REPORT NO. DOT-TSC-OST-74-14. VB

AUTOMATION APPLICATIONS IN AN ADVANCED AIR TRAFFIC MANAGEMENT SYSTEM

Volume VB: DELTA Simulation Model - Programmer's Guide

F. Mertes
K. Willis
E.C. Barkley

AUGUST 1974

FINAL REPORT

Prepared for

U.S. DEPARTMENT OF TRANSPORTATION
OFFICE OF THE SECRETARY
Office of the Assistant Secretary
for Systems Development and Technology
Office of Systems Engineering
Washington DC 20590

NOTICE

This document is disseminated under the sponsorship of the Department of Transportation in the interest of information exchange. The United States Government assumes no liability for its contents or use thereof.

Technical Report Documentation Page

1. Report No. DOT-TSC-OST-74-14. VB	2. Government Accession No.	3. Recipient's Catalog No. PB 236 810
4. Title and Subtitle AUTOMATION APPLICATIONS IN AN ADVANCED AIR TRAFFIC MANAGEMENT SYSTEM Volume VB: DELTA Simulation Model - Programmer's Guide		5. Report Date August 1974
		6. Performing Organization Code
7. Author(s) F. Mertes, K. Willis, E.C. Barkley		8. Performing Organization Report No. DOT-TSC-OST-74-14. VB
9. Performing Organization Name and Address TRW Incorporated* Westgate Research Park 7600 Colshire Drive McLean VA 22010		10. Work Unit No. (TRAIS) PPA-OS404/R4509
		11. Contract or Grant No. DOT-TSC-512-5b
12. Sponsoring Agency Name and Address U.S. Department of Transportation Office of the Secretary Office of the Ass't. Sec. for Sys. Dev. & Tech. Office of Systems Engineering Washington DC 20590		13. Type of Report and Period Covered Final Report Nov. 72 to Jan. 74
		14. Sponsoring Agency Code
15. Supplementary Notes *Under contract to: U.S. Department of Transportation Transportation Systems Center Kendall Square Cambridge MA 02142		

16. Abstract

The Advanced Air Traffic Management System (AATMS) program is a long-range investigation of new concepts and techniques for controlling traffic and providing services to the growing number of commercial, military, and general aviation users of the national airspace. This study of the applications of automation was undertaken as part of the AATMS program. The purposes were to specify and describe the desirable extent of automation in AATMS, to estimate the requirements for man and machine resources associated with such a degree of automation, and to examine the prospective employment of humans and automata as air traffic management is converted from a labor-intensive to a machine-intensive activity.

Volume V describes the DELTA Simulation Model. It includes all documentation of the DELTA (Determine Effective Levels of Task Automation) computer simulation developed by TRW for use in the Automation Applications Study. Volume VA includes a user's manual, test case, and test case results. Volume VB includes a programmer's manual

PRICES SUBJECT TO CHANGE

17. Key Words Automation, Air Traffic Control, Human Factors, Computers, Computer Architecture, Man-Machine Interface, Advanced ATC System, Manpower Requirements, Failure Analysis		18. Distribution Statement DOCUMENT IS AVAILABLE TO THE PUBLIC THROUGH THE NATIONAL TECHNICAL INFORMATION SERVICE, SPRINGFIELD, VIRGINIA 22151.	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages	22. Price

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

CONTENTS - VOL. VB

LIST OF FIGURES - VOL. VB

LIST OF FIGURES - Vol. VB
(Continued)

LIST OF TABLES - VOL. VB

1.0 INTRODUCTION

This is Volume V-Book II of a five volume report produced for the Automation Applications Study of the Advanced Air Traffic Management System. This volume describes the DELTA (Determine Effectiveness Levels for Task Allocation) digital computer simulation model. The volume is divided into two Books: Book I - User's Guide and Book II - Programmer's Guide.

The User's Guide is designed to permit an analyst to understand what the model simulates and how to exercise the model. It discusses the concepts from which the model was constructed. A complete input data specification and deck setup is described to show how cases are set up. A test case is provided to show how a case would look. Several auxilliary programs which are used to produce input files are described along with the Post Processor Program which is used to produce an additional output report for the DELTA model.

The Programmer's Guide is designed to give a programmer insight as to how the DELTA model was built and to enable him to make extensions or modifications to the model's code. There is a general discussion of the program structure and a description of the link structure and the system considerations.

There are detailed descriptions of the subroutines and the dynamic memory file structures. Samples of overlay structures are given and a list of subroutines which reference the files is shown.

2.0 PROGRAM STRUCTURE

2.1 General Discussion

In this documentation a "run" refers to an exercise of the model for which the input data remains fixed. Each approach to the computer is called a "job" and will usually be synonomous with run for this model. The complete input data set for a run specifies a "case".

The computer software for the DELTA model is composed of a program with three major components: input processor, simulation, and report generator, plus three separately executable programs: Terminal Generation Program, Interarrival Times Generation Program, and Post Processor. The input processor component sets up SALSIM files, allocates dynamic memory, reads scenario data and performs limited validity checking on the data. The simulation component simulates the interaction between resources and service requestors in an Air Traffic Management environment and writes a history tape. The report generator component summarizes information available during the execution of the run. The Post Processor program sorts information on the history tape and produces an additional report. The Terminal General Program is used to produce a random access data file used by the simulation. The Interarrival Times Generation Program is used as an aid in defining flight plans to stimulate the model. The separately executable programs are described in specific sections of both books of this volume.

2.1.1 Input Processor

That data which describes the SALSIM parameters and the scenario are read by the input processor. The SALSIM routine XXSTRT allocates the dynamic memory and sets up the bookkeeping for the logic control chains which will be executed. The routine INITIA sets up the bookkeeping for the user's dynamic memory files and reads the scenario data: namelist, task allocations and times, aircraft types, jurisdiction and resource descriptions, and initiates the tasks which are performed cyclically.

2.1.2 Simulation

The execution of the simulation component is controlled by SIMRUN which alternately transfers control between EXOG and ELIST. EXOG reads the input file for exogenous events which are used to stimulate various tasks in the model, while ELIST calls the Logic Control Chain (LCC, subroutine) which is next in the time queue to be performed. SIMRUN determines when it is appropriate to read the next exogenous event and updates the simulation clock after all tasks for a particular time have been executed.

The LCC's are called, in general, directly by ELIST. If one LCC is supposed to cause another LCC to begin execution, then it will create an "event notice" for that LCC and file it in the simulation time queue. When the appropriate time occurs for the LCC to be executed, then ELIST calls the LCC. The LCC immediately calls ALLOC to get a resource allocated to perform the task. If none is available, the task is put into a queue to be performed when a resource can be assigned; otherwise, the resource is set busy and the task is performed.

There are two types of tasks: cyclical (time stimulated) and event stimulated. The cyclical tasks are executed at some regular frequency during the run and stimulate themselves. Event stimulated tasks occur either exogenously - externally stimulated, or endogenously - internally stimulated. An exogenous event, read from the input tape by EXOG, can cause an entire chain of events to occur. For example, "filing a flight plan" is an exogenous event which can generate events throughout the air traffic management system until the aircraft either lands or flies out of the geography being modelled. A typical run combines the cyclical tasks, which appear as "background" consumers of resources, and the user generated tasks, which are initiated by input.

A run is terminated by input of an exogenous event to call the run summary at some specified game time. The simplest type of run would be one where there were no external events other than the termination. This would give a measure of how many resources were being consumed by background tasks.

2.1.3 Report Generator

The DELTA model has a run summary which presents information on resource utilization for the run which was calculated during execution. The data is broken down by jurisdiction, resource pool and element. The summary also lists statistics on the utilization of the dynamic memory.

2.2 Subroutine and Link Structure

Figure 2.2-1 shows the basic subroutine linkage of the DELTA model. In general the figure is read from left to right, then top to bottom. Since all 165 functional analysis task subroutines are called directly or indirectly by ELIST, the figure merely references the function number. Also the algorithms are shown only by their primary subroutine name; their structures are shown in the following figures. Figure 2.2-2 shows the structure of the algorithm which moves aircraft, simulates error and performs boundary crossings. The metering and sequencing algorithm is shown in Figure 2.2-3. Figure 2.2-4 illustrates the calling structure of the hazard evaluation and conflict resolution algorithm.

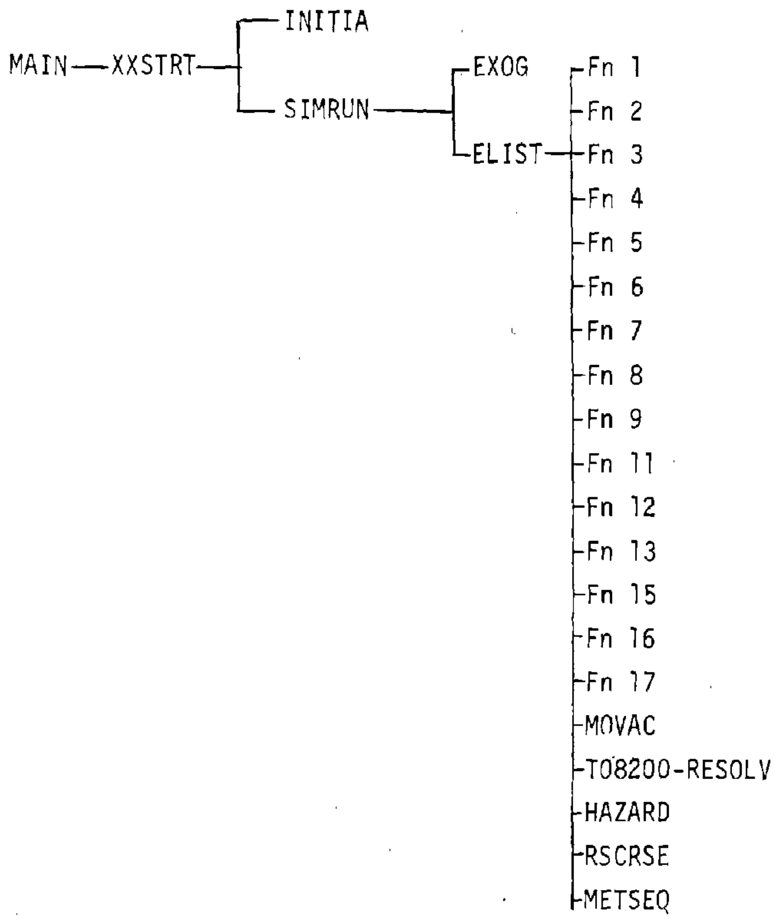

Figure 2.2-1 Basic Linkage of DELTA Model

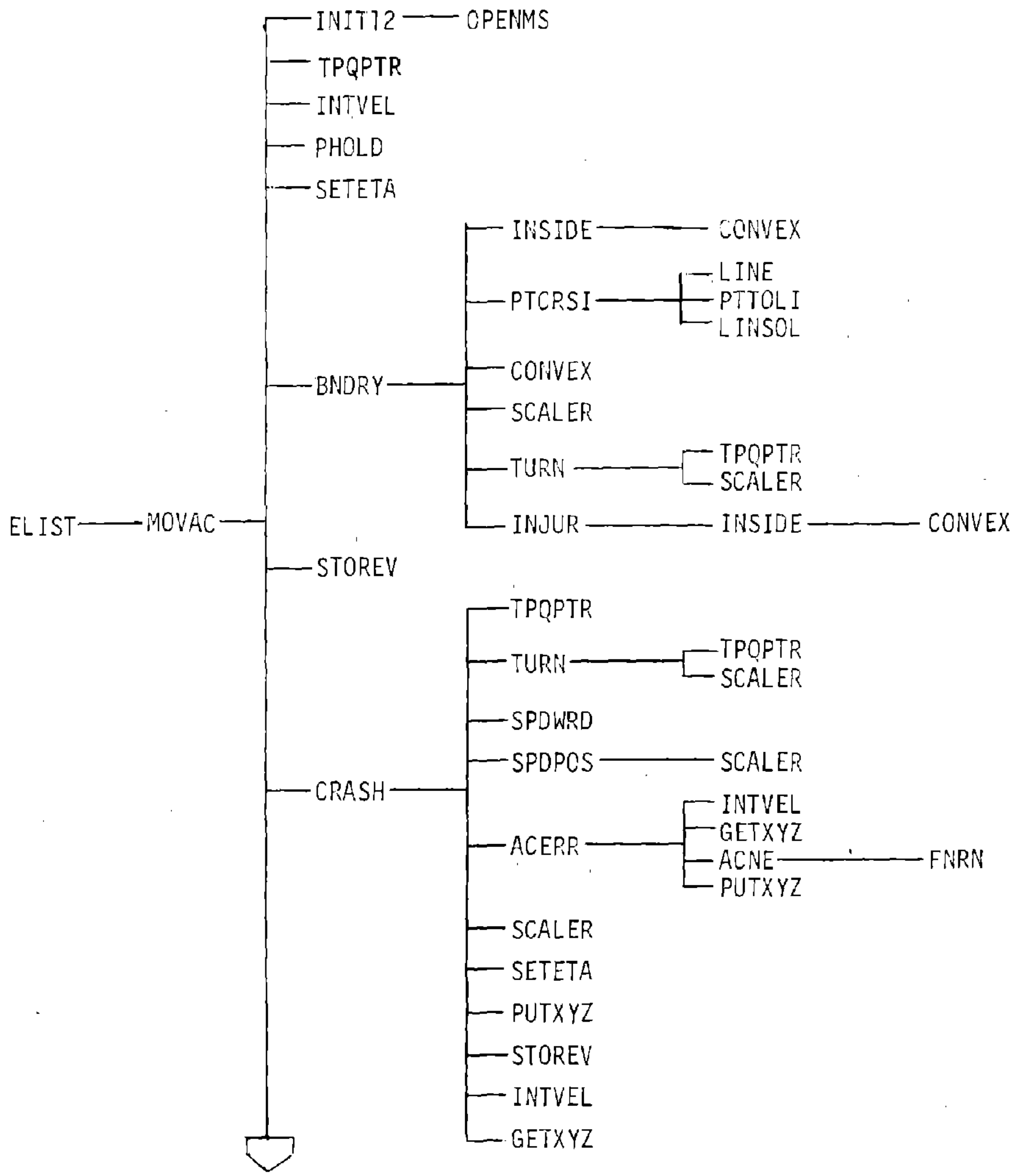

Figure 2.2-2 Linkage Of Kinematics Algorithm

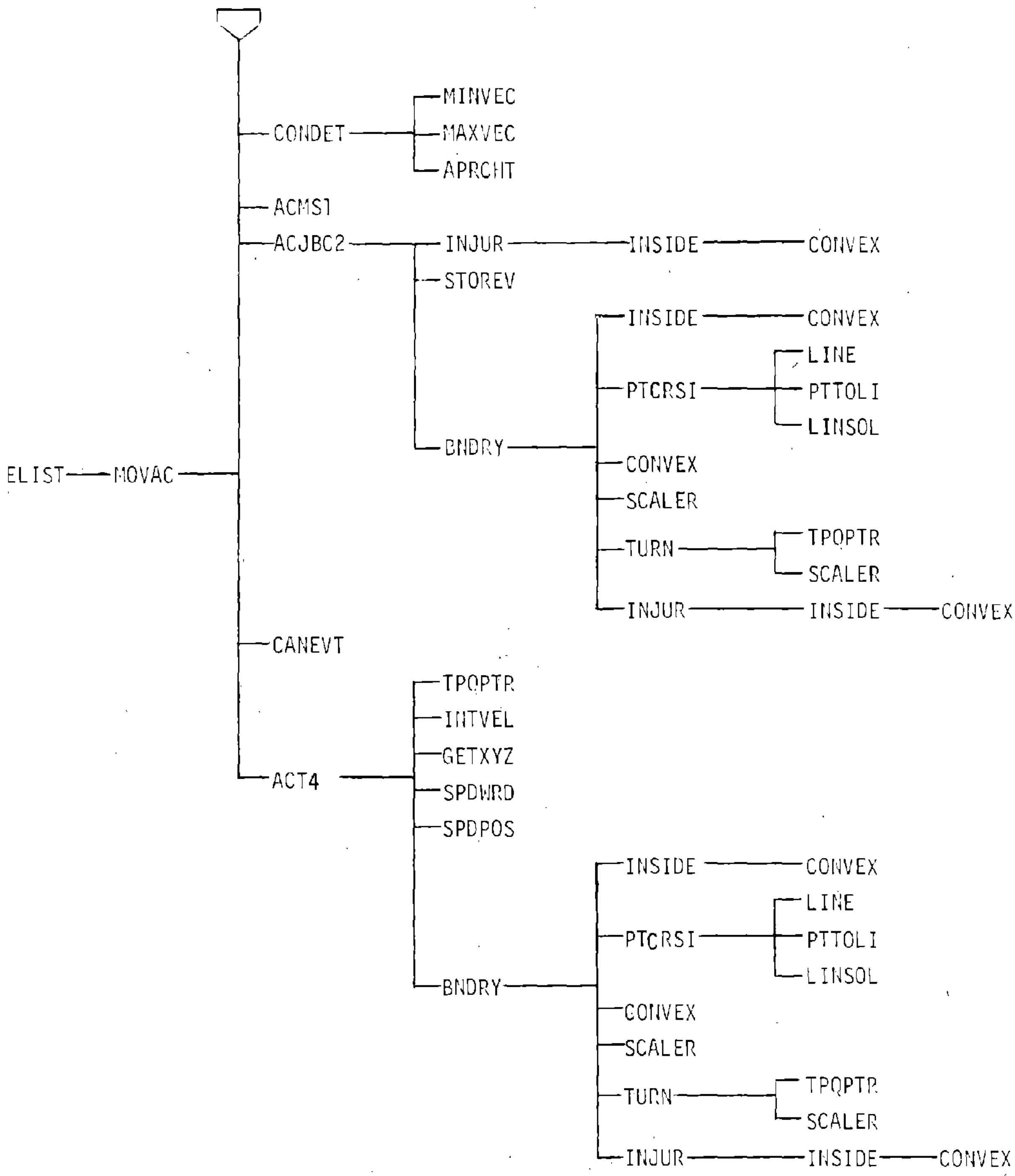

Figure 2.2-2 continued

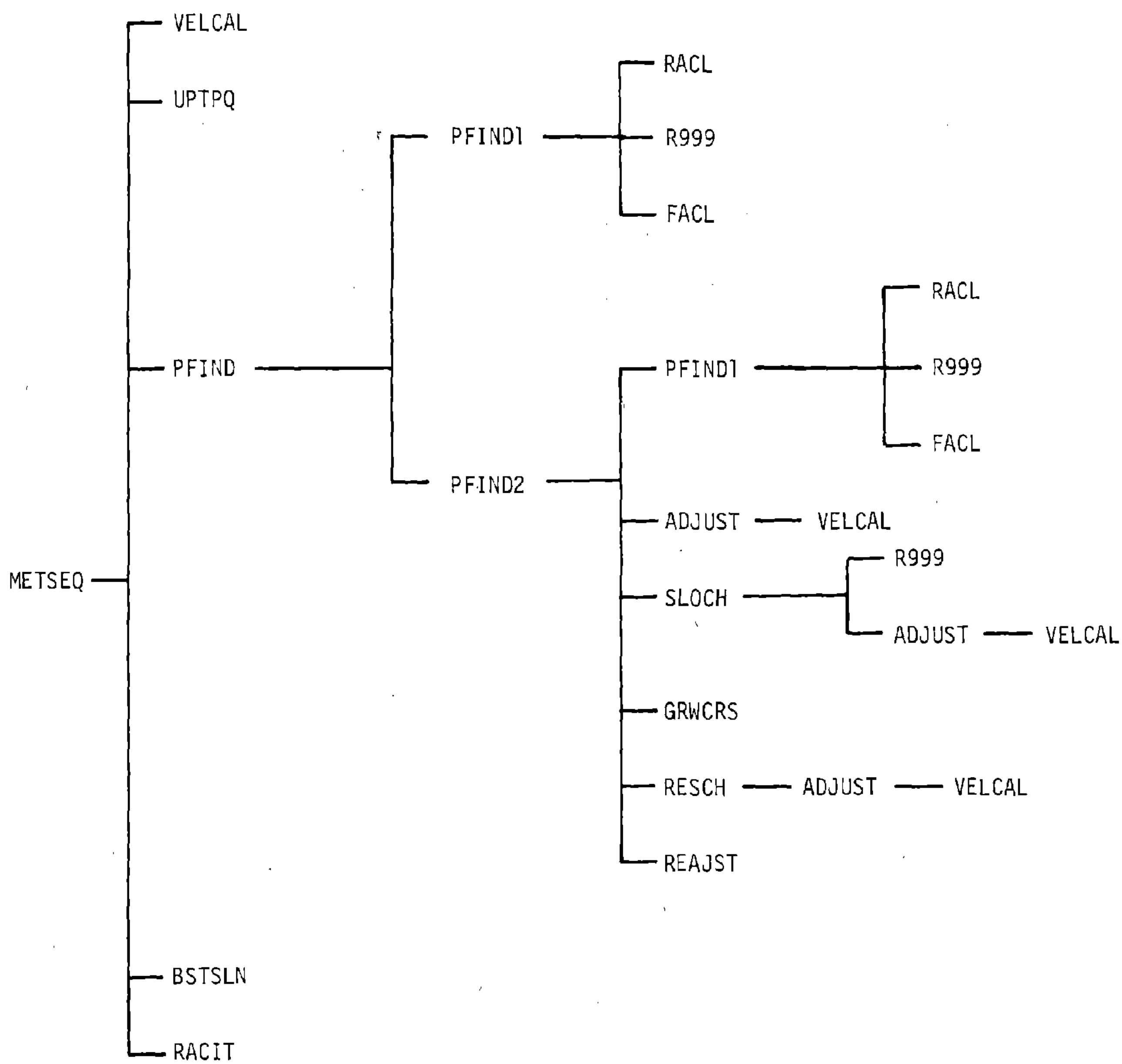

Figure 2.2-3 Linkage Of Metering and Sequencing Algorithm

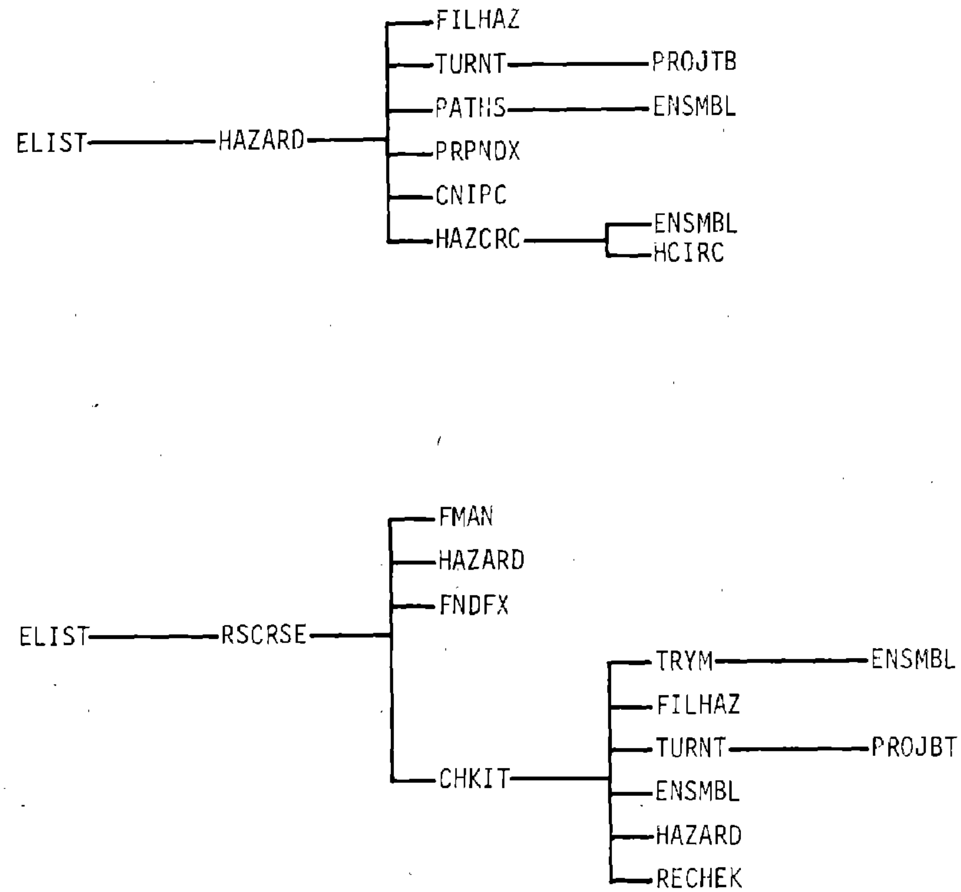

Figure 2.2-4 Linkage Of Hazard Evaluation and Conflict Resolution Algorithms

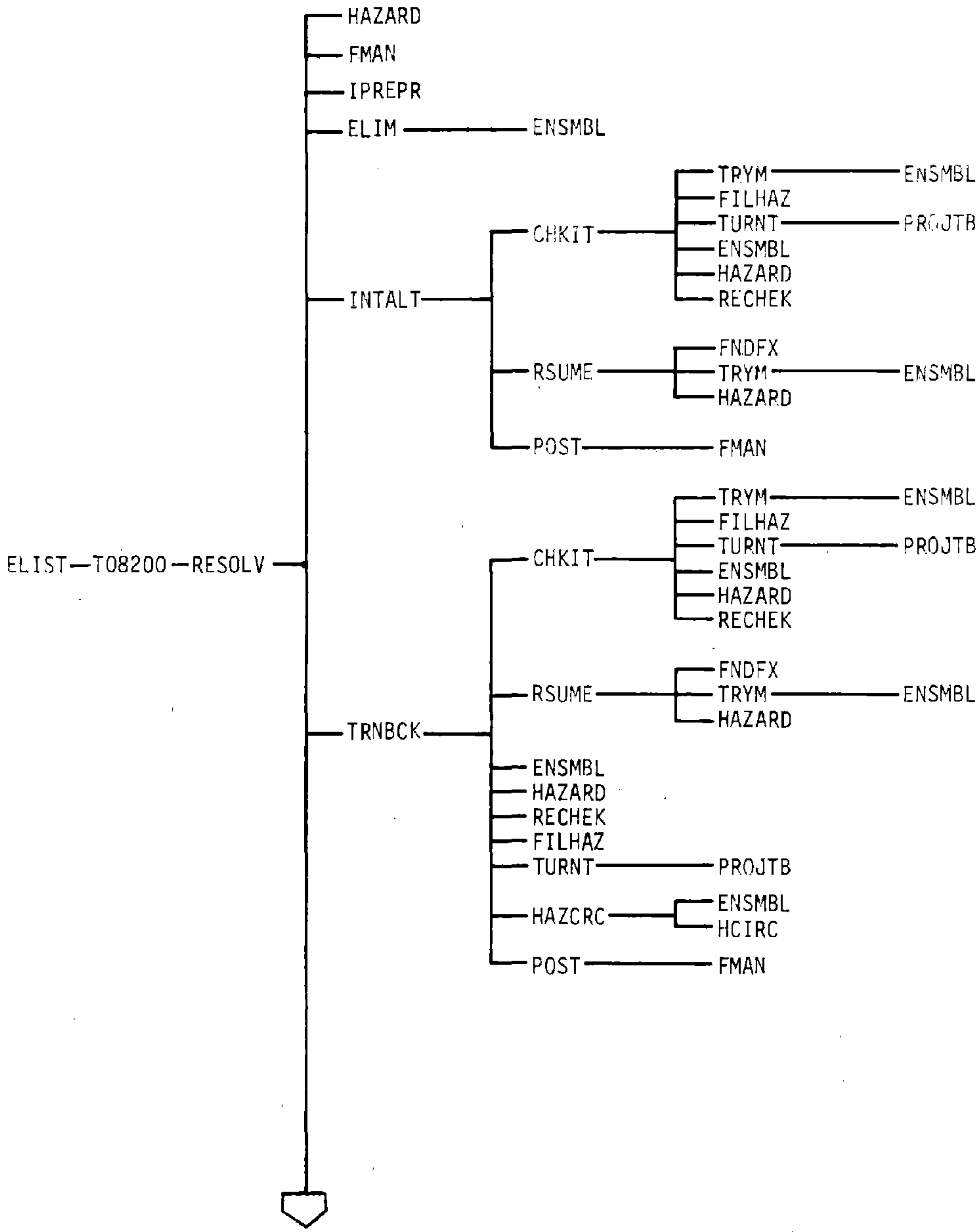

Figure 2.2-4 continued

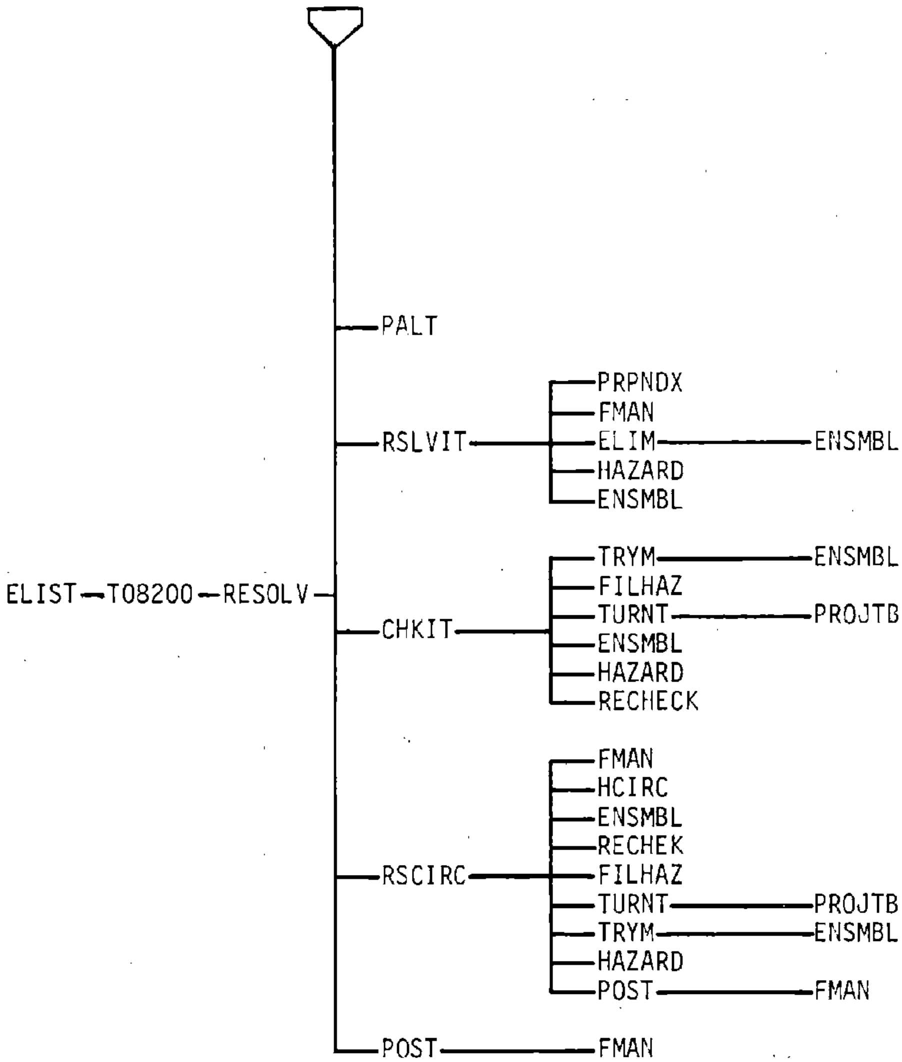

Figure 2.2-4 continued

2.3 System and Hardware Considerations

The DELTA Model software was designed to be compiled and executed on a CDC 6600 compatible system using a MACE operating system. Elsewhere in this volume is a detailed account of the job control cards required to execute the model on the TRW in-house configuration. As described in Section 3.1, the model uses some CDC and TRW-unique software. The model routines all compile using the RUNX compiler.

The software making up the DELTA Model requires a large amount of core if no overlays are used. During the debug phase of development, an overlay structure was configured permitting the model to fit within 60,000 octal words of memory. (Note that the core sizes are indicated in octal numbers, following the CDC conventions.) This was done at the expense of the dynamic memory and the input/output buffers. During model exercise, a more optimal overlay was used, requiring 117,000 octal words. This overlay is described in Section 5. The overlays are designed to utilize the Automatic Overlay Loading feature of the CDC Loader.

In order to run DELTA on other systems, several software modifications must be made. Of primary importance is the sixty bit word length found on the CDC systems discussed above. This word length permits fifteen significant digits on real numbers. This is especially important with respect to the model's internal clock, which must maintain flight durations of multiple hours, manual task times in tenths of seconds and automated task times in nanoseconds. Here the sixty bits is of greatest value.

Several instances can be found within the MØVAC logic of the use of the sixty bits as holders of three twenty bit words, such as normalize error terms or pointers within dynamic memory. However, these conditions are in the minority, since the model was written to be as hardware independent as possible. The MAIN routine is also specific to the CDC system, and would require modification should the system be changed. See Section 3.3.

The format statements are another area which would require modification. The CDC FORTRAN permits the use of the asterisk (*) for delimiting literals within Format statements and the use of Ø-formats to read and write octal variables. These must be changed. However, these areas appear to be only system dependencies in the model software.

3.0 SUBROUTINE DESCRIPTIONS

3.1 Introduction

The DELTA Model is composed of five types of routines. These are the following:

A. FORTRAN or system library routines, such as SIN and ABS.

B. SALSIM utilities, such as CAUSE and DELAY.

C. Model utility routines, such as ALLØC and TSELEC.

D. Algorithms, such as METSEQ and MØVAC.

E. Functional analysis routines, T01101 through T17300.

These routines combine to form the simulation program. Most of the routines are written in FORTRAN IV. The software developed for this model (types C, D and E) is heavily commented to eliminate the need for detailed flow charts or narrative descriptions. This section is intended to provide the programmer with a basic background for understanding the program. The descriptions are intended to amplify those in the User's Guide, Book I of this volume.

3.1.1 System Library Routines

It is necessary to discuss several of the routines of the first type, those supplied in the FORTRAN Library of Subroutines or in the TRW Subroutine Library. Most of the routines that belong to type A are common to all FORTRAN Packages. There are five exceptions, however. These are routines that are peculiar to CDC FORTRAN or the TRW Subroutine Library. These routines are described below.

LBYT - This routine is used to extract the values of bits within a word. It will extract from one to sixty bits from a word. The routine is part of the TRW Subroutine Library.

SBYT - This routine is used to set the values of bits within a word. It will set from one to sixty bits of a word. The routine is part of the TRW Subroutine Library.

UPR1 - This routine is a uniform pseudo random number generator. It is called by function subprogram RANF. UPR1 is part of the TRW Subroutine Library. It is similar to IBM's RANDU subprogram in many respects.

FNRN - This routine is a normal random number generator. It is part of the TRW Subroutine Library.

Finally, the direct access input/output routines used in the model are specific to the CDC FORTRAN Library of Subroutines. These are ØPENMS, CLØSEMS, READMS and WRITMS. Since there is no standard direct access I/O package defined for FORTRAN, it is common for each manufacturer to use his own I/O routines.

3.1.2 SALSIM Utilities

The remaining types of routines will be discussed below. The SALSIM utilities are FORTRAN subroutines which facilitate the construction of the event stepped features of the simulations. For example, they permit the use of dynamic memory, providing efficient core allocation. They provide for an internal model clock, interaction between exogenously and endogenously stimulated events and event scheduling. These routines are general purpose in nature.

3.1.3 Model Utilities

The model utilities are specifically designed for the DELTA Model. They are intended to facilitate the special-purpose processing common to many of the routines in the model. Five of these routines deserve special mention. These are the routines most directly concerned with resource utilization: ALLØC, GETASK, GETIME, DELAY and TSELEC.

ALLØC is used to assign a resource element to a task. If no resource element is available, this routine attempts to increase the capability of the resource pool (either by incrementing the number of controllers for a manual pool or by increasing the computer's speed for an automated pool), if the pool is not at maximum capability. Failing this, ALLØC will place the task in the Task Queue.

GETASK is a routine used to obtain all pertinent information on the allocation and task time for a particular task. It is used by ALLØC, GETIME, TSELEC and other utilities. Given the LCC Number (see Table 3.2-3), GETASK searches the task data files and returns the desired data.

GETIME is used to determine the amount of time required for the performance of a task. It uses GETASK to select the distribution or number of instructions, and then picks a time from the distribution or determines the length of time from the number of instructions and the computer's processing rate. This time is used by DELAY as the length of time to interrupt processing and the time is credited to the resource element performing the task.

TSELEC is used to relieve the resource element from its current assignment at completion, and attempt to reassign it to either the next task in a task chain or to the highest priority task in the Task Queue. If there are no further demands on the resource element, it is placed in a not busy status and returned to the resource pool.

This set of model utilities is the basic requirement for the treating of resource utilization within the DELTA Model. It will be discussed below.

3.1.4 Algorithms

There are three principle algorithms used in this model. They are responsible for the motion of the aircraft through the system. The first of these is MØVAC, which contains the aircraft kinematics, the navigation error model, and the boundry crossing logic. The scheduling and interleaving of arrivals and departures, metering and sequencing, is performed by the METSEQ algorithm. Separation assurance, conflict detection, hazard evaluation, and conflict resolution are performed by CØNDET algorithm. The analysis on which these algorithms is founded is described in the User's Guide, Book I of this volume. The resulting routines are described briefly below.

3.1.5 Functional Analysis Routines

The functional analysis resulted in the generation of about two hundred routines. The functional analysis is described in great detail in Volume II. The resulting software is described briefly below.

Of the seventeen functions defined by the functional analysis, Functions 10. and 14. were not modelled. In order to simplify some of the programming, some functional analysis tasks were grouped into single routines. Except for these routines, there is a direct relationship between functional analysis tasks and functional analysis routines.

3.2 Conventions

The DELTA Model employs several programming conventions or standard practices which, during model development and testing, permitted a high degree of commonality among the routines. This is especially true of the functional analysis routines. These conventions will be described in this section to provide the programmer with greater insight into the model construction.

3.2.1 Subroutine Naming Convention

As discussed in the previous section, there is a close relationship between the functional analysis described in Volume II and the functional analysis routines. To foster this relationship, a naming convention was adopted in which the task name is embedded in the subroutine name. Thus, task one, of subfunction one, of function one (Task 1.1.1) is simulated by a routine named T01101. In general, the routine name is of the form, Tffstt, where ff is the function number, s is the subfunction number and tt is the task number. Table 3.2-1 contains a complete list of the routines of this sort.

This table also contains the Logical Control Chain (LCC) number for each routine which is an LCC. The LCC is a SALSIM-based convention. It represents a triggerable routine, as opposed to those which are simply FORTRAN subprograms. This difference will be described in a discussion of the SALSIM utility routines, Section 3.4. The use of dynamic memory and the dynamic memory file structures will be discussed in Section 4. Dynamic memory is another SALSIM-based programming convention.

3.2.2 SALSIM Conventions

The use of the model and SALSIM utility routines is another convention of the DELTA Model. It imposes a great deal of similarity among the task-simulating routines. Figure 3.2-1 represents the typical structure of one of these routines. The Standard Event Notice, Page 4.0-23 , carries all the information necessary for the operation of each of these LCCs. This is done by means of standard locations for pointers and indicators within dynamic memory.

TSELEC - 1	T06100 - 53	T11101 - 142
MØVAC - 2	T061AU - 54	T11102 - 143
T01101 - 3	T061MN - 55	T11201 - 144
T01102 - 4	T06201 - 56	T11202 - 145
T01103 - 5	T06303 - 57	T11203 - 146
T01201 - 6	T06400 - 58	T11204 - 147
T01202 - 7	T06401 - 59	T11301 - 148
T01301 - 8	T06402 - 60	T11302 - 149
T01302 - 9	T06402 - 61	T11303 - 150
T01303 - 10	T06403 - 62	T11401 - 151
	T06405 - 63	T11402 - 152
T02000 - 11	T06406 - 64	T11403 - 153
T02MAN - 12	T06407 - 65	T11501 - 154
T02AUT - 13		T11502 - 155
T02ALG - 14	T07101 - 66	T11503 - 156
T02202 - 15	T07102 - 67	
	T07103 - 68	T12101 - 102
T03000 - 16	T07104 - 69	T12102 - 103 & 104
	T07105 - 70	T12103 - 105
T04100 - 17	T07201 - 71	T12104 - 106
T04101 - 18	T07202 - 72	T12105 - 107
T04102 - 19	T07301 - 73	T12106 - 108
T04201 - 20	T07302 - 74	T12107 - 109
T04202 - 21	T07303 - 75	T12201 - 110
T04203 - 22	T07401 - 76	T12202 - 111
T04204 - 23	T07402 - 77	T12204 - 112
T04205 - 24	T07403 - 78	T12205 - 113
T04206 - 25	T07404 - 79	T12206 - 114
T04207 - 26		T12207 - 115
T04208 - 27	T08101 - 80	T12301 - 116
T04209 - 28	T08102 - 81	T12302 - 117
T04210 - 29	T08103 - 82	T12303 - 118
T04211 - 30	T08104 - 83	T12304 - 119
T04212 - 31	T08105 - 84	
T04213 - 32	T08107 - 85	T13100 - 120
T04301 - 33	T08108 - 86	T13102 - 121
T04302 - 34	T08109 - 87	T13104 - 122
T04303 - 35	T08200 - 88	T13105 - 123
T04304 - 36	T08201 - 89	T13201 - 124
T04401 - 37	T08202 - 90	T13202 - 125
T04402 - 38	T08203 - 91	T13203 - 126
T04403-- 39	T08204 - 92	T13301 - 127
T04404 - 40	T08205 - 93	T13302 - 128
T04405 - 41 (DELETED)		
	T09100 - 94	(F14.0 Not Modeled)
T05101 - 42	T09201 - 95	
T05102 - 43	T09202 - 96	
T05103 - 44	T09300 - 97	
T05201 - 45	T09400 - 98	
T05202 - 46	T09501 - 99	
T05203 - 47	T09505 - 100	
T05204 - 48	T09506 - 101	
T05301 - 49		
T05302 - 50	(F10.0 Not Modeled)	
T05303 - 51		
T0522A - 52		

Table 3.2-1 Table of LCC Numbers for Routine Names

T15101 - 160
T15102 - 161
T15201 - 162
T15202 - 163
T15203 - 164
T15204 - 165
T15205 - 166
T15206 - 167

T16101 - 129
T16102 - 130
T16103 - 131
T16201 - 132
T16202 - 133
T16203 - 134
T16204 - 135
T16205 - 136
T16206 - 137
T16207 - 138
T16208 - 139
T16209 - 140
T16210 - 141

T17100 - 157
T17200 - 158
T17300 - 159

Utilities and Algorithms

UDPØF - 168
UDMSW - 169 (not used)
DRPAC - 170
METSEQ - 171
HAZARD - 172
TCANAC - 173
RSCRSE - 174

Table 3.2-1 (contd.) Table of LCC numbers for Routine Names

Grouped Names	Functional Analysis Names
T061AU T061MN	6.1.1
	6.1.2
	6.1.3
	6.1.4
	6.1.5
	6.3.1
	6.3.2
T08105	8.1.5
	8.1.6
T09100	9.1.1
	9.1.2
	9.1.3
T09300	9.3.1
	9.3.2
T09400	9.4.1
	9.4.2
T09501	9.5.1
	9.5.2
	9.5.3
	9.5.4
T09506	9.5.6
	9.5.7
T12202	12.2.2
	12.2.3
T13100	13.1.1
	13.1.3

Grouped Names	Functional Analysis Names
T17100	17.1.1
	17.1.2
	17.1.3
	17.1.4
	17.1.5
	17.1.6
	17.1.7
	17.1.8
T17200	17.11.1
	17.11.2
T17300	17.2.1
	17.2.2
	17.2.3
	17.2.4
	17.2.5
	17.2.6
	. . .
	17.7.1
	17.7.2
	17.7.3
	17.7.4
	17.7.5
	17.8.1
	17.8.2
	17.8.3
	17.8.4
	17.8.5
	17.9.1
	. . .
	17.10.1
	. . . .

Table 3.2 - 2 Grouped Functional Analysis Tasks

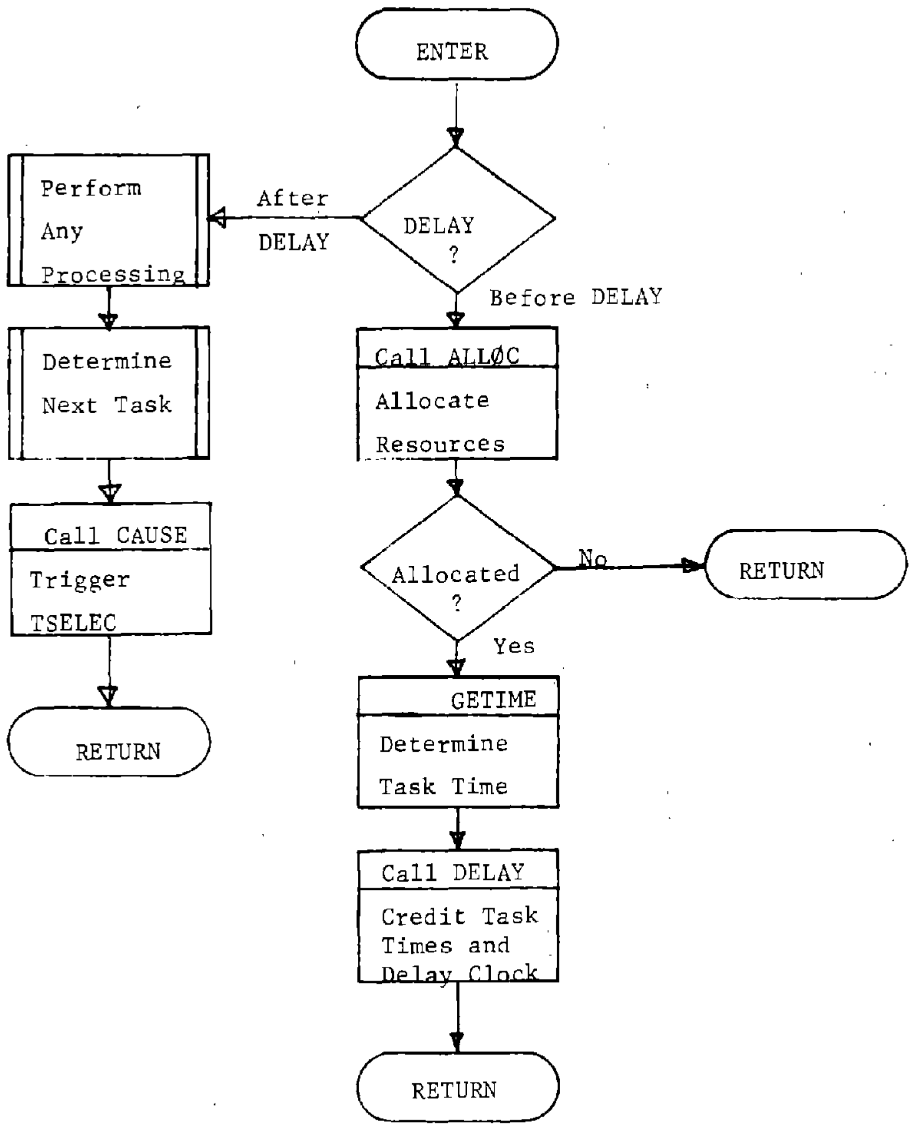

Figure 3.2- 1 Flow Chart of a Typical Functional Analysis Routine.

3.2.3 Typical LCC

A positive value for NXTSK is used to indicate that the processing has resumed after a delay. ALLOC sets the value of IRESRC, which is then tested to determine if the task was placed in the Task Queue due to the unavailability of a resource element for the current task indicated by a negative IRESRC. If not, GETIME is used to generate a task time which is then passed to DELAY. Following the delay, any processing required by the task is performed and the next task to be performed, if any, is determined. Triggering TSELEC then results in the triggering of the selected task or next activity for the resource element allocated for this task. This structure is basic to the resource-using routines.

3.2.4 Variable Naming Convention

Where possible, variable names for the task routines are related to both their usage and their task number. Figure 3.2-2 contains an outline of this naming convention. Further discussion of input variables will be found in Book I of this Volume.

3.2.5 Task Grouping Convention

As will be noticed in Table 3.2-1, there is not a one-to-one relationship between functional analysis tasks and LCCs. This is due to a simplification made to the programming logic. It was decided that in several instances, there are tasks which could be grouped together in all reasonable system designs. These groupings are indicated in Table 3.2-2. The groupings reduce the number of LCCs required to model all the tasks. Another form of grouping of task performance used in the model is the blocking of tasks. This is done typically for time-cyclic tasks performed for all aircraft in a jurisdiction. Blocking involves performing the task for several aircraft together by multiplying the task performance times by the appropriate number of aircraft. This is determined by input, to give the user as much control as possible. A third grouping technique is along allocation lines. Thus, all manual tasks in such a grouping are simulated by one routine and the automated tasks, by another.

As many variables as possible will have names made up in the following way:

affstt

where ff is the Function number

s is the Subfunction number

tt is the Task number

and a is an alphabetic with the following characteristics:

C = Duration of one cycle of a periodic task (hrs.).

F = Rate (inverse of frequency) of a periodic task.

I = ELIST block pointer (defined in Block Data).

M = Mask for a test under mask.

P = Probability of condition occurring.

R = Requirement (Comparison word for test under mask).

T = Subroutine name containing referenced task.

FIGURE 3.2-2 NAMING SCHEME FOR STANDARD VARIABLES

3.2.6 Parallel Tasks

Some tasks may not be triggered until a set of parallel tasks have been completed, as in the case of Function 4, for example. Towards this end, the ACCUM routine was designed to accumulate task completions until all required tasks had been performed, before the indicated next task could be triggered.

3.2.7 Bit Logic Conventions

In order to carry as much information as possible per word of data, the model often uses the sign of a variable as a logical indicator. For example, in the Standard Event Notice, a positive value for IACFT implies that the value is the pointer to an aircraft, while a negative value indicates that the absolute value is a pointer to a jurisdiction. Similarly, a positive IRESRC is a pointer to a resource element, while a negative value indicates a pointer to a resource pool.

ACBITS of the Aircraft File is a collection of indicators describing the status and other information on the aircraft to which it belongs. This is done by considering the word to be a collection of binary switches, which are for the most part independent. LBYT and SBYT, described above, are used to test and set these bits. Table 3.2-3 contains a description of their uses. Table 3.2-4 indicates which routines use particular bits. As can be seen, only thirty two of the sixty bits have been used, in order to permit the conversion of the program to other computer systems.

3.2.8 Internal Units

Within the algorithms, a units convention was adopted, covering the units used within the model to reduce repeated conversion. The DELTA Model uses nautical miles, hours and radians as standard units, converting inputs when read. Hence, speeds are in knots (nautical miles per hour) and turn rates are in radians per hour. All distances, including altitudes, are in nautical miles.

TABLE 3.2-3 ACBITS USAGE

BIT NUMBER	DESCRIPTION
1	Discrepancy between flight plan and capability and status of Aircraft
2	Discrepancy between flight plan and operational and environmental condition
3	Discrepancy between flight plan and other approved flight plans
4	Discrepancy between flight plan and flow control directives
5	Discrepancy between flight plan and rules and procedures
6	Discrepancy between flight plan and flight progress
7	Discrepancy between flight plan and user class/pilot qualifications
8	Flight plan being reviewed (subsequent reviews)
9	Current deviation out-of-tolerances
10	Enlarged tolerances being used
11	End of flight Indicator
12	Flight plan revision from Task 7.1.5
13	Flight plan under revision
14	Short range deviation out of tolerances
15	Long range deviation out of tolerances
16	Controlled aircraft
17	Intensions known aircraft
18	IFR aircraft
19	VFR aircraft
20	Flight plan being resubmitted (subsequent submission)
21 22 23	Metering and Sequencing status switch 1 = M&S Queue entry created, 2 = Missed approach selected 3 = M&S hold complete 4 = M&S no hold solution 5 = M&S hold solution 6 = Missed approach solution
24	Not used
25	Not used
26	Special call 2 for MOVAC triggered
27	Hold type indicator for aircraft in holds (T→en route, F→M&S)

TABLE 3.2-3 ACBITS USAGE (Cont'd)

BIT NUMBER	DESCRIPTION
28	Hold status indicator (F→never in hold), see Bit 31
29	Not used
30	Aircraft handover initiated
31	Hold status indicator (F→had been held, not currently, T→currently in hold)
32	Emergency indicator

LEGEND: T = Set On; F = Set Off; U = Use

SUB-ROUTINE	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
ACJBC2			F		F		F									
CØNDET																U
FILHAZ										U	U	U				
MØVAC		TFU	TU		TU	TU	TFU			FU	FU	TU				
RECHECK																U
TØLFX5																U
T03000													TFU	TF	TF	TF
T04101																
T04201																
T04202																
T04203																
T04204																
T04205																
T04206																
T04207																
T04210																
T04302																
T04401																
T04402																
T05202																
T0522A																
T06403	T															
T07102	U															
T07104																U
T07201																
T07301																
T07302																
T07303																
T07401																
T07403																
T07404	U															
T16103	F															
UDPØF										F	T	F				U
	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17

TABLE 3.2-4 ACBITS CROSS REFERENCE

LEGEND: T = Set On; F = Set Off; U = Use

SUB-ROUTINE	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
ACJBC2																
CØNDET	U															
FILHAZ	U															
MØVAC																
RECHECK	U															
TØLFX5		U						U								
T03000	TF															
T04101									F							
T04201									U							TF
T04202									U						TF	
T04203									U					TF		
T04204													TF			
T04205									U			TF				
T04206									U		TF					
T04207									U	TF						
T04210										U	U	U	U	U	U	U
T04302									T							
T04401				F	FU				FU							
T04402					F				T							
T05202							T									
T0522A							F									
T06403																
T07102						U										
T07104				TU	T											
T07201				U	U		U	TF								
T07301				U	U											
T07302			TF													
T07303		TF														
T07401		U	U				T	U								
T07403				T												
T07404																
T16103																
UDPØF						TF										
	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

TABLE 3.2-4 ACBITS CROSS REFERENCE (Cont'd)

3.3 MAIN Routine

Program MAIN is the entry point for the model. Within CDC program structure, the main routine includes the input/output device list for the remainder of the routines. It is also the first routine loaded, and, CDC convention indicated that the lengths of the common blocks are determined at the first encounter of a common block. Hence, it is in MAIN that the length of the IV array for dynamic memory is set. For batch processing, a length of 15,000 words has been used. This value is compiled into MAIN. The minimum length permitted is 2000 words, since this length is compiled into most of the LCCs.

The DELTA Model uses the following set of FORTRAN logical units:

TAPE5 is the main input file, containing the SALSIM card, Name list, task input data, pilot response times, aircraft type data, jurisdiction data, scenario data and exogenous inputs.

TAPE6 is the primary scenario output file, containing diagnostic and warning messages and the DATAFIL end of run summary.

TAPE10 is used for the ACPTS file, maintained as a temporary file by GETAC and MØDAC.

TAPE12 is the Conflict Detection Grid file, maintained as a temporary file by CØNDET.

TAPE14 is the simulation history file which is used by the Post Processor in various forms.

TAPE21 is the prime file for aircraft pair data, used as a temporary file by the Conflict Detection routines.

TAPE22 is the Conflict Resolution Maneuver file.

TAPE23 is the file used by the Conflict Detection routines to check for new conflicts.

TAPE30 is the Terminal Data input file used by METSEQ.

The only other activity performed by MAIN is the calling of XXSTRT, the SALSIM initialization routine.

3.4 Utility Routines

3.4.1 SALSIM Subroutines

All of the list processing requirements described in section 5.0, Dynamic Memory File Structure, utilize a TRW simulation language called SALSIM.

SALSIM is comprised of FORTRAN subroutines and functions which allow the user to create lists having an arbitrary number of entries. The system uses a large one dimensional array, IV, which contains all the lists.

For example, a typical list of information might have 8 records, each 10 words long. Ten consecutive words (subscripts) of the IV array then holds 1 record. Each record is located somewhere in the IV array. Note that although words of a record are consecutive, the records themselves usually are not consecutive. One of the ten bytes of each record contains the IV array subscript of the first word for the next record in the list.

Each record of a few lists also contains a similar subscript for the preceding record in a list. When there is no preceding or following record in the list, the byte containing the IV subscript is zero. Each item in a record for a single list has a name equivalenced relative to the IV array for easier accessing.

The SALSIM subroutines are used for bookkeeping of the lists. They allow the user to easily insert a record into a list without consciously changing pointers to or from that record.

The primary advantage of using this list processing technique is that data storage can be allocated dynamically. Once a fixed amount of core has been allocated to the IV array, the amount of core used by each list can be allocated or freed repeatedly during execution according to the requirements of the current model run.

Each list has an ordering which is specified when the list is created. Each list has exactly one of the following orderings for filing and accessing records: last in first out, first in first out, increasing order of attribute (ranked low is first), decreasing order of attribute (ranked high is first).

The ordering as well as other characteristics for a list are defined exactly once when the list is created by use of the SALSIM DEFSET function, which is described in detail later in this section.

SALSIM is used extensively by the utilities, algorithms, and functional analysis tasks described in sections 3.4, 3.5, and 3.6. Besides using SALSIM to handle lists of data, these routines can trigger (schedule) events to occur at a future simulation time. Events are placed in a queue (another list) until the simulation time has caught up with the event. The event is then removed from the queue and processed by the indicated subroutine, called an LCC.

Each LCC has associated with it a number which the user indicates when triggering an LCC for execution by use of the CAUSE subroutine described in this section. A complete list of the LCC numbers is given in section 3.2, CONVENTIONS.

SALSIM Subroutine: CANCEL (LCCNO, IPT)

A. PURPOSE

This routine cancels an event notice in the event queue.

B. INPUT

LCCNO: LCC number of the event. (Integer)
IPT: Pointer to the event in the IV array. (Integer)

C. OUTPUT

None

D. USAGE

CALL CANCEL (LCCNO, IPT)

SALSIM Subroutine: CAUSE (LCCNO, IPT, TIME)

A. PURPOSE

This routine schedules an event to be processed by a specified subroutine at a given simulation time.

B. INPUTS

LCCNO: LCC number of subroutine to be triggered. (Integer)
IPT: Current event number, a number assigned by the system and always acessed from the integer variable EVENT. (Integer)
TIME: Simulation time that subroutine is to be called. (Integer)

C. OUTPUT

None

D. USAGE

CALL CAUSE (LCCNO, IPT, TIME)

SALSIM Function I=CREATE (L)

A. PURPOSE

This routine allocates a specified amount of consecutive words in the IV array to store a record in a list.

B. INPUTS

L: Length of record to be created. (Integer)

C. OUTPUT

I: The pointer to record, i.e., the IV subscript number preceding the location of the record to be stored. (Integer)

D. USAGE

I=CREATE (L)

E. REMARK

This routine does not store the record. The record is stored in a list by the FILE command, which uses the I parameter as input.

SALSIM Function ISET=DEFSET (T,F,L,S,P,R)

A. PURPOSE

DEFSET defines the attributes of a list to be created. These attributes remain unchanged for the duration of the model run or until the list is deleted (not just emptied).

B. INPUTS

T: An integer that indicates the ordering rule for the list. The permissible values of T are:

1, last in first out;
2, first in first out;
3, increasing order of attribute (ranked low is first);
4, decreasing order of attribute (ranked high is first).

F: First pointer (Integer)

L: Last pointer (Integer)

S: The word position in a record for holding the pointer (subscript) to the record's predecessor. (Integer)

P: The word position in a record for holding the pointer to the record's successor. (Integer)

R: Used only for ranked files (T=3 or 4). This is the word position in a record upon which the list is ordered. R is zero for (T=1 or 2). (Integer)

C. OUTPUT

ISET: The pointer (subscript) in the IV array that holds the DEFSET information. (Integer)

D. USAGE

ISET= DEFSET (T,F,L,S,P,R)

E. REMARK

The DEFSET is called exactly once to create a list. The ISET parameter is used as input to many SALSIM subroutines.

SALSIM Subroutine DELAY (EVENT, D)

A. PURPOSE

This routine delays the current event, whose number is EVENT, for a specified amount of time. When this time occurs, processing of the event resumes.

In addition, this subroutine talleys the resource utilization time and stores the result in the Resource Element File.

B. INPUTS

EVENT: The current event number, which is always accessed from the variable EVENT. (Integer)

D: The time delay from current time. (Integer)

C. OUTPUT

None

D. USAGE

CALL DELAY (EVENT, D)

SALSIM Subroutine DSTROY (IPTR, L)

A. PURPOSE

DSTROY releases the locations in the IV array for a record no longer needed. The space is then available to store a record for any list.

B. INPUTS

IPTR: The pointer to the record to be destroyed. (Integer)

L: The length of the record to be destroyed. (Integer)

C. OUTPUT

None

D. USAGE

CALL DSTROY (IPTR, L)

SALSIM Subroutine FILE (IPTR, ISET, IPT1)

A. PURPOSE

FILE inserts a record into a list according to the ordering defined by the DEFSET function.

B. INPUTS

IPTR: The pointer to the first word of the record in the IV array. IPTR is obtained from the CREATE function, which is usually performed prior to executing the FILE subroutine. (Integer)

ISET: The pointer to the list's DEFSET characteristics. (Integer)

IPT1: Index (owner) of the list. (Integer)

C. OUTPUT

None

D. USAGE

CALL FILE (IPTR, ISET, IPT1)

SALSIM Function I1= FIRST (ISET, J)

A. PURPOSE

FIRST stores in the variable I1, the address of the first record in the list associated with list ISET, J.

B. INPUT

ISET: The pointer to the list's DEFSET characteristics (Integer)

J: Index (owner) of list. (Integer)

C. OUTPUT

I1: The pointer to the first record of the list, i.e., the subscript in the IV array. I1 is zero if the list is empty. (Integer)

D. USAGE

I1= FIRST (ISET,J)

SALSIM Function I2=NEXT (ISET, IPTR)

A. PURPOSE

The NEXT function retrieves the address (subscript) of the record following a specified record address.

B. INPUT

ISET: The pointer to the list's DEFSET characteristics (Integer)

IPTR: The address of the record for which the next record is desired. (Integer)

C. OUTPUT

I2: The pointer to the next record. (Integer)

D. USAGE

I2= NEXT (ISET, IPTR)

SALSIM Function I1= RFIRST (ISET,J)

A. PURPOSE

This function removes the first member from a specified list.

B. INPUT

ISET: The pointer to the list's DEFSET characteristics. (Integer)

IPTR: The index (owner) of list. (Integer)

C. OUTPUT

I1: The address of the record removed from the list. (Integer)

D. USAGE

I1= RFIRST (ISET,J)

E. REMARKS

This function does not release the space occupied by the record removed from the IV array. To do so, the DSTROY subroutine should follow the RFIRST function.

SALSIM Subroutines RSPEC (IPTR, ISET, J)
or RSPEC1 (IPTR, ISET, J)

A. PURPOSE

Subroutine RSPEC removes a record from a specified ranked file. Subroutine RSPEC1 removes a record from a specified last-in-first-out file.

B. INPUT

IPTR: The pointer to the record to be removed. (Integer)
ISET: The pointer to the list's DEFSET characteristics.
J: The index of the list.

C. OUTPUT

None

D. USAGE

CALL RSPEC (IPTR, ISET, J)
CALL RSPEC1 (IPTR, ISET, J)

E. REMARKS

These subroutines do not release the space occupied by the record removed from the IV array. To do this, the DSTROY subroutine should follow the RSPEC or RSPEC1 subroutine.

3.4.2 Model Utility Descriptions

The DELTA Model utilities are described in this section. Briefly, these are the following:

ABØRTT	-	Abnormal end of run routine.
ACCUM	-	Accumulate completions of parallel tasks.
ALLØC	-	Allocate resources to perform tasks.
CANAC	-	Cancel the aircraft.
DATAFIL	-	Write out end of run summary.
DRPAC	-	Drop the aircraft from memory.
ELIST	-	LCC driver routine.
GETAC	-	Search for aircraft.
GETASK	-	Get task information.
GETIME	-	Get task performance times.
GETØLR	-	Get tolerance data.
IDECSN	-	Search files using test under mask.
IDPTR	-	Search LIFO files for matching mnemonics.
MØDAC	-	Maintain the aircraft file.
PILØTR	-	Determine pilot's response time.
TCANAC	-	LCC call to CANAC.
TSELEC	-	Select next task to be performed.
UDPØF	-	Update phase of flight.

SUBROUTINE NAME: ABØRTT

A. PURPOSE

This routine is responsible for terminating a model run due to any abnormal condition detected by either SALSIM routines or model routines. It will print an error message, and, if possible, call DATAFIL routine.

B. DISCUSSION

ABØRTT is required by the SALSIM utilities. All abort conditions in the model will result in the calling of this routine. After printing its message, ABØRTT determines if it has been called from DATAFIL routine. If this is not the case, DATAFIL is called and the run terminates in a STØP 7777. Otherwise, the run is terminated without calling DATAFIL again, and so avoids a closed loop situation.

C. INPUTS

Common variables input are the following:

TIME, model clock time at end of run;

IT, LCC number in which error was found;

EVENT, pointer to the event notice being processed;

IERRØR, error number set by SALSIM utilities.

D. OUTPUTS

ABØRTT prints out an error message and, after calling DATAFIL, it uses a numbered stop, STØP 7777.

E. STIMULI

This routine is called by any routine which has diagnostic error tests as opposed to warning-level error messages.

F. SUBROUTINES CALLED

DATAFIL is called to list the dynamic memory files, giving resource utilization data and end of run statistics useful in tracking down the cause of the error.

SUBROUTINE NAME: ACCUM

A. PURPOSE

This routine increments and compares IACCUM to ILIMIT.

B. DISCUSSION

ACCUM is used to stimulate the next LCC after the completion of the last LCC of a set of LCCs stimulated in parallel. It makes use of an accumulator block defined by the appropriate driver routine. Each of the LCCs contains a pointer to the accumulator block (IACBLK), which is passed in the calling sequence. The driver sets the limit (ILIMIT) which is the number of parallel LCCs, and the next LCC pointer (NXTSK) into the accumulator block.

When called, ACCUM increments IACCUM by one. It compares IACCUM to ILIMIT and triggers NXTSK when IACCUM is not less than ILIMIT. ACCUM then returns to the calling LCC.

C. INPUTS

1. Calling Sequence: ACCUM(IPTR) where IPTR is the pointer to the appropriate accumulator block

2. COMMON Variables used: IACCUM(IPTR) is the counter, ILIMIT(IPTR) is the limit on the count, NXTSK(IPTR) is the LCC to be triggered.

D. OUTPUT

None

E. STIMULI

1. The subroutine is called by every LCC triggered in parallel where it is necessary to know when the last LCC is completed.

2. ACCUM may trigger NXTSK LCC if the limit is reached.

F. SUBROUTINES CALLED

ACCUM calls CAUSE to trigger NXTSK.

SUBROUTINE NAME: ALLØC

A. PURPOSE

This utility routine allocates the given task to either the appropriate queue or the appropriate resource. This routine is also responsible for monitoring the back log for the given resource pool and adding additional resource elements when necessary.

B. DISCUSSION

ALLØC allocates the task pointed to by the event notice and IT. It determines the jurisdiction and resource pool responsible for the task and attempts to acquire a resource element. If none is available, it stores the task in the queue for that resource pool. If the queues are too long or too much delay is expected, additional elements may be added.

Specifically, IRESRC .GT. 0 is the Allocated Task Indicator. IRESRC contains the resource element pointer. IRESRC .EQ. 0 indicates not from queue, and IRESRC .LT. 0 indicates placed in queue. IACFT .GT. 0 is the pointer to the Aircraft File which includes the jurisdiction pointer, IJURIS. IACFT .LE. 0 indicates the jurisdiction directly; the jurisdiction pointer IJURIS. IACFT .LE. 0 indicates the jurisdiction directly; the jurisdiction pointer is absolute value of IACFT, in this case. From the jurisdiction, it is possible to determine the appropriate resource pool. JPØØL, returned from GETASK, defines which resource pool of the jurisdiction is to be used, and is pointed to by JRSCPL.

The elements of the pool which are not busy are stored in the IREFIL. If the file is empty, no elements are available, so the task is placed in the ITASKQ, task queue. Otherwise, the element is removed from the IREFIL and the task is allocated to the element.

C. INPUT

1. Common Variables Used: IT, the current task pointer;
EVENT, the current event notice pointer;
IACFT (EVENT), the current aircraft or jurisdiction pointer;

IRESRC (EVENT), the current resource pointer;

IJURIS, the current aircraft's jurisdiction pointer;

JRSCPL, the pointer to the jurisdiction's resource pool.

2. Subroutine returns: GETASK returns the following:

JPRIØ, task priority;

JPØØL, characteristic resource pool;

RFIRST returns pointer to first available resource element, if any.

D. OUTPUTS

Common Variables: IPRIØ (EVENT), the task queue priority, is set to JPRIØ;

IRESRC (EVENT), is set as FROM QUEUE flag;

IRSTAT, resource element status set to 1 to indicate element is busy.

E. STIMULI

ALLØC is called in each routine which utilizes a resource. ALLØC does not trigger any LCCs.

F. SUBROUTINES CALLED

GETASK, called to determine priority and resource pool required for task;

RFIRST removes resource element from file for pool;

FILE places current task into ITASKQ file for pool.

SUBROUTINE NAME: CANAC

A. PURPOSE

CANAC is intended to remove the aircraft from the system.

B. DISCUSSION

This routine is designed to remove all indication of the aircraft from the system. It will destroy all traces of the aircraft from the system, including the aircraft's Time/Position Queue, Conflict File, Runway Queue, and cancel as many scheduled events as possible. The aircraft file is allowed to remain in dynamic memory for a tenth of an hour after the aircraft leaves the system, to clear up any in-progress tasks. IJURIS is set negative as an indicator that CANAC HAS PROCESSED THE AIRCRAFT.

C. INPUTS

CANAC calling sequence passes the aircraft pointer and an indicator used to determine the reason for the call.

Common variables used are those in the aircraft file, the conflict history file, the time/position queue and the runway and terminal files.

D. OUTPUTS

None.

E. STIMULI

This routine is called as an LCC using routine TCANAC and as subroutine by UDPØF, and T04402.

F. SUBROUTINES CALLED

CANEVT1 is called to cancel special call to MOVAC; RSPEC and RFIRST are used to remove files from queues; DSTRØY is called to remove files from dynamic memory; CAUSE is called to trigger DRPAC.

SUBROUTINE NAME: DATAFIL

A. PURPOSE

DATAFIL is intended to write out the dynamic memory files dealing with resource utilization.

B. DISCUSSION

This routine will write out the end of run information generated by the model. This includes information from the Resource Element File, the Resource Pool File, the Aircraft File, the Task Queue, the Jurisdiction File, the Terminal File, the Runway Queue, and the Metering and Sequencing File. The principle concern is with resource utilization. The print out is described elsewhere.

C. INPUTS

DATAFIL uses information from the files indicated above. Additionally, if the run has terminated normally, it will read in a run title from the exogenous event file.

D. OUTPUTS

DATAFIL writes out the end of run summary.

E. STIMULI

This routine is called from either ELIST when exogenous event 180 is encountered or ABØRTT.

F. SUBROUTINES CALLED

None.

SUBROUTINE NAME: DRPAC

A. PURPOSE

This routine is responsible for the final removal of an aircraft from the dynamic memory.

B. DISCUSSION

DRPAC performs two duties, it destroys the indicated aircraft file and its own event notice. This routine is an LCC. The aircraft file is permitted to remain available after aircraft cancellation so that tasks in progress may have valid pointers to the aircraft with indications that the aircraft is cancelled to avoid additional processing. A negative IJURIS is this indicator.

C. INPUTS

The pointer to the aircraft file is the only input to this routine. It is passed via the IACFT variable of the standard event notice.

D. OUTPUTS

None.

E. STIMULI

DRPAC is triggered by CANAC.

F. SUBROUTINES CALLED

DSTRØY is called to relinquish the aircraft file back to dynamic memory.

SUBROUTINE NAME: ELIST

A. PURPOSE

ELIST is the central driver for all model logical control chain(LCC) routines. It is responsible for calling all LCCs.

B. DISCUSSION

ELIST and its EL-subroutines are the main driving utilities required by the SALSIM model structure. ELIST decodes the LCC number, identifying which subroutine should be called for a given LCC number. Due to the overlay constraints, several small routines which behave like ELIST but that are only concerned with LCCs for a particular function were used. These are EL01, EL04, EL05, EL06, EL07, EL08, EL09, EL12, EL15, EL16 and EL17. They are associated with Functions 1., 4., and so on.

C. INPUTS

ELIST receives the LCC number and event notice pointer from SIMRUN, the SALSIM driver, through a calling sequence.

D. OUTPUTS

None.

E. STIMULI

Every LCC uses the ELIST routine as a driver.

F. SUBROUTINES CALLED

ELIST calls all LCCs and the functional drivers listed in B.

SUBROUTINE NAME: GETAC

A. PURPOSE

This routine determines the pointer to the Aircraft Data File for the aircraft named or identified in ACNAME and returns the pointer to IPTRAC. If the aircraft identification is not found, the pointer is set to zero.

The file of id versus pointer is kept on a high speed storage device, IACPTS, and read if a test reveals that the data has been overlaid.

B. DISCUSSION

The variable ITEST is set in a DATA statement. When GETAC is overlaid and returned to core, the initial value from the DATA statement is present, indicating that IACPTS must be accessed to read the file. ITEST is set different from its initial value after the read and rewind. The file is rewound after each accessing to assure its readiness to be read.

ACNAME is compared to each ACIDS aircraft identification. (The file may seem redundant, and it is. The alternative to this file search is to search each jurisdiction for its associated Aircraft Data Files, which also include the aircraft identifiers. This process is more time consuming than the maintenance of a table of ids versus pointers.) When a match between ACNAME and ACIDS is found, the pointer value IPTRAC is set to the appropriate value of IPTRS. If no match is found, zero is returned. The DO-LOOP index IAC is retained in COMMON ACPTS. It will contain the index of the match or one more than the number of aircraft, NAC, if no match is found.

C. INPUTS

1. Calling Sequence: GETAC(ACNAME, IPTRAC) where ACNAME contains and aircraft identifier.

2. Common Variables: From COMMON ACPTS,
   NAC, number of aircraft active in the system,
   ACIDS, array containing the aircraft identifiers,
   IPTRS, array containing the pointers to the Aircraft Data File.

3. Internal Variables: ITEST, used to determine whether it is necessary to read IACPTS, set to zero in DATA statement, reset to 1 if IACPTS is read.

D. OUTPUTS

1. Calling Sequence: GETAC(ACNAME, IPTRAC) where IPTRAC contains the pointer to the Aircraft Data File or zero if none found.

2. Common Variables: IAC, the index of arrays ACIDS and IPTRS.

E. STIMULI

This routine is called from MODAC and anywhere it is necessary to convert aircraft id to pointer, such as aircraft exogenous events, where only the id of the aircraft is available.

F. SUBROUTINES CALLED

None

SUBROUTINE NAME: GETASK

A. PURPOSE

GETASK is used to retrieve task information by the model utilities.

B. DISCUSSION

This routine provides a table look up capability within the task data files. It is intended to first check the value of the input task pointer, ITSKPT to determine if there is a valid pointer. If not, IDECSN is called to generate the needed subscript. The task file is searched using a test value composed of the LCC number, the aircraft user class and phase of flight. IDECSN returns with the needed value for ITSKPT. With this, the resource pool indicator, priority value, and task time indicator are retrieved.

C. INPUTS

GETASK inputs the aircraft pointer and phase of flight and user class from common. The task pointers ITASK and ITSKPT are passed through the calling sequence.

D. OUTPUTS

GETASK puts out the following parameters via its calling sequence:

ITIME, the number of instructions, if negative, or the subscript to the appropriate task time distribution.

ITSKPT, the task subscript (may also be an input).

JPRIØ, the task priority.

JPØØL, the resource pool indicator.

E. STIMULI

GETASK is called by ALLØC, TSELEC, and GETIME.

F. SUBROUTINES CALLED

IDECSN is used to search the task data files for the task pointer.

SUBROUTINE NAME: GETIME

A. PURPOSE

This routine is responsible for calculating the time required to perform the indicated task, including input time, set-up time, performance time, and output time. Performance time is taken from a specific distribution.

B. DISCUSSION

The GETIME function selects a manual task time from the input distribution by determining the half of the distribution randomly and then adding the interquartile distance times a uniformly distributed random number between 0 and 1. An automated task time is generated by dividing number of instructions by computer speed.

C. INPUTS

Calling Sequence: GETIME (ITSKPT) where ITSKPT is a dummy variable.

D. OUTPUTS

Calling Sequence: GETIME (ITSKPT) where GETIME returns the resultant time for one repetition.

E. STIMULI

Every LCC using a resource employs this function to determine the delay time.

F. SUBROUTINES CALLED

GETASK is called to determine instruction count or time distribution.

SUBROUTINE NAME: GETØLR

A. PURPOSE

This routine is responsible for determining the tolerances which are applicable to the given aircraft.

B. DISCUSSION

Tolerances are set to the predetermined values shown below.

C. INPUTS

Calling Sequences: GETØLR (ITSKPT, TØLHØR, TØLVRT) where ITSKPT is a dummy variable.

D. OUTPUTS

Calling Sequence: GETØLR (ITSKPT, TØLHØR, TØLVRT) where GETØLR is the time tolerance, set to 5 minutes, TØLHØR is the lateral (horizontal) tolerance, set to 10 nautical miles, and TØLVRT is the vertical tolerance set to 1 nautical mile.

E. STIMULI

The function is used in Tasks 7.2.1 and 7.3.3 and 9.5.1. No LCCs are stimulated by this routine.

F. SUBROUTINES CALLED

None.

SUBROUTINE NAME: IDECSN

A. PURPOSE

IDECSN is used to search the task files to determine task pointer.

B. DISCUSSION

This routine performs a search on the comparison data and the mask word passed from GETASK or PILØTR. The input state word is anded with the mask word and then compared with each word in the comparison array until a match is found. The first match ends this search. The subscript of the matching element is returned. If no match is found, a zero subscript is returned.

C. INPUTS

ISTATE, NSTATE, MASKS and CØMPS are passed by means of the calling sequence.

ISTATE is the word being tested;

NSTATE is the length of the CØMPS array being searched;

MASKS is the mask word, with which ISTATE is anded, to produce a test under mask;

CØMPS is the array being searched.

D. OUTPUTS

IDECSN, the value returned by the function subprogram IDECSN, is the subscript of the matching element of CØMPS found.

E. STIMULI

GETASK and PILØTR are the only routines calling IDECSN, though the routine was designed to be general purpose in structure.

F. SUBROUTINES CALLED

Only the AND routine is called, it is designed to return the result of a logical AND operation between the elements of its calling sequence.

SUBROUTINE NAME: IDPTR

A. PURPOSE

This routine compares the ID to the file ISET member-id and returns with the pointer to the matching member.

B. DISCUSSION

IDPTR is a function. It is used to scan all members of file ISET, from FIRST, through NEXT, until it has exhausted the file or found a member with id MEMID that matches ID. If a match is found, the function is set to that pointer, otherwise a zero is returned. It is assumed that all files are one index LIFO files and MEMID is the second word of the member.

C. INPUTS

1. Calling Sequence: IDPTR(ISET, ID) where ISET is the pointer to the set definition block defined by DEFSET, and ID is the identifier to be searched for.

2. Common Variables: MEMID (MPTR) is the member identifier of the MPTR member.

D. OUTPUTS

The function value returned is either the pointer to the matching member or zero if no match is found.

E. STIMULI

The routine is used in T03000.

F. SUBROUTINES CALLED

FIRST, used to find first member of file, and NEXT, used to find ensuing members.

SUBROUTINE NAME: MØDAC

A. PURPOSE

This utility routine modifies the Aircraft Identification File by adding or deleting entries.

B. DISCUSSION

MØDAC modifies the data manipulated by GETAC. Entry addition is denoted by IMØD greater than zero, IMØD is equal to the pointer to the Aircraft Data File. Deletion is denoted by IMØD less than or equal to zero.

GETAC is called to determine the existence of a matching aircraft identifier and to insure the presence of the data files in core. (GETAC, COMMON ACPTS and MØDAC must be in the same overlay.) If match is found, and the modification was to be a deletion, the indicated entry is deleted from both ACIDS and IPTRS, indexed by IAC, and the arrays are condensed. If no match was found and the modification was to be an addition, the entry is appended onto the end of the arrays. In both of the latter cases, the arrays are written out to IACPTS for maintenance.

In any other case, an error is indicated by setting IERR non-zero. The file is not changed and not written out to IACPTS. Also, if more than 2000 aircraft are loaded, no action is taken.

C. INPUTS

1. Calling Sequence: MØDAC(ACNAME, IMØD, IERR) where ACNAME is the aircraft identifier, and IMØD is a switch equal to either the Aircraft Data File pointer to indicate an ADD or is less than or equal to zero to indicate a DELETE.

2. Common Variables: IAC, index of ACIDS and IPTRS, as returned from GETAC,
ACIDS, aircraft id array,
IPTRS, pointers to Aircraft Data File,
NAC, number of aircraft.

D. OUTPUTS

1. Calling Sequence: MØDAC(ACNAME, IMØD, IERR) where

IERR = 1 for redundant aircraft id found during an ADD
IERR = 2 for no match found during a DELETE
IERR = 3 for overflow (number of aircraft 2000, maximum)
IERR = 0 for no error.

E. STIMULI

This routine is called in Function 3, to add an aircraft and in 4.4.2 to cancel an aircraft. INITIA uses MØDAC to add the initial load of aircraft.

F. SUBROUTINES CALLED

GETAC is called to seek a matching aircraft id.

SUBROUTINE NAME: PILØTR

A. PURPOSE

This routine is responsible for generating the time delay required for the pilot to respond to a request for data coming from the indicated task.

B. DISCUSSION

This routine uses a set of response time distributions input to the model. These times should represent nominal response times. The times are generated from the distributions in the same manner used by GETIME for task times.

C. INPUTS

The distributions are passed through common. The task numbers are passed via ITSKPT of the calling sequence.

D. OUTPUTS

PILØTR returns the time for the pilot's response as a delta time.

E. STIMULI

PILØTR is called by those routines which have requests made of the pilot for which a response is needed before further processing can occur.

F. SUBROUTINES CALLED

IDECSN is called to determine the task pointer.

SUBROUTINE NAME: TCANAC

A. PURPOSE

This routine cancels an aircraft from the system after it lands or flies out of the system.

B. DISCUSSION

TCANAC is an LCC and thus is executed at the precise moment an aircraft lands or leaves the system. It is triggered by a call to the SALSIM subroutine CAUSE.

C. INPUTS

1. Pointer to aircraft file ACFIL.
2. Simulation time that aircraft leaves system.

D. OUTPUT

None.

E. STIMULI

The routine is triggered by numerous subroutines within the algorithms MØVAC, METSEØ, and CØNDET.

F. SUBROUTINES CALLED

DRPAC (drop aircraft).

CANAC which performs the actual cancellation.

SUBROUTINE NAME: TSELEC

A. PURPOSE

This utility LCC is responsible for concluding the utilization of a resource element by one task and setting the element busy on either the next task in a chain or the highest priority task in the queue for that resource pool or setting the element not busy.

B. DISCUSSION

TSELEC is the only routine able to destroy an event notice of resource-using LCCs. This is done when NXTSK (EVENT) is zero, indicating no next task in a chain. If there is a next task which is not performed by the indicated resource type, the task is triggered with no pre-allocation of resource. If the current resource may perform the next task, the priorities of the next task and the highest priority task in the resources task queue are compared. If the next task is of comparable or higher task, the next task is triggered with a preallocated resource. If the highest priority task has the higher priority, it is removed from the task queue and triggered with a pre-allocated resource, also triggering the next task with no pre-allocated resource.

TSELEC checks for short term overloading of manual resources, and assigns alternate resource elements if available. TSELEC returns non-busy elements to the resource pools. As can be seen from the above, TSELEC and ALLØC are complementary routines.

C. INPUT

1. Common Variables Used: IRESRC (EVENT), resource element; NXTSK (EVENT), pointer to next task, if any; ITASKQ data, including IPRIØ, the highest priority task in the task queue. IREFIL data, including IPØØL and ISTCBT data;

2. Subroutine returns: GETASK returns the following parameters
   - JPRIØ, the task priority for the NXTSK
   - JPØØL, the characteristic resource pool for NXTSK
   - ITSKPT, the subscript of the Task File for NXTSK

D. OUTPUTS

Common Variables: IRESRC (EVENT) and IRESRC (KPTR) the allocation flags

E. STIMULI

TSELEC is triggered in each routine which utilizes a resource. This routine, in turn, may trigger either or both the NXTSK and/or the highest priority task in the Task Queue.

F. SUBRØUTINES CALLED

FIRST, called to determine the highest priority task;
DSTRØY, called to destroy the event notice;
GETASK, called to determine data on NXTSK;
CAUSE, used to trigger LCCs;
RFIRST, used to remove resource element from IREFIL or highest priority task from ITASKQ;
FILE, used to return resource element to IREFIL.

SUBROUTINE NAME: UDPOF

A. PURPOSE

Subroutine UDPOF updates the phase of flight for a single aircraft and performs tasks associated with change in status for the aircraft.

B. DISCUSSION

All input to UDPØF is entered into the Event Notice for UDPØF.

C. INPUTS

(Subroutine MØVAC supplies only the parameter IACPØF. The other triggering routines supply IACPØF as well as most of the other parameters.)

IACPØF: Pointer to aircraft file whose aircraft's phase of flight is to be updated.

TESCAP: Time between approach and missed approach, if any.

IRWPTR: Pointer to IRWQUE member.

ITAKEØF: Time between take-off and departure.

TARRTR: Time between arrival and approach, if any.

TMSDAP: Time between missed approach and approach, if any.

IDEPTR: Time between departure and enroute time.

TAPRCH: Time between approach and landing.

TLNDING: Time between landing and the cancelling of the aircraft from the system.

D. OUTPUTS

None.

E. STIMULI

UDPØF is stimulated by subroutine MØVAC, METSEQ, or T03000.

F. SUBROUTINES CALLED

I06401, RSPEC, DSTRØY, SBYT, MØVAC, METSEQ, CANCEL, I07201, I05103, I05102, UDPØF, I06401, I06201, I07102 and I07301.

3.5 Algorithms

The algorithms are concerned with guiding an aircraft from pre-flight or from the time the aircraft enters the system until it either lands or leaves the system. This section gives a synopsis of the tasks performed by each of these algorithms and the highest subroutine that performs the tasks, as indicated in the flowcharts in section 2.2, Subroutines and Link Structure. A detailed explanation of the operation and logic of these algorithms is presented in section 2 of the User's Guide.

Subroutine METSEQ (metering and sequencing) maintains aircraft which are taking off and landing. It primarily flies aircraft in the terminal area and schedules their maneuvers on the runways.

Subroutine MOVAC maintains the aircraft outside terminal areas. It is responsible for moving aircraft along their flight paths (time-position queues) and determining when an aircraft must maintain a holding pattern. The aircraft usually do not fly along their exact flight paths because random errors due to motion and changes in speed are generated for each aircraft.

A subroutine of MOVAC, BNDRY, is responsible for maintaining the current jurisdiction for each aircraft. It also records information about any aircraft that leaves the system.

Subroutine CRASH, which is called by MOVAC, performs the actual movement of aircraft outside the terminal area. In addition, it projects for conflict detection (CONDET), the region in which an aircraft probably can be found for a given time interval.

The detection of a conflict in actual flight paths for any pair of aircraft is performed in the two subroutines, CONDET and HAZARD. Subroutine CONDET performs preliminary analysis in order to eliminate easily pairs of aircraft which definitely have no conflict. Subroutine HAZARD performs analysis in depth for pairs of aircraft not already eliminated by CONDET.

If HAZARD finds a conflict between a pair of aircraft, subroutine RESOLV determines what maneuvers are necessary to resolve the conflict safely. Subroutine RSCRSE then maneuvers the two aircraft to resume their intended courses.

3.6 Functional Analysis Tasks

On the following pages are flowcharts which show the calling sequences of the LCC's for each function.

Each LCC is a subroutine which is initially triggered (executed) at a simulation time determined by user input data or, more frequently, by a simulated event. Each LCC can in turn trigger one or more LCC's including itself, or can be the last in a sequence of LCC's. The flowcharts for each function display the LCC's that triggered that function, the logical flow within the function, and the action taken when the function completes its chain of LCC's.

Each LCC is tested by a numerical code. LCC number ff.s.tt means function number ff, subfunction number s, and task number tt. The subroutine performing the LCC has name Tffstt.

A complete description of the tasks performed by each of these routines is given in "Function Analysis of Air Traffic Management, Final Report, Volume II." It should be noted that many of the tasks described in this reference have been grouped and written as a single FORTRAN subroutine. The flowcharts indicate only the group name.

Below are listed the group names with the tasks they encompass:

Subroutine Name of Grouped Tasks	Group Name In Flowcharts	Tasks in Group (Volume II)
T02000	2.0.0	2.1.1
T02MAN (manual	2.MAN	2.1.2
T02AUT (Automatic Operations)	2.AUT	2.1.3
		2.1.4
		2.1.5
		2.1.6
		2.2.1
		2.2.3
		2.2.4
		2.3.1
		2.3.2

		2.3.3
		2.3.4
		2.3.5
T03000	3.0.0	3.1.1
		3.1.2
		3.1.3
		3.2.1
		3.2.2
		3.2.3
		3.3.1
		3.3.2
		3.3.3
T061AU (automatic)	6.1.AU and	6.1.1
T061MN (manual operations)	6.1.MN	6.1.2
		6.1.3
		6.1.4
		6.1.5
		6.3.1
		6.3.2
T08105	8.1.5	8.1.5
		8.1.6
T09100	9.1.0	9.1.1
		9.1.2
		9.1.3
T09300	9.3.0	9.3.1
		9.3.2
T09400	9.4.0	9.4.1
		9.4.2

T09501	9.5.1	9.5.1
		9.5.2
		9.5.3
		9.5.4
T09506	9.5.6	9.5.6
		9.5.7
T12202	12.2.2	12.2.2
		12.2.3
T13100	13.1.00	13.1.1
		13.1.3
T17100	17.1.0	17.1.1
		17.1.2
		17.1.3
		17.1.4
		17.1.5
		17.1.6
		17.1.7
		17.1.8
T17200	17.2.0	17.11.1
		17.11.2
T17300	17.3.0	17.2.1
		17.2.2
		17.2.3
		17.2.4
		17.2.5
		17.2.6
		17.7.1
		17.7.2
		17.7.3
		17.7.4

17.7.5
17.8.1
17.8.2
17.8.3
17.8.4
17.8.5
17.9.1
17.9.2
17.9.3
17.9.4
17.9.5
17.9.6
17.10.1
17.10.2
17.10.3
17.11.1
17.11.2

User input data that triggers an LCC is called an exogenous event. One example is the Preparation of Flight Plan. For each aircraft in the system, the user specifies data such as aircraft characteristics, flight options, time information and aircraft location. During model execution, subroutine T03000 is executed when the aircraft enters the system. This routine sets up all queues, switches, and other information that signifies this aircraft has entered a phase of flight.

The list below gives each exogenous event and the LCC it triggers. The input formats for these events can be found elsewhere.

EXOGENOUS EVENT	LCC TRIGGERED
Prepare flight plan	T03000
Accept data link request	T01101
Accept telephone request	T01102
Capability change; Status change; emergency	T06400
Acquire and analyze data on progress of service	T15102

LCC's are triggered most often in the model by algorithms and utilities, which do so in response to events simulated. For example, UDPØF, which updates phase of flight for a single aircraft, triggers subroutines T05102, T05103, T06201, T06401, T07102, T07201, and T07301. A complete description of the utility routines is given in section 3.4 UTILITY ROUTINES, and the algorithms in section 3.5 ALGORITHMS.

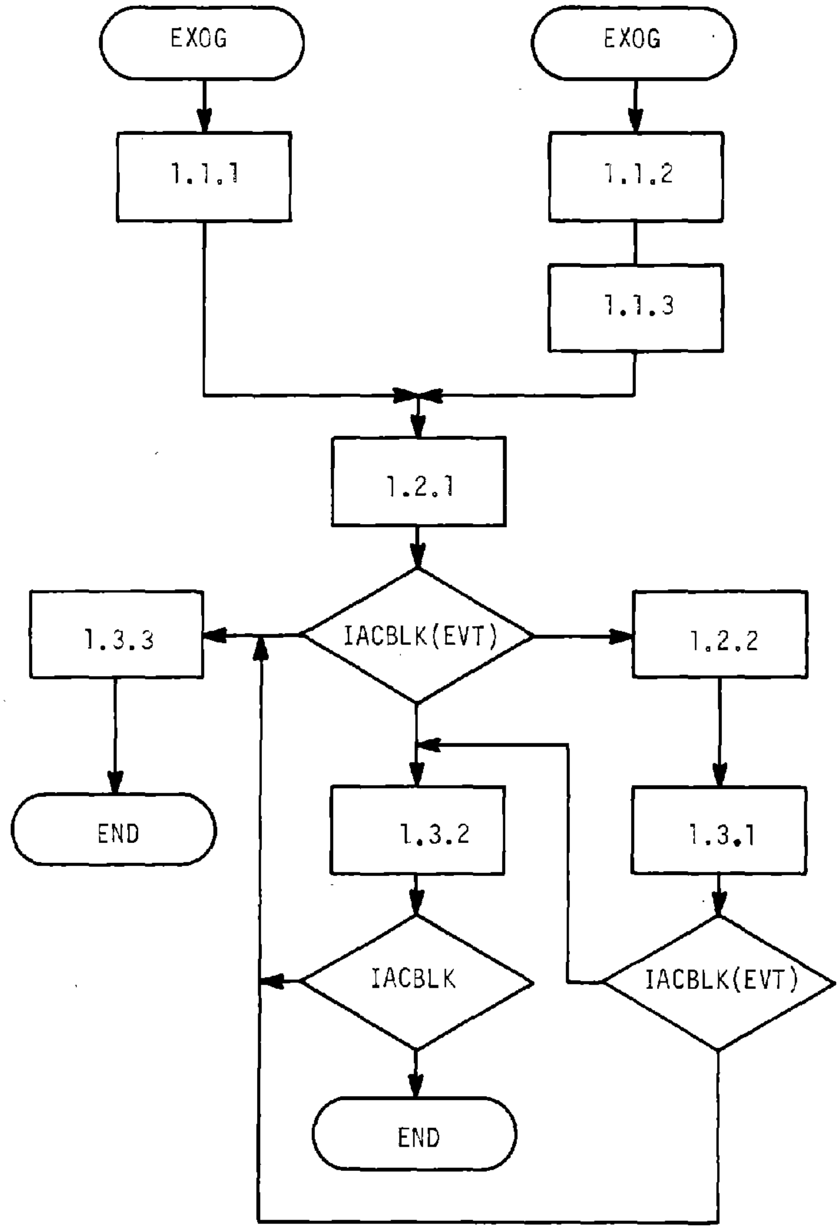

FIGURE 3.6-1 LOGICAL FLOW OF FUNCTION 1.0

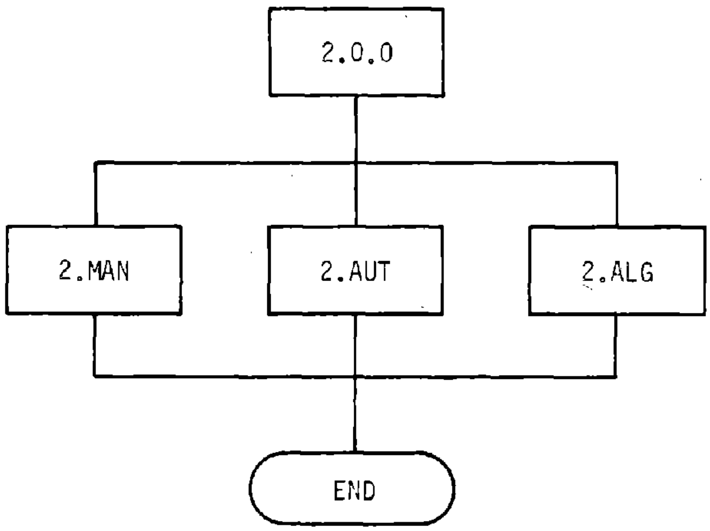

FIGURE 3.6-2 LOGICAL FLOW OF FUNCTION 2.0

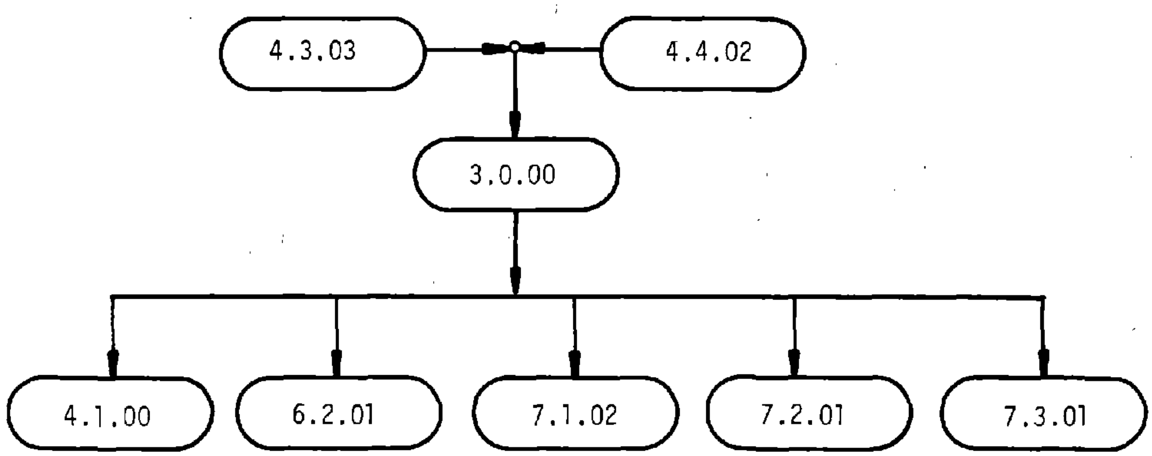

Figure 3.6-3 Logical Flow of Function 3.0

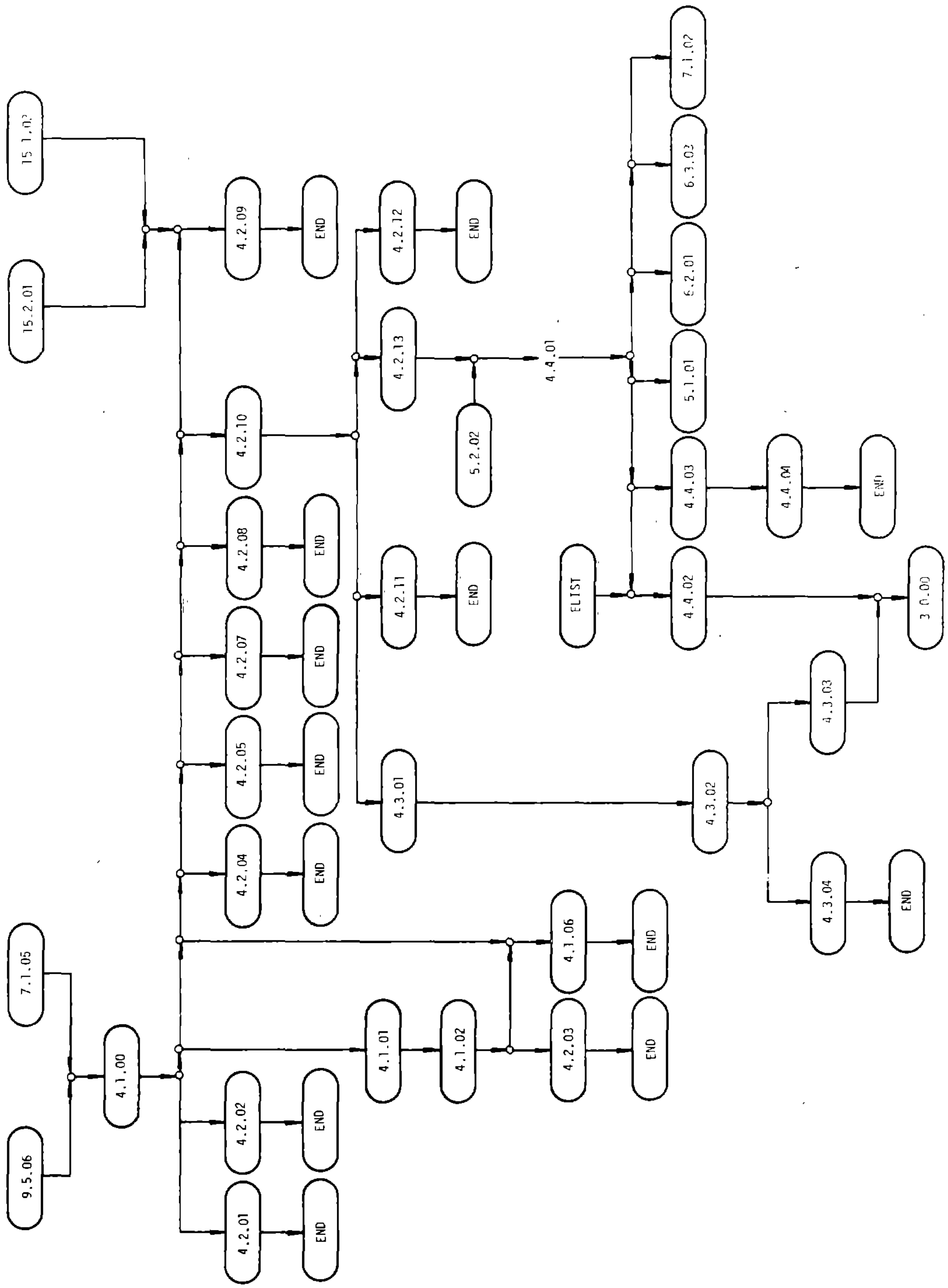

Figure 3.6-4 Logical Flow of Function 4.0

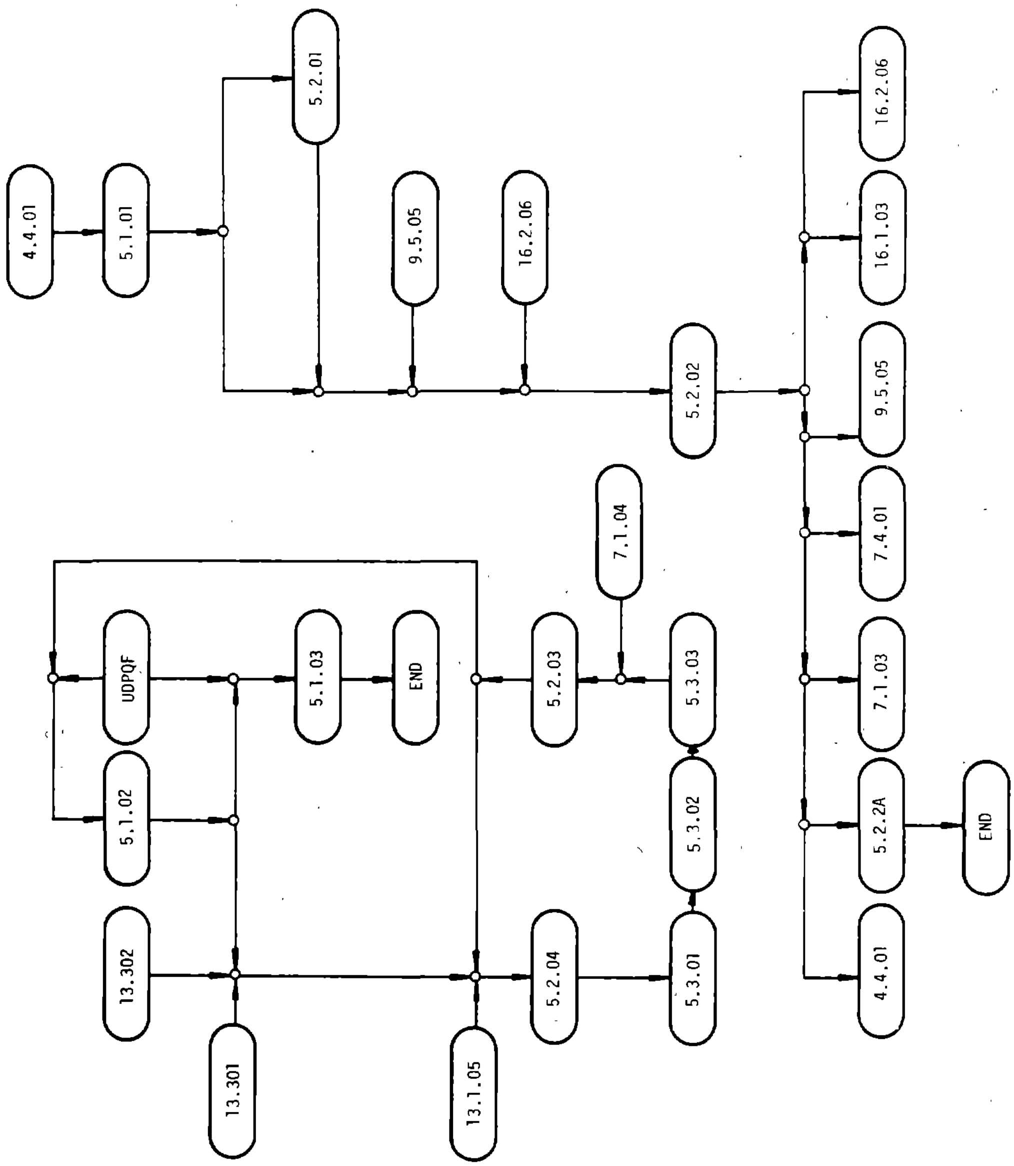

Figure 3.6-5 Logical Flow of Function 5.0

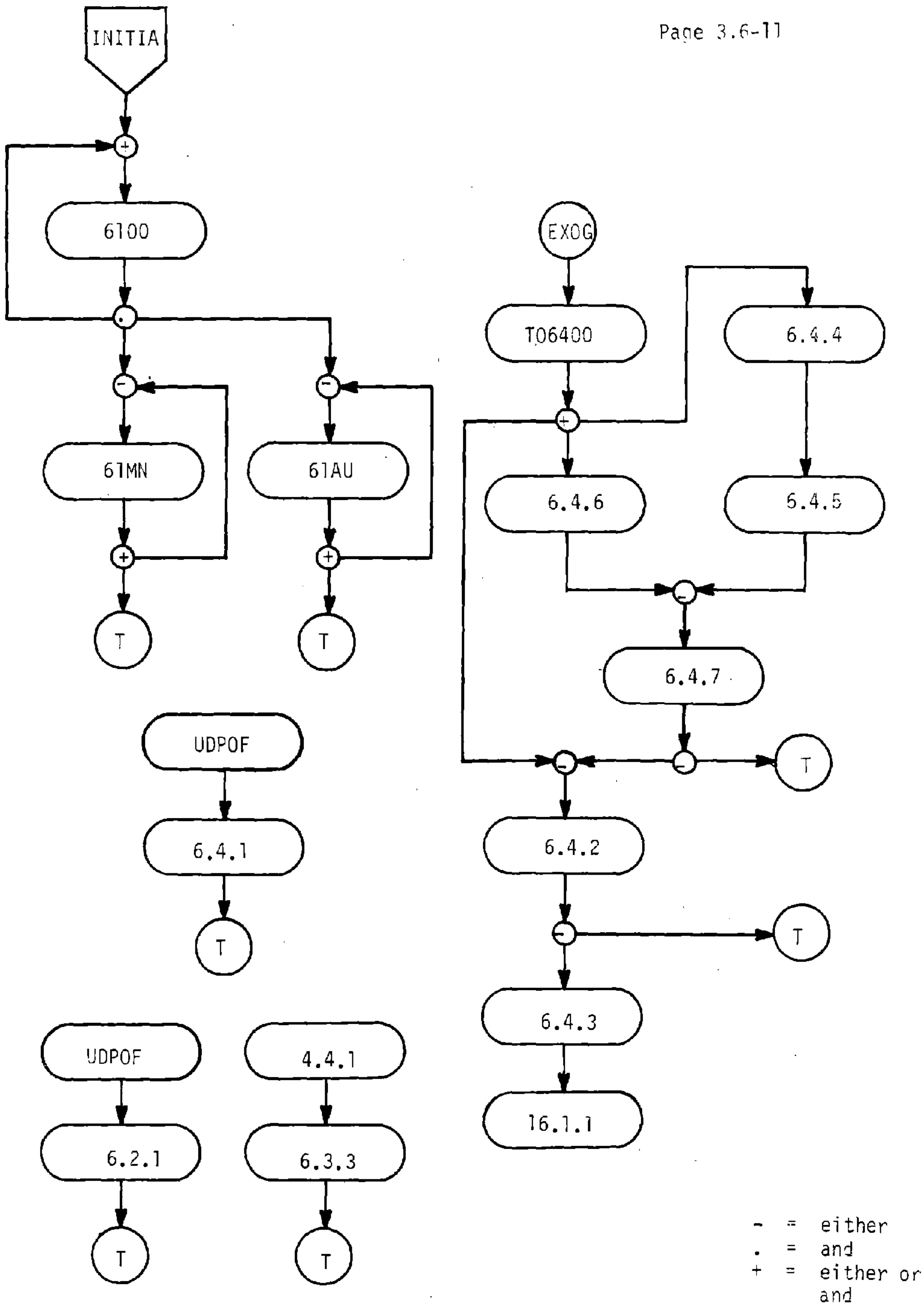

FIGURE 3.6-6 LOGICAL FLOW - FUNCTION 6.0

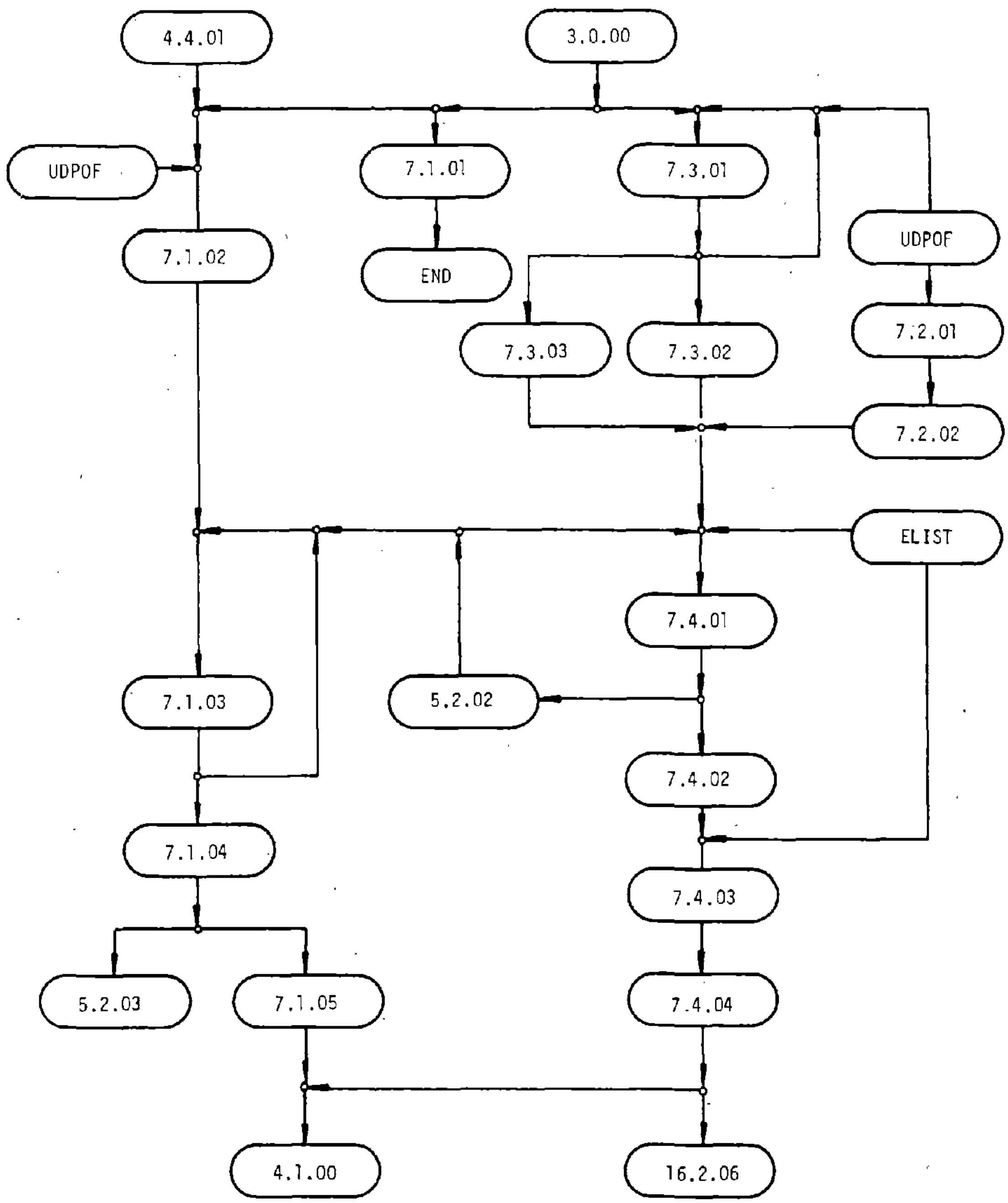

Figure 3.6-7 Logical Flow of Function 7.0

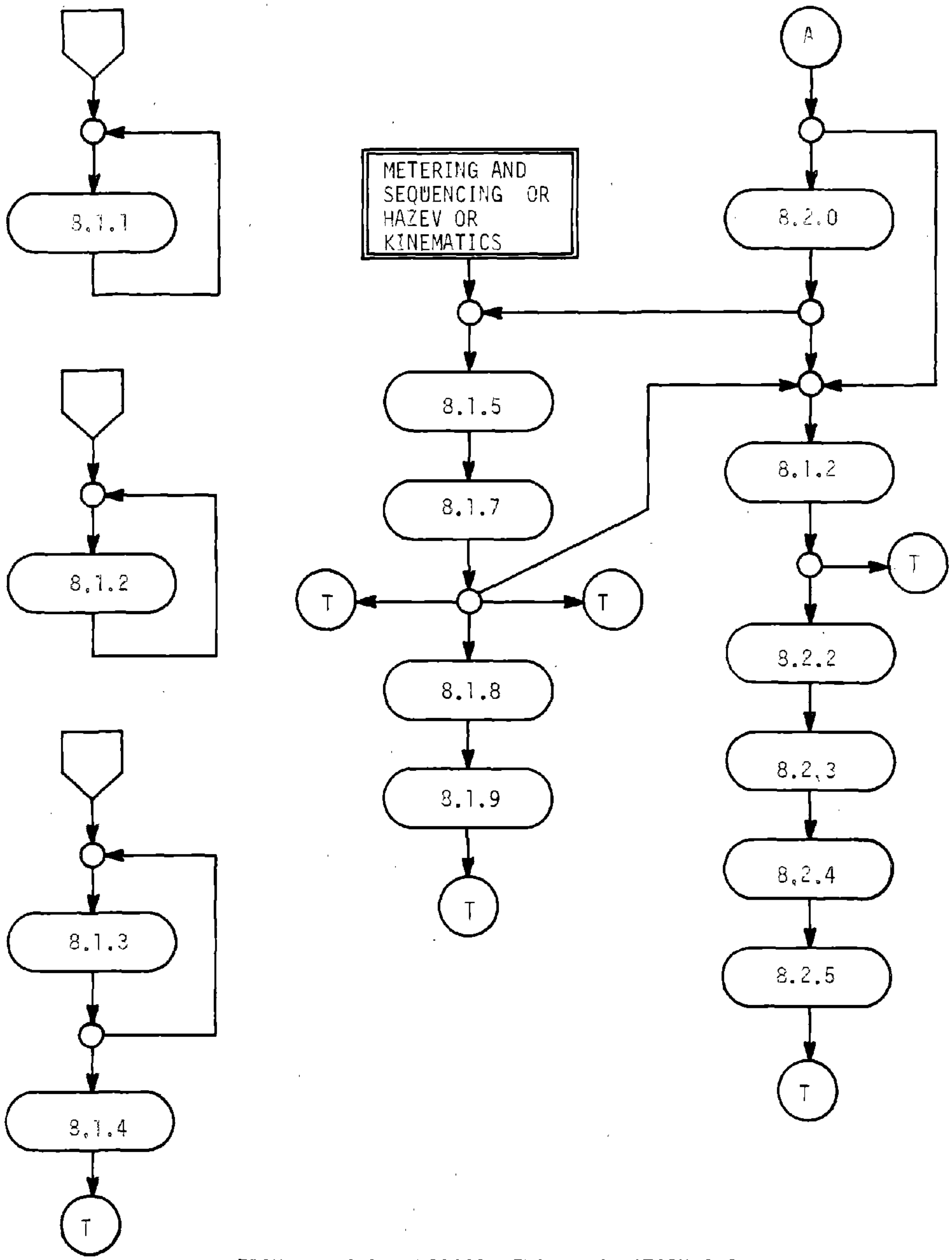

FIGURE 3.6-8 LOGICAL FLOW - FUNCTION 8.0

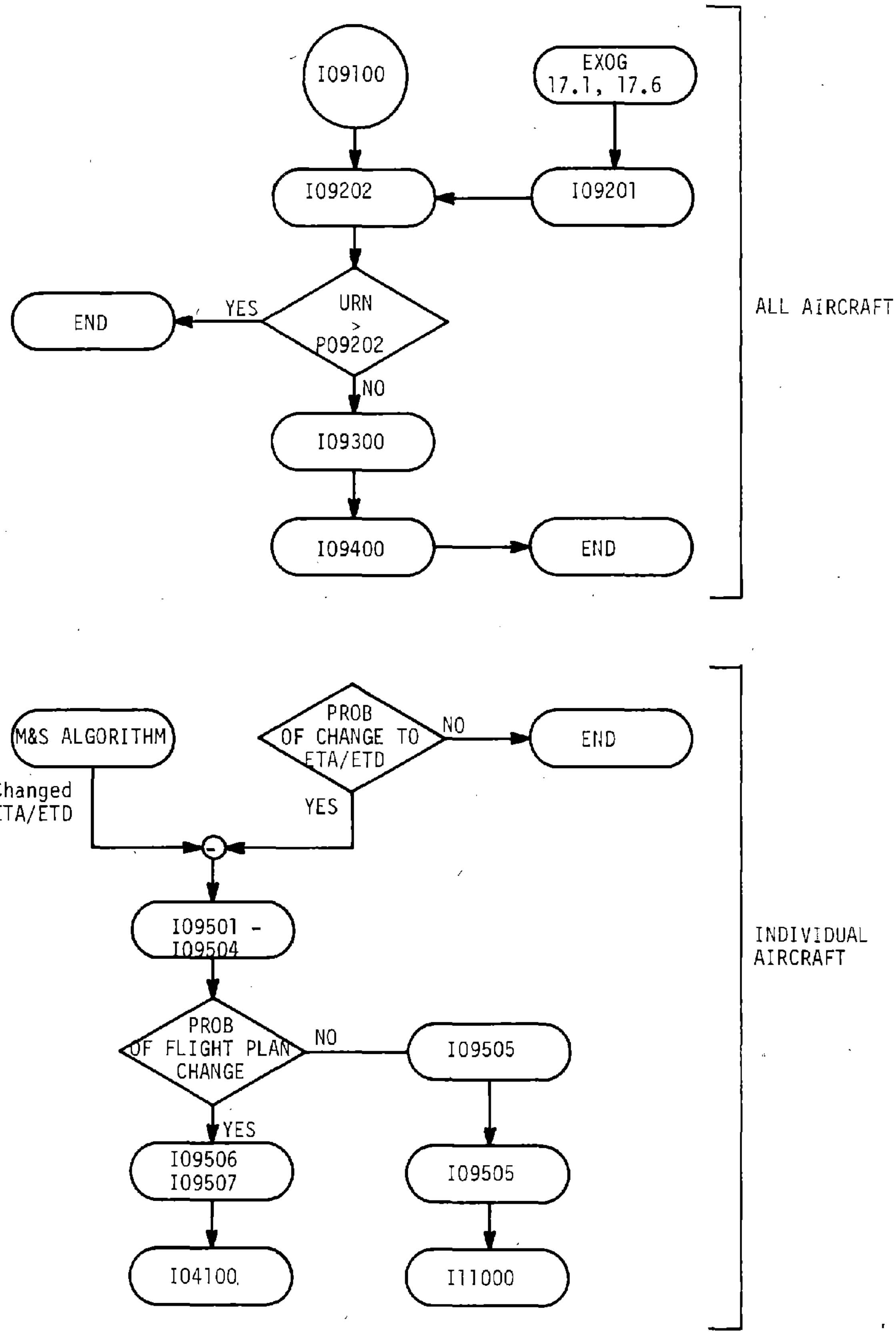

FIGURE 3.6-9 LOGICAL FLOW OF FUNCTION 9.0

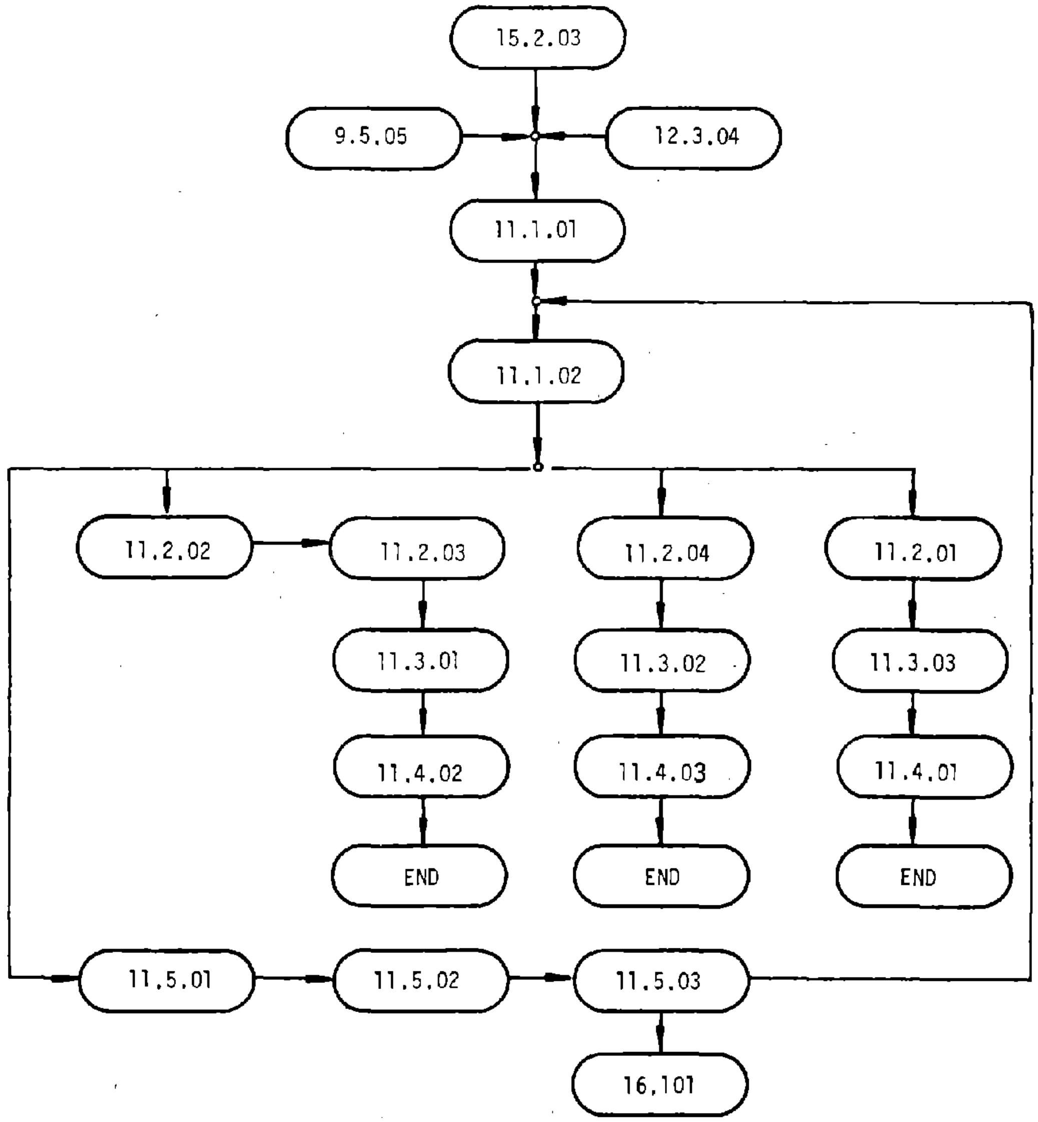

Figure 3.6-10 Logical Flow of Function 11

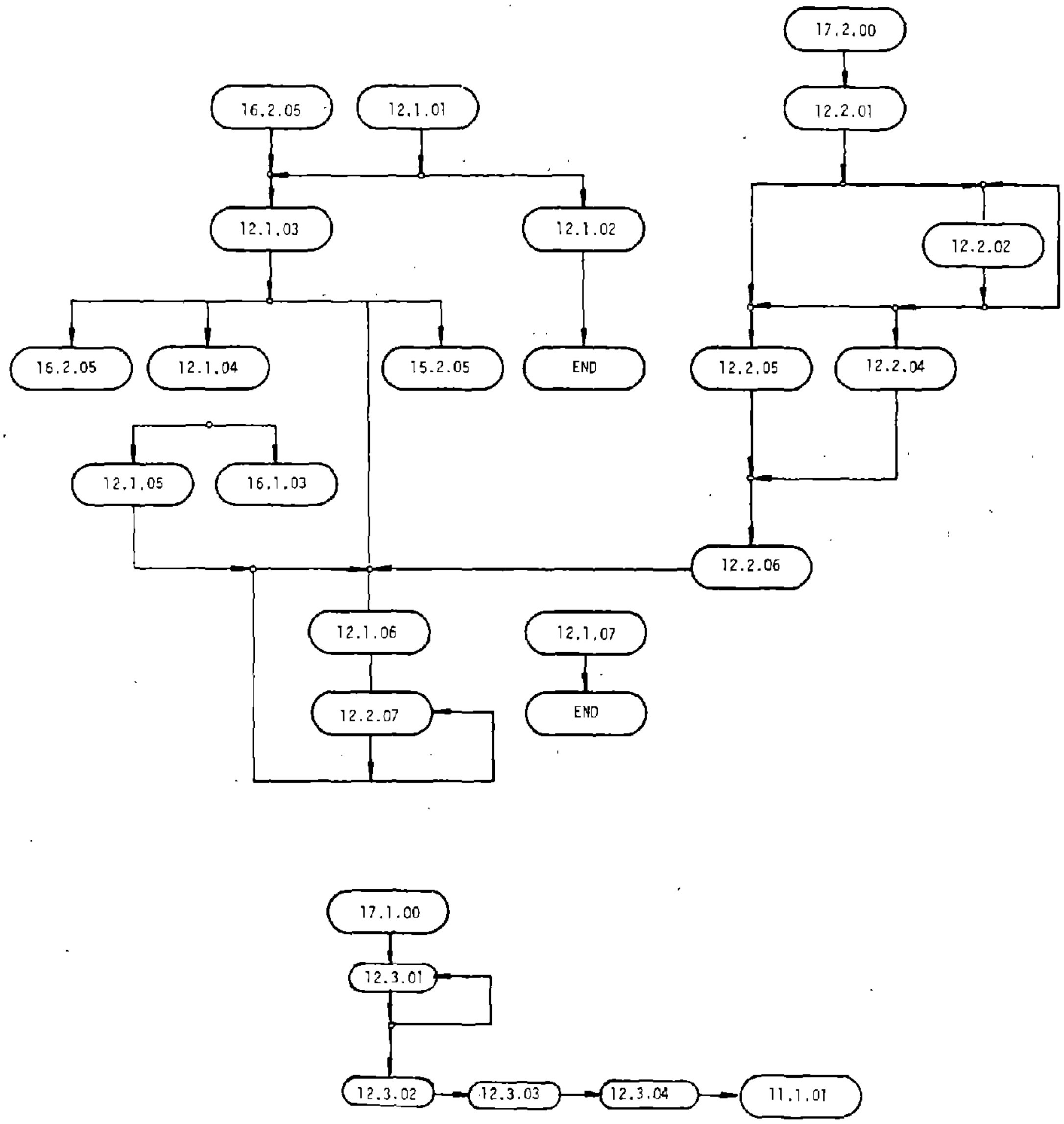

Figure 3.6-11 Logical Flow of Function 12

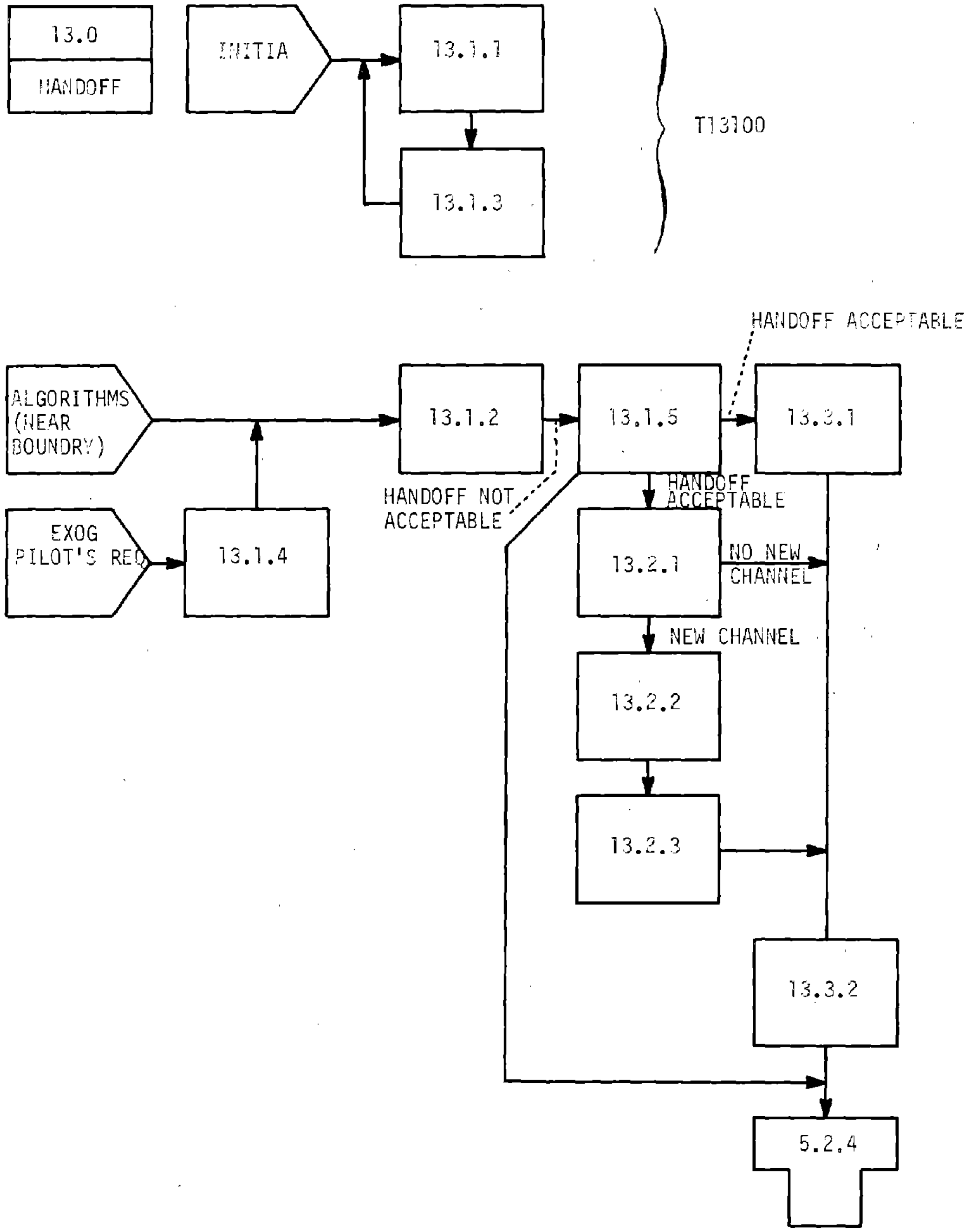

FIGURE 3.6-12 LOGICAL FLOW OF FUNCTION 13.0

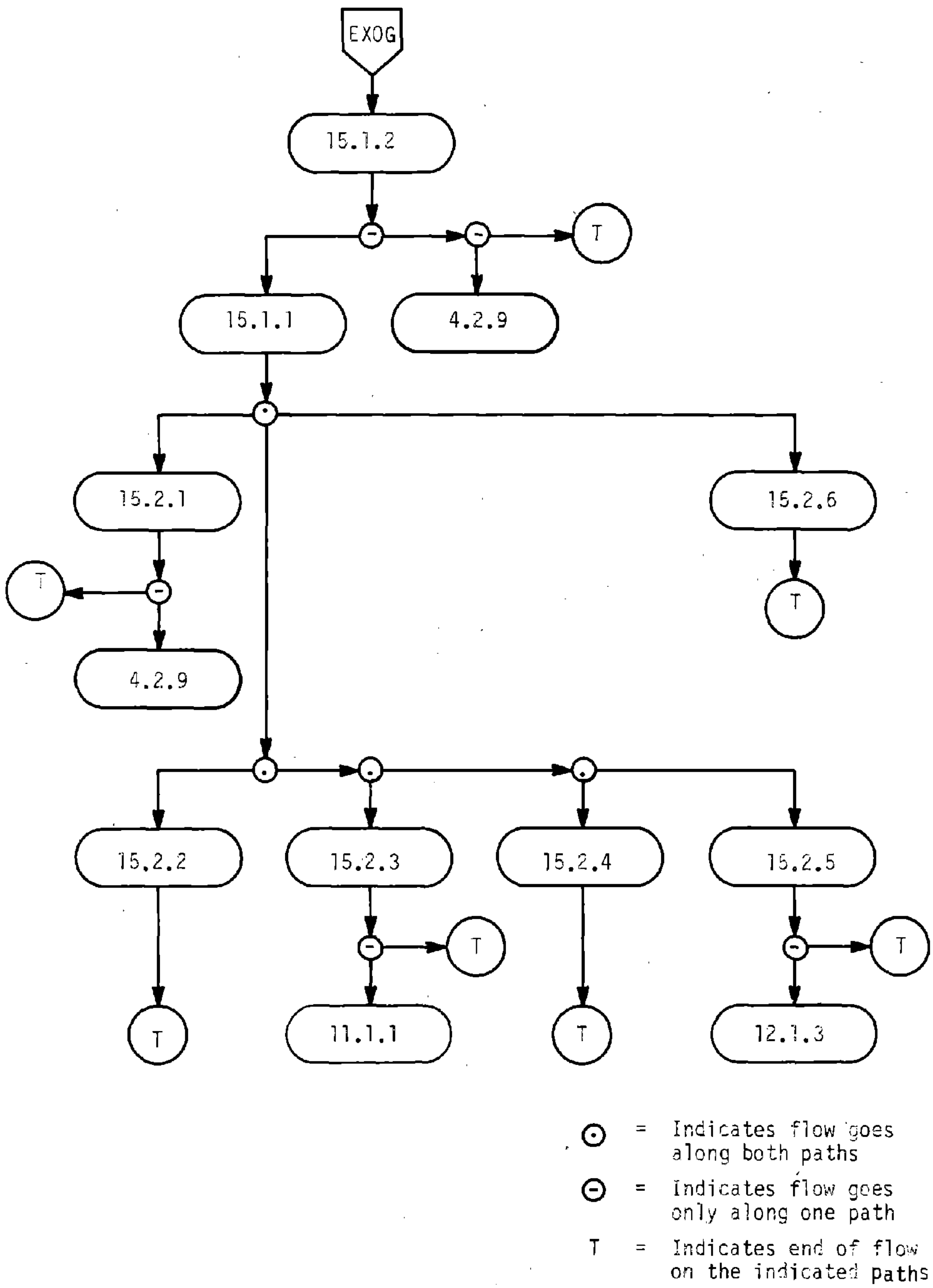

FIGURE 3.6-13 LOGICAL FLOW OF FUNCTION 15.0

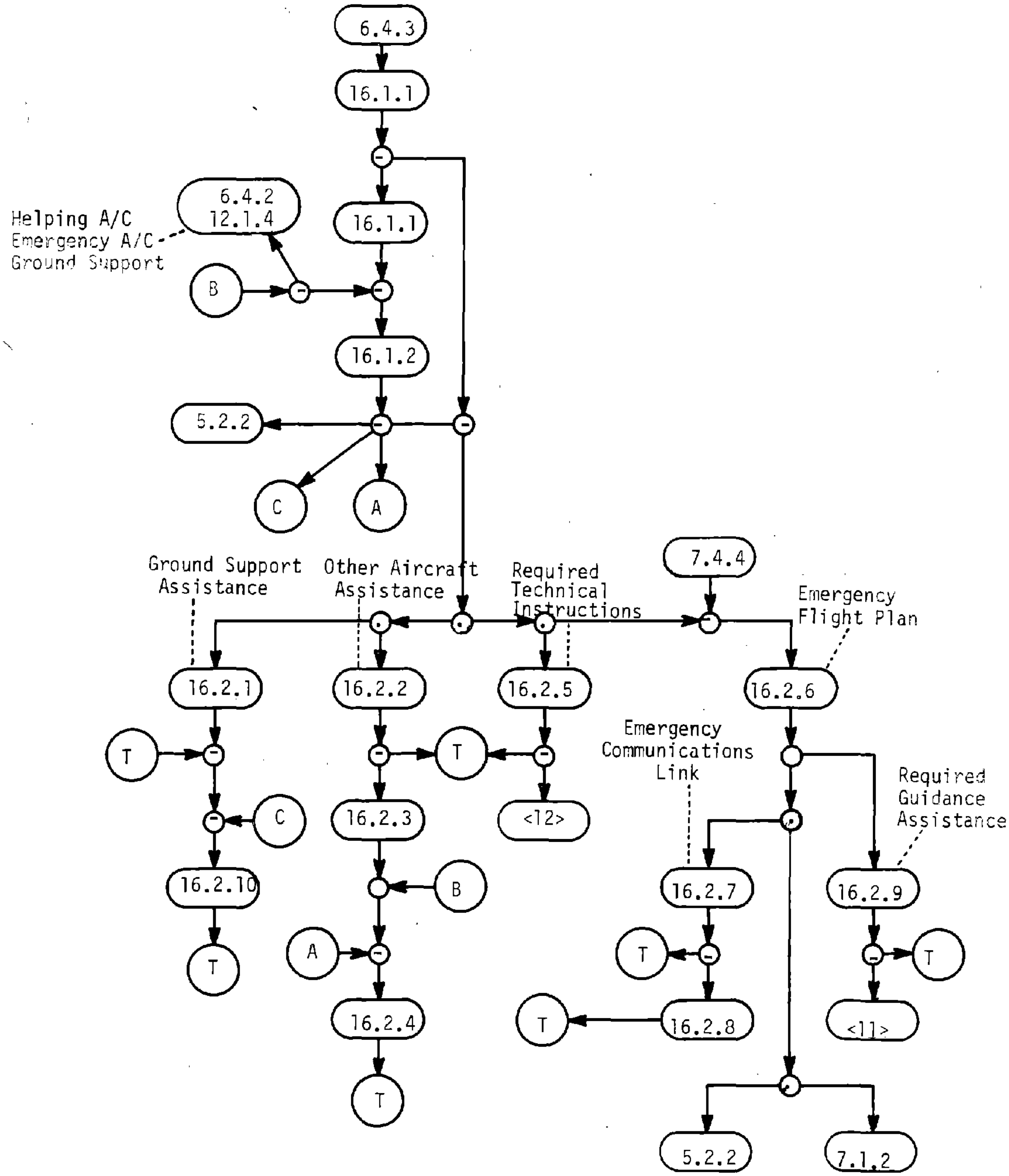

Figure 3.6-14 Logical Flow Of Function 16.0

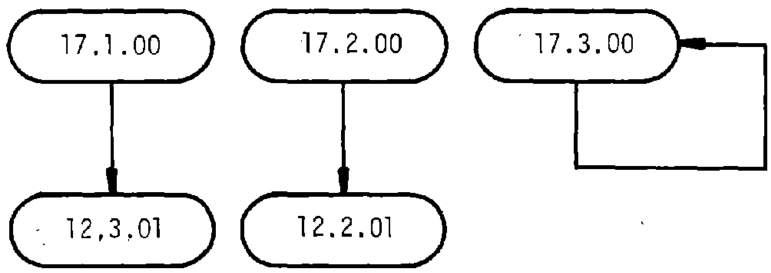

Figure 3.6-15 Logical Flow of Function 17

4.0 DYNAMIC MEMORY FILES

The dynamic memory files are sets of data, each having a variable number of fixed length records. A detailed explanation of the structure and manipulation of these files is given in Section 3.4.2, SALSIM.

Some of the files contain lists of items with their properties. For example, file IACFIL has one record for each aircraft in the system. Each record includes the aircraft type and location. The remaining files consist of queues of events which are processed in a specified order and then deleted from the list. For example, events are both stored in and deleted from the Standard Event Notice file in the order they are to be processed.

Many of the lists are linked to form essentially a list of lists. The relation of these data files is shown in figure 4.0-1, MODEL DATA STRUCTURES. Following this figure are tables showing the contents of each file.

Below is a list of all the dynamic memory files and a synopsis of their contents.

Name	Contents
____	Accumulator block, which contains a list of events to be processed by subroutine ACCUM, as well as a count of parallel tasks.
IADJC	List of jurisdictions adjacent to the floor of the owner jurisdiction.
IADJS	List of jurisdictions contiguous to the side of the owning jurisdiction vertex.
IACFIL	Aircraft file which contains one record for each aircraft in the system.
IACTYP	File of aircraft characteristics. There is one record for each aircraft type.
ICONFL	List of aircraft which required resolution of a conflict in their flight paths. Conflict was recognized by subroutine CONDET.

———	Event notice for subroutine METSEQ, which contains the standard event notice for the periodic stimulation of METSEQ, as well as a list of aircraft incurring a missed approach.
———	Event notice for UDPOF which contains lists of time intervals for aircraft being in a given phase of flight.
HEFIL	List of aircraft which required resolution of a conflict in flight paths. Conflict was confirmed by subroutine HAZARD.
IJURFL	List of jurisdictions.
IVERTX	List of floor vertices for the owner jurisdiction.
IMSQUE	List of aircraft to be processed by subroutine METSEQ. These events are created by subroutines MOVAC and T03000.
———	MOVAC Special Call 4 Event Notice which is a list of aircraft scheduled to make turn.
IREFIL	Resource element file, which holds the tallies for resource utilization.
IRPFIL	Resource pool file which defines the groups of resource elements.
TRWQUE	Runway queue which contains, for each runway, a list of aircraft scheduled to land on the runway.
———	Standard event notice, which is a list of LCC's to be triggered at a specified time.
ITASKQ	List of events (from the standard event notice above) which could not be processed when triggered.
ITPQUE	Time-position queue, which contains for each aircraft the set of fixed points an aircraft must follow from take-off to landing.

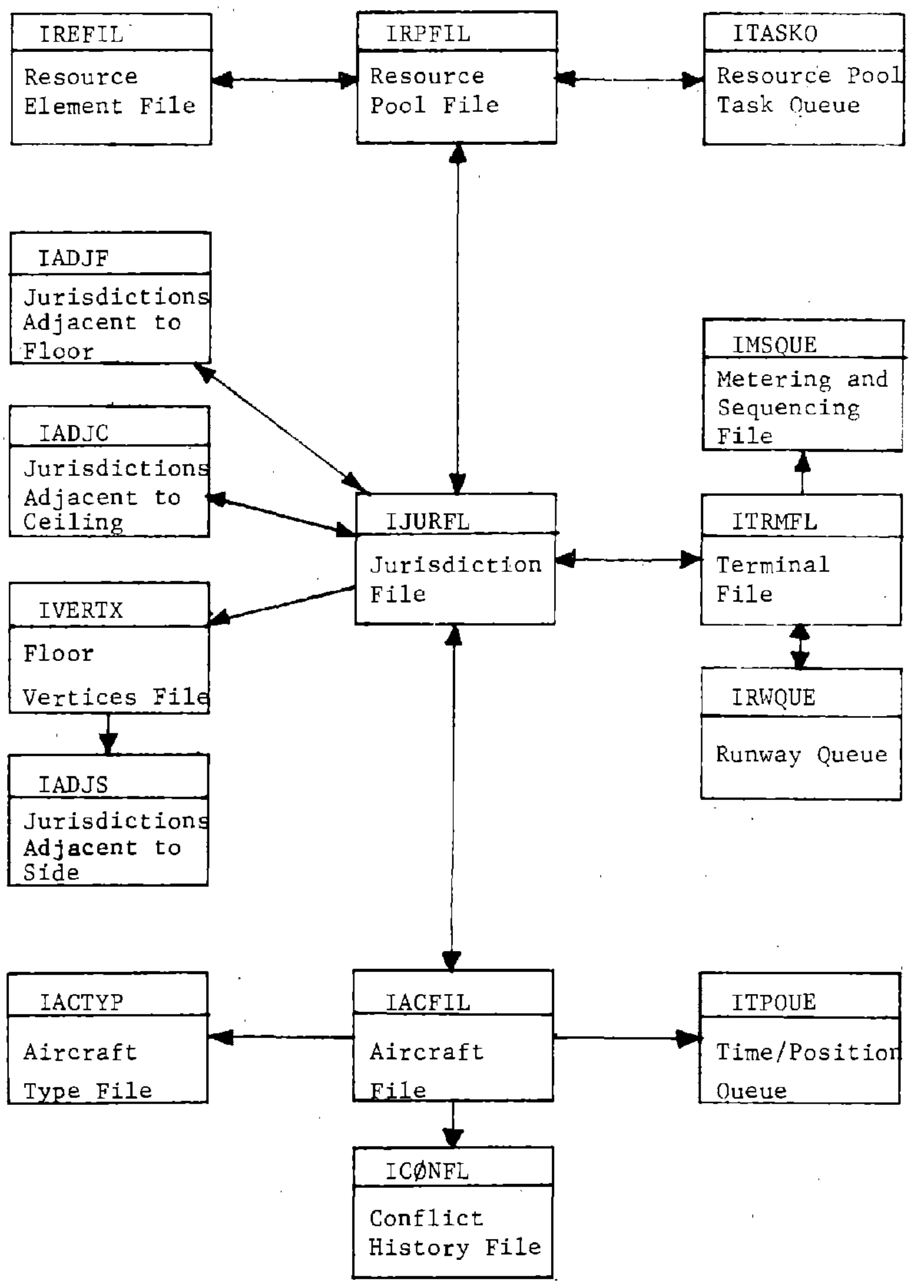

FIGURE 4.0-1 MODEL DATA STRUCTURES

FILE STRUCTURE

PAGE: 1 OF 1

DESCRIPTION: Accumulator Block
FILE NAME: N/A ORDERING SCHEME: N/A
OWNER: N/A LENGTH NAME: LEVNTL

This Block is Used by ACCUM.

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS
IACCUM	1	I	Accumulator	
ILIMIT	2	I	Limit	
	3	-	Not Used	
NXTSK	4	I	Task to be Triggered When IACCUM.GE.ILIMIT	
IHØLD	5	-	Utility Word	
IACFT	6	I	Aircraft or Jurisdiction Pointer	
IRESRC	7	-	Not Used	
IACBLK	8	-	Utility Word	

FILE STRUCTURE

PAGE: 1 OF 1

DESCRIPTION: Adjacent Ceiling Jurisdiction File
FILE NAME: IADJC ORDERING SCHEME: LIFO
OWNER: Jurisdiction File LENGTH NAME: LADJC

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS
IAC	1	I	Successor	
JACJC	2	I	Pointer to Neighboring Jurisdiction (If Negative,	
			No Handoff is Required)	

FILE STRUCTURE

PAGE: 1 OF 1

DESCRIPTION: Adjacent Floor Jurisdictions File
FILE NAME: IADJF ORDERING SCHEME: LIFO
OWNER: Jurisdiction File LENGTH NAME: LADJF

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS
IAF	1	I	Successor	
JADJF	2	I	Pointer to Neighboring Jurisdiction (If Negative,	
			No Handoff is Required)	

FILE STRUCTURE

PAGE: 1 OF 1

DESCRIPTION: Adjacent Side Jurisdiction File
FILE NAME: IADJS ORDERING SCHEME: LIFO
OWNER: Jurisdiction Vertices File LENGTH NAME: LADJS

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS
IAS	1	I	Successor	
JADJS	2	I	Pointer to Neighboring Jurisdiction (If Negative,	
			No Handoff is Required)	

FILE STRUCTURE

PAGE: 1 OF 2

DESCRIPTION: Aircraft File
FILE NAME: IACFIL ORDERING SCHEME: RL on ENDUR
OWNER: Jurisdiction File LENGTH NAME: LACFIL

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS
	1	I	Successor	
	2	I	Predecessor	
ACID	3	A	Identification	
ICLASS	4	I	Avionics User Class	
ITYPE	5	I	Aircraft Type (Pointer to Aircraft Characteristics File	
IPRTY	6	I	Priority of Flight Plan	
IPHASE	7	I	Current Phase of Flight	
ACBITS	8	L	State Vector (CDL)	
ENDUR	9	R	Endurance	Hrs.
IORGAP IJURIS	10	I I	Pointer to Origin Pointer to Current Jurisdiction	
ACETD ETLJ	11	R R	Estimate Time of Departure Estimated Time Left in Jurisdiction	Hrs. Hrs.
NJURIS	12	I	Pointer to Next Jurisdiction	
ACETA	13	R	Estimated Time of Arrival at Destination	Time of Day
IDENT	14	I	Pointer to Destination	
IALTD	15	I I	Pointer to Alternate Destination Pointer to Owner of Runway Que during Aprch or Dep	
IFSTPQ	16	I	Pointer to First Member of Time/Position Queue	
IFFIX	17	I	Pointer to Last Member of Time/Position Queue	

FILE STRUCTURE

PAGE: 2 OF 2

DESCRIPTION: Aircraft File (Cont'd.)

FILE NAME: IACFIL ORDERING SCHEME: RL on ENDUR

OWNER: Jurisdiction File LENGTH NAME: LACFIL

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS
TUPDAT	18	R	Time of Last Update	Time of Day
IDPRW	19	I	Pointer to First Member of Conflict History File Pointer to Departure Runway for Departing Aircraft	
HØLDTM	20	R	Hold Time for Metering and Sequencing Holds	Hrs.
X	21	R	X-coordinate of Aircraft Position	N.Mi.
Y	22	R	Y-coordinate of Aircraft Position	N.Mi
Z	23	R	Z-coordinate of Aircraft Position	N.Mi
XVEL	24	R	X-component of Aircraft Velocity	Kts.
YVEL	25	R	Y-component of Aircraft Velocity	Kts.
ZVEL	26	R	Z-component of Aircraft Velocity	Kts.
N07201	27	I	Pointer to Event Notice for 7.2.1 LCC	
N07301	28	I	Pointer to Event Notice for 7.3.1 LCC	
IPTEVT XPTEVT	29	I R	3 Pointers to MØVAC Special Calls Packed Into One Word	
ERRØR	30	R	3 Coded Navigation Error Terms Packed Into One Word	

FILE STRUCTURE

PAGE: 1 OF 1

DESCRIPTION: Aircraft Type File

FILE NAME: IACTYP   ORDERING SCHEME: LIFO

OWNER: Indexed   LENGTH NAME: LACTYP

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS
	1	I	Successor	
MEMID	2	A	Aircraft Type Identifier	
JSPD	3	I	Speed Class (Velocity/100)	
THETT	4	R	Turn Rate	Rad/Hr.
RMSQA	5	R	Desired Aircraft Horizontal Miss Distance Squared	$(NM)^2$
RALTA	6	R	Desired Aircraft Vertical Miss Distance	NM
IALT	7	I	0 - No Altitude Information Available 1 - Altitude Information Available	
ALTL	8	R	Minimum Altitude	NM
ALTH	9	R	Maximum Altitude	NM
CRATE	10	R	Climb or Dive Rate	NM/Hr.
TLEAD	11	R	Minimum Lead Time for Metering and Sequencing	Hrs.
FSF	12	R	Flight Endurance Margin	Hrs.
ACRØT	13	R	Runway Occupancy Time	Hrs.
THETH	14	R	Turn Rate in Hold Pattern	Rad/Hr
DTRTM	15	R	Departure Transition Time Interval	Hrs.
ITYPAC	16	I	Aircraft Type Number	

FILE STRUCTURE

DESCRIPTION: Conflict History File
FILE NAME: ICØNFL ORDERING SCHEME: LIFØ
OWNER: Aircraft File LENGTH NAME: KNFLEN

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS
	1	I	Successor	
ID2	2	I	Pointer to Second Aircraft in Conflict Pair	
CNTR	3	I	Indicator for Action History	

FILE STRUCTURE

PAGE: 1 OF 1

DESCRIPTION: Event Notice For METSEQ
FILE NAME: N/A ORDERING SCHEME: RL on Time
OWNER: N/A LENGTH NAME: LEVNTL

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS	
	1	I	Successor		
	2	I	Predecessor		
	3	R	Time	Hrs.	
TESCAP	3	R	Time Between IAPRCH and MSDAPR if Any	Δ Hrs.	*
	4	I	Standard Call Indicator (.EQ. 1)		
IACPØF	4	I	Pointer to Aircraft if Special Call (.NE. 1)		*
	5	I	Not Used		
TMSKAP	6	R	Time Between MSDAPR and IAPRCH	Δ Hrs.	*
TAPRCH	7	R	Time Between IAPRCH and LANDING	Δ Hrs.	*
TLNDNG	8	R	Time Between LANDING and CANAC	Δ Hrs.	*

* For Special Calls Due to MSDAPR

FILE STRUCTURE

PAGE: 1 OF 1

DESCRIPTION: Event Notice for UDPØF
FILE NAME: N/A ORDERING SCHEME: RL on Time
OWNER: N/A LENGTH NAME: LEVNTL

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS
	1	I	Successor	
	2	I	Predecessor	
	3	R	Time	Hrs.
TESCAP	3	R	Time Between IAPRCH And MSDAPR if Any	ΔHrs.
IACPØF	4	I	Pointer to Aircraft (.NE.1)	
IRWPTR	5	I	Pointer to IRWQUE Member	
ITAKØF	6	R	Time Between ITAKØF And IDEPTR	ΔHrs.
TARRTR	6	R	Time Between IARRTR And IAPRCH If Any	ΔHrs.
TMSDAP	6	R	Time Between MSDAPR And IAPRCH If Any	ΔHrs.
IDEPTR	7	R	Time Between IDEPTR And INRØUT	ΔHrs.
TAPRCH	7	R	Time Between IAPRCH And LANDING	ΔHrs.
TENRTE	8	R	Time Between INRØUT And Anything Else (Not Used)	ΔHrs.
TLNDNG	8	R	Time Between LANDING And CANAC	ΔHrs.

FILE STRUCTURE

PAGE: 1 OF 1

DESCRIPTION: Hazard Evaluation Data
FILE NAME: HEFIL   ORDERING SCHEME: N/A
OWNER: N/A   LENGTH NAME: LEVNTL

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS
IDCØL1	1	I	Pointer to First Aircraft of Conflict Pair	
IDCØN2	2	I	Pointer to Second Aircraft of Conflict Pair	
HEDELT	3	R	Utility Word*	
HEIMM	4	R	Utility Word*	
HERISK	5	R	Utility Word*	
INDXHE	6	I	Utility Word*	
HEPACK	7	I	Utility Word*	
	8		Not Used	
			* See Table 4.0-1 Cross Reference of the	
			Conflict Pair Information Block	

Haz. Eval. Data Alg. Variable or Subroutine	IDCON1 (IPTR)	IDCON2 (IPTR)	HEDELT (IPTR)	HEIMM (IPTR)	HERISK (IPTR)	INDXHE (IPTR)	HEPACK (IPTR)
METERING AND SEQUENCING	Pointer to Aircraft (lower number pointer)	Pointer to Aircraft (higher number pointer)	Length of time which this pair is to be checked by 8.1.5	Imminence	Risk		
KINEMATICS	Pointer to Aircraft (lower number pointer)	Pointer to Aircraft (higher number pointer)	Length of time which this pair is to be checked by 8.1.5	Imminence	Risk		
HAZARD EVALUATION (HAZEV)	Pointer to Aircraft (lower number pointer)	Pointer to Aircraft (higher number pointer)	Time until HAVEV next checks this aircraft pair	Imminence	Risk	Index for this aircraft pair for RESØLV	Packed information variable for RESØLV
T08105			X				
T08107			X	X	X		
T08108			X				
T08109			X				
T08200 (before RESØLV call)			A coefficient for lag time determination	B coefficient for lag time determination			
RESØLV	X	X	X	X	X	X	X Hypothesize /Analyze # of loops
T08200 (after RESØLV call)			X Number of loops through this subfunction	X Additional pilot notification loop (0 or 1) (ADDPRL)	Total pilot delay between 8.2.4 and 8.2.5		
T08201			X				X
T08202			X				X
T08203			X				
T08204			X	X	X		
T08205			X	X			

NOTE: X - Last previsouly defined variable used in the indicated subroutine. Writing indicates the value that variable is set to in the subroutine.

TABLE 4.0-1 CROSS REFERENCE OF THE CONFLICT PAIR INFORMATION BLOCK

FILE STRUCTURE

PAGE: 1 OF 1

DESCRIPTION: Jurisdiction File
FILE NAME: IJURFL ORDERING SCHEME: LIFO
OWNER: Indexed LENGTH NAME: LJURFL

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS
	1	I	Successor	
MEMID	2	A	Jurisdiction Identifier	
	3		Not Used	
	4	I	Pointer to First Aircraft Owned by Jurisdiction	
	5	I	Pointer to Last Aircraft Owned by Jurisdiction	
NACFT	6	I	Number of Aircraft Owned by Jurisdiction	
ICAP	7	I	Capacity of Jurisdiction if .GT.0 Pointer to Terminal File if .LT.0	
IPTVRT	8	I	Pointer to Jurisdiction Vertices File	
IPTJAC	9	I	Pointer to Adjacent Floor Jurisdictions File	
IPTJAF	10	I	Pointer to Adjacent Ceiling Jurisdictions File	
CEIL	11	R	Jurisdiction Ceiling Altitude	N.MI
FLØØR	12	R	Jurisdiction Floor Altitude	N.MI
JRSCPL + 1	13	I	Pointer to Resource Pool File for Pool 1	
	14	I	Pointer to Resource Pool File for Pool 2	
	⋮			
	12+n	I	Pointer to Resource Pool File for Pool n	

FILE STRUCTURE

PAGE: 1 OF 1

DESCRIPTION: Jurisdiction Vertices File
FILE NAME: IVERTX
ORDERING SCHEME: LIFO (Clockwise)
OWNER: Jurisdiction File
LENGTH NAME: LVERTX

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS
IVX	1	I	Successor	
IDS	2	I	Pointer to IADJS File	
XVERT	3	R	X-coordinate of Vertex	N.MI
YVERT	4	R	Y-coordinate of Vertex	N.MI

FILE STRUCTURE

PAGE: 1 OF 1

DESCRIPTION: Metering and Sequencing Queue
FILE NAME: IMSQUE ORDERING SCHEME: RL on RANKS
OWNER: Terminal File LENGTH NAME: LMSQUE

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS
	1	I	Successor	
	2	I	Predecessor	
RANKS	3	R	Ranking Variable (Speed Class and ETA or ETD)	
IMSAC	4	I	Pointer to Aircraft File	

FILE STRUCTURE

PAGE: 1 OF 1

DESCRIPTION: MØVAC Special Call 4 Event Notice
FILE NAME: N/A   ORDERING SCHEME: N/A
OWNER: N/A   LENGTH NAME: LEVNTL

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS
	1	I	Successor	
	2	I	Predecessor	
	3	I	Time	
XTØ	4	R	Proposed X-coordinant	N.MI
IHØLD	5	I	Special Call Indicator (.EQ. 4)	
IACID	6	I	Pointer to Aircraft File	
YTØ	7	R	Proposed Y-coordinant	N.MI
ZTØ	8	R	Proposed Z-coordinant	N.MI

FILE STRUCTURE

DESCRIPTION: Resource Element File
FILE NAME: IREFIL ORDERING SCHEME: RL on TNA
OWNER: Resource Pool File LENGTH NAME: LREFIL

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS
STCBT	1	I	Successor If Not Busy Short Term Cumulative Busy Time If Busy	Hrs.
IRSTAT	2	I	Predecessor If Not Busy Resource Status or ITSKPT If .LT.0	
TNA	3	R	Time Next Available (TNA=TFA+$\frac{CBT}{ØLT}$)	Hrs.
TFA	4	R	Time First Available	Hrs.
CBT	5	R	Cumulative Busy Time	Hrs.
IPØØL	6	I	Pointer to Resource Pool	
	7	I	Successor of Resource Element	

FILE STRUCTURE

PAGE: 1 OF 1

DESCRIPTION: Resource Pool File
FILE NAME: IRPFIL   ORDERING SCHEME: N/A
OWNER: Pointed to by Jurisdiction File   LENGTH NAME: LRPFIL

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS
IJORPL	1	I	Pointer to Jurisdiction File.	
NØELMT	2	I	Current { Computer Rate for Automated Resource Number of Elements for Manual Resource	10^2 Inst/Hr. Elements
MEXLMT	3	I	Maximum { Computer Rate Number of Elements	10^2 Inst/Hr. Elements
	4	I	Pointer to First Task in Queue	
	5	I	Pointer to Last Task in Queue	
IFSTRE	6	I	Pointer to First Available Resource Element in Pool	
IPLTYP	7	I	Pointer to Last Available Resource Element. (If .LT.0, this word indicates an Automated Pool)	
IPLCHR	8	I	Pool Characteristic Displacement in Jurisdiction File	
	9	I	Pointer to First Resource Element Owned by Pool	

FILE STRUCTURE

PAGE: 1 OF 1

DESCRIPTION: Runway Queue
FILE NAME: TRWQUE ORDERING SCHEME: RL on ARTIME
OWNER: Terminal File LENGTH NAME: LRWQUE

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS
ISUCR	1	I	Successor	
IPRDR	2	I	Predecessor	
IACPTR	3	I	Pointer to Aircraft File	
ARTIME	4	R	Runway Arrival Time	Hrs.
RWCTM	5	R	Runway Clearance Time	Hrs.
VELAC	6	R	Average Sequencing Route Velocity	Kt.
JALTD ACTA	7	I R	Holder for IALTD for Departures Current Advance Time Available for Arrivals	Δ Hrs.
ACTR	8	R	Current Retard Time Available for Arrivals	Δ Hrs.
TIMDEP IFETA	9	R I	Departure Time Interval Pointer to Feeder Fix for Arrivals	Δ Hrs.
IPØF	10	I	Pointer to UDPØF Event Notice	

FILE STRUCTURE

PAGE: 1 OF 1

DESCRIPTION: Standard Event Notice
FILE NAME: N/A ORDERING SCHEME: RL on TIME
OWNER: N/A LENGTH NAME: LEVNTL

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS
	1	I	Successor	
	2	I	Predecessor	
	3	R	Time	Hrs.
IBLØCK NXTSK	4	I	Next Task for Select or Delay Return Flag	
IHØLD	5	I	Utility Word	
IACFT	6	I	Pointer to Aircraft File If .GT.0 Pointer to Jurisdiction File If .LT.0	
IRESRC	7	I	Pointer to Resource Element File If .GT.0 Pointer to Resource Pool If .LT.0	
			Indicator that the Task is Unassigned if .EQ.0	
WØRKTM IACBLK	8	R I	Accumulator or Utility Word	

FILE STRUCTURE

PAGE: 1 OF 1

DESCRIPTION: Task Queue
FILE NAME: ITASKQ
ORDERING SCHEME: RH on IPRIØ
OWNER: Resource Pool File
LENGTH NAME: LEVNTL

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS
	1	I	Successor	
	2	I	Prececessor	
IPRIØ	3	I	Priority	
NXTSK	4	I	Task Number	
IHØLD	5	I	Utility Word	
IACFT	6	I	Aircraft or Jurisdiction Pointer	
IRESRC	7	I	Pointer to Resource Pool (Negative)	
WØRKTM IACBLK	8	R I	Utility Word	

FILE STRUCTURE

PAGE: 1 OF 1

DESCRIPTION: Terminal File

FILE NAME: ITRMFL ORDERING SCHEME: LIFO

OWNER: Indexed, Pointed to by Jurisdiction File LENGTH NAME: LTRMFL

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS
	1	I	Successor	
	2	I	Pointer to Jurisdiction File	
ITERM	3	I	Pointer to Mass-Storage Terminal File	
	4	I	Pointer to First Metering and Sequencing Queue Entry	
	5	I	Pointer to Last Metering and Sequencing Queue Entry	
PMSDAP	6	R	Probability of Missed Approach	
IQAQD	7	I	Total Number of Runway Queue Entries (Carried Negative)	
NRW	8	I	Number of Runways	
	9	I	Pointer to First Runway Queue Entry for Runway 1	
	10	I	Pointer to Last Runway Queue Entry for Runway 1	
	11	I	Pointer to First Runway Queue Entry for Runway 2	
	12	I	Pointer to Last Runway Queue Entry for Runway 2	
	⋮			
	9+2*(NRW-1)	I	Pointer to First Runway Queue Entry for Runway NRW	
	10+2*(NRW-1)	I	Pointer to Last Runway Queue Entry for Runway NRW	

FILE STRUCTURE

PAGE: 1 OF 1

DESCRIPTION: Time/Position Queue
FILE NAME: ITPQUE ORDERING SCHEME: RL on ETA
OWNER: Aircraft File LENGTH NAME: LTPQUE

ATTRIBUTES

NAME	WORD NO.	DATA TYPE	DESCRIPTION	UNITS
ISUCC	1	I	Successor	
IPRED	2	I	Predecessor	
ETA	3	R	Estimated or Intended Time of Arrival	Hrs.
XINTND	4	R	Intended X-coordinant	N.MI
YINTND	5	R	Intended Y-coordinant	N.MI
ZINTND	6	R	Intended Z-coordinant	N.MI
SPDLEG	7	R	3-Dimensional Speed for Next Route Leg if .GT.0. Pointer to Arrival Runway if .LT.0.	Kts.
TLEG	8	R	Time to the Next Fix Point if .GT.0. Pointer to Feeder Fix if .LT.0.	ΔHr. --
			End of Flight Plan Indicator if .EQ.0.	

5.0 PROGRAM OVERLAYS

The DELTA simulations consists of over 200 subroutines and has variable core requirements depending on the dynamic memory specified. The model has an input and utilization phase and a simulation phase. There is an obvious split between these two phases which lends itself to overlaying. However, the bulk of the executable code is within the simulation phase and it is desirable to overlay within this phase so as to not use an inordinate amount of central memory. Because the 165 functional task subroutines are executed in what amounts to an unpredictable sequence, it is extremely difficult to define a "logical" overlay structure which will avoid excessive loading of overlays during execution. Consequentially, the subroutines were grouped together by functions and algorithms.

During development it was desirable to execute the DELTA model in timeshare mode to be able to use the Debug facility available with TRW/TSS. In order to do this, the model was forced to fit in 60_8 K words of core. This was accomplished by heavily overlaying the algorithms and keeping the dynamic memory to the minimum value of 2_{10} K words. Figure 5.0-1 shows the LINK directives needed to fit the DELTA simulation into 60 K core.

However, the 60 K overlay is extremely inefficient because of excessive loading of overlay segments. For production runs, it is desirable to execute them in batch mode. This can be accomplished by using the SUBMIT command to make remote job submission and makes available up to 380_8 K words of central memory for use. Figure 5.0-2 shows the LINK directives used for production runs.

This overlay requires approximately 117_8 K words of core for a 15_{10} K word dynamic memory allocation. Output from the Post Processor showed certain tasks were being performed with high frequency compared to most other tasks. To improve the overlay efficiency, these tasks were moved into Segment B which is resident continuously during the simulation phase. Further experience with the DELTA model may suggest changes in this overlay structure to improve the overall execution efficiency and cost.

Figure 5.0-1 Link Directives for DELTA Simulation to fit into 60_8 K Core.

```
ENTRY MAIN
OPTIONS AL,CL,CM,LP
   OVERLAY ROOT
     INSERT MAIN,ABORTT,XXSTRT,BLKDATA,B,C,RANF
   OVEPLAY A,ROOT
     INSERT INITIA,DEFSET,CYSTART,INIT,IOPTR
   OVERLAY B,ROOT
     INSERT SIMRUN,EXOG,ELIST
     INSERT ACCUM,ALLOC,GETAC,GETASK,GETIME,GFTOLR
     INSERT IDECSN,MODAC,PILOTR,TSELEC,RSPEC1
    INSFRT INITMS,READMS,CLOSEMS,WRITMS
     INSERT T06100,T061MN,T061AU
     INSERT T08102,T08103,T08104
     INSERT T13100,T13102,T13104,T13105,T13201
     INSERT T13202,T13203,T13301,T13302
     INSERT UDPOF,T06401
 REGION R0,B
   OVERLAY F01,*R0
     INSERT EL01
     INSERT T01101,T01102,T01103,T01201,T01202
     INSERT T01301,T01302,T01303
   OVERLAY F02,*R0
     INSERT T02000,T02AUT,T02MAN,T02ALG,T02202
   OVERLAY F03,*R0
     INSERT T03000,IOPTR
     INSERT EL04
     TNSERT T04100,T04101,T04102,T04201,T04202,T04203
     INSERT T04204,T04205,T04206,T04207,T04208
     INSERT T04209,T04210,T04211,T04212,T04213
     INSERT T04301,T04302,T04303,T04304
     IVSERT T04401,T04403,T04404
   OVERLAY F05,*R0
     INSERT EL05
     INSERT T05101,T05102,T05103
     INSERT T05201,T05202,T05224,T05203,T05204
     INSERT T05301,T05302,T05303
   OVERLAY F06,*R0
     INSERT EL06
     INSERT T06201,T06303
     INSERT T06400,T06402,T06403,T06404
     INSERT T06405,T06406,T06407
   OVERLAY F07,*R0
```

(Continued)

Figure 5.0-1 Link Directives for DELTA Simulation to fit into 60_8 K Core.

```
  INSERT EL07
  INSERT T07101,T07102,T07103,T07104,T07105
  INSERT T07201,T07202
  INSERT T07301,T07302,T07303
  INSERT T07402,T07404
OVERLAY F08,*R0
  INSERT EL08
  INSERT T08101,T08105
  INSERT T08107,T08108,T08109,T08201
  INSERT T08202,T08203,T08204,T08205
OVERLAY F09,*R0
  INSERT EL09
  INSERT T09100,T09201,T09202,T09300
  INSERT T09400,T09501,T09505,T09506
OVERLAY F11,*R0
  INSERT EL11
  INSERT T11101,T11102,T11201,T11202,T11203
  INSERT T11204,T11301,T11302,T11303,T11401
  INSERT T11402,T11403,T11501,T11502,T11503
OVERLAY F12,*R0
  INSERT EL12
  INSERT T12101,T12102,T12103,T12104,T12105
  INSERT T12106,T12107,T12201,T12202,T12204
  INSERT T12205,T12206,T12207,T12301,T12302
  INSERT T12303,T12304
OVERLAY F15,*R0
  INSERT EL15
  INSERT T15101,T15102,T15201,T15202,T15203
  INSERT T15204,T15205,T15206
OVERLAY F16,*R0
  INSERT EL16
  INSERT T16101,T16102,T16103,T16201,T16202,T16203
  INSERT T16204,T16205,T16206,T16207,T16208,T16209
  INSERT T16210
OVERLAY F17,*R0
  INSERT EL17
  INSERT T17100,T17200,T17300,IOPTR
OVERLAY F18,*R0
  INSERT CANAC,T04402
  INSERT ORPAC
  INSERT TCANAC
  INSERT RESEQ,CANEVT1
```

```
  OVERLAY F19,*R0
    INSERT DATAFIL
  OVERLAY F20,*R0
    INSERT PFIND2,PFIND,METSEQ,VELCAL,R999
    INSERT PFIND1,FACL,RACL,GRWCRS
    INSERT SLOCH,RESCH,ADJUST,UPTPQ,BSTSLN,RACIT
  OVERLAY F21,*R0
   INSERT MOVAC,SCALER
   INSERT PHOLD,ACMS1,ACJBC2,CANEVT,ACT4
   INSERT CONDET,MINVEC,MAXVEC,APRCHT
   INSERT CRASH
   INSERT INTVEL,ACNE,GETXYZ,PUTXYZ,ACERR,SETETA,STOREV
   INSERT SPDWRD,SPOPOS,SPEEB
   INSERT BNDRY
   INSERT INJUR,INSIDE,CONVEX,JURFND
   INSERT PTCRSI,LINE,PTTOLI,LINSOL
   INSERT TURN,TPQPTR
  OVERLAY F22,*R0
    INSERT T07401,T07403,TOLFX5,TOLRES
    INSERT TPQPTR,INTVEL,FLYTM,SCALER,SETETA
    INSERT CANEVT,SPDWRD,SPEEB,STOREV,TURN
 ENDREG R0
END
```

(Continued)

Figure 5.0-1 Link Directives for DELTA Simulation to fit into 60_8 K Core.

Figure 5.0-2 Link Directives for Production Runs.

```
ENTRY MAIN
OPTIONS AL,CL,CM,LP
   OVERLAY ROOT
     INSERT MAIN,ABORTT,XXSTRT,BLKDATA,B,C,RANF
   OVERLAY A,ROOT
     INSERT INITIA,DEFSET,CYSTART,INIT,IOPTR
   OVERLAY B,ROOT
     INSERT SIMRUN,EXOG,ELIST
     INSERT ACCUM,ALLOC,GETAC,GETASK,GETIME,GETOLR
     INSERT IDECSN,MODAC,PILOTR,TSELEC,RSPEC1
    INSERT INITMS,READMS,CLOSEMS,WRITMS
 REGION R0,B
   OVERLAY F01,*R0
     INSERT EL01
     INSERT T01101,T01102,T01103,T01201,T01202
     INSERT T01301,T01302,T01303
   OVERLAY F02,*R0
     INSERT T02000,T02AUT,T02MAN,T02ALG,T02202
   OVERLAY F03,*R0
     INSERT T03000,IOPTR
   OVERLAY F04,*R0
     INSERT EL04
     INSERT T04100,T04101,T04102,T04201,T04202,T04203
     INSERT T04204,T04205,T04206,T04207,T04208
     INSERT T04209,T04210,T04211,T04212,T04213
     INSERT T04301,T04302,T04303,T04304
     INSERT T04401,T04403,T04404
   OVERLAY F05,*R0
     INSERT EL05
     INSERT T05101,T05102,T05103
     INSERT T05201,T05202,T0522A,T05203,T05204
     INSERT T05301,T05302,T05303
   OVERLAY F06,*R0
     INSERT EL06
     INSERT T06100,T061AU,T061MN,T06201,T06303
     INSERT T06400,T06401,T06402,T06403,T06404
     INSERT T06405,T06406,T06407
   OVERLAY F07,*R0
     INSERT EL07
     INSERT T07101,T07102,T07103,T07104,T07105
     INSERT T07201,T07202
     INSERT T07301,T07302,T07303
```

(Continued)

Figure 5.0-2 Link Directives for Production Runs.

```
  INSERT T07402,T07404
OVERLAY F08,*R0
  INSERT EL08
  INSERT T08101,T08102,T08103,T08104,T08105
  INSERT T08107,T08108,T08109,T08201
  INSERT T08202,T08203,T08204,T08205
OVERLAY F09,*R0
  INSERT EL09
  INSERT T09100,T09201,T09202,T09300
  INSERT T09400,T09501,T09505,T09506
OVERLAY F11,*R0
  INSERT EL11
  INSERT T11101,T11102,T11201,T11202,T11203
  INSERT T11204,T11301,T11302,T11303,T11401
  INSERT T11402,T11403,T11501,T11502,T11503
OVERLAY F12,*R0
  INSERT EL12
  INSERT T12101,T12102,T12103,T12104,T12105
  INSERT T12106,T12107,T12201,T12202,T12204
  INSERT T12205,T12206,T12207,T12301,T12302
  INSERT T12303,T12304
OVERLAY F13,*R0
  INSERT EL13
  INSERT T13100,T13102,T13104,T13105,T13201
  INSERT T13202,T13203,T13301,T13302
OVERLAY F15,*R0
  INSERT EL15
  INSERT T15101,T15102,T15201,T15202,T15203
  INSERT T15204,T15205,T15206
OVERLAY F16,*R0
  INSERT EL16
  INSERT T16101,T16102,T16103,T16201,T16202,T16203
  INSERT T16204,T16205,T16206,T16207,T16208,T16209
  INSERT T16210
OVERLAY F17,*R0
  INSERT EL17
  INSERT T17100,T17200,T17300,IOPTR
OVERLAY F18,*R0
  INSERT CANAC,T04402
  INSERT ORPAC
  INSERT TCANAC
  INSERT UDPOF,RESEQ,CANEVT1
```

(Continued)

Figure 5.0-2 Link Directives for Production Runs.

```
    OVERLAY F19,*R0
      INSERT DATAFIL
    OVERLAY F20,*R0
      INSERT PFIND2,PFIND,METSEQ,VELCAL,R999
    OVERLAY F201,F20
      INSERT PFIND1,FACL,RACL,GRWCRS
    OVERLAY F202,F20
      INSERT SLOCH,RESCH,ADJUST,UPTPO,BSTSLN,RACIT
    OVERLAY F21,*R0
     INSERT MOVAC,SCALER
REGION R4,F21
  OVERLAY F211,*R4
    INSERT PHOLD,ACMS1,ACJBC2,CANEVT,ACT4
  OVERLAY F212,*R4
    INSERT CONDET,MINVEC,MAXVEC,APRCHT
  OVERLAY F213,*R4
    INSERT CRASH
ENDREG R4
REGION R5,R4
  OVERLAY F2131,*R5
    INSERT INTVEL,ACNE,GETXYZ,PUTXYZ,ACERR,SETETA,STOREV
    INSERT SPDWRD,SPDPOS,SPEEB
  OVERLAY F2132,*R5
    INSERT BNDRY
  OVERLAY F21321,F2132
    INSERT INJUR,INSIDE,CONVEX,JURFND
  OVERLAY F21322,F2132
    INSERT PTCRSI,LINE,PTTOLI,LINSOL
  OVERLAY F21323,F2132
    INSERT TURN,TPQPTR
 ENDREG R5
    OVERLAY F22,*R0
      INSERT T07401,T07403,TOLFX5,TOLRES
      INSERT TPQPTR,INTVEL,FLYTM,SCALER,SETETA
      INSERT CANEVT,SPDWRD,SPEEB,STOREV,TURN
 ENDREG R0
END
```

6.0 TERMINAL GENERATION PROGRAM

This section presents a FORTRAN source listing of the Terminal Generation Program. A description of the use of this program can be found in Book I of this volume.

Figure 6.0-1 Terminal Generation Program.

```
      PROGRAM DSKWRT(INPUT,OUTPUT,TAPE6=OUTPUT,TAPE30,
     1      TAPE2,TAPE3)
C
C THIS PROGRAM IS USED TO WRITE THE OFF-LINE RANDOM, INDEXED DISK
C FILE USED IN METSEQ TO READ IN TERMINAL INFORMATION.
C
C
C COMMONS:
C
      COMMON/T1/NORWY,IPRWY(20),NOFF,IPFF(20),IPESC,TA,TR,DGATE,
     1      SRVEL(25),SRLNG,NOSR,SRXYZ(3,20),NOER,IPER(10),NOFRPT,
     2      IEFFX,ERVEL(25),NOEPTS,EDIST,ELNTH,ERXYZ(3,20),
     3      INIX(300),NREC
C
      DATA FTNM/1.645788468E-04/,NREC/1/
C
C
C FORMATS:
C
 901  FORMAT(I2)
 902  FORMAT(3F10.5)
 903  FORMAT(F10.5)
 904  FORMAT(F10.5,I2)
 905  FORMAT(3F12.5)
 908  FORMAT(A4)
 910  FORMAT(10H TERMINAL ,A4,11H POINTER = ,I10)
 911  FORMAT(33H ERROR READING FILE - TERMINATING)
 912  FORMAT(22H1FIRST RECORD WRITTEN=,I5)
 1    REWIND 2
      REWIND 3
      IESW = 0
C
      CALL OPENMS(30,INIX,300,0)
C READ IN NUMBER OF TERMINALS AND SET UP OUTSIDE DO-LOOP.
C
 60   WRITE(6,912) NREC
      READ(2,901)NOTRM
      IF(EOF,2) 700,100
 100  DO 600 NN=1,NOTRM
C
C READ IN TERMINAL MNEMONIC AND NO. OF RUNWAYS, SET UP INNER DO LOOP.
C
```

Figure 6.0-1 Terminal Generation Program.
(Continued)

```
      READ(2,908)ATERM
      IF(EOF,2) 700,101
 101  READ(2,901)NORWY
      IF(EOF,2) 700,110
 110  DO 500 N=1,NORWY
C
C ESCRT MAY BE CALLED HERE. IESW IS ERROR SWITCH.
C
      CALL ESCRT(IESW)
      IF(IESW.EQ.1) GO TO 700
C
C READ IN NO. OF FEEDER FIXES FOR RUNWAY ANS SET UP INNER-INNER DO LOOP.
C
      READ(2,901)NOFF
      IF(EOF,2) 700,115
 115  DO 200 I=1,NOFF
      READ(2,902) TA,TR,DGATE
      IF(EOF,2) 700,120
 120  TA = TA/60.
      TR = TR/60.
      READ(2,903)(SRVEL(J),J=1,25)
      IF(EOF,2) 700,130
 130  READ(2,904)SRLNG,NOSR
      IF(EOF,2) 700,140
 140  READ(2,905)((SRXYZ(J,K),J=1,3),K=1,NOSR)
      IF(EOF,2) 700,150
 150  DO 160 K=1,NOSR
 160  SRXYZ(3,K)=SRXYZ(3,K) * FTNM
      SRLNG=0.
      L = NOSR -1
      DO 170 K = 2, L
      J = K - 1
      SRLNG=SRLNG+SQRT((SRXYZ(1,K)-SRXYZ(1,J))**2 +
     + (SRXYZ(2,K)-SRXYZ(2,J))**2 + (SRXYZ(3,K)-SRXYZ(3,J))**2)
 170  CONTINUE
      CALL WRITMS(30,IPESC,91,NREC)
      IPFF(I)=NREC
      NREC=NREC+1
 200  CONTINUE
C
C SEQUENCE ROUTE RECORD WRITTEN. USE IPFF AND WRITE FEEDER FIX RECORDS.
C
```

```
      K=I + 1
      DO 220 J=K,20
 220  IPFF(J) =0.
      CALL WRITMS(30,NOFF,21,NREC)
      IPRWY(N) = NREC
      NREC = NREC + 1
C
C USE IPRWY AND WRITE TERMINAL RECORD
C
 500  CONTINUE
      CALL WRITMS(30,NORWY,21,NREC)
C
C DISPLAY TERMINAL MNEMONIC AND POINTER
C
      WRITE(6,910)ATERM,NREC
      NREC = NREC + 1
 600  CONTINUE
      CALL CLOSEMS(30)
C
C ALL TERMINALS COMPLETED. END JOB.
C
 800  STOP
C
C ERROR OCCURRED IN READING FILE.  WRITE MESSAGE AND TERMINATE.
C
 700  WRITE(6,911)
      STOP
      END
      SUBROUTINE ESCRT(IESW)
C
C COMMONS:
C
      COMMON/T1/NORWY,IPRWY(20),NOFF,IPFF(20),IPESC,TA,TR,DGATE,
     1    SRVEL(25),SPLNG,NOSR,SRXYZ(3,20),NOER,IPEP(10),NOFRPT,
     2    IEFFX,ERVEL(25),NOEPTS,EDIST,ELNTH,ERXYZ(3,20),
     3    INIX(300),NREC
C
      DATA FINM/1.645783468E-04/
C
C
C FORMATS:
C
```

Figure 6.0-1 Terminal Generation Program.

(Continued)

```
 901   FORMAT(I2)
 903   FORMAT(F10.5,24(/,F10.5))
 905   FORMAT(3F12.5)
 906   FORMAT(2I2)
C
 907   FORMAT(I2,2F10.5)
 908   FORMAT(11H0***NOERPT=,I5, 8H .GT. 10)
C THIS ROUTINE WRITES RANDOM, INDEXED DISK FILE FOR MISSED APPROACH
C INFORMATION.
C
C
C READ NUMBER OF ESCAPE ROUTES AND SET UP OUTER DO-LOOP.
C
       READ(3,901)NOER
       IF(EOF,3) 700,200
 200   DO 300 KK=1,NOER
       READ(3,906)NOERPT,IEFFX
       IF(EOF,3) 700,210
 210   IF (NOERPT.LT.11)  GO TO 220
       WRITE(6,908) NOERPT
       GO TO 700
 220   CONTINUE
 230   READ(3,903)(ERVEL(J),J=1,25)
       IF(EOF,3) 700,240
 240   READ(3,907)NOEPTS,EDIST,ELNTH
       IF(EOF,3) 700,250
 250   READ(3,905)((ERXYZ(J,K),J=1,3),K=1,NOEPTS)
       IF(EOF,3) 700,260
 260   DO 270 K=1,NOEPTS
 270   ERXYZ(3,K) = ERXYZ(3,K) * FTNM
       ELNTH = 0.
       L = NOEPTS - 1
       DO 280 K = 2, L
       J = K-1
       ELNTH=ELNTH+SQRT((ERXYZ(1,K)-ERXYZ(1,J))**2 +
     + (ERXYZ(2,K)-ERXYZ(2,J))**2 + (ERXYZ(3,K)-ERXYZ(3,J))**2)
 280   CONTINUE
       CALL WRITMS(30,NOERPT,30,NREC)
       IPER(KK) = NREC
       NREC = NREC + 1
 300   CONTINUE
C
```

Figure 6.0-1 Terminal Generation Program.

(Continued)

```
C ESCAPE ROUTE RECORD IS WRITTEN. WRITE ESCAPE RECORD.
C
      K=NOER + 1
      DO 320 J=K,10
 320  IPER(J) = 0.
      CALL WRITMS(30,NOER,11,NREC)
      IPESC = NREC
      NREC = NREC + 1
      RETURN
C
C ERROR OCCURRED.
C
 700  IESW = 1
      RETURN
      END
```


(Continued)

Figure 6.0-1 Terminal Generation Program.

7.0 INTERARRIVAL TIMES GENERATION PROGRAM

This section presents a FORTRAN source listing of the Interarrival Times Generation Program. A description of the use of this program can be found in Book I of this volume.

LIST, SDATA

```
      PROGRAM SDATA(VALUES,OUTPUT,TAPE5=VALUES,TAPE6=OUTPUT)
      NAMELIST/VALUES/NUMX,XMU,RNDPRM,XMAX
    1 READ (5,VALUES)
      IF (NUMX .EQ. 0) GO TO 100
      QZ=RANF(RNDPRM)
      RNDPRM=0.
      WRITE (6,90) XMU
      SUM=0.
      DO 10 I=1,NUMX
      R=RANF(0.)
      IF (R.EQ.0.) GO TO 10
      X=-1.*XMU*ALOG(R)
       IF(X.GT.XMAX) X=XMU
      SUM=SUM+X
      WRITE (6,95) X,SUM
   10 CONTINUE
      GO TO 1
   90 FORMAT (6HMEAN =,2X,F7.6)
   95 FORMAT (F9.6,2X,F8.6)
  100 STOP
      END
```

Figure 7.0-1 Interarrival Times Generation Program.

8.0 POST PROCESSOR PROGRAM

8.1 Description Of Log Tape

In ELIST, each time a "resource-using" task is completed, the following record is written on the Post Processor, log tape:

> Time, Task Number, Aircraft Id (or spaces), Jurisdiction Id, Phase of Flight (or spaces), Resource Element Number, Resource Pool Type.

In TO3000, each time an aircraft is added, the following record is written:

> Time, 2999, Aircraft Id, Jurisdiction Id, Phase of Flight, A/C INP.

In MØVAC, when an aircraft is removed from an enroute hold, the following record is written:

> Time, 4999, Aircraft Id, Jurisdiction Id, Phase of Flight, Length of Hold.

In MØVAC, when an aircraft is removed from a Metering and Sequencing Hold, the following record is written:

> Time, 5999, Aircraft Id, Jurisdiction Id, Phase of Flight, Length of Hold.

In ACJBC2, each time an aircraft crosses a jurisdiction boundary causing a handoff, the following record is written:

> Time, 6999, Aircraft Id, New Jurisdiction Id, Phase of Flight, Old Jurisdiction Id.

In UDPØF, each time an aircraft changes its phase of flight, the following record is written:

> Time, 7999, Aircraft Id, Jurisdiction Id, New Phase of Flight, and on landings, the difference, ENDURANCE - TIME.

In CANAC, each time an aircraft is dropped, the following record is written:

Time, 8999, Aircraft Id, Jurisdiction Id, Phase of Flight, DLD, Source of Call Indicator, set as follows:

2 - BNDRY (Flew out of system)
4 - ACT4 (Reached last point in flight plan)
38 - T04402 (Cancelled Flight Plan)
168 - UDPOF (Aircraft Landed, usually superceded by ACT4)

If the program ends with aircraft still in the system, the following information is printed for each aircraft remaining:

Time, 9999, Aircraft Id, Jurisdiction Id, Phase of Flight, DATAFIL.

8.2 Source Listing

This section presents a FORTRAN source listing of the DELTA Post Processor Program. A description of the use of this program can be found in Book I of this volume.

```
              PROGRAM POSTPRC (INPUT, OUTPUT, TAPE5=INPUT, TAPE6=OUTPUT,
             +          TAPE7, TAPE8)
          C
          C        THIS PROGRAM PROCESSES THE SORTED OUTPUT OF THE DELTA MODEL.
          C             TAPE6 IS THE OUTPUT FILE.
          C             TAPE7 IS THE INPUT FILE SORTED ON LCC, JUR.
          C             TAPE8 IS THE INPUT FILE SORTED ON AC, LCC.
          C
000004        DIMENSION PHASE(3, 8)
          C
000004        DATA PHASE/10HPRE-FLIGHT,10H.          ,1H ,
000004       1          10HTAKE OFF. ,10H            ,1H ,
000004       2          10HDEPARTURE ,10HTRANSITION,1H.,
000004       3          10HEN ROUTE. ,10H            ,1H ,
000004       4          10HARRIVAL TR,10HANSITION. ,1H ,
000004       5          10HAPPROACH. ,10H            ,1H ,
000004       6          10HLANDING.  ,10H            ,1H ,
000004       7          10HMISSED APP,10HROACH.     ,1H /
          C
000004        DATA IBLANK/4H    /, NLCC, NJUR, NAC/3*1/,
000004       1        TLAND1, TLANDL, NLAND/1.E30, -1.E30, 0/
          C
000004        EQUIVALENCE (BLANK, IBLANK)
          C
          C        HEAD OUTPUT PAGE
000004        WRITE (6, 9671)
          C
          C        FIRST READING OF TAPE7
000010        REWIND 7
          C
000012        READ (7, 9072)  LCCOLD, JUROLD
000022        LCC = LCCOLD
          C
000024    100 IF (LCC .GE. 999)  GO TO 500
          C
          C        SUBSEQUENT READING OF TAPE7
000027        READ (7, 9072) LCC, JUR
```

```
        C
000037        IF (EOF, 7)  105, 110
        C
        C          SET LCC TO 999
000042    105 LCC = 999
000043        JUR = IBLANK
        C
        C
        C          SAME JURISDICTION?
000045    110 IF ((JUR .EQ. JUROLD) .AND. (LCC .EQ. LCCOLD))  GO TO 115
        C
        C          NEW JURISDICTION AND/OR NEW LCC
000055        WRITE (6, 9672)  LCCOLD, JUROLD, NJUR
000067        JUROLD = JUR
000071        NJUR = 0
        C
        C          INCREMENT JURISDICTION COUNTER
000072    115 NJUR = NJUR + 1
        C
        C          SAME LCC?
000074        IF (LCC .NE. LCCOLD)  GO TO 125
        C
        C          INCREMENT LCC COUNTER
000076        NLCC = NLCC + 1
000100        GO TO 100
        C
        C          NEW LCC
000101    125 WRITE (6, 9673) LCCOLD, NLCC
000111        LCCOLD = LCC
000113        NLCC = 1
000114        GO TO 100
        C
        C          NO FURTHER LCC DATA OF INTEREST ON THIS TAPE,
        C              PROCESS TAPE8
000115    500 REWIND 8
        C
        C          HEAD OUTPUT PAGE
000117        WRITE (6, 9681)
```

```
         C
         C
         C           FIRST READING OF TAPE8
000123     502 READ (8, 9083) TIME, LCCOLD, ACOLD, JUROLD, IPOF, MOREDAT
000143         IF (EOF, 8) 9000, 504
         C
000146     504 IF (ACOLD .EQ. BLANK)  GO TO 502
         C
         C
000150         NLCC = 1
         C
         C           SUBSEQUENT READING OF TAPE8
000151     510 READ (8, 9083)  TIME, LCC, AC, JUR, IPOF, MOREDAT
         C
000171         IF (EOF, 8)  750, 515
         C
         C           SAME AC>
000174     515 IF (AC .NE. ACOLD)  GO TO 580
         C
         C           SAME AC.
000176     520 IF (LCC .LT. 999)  GO TO 550
         C
         C           DETAIL INFORMATION
000201     530 IF (LCCOLD .LT. 999)  WRITE (6, 9683) ACOLD, LCCOLD, NLCC
000215         LCCOLD = LCC
000217         NLCC = 1
000220         KEY = LCC / 1000
000223         IF (KEY .GT. 0)
000223       +     GO TO (610, 620, 630, 640, 650, 660, 670, 680, 690), KEY
         C
         C
         C           BAD KEY VALUE
000241         GO TO 700
         C
         C           TALLY LCC
         C           SAME LCC>
000242     550 IF (LCC .NE. LCCOLD)  GO TO 575
         C
```

```
         C          SAME LCC.
000244          NLCC = NLCC + 1
000246          GO TO 510
         C
         C          NEW LCC
000247      575 WRITE (6, 9683)  ACOLD, LCCOLD, NLCC
000261          LCCOLD = LCC
000263          NLCC = 1
000264          GO TO 510
         C
         C          NEW AC.
000265      580 IF (LCCOLD .LT. 999)  WRITE (6, 9683) ACOLD, LCCOLD, NLCC
000301          LCCOLD = LCC
000303          NLCC = 1
000304          ACOLD = AC
000306          NAC = NAC + 1
000310          IF (LCC .LT. 999)  GO TO 510
000313          GO TO 530
         C
         C          SPECIAL EVENT DATA FOUND.
         C
         C          KEY .EQ. 1, NOT USED.
000314      610 GO TO 700
         C
         C          KEY .EQ. 2, AIRCRAFT ENTERING THE SYSTEM (T03000)
000315      620 WRITE (6,9862)  AC, TIME, JUR
000327          GO TO 700
         C
         C          KEY .EQ. 3, NOT USED.
000330      630 GO TO 700
         C
         C          KEY .EQ. 4, AIRCRAFT COMPLETING AN EN ROUTE HOLD (MOVAC)
000331      640 WRITE (6,9864) AC, TIME, JUR, MOREDAT
000345          GO TO 700
         C
         C          KEY .EQ. 5, AIRCRAFT COMPLETING A M ^ S HOLD (MOVAC)
000346      650 WRITE (6,9865) AC, TIME, JUR, MOREDAT
000362          GO TO 700
```

```
             C
             C         KEY .EQ. 6, AIRCRAFT CHANGING JURISDICTIONS (ACJBC2)
000363        660 WRITE (6,9866) AC, TIME, JUR
000375            GO TO 700
             C
             C         KEY .EQ. 7, AIRCRAFT CHANGING PHASE OF FLIGHT (UDPOF)
000376        670 IF (IPOF .GT. 0) WRITE (6, 9867) AC, TIME,
000376          +          PHASE(1,IPOF),PHASE(2,IPOF),PHASE(3,IPOF)
000420            IF (IPOF .NE. -1)  GO TO 700
             C
             C         TALLY LANDINGS
000422            IF (TIME .LT. TLAND1)  TLAND1 = TIME
000426            IF (TIME .GT. TLANDL)  TLANDL = TIME
000432            NLAND = NLAND +1
000434            GO TO 700
             C
             C         KEY .EQ. 8, AIRCRAFT CANCELLED (CANAC)
000435        680 WRITE (6, 9868) AC, TIME, JUR
000447            GO TO 700
             C
             C         KEY .EQ. 9, AIRCRAFT LEFT IN SYSTEM AT EOJ (DATAFIL)
000450        690 WRITE (6, 9869) AC, JUR
             C
000460        700 CONTINUE
000460            GO TO 510
             C
             C         END OF FILE ON TAPE8
000461        750 IF (LCCOLD .LT. 999)  WRITE (6, 9683)  ACOLD, LCCOLD, NLCC
             C
000475            TIME = 0.
000476            IF (TLANDL .GT. TLAND1)  TIME = FLOAT (NLAND) / (TLANDL - TLAND1)
000504            WRITE (6, 9685) NLAND, TIME, NAC
             C
             C
000516       9000 REWIND 7
000520            REWIND 8
             C
000522            STOP 0002
```

```
          C
          C          FORMATS
000524     9072 FORMAT (16X, I4, 4X, A4)
000524     9083 FORMAT (1X, F15.10, I4,2A4, I2, A8)
000524     9671 FORMAT (1H1, 15X, 17HJURISDICTION LIST,//,
000524         +    25H  TASK JURISDICTION COUNT,/)
000524     9672 FORMAT (I6, 4X, A4, 6X, I5)
000524     9673 FORMAT (/,22H TOTAL COUNT FOR TASK , I4, 4H IS , I5, 1H.,//)
000524     9681 FORMAT (1H1, 18X, 13HAIRCRAFT LIST, //,
000524         +    21H  AIRCRAFT TASK COUNT, /)
000524     9683 FORMAT (4X, A4, I7, I6)
000524     9685 FORMAT (//, I6, 11H LANDINGS, ,F8.3, 15H LANDINGS/HOUR., //,
000524         +    I10, 26H TOTAL AIRCRAFT PROCESSED.,//)
000524     9862 FORMAT (22X, 9HAIRCRAFT , A4, 11H ENTERED AT , F15.10,
000524         +    4H IN , A4, 1H.)
000524     9864 FORMAT (22X, 9HAIRCRAFT , A4, 4H AT , F15.10, 4H IN , A4,
000524         +    24H ENDED EN ROUTE HOLD OF , A8, 7H HOURS.)
000524     9865 FORMAT (22X, 9HAIRCRAFT , A4, 4H AT , F15.10, 4H IN , A4,
000524         +    23H ENDED M AND S HOLD OF , A8, 7H HOURS.)
000524     9866 FORMAT (22X, 9HAIRCRAFT , A4, 15H HANDED OFF AT , F15.10,
000524         +    4H TO , A4, 1H.)
000524     9867 FORMAT (22X, 9HAIRCRAFT , A4, 18H CHANGED PHASE AT , F15.10,
000524         +    4H TO , 2A10, A1)
000524     9868 FORMAT (22X, 9HAIRCRAFT , A4, 14H CANCELLED AT , F15.10,
000524         +    4H IN , A4, 1H.)
000524     9869 FORMAT (22X, 24HAT END OF RUN, AIRCRAFT , A4, 4H IN , A4, 1H.)
          C
000524           END
```

RUNX COMPILER (VER.26) 12/13/73. 10.47.03. POSTPRO

PROGRAM LENGTH
001021

STATEMENT FUNCTION REFERENCES

LOCATION	GEN TAG	SYM TAG	REFERENCES

STATEMENT NUMBER REFERENCES

LOCATION	GEN TAG	SYM TAG	REFERENCES					
000024	L00014	100	000100	000114				
000042	L00023	105	000041					
000045	L00025	110	000041					
000072	L00037	115	000054					
000101	L00044	125	000075					
000115	L00052	500	000025	000026				
000123	L00057	502	000147					
000146	L00064	504	000145					
000151	L00067	510	000246	000264	000312	000460		
000174	L00074	515	000173					
000176	L00076	520	NONE					
000201	L00100	530	000313					
000242	L00115	550	000200					
000247	L00121	575	000243					
000265	L00127	580	000175					
000314	L00143	610	000230					
000315	L00144	620	000231					
000330	L00150	630	000232					
000331	L00151	640	000233					
000346	L00155	650	000234					
000363	L00161	660	000235					
000376	L00165	670	000236					
000435	L00204	680	000237					
000450	L00210	690	000240					
000460	L00213	700	000241	000314	000327	000330	000345	000362
			000375	000421	000434	000447		
000461	L00214	750	000173					
000516	L00230	9000	000145					

000544	C00016	3072	000012	000027		
000547	C00021	9033	000123	000151		
000553	C00025	9671	000004			
000566	C00040	9672	000055			
000572	C00044	9673	000101			
000601	C00053	9681	000117			
000614	C00066	9683	000203	000247	000267	000463
000617	C00071	9685	000504			
000633	C00105	9862	000315			
000645	C00117	9864	000331			
000661	C00133	9865	000346			
000675	C00147	9866	000363			
000707	C00161	9867	000377			
000721	C00173	9868	000435			
000733	C00205	9869	000450			

BLOCK NAMES AND LENGTHS
FIOBUF$- 004114

VARIABLE REFERENCES

LOCATION	GEN TAG	SYM TAG	REFERENCES					
001017	V00022	AC	000160	000174	000304	000320	000334	000351
			000366	000402	000440	000453		
001014	V00016	ACOLD	000132	000146	000174	000206	000252	000272
			000305	000466				
000750	V00021	BLANK	000146					
000750	V00002	IBLANK	000043					
001015	V00017	IPOF	000136	000164	000376	000406	000411	000414
			000420					
001012	V00014	JUR	000034	000044	000050	000067	000162	000324
			000340	000355	000372	000444	000455	
001010	V00012	JUROLD	000017	000051	000062	000070	000134	
001020	V00023	KEY	000222	000223				
001011	V00013	LCC	000023	000024	000032	000042	000045	000074
			000111	000156	000176	000215	000220	000242
			000261	000301	000310			
001007	V00011	LCCOLD	000015	000022	000045	000060	000074	000104

```
RUNX COMPILER (VER.26)                  12/13/73. 10.47.03.        POSTPRC

                                      000112   000130   000201   000210   000216   000242
                                      000254   000262   000265   000274   000302   000461
                                      000470
       001016    V00020    MOREDAT    000140   000166   000342   000357
       001003    V00005    NAC        000306   000513
       001002    V00004    NJUR       000064   000071   000072
       001006    V00010    NLAND      000432   000502   000507
       001001    V00003    NLCC       000076   000106   000113   000150   000212   000217
                                      000244   000256   000263   000276   000303   000472
       000751    A00001    PHASE      NONE
       001013    V00015    TIME       000126   000154   000322   000336   000353   000370
                                      000404   000422   000424   000426   000430   000442
                                      000475   000503   000511
       001005    V00007    TLANDL     000426   000431   000476   000500
       001004    V00006    TLAND1     000422   000425   000476   000501

START OF CONSTANTS
000526

START OF TEMPORARIES
000742

START OF INDIRECTS
000750

COMPILER SPACE
UNUSED -  010000    USED   -  040000
```

APPENDIX REPORT OF INVENTIONS

A diligent review of the work performed under this contract, has revealed no new innovation, discovery, improvement, or invention.