NEW.STOP = CMPRSFE(EDT.STRT,EDT.STOP)                     *
        *                                                               *
        *       THIS ROUTINE EXPANDS THE SPAN OF THE EDIT TO INCLUDE ANY *
        *       TRAILING EVENTS DEPENDENT UPON EDITED LATERAL EVENTS. IT *
        *       THEN COMPRESSES THE FET BY MOVING ANY ITEMS BETWEEN FESTRT *
        *       AND THE START OF THE EDIT. A POINTER TO THE LAST ITEM IN *
        *       THE EXPANDED EDIT SPAN IS RETURNED.                      *
        *                                                               *
        *****************************************************************
        *                                                               //
        DEFINE POINTER PROCEDURE CMPRSFE(EDT.STRT,EDT.STOP)
              WHERE POINTER EDT.STRT,EDT.STOP TOBE
                BEGIN                           ... ***** PROC CMPRSFE ******//
                OWN POINTER STRT;
                OWN POINTER STOP;
                STOP=EDT.STOP+ITM.SIZE;         ... LOAD OWN REGISTER        //
                STRT = EDT.STRT;
                IF SRCHFET(STRT-ITM.SIZE,LDMSK) LES STOP
                THEN STOP=SRCHFET(STOP-ITM.SIZE,CULMSK);  ... INCL DEP EVTS//
                CMPRSFE = STOP;                  ... RETURN NEW END POINT// 00000651
        LOOP:
                STRT = STRT-1;                  ... FROM POINTER             //
                STOP = STOP-1;                  ... TO POINTER               //
                IF STRT NEQ FESTRT              ... LIMIT CHECK              //
                THEN BEGIN                      ... ***** MOVE A WORD *******//
                     WD(STOP)=WD(STRT);         ... SINGLE WORD TRANSFER     //
                     GOTO LOOP;
                     END;                       ... ***** MOVE A WORD *******//
                FESTRT = FESTRT+(STOP-STRT);    ... CALC NEW TABLE START     //
                GOTO RETURN;
                END;                            ... ***** PROC CMPRSFE ******//
        ...
        *****************************************************************
        *                                                               *
        *                                FIXFET                         *
        *                                                               *
        *     FIXFET(LASTITM,NOITEMS)                                   * 00000668
        *                                                               *
        *       THIS ROUTINE REPAIRS THE FET FOLLOWING AN EDIT, AND IT   *
        *       REVALIDATES ALL ITEMS IN THE RWS. IT MAKES A SINGLE PASS *
        *       OVER THE FET. DURING THIS PASS, IT CALCULATES ALL VERTICAL *
        *       PATH ANGLES. WITHIN THE SPAN OF THE EDIT, IT CALCULATES  *
```

```
 *          LATERAL PATH ANGLES AND DISTANCE FROM FIRST EVENT IN THE    *
 *          FET. BEYOND THE SPAN OF THE EDIT, IT ADDS THE DISTANCE      *
 *          INCREMENT TO ALL LATERAL EVENTS. IF POFS HAS CHANGED, THE   *
 *          SPAN OF THE EDIT IS CHANGED TO INCLUDE ALL EVENTS. THUS ALL *
 *          FLIGHT PARAMETERS ARE RECALCULATED.                         *
 *                                                                      *
 ************************************************************************
 *                                                                     //
DEFINE PROCEDURE FIXFET(LASTITM,NOITEMS)                                   00000682
          WHERE  POINTER LASTITM;                                          00000683
                 INTEGER NOITEMS TODE
          BEGIN                                  ... ***** PROC FIXFET *******//
                                                 ... OWN REGISTERS           //
          OWN REAL TMP.REAL;
          OWN POINTER PRV.LE;
          OWN POINTER PRV.VE;
          OWN POINTER ITEM;
          OWN POINTER V.EVNT;
          OWN POINTER PNTR;
          OWN INTEGER I;
          OWN REAL DELTA;
          OWN INTEGER TMP.INTG;
          TMP.INTG $=$ TMP.REAL;                 ... SO WE CAN PACK REAL & INT//
                                                 ... LOCAL VARIABLES         //
          POINTER NAYBOR1,
                  NAYBOR2;
          REAL OLD.DIST,
               VE.DIST;
          REAL POFS.TMP;
          PRESET POFS.TMP = 0.0;
                                                 ... PROGRAM START           //
          FOR PNTR=STRT.RWS STEP 3               ... INVALIDATE ALL          //
              UNTIL STOP.RWS                     ... RWS ITEMS               //
          DO W1(PNTR)=W1(PNTR) .V. "80";
    INITL:
          IF TYPEFF(FESTRT) EQL CTYP        ... AUTO DELETE OF EOC EVENT //
          THEN FESTRT=ABS.1=FESTRT+ITM.SIZE; ... WHEN EOC IS EVENT 1 //
          ITEM=FESTRT;                           ... FESTRT ALWAYS LE OR OF.  //
          PRV.VE=V.EVNT=LOC.PTR=NULL;            ... INIT VE POINTERS.        //
          VE.DIST = 0.0;                         ... INIT DIST TO PREVIOUS VE //
          NAYBOR2 = SRCHFET(LASTITM,LDMSK);       ... END OF EDIT SPAN     //
                                                 ... START OF EDIT IS LAST ITEM
                                                     IN EDIT - ITEMS IN EDIT -
                                                     4. THE -4 INSURES THAT LE
                                                     PROCEEDING EDIT IS INCLUDED.
                                                     IT IS 4 ITEMS, EOC EVENT,
                                                     2 DEP EVENTS, LE ITSELF. //
          NAYBOR1=LASTITM-((NOITEMS+4)*ITM.SIZE);  ... START OF EDIT //
                                                 ... IF PARALLEL OFFSET (POFS)
                                                     HAS CHANGED WITH THIS
                                                     EDIT,THE SPAN OF THE EDIT
                                                     IS INCREASED TO INCLUDE
                                                     ALL ITEMS IN THE FET.
                                                     THIS MEANS ALL FLIGHT
                                                     PARAMS ARE RECALCULATED  //
          IF POFSFF NEQ POFS.TMP
          THEN BEGIN                             ... ***** POFS CHANGED *******//
```

```
                POFS.TMP = POFSFE;              ... UPDATE FIXFET POFS         //
                NAYBOR2 = FETEND;               ... GO TO END OF FET           //
                NAYBOR1 = FESTRT;               ... START AT FET START         //
                END;                            ... ***** POFS CHANGED ******//
           IF NAYBOR1 LEQ FESTRT                ... IF SPAN OF EDIT INCLUDES //
           THEN DISTFE(FESTRT)=0.0;             ... 1ST LE, REDEFINE DIST.   //
           GOTO DONE1;                          ... FINISH INIT OF FIRST EVNT//
                                                ... MAIN LOOP OF FIXFET      //
     NXT.ITM:
           IF ITEM=SRCHFET(ITEM.DLVMSK) EQL NULL    ... DEST, LE, VE       //
           THEN BEGIN                           ... ***** OUT OF ITEMS ******//
                IF C.BITS(FESTRT: .A. "80" NEQ ZERO   ... UNLESS CRS ED,//
                THEN CRSIFE(FESTRT)=CRSOFE(FESTRT);   ... INIT 1ST EVT //
                FOR PNTR=STRT.RWS STEP 3          ... ZERO ID IN ALL       //
                    UNTIL STOP.RWS                ... INVALID RWS ITEMS.   //
                DO   IF CTLWRD(PNTR) GRT "7F" ... MSB OF BYTE 0 SET        //
                     THEN WO(PNTR) = 0;           ... MEANS INVALID ITEM   //
                GOTO RETURN;
                END;                            ... ***** OUT OF ITEMS ******//
           IF TYPEFE(ITEM) EQL DTYP
                                                ... IF ITEM IS DEST EVENT, WE
                                                    LOOP THRU DONE1 TO CALC
                                                    VERT PATH ANGLES FOR THE
                                                    LAST VERT PATH LEG. THESE
                                                    WERE NOT CALCULATED IT THE
                                                    FET'S LAST EVENT WAS VERT//
           THEN BEGIN                           ... ***** DEST EVENT *****    //
                CRSOFE(PRV.LE)=0.0;             ... IF 1 EVENT, CRSI = 0      //
                GOTO DONE1;
                END;                            ... ***** DEST EVENT *****    //
                                                 .. PROCESS AN ITEM. IF VE
                                                    WAS FOUND, REMEMBER IT
                                                    UNTILL THE DIST CALC FOR
                                                    THE ASSOCIATED LE IS MADE//
           IF TYPEFE(ITEM) EQL VTYP
           THEN BEGIN                           ... ***** REMEMBER VE *******//
                V.EVNT = ITEM;                  ... STORE POINTER AT VE       //
                VPAOFE(V.EVNT)=0.0;             ... FIXES LAST VE IN FET      //
                VPAIFE(V.EVNT)=0.0;             ... FIXES FIRST VE IN FET     //
                GOTO NXT.ITM;                   ... LOOK SOME MORE            //
                END;                            ... ***** REMEMBER VE *******//
           IF ITEM LES NAYBOR1                  ... NAYBOR1 IS START OF EDIT //
                                                ... PRIOR TO SPAN OF EDIT NO
                                                    LE PARAMS ARE CALCULATED //
           THEN GOTO DONE;
           IF PRV.LE EQL NAYBOR2
           THEN BEGIN                           ... ****END OF EDIT SPAN****//
                TMP.REAL=DSTWRD(PRV.LE)+AOFSFE(PRV.LE);
                DSTWRD(PRV.LE)=TMP.REAL;  ... COMPLETE DIST FOR NAYBOR2//
                TYPEFE(PRV.LE)=LTYP;      ... RESTORE ITEM TYPE           //
                DELTA=TMP.REAL-OLD.DIST;  ... CALC INCREASED DIST.        //
                END;                            ... ****END OF EDIT SPAN****//
           IF ITEM GRT NAYBOR2                  ... NAYBOR2 HAS NEW LIST TOO //
           THEN BEGIN                           ... ***** BEYOND EDIT *****  //
                                                ... BEYOND SPAN OF EDIT,
                                                    INCREASED DIST IS ADDED
                                                    TO LE'S                  //
```

```
           DST.RD(ITEM)=DSTWRD(ITEM)+DELTA;   ... FIX DISTANCE.  //
           GOTO DONE;
           END;                         ... ***** BEYOND EDIT *****  //
                                        ... WITHIN SPAN OF EDIT, ALL
                                            FLIGHT PARAMS ARE
                                            CALCULATED                //
       IF RWSFE(PRV.LE) EQL ZERO OR     ... IF LAT/LONG INVALIDITY    //
          (CTLWRD(RWSPTR(PRV.LE)) .V.  ... BITS FOR EITHER POINT   // 00000849
          CTLWRD(RWSPTR(ITEM))) .A.    ... FIRST LINK IN CHAIN      // 00000850
          "30" NEQ 0                    ... OR PRV.LE IS ORIGIN EVNT.//
        THEN BEGIN                      ... ***** BYPASS LLTBD ******//
           DST.RD(ITEM)=DSTWRD(PRV.LE)+TENTH; ... DEFEAT 1/10 LIM//
           CRSIFE(ITEM)=CKSOFE(PRV.LE)    ... ALL LAT ANGLES =      //
                       =CKSIFE(PRV.LE);  ... LAST VALID ONE.        //
           IF EOC.PTR EQL NULL                 ... VPAI IS FLAGGED//
           THEN LOC.PTR=SMCHFET(ITEM,VMSK);    ... FOR VF PAST EOC//
           GOTO DONE;
           END;                         ... ***** BYPASS LLTBD ******//
        LLTBD(RWSPTR(PRV.LE)+1,RWSPTR(ITEM)+1,ANSBD); ... BRNG-DIST//
        TMP.REAL=RO(ANSBD);             ... CALC CRSO                //
        TMP.INTG=(TMP.INTG              ... SWITCH NAME OF OWN REG SC// 00000860
               .A. "FFFFFF00")          ... WE CAN PROTECT BYTE 3   // 00000861
               .V. C.BITS(PRV.LE);
        CRSOWD(PRV.LE)=TMP.REAL;
        IF C.BITS(ITEM) .A. "80" EQL ZERO
        THEN BEGIN                      ... ***** COURSE EDIT SET ***//
           IF ITEM EQL ABS.1            ... COURSE EDIT ALLOWED ON   //
           THEN GOTO CALCAOFS;          ... TO EVENT ONLY.            //
           C.BITS(ITEM)=C.BITS(ITEM) .V. "80";  ... CRSED INVALIC//
           END;                         ... ***** COURSE EDIT SET ***//
        TMP.REAL=R1(ANSBD)-DEG180;      ... CALC CRST                //
        TMP.INTG=(TMP.INTG
               .A. "FFFFFF00")
               .V. RWSFE(ITEM);         ... ADD BYTE 3                //
        CRSIWD(ITEM)=TMP.REAL;          ... STORE                     //
 CALCAOFS:
        TMP.REAL=AOFSFE(PRV.LE);        ... CALC AOFS                 //
        DSTWRD(PRV.LE)=DSTWRD(PRV.LE)+TMP.REAL;
        TYPEFF(PRV.LE)=LTYP;            ... REPAIR BYTE 3             //
        OLD.DIST=DSTWRD(ITEM);          ... SAVE OLD DIST FOR DELTA  //
        TMP.REAL=DSTWRD(PRV.LE)         ... STORE PARTIAL DIST. AOFS //
                 +R2(ANSBD)+TMP.REAL;   ... IS ADDED NEXT LE.         //
                                        ... THE FOLLOWING LOGIC CORRECTS
                                            THE DISTANCE VALUE FOR BASE
                                            OFFSETS. THE DIST VALUES FOR
                                            THE 1ST WAYPOINT OF THE
                                            OFFSET CONFIG & THE WAYPOINT
                                            PAST THE LAST OFFSET EVENT
                                            HAVE THE OFFSET VALUE
                                            ADDED.                    //
        IF OFTFG(ITEM) EQL STRT90       ... START OF 2-EVENT PATTERN  //
        OR OFTFG(ITEM) EQL STRTBASE     ... START OF 3-EVENT PATTERN  //
        OR OFTFG(PRV.LE) EQL END90      ... EVT PAST 2-EVENT PATTERN  //
        OR OFTFG(PRV.LE) EQL INTERCEP   ... EVT PAST 3-EVENT PATTERN  //
        THEN TMP.REAL=TMP.REAL+POFSFE;  ... ADD OFFSET TO DISTANCE   //
        DSTWRD(ITEM)=TMP.REAL;          ... PUT IN DIST FIELD       //
        IF RWSFE(ITEM) EQL RWSFE(PRV.LE)   ... REMOVE DUPLICATE EVTS//
```

```
        THEN BEGIN                          ... ***** DELETE 1 ITEM *****//
             ITEM=CMPRSFE(ITEM,ITEM);         ... OVERWRITE EVENT       // 00000900
             ADDSTRNG(ITEM,NULL,0,NULL); ... FIX FET POINTERS          //
             GOTO INITL;                    ... START FIXFET OVER       // 00000902
             END;                           ... ***** DELETE 1 ITEM *****//
DONE:
        TYPEFF(ITEM)=LTYP;
DONE1:
        IF V.EVNT NEQ NULL                  ... ANY VE ASSOC WITH PREV.LE//
        THEN BEGIN                          ... ***** VERTICAL EVENT ****//
             IF PRV.VE NEQ NULL             ... DON'T CALC FOR FIRST VE  //
             THEN BEGIN                     ... ***** VER PATH ANGS *****//
                  TMP.REAL=                 ... CALC VPAI               //
                  ATAN((ALTOFE(V.EVNT)-ALTOFE(PRV.VE))*ALTCONV,
                         DSTWRD(PRV.LE)-VE.DIST);  ... ARC TAN          //
                  TMP.INTG=(TMP.INTG                  ... CHANGE NAME OF //00000903
                            .A. "FFFFFF00")         ... OWN REG SO WE  // 00000904
                            .V. IBYTE3(V.EVNT);  ... CAN ADD BYTE 3 //
                  VPAIWD(V.EVNT)=TMP.REAL;
                  TMP.INTG=(TMP.INTG                  ... VPAO=VPAI       //00000907
                            .A. "FFFFFF00")                              00000908
                            .V. C.BITS(PRV.VE);  ... ADD BYTE 3       //
                 VPAOWD(PRV.VE)=TMP.REAL;            ... STORE            //
                  END;                      ... ***** VER PATH ANGS *****//
             PRV.VE = V.EVNT;               ... NEW VE BECOMES OLD VE    //
             VE.DIST=DSTWRD(PRV.LE);        ... SAVE DIST TO LAST VE.    //
             V.EVNT = NULL;                 ... SHOW NO VE PENDING.      //
             END;                           ... ***** VERTICAL EVENT ****//
        PRV.LE = ITEM;                      ... NEW LE BECOMES OLD LE    //
        I = RWSFE(ITEM);
                                            ... FOLLOW RWS CHAIN FM EACH
                                                LE VALIDATING ALL RWS
                                                ITEMS IN THE CHAIN      //
CHAIN:
        IF I EQL ZERO                       ... CHECK FOR END OF CHAIN   //
        THEN GOTO NXT.ITM;                  ... END OF CHAIN             //
        PNTR = STRT.RWS+(I*3)-3;            ... CALC RWS POINTER         //
        I = REFPTR(PNTR);                   ... PICK UP RWS ITEM NUM     //
        CTLWRD(PNTR)=CTLWRD(PNTR) .A. "7F";  ... MAKE ITEM VALID         //
        GOTO CHAIN;
        END;                                ... ***** PROC FIXFET *******//
...
*****************************************************************************
*                                                                           *
*                                RWSPTR                                     *
*    POINTER=RWSPTR(ITM,PTR)                                                *
*                                                                           *
*       THIS PROCEDURE CALCULATES AN RWS ITEM POINTER BASED                 *
*       UPON A POINTER TO A LATERAL EVENT.                                  *
*                                                                           *
*****************************************************************************
*                                                                          //
DEFINE POINTER RECURSIVE PROCEDURE RWSPTR(ITM,PTR)
        WHERE POINTER ITM,PTR TOHF
        BEGIN                               ... ***** PROC RWSPTR *******//
        RWSPTR = STRT.RWS+(RWSFE(ITM,PTR)*3)-3; ... ITM NUM TO PTR //
        GOTO RETURN;
```

```
            END;                                ... ***** PROC RWSPTR *******//
    ...
    ******************************************************************************
    *                                   OGDEINIT                                 *
    *       OGDEINIT();                                                          *
    *                                                                            *
    *           THIS PROCEDURE IS CALLED TO SET THE FET TO ORG-DEST.             *
    *                                                                            *
    ******************************************************************************
                                                                               //
    DEFINE PROCEDURE OGDEINIT TOBE
            BEGIN                               ... ***** PROC OGDEINIT *****//
            OWN POINTER PNTR;
            PNTR=FETENO-ITM.SIZE;               ... TWO EVTS. ORG & DEST.      //
            FESTRT = PNTR;                      ... RESET START OF FET PNTR    //
            ABS.1 = PNTR;                       ... ORG EVENT BECOMES EVNT 1 //
            C.BITS(PNTR) = "FD";                ... ORIGIN EVENT = TO EVENT    //
            TYPEFF(PNTR) = OTYP;                ... ITEM BEFORE DEST IS ORG    //
            RWSFE(PNTR)=ZERO;                   ... ORIGIN MUST HAVE NULL PTR//
            BCLEAR();                           ... CLEAR ALL OFFSET PARAMS    //
            ZEROF(STRT.RWS,STOP+RWS);           ... INIT THE RWS               //
            VERSFE=1;                           ... INIT VERSION NUMBER        //
            GOTO RETURN;
            END;                                ... ***** PROC OGDEINIT *****//
    ...
    ******************************************************************************
    *                                   GETITMFE                                 *
    *       RESULT = GETITMFE(TYPMASK,EVENO,WKPTR)                               *
    *                                                                            *
    *           GETITMFE SEARCHES THE FET FROM THE 'TO' EVENT COUNTING ITEMS*
    *           OF TYPE TYPMASK UNTIL EVENO HAVE BEEN COUNTED. THIS MARKS A *
    *           SUCCESSFUL SEARCH AND THE 'EVENO'TH ITEM IS RETURNED IN THE *
    *           WORK AREA POINTED AT BY WKPTR.                                   *
    *                                                                            *
    ******************************************************************************
    *                                                                          //
    DEFINE INTEGER RECURSIVE PROCEDURE GETITMFE(TYPMASK,EVENO,WKPTR)
            WHERE INTEGER TYPMASK,EVENO;
                  POINTER WKPTR TOBE
            BEGIN                               ... ***** PROC GETITMFE *****//
            OWN INTEGER CNTR;
            OWN POINTER PNTR;
            PNTR=ABS.1-ITM.SIZE;                ... POINT AT EVENT -1          //
            IF CNTR=EVENO EQL ZERO              ... NO SUCH ITEM AS EVENT 0  //
            THEN GOTO NOITEM;
            IF CNTR GRT ZERO                    ... WHICH DIR IS SEARCH?       //
            THEN BEGIN                          ... ***** FORWARD SEARCH ****//
    LOOP1:
                  IF PNTR=SRCHFE(PNTR,TYPMASK) ... USE SRCHFET TO FIND //
                        EQL NULL                 ... NEXT ITEM OF A TYPE.//
                  THEN GOTO NOITEM;             ... SEARCH FAILED.             //
                  IF CNTR=CNTR-1 NEQ 0          ... DECREMENT SUCCESS COUNTER//
                  THEN GOTO LOOP1;              ... NOT DONE. CONTINUE SEARCH//
                  GOTO FOUND;                   ... SEARCH SUCCESSFUL.         //
                  END;                          ... ***** FORWARD SEARCH ****//
    LOOP2:
            IF PNTR LES FESTRT                  ... CHECK LIMIT                //
```

```
          THEN BEGIN                    ... ***** SEARCH FAILED *****//
NOITEM:
               GETITMFE = NOEVT;        ... REPORT FAILURE.           //
               GOTO RETURN;
               END;                     ... ***** SEARCH FAILED *****//
          IF TYPMASK .A. EVMASK(TYPEFE(PNTR))     ... COMPARE TYPMASK//
               EQL 0                    ... WITH LOOKUP TABLE MASK    //
          THEN GOTO NXT.ITM;            ... NO MATCH                  //
          IF CNTR=CNTR+1 NEQ 0          ... FOUND ONE. BUMP COUNTER   //
          THEN BEGIN                    ... ***** BACK-UP POINTER ****//
NXT.ITM:
               PNTR=PNTR-ITM.SIZE;      ... DECREMENT POINTER         //
               GOTO LOOP2;              ... TRY AGAIN                 //
               END;
FOUND:
          GETITMFE=BUILDFE(PNTR,WKPTR); ... BUILD A WORK AREA         //
          GOTO RETURN;
          END;                          ... ***** PROC GETITMFE *****//
...
***********************************************************************
*                                                                     *
*                              GETNXAFE                               *
*                                                                     *
*     RESULT = GETNXAFE(TYPMASK,ABSEVNO,WKPTR)                        *
*                                                                     *
*         THIS ROUTINE FINDS THE FIRST EVENT OF THE TYPE INDICATED    *
*         BY TYPMASK AFTER THE EVENT SPECIFIED BY ABSEVNO. IF AN      *
*         EVENT IS FOUND, IT IS RETURNED IN THE WORK AREA SPECIFIED   *
*         BY WKPTR.                                                   *
*                                                                     *
***********************************************************************
*                                                                     //
DEFINE INTEGER RECURSIVE PROCEDURE GETNXAFE(TYPMASK,ABSEVNO,WKPTR)
          WHERE INTEGER TYPMASK,ABSEVNO;
                POINTER WKPTR TOBE
          BEGIN                         ... ***** PROC GETNXAFE *****//
          OWN POINTER PNTR;
          IF PNTR=SRCHFET(CONVFE(ABSEVNO),TYPMASK)
               EQL NULL                 ... SRCHFET DOES ALL THE WORK//
          THEN BEGIN                    ... ***** SEARCH FAILED *****//
               GETNXAFE=NOEVT;          ... ADMIT FAILURE             //
               GOTO RETURN;
               END;                     ... ***** SEARCH FAILED *****//
          GETNXAFE=BUILDFE(PNTR,WKPTR); ... SUCESS. BUILD WORK AREA   //
          GOTO RETURN;
          END;                          ... ***** PROC GETNXAFE *****//
...
***********************************************************************
*                                                                     *
*                              BUILDFE                                *
*     VALUE=BUILDFE(ITEM.PTR,WKPTR)                                   * 00000960
*                                                                     *
*         THIS ROUTINE BUILDS THE EXTERNAL WORK AREA FOLLOWING A      *
*         SUCCESSFUL FETCH.                                           *
*                                                                     *
***********************************************************************
*                                                                     //
```

```
DEFINE INTEGER RECURSIVE PROCEDURE BUILDFE(ITEM.PTR,WKPTR)
          WHERE POINTER ITEM.PTR,WKPTR TOBE
          BEGIN                          ... ***** PROC BUILDFE ******//
          OWN REAL RTNSIGN;
          OWN REAL CUCH;
          OWN POINTER TO.PTR;
          OWN POINTER LE.PTR;
          OWN POINTER ITM.PTR;
          OWN POINTER RWS.PTR;
          OWN INTEGER TEMP;
          OWN INTEGER INDX;
          OWN INTEGER SVE.VERS;
          INDX ==S CUCH;
                                         ... IF VERSFE=0, FET INHIRITED
                                             AND VERSION MISMATCH IS
                                             RETURNED.              //
          IF SVE.VERS=VERSFE EQL ZERO
          THEN GOTO NO.MATCH;
                                         ... INITIALIZATION SECTION    //
          TO.PTR=WKPTR;                  ... LOAD OWN REGISTERS        //
          ITM.PTR=LE.PTR=ITEM.PTR;       ... FET ITEM POINTER          //
          VERS(TO.PTR)=VERSFE;
          INDX=ITM.PTR-ABS.1;            ... CALC ABSOLUTE EVENT NUM  //
          IF INDX LEQ 0                  ... COMPENSATE FOR NO ZERO EV//
          THEN INDX=INDX+ITM.SIZE;
          ABEVNO(TO.PTR)=INDX/ITM.SIZE;  ... CONVERT PNTR TO ABS EVT NO//
          INDX=FCWTYP(TO.PTR)=TYPEFE(ITM.PTR);     ... PICK UP TYPE   //
          IF INDX LES HTYP               ... ORIGIN, DEST, AND LOC    //
          THEN BEGIN                     ... ***** INVALIDATE CNTRWORD//
               CNTRWORD(TO.PTR)=CNTRWORD(TO.PTR) .V. "FFFFFF";
               GOTO EXIT;
                END;                     ... ***** INVALIDATE CNTRWORD//
FIND.LE:
          IF TYPEFE(LE.PTR) NEQ LTYP     ... FIND LE ASSOC WITH EVENT //
          THEN BEGIN                     ... ***** LAT EVENT SEARCH **//
               LE.PTR=LE.PTR-ITM.SIZE;   ... BACK UP POINTER           //
               GOTO FIND.LE;             ... TRY AGAIN                 //
               END;                      ... ***** LAT EVENT SEARCH **//
         RWS.PTR=RWSPTR(LE.PTR);         ... POINT AT RWS ITEM         //
                                         ... LOAD LATERAL EVENT. USED
                                             AT ALL EVENTS.            //
                                         ... BUILD CNTRWORD. C.BITS HAS
                                             BITS 3-10. RWS ITEM HAS
                                             BITS 21-23. TYPE IS STORED
                                             IN BYTE 3. ALL OTHER BITS
                                             ARE SET TO VALID EXCEPT
                                             BIT 16 WHICH IS EAPOF AND
                                             BITS 12,13 FOR HOLD AND VER
                                             EVENTS WHICH COME FROM
                                             AUX.CTL FIELD.            //
          TEMP=(C.BITS(ITM.PTR) .LS. 3) .V.
                          (CILWRD(RWS.PTR) .LS. 17)                       00001102
                                .V. "50000";  ... EAOFS & EAPOF INVALID  // 00001134
          IF ITM.PTR EQL FESTRT AND PPSNST
                                         ... IF PRESENT POSITION IS NOT
                                             VALID, THE COURSE-IN OF THE
                                             FIRST EVENT IS INVALID.  //
```

```
                 THEN TEMP=TEMP .V. "8000";
                 CNTRWORD(TO.PTR)=TEMP;           ... STORE INVALIDITY BITS    //
                 ECWTYP(TO.PTR)=INDX;
                 EAWYID(TO.PTR) = NOT APTAG(TO.PTR);    ... ONE OR THE OTHER //
                 IF INDX NEQ LTYP
                 THEN AUX.CTL(TO.PTR)=
                      WO(ITM.PTR) .A. "30";
                 CPYRTN(TO.PTR.RWS.PTR);          ... COPY WYPT ID, LAT, LONG. //
                 AOFS(TO.PTR)=APOF(TO.PTR)=AOFSFE(ITM.PTR); ... CALC AOFS   // 00001151
                 DFTO(TO.PTR)=DSTWRD(LE.PTR)-DSTWRD(ABS.1);    ... CALC DFTO // 00001015
                 IF INDX EQL VTYP                                              00001005
                 THEN BEGIN                       ... ***** EXIT VE'S *****    // 00001006
                      CPYRTN(TO.PTR+8.ITM.PTR); ... VPAI,VPAO, AND ALTD       // 00001007
                      IF FOC.PTR EQL ITM.PTR   ... FIXFLT FLAGS VE AFTER FOC// 00001059
                      THEN EVPAI(TO.PTR)=TRUE; ... AS VPAI HERE IS INVALID. // 00001060
                      GOTO EXIT1;                                              00001138
                      END;                        ... ***** EXIT VE'S *****    // 00001009
                 CPYRTN(TO.PTR+B.LE.PTR);         ... CRSI, CRSO, AND AWYID    // 00001010
                 IASI(TO.PTR)=SPEED(LE.PTR);      ... RETURN IAS WORD          //
        EXIT1:                                                                 00001164
                 NOISE1(TO.PTR)=NOISE2(TO.PTR)=ZERO;  ... BLANK OUT GARBAGE //
                 POFS(TO.PTR)=POFSFE;             ... STORE FOR NOPOFS & PAPAL //
                 IF POFSF(TO.PTR)=PFSTAT EQL BASEOFS;
                 THEN BEGIN                       ... ***** BASE OFFSET *****  //
                      POFS(TO.PTR)=0*0;           ... FOR STRTBASE & STRT90    //
                      COCH=CRSOWD(LE.PTR)-CRSIWD(LE.PTR);  ... COURSE CHG    //
                      RTNSIGN=SIGN(COCH);                    ... SIGN OF CRS CHG//
                      IF TEMP=OFTFG(LE.PTR) EQL ENDBASE
                      THEN POFS(TO.PTR)=-POFSFE*RTNSIGN;         ... ENDBASE   //
                      IF TEMP EQL INTERCEP
                      THEN BEGIN                                 ... INTERCEP  //
                           POFS(TO.PTR)=
                                -POFSFE*RTNSIGN*NSIN(ABS(COCH));
                           AOFS(TO.PTR)=
                                -POFSFE*NCOS(ABS(COCH));
                           END;
                      IF TEMP EQL STRTBASE OR TEMP EQL STRT90  ... STRTBASE //
                      THEN AOFS(TO.PTR)=POFSFE;                   ... OR STRT90 //
                      POFSF(TO.PTR)=TEMP;         ... OVERWRITE FLAG COMPONENT //
                      END;                        ... ***** BASE OFFSET *****  //
        EXIT:
                 BUILDFE=GONTN;
                 IF VERSFE EQL SVE.VERS           ... IF VERSFE IS UNCHANGED,  //
                 THEN GOTO RETURN;                ... RETURN A STATUS OF GOOD. //
        NO.MATCH:                                 ... XTLK OR EDIT CHANGED VERS//
                 BUILDFE=NOVER;                   ... RETURN VERSION MISMATCH. //
                 GOTO RETURN;
                 END;                             ... ***** PROC BUILDFE ***** //
                 END;                             ... ***** PROCEDURES *****   //
                 END FINI                         ... ***** GLOBAL SPAN *******//

.PRT,S D.UTILF6AZ
```

```
4D.WORKFILE.UTILF6AZ
... *****************************************************************
*                                                                   *
*              3D/4D FLIGHT PLAN EDITOR UTILITY MODULE              *
*                                                                   *
*          THIS MODULE CONTAINS THE SPECIAL INTERFACE BETWEEN THE   *
*          FLIGHT PLAN EDITOR AND THE 3D/4D SYSTEM. IT CONTAINS     *
*          THE FOLLOWING PROCEDURES:                                *
*                                                                   *
*                    ADD4D          ADD EVENTS TO FLIGHT PLAN       *
*                                                                   *
*                    REPL4D         REPLACE EVENTS IN FLIGHT        *
*                                   PLAN & ENTER A BASE OFFSET      *
*                                                                   *
*                    ADDDIR         SPECIAL "DIRECT TO" INTERFACE   *
*                                   FOR GOING DIRECT TO ORIGIN      *
*                                   REPLACEMENT.                    *
*                                                                   *
*                    REPDIR         SPECIAL "DIRECT TO" INTERFACE   *
*                                   FOR USUAL "DIRECT TO".          *
*                                                                   *
*                    SCANBASE       FIND THE SPAN OF THE BASE       *
*                                   OFFSET IN THE FET.              *
*                                                                   *
*                    BCLEAR         CLEAR ALL BASE OFFSET FLAGS     *
*                                                                   *
*                    CK90           CALC ANGLE ERROR FOR W(I-1)     *
*                                                                   *
*                    TOTLERR        CHECK TOTAL DISTANCE ERROR      *
*                                                                   *
*                    ABSCOCH        CALC ABS VALUE OF COURSE        *
*                                   CHANGES FM W(X) TO W(Y).        *
*                                                                   *
*******************************************************************//
BEGIN                                   ... ***** GLOBAL SPAN ***** //
.INSERT          FPENF6I1;              ... FPE EXTERNAL INSERTS    //
.INSERT          FPENF6I2;              ... FPE INTERNAL INSERTS    //
.INSERT          DATAZ6I1;
... ************************** REALS ***************************** //
          REAL ARRAY
               ANGLER(2);               ... ANGLE ERROR ARRAY FOR TEST//

          PRESET ANGLER(0) TO ANGLER(2) = 0.0;
          EXTERNAL ANGLER;

          REAL ARRAY
               TDISTER(2);              ... TOT DIST ERROR FOR TESTING//

          PRESET TDISTER(0) TO TDISTER(2) = 0.0;
          EXTERNAL TDISTER;

          REAL
               R180,                    ... 180 DEGREES             //
               R90,                     ... 90 DEGREES              //
               R2,                      ... 2 DEGREES               //
               ONEVPI,                  ... 1/PI                    //
               RFET2;                   ... 0.2 NAUTICAL MILES      //
```

```
          EXTERNAL
               RNEWPI,
               R180,
               R90;
          PRESET BEGIN
               R2 = 0.011111111;          ... 2 DEGREES                //
               NMPT2 = 0.0000185111;      ... 2/10 NAUTICAL MILE       //
          END;
    ... ************************* POINTERS *****************************//
          POINTER
               BSTRT,                     ... 1ST OFFSET ITM FM SCANBASE//
               BSTOP,                     ... LAST OFFSET ITM          //
               ABS.1,                     ... POINTER TO FET ABS ITEM 1 //
               FESTRT,                    ... POINTER TO FIRST FET ITEM //
               FETEND;                    ... POINTER TO LAST  FET ITEM //
          EXTERNAL
               ABS.1,
               FESTRT,
               FETEND;
    ... *************************************************************
    *                                                               *
    *                          ADD4D                                *
    *                                                               *
    *          RTN.VLU = ADD4D(ABSEVNO,NOITEMS,PROC,WKPTR)          *
    *                                                               *
    ****************************************************************//

    DEFINE INTEGER PROCEDURE ADD4D(ABSEVNO,NOITEMS,PROC,WKPTR)
         WHERE INTEGER ABSEVNO,NOITEMS;
              POINTER PROC,WKPTR
         TOBE BEGIN                       ... ***** PROC ADD4D *****    //
              OWN POINTER PNT1;

              ADD4D=BADBASE;              ... SHOW OUR PESSIMISM        //
              IF SCANBASE() EQL GURTN
              THEN BEGIN                  ... ***** BASE OFFSET EXISTS *//
                   PNT1=CONVFE(ABSEVNO);  ... CONVERT TO FET ITEM PTR  //
                   IF PNT1 GEQ BSTRT AND  ... IF EDIT LIES WITHIN THE   //
                      PNT1 LEQ BSTOP      ... SPAN OF THE BASE OFFSET,  //
                   THEN GOTO RETURN;      ... EDIT IS NOT ALLOWED       //
                   END;                   ... ***** BASE OFFSET EXISTS *//
              IASF(WKPTR)=NOIAS;          ... CDU HAS NOT SET IAS FIELD.//
              COMPRF = VALID  $,          ... SET SPEED PROFILE FLAG    //
              ADD4D=ADDAIFE(ABSEVNO,NOITEMS,PROC,WKPTR);    ... PERMIT EDIT//
              GOTO RETURN;
              END;                        ... ***** PROC ADD4D *****    //
    ... *************************************************************
    *                                                               *
    *                          REPL4D                               *
    *                                                               *
    *    RTN.VLU = REPL4D(STABEVNO,SPABEVNO,NOITEMS,PROC,WKPTR)     *
    *                                                               *
    ****************************************************************//

    DEFINE INTEGER PROCEDURE REPL4D(STABEVNO,SPABEVNO,NOITEMS,PROC,WKPTR)
         WHERE INTEGER STABEVNO,SPABEVNO,NOITEMS;
```

```
            POINTER PROC, WKPTR
      TOBE BEGIN                            ... ***** PROC REPL4D *****  //

           OWN POINTER WPNTR;
           OWN POINTER D.STRT;
           OWN REAL ANG.ERR;
           OWN INTEGER SVERSFE;
           OWN INTEGER POFS.FLG;
           OWN INTEGER I;
           I $=$ POFS.FLG;

           FOR I=0 STEP 1 UNTIL 2
           DO   BEGIN                       ... ***** INIT ERROR ARRAYS **//
                TDISTER(I)=0.0;
                ANGLER(I)=0.0;
                END;                        ... ***** INIT ERROR ARRAYS **//
           REPL4D=BADBASE;                  ... FOREVER PESSIMISTIC       //
           WPNTR = WKPTR;                   ... PICK UP WORK POINTER       //
           POFS.FLG=POFSF(WPNTR);           ... PICK UP POFS FLAG         //
           D.STRT = CONVFE(STABEVNO);       ... CONVERT TO FET ITEM PTR  //
           IF POFS.FLG EQL BASEOFST
           THEN BEGIN                       ... ***** BASE OFFSET EDIT ***//
                IF SVERSFE=VERSFE EQL ZERO
                THEN BEGIN                  ... ***** FET INHIBITED ***** //
                     REPL4D=NOVER;
                     GOTO RETURN;
                     END;                   ... ***** FET INHIBITED ***** //
                IF ANG.ERR=ANGLER(0)=CK90(D.STRT) GRT D2 ... CK W(I-1) //
                THEN GOTO RETURN;           ... ERROR, RETURN BADBASE.   //
                IF TOTLERR(WPNTR,ANG.ERR,TDISTER(0))      ... CK DIST ERR//
                THEN GOTO RETURN;           ... ERROR, RETURN BADBASE.   //
                                            ... CALC ABS COURSE CHANGE
                                                FM W(I-1) TO W(I)         //
                ANG.ERR=ABS(ABSCOCH(1.0.D.STRT)-D180);
                ANGLER(1)=ANG.ERR;          ... SAVE ERROR FOR TESTING   //
                IF ANG.ERR GRT D2           ... CK ABS COCH ANGLE ERROR   //
                THEN GOTO TRIAD;            ... TOO BIG. CK SHALLOW GEOM. //
                IF TOTLERR(WPNTR,ANG.ERR,TDISTER(1))      ... CK DIST ERR//
                THEN GOTO TRIAD;                ... TOO BIG.
                                            ... ANGLE AT W(I) MUST BE 90
                                                DEG +/- 2 DEG             //
                IF ANG.ERR=ANGLER(2)=CK90(D.STRT+ITM.SIZE) ... CK90 CKS//
                   GRT D2                   ... W(I) INSTEAD OF W(I-1). //
                THEN GOTO TRIAD;            ... TOO MUCH ERROR            //
                IF TOTLERR(WPNTR,ANG.ERR,TDISTER(2))      ... CK DIST ERR//
                THEN GOTO TRIAD;            ... TOO BIG.                  //
                                            ... SATISFIES 2-WAYPOINT,
                                                BASE-OFFSET GEOMETRY      //
                RCLEAR();                   ... CLEAR FET OF OLD OFFSETS  //
                OFTFG(D.STRT)=END90;        ... FLAG W(I) AS END90       //
                OFTFG(PREV.LE(D.STRT))=STRT90; ... FLAG W(I-1) = STRT90//
   CLOSE:
                VERSFE=SVERSFE+1;           ... BUMP FET VERSION NUMBER  //
                PFSTAT=BASEOFST;            ... OFFSET STATUS             //
                POFSFE=POFS(WPNTR);         ... OFFSET VALUE              //
                FIXFET(STABEVNO,NOITEMS);    ... CALC VPAI,VPAO,AOFS
                                          AND OFTO COMPONENTS  //
```

```
                GOTO GREXIT;

                                          ... TWO-WAYPOINT BASE OFFSET
                                              GEOMETRY WAS NOT SATISFIED.
                                              THIS SECTION CHECKS THREE-
                                              WAYPOINT BASE OFFSET
                                              GEOMETRY.                    //
        THIAD:
                                          ... CALC ABS COURSE CHANGE ERR
                                              FM W(I) TO W(I+1).           //
                ANG.ERR=ABS(ABSCOCH(0,1,D.STRT)-D90);
                ANGLER(1)=ANG.ERR;        ... SAVE ERROR FOR TESTING       //
                IF ANG.ERR GRT D2
                THEN GOTO RETURN;         ... ERROR. RETURN BADBASE.       //
                IF TOTLERR(WPNTR,ANG.ERR,TDISTER(1))    ... CK DIST ERR//
                THEN GOTO RETURN;         ... ERROR. RETURN BADBASE.       //
                                          ... CALC ABS COURSE CHANGE ERR
                                              FM W(I-1) TO W(I+1).         //
                ANG.ERR=ABS(ABSCOCH(1,1,D.STRT)-D180);
                ANGLER(2)=ANG.ERR;        ... SAVE ERROR FOR TESTING       //
                IF ANG.ERR GRT D2         ... TWO-DEG ERR IS PERMITTED     //
                THEN GOTO RETURN;         ... ERROR. RETURN BADBASE.       //
                IF TOTLERR(WPNTR,ANG.ERR,TDISTER(2))    ... CK DIST ERR//
                THEN GOTO RETURN;         ... ERROR. RETURN BADBASE.       //
                                          ... 3-WAYPOINT BASE OFFSET
                                              GEOMETRY SATISFIED           //
                RCLEAR();                 ... CLEAR ANY OLD OFFSET         //
                OFTEG(PREV.LE(D.STRT))=STRTBASE; ... W(I-1) = STRTBASE//
                OFTEG(D.STRT)=ENDBASE;           ... W(I) = ENDBASE     //
                OFTEG(SRCHFET(D.STRT,LMSK))=INTERCEP;   ... W(I+1)      //
                GOTO CLOSE;
                END;                      ... ***** BASE OFFSET EDIT ***//

            IF POFS.FLG EQL PARALLEL
            OR POFS.FLG EQL NOPOFS
            THEN BEGIN                    ... ***** SIMPLE REPLACE **** //
                IF SCANBASE() EQL GORTN
                THEN BEGIN                ... ***** BASE OFFSET EXISTS *//
                                          ... DELETE MUST NOT TOUCH THE
                                              BASE OFFSET SPAN             //
                    IF D.STRT LEQ BSTOP
                    AND CONVFE(SPABEVNO) GEQ BSTRT
                    THEN GOTO RETURN;           ... RETURN BADBASE  //
                    GOTO REPL;            ... EDIT PERMITTED.              //
                    END;                  ... ***** BASE OFFSET EXISTS *//
                OFSTAT=POFS.FLG;          ... OFFSET STATUS                //
                POFSFE=POFS(WPNTR);       ... OFFSET VALUE                 //
        REPLS   IF CRSED(WKPTR)          ... IF THERE IS NOT A CKSEDIT  //
                THEN IASF(WKPTR)=NOIAS ; ... CDU HAS NOT SET IAS FIELD //
                COMPRF = VALID  $;        ... SET SPEED PROFILE FLAG       //
                REPLVD=REPLACFE(STAREVNO,SPABEVNO,NOITEMS,PROC,WKPTR);
                GOTO RETURN;
                END;                      ... ***** SIMPLE REPLACE *****//
            IF POFS.FLG EQL CANCEL
            THEN BEGIN                    ... ***** CANCEL OFFSETS *****//
                RCLEAR();
                POFSF(WKPTR)=NOPOFS ; ... SET OFFSET STATUS TO NO OFFST//
```

```
                REPL4D=REPLACFE(STABEVNO,SPABEVNO,NOITEMS,PROC,WKPTR) ;
                END ;                             ... ***** CANCEL OFFSETS ***** //
            GOTO RETURN ;                         ... IF EVT IS BASE OFFSET EVT  //
GDEXIT:         REPL4D=GDRTN;
                GOTO RETURN;
            END;                                  ... ***** PROC REPL4D *****   //
... ****************************************************************************
*                                                                              *
*                              ADDDIR                                          *
*                                                                              *
*         RTN.VLU = ADDDIR(ABSEVNO,NOITEMS,PROC,WKPTR)                         *
*                                                                              *
****************************************************************************//

DEFINE INTEGER PROCEDURE ADDDIR(ABSEVNO,NOITEMS,PROC,WKPTR)
     WHERE INTEGER ABSEVNO,NOITEMS;
            POINTER PROC,WKPTR
     TOBE BEGIN                                   ... ***** PROC ADDDIR *****   //
          IF OFTFG(ABS,1) NEQ NOPOFS
          THEN BCLEAR();                          ... CANCEL BASE OFFSET         //
          IASF(WKPTR) = NOIAS  $,                 ... SET IAS FLAG               //
          COMPRF = VALID  $,                      ... SET SPEED PROFILE FLAG     //
          ADDDIR=ADDATFE(ABSEVNO,NOITEMS,PROC,WKPTR); ... MAKE EDIT              //
          GOTO RETURN;
          END;                                    ... ***** PROC ADDDIR *****   //
... ****************************************************************************
*                                                                              *
*                              REPDIR                                          *
*    RTN.VLU = REPDIR(STABEVNO,SPABEVNO,NOITEMS,PROC,WKPTR)                    *
*                                                                              *
****************************************************************************//

DEFINE INTEGER PROCEDURE REPDIR(STABEVNO,SPABEVNO,NOITEMS,PROC,WKPTR)
     WHERE INTEGER STABEVNO,SPABEVNO,NOITEMS;
            POINTER PROC,WKPTR
     TOBE BEGIN                                   ... ***** PROC REPDIR ****** //
          IF SCANBASE() EQL GDRTN                  ... IF BASE OFFSET EXISTS//
          AND CONVFE(SPABEVNO) GEQ BSTRT           ... AND DELETE TOUCHES IT//
          THEN BCLEAR();                           ... THEN CLEAR BASE OFFST//
          COMPRF = VALID  $,                      ... SET SPEED PROFILE FLAG     //
          REPDIR=REPLACFE(STABEVNO,SPABEVNO,NOITEMS,PROC,WKPTR);
          GOTO RETURN;
          END;                                    ... ***** PROC REPDIR ****** //
... ****************************************************************************
*                                                                              *
*                              SCANBASE                                        *
*                                                                              *
*                       RTN.VLU = SCANBASE()                                   *
*                                                                              *
****************************************************************************//

DEFINE INTEGER PROCEDURE SCANBASE
       TOBE BEGIN                                 ... ***** PROC SCANBASE ***** //
          OWN INTEGER STATE;
          OWN POINTER LE.PTR;
          SYNONYMS  BEGIN
                    0 = INIT,
```

```
                  1 = FLIP;
                  END;
          STATE=INIT;                        ... INITIALIZE SWITCH          //
          LE.PTR = ABS.1-ITM.SIZE;           ... INIT SEARCH POINTER        //
          BSTRT = SRCHFET(LE.PTR,LMSK);      ... INITIALIZE B-OFFSET STRT  //
LOOP:
          LE.PTR = SRCHFET(LE.PTR.LMSK);  ... FIND NEXT LE IN FET          //
          IF LE.PTR EQL NULL
          THEN BEGIN                         ... ***** NO BASE OFFSET *****//
               SCANBASE=NOEVT;               ... RETURN NOEVT               //
               GOTO RETURN;
               END;                          ... ***** NO BASE OFFSET *****//
          IF STATE EQL INIT
          THEN BEGIN                         ... ***** FIND OFFSET STRT ***//
               IF OFTFG(LE.PTR) NEQ NOPOFS
               THEN BEGIN                    ... ***** FOUND OFFSET *****  //
                    STATE=FLIP;              ... FLIP THE SWITCH            //
                    GOTO LOOP;
                    END;                     ... ***** FOUND OFFSET *****  //
               BSTRT = LE.PTR;               ... RTN LAST NON-OFFSET ITMPTR//
               GOTO LOOP;
               END;                          ... ***** FIND OFFSET STRT ***//
          BSTOP = LE.PTR;                    ... RTN 1ST NON-OFFSET ITM PTR//
          IF OFTFG(LE.PTR) NEQ NOPOFS
          THEN GOTO LOOP;                    ... FIND END OF OFFSET         //
          SCANBASE = GDRTN;
          GOTO RETURN;
          END;                               ... ***** PROC SCANBASE ***** //
... ***************************************************************************
                                                                          *
                               BCLEAR()                                   *
                                                                          *
*************************************************************************//

DEFINE PROCEDURE BCLEAR
     TOSE BEGIN                              ... ***** PROC BCLEAR *****  //
          OWN POINTER PNTR;
          PNTR=FESTRT;                       ... INIT POINTER               //
LOOP:     IF PNTR LEQ FETEND
          THEN BEGIN                         ... ***** FET ITEM *****       //
               OFTFG(PNTR)=NOPOFS;           ... INIT OFFSET FIELD          //
               PNTR=PNTR+ITM.SIZE;           ... BUMP POINTER               //
               GOTO LOOP;
               END;                          ... ***** FET ITEM *****       //
          POFSFF=0.0;                        ... INIT OFFSET PARAMS         //
          PFSTAT=NOPOFS;
          GOTO RETURN;
          END;                               ... ***** PROC BCLEAR ****** //
... ***************************************************************************
*                              CK90                                       *
*                                                                         *
*                       RTN.VLU = CK90(FET.PTR)                           *
*                                                                         *
*************************************************************************//
                                             ... THIS ROUTINE CALCS THE DIFF
                                                 BETWEEN THE COCH AT LE PRIOR
                                                 TO FET.PTR & 90 DEGREES. //
```

```
DEFINE REAL PROCEDURE CK90(FET.PTR)
     WHERE POINTER FET.PTR
     TOBE BEGIN                              ... ***** PROC CK90 ******   //
          OWN POINTER PNTR;
          PNTR=PREV.LE(FET.PTR);             ... FIND PREV LAT EVENT       //
          IF PNTR EQL NULL
          THEN BEGIN                         ... ***** NO PREV LE ******  //
               CK90=D90;                     ... SHOW MAX ERROR            //
               GOTO RETURN;
               END;                          ... ***** NO PREV LE ******  //
          CK90=ABS(ABS(CRSOWD(PNTR)-CRSIWD(PNTR))-D90);
          GOTO RETURN;
          END;                               ... ***** PROC CK90 ******   //
... ********************************************************************
*                                 ABSCOCH                              *
*                                                                      *
*                   RTN.VLU = ABSCOCH(I.J.FET.PTR)                     *
*                                                                      *
*********************************************************************//
                                             ... THIS ROUTINE CALCS THE
                                                 ABSOLUTE VALUE OF THE SUM
                                                 OF COURSE CHANGES FROM
                                                 I LE'S PRIOR TO FET.PTR
                                                 TO J LE'S AFTER FET.PTR. //
DEFINE REAL PROCEDURE ABSCOCH(I.J.FET.PTR)
     WHERE INTEGER I.J;
           POINTER FET.PTR
     TOBE BEGIN                              ... ***** PROC ABSCOCH ***** //
          OWN POINTER PTR;
          OWN INTEGER X;
          OWN INTEGER Y;
          OWN REAL SUM;
          ABSCOCH=SUM=0.0;                   ... INIT REAL VALUES          //
          PTR = FET.PTR;                     ... INIT OWN POINTER          //
          FOR X=0 STEP 1 UNTIL I-1           ... FIND STRT OF EVT STRING  //
          DO   PTR=PREV.LE(PTR);

                                             ... SUM COURSE CHANGES        //

          X = I + J;                         ... TOTAL NUM OF LE'S TO SUM  //
          FOR Y=0 STEP 1 UNTIL X
          DO   BEGIN                         ... ***** SUM COCH *****      //
               IF PTR EQL NULL               ... PTR=NULL IF NO PREV LE    //
               THEN GOTO RETURN;             ... RTN 0.0 SO GEOM TEST FAILS//
               SUM=SUM + (CRSOWD(PTR)-CRSIWD(PTR));
               PTR=SRCHFET(PTR.LMSK);        ... SEARCH FOR NEXT LE        //
               END;                          ... ***** SUM COCH *****      //
          ABSCOCH=ABS(SUM);                  ... RETURN ABSOLUTE VALUE     //
          GOTO RETURN;
          END;                               ... ***** PROC ABSCOCH ***** //
... ********************************************************************
*                                                                      *
*                                 PREV.LE                              *
*                                                                      *
*                       PTR = PREV.LE(FET.PTR)                         *
*                                                                      *
*********************************************************************//
```

```
                                                  ... THIS PROC FINDS THE LF
                                                      PRIOR TO THE ONE POINTED
                                                      AT IN THE FET. THUS IT IS
                                                      A BACKWARD SEARCH ROUTINE.//
DEFINE POINTER PROCEDURE PREV.LE(FET.PTR)
    WHERE POINTER FET.PTR
    TOBE BEGIN                                    ... ***** PROC PREV.LE ***** //
         OWN POINTER PNTR;
         PREV.LE = NULL;
         PNTR = FET.PTR;                          ... INIT REGISTER             //
LOOP:    PNTR=PNTR-ITM.SIZE;                      ... BACK-UP ONE ITEM          //
         IF PNTR LES FESTRT                       ... LIMIT CHECK               //
         THEN GOTO RETURN;                        ... RETURN NULL               //
         IF TYPEFF(PNTR) NEQ LTYP                 ... CHECK FOR LATERAL EVENT   //
         THEN GOTO LOOP;                          ... NO HIT                    //
         PREV.LE = PNTR;                          ... HIT                       //
         GOTO RETURN;
         END;                                     ... ***** PROC PREV.LE ***** //
... *********************************************************************
*                                                                       *
*                              TOTLERR                                  *
*                                                                       *
*                  BOOL = TOTLERR(WKPTR,ANG.ERR)                        *
*                                                                       *
*********************************************************************//
                                                  ... THIS ROUTINE TESTS THE
                                                      TOTAL DISTANCE ERROR ASSOC
                                                      WITH A GIVEN ANGULAR ERROR//
DEFINE BOOLEAN PROCEDURE TOTLERR(WKPTR,ANG.ERR,ERROR)
     WHERE POINTER WKPTR;
         REAL ANG.ERR, ERROR
    TOBE BEGIN
                                                  ... ***** PROC TOTLERR ******//
                                                  ... DIV BY 1/PI TO CONVERT
                                                      ANG.ERR TO RADIANS.       //
         TOTLERR = TRUE;
         IF ERROR=ABS(POFS(WKPTR)*(ANG.ERR/ONEVPI)) LEQ NMPT2
         THEN TOTLERR = FALSE;                    ... ERROR LEQ 0.2 NM          //
         GOTO RETURN;
         END;                                     ... ***** PROC TOTLERR ******//
END FINI                                          ... ***** GLOBAL SPAN ****** //
```

APPENDIX B

Program Listing of the Speed Profile Algorithm

```
@D,WORKFILE.TALG26A2
        BEGIN
        .INSERT FREDF611 ;
        .INSERT STRCTISY ;
        .INSERT CONTUIPS ;
        .INSERT DATA2611 ;
        .INSERT AEHLIB ;
        .INSERT MATHPACK ;
        .INSERT NTAPE611 ;


        COMMENT
                             ***** EXTERNAL DATA STRUCTURE *****
        1)  FREDF611 INSERT FILE
                SYMBOLIC NAME  DECLARATION    DESCRIPTION
                               TYPE

                REPLACFE       INTEGER        FLIGHT PLAN EDITOR ROUTINE THAT
                                 PROCEDURE      PERFORMS FET MAINTENANCE FOR
                                                FLIGHT PLAN EDITS THAT REPLACE
                                                A SPECIFIED SET OF EXISTING
                                                FLIGHT PLAN ENTRIES WITH A
                                                SECOND SET OF ENTRIES.
                GETNXAFE       INTEGER        FLIGHT PLAN EDITOR ROUTINE THAT
                                 PROCEDURE      RETRIEVES EVENT (N+1)OF TYPE M
                                                WHERE N AND M ARE SPECIFIED
                                                AS INPUT ARGUMENTS.
                GETITMFE       INTEGER        FLIGHT PLAN EDITOR ROUTINE THAT
                                 PROCEDURE      RETRIEVES EVENT N OF TYPE M
                                                WHERE N AND M ARE SPECIFIED
                                                AS INPUT ARGUMENTS.
                GDRTN          SYNONYM        ARGUMENT RETURNED BY THE FLIGHT
                                                PLAN EDITOR THAT INDICATES
                                                THAT THE DESIRED FLIGHT PLAN
                                                REVISION IS SUCCESSFUL.
                LMSK           SYNONYM        MASK USED TO SPECIFY FLIGHT PLAN
                                                EDITOR OPERATIONS THAT ARE TO
                                                BE PERFORMED FOR A LATERAL
                                                EVENT.
                AMSK           SYNONYM        MASK USED TO SPECIFY FLIGHT PLAN
                                                EDITOR OPERATIONS THAT ARE TO
                                                BE PERFORMED INDEPENDENTLY OF
                                                EVENT TYPE.
                LTYP           SYNONYM        CODE USED TO DESIGNATE LATERAL
                                                TYPE FOR THE ECATYP COMPONENT.
                VTYP           SYNONYM        CODE USED TO DESIGNATE VERTICAL
                                                TYPE FOR THE ECATYP COMPONENT.
                ECATYP         INTEGER        FLIGHT PLAN EDITOR FLIGHT PLAN
                                 COMPONENT      COMPONENT USED TO RECORD EVENT
                                                TYPE.
                ABEVNO         INTEGER
                                 COMPONENT    FLIGHT PLAN EDITOR FLIGHT PLAN
                                                COMPONENT USED TO RECORD
                                                ABSOLUTE EVENT NUMBER.
                [illegible]    REAL           FLIGHT PLAN EDITOR FLIGHT PLAN
                                 COMPONENT      COMPONENT USED TO RECORD
                                                ALONG-TRACK DISTANCE FROM THE
```

```
                                          SPECIFIED WAYPOINT TO THE
                                            TO-WAYPOINT.
           ALTD          REAL            FLIGHT PLAN EDITOR FLIGHT PLAN
                           COMPONENT       COMPONENT USED TO RECORD
                                            ALTITUDE FOR A VERTICAL
                                            EVENT.
           IASR          REAL            FLIGHT PLAN EDITOR FLIGHT PLAN
                           COMPONENT       COMPONENT USED TO RECORD
                                            INDICATED AIRSPEED UNITS MADE
                                            VIA THE SPEED PAGE.
           IASF          INTEGER         FLIGHT PLAN EDITOR FLIGHT PLAN
                           COMPONENT       COMPONENT USED TO RECORD IAS
                                            TYPE.
           PIAS          SYNONYM         CODE RETURNED IN THE IASF
                                            COMPONENT THAT INDICATES A
                                            PILOT-DEFINED IAS.
           PRDUOTF       SYNONYM         CODE RETURNED IN THE IASF
                                            COMPONENT THAT INDICATES THAT
                                            THE IAS ENTRY FOR THE GIVEN
                                            WAYPOINT IS IDENTICAL TO THAT
                                            OF THE PRECEDING WAYPOINT IN
                                            THE WAYPOINT.
           PRDOWN        SYNONYM         CODE RETURNED IN THE IASF
                                            COMPONENT THAT INDICATES THAT
                                            THE GIVEN IAS IS PART OF A
                                            DECREASING PROFILE.
           PRUP          SYNONYM         CODE RETURNED IN THE IASF
                                            COMPONENT THAT INDICATES THAT
                                            THE GIVEN IAS IS PART OF AN
                                            ASCENDING PROFILE.
     3)  CONTUIPS INSERT FILE
           SYMBOLIC NAME  DECLARATION    DESCRIPTION
                          TYPE

           WKA1          POINTER         CONTAINS ADDRESS OF WORK AREA
                                            USED BY THE CDU CHANNEL.
           ZILCH         INTEGER         NULL PROCEDURE USED TO SATISFY
                           PROCEDURE       PROCEDURE ARGUMENT REQUIREMENTS
                                            FOR CASES WHERE SPECIAL DEFAULT
                                            LOGIC IS NOT REQUIRED.
     2)  STMCTISY INSERT FILE
           SYMBOLIC NAME  DECLARATION    DESCRIPTION
                          TYPE

           MAX64         REAL            MAXIMUM VALUE THAT MAY BE
                                          ASSIGNED TO A REAL PARAMETER.
     4)  DATA2611 INSERT FILE
           SYMBOLIC NAME  DECLARATION    DESCRIPTION
                          TYPE

           COMPFF        INTEGER         VALIDITY FLAG THAT IS ASSIGNED
                                            A VALID STATUS BY THE FLIGHT
                                            PLAN MONITOR WHENEVER A FLIGHT
                                            PLAN REVISION HAS OCCURRED.
                                            SPCORE ASSIGNS AN INVALID
                                            STATUS TO COMPFF WHENEVER SPEED
                                            PROFILE COMPUTATIONS ARE
```

```
                                          PERFORMED.
              VALID          SYNONYM     VALUE ASSIGNED TO INTEGER FLAGS
                                           TO INDICATE VALID STATUS.
              TCONF          INTEGER     VALILITY FLAG THAT IS ASSIGNED
                                           A VALID STATUS BY SPDPRF
                                           WHENEVER SPEED PROFILE
                                           COMPUTATIONS ARE UPDATED.
                                           TCONF IS ASSIGNED AN INVALID
                                           STATUS BY THE TIME CONTROL
                                           PROCEDURE WHENEVER TIME
                                           CONTROL COMPUTATIONS ARE
                                           PERFORMED.
      5)  AFDLIB INSERT FILE
              SYMBOLIC NAME  DECLARATION    DESCRIPTION
                             TYPE

              .CKPT          PROCEDURE   SUPERVISORY SOFTWARE ROUTINE
                                           THAT IS CALLED TO YIELD OWN
                                           CHANNEL TIME FOR CURRENT
                                           CYCLE.
      6)  MATHPACK INSERT FILE
              SYMBOLIC NAME  DECLARATION    DESCRIPTION
                             TYPE

              RLMTF          REAL        ROUTINE THAT LIMITS THE FIRST
                               PROCEDURE   ARGUMENT TO THE MAGNITUDE
                                           GIVEN BY THE SECOND ARGUMENT.
              RMAXF          REAL
                               PROCEDURE ROUTINE WHOSE RETURN ARGUMENT
                                           IF THE LARGER OF TWO VALUES
                                           SPECIFIED AS INPUT ARGUMENTS.
              RMINF          REAL
                               PROCEDURE ROUTINE WHOSE RETURN ARGUMENT
                                           IS THE SMALLER OF TWO VALUES
                                           SPECIFIED AS INPUT ARGUMENTS.
              ABS            REAL        PROCEDURE THAT DETERMINES THE
                               PROCEDURE   MAGNITUDE OF A GIVEN INPUT
                                           ARGUMENT.
              RROOT2         REAL        PROCEDURE THAT DETERMINES THE
                               PROCEDURE   SQUARE ROOT OF A GIVEN INPUT
                                           ARGUMENT.
      7)  NTABN6I1 INSERT FILE
              SYMBOLIC NAME  DECLARATION    DESCRIPTION
                             TYPE

              ADHD           POINTER     POINTER TO ARRAY WHERE SENSOR
                                           INPUT FOR HEADING AND AIR DATA
                                           INFORMATION IS RECORDED.
              TAS            REAL        ADHD COMPONENT USED TO RECORD
                               COMPONENT   SENSOR INPUT FOR TRUE AIRSPEED.
              DELV           REAL        ADHD COMPONENT USED TO RECORD
                               COMPONENT   SENSOR INPUT FOR IAS INSTRUMENT
                                           ERROR.
              HZHZ           REAL        REAL PARAMETER USED TO RECORD
                                           PRESENT ESTIMATE OF AIRCRAFT
                                           ALTITUDE.
```

```
                         ***** INTERNAL DATA STRUCTURE *****
                                                                      ;

        COMMENT          INTEGER DECLARATIONS                         ;

        INTEGER
        FRSTIASF,                    ... STATUS OF IASF COMPONENEN FOR WYPT
                                           TAGGED AS FRST FOR A GIVEN PROFILE   //
        ABEVJ,                       ... ABS EVNT NO. OF JTH LAT EVENT          //
        FROMEVNO,                    ... SPECIFIES LATERAL FROM-EVENT NUMBER    //
        LEVNTJ,                      ... SPECIFIES LATERAL EVENT NUMBERS        //
        SAVE ,                       ... IAS DISPLAY CLEAR PARAMETER            //
        LCNT,                        ... FLAG TO INDICATE IF ADJACENT WYPTS
                                         HAVE PIAS                              //
        ONEITM,                      ... ONE ITEM IS  TO BE REPLACED            //
        FRST,                        ... EVENT NO. FOR INITIAL SPEED CONTROL
                                           WAYPOINT OF A GIVEN PROFILE          //
        LAST,                        ... EVENT NO. INDEX FOR FINAL SPEED-CONTROL
                                         WAYPOINT OF A GIVEN PROFILE            //
        FRST.STATUS ;                ... PARAMETER USED TO RECORD EXISTENCE OF
                                           ANOTHER SPEED PROFILE IN THE REMAINING
                                           FLIGHT PLAN                          //


        COMMENT          REAL DECLARATIONS                            ;

        REAL
        ACTAS,                       ... AIRCRAFT TRUE AIR SPEED                //
        ACIASERR,                    ... IAS INSTRUMENT ERROR                   //
        ACALT,                       ... AIRCRAFT ALTITUDE                      //
        HZ,                          ... AIRCRAFT OR WYPT ALTITUDE              //
        FRSTIASR,                    ... SAVES THE IAS VALUE OF W(FRST)         //
        LASTIASR,                    ... SAVES THE IAS VALUE OF W(LAST)         //
        DFTOFRST,                    ... DIST FROM THE TO-EVENT TO W(FRST)      //
        DFTOLAST,                    ... DIST FROM THE TO-EVENT TO W(LAST)      //
        PREVDFTO,                    ... DIST BETWEEN TO-EVENT AND W(J-1)       //
        DFTOJ  ,                     ... DIST BETWEEN TO-EVENT AND W(J)         //
        PREVIASR,                    ... IAS OF W(J-1)                          //
        DISTJ,                       ... DIST BETWEEN W(FRST) AND W(J)          //
        JIASR,                       ... TEMP VAR TO HANDLE IAS CALC OF W(J)    //
        KALT,                        ... ALTITUDE PROPORTIONALITY CONSTANT
                                         USED TO CONVERT IAS TO TAS             //
        KALTSCALE,                       ... KALT SCALING FACTOR                //
        PRVISR2,                     ... PREVIASP SQUARED                       //
        PROFDST,                     ... ALONG TRK DIST FROM W(FRST) TO W(LAST) //
        DVEL,                        ... VEL CHNG BETWEEN W(J-1) AND W(LAST)    //
        DJ,                          ... ALONG TRK DIST BETWEEN W(J-1)AND W(J)  //
        DELDIST,                     ... ALONG TRK DIST BETWEEN W(J) AND W(LAST)//
        T1,                          ... TIME TO FLY FROM W(J-1) TO W(LAST)
                                         AT RATE OF MINRATE                     //
        AVEV,                        ... AVE VEL REQUIRED FOR A LINEAR SPEED
                                         PROFILE BETWEEN W(J-1) AND W(LAST)     //
        T2,                          ... TIME TO FLY FROM W(J-1) TO W(LAST)
                                         WITH CONSTANT VELOCITY OF AVEV         //
        LTP ;                        ... DIST TO REALIZE VEL CHNG OF DVEL AT
                                         RATE OF MINRATE                        //
```

```

COMMENT              POINTER DECLARATIONS                                 ;

POINTER
WAPT ;                           ... WORK AREA POINTER                    //



COMMENT              INTERNAL PRESET DECLARATIONS                         ;

PRESET
BEGIN
FROMEVNO=-1 ;
ONEITM=1 ;
LCNT=0 ;
FRST=0 ;
LAST=0. ;
FRST.STATUS=0 ;
ACTAS=0. ;
ACIASERR=0. ;
ACALT=0. ;
TK=0. ;
FRSTIASR=0. ;
LASTIASR=0. ;
DFTOFRST=0. ;
DFTOLAST=0. ;
PREVDFTO=0. ;
DFTOJ=0. ;
KALT=0. ;
KALTSCALE=.1 ;
PRVISP2=0.  ;
PROFDST=0. ;
DVEL=0. ;
T1=0. ;
AVEV=0. ;
T2=0. ;
END ;
DEFINE PROCEDURE SPDPRF
TOKE
BEGIN


COMMENT
                PROCEDURE SPDPRF PROVIDES SOFTWARE IMPLEMENTATION OF THE
                30/40 TASK II SPEED PROFILE ALGORITHM.  SPEED PROFILE
                COMPUTATIONS ARE PERFORMED FOR ALL LATERAL EVENTS THAT ARE
                NOT CONSIDERED TO BE SPEED-CONTROL WAYPOINTS (WAYPOINTS FOR
                WHICH SPEED HAS BEEN ENTERED VIA THE SPEED PAGE).
                THE LATERAL FROM-EVENT IS TREATED AS A SPEED-CONTROL WAYPOINT
                TO ENSURE THAT A SPEED PROFILE PRESENTLY BEING FLOWN WILL NOT
                CHANGE AT WAYPOINT PASSAGE.-
                RESULTS OF PROFILE COMPUTATIONS ARE RECORDED IN THE IASR/IASI/
                IASF COMPONENTS FOR LATERAL EVENTS AS MAINTAINED IN THE
                FLIGHT EVENT TABLE (FET) BY THE FLIGHT PLAN EDITOR.               ;

            OWN POINTER WAPT ;
            [illegible]
```

```
      OWN REAL PREVIASR,DJ,DPP,DELDIST,JIASR,DISTJ ;
      IF COMPRF NEQ VALID          ... IF FLIGHT PLAN HAS BEEN MODIFIED
                                       SINCE THE LAST SPEED PROFILE
                                       COMPUTATION, THEN UPDATE SPEED
                                       PROFILE, OTHERWISE RETURN,        //
         THEN GOTO VERUPDATE ;
      WKPT=WKA1 ;                  ... INITIALIZE EVENT TABLE WORK AREA   //
      TCONF=VALID ;                ... SET FLAG TO EXECUTE TIME CONTROL   //


COMMENT  SCAN FLIGHT PLAN FOR LAST SPEED PROFILE WAYPOINT.  ASSIGN
         THE IAS ASSOCIATED WITH THIS WAYPOINT TO ALL REMAINING
         WAYPOINTS IN THE FLIGHT PLAN                                     ;

      SAVE=1 ;                     ... THE FIRST SAVE IS THE 1ST TO-EVENT //
      LEVNTJ=FROMEVNO+2 ;
      WHILE GETITMFF(LMSK,LEVNTJ,WKPT) EQL BEGIN
         DO
         BEGIN
         IF IASF(WKPT) EQL PIAS
            THEN
            BEGIN
            LASTIASR=IASR(WKPT) ;  ... RECORD LAST PILOT-DEFINED IAS
                                       IN LASTIASR                        //
            SAVE=LEVNTJ+1 ;
            END ;
         LEVNTJ=LEVNTJ+1 ;
         END ;
      LEVNTJ=SAVE ;                ... QUOTE ALL SPEEDS FOR WAYPOINTS AFTER
                                   THE LAST PIAS DEFINED WYPT             //
      WHILE GETITMFF(LMSK,LEVNTJ,WKPT) EQL BEGIN
         DO
         BEGIN
         IASR(WKPT)=LASTIASR ;  ... ENTER LAST PIAS SPEED FOR WAYPT SPD//
         IASF(WKPT)=PRQUOTE ;   ... AND DISPLAY "QUOTES"                  //
         IF REPLACFF(ABEVNO(WKPT),ABEVNO(WKPT),ONEITM,ZTLCM,WKPT)
            NEQ BEGIN
            THEN GOTO VERUPDATE ;
         LEVNTJ=LEVNTJ+1 ;
         END ;


COMMENT  TAKE OWN SNAPSHOT OF CURRENT SENSOR DATA                         ;

      .CKPT() ;                    ... YIELD CHANNEL TIME TO ENSURE TIME
                                      TO GET A GOOD SNAPSHOT OF SENSOR DTA//
      ACTAS=TAS(ADHD) ;            ... AIRCRAFT TRUE AIR SPEED            //
      ACIASERR=DELV(ADHD) ;        ... IAS INSTRUMENT ERROR
                                                                          //
      HZ=ACALT=HZHZ ;              ... INITIALIZE ALTITUDE TO AIRCRAFT
                                       ALTITUDE                           //


COMMENT  RETRIEVE FLT PLAN PARAMETERS.  IF FROM-WAYPOINT HAS AN IAS
         TAG IT AS PILOT-DEFINED                                          ;

      IF GETITMFF(LMSK,FROMEVNO,WKPT) NEQ BEGIN
```

```
            THEN GOTO VERUPDATE ;
         FRST=-1 ;                     ... INITIALIZE FRST TO -1             //
         LEVNTJ=+1 ;                   ... BEGIN SEARCH FOR PIAS WITH TO-EVNT //
         IF FRSTIASF=IASF(WKPT) NEQ NOIAS
                                       ... IF FROM-EVNT HAS AN AIRSPEED
                                           ASSOCIATED WITH IT                //
            THEN FRSTIASF=PIAS         ... THEN MAKE IT A PIAS               //


   COMMENT  FROM-WAYPOINT HAS NO IAS.  SCAN FLT PLAN FOR FIRST PILOT-
            ENTERED IAS AND ASSIGN THIS TO THE FROM-WAYPOINT                 ;

            ELSE
            BEGIN
            WHILE GETITMFF(LASK,LEVNTJ,WKPT) EQL GORTN
                  AND FRSTIASF=IASF(WKPT) NEQ PIAS
               DO LEVNTJ=LEVNTJ+1 ;
            END ;
         IF FRSTIASF NEQ PIAS          ... IF NO PIAS WAS FOUND ,RETURN      //
            THEN GOTO VERUPDATE ;
         FRSTIASR=IASR(WKPT) ;         ... RETAIN THE FOUND TASR .  THIS IS TO
                                           REPLACE THE FROM-EVENT IASR       //
         IF GETITMFF(LASK,FROMEVNO,WKPT) NEQ GORTN
                                       ... GET FROM-EVENT INTO WORK AREA     //
            THEN GOTO VERUPDATE ;


   COMMENT  UPDATE FLIGHT PLAN TABLE FOR THE FROM-WAYPOINT.  INITIALIZE
            PARAMETERS FOR FIRST SPEED PROFILE COMPUTATION                   ;

         IASR(WKPT)=FRSTIASR ;         ... ASSIGN SPEED TO FROM-EVENT        //
         IASF(WKPT)=PIAS ;             ... ASSIGN PIAS TO FROM-EVENT         //
         IF REPLACEE(AREVFU(WKPT)+AREVNO(WKPT),OUTITM,ZIICH,WKPT) NEQ GORTN
            THEN GOTO VERUPDATE ;
         DFTOFRST=DFTO(WKP) ;          ... DEFINED IN                        //
         DFTOLAST=DFTOFRST ;           ... THE REAL                          //
         LASTIASR=IASR(WKPT) ;         ... DECLARATIONS ABOVE                //
         LEVNTJ=+1 ;                   ... START SEARCH FOR NEXT ENTERED SPEED
                                           WITH EVENT AFTER TO-EVENT         //


   COMMENT FRST VALID LOOP.
              INITIALIZE PARAMETERS FOR PREVIOUS PROFILE                     ;

         FRST.STATUS=VALID ;           ... VALIDATE FRST.STATUS  CONDITION   //
         WHILE FRST.STATUS EQL VALID
                                       ... THIS LOOP CONSTRUCTS EACH PROFILE.
                                           FIRST IT MUST FIND THE NEXT SPEED
                                           CONTROL WAYPOINT.                 //
         DO
         BEGIN
         PREVDFTO=DFTOLAST ;
         PREVIASR=LASTIASR ;
         ICNT=0 ;                      ... SET ADJACENT SPEED CONTROLS CHECK //


   COMMENT  NEXT SPEED LOOP
```

```
             SCAN FLIGHT PLAN FOR LAST WAYPOINT OF GIVEN PROFILE.                    ;

   GETNEXT.SPEEDWYPT :
         IF GETITMFF(LASK,LEVNTJ,WKPT) NEQ GORTN  ... GET NEXT EVENT           //
            THEN GOTO VERUPDATE ;
            IF IASF(WKPT) NEQ PIAS ... YES.IS IT A SPEED CONTROL WYPT ?  //
               THEN
               BEGIN
               LCNT=1 ;
               LEVNTJ=LEVNTJ+1 ;
               GOTO GETNEXT.SPEEDWYPT ;
               END ;


   COMMENT END OF NEXT SPEED LOOP
           UPDATE PARAMETERS RELATED TO W(LAST)                                ;

         LAST.ASR=IASR(WKPT) ;        ... SAVE LAST SPEED CONTROL VALUE       //
         LAST= ABEVNO(WKPT)-1 ;       ... SAVE ONE LESS THAN ACTUAL ABS EV NO.
                                          THIS IS A RESULT OF BEING USED
                                          WITH GETNXAFE                        //
         DFTOLAST=DFTO(WKPT) ;        ... DEFINED IN THE                       //
         PROFDST=DFTOLAST-DFTOFRST ;   ... REAL DECLARATIONS                   //


   COMMENT  BYPASS COMPUTATIONS FOR GIVEN PROFILE IF SUCCESSIVE WYPTS ARE
            TAGGED AS SPEED-CONTROL WYPTS                                      ;

         IF LCNT EQL 0                ... DO WE HAVE SUCCESSIVE LATERAL EVENTS
                                                                               //
            THEN GOTO NEXTPROFILE; ... DEFINED AS SPEED CONTROL WYPTS ?    //
         IF FRSTIASR EQL LASTIASR     ... ARE SPEEDS W(FRST) AND W(LAST) EQL?//
            THEN


   COMMENT  SAME SPEED LOOP.
            W(FRST) AND W(LAST) HAVE THE SAME SPEED.  ASSIGN THIS SPEED
            TO ALL INTERMEDIATE PROFILE WYPTS                                  ;

         BEGIN
         FOR ABEVJ=FRST+1 STEP 1 UNTIL LAST-1
                                      ... ASSIGN FRSTIASF TO IASR COMPONENT  //
         DO                           ... AND QUOTES TO IASF COMPONENT        //
         BEGIN                        ... FOR ALL LAT EVENTS BETWEEN
                                          FRST AND LAST                        //
         IF GETNXAFE(AMSK,ABEVJ,WKPT) NEQ GORTN
            THEN GOTO VERUPDATE ;
         IF FCNTYP(WKPT) EQL LTYP ... IS IT A LAT EVENT ?                     //
            THEN                      ... YES                                  //
            BEGIN
            IASR(WKPT)=FRSTIASR ;     ... ASSIGN SPEED TO IASR                 //
            IASF(WKPT)=PRQUOTE ;      ... ASSIGN QUOTES                        //
         IF REPLACFE(ABEVNO(WKPT),ABEVNO(WKPT),ONEITM,ZILCH,WKPT)
               NEQ GORTN
               THEN GOTO VERUPDATE ;
            END   ;
         END ;
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

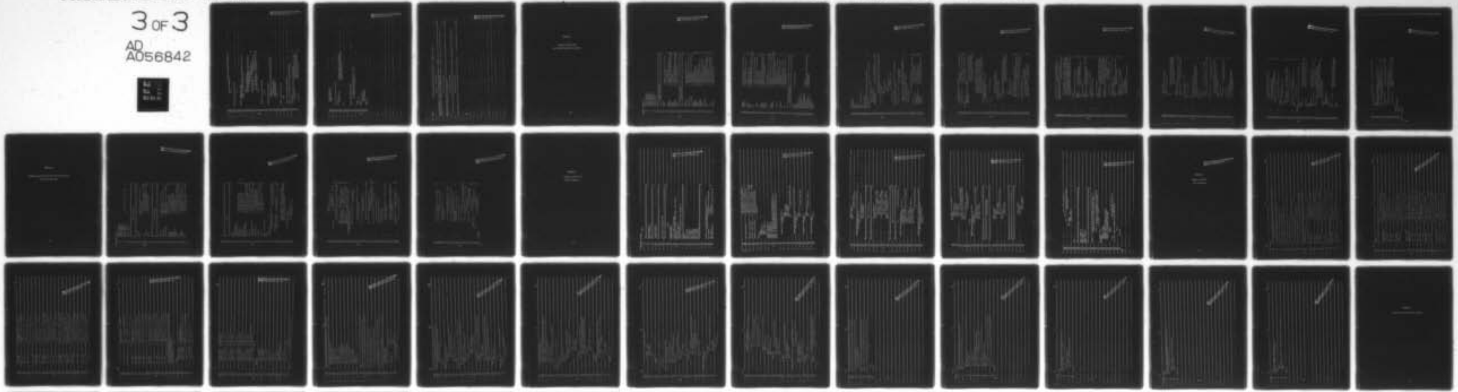
```
        END


COMMENT  END OF SAME SPEED LOOP.                                  ;

        ELSE


COMMENT  VELOCITY CHANGE LOOP.
         W(FRST) AND W(LAST) HAVE DIFFERENT SPEEDS.  RETRIEVE ALL
         EVENTS BETWEEN W(FRST) AND W(LAST) AND ASSIGN APPROPRIATE
         PROFILE SPEED TO THE LATERAL EVENTS.                     ;

        FOR ABEVJ=FRST+1 STEP 1 UNTIL LAST-1
        DO
        BEGIN
        IF GETNXAFF(AMSK.ABEVJ.WKPT) NEQ GORTH
           THEN GOTO VERUPDATE ;
        IF ECWTYP(WKPT) EQL VTYP     ... IS THIS A VERT EVENT ?          //
           THEN                      ... Y S. SET HZ .                   //


COMMENT  VERT LOOP.
         EVENT IS OF VERTICAL TYPE. UPDATE ALTITUDE THAT IS REQUIRED
         FOR IAS TO TAS CONVERSION                                ;

              BEGIN
              IF NOT TOEVE(WKPT)   ... IS IT ALSO A TO-EVENT              //
                 THEN HZ=HZHZ      ... FOR TO-EVENT HZ=AIRCRAFT ALTITUDE//
                 ELSE HZ=ALTU(WKPT) ;
                                   ... OTHERWISE HZ=PILOT ENTERED ALT     //
              END


COMMENT  END OF VERT LOOP                                         ;

COMMENT  LATERAL LOOP.
         EVENT IS OF LATERAL TYPE.  UPDATE PROFILE VELOCITY.      ;

           ELSE
           IF ECWTYP(WKPT) EQL LTYP  ... IS THIS A L.  EVENT ?
                                         NOTICE THAT EVENTS OTHER THAN
                                         LAT AND VERT ARE BYPASSED    //
           THEN
           BEGIN
           DFTOJ=DFTU(WKPT) ;
           DJ=DFTOJ-PREVDFTO ;
           DELDIST= DFTOLAST-DFTOJ ;
           DISTJ=PROFDST-DELD.AT ; ... W(FRST) TO W(J)                     //
           KALT=REWTF((ACTAS-VC MI-ACIASERR).KALTSCALE.HZ*ACTAS-MINK) ;
           KALT=KALT/(ACIAS*HZ) ;
           KALT=RMAXF(RMINF(KALT,.15),.10) ;
                                 ... LIMIT KALT TO THE INTERVAL
                                     BETWEEN .1 AND .10  ( IN THE REAL
                                     WORLD THIS IS EQUIVALENT TO
                                     .000015 TO .000010 )              //
           DVEL = IAST.ASR - PREVIASR ;
```

```
                T1=ABS(AVEL-MINRATE) ;
                AVEV=.5*(PREVIASR+LASTIASR)/(MAX(1-(KALT*HZ)/KALTSCALE) ;
                T2=(DELDIST+DJ)/AVEV ;


                IF T2 LEQ T1          ... IF T2 LEQ T1 THEN THE VELOCITY
                                          CHNG BETWEEN W(J-1) AND W(LAST)
                                          REQUIRES A VEL CHNG RATE GT OR EGL
                                          TO MINRATE                         //


       COMMENT  LINEAR LOOP.
                CALCULATE A VELOCITY, JIASR, THAT IS BASED ON A LINEAR SPEED
                PROFILE BETWEEN W(J-1) AND W(LAST).                          ;

                   THEN
                   BEGIN
                   PRVISR2=PREVIASR*PREVIASR ;
                   JIASR=RROOT2(PRVISR2+(LASTIASR*LASTIASR-PRVISR2)*
                               DJ/(DELDIST+DJ)) ;
                                      ... CALCULATE VEL FOR W(J) BASED ON
                                          LINEAR PROFILE BETWEEN W(J-1)
                                          AND W(LAST)                        //
                   IASR(W  T)=JIASR ;   ... SET SPEED COMPONENT TO NEW SPD //
                   IF JIASR LES PREVIASR   ... IF SPEED DECREASES FROM
                                              W(J-1) TO W(J)                 //
                      THEN IASF(WKPT)=PRDOWN
                                      ... THEN MAKE IASF=ARROW DOWN          //
                      ELSE IASF(WKPT)=PRUP ;
                                      ... ELSE EQUAL TO ARROW UP             //
                   END


       COMMENT  END OF LINEAR LOOP                                           ;

                   ELSE


       COMMENT  MINRATE LOOP.
                DETERMINE DPP, THE DISTANCE REQUIRED TO REALIZE A VELOCITY
                CHANGE OF DVEL AT RATE OF 40 KTS/MIN .                       ;

                   BEGIN
                   DP =T1*AVEV ;       ... DIST REQUIRED TO FLY AT MINRATE
                                          TO REACH V(LAST)                   //
                   IF DELDIST GEQ DPP
                                      ... IF DPP LT DIST FROM W(J) TO W(LAST)
                                          THEN NO CALCS FOR THIS WYPT.       //
                      THEN
                      BEGIN


       COMMENT  NO CHANGE L   .
                VELOCITY RATE IS LESS THAN MINRATE.  ASSIGN W(J) THE SPEED
                OF W(J-1).                                                   ;

                      IASR(WKPT)=PREVIASR ;
```

AD-A056 842   ROCKWELL INTERNATIONAL CEDAR RAPIDS IA COLLINS AVION--ETC F/G 1/5
3D/4D AREA NAVIGATION SYSTEM DESIGN, DEVELOPMENT AND IMPLEMENTA--ETC(U)
JUN 77 J M BRUCKNER, F B BENSON   DOT-FA72WA-3123

UNCLASSIFIED   FAA-RD-77-79-VOL-2   NL

3 OF 3
AD
A056842



END
DATE
FILMED
9-78
DDC

```
                                    ... ASSIGN SAVE SPEED AS PREVIOUS WYPT //
                IASF(WKPT)=PRQUOTE ;
                END


COMMENT  END OF NO CHANGE LOOP                                             ;

                ELSE


COMMENT  MINRATE ZONE LOOP.
         CALCULATE A VELOCITY FOR W(J) THAT IS BASED ON A 40 KT/MIN
         VELOCITY RATE BETWEEN W(J-1) AND W(J).                            ;

                BEGIN
                PRVISR2=PREVIASR*PREVIASR ;
                JIASR=RROOT2(PRVISR2+(LASTIASR*LASTIASR-PRVISR2)*
                          (DPP-DELDIST)/DPP) ;
                                    ... CALC SPEED AT W(J)                 //
                IASR(WKPT)=JIASR ;  ... ASSIGN NEW SPEED                   //
                IF JIASR LES PREVIASR   ... IF SPEED DECREASES FROM
                                            W(J-1) TO W(J)                 //
                   THEN IASF(WKPT)=PRDOWN  ... AND EITHER ARROW DOWN //
                   ELSE IASF(WKPT)=PRUP ;  ... OR ARROW UP                 //
                END ;


COMMENT  END OF MINRATE ZONE LOOP                                          ;

              END ;


COMMENT  END OF MINRATE LOOP.                                              ;

          IF REPLACFE(ABEVNO(WKPT).AEEVNO(WKPT).ONEITM.ZILCH.WKPT)
             NEQ CDRTN
                THEN GOTO VERUPDATE ;
             PREVIASR=IASR(WKPT) ;
             PREVDFTG=DFTO(WKPT) ;
             END ;


COMMENT  END OF LATERAL LOOP.                                              ;

          END ;


COMMENT  END OF VELOCITY CHANGE LOOP.                                      ;

COMMENT  PRIME PARAMETERS FOR NEXT PROFILE COMPUTATION                     ;

NEXTPROFILE $
...      REDEFINE PARAMETERS FOR NEXT PROFILE COMPS                        //
     LEVNTJ=LEVNTJ+1 ;           ... WILL START SEARCH FOR NEXT SPEED
                                     CONTROL WYPT WITH THE LAT. EVENT
                                     FOLLOWING THE LAST SPEED CONTROL      //
     FRST=LAST ;                 ... MAKE FRST PARAMETERS                  //
```

```
          DFTOFRST=DFTOLAST ;
          FRSTIASR=LASTIASR ;
          END ;


     COMMENT  END OF FRST VALID LOOP                               ;

     VERUPDATE $
          COMPRF=INVALID ;              ... INVALIDATE COMPRF TO INDICATE THAT
                                            SPEED PROFILES HAVE BEEN UPDATED FOR
                                            EXISTING FLIGHT PLAN.            //
     END ;




     DEFINE REAL PROCEDURE SIGN(X)
     ...
             THIS PROCEDURE TAKES THE SIGN OF A NUMBER .
             IE    SIGN(-5) = -1, SIGN(10) = 1 .
                                                                     //
     WHERE
          REAL X ;
     TURE
     BEGIN
          IF X GEQ 0.
          THEN SIGN=MAXA4
          ELSE SIGN=MMAX ;
     END ;
     END FINI
```

```
@NEWS
                    COLLINS RADIO COMPANY NEWS PROCESSOR LEVEL 1.0   THURSDAY, OCT 23, 1975

                         MULTI-TPR    OCT 22 1975

A NEW TPR PROCESSOP NOW ON THE SYSTEM WILL ALLOW MULTIPLF COPIES TO MULTIPLE SITES.  MAXIMUM SITES=3, MAXIMUM COPIES PER SITE=9.
*****  EXAMPLE *****      7/RTPR MM1061/3,MM5011/2,PR/5
WOULD SEND 3 COPIES TO REMOTE MM1061, 2 COPIES TO REMOTE MM5011, AND 5 COPIES TOTHE ONSITE 1104 PRINTER IN BUILDING 406.

                         DEMAND-LIMIT OCT 22 1975

EFFECTIVE IMMEDIATELY NO DEMAND USER WILL BE ALLOWED TO SIGN ON WITH MORE THAN 15 MINUTES ESTIMATED RUN TIME DURING PRIME SHIFT.
PRIME SHIFT IS 8:00 AM TO 6:00 PM DALLAS TIME.

                         U1108-OUTAGE OCT 22 1975

  THE U-1108 WILL BE OUT OF SERVICE FROM 0400 P.M. UNTIL 1200 P.M. SATURDAY OCTOBER 25,1975 FOR SYSTEM PROGRAM TEST.

                         SFTWARE-INDX AUG 29 1974

INDEX OF SOFTWARE AVAILABLE ON THE COLLINS U-1108 SYSTEM - FOR LISTING INSERT ..     '7/NEWS,F SFTWARE-INDX' IN YOUR RUNSTREAM ...'
```

APPENDIX C

Program Listing of the Base System Time Control Algorithm

```
4D*GRWFILE.TCON76A2
        BEGIN
        .INSERT FPEOF611 ;
        .INSERT STMCTISY ;
        .INSERT CONTUIPS ;
        .INSERT DATA2611 ;
        .INSERT AEDLIB ;
        .INSERT MATHPACK ;
        .INSERT NTABN611 ;
        EXTERNAL GMTSF ;
        EXTERNAL C880 ;
        ...
        ****************************************************************
                INTEGER DECLARATIONS
        ****************************************************************
                                                                      //
        INTEGER
        C880,                      ... AIRCRAFT TYPE SPECIFIED BY LINK EDIT
                                       C880 = 0 FOR G-I
                                            = 1 FOR C880              //
        ABEVJ,                     ... ABS EVNT NO. OF JTH LAT EVENT  //
        FROMEVNO,                  ... SPECIFIES LATERAL FROM-EVENT NUMBER  //
        VTIMF,                     ... TO-EVENT FLAG. EXPLAINED IN DETAILLATER//
        GMTI ,                     ... GMT INTEGER                    //
        GMTSF,                     ... GMT NOT ENTERED FLAG           //
        TENSEC,                    ... INTERNAL VALUE REPRESENTING TEN SECS  //
        TALCLOCK,                  ... INTERNAL TIME UPDATED EVERY TEN SECONDS//
        KVAL ;                     ... K CONSTANT VALIDITY FLAG       //
        ...
        ****************************************************************
                REAL DECLARATIONS
        ****************************************************************
                                                                      //
        REAL
        VCOMIMAX,                  ... IAS INSTRUMENT UPPER LIMIT FOR
                                       COMMANDED IAS                  //
        VCOMIMAX.GI,               ... VCOMI UPPER LIMIT (300 KTS) FOR G-I
                                       IAS INSTRUMENT                 //
        VCOMIMAX.C880,             ... VCOMI UPPER LIMIT (410 KTS) FOR C-880
                                       IAS INSTRUMENT                 //
        ABSCOCH,                   ...  =ABS(COCHPREV)                //
        ACTAS,                     ... AIRCRAFT TRUE AIR SPEED        //
        ACIASERR,                  ... IAS INSTRUMENT ERROR           //
        ACALT,                     ... AIRCRAFT ALTITUDE              //
        HZ,                        ... AIRCRAFT OR WYPT ALTITUDE      //
        PREVDFTO,                  ... DIST BETWEEN TO-EVENT AND W(J-1)  //
        PREVIASR,                  ... IAS OF W(J-1)                  //
        KALT,                      ... ALTITUDE PROPORTIONALITY CONSTANT
                                       USED TO CONVERT IAS TO TAS     //
        KALTSCALE,                     ... KALT SCALING FACTOR        //
        DJ,                        ... ALONG TRK DIST BETWEEN W(J-1)AND W(J)  //
        T1,                        ... TIME TO FLY FROM W(J-1) TO W(LAST)
                                       AT RATE OF MINRATE             //
        AVEV,                      ... AVE VEL REQUIRED FOR A LINEAR SPEED
                                       PROFILE BETWEEN W(J-1) AND W(LAST)  //
        ACEWND,                    ... EAST COMPONENT OF WIND         //
        ACNWND,                    ... NORTH COMPONENT OF WIND        //
```

```
TIGNOM,                    ... ESTIMATED TIME TO REACH THE TIME FIX   //
COCHPREV,                  ... CRSO(J-1)-CRSI(J-1)                    //
WATK,                      ... ALNG TRK COMPONENT OF WIND APPLIED
                               TO W(J) .                              //
TASPREV,                   ... COMPUTED IAS FOR W(J-1)                //
TASJ,                      ... COMPUTED IAS FOR W(J)                  //
DV,                        ... SPEED CHNG BETWEEN W(J-1) AND W(J)     //
DELT,                      ... THE EXPECTED TIME TO FLY FROM W(J-1) TO W(J)
                                                                      //
DP,                        ... PORTION OF LEG FLOWN AT VEL ASSIGNED
                               TO W(J-1)                              //
WATKTO,                    ... ALONG TRACK COMPONENT OF WIND FOR THE
                               LEG BEING FLOWN                        //
VO,                        ... IAS OF W(J-1)+WATK                     //
VF,                        ... IAS OF W(J) + WATK                     //
VGNOM,                     ... NOMINAL GROUND VEL FOR THE PRESENT LEG //
TIGCOM,                    ... COMMANDED TIME TO REACH THE TIME-FIX   //
KSCALE,                    ... SCALING TO PREVENT OVERFLOW            //
GMTP,
ONHZKT,                    ... =MAX64-HZ*KALT                         //
K,                         ... PROPORTIONALITY CONSTANT USED TO DETERMINE
                               WHAT VCOMI WILL BE                     //
KK,                        ... INTERMEDIATE NUMBER USED TO DETERMINE K//
MIGDPP,                    ... =MIG/DPP                               //
PIGTAN25,                 .... =PI*(6076.1 FT/NM*1000 NM/HR)/(32.2 FT/SEC**2
                               *(3600 SEC/HR)**2 *10.8039624 HR
                               *TAN(25 DEGREES))                      //
GMTCONV,                   ... 10.8039624/12.0  USED TO CONVERT GMT
                               UNITS TO NAV TIME UNITS
                               - GMTS/GMTCONV=TIME IN NAV UNITS       //
ONEOVRPI,                  ... 1.0/3.141596                           //
T2,                        ... TIME TO FLY FROM W(J-1) TO W(LAST)
                               WITH CONSTANT VEL AVEV                 //
DPP ;                      ... DIST TO REALIZE VEL CHNG OF DVEL AT
                               RATE OF MINRATE                        //
...
************************************************************************
         POINTER DECLARATIONS
************************************************************************
                                                                      //
POINTER
WAPT ;                     ... WORK AREA POINTER                      //
PRESET
BEGIN
VCOMIMAX = .5 ;
VCOMIMAX.G1 = .5 ;
VCOMIMAX.C880 = .41 ;
ABSCOCH=0. ;
PIGTAN25=.00907942 ;
ONEOVRPI=.318309886 ;      ... 1.0/PI                                 //
KSCALE=.25 ;               ... SCALING USED FOR TIGCOM/TIGNOM COMPUTN //
TENSEC=24414 ;             ... 10*1000000/409.6                       //
GMTCONV=.900330200 ;       ... 10.8039624/12.0                        //
FROMCVWO=-1 ;
ACIASERR=0. ;
ACALT=0. ;
HZ=0. ;
```

```
PHLVDF10=0. ;
KALTSCALE=.1 ;
KVAL=0 ;
K=0 ;
KK=0 ;
AVLV=0. ;
END ;
DEFINE PROCEDURE TIMCNT
TOBE
BEGIN
OWN POINTER WKPT ;
OWN INTEGER ABEVJ ;
OWN REAL ACTAS,DELT,ITGNOM,TASPREV,T1,T2,KALT,VGNOM ;
GMIP $=$ GMT1 ;
WKPT=WKA1 ;                          ... INITIALIZE WORK AREA POINTER        //
      IF .CLOCK()-TALCLOCK LES TLMSEC
         AND ICOMF NEQ VALID
         THEN GOTO EGGSIT ;
      TALCLOCK=.CLOCK() ;            ... UPDATE TALCLOCK READING             //
      TIMERF=INVALID ;               ... INVALIDATE ARRIVAL TIME ERROR FLAG //

...                   DETERMINE VALIDITY STATUS OF GMT,IAS,AND ACALT. STATUS
                      OF ALL THREE MUST BE VALID IN ORDER TO CONTINUE.   //
      IF VCOMIF=(IF GMTSF GRT 0
                      THEN IVCOMI
                      ELSE
                      IF IASF(ADHD) GRT 0
                          THEN IVIAS
                          ELSE
                          IF INTIEV(LOC HZHZF) GRT 0
                              THEN IVALT
                              ELSE NODATA) NEQ NODATA
         THEN GOTO EGGSIT ;

...DETERMINE VCOMIMAX, THE UPPER LIMIT FOR VCOMI, AS REQUIRED
      BY THE AIRCRAFT IAS INSTRUMENT                                         //

      VCOMIMAX = IF LOC C880 GRT 0 THEN VCOMIMAX.C880
                                    ELSE VCOMIMAX.GI ;

...        NOW CHECK VALIDITY STATUS OF FROM-EVENT. WILL EXIT IF
           FROM-EVENT IS NOT IN FL.PLAN, THERE IS A VERSION MISMATCH
           OR NO SPEED IS DEFINED FOR FROM-EVENT                             //
      IF GETTIXEL(LXSK,FROMEVNO,WKPT) NEQ GORIN
         OR IASF(WKPT) LSL NOIAS
         THEN GOTO EGGSIT ;

...                   GET A SNAPSHOT OF SENSOR DATA                          //
      .CKPT() ;                      ... YEILD CHANNEL TIME TO ENSURE TIME
                                         TO GET A GOOD SNAPSHOT              //
      ACIAS=IAS(ADHD) ;              ... AIRCRAFT TRUE AIR SPEED             //
      ACALT=HZHZ ;                   ... AIRCRAFT ALTITUDE                   //
      ACIASERR=DELV(ADHD) ;          ... IAS INSTRUMENT ERROR                //
      ACNWND=NWND ;                  ... NORTH COMPONENT OF WIND             //
      ACEWND=EWND ;                  ... EAST  COMPONENT OF WIND             //
      HZ=ACALT ;                     ... INITIALIZE HZ TO THE AIRCRAFTS ALT //
```

```
...    INITIALIZE NECESSARY PARAMETERS                          //
    PREVIASH=IASH(WKPT) :   ... INIT. PREVIASH  TO IASR COMPONENT
                                OF FROM-EVENT                   //
    PREVDFTO=DFTO(WKPT) :   ... INIT. PREVDFTO TO DIST FROM TO-EVENT
                                TO THE FROM-EVENT               //
ABEVJ=0 :                   ... START WITH ABSOLUTE EVENT NUMBER 0 //
    TTGNOM=0.0 :            ... TIME TO GO = 0.0                //
    COCHPREV=CRSO(WKPT)-CRSI(WKPT) :
    VTIMF=VALID :           ... IF THERE IS NO TFIX ENTERED THEN
                                TO-EVENT IS TREATED AS TFIX. SO WHEN
                                TO-EVENT BLOCK IS EXECUTED WE SET
                                VTIMF=TFIXF. IF THERE IS NO TFIX THEN
                                VTIMF IS INVALID                //

...            BEGIN CALCTTG (CALCULATE TIME TO GO) BLOCK
               THIS BLOCK DETERMINES THE ACTUAL (NOMINAL) TIME
               THE AIRCRAFT WILL USE TO FLY FROM ITS PRESENT
               POSITION TO THE TIME-FIX. TTGNOM IS DETERMINED
               AND SUMMED FOR EACH LEG UP TO THE TIME-FIX. IF
               THERE IS NO TFIX THE TO-EVENT IS TREATED AS ONE    //
    WHILE WYPTID(WKPT) NEQ TFIX AND
         GETNXAFE(AMSK,ABEVJ,WKPT) EQL GOHTN
         AND VTIMF EQL VALID
      DO

...             BEGIN CALCTTG BLOCK                              //
      BEGIN


...            BEGIN LATERAL EVENT BLOCK
               CALL TTGNOM FOR THE GIVEN LEG                     //
         IF ECWITP(WKPT) EQL LTYP
         AND IASF(WKPT) NEQ NOIAS
           THEN
           BEGIN
...            NOTE THAT END-OF-CONTINUITY EVENTS ARE PASSED      //
           IF ELAT(WKPT)     ... IF THE PRESENT LATERAL EVENT HAS
                                AN INVALID LAT-LONG SET COMMANDED
                                VELOCITY AND TIMERROR FLAGS INVALID//
              THEN
              BEGIN
              VCOMIT=TIMERF=INVALID :
              GOTO EGGSIT :
              END :
           DJ=DFTO(WKPT)-PREVDFTO : ... W(J-1) TO W(J)           //
           PREVDFTO=DFTO(WKPT) :  ... W(TO) TO W(J)              //
           WAIRS=-(ACNWND*NCOS(CRSI(WKPT))+ACEWND*NSIN(CRSI(WKPT))) :
                                ... WIND COMPONENT AT W(J)       //
         KALT=RLMTF((ACTAS-VCOMI-ACIASERR)*KALTSCALE,HZ*ACTAS-MINR) :
                                ... IAS-TAS PROPORTIONAL. CONST  //
           KALT=KALT/(HZ*ACTAS) :
         KALT=RMAXF(RMINF(KALT,.15),.10) :
                                ... KALT LIMITED                 //
              OHHZKT=MAXHH-(HZ*KALT)/KALTSCALE :
           TASPREV=PREVIASH/OHHZKT :  ... TAS OF W(J-1)          //
           TASJ=IASH(WKPT)/OHHZKT : ... TAS OF W(J)              //
           DV=TASJ-TASPREV :     ... DELTA TAS W(J)-W(J-1)       //
```

```
                T1=ABS(DV*MINRATE) ; ... TIME TO FLY REQUIRED DV AT
                                          ACCELERATION OF 1/MINRATE        //
                AVEV=0.5*(IASPREV+TASJ)+WATK ;
                T2=ABS(DJ/AVEV) ; ... TIME TO FLY THE PRESENT LEG
                                               ASSUMING IT IS FLOWN AT AVEV //

    ...             CAN NOW DETERMINE THE TYPE OF PROFILE TO BE FLOWN
                    FOR THIS LEG. IF T2 LEQ T1 THEN THE VELOCITY CHANGE
                    BETWEEN W(J-1) AND W(J) WILL BE AT A CONSTANT RATE
                    EQ MINRATE FOR A TIME T2. OTHERWISE THE LEG IS FLOWN
                    AT A CONST VELOCITY FOR DISTANCE DP AND AT AN
                    ACCELERATION OF MINRATE FOR DISTANCE DPP AND TIME T1
                                                                        //
                IF T2 LEQ T1
                  THEN
    ...             FOR GREATER THAN OR EQUAL TO MINRATE               //
                  BEGIN
                  DELT=T2 ;      ... TIME TO FLY THE LEG                //
                  DPP=DJ ;       ... DPP IS THE LEG DISTANCE            //
                  END            ... END GREATER THAN MINRATE BLOCK     //
                  ELSE
    ...             FOR LESS THAN MINRATE                              //
                  BEGIN
                  DELT=T1 ;      ... TIME TO FLY THE CHANGE IN VELOCITY //
                  DPP=T1*AVEV ;  ... DISTANCE TO DO THE SAME            //
                  END ;          ... END MINRATE BLOCK                  //

    ...             DETERMINE IF W(J) IS THE TO-EVENT                  //
                IF NOT TOEVE(WPT)
    ...             TRUE IF W(J) IS TO-EVENT                           //
                  THEN

    ...             BEGIN TO-EVENT BLOCK                               //
                  BEGIN
                  VCOMIF=IASVALID ; ... VALIDATE COMMAN IAS VAL FLAG    //
                  VTIMF=TFIXF ; ... TFIXF IS SET VALID-INVALID BY FPMON.
                                    IF THERE IS NO TIME FIX WE LEAVE
                                    CALCTIC BLOCK AFTER CALCING TIGNOR //
                  WATKTO=WATK ; ... ALONG TRACK WIND COMPONENT TO-EVNT //
                  IF LOWIAS NEQ VALID ... IF TRUE AIR SPEED IS LESS
                                          THAN TASLIM (150 KTS) , HOLD
                                          KALTTO CONSTANT               //
                    THEN KALTTO=KALT ; ... OTHERWISE UPDATE KALTTO      //

    ...             NOW DECIDE IF THERE WILL BE A VEL CHNG BETWEEN THE
                    FROM AND TO EVENTS                                  //
                  IF T2 LEQ T1 OR DMTG LEQ DPP
                    THEN         ... DETERMINE DELT FOR A VELOCITY RATE
                                     GREATER THAN MINRATE               //
                    BEGIN
                    VO=IASPREV+WATK ;  ... W(J-1)+WIND(J)               //
                    VF=IASJ+WATK ;     ... W(J)+WIND(J)                 //
    ...              LIMIT DMTG TO LEQ DPP                              //
                    IF DMTG GRT DPP
                       THEN MTGDPP=MAX64
                       ELSE MTGDPP=DMTG/DPP ;
                    VGNOM=RROOT(VF*VF+(VO*VO-VF*VF)*MTGDPP) ;
```

```
                 IF DMTG GRT DPP
                    THEN DELT=T2+(DMTG-DPP)/VGNOM
                    ELSE
                 DELT=DMTG/(0.5*(VGNOM+VF)) ;
                 END

                 ELSE          ... NO VEL CHNG REQUIRED FOR FIRST PART
                                   OF THIS LEG INTO TO-EVENT. DETERMINE
                                   HOW MUCH OF LEG IS LEFT TO FLY AND
                                   HOW LONG IT WILL TAKE                 //
                 BEGIN
                 VGNOM=TASPREV+WATK ;
                 DP=DMTG-DPP ; ... PORTION OF LEG REMAINING TO BE FLOWN
                                     AT CONSTANT VELOCITY                //
                 DELT=T1+DP/VGNOM ;    ... TIME TO FLY THIS PORTION //
                 END ;

...            NOW DETERMINE TTG TO TO-EVENT                             //
              TTGNOM=TTGNOM+DELT ;
              END
...            END TO EVENT BLOCK                                        //

              ELSE

...            BEGIN BEYOND TO-EVENT BLOCK
               DETERMINE TTG UP TO THE TIME FIX. INCLUDED
               ARE COMPENSATIONS FOR LEG CAPTURE AT W(J-1)
               AND INVALIDATE CALCTTG COMPUTATIONS FOR
               COCH GEQ 160                                              //
              BEGIN
              IF T2 GRT T1
                 THEN
                 BEGIN
              DP=DJ-DPP ;    ... SECTION OF LEG FLOWN AT CONSTANT
                                              TASPREV.                   //
                 DELT=T1+DP/(TASPREV+WATK) ; ... TOTAL TIME TO FLY LEG
                                                                         //
                 END ;
              TTGNOM=TTGNOM+DELT ;  ... ADD THIS LEG TO THE TOTAL
                                         TIME TO GO TO T-FIX             //

...                 WILL NOW MAKE ADJUSTMENT FOR SLIGHTLY DIFFERENT
                    PATH FLOWN DUE TO TURNS.ECT.NOT ALLOWING
                    COURSE CHANGES GEQ 160 DEGREES                       //
              IF ABS(COCHPREV) GEQ D160
                 THEN
...                 INVALID COCH                                         //
                 BEGIN
                 VCOMIF=IVCOCH ;  ... INVALIDATE COMMANDED IAS FOR
                                        COURSE CHANGE GEQ D160           //
                 GOTO LOGSIT ;
                 END ;
...                 VALID COCH                                           //
              ABSCOCH=ABS(COCHPREV) ;
                DELT = 0.5*ABSCOCH*NCOS(0.5*ABSCOCH)
                       - NSIN(0.5*ABSCOCH)*ONEOVRPI ;
                DELT=TASPREV*DELT*PIGTAN25 ;
```

```
                    DELT = DELT/(0.5*NCOS(0.5*ABSCOCH)) ;
                  TIGNOM=TIGNOM+DELT ;   ... ADD TO TIGO                  //
                  END ;
...                 END BEYOND TO-EVENT BLOCK                             //
               COCHPREV=CRSO(WKPT)-CRSI(WKPT) ;
                                  ... UPDATE COCHPREV                     //
               PREVIASR=IASR(WKPT) ; ... UPDATE PREVIASR                  //
               END
...                 END LATERAL EVENT BLOCK                               //
...                 IS EVENT ABEVJ A VERTICAL EVENT ?                     //
            ELSE
         IF EC=TYP(WKPT) EQL VTYP
...                 BEGIN VERTICAL EVENT BLOCK                            //
            THEN
            BEGIN
            IF NOT TOEVE(WKPT)   ... TRUE IF W(J) IS TO-EVENT             //
               THEN HZ=ACALT     ... ASSIGN AIRCRAFT ALT TO HZ            //
               ELSE HZ=ALTD(WKPT) ;
                                  ... ASSIGN ALTD(J) TO HZ                //
            END ;
...                 END VERTICAL EVENT BLOCK                              //
         ABEVJ=ABEVJ+1 ;          ... INCREMENT EVENT NUMBER              //
         END ;
...                 END CALCTIG BLOCK                                     //
TIMCALCS
      IF(TIMERF=TCOMF .V. TFIXF) NEQ VALID ... SET TIMERR FLAG            //
         THEN
         BEGIN
         K=KSCALE ;               ... IF THERE IS NO TIME-FIX OR TCOMF
                                      THEN TTGCOM AND HENCE ALSO K CAN NOT BE
                                      DETERMINED SO LET K=KSCALE          //
         KVAL=VALID ;             ... VALID SO WE CAN CALC VCOMI.VCOMI    //
         END
         ELSE
         BEGIN
         GMTI=TCOM-GMTS ;         ... CONVERTS INTEGER TO REAL
                                      BECAUSE GMTR $=$ GMTI               //
         TTGCOM=GMTR/GMTCONV ; ... CALC THE COMMANDED TTG FOR ARRIVAL
                                            AT THE TIME FIX               //
         KVAL=INVALID ;           ... INITIALLY SET KVAL INVALID          //
...                 IF THE PRODUCT OF TIGNOM AND KSCALE IS LESS
                    THAN THE ABS(TTGCOM) WE WILL HAVE A MEANINGFULL
                    CONSTANT K. GENERALY THIS PROTECTS AGAINSTTHE CASE
                    WHERE TIGCOM GOES TO ZERO                             //
         IF KK=TIGNOM*KSCALE LEQ ABS(TTGCOM)
            THEN
            BEGIN
            K=KK/ABS(TTGCOM) ;
            KVAL=VALID ;
            END ;
         END ;

      IF VCOMIF EQL IASVALID     ... VCOMIF WILL NOT BE VALID FOR
                                     THE CASE WHERE THERE IS NO SPEED DEFINED
                                     FOR THE TO -EVENT OTHERWISEITS VALD//
         THEN
         BEGIN
```

```
      ...              IF WE HAVE A VALID K THEN CAN CALC VCOMT AND VCOMI //
               IF KVAL EQL VALID
               THEN
               BEGIN
               VCOMT=(K*VGNOM)/KSCALE-WATKTO ; ... CALC COMMANDED TAS        //
               VCOMI=(MAX6H-(AALTIO*ACALT)/KALTSCALE)*VCOMT ;
                                          ... CONVERT VCOMT TO IAS         //
               VCOMI = RMAXF(RMINF(VCOMI,VCOMIMAX), .0) ;  ... RESTRICT VCOMI
                          TO LIMITS IMPOSED BY THE IAS INSTRUMENT          //
               END ;
               IF TIMERF EQL VALID     ... IF THERE IS A TIME-FIX OR COMMANDED
                                           TIME THEN CAN CALC TIMERR       //
                  THEN
                  EARLY=((TGNUM*VGNCMI/(ACTAS+WATKTO)-TIGCOM ;
                                          ... CALC THE EXPECTED TIME ERROR FOR
                                           ARRIVAL AT THE TIME FIX         //
                  TIMERR=ABS(EARLY) ; ... DO THIS TO MAKE CORRECTION FOR NEGATIVE
                                           NUMBERS                         //
               END ;
          EUUSITS
               TCONF=INVALID ;
          END ;
          END FINI

@PRT,S  D.TCONZEA1
```

APPENDIX D

Program Listing of the 2D Plus Time Control System

Time Control Algorithm

```
4D*WORKFILE.TCON7EA1
BEGIN
.INSERT DATA2E11 ;
.INSERT DATA2611 ;
.INSERT STACTISY ;
.INSERT HTAHN611 ;
.INSERT AEDLIB ;
.INSERT ASCINIFO ;                ... CONTAINS DECLARATION FOR GMISF      //
.INSERT MATHPACK ;

EXTERNAL C880 ;
...
.......................................................................
.                                                                     .
.                     INTEGER DECLARATIONS                            .
.                                                                     .
.......................................................................
                                                                     //
INTEGER
C880,                     ... AIRCRAFT TYPE SPECIFIED BY LINK EDIT
                              C880 = 0 FOR G-I
                                   = 1 FOR C880                      //
GMTI,                         ... OWN COPY OF GMTS EXPRESSED IN TERMS
                                  OF GMI UNITS (12 HOURS)            //
NALCLOCK,                     ... OWN RECORD OF CLOCK READING THE LAST
                                  TIME TCON WAS EXECUTED             //
TENSEC:                       ... TEN SECONDS ON CLOCK               //

...
.......................................................................
.                                                                     .
.                     REAL DECLARATIONS                               .
.                                                                     .
.......................................................................
                                                                     //
REAL
VCOMIMAX,                 ... IAS INSTRUMENT UPPER LIMIT FOR
                              COMMANDED IAS                         //
VCOMIMAX.GI,              ... VCOMI UPPER LIMIT (300 KTS) FOR G-I
                              IAS INSTRUMENT                        //
VCOMIMAX.C880,            ... VCOMI UPPER LIMIT (410 KTS) FOR C-880
                              IAS INSTRUMENT                         //
ACALT,                        ... AIRCRAFT BARO ALTITUDE             //
ACEWND,                       ... EAST COMPONENT OF WIND             //
ACIASERR,                     ... IAS INSTRUMENT ERROR               //
ACNWND,                       ... NORTH COMPONENT OF WIND            //
ACTAS,                        ... AIRCRAFT TRUE AIR SPEED            //
DESIRED.CRS,                  ... HSI COURSE INPUT                   //
GMTCONV,                      ... USED TO CONVERT GMT TO NAV TIME    //
GMTR,                         ... OWN COPY OF GMTS IN NAV TIME UNITS //
KALT,                         ... ALTITUDE PROPORTIONALITY FACTOR USED
                                  TO CONVERT TAS TO IAS              //
KALTSCALE,                    ... KALT OVERFLOW SCALING FACTOR       //
TTGCOM,                       ... COMMANDED TIME TO GO TO ARRIVE AT
                                  THE TO-WYPT                        //
WATK:                         ... ALONG TRACK COMPONENT OF WIND BASED
                                  UPON THE DESIRED COURSE            //
```

```

...
......................................................................................
.                                                                                    .
.                        PRESETS                                                     .
.                                                                                    .
......................................................................................
                                                                                    //
PRESET
BEGIN
VCOMIMAX = .5 ;
VCOMIMAX.GI = .5 ;
VCOMIMAX.C880 = .41 ;
ACALT=0. ;                           ... AIRCRAFT BARO ALTITUDE                     //
ACEWND=0. ;                          ... EAST COMPONENT OF WIND                     //
ACIASERR=0. ;                             ... IAS INSTRUMENT ERROR                       //
ACNWND=0. ;                          ... NORTH COMPONENT OF WIND                    //
ACTAS=0. ;                           ... AIRCRAFT TRUE AIR SPEED                    //
DESIRED.CRS=0. ;                     ... HSI COURSE INPUT                           //
GMICONV= .900330200 ;                ... 10.8039624 HOURS NAV/12.0 HOURS GMT//
GMTI=0 ;                             ... OWN COPY OF GMTS EXPRESSED IN TERMS
                                         OF GMT UNITS (12 HOURS)                    //
KALT=0. ;                            ... ALTITUDE PROPORTIONALITY FACTOR USED
                                         TO CONVERT TAS TO IAS                      //
KALTSCALE=.100000000 ;               ... KALT OVERFLOW SCALING FACTOR               //
TALCLOCK=0 ;                         ... OWN RECORD OF CLOCK READING THE LAST
                                         TIME TCON WAS EXECUTED                     //
TENSEC=24414 ;                       ... (10SEC)(1000000MICROSEC/SEC)/
                                                (409.6MICROSEC/COUNT)               //
TTGCOM=0. ;                          ... COMMANDED TIME TO GO TO ARRIVE AT
                                         THE TO-WYPT                                //
WATK=0. ;                            ... ALONG TRACK COMPONENT OF WIND BASED
                                         UPON THE DESIRED COURSE                    //
END ;                                ... END PRESETS                                //
DEFINE PROCEDURE TIMCNT
TUNE
BEGIN

GMIR $=$ GMTI ;
        IF .CLOCK()-TALCLOCK GEQ TENSEC  ... IF TEN SECONDS HAVE ELAPSED//
           OR NLTOCHG EQL VALID          ... OR A FLT PLAN REVISION HAS
                                              OCCURED SINCE LAST CYCLE   //
        THEN BEGIN
             TALCLOCK = .CLOCK() ; ... UPDATE TALCLOCK                      //
             TIMERR = INVALID ;    ... INVALIDATE ARRIVAL TIME ERROR FLG//

...
                         DETERMINE VALIDITY STATUS OF VCOMIF.  VCOMIF
                         IS DEPENDENT UPON THE MAJOR QUANTITIES USED TO CALC
                         VCOMI AND TIMERR & TAS,ACALT,AND GMT. EVERYTHING
                         MUST BE TRUE IN ORDER TO CONTINUE
                                                                                    //
             IF VCOMIF=(IF GMTSF GRT 0
                           THEN IVGMT
                           ELSE IF TASF(ADHO) GRT 0
                                THEN IVTAS
                                ELSE IF BALF(ADHO) GRT 0
```

```
                                THEN IVALT
                                ELSE IF ICOMF EQL INVALID
                                        THEN NODATA
                                        ELSE IASVALID) EQL IASVALID
           THEN BEGIN            ... VCOMI VALID BLOCK                //
...
                     GET A SNAPSHOT OF SENSOR DATA USED FOR TIME CONTROLS
                     OWN CALCULATIONS. MUST YEILD CHANNEL TIME FIRST
                                                                      //
                .CKPT() ;        ... YEILD CHANNEL TIME               //
                ACIAS = IAS(ADHD) ;   ... AIRCRAFT TRUE AIR SPEED     //
                ACALT = BALT(ADHD) ;  ... AIRCRAFT ALTITUDE           //
                ACIASERR = DELV(ADHD); ... IAS INSTRUMENT ERROR       //
                ACNWND = NWND ;        ... NORTH COMPONENT OF WIND    //
                ACEWND = EWND ;        ... EAST COMPONENT OF WIND     //
                DESIRED.CRS = HSICRS(ADHD); ... HSI COURSE INPUT      //
... DETERMINE VCOMIMAX, THE UPPER LIMIT FOR VCOMI, AS REQUIRED
     BY THE AIRCRAFT IAS INSTRUMENT                                   //

                VCOMIMAX = IF LOC C880 GRT 0 THEN VCOMIMAX.C880
                                             ELSE VCOMIMAX.GI ;


...
                     FIND THE ALONG TRACK COMPONENT OF THE WIND FOR
                     THE GIVEN DESIRED COURSE
                                                                      //

                WATK = -(ACNWND*NCOS(DESIRED.CRS)+ACEWND*
                                NSIN(DESIRED.CRS)) ;
...
                     CALCULATE KALT,THE ALTITUDE PROPORTIONALITY CONSTANT
                                                                      //
                KALT = RLMIF((ACIAS-VCOMI-ACIASERR)*KALTSCALE,
                                  ACALT*ACIAS-MINR) ;
                                   ... LIMIT FUNCTION AND KALTSCALE ARE USED
                                       TO PROTECT AGAINST OVERFLOW    //
                KALT = KALT/(ACIAS*ACALT) ;  ... CALC KALT            //
...
                     LIMIT KALT TO THE INTERVAL (.1,.15)
                                                                      //
                KALT = RMAXF(RMINF(KALT,.15),.10) ;

                IF LOWIAS NEQ VALID
                   THEN KALTIO = KALT ;
...
                     IF TRUE AIR SPEED IS LESS THAN IASLIM (150 KNOTS) ,
                     HOLD KALTIO CONSTANT
                                                                     //
...
                     IF A COMMANDED ARRIVAL TIME HAS BEEN ENTERED VIA
                     THE CDU, THEN COMMANDED IAS AND ARRIVAL TIME
                     ERROR MAY BE CALCULATED.
                                                                      //
                IF ICOMF EQL VALID  ... ARRIVAL TIME ENTERED ?        //
                THEN BEGIN          ... BEGIN TIMERR BLOCK            //
...
                      DETERMINE COMMANDED TIME TO GO
```

```
                                                                                   //
                         GMTI = TCOM - GMTS :
                                        ... INTEGER ARITHMETIC REQUIRED            //
                         TTGCOM = GMTR/GMTCONV :
                                        ... CONVERT TO NAV TIME UNITS              //
         ...
                           CALCULATE VALUES FOR COMMANDED TAS
                           AND COMMANDED IAS.
                                                                                   //

                         VCOMT = RLMTF(DMTG.TTGCOM-MINR)/TTGCOM - WATK :
                                        ... CALC COMMANDED TAS                     //

                         VCOMI = (MAX64-(KALT*ACALT)/KALTSCALE)*VCOMT :
                                        ... CONVERT TAS TO IAS                     //
         ...             RESTRICT VCOMI TO LIMITS IMPOSED BY THE
                         IAS INSTRUMENT                                            //
                         VCOMI = RMAXF(RMINF(VCOMI.VCOMIMAX), .0) :
         ...
                           DETERMINE EARLY/LATE AND TIMERR
                                                                                   //
                         EARLY = DTTG - TTGCOM :
                                        ... THE SIGN OF EARLY PROVIDES EARLY OR
                                            LATE INDICATION FOR CDU DISPLAY        //
                         TIMERR = ABS(EARLY) :
                                        ... MAGNITUDE OF ARRIVAL TIME ERROR
                                            FOR CDU DISPLAY                        //
                         TIMERF = VALID :  ... INDICATE VALID ARRIVAL
                                               TIME ERROR COMPUTATION              //
                         END :          ... END TIMERR BLOCK                       //
                   END :                ... VCOMI VALID BLOCK                      //
              NLTOCHG = INVALID :          ... ACKNOWLEDGE REVISION TO
                                               THE FLT PLAN                        //
              END :                     ... END MAIN BLOCK                         //
         END :                        ... END PROCEDURE TIMCNT                     //
         END FINI

*FIN
```

APPENDIX E

Program Listing of ILS Executive Routines

```
@D.WORKFI.E.ILSBMDA1
                     BEGIN


       ...
       ><><><><><><><2<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                                                                               //
       .INSERT LCGSNDI1 $.
       .INSERT MATHPACK $.
       ...
       ><><><><><><><2<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                                                                               //
        .INSERT NTABN6T1 $.
        .INSERT AEPLIB $.
       ...
       ><><><><><><><2<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                                                                               //

       DEFINE PROCEDURE MAINLCGS TOBL
       ...
       ><><><><><><><B<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                                                                               //
                         BEGIN

       PROCEDURE GSCMP $.
       PROCEDURE LOCCMP $.

       COMMENT INTERNAL DECLARATIONS $.
       BOOLEAN LOCARM, LOCCAP , GSARM , GSCAP , RESETL, RESETG,
       LOCTRK  $.
       REAL LATDIS, LNGDIS $.
       INTEGER AC.MDE $.
       COMMENT SET ILS VARIABLES TO ANAV VARIABLES $.
       ...
       ><><><><><><><2<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
       SYNONYMS
       BEGIN
       HDTZ = ALTRTE
       H2HZ = ALT
       GNDV = GNDSPD
       CT = GAMMA
       LAMP = LATAP
       PHIP = LONGAP
       DTOTD = RNGTD
       RWMD = RLGMOD
       AENG = APENG
       END - SYNONYMS
       ><><><><><><><B<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                                                                               //



       COMMENT ***** DATA DECLARATIONS FOR LOGIC USED TO DRIVE
                     THE HSI TO/FROM FLAG ***** $.
       REAL TOCRSDEV !                   ... TAE RELATIVE TO TOUCHDOWN AND
                                             RWYHDG                         //
       REAL DEG90 !                      ... 90 DEGREES                     //
       REAL ARRAY TDPOS(1) !             ... COORDINATES OF TOUCHDOWN USED FOR
```

```
                    CALL TO LLTBD.  ARRAY IS NEEDED TO
                    ENSURE CONSECUTIVE CORE LOCATIONS
                    FOR TOUCHDOWN COORDINATES.          //
REAL ARRAY ACTOTD(3) $       ... USED TO RECORD LLTBD BEARING AND
                    DISTANCE COMPUTATIONS BETWEEN THE
                    AIRCRAFT AND TOUCHDOWN.             //
SYNONYMS 0 = BRG.ACTOTD $    ... ACTOTD ELEMENT USED TO RECORD BEARING
                    FROM AIRCRAFT TO TOUCHDOWN          //
SYNONYMS 1 = BRG.TDTOAC $    ... ACTOTD ELEMENT USED TO RECORD BEARING
                    FROM TOUCHDOWN TO AIRCRAFT          //
SYNONYMS 2 = DIST.ACTOTD $   ... ACTOTD ELEMENT USED TO RECORD
                    DISTANCE BETWEEN AIRCRAFT AND
                    TOUCHDOWN                           //
PRESET
   BEGIN
   TDCKSDEV = .0 $
   DEG90 = .5 $
   TDPOS = .0,.0 $
   ACTOTD = .0,.0,.0 $
   END $
LOCDEV = LOCD(ADHD) $.
GSDEV = GS(ADHD) $.
AIRSPD = TAS(ADHD) $.
ACHDG = PSIM(ADHD) $.
SFLVAL = IF LOCF(ADHD) EQL 0 THEN TRUE ELSE FALSE $.
COMVAL = IF IBYTEO (LOC HEADF) EQL 0 THEN TRUE ELSE FALSE $.
SFGVAL = IF GSF(ADHD) EQL 0 THEN TRUE ELSE FALSE $.
ALTVAL = IF IBYTEO(LOC HZHZF) EQL 0 THEN TRUE ELSE FALSE $.



   COMMENT SET ANNUNCIATORS FALSE $.
             PRESET
                 BEGIN
                 LOCARM = FALSE $.
                 LOCCAP = FALSE $.
                 GSARM = FALSE $.
                 GSCAP = FALSE $.
                 END        ... PRESET ANNUNCIATORS FALSE
                              $.

   COMMENT SET ACTIVE MODE TO RNAV ** RNAV IS ONE ** LOC IS TWO **
        APPR AUTO IS THREE $.
             PRESET
                 BEGIN
                 ACTMDE = 1 $.
                 END        ... PRESET ACTIVE MODE $.

   COMMENT SET LOCTRK FALSE $.
             PRESET
                 BEGIN
                 LOCTRK = FALSE $.
                 END        ... SETTING LOCTRK $.

   COMMENT INITIATE RESETS $.
             PRESET
                 BEGIN
```

```
                        RESETL = TRUE $,
                        RESETG = TRUE $,
                        END           ... PRESET RESETS $,
                                      ... END INITIATE BLOCK $,

        COMMENT DISTANCE CALCULATIONS FOR DISTANCE TO TOUCHDOWN AND
                DISTANCE TO LOC ANTENNA $,
                     BEGIN
                        REAL LATFIX,DLAT,DLONG,IRES,IPI $,
                        IRES = 0.02907815 $,
                        IPI = 0.31830989 $,
   ...
   ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
                                                                    //
                        LATFIX = NCOS(LATTD) $,
   ...
   ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
                                                                    //
                        DLAT = LATTD-LATAP $,
                        IF LONGAP LES 0.0
                        THEN IF (.999999 + LONGAP) LES LONGTD
                             THEN DLONG = -(.999999 - LONGTD) -
                              (.999999 + LONGAP)
                             ELSE DLONG = LONGTD - LONGAP
                        ELSE IF (.999999 - LONGAP) LES -LONGTD
                             THEN DLONG = (.999999 + LONGTD) +
                              (.999999 - LONGAP)
                             ELSE DLONG = LONGTD - LONGAP $,
                        LNGDIS = (DLONG*LATFIX)/(IRES*IPI) $,
                        LATDIS = DLAT/(IRES*IPI) $,
   ...
   ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
                                                                    //
                     RNGTD = RROOT2((LNGDIS*LNGDIS)+(LATDIS*LATDIS)) $,
   ...
   ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
                                                                    //
                     END                ... DISTANCE CALCULATIONS $,

        COMMENT CHECK MODE VALIDITIES $,
                     IF COMVAL AND SFLVAL AND LCFVAL
                     THEN LOCVAL = TRUF
                     ELSE LOCVAL = FALSE $,
                     IF COMVAL AND SFLVAL AND LCFVAL AND SFGVAL AND
                      ALTVAL
                     THEN AAVAL = TRUE
                     ELSE AAVAL = FALSE $,
                     IF APENG AND (LOCCAP OR LOCTRK)
                     THEN RNAVAL = FALSE
                     ELSE RNAVAL = TRUE $,
                                        ... END CHECK MODE VALIDITIES
                                            $,

        COMMENT BEGIN DETERMINING IF REQESTED MODE IS VALID AND SETTING
                ACTIVE MODE EQUAL TO REQUESTED MODE $,
                     IF ACTMDE NEQ REQMOD
                     THEN BEGIN
```

```
                           IF ((REQMOD EQL 1) AND RNAVAL)
                           THEN BEGIN
                                LOCARM = FALSE $,
                                NAVCAP = FALSE $,
                                LOCCAP = FALSE $,
                                GSARM = FALSE $,
                                GSCAP = FALSE $,
                                END     ... RESET ANNUNCIATORS FALSE $,
                           IF ((REQMOD EQL 1) AND RNAVAL) OR ((REQMOD
                            EQL 2) AND LOCVAL) OR ((REQMOD EQL 3) AND
                            AAVAL)
                           THEN ACTMDE = REQMOD $,
                                      ... END CHECKING REQUESTED
                                          MODE VALIDITIES $,
                      END $,

     COMMENT SERVICE COMPUTATIONS , SET FLAGS $,
                      IF ACTMDE NEQ 1
                      THEN  BEGIN
                            IF ((ACTMDE EQL 2) AND LOCVAL) OR ((ACTMDE
                             EQL 3) AND AAVAL)
                            THEN CMPFLG = TRUE
                            ELSE CMPFLG = FALSE $,

...
><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
                                                                       //
LOCCMP(RESETL,LOCDEV,BANK,LNGDIS,LATDIS,LOCARM,LOCCAP,
                                 LOCTRK,BKCMDA,BKCMDF)               $,
...
><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
                                                                       //

                            RESETL = FALSE $,
                            IF NOT LOCARM
                            THEN  BEGIN
                                  PHICA = BKCMDA $,
                                  PHICF = BKCMDF $,
                                  END    ... OUT LATERAL BANK CMDS $,
                            IF ACTMDE NEQ 2
                            THEN  BEGIN
                                  PTCMDA = VSAC $,
                                  PTCMDF = VSAC $,

...
><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
                                                                       //
GSCMP(GSDEV,RNGTD,ALTRTE,TIMD,RESETG,GSARM,GSCAP,LOCARM,PTCMDA) $,
...
><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
                                                                       //

                                  RESETG = FALSE $,
                                  IF NOT GSARM
                                  THEN 'BEGIN
                                        VSCCA = PTCMDA $,
                                        VSCCF = PTCMDF $,
```

```
                                        END ... OUT PITCH CMDS $,
                                  END $,
                            END         ... END OF PROCESSING STEERING
                                            COMPUTATIONS //
                      ELSE  BEGIN
                            RESETL = TRUE $,
                            RESETG = TRUE $,

END $,

...
<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                                                                        //
LOCAA = LOCARM  $,
LOCCA = LOCCAP  $,
GSAA = GSARM  $.
GSCA = GSCAP  $,
STATMD = 'CTMDE  $,
...
<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                                                                        //

COMMENT ***** CONTROL THE HSI TO/FROM FLAG BY ASSIGNING ONE OF THE
              FOLLOWING VALUES TO THE RNAV FLAG CALLED TOWARD

              TOWARD = 1 IF AIRCRAFT HAS NOT PASSED THE PERPENDICULAR
                         TO RWYHDG AS CONSTRUCTED AT TOUCHDOWN.
                     = 0 OTHERWISE .  ***** $'
IF LOCARM OR LOCCAP
THEN BEGIN
     TDPOS(0) = LATTD!
     TDPOS(1) = LONGTD !
     !LTBD(LOC LATAP, LOC TDPOS, LOC ACTOTD) !
     TOCRSDEV = RWYHDG - ACTOTD(BRG,TDTOAC) !
     TOWARD = IF ABS(TOCRSDEV) GRT DEG90
              THEN 1
              ELSE 0 !
     END !                    ... END OF LOGIC THAT CONTROLS THE
                                  HSI TO/FROM FLAG  //
 ... END OF SERVICING COMPUTATIONS, SETTING FLAGS $.

                END $,

          END FINI

#FIN
```

APPENDIX F

Program Listing Of
Localizer Routine

```
ILS*AFS.LOCCMP
      BEGIN

      COMMENT ***** THIS IS THE LOCALIZER COMPUTATION MODULE FOR THE 3D / 4D
              ILS PROGRAM ***** ;

      COMMENT ********** SCALED FRACTION VERSION ********** ;

      COMMENT ALONG-TRACK DISTANCE SCALED TO 16 FT ;

      COMMENT CROSS-TRACK DISTANCE SCALED TO 155 FT ;

      COMMENT CROSS-TRACK RATE SCALED TO 1000 FT / SEC ;

      COMMENT CROSS-TRACK ACCELERATION SCALED TO 100 FT / SEC**2 ;

      COMMENT AIRSPEED SCALED TO 1000 KNOTS ;

      COMMENT GROUNDSPEED SCALED TO 1000 KNOTS AND 1000 FT / SEC ;

      COMMENT ANGLES SCALED TO 180 DEGREES ;

      COMMENT LOCALIZER DEVIATION SCALED TO 300 MICROAMPS ;

      COMMENT ******************************************************** ;

      DEFINE REAL PROCEDURE LOCCMP(RESETL,RWYLTH,LOCDEV,BANK,MAGVAR,ACHDG,
       AIRSPD,GNDSPD,RWYHDG,GAMMA,RNGTD,LNGDIS,LATDIS,LOCARM,LOCCAP,LOCTRK,
       BKCMDA,BKCMDF) WHERE               ... //

      COMMENT ***** INPUT VARIABLES ***** ;
      BOOLEAN RESETL                      ... INITIALIZATION LOGIC FLAG ;
      REAL RWYLTH                         ... RUNWAY LENGTH (NM) ;
      REAL LOCDEV                         ... LOCALIZER DEVIATION FROM NAV DATA
                                              (MICROAMPS) ;
      REAL BANK                           ... ROLL ANGLE (RADIANS) ;
      REAL MAGVAR                         ... MAGNETIC VARIATION (RADIANS) ;
      REAL ACHDG                          ... MAGNETIC COMPASS HEADING (RADIANS) ;
      REAL AIRSPD                         ... AIRSPEED (KNOTS) ;
      REAL GNDSPD                         ... GROUNDSPEED (KNOTS) ;
      REAL RWYHDG                         ... MAGNETIC RUNWAY HEADING (RADIANS) ;
      REAL GAMMA                          ... TRUE LATERAL FLIGHT PATH ANGLE (TRACK
                                              ANGLE) - (RADIANS) ;
      REAL RNGTD                          ... RANGE TO TOUCHDOWN (NM) ;
      REAL LNGDIS                         ... LONGITUDE DISTANCE TO TOUCHDOWN (NM)
                                              ;
      REAL LATDIS                         ... LATITUDE DISTANCE TO TOUCHDOWN (NM) ;

      COMMENT ***** OUTPUT VARIABLES ***** ;
      BOOLEAN LOCARM                      ... LOC ARM LOGIC AND ANNUNCIATOR FLAG ;
      BOOLEAN LOCCAP                      ... LOC CAPTURE LOGIC AND ANNUNCIATOR
                                              FLAG ;
      BOOLEAN LOCTRK                      ... LOC TRACK LOGIC FLAG ;
      REAL BKCMDA                         ... BANK CMD FOR AUTOPILOT (RADIANS) ;
      REAL BKCMDF                         ... BANK CMD FOR FLIGHT DIRECTOR
                                              (RADIANS) //
       DONE
```

```
        BEGIN

  COMMENT ***** DESCRIPTION OF PROGRAM VARIABLES ***** ; ... //
                                    ... //
                                    ... ** ** * NAV CAPTURE VARIABLES ** ** *
                                    ... //
        BOOLEAN NAVCAP              ... NAV CAPTURE LOGIC FLAG ;
        BOOLEAN BLNDWN              ... NAV CAPTURE BANK BLEND DOWN COMMAND
                                        LOGIC FLAG ;
        REAL ILSLOK                 ... RANGE PARAMETER USED IN MODE
                                        GENERATOR FOR NAVCAP (FT) ;
        REAL ABSBNK                 ... ABSOLUTE VALUE OF BCMDNC (RADIANS) ;
        REAL BELCMP                 ... VARIABLE USED IN BEAM EDGE
                                        COMPUTATION ;
        REAL TALBE                  ... TRACK ANGLE ERROR AT BEAM EDGE
                                      . (RADIANS) ;
        REAL BCMDNC                 ... BANK CMD - NAV CAP (DEG) ;
        REAL KBLNDN                 ... GAIN ON NAV CAP BANK CMD BLEND DOWN ;
        REAL EXPARG                 ... EXPONENTIATION ARGUMENT OF KBLNDN ;
        REAL DELBEN                 ... FRACTIONAL MULTIPLIER ON TALBE IN
                                        KBLNDN COMPUTATION ; ... //
                                    ... //
                                    ... ** ** * ILS CAPTURE VARIABLES ** ** *
                                    ... //
        BOOLEAN FSTCAL              ... TRUE UNTIL ABS(LOCDRL) IS LESS THAN
                                        240. ;
        BOOLEAN ILSCAP              ... ILS CAPTURE LOGIC FLAG ;
        REAL BCMDBC                 ... BANK CMD - BEAM CAPTURE (ILS CAP)
                                        (DEG) ;
        REAL CTRBCN                 ... CROSS TRACK RATE - BEAM CAPTURE - NAV
                                        DATA (FT / SEC) ;
        REAL CTRBCI                 ... CROSS TRACK RATE - BEAM CAPTURE - ILS
                                        DATA (FT / SEC) ;
        REAL CTRBC                  ... CROSS TRACK RATE - BEAM CAPTURE (FT /
                                        SEC) ;
        REAL CTDBCU                 ... CROSS TRACK DISTANCE UNFILTERED -
                                        BEAM CAPTURE (FT) ;
        REAL CTDBC                  ... CROSS TRACK DISTANCE - BEAM CAPTURE
                                        (FT) ;
        REAL KFADBC                 ... GAIN ON FADER IN BEAM CAPTURE ;
        REAL TFADBC                 ... TIME CONSTANT OF FADER IN BEAM
                                        CAPTURE (SEC) ;
        REAL KYDBLN                 ... GAIN FOR BLENDING Y DOT FROM NAV DATA
                                        TO BEAM DATA ;
        REAL DELCAP                 ... OUTPUT OF BEAM CAPTURE FADER (DEG) ;
                                    ... //
                                    ... ** ** * COMMON CAPTURE VARIABLES **
                                        ** * //
        BOOLEAN DEVFLG              ... TRUE WHEN BEAM ANGLE IS LESS THAN 7
                                        DEGREES ;
        REAL CAPNAV                 ... CAPTRP PARAMETER COMPUTED FROM NAV
                                        DATA (FT) ;
        REAL CAPILS                 ... CAPTRP PARAMETER COMPUTED FROM ILS
                                        DATA (FT) ;
        REAL CAPTRP                 ... LOCALIZER CAPTURE TRIP POINT (FT) ;
        REAL CAPBNK                 ... THEORETICAL BANK USED IN CAPTURE
                                        (RADIANS) ;
```

```
REAL CPBKMX          ... MAX VALUE OF CAPBNK (RADIANS) ;
REAL REFROT          ... REFERENCE FRAME ROTATION ANGLE
                         (RADIANS) ;
REAL CTD             ... CROSS TRACK DISTANCE FROM NAV DATA
                         (NM) ;
REAL CTDNAV          ... CROSS TRACK DISTANCE FROM NAV DATA
                         (FT) ;
REAL TAE             ... TRACK ANGLE ERROR (RADIANS) ;
REAL VG              ... GROUNDSPEED (FT / SEC) ; ... //
                     ... //
                     ... ** ** * TRACK VARIABLES ** ** * //
                     ... //
REAL TRKTRP          ... LOCALIZER TRACK TRIP POINT
                         (MICROAMPS) ;
REAL ABSBC           ... ABSOLUTE VALUE OF BCMDLC (RADIANS) ;
REAL TAECMP          ... VARIABLE USED IN COMPUTATION OF
                         TAETRK ;
REAL TAETRK          ... THEORETICAL TRACK ANGLE ERROR AT
                         TRKTRP (RADIANS) ;
REAL BCMDTK          ... BANK CMD - TRACK (DEG) ;
REAL CTRTRK          ... CROSS TRACK RATE - TRACK (FT / SEC) ;
REAL CTDTRK          ... CROSS TRACK DISTANCE - TRACK (FT) ;
REAL KCTDTK          ... GAIN ON CROSS TRACK DISTANCE - TRACK
                         (DEG / FT) ;
REAL FCTDTK          ... FRACTIONAL MULTIPLIER IN GAIN ON
                         CROSS-TRACK DISTANCE ;
REAL KCTRTK          ... GAIN ON CROSS TRACK RATE - TRACK (DEG
                         / (FT / SEC)) ;
REAL FCTRTK          ... FRACTIONAL MULTIPLIER IN GAIN ON
                         CROSS-TRACK RATE - TRACK ;
REAL MAXRNG          ... MAXIMUM RANGE TO LINEARIZE TRACK LAWS
                         (FT) ;
REAL RNGPGM          ... GAIN ON OUTPUT OF RDSCE FOR TRACK ;
REAL KFADTK          ... GAIN ON FADER IN TRACK ;
REAL TFADTK          ... TIME CONSTANT OF FADER IN TRACK (SEC)
                         ;
REAL DELTRK          ... OUTPUT OF TRACK FADER (DEG) ;
REAL TBKWOT          ... BANK WASHOUT TIME CONSTANT (SEC) ;
REAL BNKWOL          ... BANK WASHOUT LIMIT (DEG) ;
REAL BANKLP          ... LO-PASSED BANK (DEG) ;
REAL WOIN            ... ELAPSED TIME AFTER LOCTRK THAT BANK
                         WASHOUT IS INSERTED (SEC) ;
REAL TRKTIM          ... ELAPSED TIME AFTER LOCTRK (SEC) ;
REAL FILFAD          ... GAIN ON FADER OF VFILC ;
REAL TRPCNT          ... COUNTER USED TO DETERMINE WHEN TO
                         TRIP IN INCREASED BANDWIDTH IN FILTER
                         C (SEC) ;
REAL FILTRP          ... NUMBER OF SECONDS AFTER TRACK WHEN
                         INCREASED BANDWIDTH IS TRIPPED INTO
                         FILTER C (SEC) ;
REAL VKFILC          ... VARYING BANDWIDTH PARAMETER OF FILTER
                         C ;
REAL DIFILC          ... VARYING PARAMETER USED IN COMPUTING
                         VKFILC ;
REAL FINALC          ... FINAL BANDWIDTH PARAMETER USED IN
                         FILTER C ;
REAL FADTIM          ... TIME CONSTANT OF THE INCREASED
```

```
                                             BANDWIDTH FADER USED IN FILTER C
                                             (SEC) : ... //
                                         ... //
                                         ... ** ** * COMMON ILS VARIABLES ** ** *.
                                             //
        REAL RNGLOC                      ... RANGE TO LOCALIZER (FT) : ... //
                                         ... //
                                         ... ** ** * COMMON VARIABLES ** ** * //
                                         ... //
        REAL LOCDRL                      ... LOCALIZER DEVIATION THROUGH RATE
                                             LIMITER (MICROAMPS) :
        REAL DLDV                        ... LOCDEV - LOCDPL (MICROAMPS) :
        REAL MXDLDV                      ... MAX LOCALIZER DEVIATION DIFFERENCE
                                             BETWEEN ITERATIONS (MICROAMPS) :
        REAL TIMSTP                      ... COMPUTATIONAL TIME INCREMENT (SEC) :
        REAL ACCLLC                      ... ACCELERATION FEED INTO FILTER C (FT /
                                             SEC) :
        REAL AFILC                       ... POLE - ZERO PARAMETER OF FILTER C :
        REAL KFILC                       ... INITIAL BANDWIDTH PARAMETER OF FILTER
                                             C :
        BOOLEAN CMPONE                   ... TRUE UNTIL FIRST GENERATION OF BKCMDA
                                             :
        REAL RATLIM                      ... RATE LIMIT ON BANK CMD (DEG / SEC) :
        REAL AMPLIM                      ... AMPLITUDE LIMIT ON BANK CMD (DEG) :
        REAL RATE                        ... MAXIMUM ALLOWABLE CHANGE OF BKCMDA
                                             (DEG) :
        REAL BKCMDS                      ... LAST VALUE OF BKCMDA, SAVED BEFORE
                                             LEAD-LAG (DEG) :
        REAL BKCMDI                      ... LAST VALUE OF BKCMDA OUTPUT BY LOPASS
                                             OF LEAD-LAG (DEG) :
        REAL DELCMD                      ... DIFFERENCE BETWEEN PRESENT BANK
                                             COMMAND AND LAST (DEG) :
        REAL TCLEAD                      ... TIME CONSTANT FOR THE LEAD ON BANK
                                             CMD (SEC) :
        REAL TCLAG                       ... TIME CONSTANT FOR THE LAG ON BANK CMD
                                             (SEC) : ... //
                                         ... //
        REAL SCAL1,SCAL2,SCAL3,SCAL4,SCAL5,SCAL6,SCAL7,SCAL8,SCAL9,SCAL10,
         SCAL11,SCAL12,SCAL13,SCAL14,SCAL15,SCAL16,SCAL17,SCAL18,SCAL19,
         COMP1,COMP2,COMP3,COMP4,COMP5,COMP6 ;
        REAL PROCEDURE LIMIT,LOPASS,NEXP ;
        PROCEDURE RHOCF ;                ... //
                                         ... //
        PRESET
             BEGIN
               TIMSTP = .2 ;
               KFADBC = .818731          ... EXP(-(TIMSTP / 1. )) :
               KFADTK = .960789          ... EXP(-(TIMSTP / 5. )) :
               FILFAD = .993356          ... EXP(-(TIMSTP / 30. )) :
               KCTDTK = .039375          ... 1. / ( .8 *((4 . / 100. ) / (70 . /
                                             1E5))) :
               KCTRTK = .360             ... 1. / ((35 . / 180. ) / (70 . / 1E3))
                                             :
               MAXKNG = .042532          ... (7 . / 100. )* .6076 :
               AMPLIM = .166667          ... 30. / 180. :
               RATLIM = .005556          ... (5 . *TIMSTP) / 180. :
               AFILC = .20               ... 2. / 10. :
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
                KFILC = .3036        ... 3.036 / 10. ;
                FILTRP = .50         ... 50. / 100. ;
                FINALC = .99999      ... 10. / 10. ;
                DELHLN = .5 ;
                SKRPOL = .011111     ... 2. / 180. ;
                KOIN = .10           ... 10. / 100. ;
                SCAL1 = .592417      ... 1. / 1.688 ;
                SCAL2 = .8           ... 240. / 300. ;
                SCAL3 = .007513      ... 300. / (75 . .57 .3 ) ;
                SCAL4 = .031082      ... 1. / 32.174 ;
                SCAL5 = .044444      ... 8. / 180. ;
                SCAL6 = .858544      ... 1. / ((17 . / 180. ) / (13 . .6076 .
                                         / 160)) ;
                SCAL7 = .138889      ... 25. / 180. ;
                SCAL8 = .027778      ... 5. / 180. ;
                SCAL9 = .666667      ... 200. / 300. ;
                SCAL10 = .046545     ... 2.667 / 57.3 ;
                SCAL11 = .5          ... 1. / TCLAG ;
                SCAL12 = .5          ... TCLAG ;
                SCAL13 = .005        ... TCLAG / 100. ;
                SCAL14 = .002        ... TIMSTP / 100. ;
                SCAL15 = .038889     ... 7. / 180. ;
                SCAL16 = .030        ... 9. / 300. ;
                SCAL17 = .76923      ... 1. / 1.3 ;
                SCAL18 = .078988     ... (13 . .6076 . ) / 165 ;
                SCAL19 = .043211     ... 7.778 / 180. ;
                END ;                ... //
                                     ... //
          IF RESETL
          THEN BEGIN
                LOCARM = TRUE ;
                LOCCAP = FALSE ;
                LOCTRK = FALSE ;
                ILSCAP = FALSE ;
                NAVCAP = FALSE ;
                BLNDWN = FALSE ;
                FSTCAL = TRUE ;
                CMPOUT = TRUE ;
                DEVFLG = FALSE ;
                BCRUNC = .0 ;
                KELAUN = .0 ;
                DELCAP = .0 ;
                DELTRK = .0 ;
                BKCPDS = .0 ;
                TPPCNT = .0 ;
                TRKTIM = .0 ;
                VKFILC = KFILC ;
                DIFILC = FINALC-KFILC ;
                DALKLP = .0 ;
                TALFL = .0 ;
                TALTRK = .0 ;
                LOCDRL = LOCDEV ;
                END ;                ... //
                                     ... //

      COMMENT ******************************************************** ;
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
COMMENT THE FOLLOWING CARDS MUST BE REMOVED BEFORE CROSS-COMPILING ;

COMMENT ************************************************************ ;
        REAL PROCEDURE NSIN,NCOS,NTAN ;
        LOCDEV = LOCDLV/300. ;
        IF LOCDEV GEQ 1.
        THEN LOCDEV = .99999 ;
        IF LOCDEV LEQ -1.
        THEN LOCDEV = -.99999 ;
        IF RESETL
        THEN LOCDRL = LOCDEV ;
        BANK = BANK/3.1415927 ;
        MAGVAR = MAGVAR/3.1415927 ;
        ACHDG = ACHDG/3.1415927 ;
        AIRSPD = AIRSPD/1000. ;
        GNDSPD = GNDSPD/1000. ;
        RWYHDG = RWYHDG/3.1415927 ;
        GAMMA = GAMMA/3.1415927 ;
        RNGTD = RNGTD/100. ;
        LNGDIS = LNGDIS/100. ;
        RWYLTH = RWYLTH/10. ;
        LATDIS = LATDIS/100. ;
        SCAL11 = 1. ;
        SCAL12 = 1. ;
        SCAL13 = .01 ;

COMMENT ************************************************************ ;

COMMENT THE ABOVE CARDS MUST BE REMOVED BEFORE CROSS-COMPILING ;

COMMENT ************************************************************ ;

COMMENT ********** DATA_PROCESSING_MODULE ********** ; ... //

COMMENT ***** RANGE TO LOCALIZER COMPUTATION ***** ;
        RNGLOC = (RNGTD+RWYLTH*.1)*.6076 ;

COMMENT ***** TRACK ANGLE ERROR COMPUTATIONS ***** ;
        TAE = GAMMA-(RWYHDG+MAGVAR) ;

COMMENT ***** CHANGE GROUNDSPEED FROM KNOTS TO FT / SEC ***** ;
        VG = GNDSPD/SCAL1 ;

COMMENT ***** RATE LIMIT ON LOCDLV ***** ;
        MXOLDV = ((ABS(TIMSTP*VG*NSIN(TAE)/SCAL3)*.001)/RNGLOC)/SCAL17 ;
        IF MXOLDV LES SCAL16
        THEN MXOLDV = SCAL16 ;
        DLDV = LOCDEV-LOCDRL ;
        LOCDRL = LOCDRL-LIMIT(DLDV,MXOLDV) ; ... //
                                    ... //
        IF NOT LOCTRK
        THEN BEGIN                  ... //
                                    ... //
            BNKLOT = .225+.015*(RNGLOC/.006076) ;
            IF RNGLOC GRT SCAL18
            THEN TRKTRP = .1*SCAL18/RNGLOC
            ELSE TRKTRP = .1 ;
```

```
COMMENT ***** NAV CROSS TRACK DISTANCE COMPUTATIONS ***** ;
                REFROT = RWYHDG*.5 ;
                CTD = LATDIS*NCOS(REFROT)+LNGDIS*NSIN(REFROT) ;
                CTDNAV = (CTD*.6076)/.1 ;
                END                          ... //
                                             ... //
        ELSE BEGIN

COMMENT ***** BANK WASHOUT ***** ;
                IF TRKTIM GEQ WOIN
                THEN BEGIN
                        BANKLP = LOPASS(BANK,BANKLP,BNKKOT,SCAL1N) ;
                        BANKLP = LIMIT(BANKLP,BNKWOL) ;
                        BANK = BANK-BANKLP ;
                        END
                ELSE TRKTIM = TRKTIM+TIMSTP*.01 ;
                END ;

COMMENT *********** LOCALIZER DATA FILTERING *********** ; ... //
                                             ... //
        IF ((ABS(LOCCRL) LEQ SCAL2) OR ( NOT FSTCAL)) AND DEVFLG
        THEN BEGIN
                CTRBCN = -VG*NSIN(TAE) ;
                CTDBCU = RNGLOC*LOCCRL*(SCAL3/.1) ;
                IF LOCTRK
                THEN BEGIN
                        IF TRPCNT GEQ FILTRP
                        THEN BEGIN
                                DIFILC = DIFILC*FILFAD ;
                                VKFILC = FINALC-DIFILC ;
                                END
                        ELSE TRPCNT = TRPCNT+TIMSTP*.01 ;
                        END ;
                ACCELC = -((NSIN(BANK)/NCOS(BANK))*NCOS(TAE))*(.01/SCAL4) ;
                IF FSTCAL
                THEN BEGIN
                        CTDBC = CTDBCU ;
                        CTRBCI = CTRBCN ;
                        END
                ELSE RWDCF(CTDBCU,AFILC,VKFILC,TIMSTP,CTDBC,CTRBCI,ACCELC) ;
                KYDBLN = NCOS(TAE) ;
                CTRBC = CTRBCN*(.99999-KYDBLN)+CTRBCI*KYDBLN ;
                FSTCAL = FALSE ;
                END
        ELSE BEGIN
                CTDBC = CTDNAV ;
                CTRBC = -VG*NSIN(TAE) ;
                CTRBCI = CTRBC ;
                END ;

COMMENT ***** CAPTURE TRIP POINT COMPUTATION ***** ;
        IF LOCARM
        THEN BEGIN
                CPMRKX = SCAL5*(RNGLOC/SCAL6) ;
                IF CPMRKX GRT SCAL7
                THEN CPMRKX = SCAL7 ;
```

```
          CAPBNK = (CPBKMX/.5)*TAE ;
          IF ABS(CAPBNK) LES SCAL8
          THEN CAPBNK = (SCAL8*TAE)/ABS(TAE) ;
          CAPNAV = (((VG**2)*(.99999-NCOS(TAE))*SCAL4)/(NSIN(CAPBNK)/
           NCOS(CAPBNK)))/.1 ;
          CAPILS = ((CTRLCI**2/.1)*SCAL4*NCOS(CAPBNK))/(NSIN(CAPBNK)*
           .99999*NSIN(CAPBNK)*NCOS(TAE)) ;
          IF ABS(LOCURL) LEQ SCAL9
          THEN CAPTRP = CAPILS
          ELSE CAPTRP = CAPNAV ;
          CAPTRP = CAPTRP+(CAPBNK*VG*.01)/(RATLIM/TIMSTP) ;
          END ;                   ... //

COMMENT ***** MODE GENERATOR ***** ; ... //
                               ... //
     IF NOT LOCTRK
     THEN BEGIN
          IF ((ABS(CTDNAV*.1)/RNGLOC)*.31831 LEQ SCAL15) OR DEVFLG
          THEN DEVFLG = TRUE ;
          IF LOCARM
          THEN BEGIN
               IF (ABS(LOCURL) LEQ SCAL9) AND DEVFLG
               THEN BEGIN
                    IF ABS(CTODC) LEQ ABS(CAPTRP) ... TEST FOR ILS
                                   CAPTURE //
                    THEN BEGIN
                         LOCARM = FALSE ;
                         ILSCAP = TRUE ;
                         NAVCAP = FALSE ;
                         END
                    END
               ELSE BEGIN     ... TEST FOR NAV CAPTURE //
                    IF ABS(TAE) LES SCAL7
                    THEN ILSLOK = (ABS(TAE*(VG-.125))/.63015)+.02127
                    ELSE ILSLOK = (VG*(ABS(TAE)-.03233))/.53722 ;
                    IF ((ABS(CTDNAV) LES ABS(CAPTRP)) AND (RNGLOC LEQ
                     ILSLOK))
                    THEN BEGIN
                         LOCARM = FALSE ;
                         NAVCAP = TRUE ... SET NAVCAP ;
                         END ;
                    END ;
               END ;
          IF NAVCAP
          THEN BEGIN
               IF ABS(TAE) LEQ .95*TAEBE ... TEST FOR BLNDWN //
               THEN BLNDWN = TRUE ;
               IF (ABS(LOCURL) LEQ SCAL9) AND DEVFLG ... TEST FOR ILS
                                   CAPTURE //
               THEN BEGIN
                    ILSCAP = TRUE ;
                    NAVCAP = FALSE ;
                    END ;
               END ;
          IF LOCARM OR ILSCAP
          THEN BEGIN
               IF (ABS(LOCURL) LEQ TRKTRP) OR (ABS(TAE) LEQ .9*TAEBEN)
```

```
                                        ... TEST FOR TRACK //
                          THEN BEGIN
                             ILSCAP = FALSE ;
                             LOCTRK = TRUE ;
                             LOCARM = FALSE ;
                             LOCCAP = TRUE ;
                             END ;
                          END ;
                      LOCCAP = ILSCAP OR NAVCAP OR LOCCAP ;
                      END ;

       COMMENT ***********_CONTROL COMPUTATIONS ********** ; ... //
                                        ... //
              IF NOT LOCARM
              THEN BEGIN

       COMMENT INTERMEDIATE TRACK COMPUTATIONS ;
                      IF RNGLOC LES MAXRNG
                      THEN RNGPGM = .99999
                      ELSE RNGPGM = MAXRNG/RNGLOC ;
                      CTDTRK = RNGPGM*CTDBC ;
                      CTRTRK = RNGPGM*CTRBCI ;
                      BCMDTK = CTDTRK/KCTDTK+CTRTRK/KCTRTK ;
                      IF NOT LOCTRK
                      THEN BEGIN
                           COMP3 = ((CTRBC**2)/.1)*SCAL4 ;
                           COMP4 = CTDBC*.99999+CTDBC*NCOS(TAE) ;
                           IF CTDBC LES .0
                           THEN BEGIN
                                COMP3 = -COMP3 ;
                                COMP4 = -COMP4 ;
                                END ;
                           BCMDBC = -NTAN(COMP3,COMP4) ;
                           IF NOT ILSCAP
                           THEN BEGIN
                                ABSBC = ABS(BCMDBC) ;
                                TAECMP = .99999-(((NSIN(ABSBC)/NCOS(ABSBC))*RNGLOC*
                                 TRKTRP*SCAL3)/(VG**2)) ;
                                COMP6 = SQRT(.99999-TAECMP**2) ;
                                TAETRK = .5-NTAN(TAECMP,COMP6) ; ... TAETRK =
                                                ACOS(TAECMP) //
                                IF TAETRK LES SCAL19
                                THEN TAETRK = SCAL19 ;
                                END ;

       COMMENT NAV CAP COMPUTATIONS ;
                           IF NAVCAP
                           THEN BEGIN
                                IF NOT BLNDWN
                                THEN BEGIN
                                     COMP2 = ((VG**2)/.1)*(.99999-NCOS(TAE))*SCAL4
                                      ;
                                     COMP5 = CTDNAV ;
                                     IF CTDNAV LES .0
                                     THEN BEGIN
                                          COMP2 = -COMP2 ;
                                          COMP5 = -CTDNAV ;
```

```
                             END ;
                        BCMONC = -ATAN(COMP2,COMP3) ;
                        IF CMPONE ... COMPUTE TAEBE ON FIRST ITERATION
                                 OF NAVCAP //
                        THEN BEGIN
                             ABSBNK = ABS(BCMONC) ;
                             BECOMP = .99999-((((KKIN(ABSBNK)/KCOS(
                              ABSBNK))+KKGLOC*SCAL15)/SCAL4)/(VG**2))
                             ;
                             COMP1 = SQRT(.99999-BECOMP**2) ;
                             TAEBE = .5-ATAN(BECOMP,COMP1) ; ... TAEBE
                                 = ACOS(BECOMP) //
                             END ;
                        EXPARG = -TIMSTP*.001/((VG.DELDLN*TAEBE*SCAL4)
                         /ABS(BANK)) ;
                        KBLNON = BEXP(EXPARG) ;
                        END
                   ELSE BCMONC = BCMONC*KBLNON ;
                   BKCMDA = BCMONC ;
                   DELCAP = BCMONC-BCMDBC ;
                   END
              ELSE BEGIN     ... //
                             ... FINAL ILS CAPTURE COMPUTATIONS //
                   DELCAP = DELCAP*KFADBC ;
                   BCMONC = BCMDBC+DELCAP ;
                   BKCMDA = BCMONC ;
                   DELTRK = BCMDBC-BCMDTK ;
                   END
              END
          ELSE BEGIN         ... //
                             ... FINAL TRACK COMPUTATIONS //
              DELTRK = DELTRK*KFADTK ;
              BCMDTK = BCMDTK+DELTRK ;
              BKCMDA = BCMDTK ;
              END ;

COMMENT ********** OUTPUT CONTROL PROCESSING ********** ; ... //
                             ... //
                             ... BANK CMD AND CMD RATE LIMIT
                                 COMPUTATIONS //
          DELCMD = BKCMDA-BKCMDS ;
          RATE = LIMIT(DELCMD,RATLIM) ;
          BKCMDA = BKCMDS+RATE ;
          BKCMDA = LIMIT(BKCMDA,AMPLIM) ;
          BKCMDS = BKCMDA ;
          IF CMPONE
          THEN BEGIN
               BKCMDI = BKCMDA ;
               CMPONE = FALSE ;
               END ;
          BKCMDA = ((BKCMDA/SCAL11)/SCAL12)+LOPASS(BKCMDA,BKCMDI,SCAL13
           ,SCAL14)-((LOPASS(BKCMDA,BKCMDI,SCAL13,SCAL14)/SCAL11)/
           SCAL12)           ... PASS BKCMDA THROUGH (2S + 1) / ( .5S
                                 + 1) ; ... //
                             ... //
          END ;
```

```
         COMMENT ************************************************************ ;

         COMMENT THE FOLLOWING CARDS MUST BE REMOVED BEFORE CROSS-COMPILING ;

         COMMENT ************************************************************ ;
              BKCMSA = BKCMLA*3.141592 ;
              RWYHDG = RWYHDG*3.1415927 ;
              RWYLTH = RWYLTH*10. ;

         COMMENT ************************************************************ ;

         COMMENT THE ABOVE CARDS MUST BE REMOVED BEFORE CROSS-COMPILING ;

         COMMENT ************************************************************ ;
                                       ... //
                                       ... //
              END ;

         END FINI
```

```
ILS*AEC.KROCF
        BEGIN

        DEFINE PROCEDURE KROCF(INPUT,PZ,FRQ,TIMSTP,POSITN,RATE,ACCEL)
         WHERE                              ... //
         REAL INPUT,PZ,FRQ,TIMSTP,POSITN,RATE,ACCEL ... //
         TORE
            BEGIN
            REAL TSQ,TOK,TOKSQ,A11,A12,A21,A22,B11,B12,B21,B22,LSTPOS,LSTRAT ;
            REAL PZPARM,FRQNRM ;
            PZPARM = .1/PZ ;
            FRQNRM = .1/FRQ ;
            TSQ = TIMSTP**2 ;
            TOK = TIMSTP*FRQNRM ;
            TOKSQ = TOK**2 ;
            A11 = .99999-(TOKSQ*.5) ;
            A12 = -(TOK*FRQNRM)*(.99993-((TOK*.5)/PZPARM)) ;
            A21 = -A12/(FRQNRM**2) ;
            A22 = (PZPARM*FRQNRM*A11+A12)/(PZPARM*FRQNRM) ;
            B11 = -A12 ;
            B12 = (PZPARM*FRQNRM*A21+(.99999-A11))/(PZPARM*FRQNRM) ;
            B21 = .99999-A22 ;
            B22 = (.99999-A11)/(FRQNRM**2) ;
            LSTPOS = POSITN ;
            LSTRAT = RATE ;
            RATE = A11*LSTRAT+((A12*LSTPOS)/.02)/.5+((B11*INPUT)/.02)/.5+(B12*
             ACCEL)*.1 ;
            POSITN = (A21*LSTRAT)*.01+A22*LSTPOS+B21*INPUT+(B22*ACCEL)*.001 ;
            END ;

        END FINI
```

```
INS*ACC.LIMIT
        BEGIN

        DEFINE REAL PROCEDURE LIMIT(INPUT,LIMVAL) WHERE ... //
         REAL INPUT,LIMVAL                       ... //
         TONE
             BEGIN
             IF ABS(INPUT) LES LIMVAL
             THEN LIMIT = INPUT
             ELSE LIMIT = (LIMVAL*ABS(INPUT))/INPUT ;
             END ;

        END FINI
```

```
ILS*AED.LOPASS
        BEGIN

        DEFINE REAL PROCEDURE LOPASS(INPUT,LSLOPS,STC,STIMST) WHERE ... //
         REAL INPUT,STC,STIMST,LSLOPS ... //
         IONE
            BEGIN
            LSLOPS = LSLOPS+(INPUT-LSLOPS)*(STIMST/STC) ;
            LOPASS = LSLOPS ;
            END ;

        END FINI
```

```
ILS*AEC.NEXP
     BEGIN
     DEFINE REAL PROCEDURE NEXP(X) WHERE ... //
      REAL X                           ... //
      FORE
          BEGIN
          X = -X ;
          NEXP = .9999998-.9990604*X+.4982929*X*X-.15253x2*X*X*X+.0223641*X*
           X*X*X ;
          END ;

     END FINI
```

APPENDIX G

Program Listing Of Glideslope Routines

```
)*WORKFILE.GSCFGDA1
                BEGIN

                DEFINE PROCEDURE GSCMP(GSDEV,D,HDOT,T,RESETG,GSARM,GSCAP,
                 LOCARM,PTCMDA) WHERE                ... ...
                                                     ****************************
                                                     ***********************
                                                     INPUTS
                                                     ****************************
                                                     ******************** //
                REAL D                               ... DISTANCE TO TOUCHDOWN 100
                                                     NM FULL SCALE (+)
                                                     APPROACHING THE RNWY :
                REAL PTCMDA                          ... EXTERNALLY SUPPLIED
                                                     STEERING COMMAND,100 DEG
                                                     FULL SCALE ***NOTE*** THE
                                                     INPUT VALUE WILL BE
                                                     OVERWRITTEN IF AND ONLY IF
                                                     THIS ROUTINE IS IN THE
                                                     GSCAP STATUS. + IMPLIES
                                                     PITCH DOWN :
                BOOLEAN LOCARM                       ... TRUE PRIOR TO LOC CAPTURE :
                BOOLEAN RESETG                       ... INITIALIZATION COMMAND :
                                                     ... GSCMP IS SET TO THE ARM
                                                     STATUS AND ALL FILTERS ARE
                                                     INITIALIZED *** NOTE ***
                                                     THIS BOOLEAN SHOULD BE
                                                     EXTERNALLY SET FALSE AFTER
                                                     THE FIRST CALL TO GSCMP //
                REAL T                               ... TIME SINCE LAST CALL TO
                                                     GSCMP .. INTERNAL
                                                     COMPUTATIONS ASSUME THE
                                                     NOMINAL SAMPLE INTERVAL ON
                                                     THE FIRST ACCESS 10.604 HRS
                                                     FULL SCALE :
                REAL GSDEV                           ... GLIDESLOPE RADIO DEVIATION
                                                     300 U-AMPS FULL SCALE (+)
                                                     ABOVE THE BEAM :
                REAL HDOT                            ... VERTICAL RATE 500 FEET /
                                                     SEC FULL SCALE (+) UPWARD :
                                                     ...
                                                     ****************************
                                                     ***********************
                                                     OUTPUTS
                                                     ****************************
                                                     ******************** //
                BOOLEAN GSARM                        ... GLIDESLOPE COMPUTATION
                                                     STATUS :
                BOOLEAN GSCAP                        ... GLIDESLOPE COMPUTATION
                                                     STATUS //
                 TOBE
                     BEGIN

           COMMENT ****** GAINS CONSTANTS AND CONVERSION FACTORS ****** :
                     REAL PROCEDURE LOPASS,WSHOUT,LIMIT :
                     REAL KY                         ... DIVIDED BY 10) EQUALS
                                                     DEGREES OF PITCH COMMAND /
```

```
                                                        FOOT OF LINEAR DEVIATION OR
                                                        MULTIPLIED BY MAXRNG /
                                                        .2016 EQUALS DEGREES OF
                                                        PITCH COMMAND PER U-AMP
                                                        BEYOND MAXRNG NM. ;
                    REAL KHDOT                      ... DEGREES OF PITCH COMMAND /
                                                        FT PER SEC OF VERTICAL RATE
                                                        ;
                    REAL KYDOT                      ... MULTIPLIED BY
                                                        KHDOT*KDIS*MAXRNG / .2016
                                                        EQUALS DEGREES OF PITCH
                                                        COMMAND PER U-AMP / SEC OR
                                                        MULTIPLIED BB KHDOT / 10.
                                                        EQUALS DEGREES OF PITCH
                                                        COMMAND PER FT / SEC OF
                                                        LINEAR DEVIATION RATE ;
                    REAL MAXRNG                     ... DISTANCE BEYOND WHICH
                                                        DEVIATION (DELTAH) IS
                                                        EXPRESSED IN U-AMPS (SCALE
                                                        IDENTICAL TO RNGTD) ;
                    REAL DYTOFT                     ... NM(DISTANCE) -SMALL
                                                        DOTS(DEVIATION) PER FOOT OF
                                                        DEVIATION A 2.5 DEGREE
                                                        GLIDESLOPR IS ASSUMED ;
                    REAL GAIN10                     ... REAL GAIN10 ...DIVISION BY
                                                        THIS CANSTANT WILL YIELD A
                                                        GAIN OF 10 ;
                    REAL GAIN5                      ... DIVIDE BY FOR GAIN OF 5 ;
                    REAL GAIN.5                     ... MULTYPLY BY GAIN.5 FOR GAIN
                                                        OF .5 ;
                    REAL ARRAY SCALE(1)             ... AND ARRAY OF SCALE FACTORS
                                                        ;
                    REAL DELTAT                     ... SAMPLE INTERVAL 1000. SEC
                                                        FULL SCALE ;
                    REAL TNOM                       ... NOMINAL SAMPLE INTERVAL
                                                        SAME SCALE AS DELTAT ;
                    REAL TC1                        ... FILTER TIME CONSTANT ;
                    REAL TC2                        ... FILTER TIME CONSTANT ;
                    REAL TC3                        ... FILTER TIME CONSTANT SCALE
                                                        THE SAME AS DELTAT ;
                    REAL TC4                        ... FILTER TIME CONSTANT ;
                    REAL TC5                        ... LAG TIME CONSTANT ;
                    REAL ONE                        ... UNITY ;
                    REAL ZERO                       ... 0.0 ;

            COMMENT ***** INTERNAL VARIABLES ***** ;
                    REAL LIN2                       ... PREVIOUS FILTER INPUT ;
                    REAL LIN3                       ... PREVIOUS FILTER INPUT ;
                    REAL LIN4                       ... PREVIOUS FILTER INPUT ;
                    REAL LOUT1                      ... PREVIOUS FILTER OUTPUT ;
                    REAL LOUT2                      ... PREVIOUS FILTER OUTPUT ;
                    REAL LOUT3                      ... PREVIOUS FILTER OUTPUT ;
                    REAL LOUT4                      ... PREVIOUS FILTER OUTPUT ;
                    REAL LOUT5                      ... PREVIOUS FILTER OUTPUT ;
                    REAL DELTAH                     ... LINEAR FEET TO THE
                                                        GLIDESLOPE BEAM //
                                                    ... OR ( BEYOND MAXRNG ) THE
```

```
                                            ANGULAR ERROR IN THE FLIGH
                                            PATH ANGLE WITH RESPECT TO
                                            THE GLIDESLOPE //
                                        ... DIVIDE MAXRNG BY DYTOFT TO
                                            GET FULL SCALE FEET - - -
                                            NOM 1000 FOR 3.76 NM AND A
                                            2.5 DEGREE GLIDESLOPE ;
           REAL YDOUT                   ... RADIO DEV RATE ( BEYOND
                                            MAXRNG ) OR CONVERGENCE
                                            RATE ( INSIDE MAXRNG ) FULL
                                            SCALE U - AMPS / SEC =
                                            MAXRNG / DYTOFT OR FULL
                                            SCALE FT / SEC = MAXRNG /
                                            DYTOFT ;
           REAL DEVRTE                  ... BEAM CLOSURE RATE U-AMPS /
                                            SEC OR FT / SEC ;
           REAL PTCMD                   ... PITCH COMMAND 100 DEGREES
                                            FULL SCALE. + IMPLIES
                                            PITCH DOWN ;
           REAL MAXCMD                  ... ILS PITCH COMMAND LIMIT 100
                                            DEGREES ;
           REAL TMAX                    ... LARGEST RECOGNIZED TIME
                                            STEP ;
           REAL TMIN                    ... SMALLEST RECOGNIZED TIME
                                            STEP ;
           REAL TEMP                    ... INITIAL COMPOSITE ILS AND
                                            EXTERNAL PITCH COMMAND FOR
                                            TRANSITION TO TOTAL ILS
                                            CONTROL ;
           REAL Y                       ... FILTERED RADIO DEVIATION ;
           REAL HDOTLL                  ... FILTERED ALTITUDE RATE ;
           REAL LDBRFR                  ... LEAD BREAK FREQUENCY RAD/SE

COMMENT ********** COMPILATION VALUES ********** ;
           PRESET
                 BEGIN
                 GAIN.5 = .5 ;
                 GAIN5 = .2 ;
                 GAIN10 = .1 ;
                 TNOM = .0002 ;
                 DYTOFT = .1008 ;
                 KHDOT = .15 ;
                 KY = .75 ;
                 KYDOT = .66 ;
                 MAXRNG = .0556 ;
                 SCALE(0) = .5667 ;
                 SCALE(1) = .025707 ;
                 LDBRFR = .5 ;
                 TC1 = .0005 ;
                 TC2 = .002 ;
                 TC3 = .02 ;
                 TC4 = .005 ;
                 TC5 = .001 ;
                 TMAX = .00025          ... SET TO ONE HALF OF THE
                                            SMALLEST FILTER TIME
                                            CONSTANT ;
                 TMIN = .0001           ... SET TO AT LEAST THE
```

```
                               RESOLUTION OF THE OBJECT
                               MACHINE OR A MINIMUM CYCLE
                               TIME IF KNOWN ;
            MAXCMD = .1 ;
            ONE = .99999999 ;
            ZERO = .0 ;
            END ;

COMMENT **********EXECUTABLE PROGRAM ********** ;
        IF RESETG                   ... INITIALIZE STATE
                               LOGIC,RADIO LOWPASS FILTER
                               AND THE TIME INTERVAL //
        THEN BEGIN
            DELTAT = TNOM ;
            GSARM = TRUE ;
            GSCAP = FALSE ;
            HDOTLL = LOUT5 = HDOT ;
                                ... INITIALIZE LEAD-LAG ON
                               HDOT //
            Y = LOUT1 = GSDEV $, ... INITIALIZE FILTERED RADIO
                               DEVIATION TO AVOIDE FALSE
                               CAPTURES //
            END
        ELSE BEGIN
            DELTAT = T/SCALE(1) ; ... CONVERT TO 1000 SEC FULL
                               SCALE //
            IF DELTAT GEQ TMAX OR DELTAT LEQ TMIN
            THEN DELTAT = TNOM ... LIMIT SAMPLE INTERVAL ;
            HDOTLL = HDOT-(LOPASS(HDOT,LOUT5,TC5,DELTAT)*LDBRFR
                ;
            HDOTLL = HDOTLL/LDBRFR ;
                                ... LEAD-LAG FILTERED HDOT //
            Y = LOPASS(GSDEV,LOUT1,TC1,DELTAT) ... FILTER
                               RAW DEVIATION ;

            END ;

COMMENT **** ESTIMATE THE DISTANCE TO THE BEAM ***** ;
        DELTAH = Y*
        IF D GEQ MAXRNG             ... BEYOND MAXRNG THE DEVIATION
                               IS ESTIMATED WITH RESPECT
                               TO ANGULAR ERROR //
        THEN MAXRNG
        ELSE D                      ... INSIDE MAXRNG DELTAH IS
                               EQUAL TO THE LINEAR
                               DISTANCE TO THE BEAM IN
                               FEET ;
        DELTAH = DELTAH/DYTOFT      ... UNITS CONVERSION ;

COMMENT ***** DERIVE BEAM CLOSURE RATE ***** ;
        IF RESETG
        THEN BEGIN
            LIN2 = DELTAH $,
            LOUT2 = ZERO            ... INITIALIZE RATE DERIVER TO
                               CURRENT INPUT AND ZERO
                               OUTPUT //
            END $,
```

```
          IF GSARM
          THEN YDOT = WSHOUT(DELTAH,LOUT2,TC2,DELTAT,LIN2)*GAIN.5
                                  ... AFTER THE CAPTURE CONDITION
                                      HAS BENN MET CLOSURE RATE
                                      WILL BE COMPUTED FROM HDOT
                                      ALONE ;
          DEVRTE = KYDOT*YDOT/GAIN10 ;
          DEVRTE = (HDOTLL+DEVRTE)/GAIN5 ;
          IF GSCAP
          THEN DEVRTE = WSHOUT(DEVRTE,LOUT3,TC3,DELTAT,LIN3)
                                  ... WASHOUT THE YDOT BIAS AND
                                      THE STEADY STATE VALUE OF
                                      HDOT //
          ELSE LIN3 = LOUT3 = DEVRTE ... ...INITIALIZE THE WASHOUT
                                      TO EXACTLY DEVRTE ;

COMMENT ***** COMPUTE THE PITCH COMMAND ***** ;
          PTCMD = KY*DELTAH       ... FORWARD FEED ;
          PTCMD = PTCMD+(DEVRTE*KHDOT) ... RATE DAMPENING ;
          PTCMD = LIMIT(PTCMD,MAXCMD) ... LIMIT THE COMMAND SIZE ;
          PTCMD = PTCMD*SCALE(0)    ... CONVERT TO 180 DEGREES FULL
                                      SCALE ;

COMMENT ***** TEST FOR CAPTURE ***** ;
          IF GSARM
          THEN BEGIN
               GSCAP = ((Y LEQ ZERO) AND (PTCMD GEQ ZERO)) OR ((Y
                GEQ ZERO) AND (PTCMD LEQ ZERO)) ... TEST FOR
                                      CAPTURE ;
               IF LOCARM THEN GSCAP = FALSE ; ... WAIT FOR LOC //
               GSARM = NOT GSCAP ;
               TEMP = LIN4 = LOUT4 = PTCMDA-PTCMD ... PREPARE TO
                                      TRANSITION TO ILS //
               END ;

COMMENT ***** OUTPUT THE PITCH STEERING COMMAND ***** ;
          IF GSARM
          THEN                    ... DO NOT OVERWRITE EXTERNAL
                                      COMMAND //
           PTCMDA = PTCMDA
          ELSE PTCMDA = PTCMD+WSHOUT(TEMP,LOUT4,TC4,DELTAT,LIN4)
                                  ... FADE OUT EXTERNAL COMMAND ;
          END ;

     END FINI
```

RT.S D.LIMKGDA1

```
D*WORKFILE.LIMRGOA1
               BEGIN

               DEFINE REAL PROCEDURE LIMIT(INPUT,LIMVAL) WHERE REAL LIMVAL,
                                                ... LARGEST ALLOWABLE OUTPUT
                                                    MAGNITUDE //
                INPUT                           ... INPUT VALUE ; 1CEF
                    BEGIN
                    REAL TEMP ;
                    IF INPUT LES .0 THEN TEMP = -INPUT
                    ELSE TEMP = INPUT ;
                    IF TEMP GRT LIMVAL          ... LIMIT TEST //
                    THEN LIMIT = LIMVAL*TEMP/INPUT ... SATURATION //
                    ELSE LIMIT = INPUT ;
                    END ;

               END FINI
FIN
```

```
@D.WORKFILE.%SHRGDA1
              BEGIN

              DEFINE REAL PROCEDURE %SHGOUT(INPUT,LSTOUT,TC,T,LASTIN)
               WHERE REAL INPUT                 ... CURRENT INPUT SIGNAL :
              REAL LASTIN                       ... LAST INPUT SIGNAL ;
              REAL LSTOUT                       ... LAST OUTPUT SIGNAL :
              REAL TC                           ... FILTER TIME CONSTANT :
              REAL T                            ... REAL T ...SAMPLE INTERVA
                                                .., NOTE TC AND T MUST HAVE
                                                THE SAME SCALING //
               TOBE
                  BEGIN
                  REAL K1,K2 ;
                            K1 = TC/(TC+.5*T) ...            FOR T = 0. TO
                                                TC / 2 ;
                            K2 = (TC-.5*T)/(TC+.5*T) ;
                  LSTOUT = K2*LSTOUT+(INPUT-LASTIN)*K1 ;
                  %SHGOUT = LSTOUT ;
                  LASTIN = INPUT ;
                  END ;

              END FINI

@PRT,S D.LIMRGDA1
```

```
@D*WORKFILE.LOPRGDA1
                 BEGIN

                 DEFINE REAL PROCEDURE LOPASS(INPUT,LSTOUT,TC,T)
                  WHERE REAL INPUT                    ... CURRENT INPUT SIGNAL ;
                 REAL LSTOUT                          ... LAST OUTPUT SIGNAL ;
                 REAL TC                              ... FILTER TIME CANSTANT ;
                 REAL T                               ... REAL T ...SAMPLE INTERVAL ;
                                                      ... NOTE TC AND T MUST HAVE THE
                                                          SAME SCALING //
                 TOBE
                      BEGIN
                      REAL K1,K2 ;
                                 K1 = T/TC ;
                                 K2 = (TC-T)/TC ;
                      LSTOUT = K1*INPUT+K2*LSTOUT ;
                      LOPASS = LSTOUT ;
                      END ;

                 END FINI

@PRT,S  D.WSHRGDA1
```

LIST OF REFERENCES

1. AGM-329, "Use of Area Navigation Computations to Aid in Localizer Capture," by T. G. Sharpe and W. A. Savold dated 21 June 1972.
2. Digital Flight Director Proposal (523-0764380-00111P), Section 2.2, "Initial Synthesis of Control Laws", by T. G. Sharpe dated 26 April 1972.
3. AGM-501, "Second Order Filter for Localizer Data to Improve Performance of Capture and Track Laws", by T. G. Sharpe dated September 4, 1975.
4. AGM-497, "Area Nav Aided Localizer Captures for 3D/4D Program, Part II: Software Development and Simulation Results", by L. H. Hogle dated 28 August 1975.
5. FAA Advisory Circular, "Automatic Landing System (ALS)", dated 12 January 1971.