AFAPL-TR-68-142
PART XVII

PROPULSION SYSTEM FLOW STABILITY PROGRAM (DYNAMIC)

PHASE I FINAL TECHNICAL REPORT

PART XVII—PROPULSION SYSTEM SIMULATION DIGITAL COMPUTER PROGRAM FORMAT AND ROUTINES

E.H. Kaplan and H.W. Wong
LOS ANGELES DIVISION OF NORTH AMERICAN ROCKWELL CORPORATION

TECHNICAL REPORT AFAPL-TR-68-142, PART XVII
F33615-67-C-1848

December 1968

Air Force Aero Propulsion Laboratory
Air Force Systems Command
Wright-Patterson Air Force Base, Ohio

AFAPL-TR-68-142
PART XVII

PROPULSION SYSTEM FLOW STABILITY PROGRAM (DYNAMIC)

PHASE I FINAL TECHNICAL REPORT

PART XVII. PROPULSION SYSTEM SIMULATION DIGITAL COMPUTER PROGRAM FORMAT AND ROUTINES

E.H. Kaplan and H.W. Wong

FOREWORD

This report describes work accomplished in Phase I of the two-phase program, "Propulsion System Flow Stability Program (Dynamic)" conducted under USAF Contract F33615-67-C-1848. The work was accomplished in the period from 20 June 1967 to 30 September 1968 by the Los Angeles Division of North American Rockwell Corporation, the prime Contractor, and the Subcontractors, the Allison Division of General Motors Corporation (supported by Northern Research and Engineering Corporation), the Autonetics Division of North American Rockwell Corporation (supported by the Aeronautical Division of Honeywell, Incorporated), and the Pratt & Whitney Aircraft Division of United Aircraft Corporation.

The program was sponsored by the Air Force Aero Propulsion Laboratory, Wright-Patterson Air Force Base, Ohio. Mr. H. J. Gratz, APTA, Turbine Engine Division, was the Project Engineer.

This volume is Part XVII of twenty parts and was prepared by the Los Angeles Division of North American Rockwell Corporation.

Publication of this report does not constitute Air Force approval of the report's findings or conclusions. It is published only for the exchange and stimulation of ideas.

Ernest C. Simpson
Chief, Turbine Engine Division

ABSTRACT

The primary objective of Task 7 of the "Propulsion System Flow Stability Program" was to develop a simulation program to be used in Phase II for the evaluation of two control systems capable of sensing and accommodating a transient condition.

Since the work on this task was being performed by three companies, every effort was made to insure compatibility in terminology, units, and program documentation as well as to provide means of communicating the myriad details involved in making computer runs of the system. This documentation format is described in Section II of this volume.

An early element of this task was the selection of a simulation language for use in programming the simulation. The choice of IBM's DSL/90 and the factors involved in making that choice are discussed in Section III.

Simulation programs have a natural tendency to be rather voluminous and, when the system being simulated is as complex as a supersonic inlet, turbofan, and an integrated control system can be, computer storage space is rapidly filled. To alleviate this crowding, numerous logic blocks which were repetitive, such as compressor logic, were removed from the simulation logic deck and made into subroutines or functions. These subprograms are discussed in Section IV.

Once the simulation logic is written, the most difficult task of all begins. The job of initialization is usually not given proper emphasis until many hours of work have convinced all concerned that it is really the most important phase. Section V discusses this task and shows an example of an initialization routine.

(The reverse of this page is blank)

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

(The reverse of this page is blank)

LIST OF TABLES

(The reverse of this page is blank)

Section I

INTRODUCTION

To accomplish the task of writing a propulsion system simulation, three groups, each knowledgeable in one or more of the three technical areas involved, induction system, engine, and control, were brought together. Initial efforts were on an individual basis with each group programming their portion of the system in their own terminology. At an early stage in the coordination of the effort, it was obvious that to avoid chaos a system of terminology, in commonly understood terms, would be required. Also, that the detailed information as to just what simulation logic was being used at any time must be recorded in a standard fashion so that all groups would know exactly what was being simulated. To accomplish this, the system described in this volume was developed. The system, although primarily developed before much simulation programming was done, continued to evolve as needs for additional capabilities arose.

Exceeding computer storage space is always a danger in a complex problem such as a propulsion system simulation. To forestall, if not prevent, this occurrence any blocks of logic which are general in nature have been removed from the simulation and placed in subprograms. This saves space in two ways. First, the variables calculated internally to the subprogram do not count against the DSL/90 limits on the number of variables. Secondly, the logic is stored only once and is used as many times as is needed. These subprograms are described within this volume with program listings presented in Appendix I.

The last step before a simulation can occur, and normally the step least thought about, is initialization. The steady state operating point must be established before the transient being simulated is introduced. The procedure followed in the program developed under this task is shown by an example discussed in this volume.

(The reverse of this page is blank)

Section II

DOCUMENTATION FORMAT

GENERAL OBJECTIVE

The system of documentation described in this section was developed to allow the three participating companies to first, have a common terminology during the development of the propulsion system simulation program and second, to have a method of recording, or documenting, each simulation run. Toward this goal, a naming convention for program parameters, a series of forms to record pertinent system information, and a format for simulation logic diagrams were developed.

VARIABLE NAMES

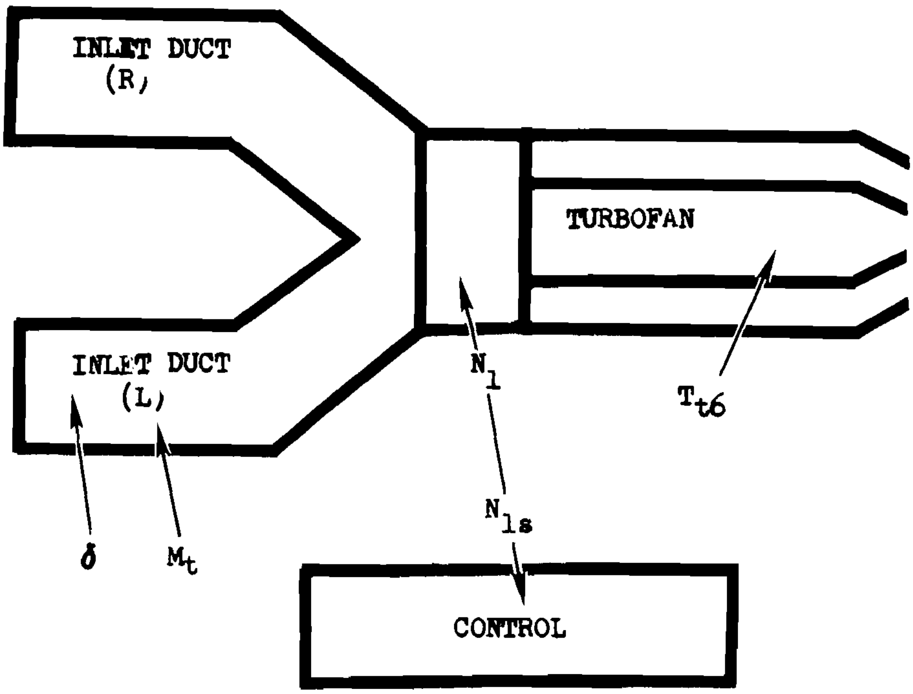
When naming variables, two opposing methods are open to the programmer. He may use a name similar to the engineering name, severly abbreviated by restrictions on length, six characters, and available symbols, no greek alphabet, no lower case letters, no sub or superscripts, and no non-alphanumeric symbols (i.e. $/$, $\sqrt{\ }$, etc). Discouraged by his inability to express more complex engineering terms in a meaningful form, the programmer can then choose the opposite extreme and just number all parameters and have a key list to identify the meaning. This is a most flexible scheme, but causes the loss of all immediate visibility to the program. Parameter 6109 does not mean much until you have memorized several hundred names or looked up its meaning. For these reasons, a compromise system hopefully combining the best features of both was adopted for the propulsion system simulation. This system is described as follows.

Each variable name is composed of six characters. The first three, and in the case of control system variable names the first four, must follow the naming convention. The remaining characters are assignable at the option of the programmer with one exception. If the name describes a table, the letter T shall appear in one of the optional character locations.

The first two characters in each parameter name must be a prefix from a standard list, shown in table I. The next character is a number which designates the subsystem within which the parameter is generated. In the case of the control system, this is carried one level further by having the third character show the control system designation and the fourth character the subsystem affected. A typical subsystem numbering scheme for a propulsion system with a twin-duct inlet, a turbofan engine, and an integrated propulsion system control is shown in figure 1 with a schematic of the propulsion system and an example of several parameters and their engineering names.

TABLE I. NAME PREFIX LIST

Prefix	Description	Units
AØ	Area	In^2
CN	Input constant	
EC	External command	
ET	Efficiency	
FØ	Thrust	Lbs
GM	Ratio of specific heats	
HD	Enthalpy difference	BTU/Lb
MN	Mach number	
NØ	Rotor speed	RPM
NR	Rotor speed ratio	
PØ	Pressure	PSI
PR	Pressure ratio	
QA	General variable originated by Autonetics	
QL	General variable originated by LAD	
QP	General variable originated by P&WA	
RE	Reynolds number	
SA	Subroutines originated by Autonetics	
SL	Subroutines originated by LAD	
SP	Subroutines originated by P&WA	
TØ	Temperature	^oR
TR	Temperature ratio	
UØ	Velocity	Ft/Sec
VØ	Volume	Ft^3
WA	Air flow	Lb/Sec
WF	Fuel flow	Lb/Sec
WG	Gas flow	Lb/Sec
WQ	Weight Quantity	Lbs
XØ	Position	In



PROPULSION SYSTEM SCHEMATIC

SUBSYSTEM NUMBERS	
EXTERNAL	000-999
INLET DUCT (L)	1000-1999
INLET DUCT (R)	2000-2999
TURBOFAN	3000-3999
CONTROL	4000-4999
Inlet (L)	4100-4199
Inlet (R)	4200-4299
Turbofan	4300-4399
Internal	4400-4499

Figure 1. Propulsion System Numbering Scheme

In order to fully define the parameter, a keying list or "dictionary" is required. Figure 2 shows a sample dictionary based on the same system described above. In this dictionary, the engineering name can be looked up to get the program name and the definition. The greek letter name is handled by spelling out the letter and offsetting the name on the tab card. This causes all greek letter names to sort out separately and in a psuedo-alphabetic order. The dictionary is also sorted by the program name to allow easy cross-reference.

FORMS

To record the information required to describe and, if necessary, duplicate a simulation run, the following series of forms were developed.

RUN SUMMARY SHEET

The basic form for the system is the run summary sheet. This form provides the information on what was run, how it was run and what happened to the run. Copies of this form are distributed to each participant and attached to the computer printout. Figures 3 and 4 illustrate the run summary sheet and its use, also the continuation sheet that may be used as needed. The information in the heading block is self explanatory until the space for set up base is encountered. The set up base states the specific deck set up used which is described on a form identified by the number in this space. The number of the tape containing the DSL/90 system program is entered, if used, in the DSL/90 tape space.

The series of boxes referring to bases are used to identify the component being simulated and the specific simulation logic, associated subprograms, tables, and output being used. The form, as shown, provides for five phases of inlet operation for left and right inlets. The phases, as used, are ST (started), UN (unstarted), EF (empty-fill), SB (subcritical), and HS (hammershock). The form then provides five columns for engine components of which the example, using a single turbofan engine, only uses one. The control system is identified by the final column. If separate inlet and engine controls were used, the engine control logic would either be included in the engine logic or be identified by one of the engine component boxes.

The input data, other than tables, used for a particular run is recorded in the Input Data columns. Space is provided to record the subsystem requiring the data, which is redundant when the name convention described above is used, the program name, the value, and the engineering name (variable). The notes column should give the purpose of the run, and, after the run, the results of the run. The disposition of the output should be stated in this column.

The continuation page for the run summary sheet is shown in figure 4.

ALPHABETIC LISTING BY ENGINEERING NAME

DELTA	RAMP ANGLE	QL1002
MT	THROAT MACH NUMBER	MN1005
N1	FAN ROTOR SPEED	NO3001
N1S	SENSED VALUE OF FAN ROTOR SPEED	NO8301
TABLE	COMPRESSOR MAP TABLE	WA3T01
TT6	HIGH TURBINE EXIT TEMPERATURE	TO3006

ALPHABETIC LISTING BY PROGRAM NAME

MT	THROAT MACH NUMBER	MN1005
N1	FAN ROTOR SPEED	NO3001
N1S	SENSED VALUE OF FAN ROTOR SPEED	NO8301
DELTA	RAMP ANGLE	QL1002
TT6	HIGH TURBINE EXIT TEMPERATURE	TO3006
TABLE	COMPRESSOR MAP TABLE	WA3T01

Figure 2. Sample Dictionary

RUN SUMMARY SHEET

RUN NO. 1414-01 DATE 2/27/68 PAGE 1 OF 1

TITLE CHECKOUT OF STARTED PHASE

ORIGINATOR WONG

SETUP BASE

DSL/90 TAPE M461

	ESTIMATED	ACTUAL
SIMULATION	2"	2"
MACHINE	2'	1'20"

SUBSYSTEM / BASE	INLET										ENGINE COMPONENT					CONTROL SYSTEM
	ST		UN		EF		SB		HS							
	L	R	L	R	L	R	L	R	L	R	TF					
SIMULATION LOGIC	/										O					O
USER SUPPLIED ROUTINES																
TABLE	/										O					O
OUTPUT	*															

* DEBUG PRINTING USED

INPUT DATA				NOTES
SUBSYSTEM	NAME	VALUE	VARIABLE	
INLET	CN1001	.53174852	λ	Turbofan and control system - dummy logic used.
	CN1100	1.0	BASE	
	CN1004	1.0	K_{AI}	
	CN1002	0.96	K_U	
	CN1003	1.0	K_{BP}	
	CN1006	10000.	K_{HS}	
	CN1042	0.4	K_{DZ}	
	CN1067	0.2	K_{YZ}	
	AΦ1008	840.0	A_c	
	XΦ1001	36.0	X_L	
	XΦ1002	196.0	X_2	
	XΦ1003	64.0	X_T	
	XΦ1021	36.0	X_I	
	XΦ1022	64.0	X_{II}	
	XΦ1023	86.0	X_{III}	

Figure 3. Run Summary Sheet

RUN SUMMARY SHEET (Continued)

RUN NO.____________________ DATE____________________ PAGE _____ OF _____

INPUT DATA				NOTES
SUBSYSTEM	NAME	VALUE	VARIABLE	

Figure 4. Run Summary Sheet

SIMULATION LOGIC BASE SHEET

This sheet lists the diagrams which define the simulation logic used for a particular subsystem. The sheet, illustrated by figure 5, contains the name of each logic block, its diagram number, and the dash number or version of the diagram.

TABLE BASE SHEET

The tables, usually tabular representation of curves, used for a simulation run are listed on this sheet, as shown in figure 6. The information is similar to that provided in the previously described form.

OUTPUT BASE SHEET

DSL/90 provides for two methods of output of data. One is a procedure by which any of the program variables can be printed at a specified print time increment by listing the names to be printed on the PRINT control card. The maximum and minimum values of any parameter can also be obtained by listing the name of that parameter on the RANGE control card. Plotted data is also available on IBM 1627 equipment using the original IBM DSL/90 system and on SC-4020 equipment using the North American Rockwell Corporation (NR) modified DSL/90 system. Provision for other equipment must be provided by the user.

The list or lists of variables desired to be printed or plotted are recorded on the output base sheet shown in figure 7.

EXECUTION CONTROL BASE SHEET

Information dealing with the actual run parameters, such as the integration method, time increment, value for run termination, and the tolerance specifications, is recorded on this form. One note of caution concerning the information on this sheet is that the time increments for printing, and plotting, specified above, override the time increment for execution of fixed step integration methods if these times are smaller.

An example of an execution control base is shown in figure 8.

SET-UP BASE SHEET

The physical deck arrangement is pictured by this form. An example of one such arrangement is shown in figure 9.

SIMULATION LOGIC BASE SHEET

DESCRIPTION	
Turbofan Inlet	SUBSYSTEM AISL
	PHASE ST
	BASE NO. 1
	DATE 1/3/68

LOGIC BLOCK	DIAGRAM	DASH NO.
Input Data	1100	01
Input Tables	1101	01
Upstream Properties	1110	01
Properties At The Terminal Shock Station	1120	01
Properties Behind The Normal Shock	1130	01
Duct Volume and Mach Number	1150	01
Subsonic Flow Total Pressure Losses	1151	01
Duct Properties	1152	01
Duct Continuity & Energy	1153	02
Properties At Station Z	1169	01
Helmholtz Volume Position	1170	02
Helmholtz Volume Properties	1171	01
Zone III Bleed Upstream Of Shock	1180	01
Zone III Bleed Downstream Of Shock	1181	01
Bypass And Engine Systems Airflow	1182	01
Phase Switches	1189	01
Throat Mach Number	1190	01

Figure 5. Simulation Logic Base Sheet - Turbofan Inlet

TABLE BASE SHEET

DESCRIPTION	
Turbofan Inlet	SUBSYSTEM AISL PHASE ST BASE NO. 1 DATE 5/3/68

NAME	TABLE DASH NO.	VARIABLE	REMARKS
PR1T50	01	P_{tx}/P_{to}	
AØ1T00	01	duct area	
VØ1T00	01	duct volume	
MN1T31	01	M_a	
QL1T44	01	ε	
QL1T45	01	ϕ_x	
QL1T46	01	ϕ_y	
WA1T32	01	W_{II}/W_o	

Figure 6. Table Base Sheet - Turbofan Inlet

OUTPUT BASE SHEET

DESCRIPTION:	PHASE ______
	BASE NO. ______
	DATE ______

PRINT TIME INCREMENT ________ SECONDS

NAMES TO BE PRINTED:

NAMES FOR WHICH MAXIMA AND MINIMA ARE TO BE PRINTED:

PLOT TIME INCREMENT ____________ SECONDS

NAMES TO BE PLOTTED:

IND. NAME	NAME 1	NAME 2	NAME 3	DESCRIPTION

Figure 7. Output Base Sheet

EXECUTION CONTROL BASE SHEET

DESCRIPTION: CHECK OUT	PHASE ____ BASE NO. 1 DATE 2/27/68

INTEGRATION METHOD

- [X] MILNE - MILNE VARIABLE TIME INCREMENT*
- [ ] RKS - RUNGE-KUTTA VARIABLE TIME INCREMENT*
- [ ] RKSFX - RUNGE-KUTTA FIXED TIME INCREMENT**
- [ ] SIMP - SIMPSON'S RULE**
- [ ] TRAPZ - TRAPEZOIDAL**
- [ ] RECT - RECTANGULAR**
- [ ] CENTRL - CENTRAL USER SUPPLIED**

* DELMIN __________ MINIMUM TIME INCREMENT (SECONDS)

** DELT __________ FIXED TIME INCREMENT (SECONDS)

NAME VALUE FOR RUN TERMINATION

NAME	VALUE	NAME	VALUE	NAME	VALUE	NAME	VALUE	NAME	VALUE

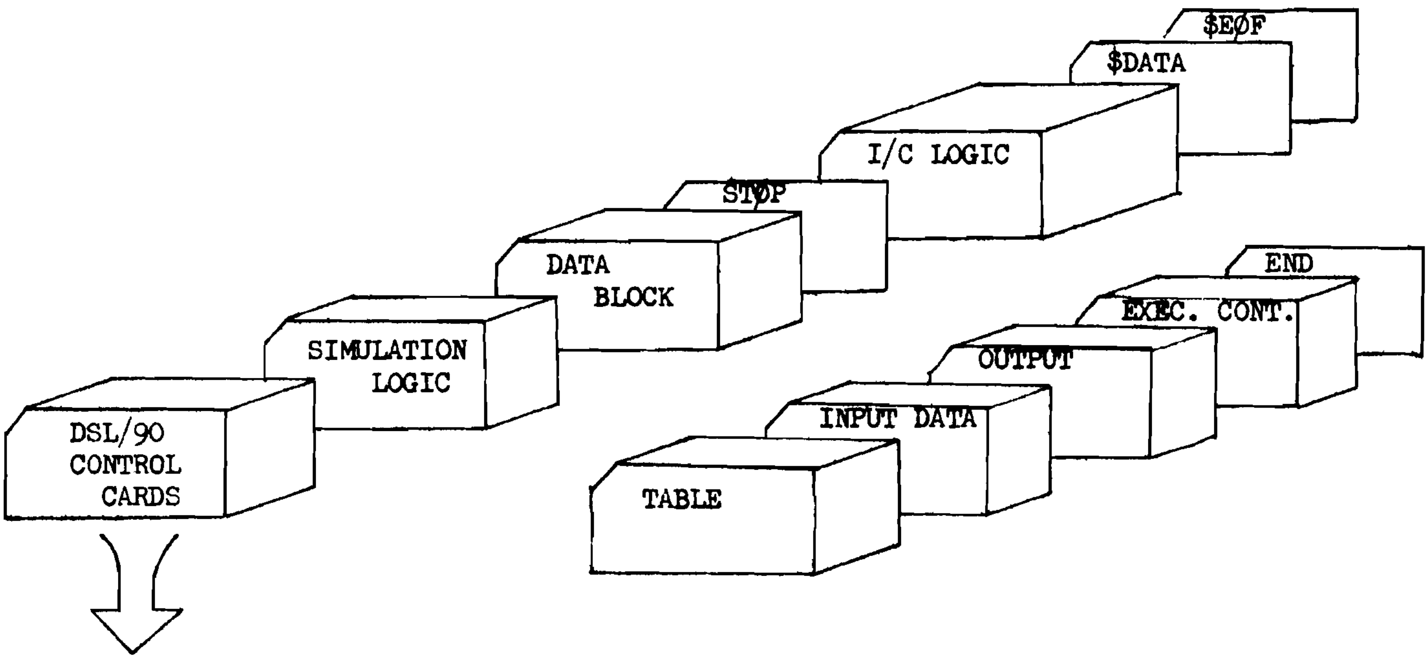
TOLERANCES

NAME	REL. ERROR (MILNE OR RKS)	ABS. ERROR RKS	NAME	REL. ERROR (MILNE OR RKS)	ABS. ERROR (RKS)	NAME	REL. ERROR (MILNE OR RKS)	ABS ERROR (RKS)

Figure 8. Execution Control Base Sheet

SET-UP BASE SHEET

DESCRIPTION:	BASE NO.______ DATE________




```
$IBJOB TRNSYM                                                DSL/9003
$IEDIT         SYSCK1,SRCH                                   DSL/9004
$IBLDR CKSTOR                                                DSL/9005
$IBLDR CONTIN                                                DSL/9006
$IBLDR FINISH                                                DSL/9007
$IBLDR INTEG                                                 DSL/9008
$IBLDR JIGSAW                                                DSL/9009
$IBLDR NAME                                                  DSL/9010
$IBLDR OUTIN                                                 DSL/9011
$IBLDR RDWRMX                                                DSL/9012
$IBLDR SCAN                                                  DSL/9013
$IBLDR STORE                                                 DSL/9014
$IBLDR TRANSL                                                DSL/9015
$IBLDR XMSG1                                                 DSL/9016
$ORIGIN        ALPHA                                         DSL/9017
$IBLDR SORT                                                  DSL/9018
$ORIGIN        ALPHA                                         DSL/9019
$IBLDR OUTPUT                                                DSL/9020
$IEDIT                                                       DSL/9021
$DATA                                                        DSL/9022
$IEDIT         SYSCK1,SRCH                                   DSL/9023
$IBLDR MAIN                                                  DSL/9024
$IBLDR CENTRL                                                DSL/9025
$IEDIT                                                       DSL/9026
```

Figure 9. Set-Up Base Sheet

DIAGRAMS

The logic for simulating each component of the propulsion system must be committed to paper in such a way that it is not only available to be coded into logic statements for the computer but also so that it may be understood by people who are not computer oriented. There are almost as many diagramming conventions as there are people so a standard diagram procedure was established for this project and is described below. Each subsystem will have one or more of each of the diagrams described.

INPUT DATA DIAGRAM

The first diagram in each subsystem lists the input data required by that subsystem to perform its calculations. As shown by the example in figure 10, the computer name, the engineering name, the description, and the units are specified.

INPUT TABLES DIAGRAM

The second diagram in the subsystem set lists the tables required by the subsystem simulation logic. The table name, description, and units are given, as well as the logic diagram in which the table is used. The example in figure 11 illustrates this diagram.

SIMULATION LOGIC DIAGRAM

The simulation logic is diagrammed according to the following procedure. An example is shown in figure 12.

Inputs to a diagram enter on the left using a dashed box with the program name of the parameter inside. The source of the input parameter is denoted by a subsystem name above the arrow to the left of the input box. If the input is from another diagram within the same subsystem the diagram number should be noted under the arrow, otherwise no entry is placed under the arrow. The engineering name appears to the right of the input box, as it does on all boxes, above the arrow showing the path of the logic. When an input box is a table name, no engineering name is used.

The numbering convention for diagrams is similar to that for parameter names in that the numbers follow the convention used for the third character of the program name. A list of these numbers with abbreviations for the subsystems of the previously used example in figure 1 are given in figure 13. Also shown are the general rules for the above described input boxes and several examples. The formats for logic boxes within the diagram are shown on figure 14. In example B, the function described is a routine which, when given the flow parameter, computes Mach number. Since there

NAME	VARIABLE	DESCRIPTION	UNITS
CN1001	λ	Sonic Flow Constant	—
CN1002	K_u	Inlet Throat Sonic Flow Coefficient	—
CN1003	K_{bp}	P_{tbp}/P_{t2}	—
CN1004	K_A	A_d/A_{dgeo}	—
CN1006	K_{HS}	Hammershock Indicator Constant	—
CN1100	Base	I/C Base No.	—
CN1042	K_{dz}	Duct Total Pressure Loss Constant Between Stations d and z	—
CN1067	K_{yz}	Helmholtz Volume Total Pressure Loss Constant	—
XØ1001	X_L	Cowl Lip Station	in.
XØ1002	X_2	Engine Face Station	in.
XØ1003	X_T	Throat Station	in.
XØ1021	X_I	Station I	in.
XØ1022	X_{II}	Station II	in.
XØ1023	X_{III}	Station III	in.
QL1101	ℓ	Helmholtz Volume Length	in.
AØ1008	A_c	Capture Area	$in.^2$

DIAGRAM 1100-01

STARTED PHASE INPUT DATA

Figure 10. Input Data List

TABLE NAME	DESCRIPTION	OUTPUT UNITS	DIAGRAM WHERE USED
AØ1TOO	Duct area versus station, throat area	$in.^2$	1120 1169
MN1T31	M_A versus M_o, α_o, ψ_o	--	1110
PR1T50	P_{tx}/P_{to} versus M_o, α_o, ψ_o	--	1110
QL1T44	ϵ versus M_A, throat area	--	1151
QL1T45	ϕ_x versus M_A, throat area	--	1180
QL1T46	ϕ_y versus M_A, throat area	--	1181
VØ1TOO	Duct volume versus station, throat area	$ft.^3$	1150
WA1T32	W_{II}/W_o versus M_o, α_o, ψ_o	--	1110

DIAGRAM 1101-01

STARTED PHASE INPUT TABLES

Figure 11. Input Data List

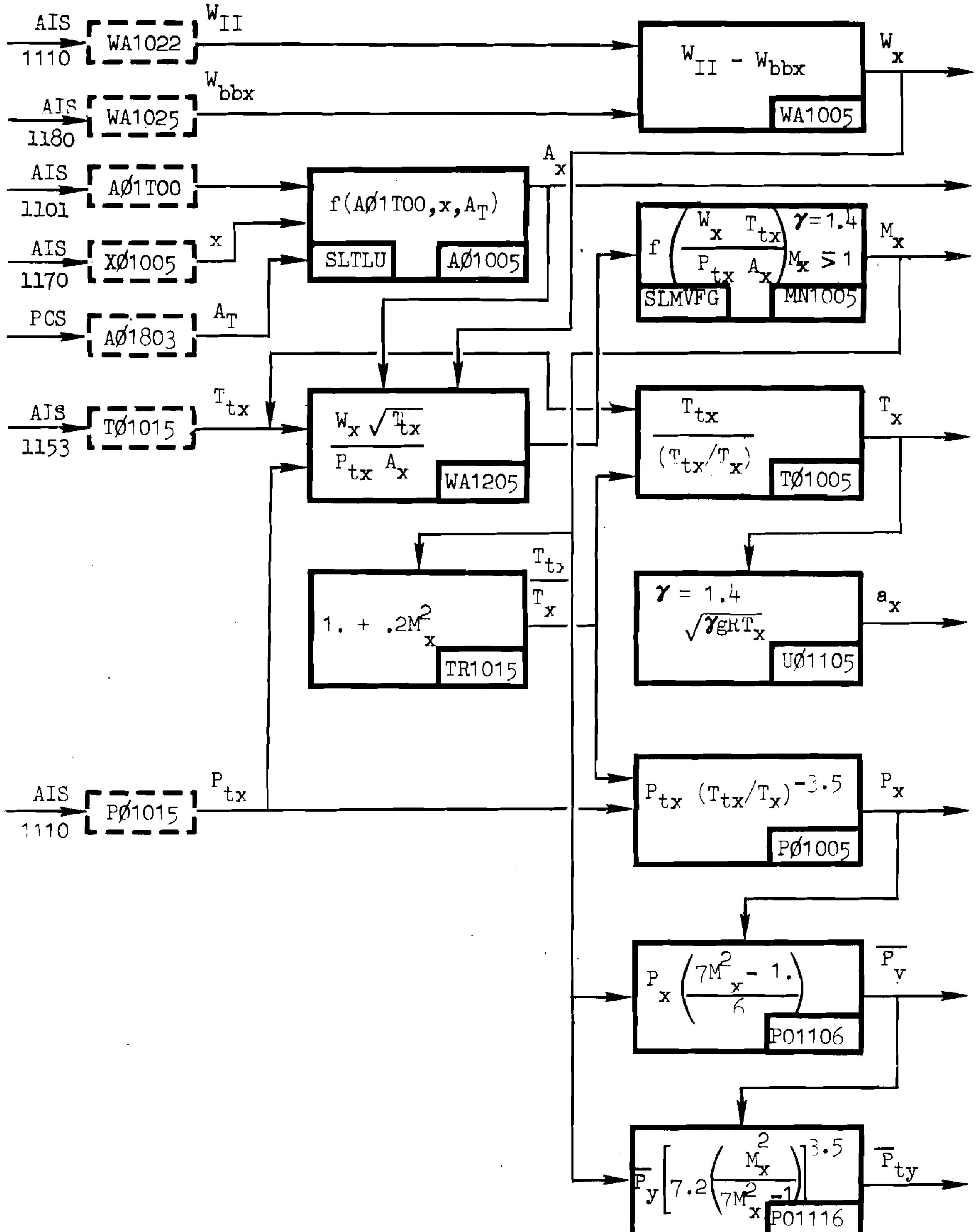


DIAGRAM 1120-01

STARTED PHASE PROPERTIES AT TERMINAL SHOCK STATION
Figure 12. Simulation Logic Diagram

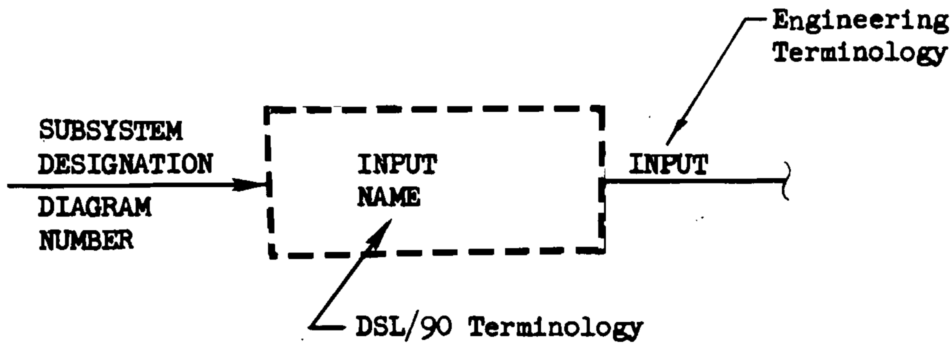
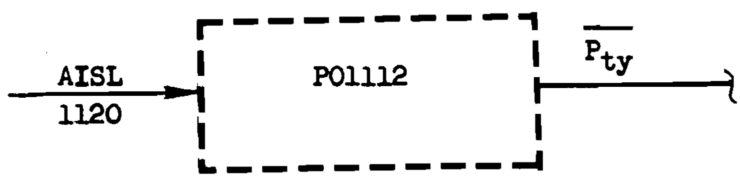


DIAGRAM TERMINOLOGY

Designation	Subsystem	Diagram Numbers
AISL	Inlet duct, left	1---
AISR	Inlet duct, right	2---
TFAN	Turbofan engine	3---
PCS	Propulsion control system	4---

Example A:

(Input from diagram in same subsystem.)



Example B:

(Input from another subsystem.)

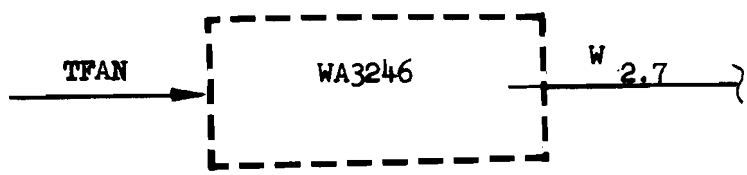


Figure 13. Diagram Input Box Format

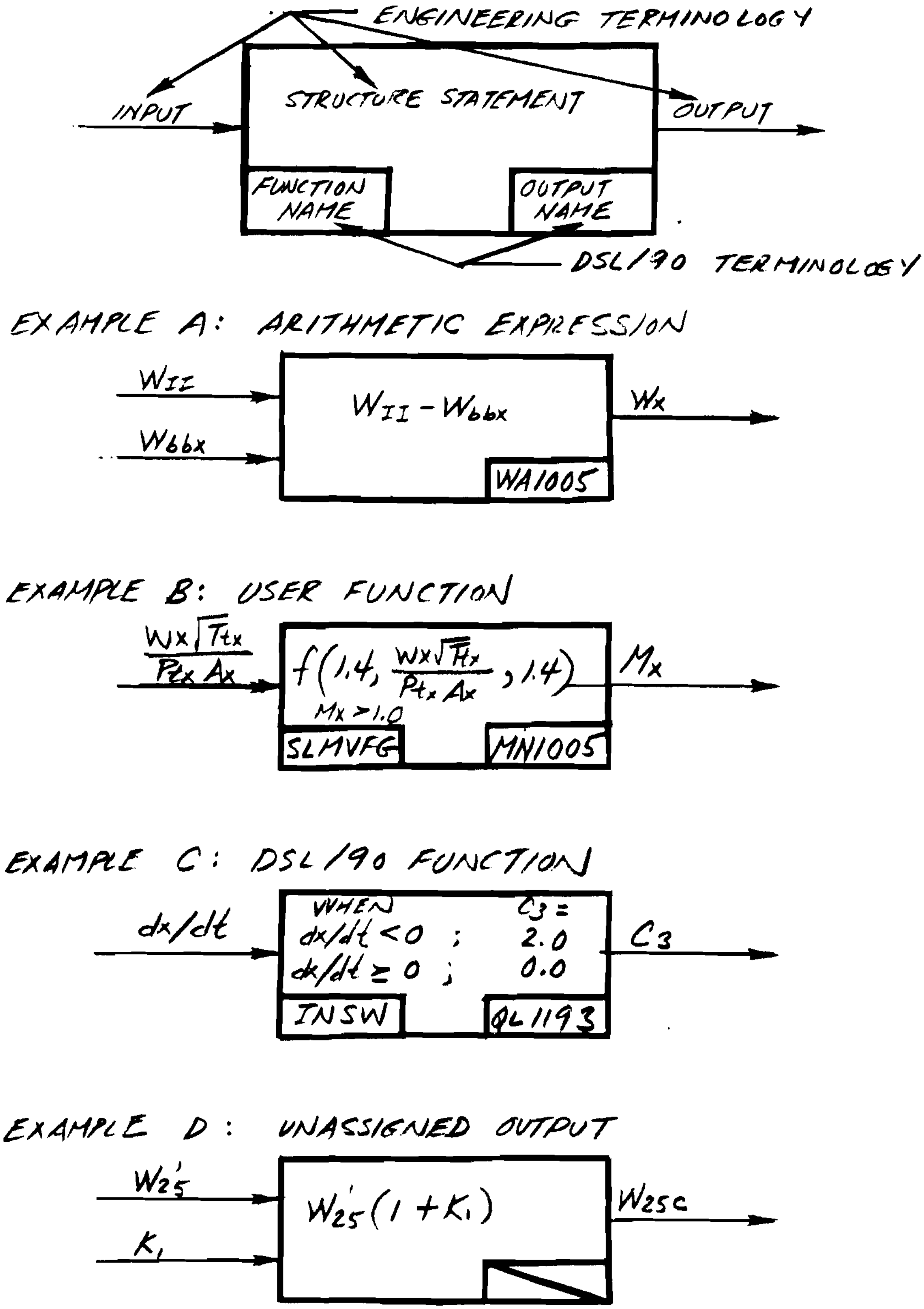


Figure 14. Box Format for Diagrams

are two answers possible, the note $M_x \geqslant 1.0$ is used to show which solution is desired. Similarly, in example C the function INSW is described in mathematical terms so that visibility to the engineer is enhanced. In example D, a box is shown in which a computation is made and no program name is given. This means that the computation is made internally to the next box upstream on the logic path and the result of that unnamed box is not available as an output.

Section III

SELECTION OF SIMULATION LANGUAGE

COMPARISON

A study was made of features of both the MIMIC and DSL/90 simulation languages. The results of that study are tabulated in table II.

An existing simulation program utilizing the General Electric Company Dynasyar language was converted to both DSL/90 and MIMIC simulation languages. Both ran satisfactorily at the USAF Aero Propulsion Laboratory and at North American Rockwell Corporation. Comparisons of engine face total pressure versus time, inlet terminal shock position versus time, and shock velocity versus time showed near identical results.

CONCLUSION

As a result of the study, it was decided to proceed with DSL/90 as the simulation language for the propulsion system simulation.

TABLE II. COMPARISON OF DSL/90 AND MIMIC

	Language			Advantage	
	DSL/90	MIMIC	Comments	DSL/90	MIMIC
1) Configuration Description	Standard FORTRAN statements and arithmetic operators.	MIMIC statement format and standard arithmetic operators (cannot distinguish operator **).	Majority familiar with standard FORTRAN.	X	
2) Integrator a) Method	Group A - Variable interval fifth order Milne predictor-corrector, fourth order Runge-Kutta; Group B - Fixed interval fourth order Runge-Kutta, Simpson's Rule, Trapezoidal, rectangular; Group C - User-supplied centralized integration method.	Fourth order Runge-Kutta with variable interval.		X	
b) Tolerance Control	Tolerances for each individual integrator. Relative and absolute tolerances for Runge-Kutta. Only relative tolerances for Milne.	Same relative or absolute tolerance for all integrators.	Individual tolerances permit looser tolerances for variables that do not require high accuracy during integration, but are the controlling variables during parts of the transient. Result is a saving of machine time.	X	

TABLE II. COMPARISON OF DSL/90 AND MIMIC (Continued)

	Language			Advantage	
	DSL/90	MIMIC	Comments	DSL/90	MIMIC
3) Function Routine					
a) Specification	FORTRAN IV and MAP.	FORTRAN IV and MAP.			
b) Changes	Can be easily altered.	Can be easily altered.			
c) Additions	Restricted only by core storage.	Only five functions with specified names can be added.		X	
d) Loading	Only functions called for in simulation are loaded.	All functions must be loaded.	Minimize storage.	X	
4) Input Data					
a) Format	Floating point, variable field identified by variable name. Restricted by card size.	Floating point, fixed field and identified by the order of input. Restricted to six values per card.		X	
b) Constant	Any order; constant not input are assumed to be zero or previous case value.	Input complete list of constants beginning and in order.		X	
c) Parameter	Any order.	Input complete list of parameters in order. Must reinput for each subsequent case.		X	

TABLE II. COMPARISON OF DSL/90 AND MIMIC (Continued)

	Language			Advantage	
	DSL/90	MIMIC	Comments	DSL/90	MIMIC
d) Tables Book Up 2 Dimensional	Linear and La Grange interpolation. Function value equal to table limit when independent variable is cutside of tabulated range. Warning is printed out.	Linear interpolation. Function value equal to zero when independent variable is outside of tabulated range.	DSL/90 input to tables is more convenient.	X	
3 Dimensional	Not available.	Linear interpolation and independent values must fall inside tabulated range.			
e) Array	Yes.	No direct provision, but can use an alternate three dimensional table input method. Only three values per card possible.		X	
5) Output					
a) Specification	Variables by name.	Heading and variables by name.	DSL/90 offers an easier callout procedure. MIMIC offers a greater flexibility in printout.		

TABLE II. COMPARISON OF DSL/90 AND MIMIC (Continued)

	Language		Comments	Advantage	
	DSL/90	MIMIC		DSL/90	MIMIC
b) Format	Time, variable name and values printed. If nine or fewer variables are printed, names are printed only once at column heads.	Variable names are printed only at initial time.	Variable names being printed only once is desirable only if there are few enough variables so that they may be printed on one line allowing columnar tabulation.	X	
c) Special Printout	Minimum and maximum. All variables at each iteration beginning at specified time.			X	
6) Special Feature	Part or all of the program can be omitted in sorting.		Saves sorting time.	X	
	Procedural logic.		Saves sorting time.	X	
	Repeatable procedural logic.	Similar to DSL/90, but no branching logic.	Additional capability in branching reduces number of statements.	X	
	Compiled simulation deck.	Control over execution of individual MIMIC statement.	Saves machine time in sorting and compiling.	X	

TABLE II. COMPARISON OF DSL/90 AND MIMIC (Concluded)

	Language			Advantage	
	DSL/90	MIMIC	Comments	DSL/90	MIMIC
7) Run Time	Program is divided into two steps, Translate and Simulate. The two steps require more time than the single step of MIMIC. Because there is a complied simulation deck, overall time should be better than MIMIC in subsequent production runs which would not require the Translate step.	Program sorts and assembles a machine language program for each run.	No comparable time study made. Load time has been shorter for MIMIC. One short run will be better with MIMIC.		
8) Program Capacity	Can be increased with overlay. Tradeoff with tables possible.	Tradeoff with tables possible.	DSL/90 has greater flexibility.	X	

Section IV

SUPPORTING SUBPROGRAMS

GENERAL APPROACH

To minimize the creation of parameter names which were not required as printed output and to economize on computer storage space usage, numerous subprograms have been written to support the propulsion system simulation program.

A naming convention for all support subprograms was established so that subprograms could be written by each of the three participants without danger of name duplication and also to allow the routine to be readily identified as to origin. This convention has all Autonetics routines begin with SA, Pratt and Whitney with SP, and NR Los Angeles Division with SL.

All basic algorithms have been removed from the simulation logic and placed in either function or subroutine form. An example of a large block of logic thus removed is the compressor subroutine SPCOMP. This logic is used three times in the simulation of the turbofan and once in a turbojet simulation with only the particular map being used and the names of the inputs and outputs being changed. Removing this section from the simulation logic removes a large number of new variable names from the restricted number DSL/90 allows and saves the locations the duplicated logic instructions would use.

DESCRIPTIONS

The subprograms are described in alphabetical order. A short description of the purpose of the subprogram is given followed by a flow diagram and the computer compilation. In the case of several control routines, the flow diagrams are presented in several forms to provide maximum understanding of their purpose.

When the coding of the subprograms began, the decision had been made, primarily on previous experience with General Electric's DYNASYAR simulation language, to use a variable time step integration method. The method chosen was the DSL/90 MILNE integration scheme. The first checkouts of separate propulsion system components, several inlets, a turbojet engine, and a turbofan engine were successfully run using MILNE. When an integrated, although simplified, propulsion system control was added to an inlet and a turbojet some strange things began to occur. In the course of tracing these strange occurrences a liberal education in DSL/90 was obtained.

The MILNE integration scheme uses six slices of history plus the current calculation to operate its predictor-corrector. The method used for "cutting back" or reducing the time slice because the current calculations exceed tolerances is an involved one.

The problem is best described by the example shown on figure 15. This example illustrates the changes in past history made by three successive failures of the current calculation to meet tolerances. The effect on the subprograms that use any past values is to require constant testing of time on each execution pass and appropriate changes to the subprogram history. It was also discovered that DSL/90's HSTRSS (hysteresis) routine did not appear to handle this history correctly. Since considerable work appeared to be required to clear up all the problems brought on by use of the variable time step it was decided to dispense with it for the time being. It will be reconsidered after the simulations required in Phase II are full operational. In the meantime, one of the fixed time step integration methods will be used.

As a result of this decision, some of the subprogram listings presented in Appendix I have sections dealing with past history that do not show up on the diagrams and flow charts in the following figures. This added logic is being removed as time is available and the subprogram decks will ultimately agree with the diagrams presented herein. If it appears that use of the variable step iteration has advantages worth the cost of implementation, a supplementary report will be issued on the variable step versions of these programs.

SUBROUTINE SAACT

Simulation of the numerous actuators within the Propulsion Control System has been achieved by use of subroutine described below and in the accompanying figures.

The actuator simulation depicted in figure 16 is composed of the following components:

1. A limiter acting on the input or command value (XC)

2. A feedback signal which may be selected from either the output of the integrator (X) or the output of the actuator (XH) which includes hysteresis effects

3. A loop gain term (KA)

4. A limiter acting on the rate (XDOT)

5. An integrator.

HISTORY STORAGE LOCATIONS							CURRENT STORAGE	TIME STEP	TOLERANCE TEST
1	2	3	4	5	6	7	8	Δt	
			0	1/4	2/4	3/4	4/4	1/4	met
		0	1/4	2/4	3/4	4/4	5/4	1/4	met
	0	1/4	2/4	3/4	4/4	5/4	6/4	1/4	met
0	1/4	2/4	3/4	4/4	5/4	6/4	8/4	1/2	met
		0	2/4	4/4	6/4	8/4	10/4	1/2	met
	0	2/4	4/4	6/4	8/4	10/4	12/4	1/2	met
0	2/4	4/4	6/4	8/4	10/4	12/4	16/4	1	failed
0	2/4	4/4	6/4	8/4	10/4	12/4	14/4	1/2	failed
		8/4	9/4	10/4	11/4	12/4	13/4	1/4	met
	8/4	9/4	10/4	11/4	12/4	13/4	14/4	1/4	failed
		11/4	23/8	12/4	25/8	13/4	27/8	1/8	met
	11/4	23/8	12/4	25/8	13/4	27/8	14/4	1/8	met
11/4	23/8	12/4	25/8	13/4	27/8	14/4	15/4	1/4	met
		11/4	12/4	13/4	14/4	15/4	16/4	1/4	met

Figure 15. Sample of Storage Sequence for MILNE Integration

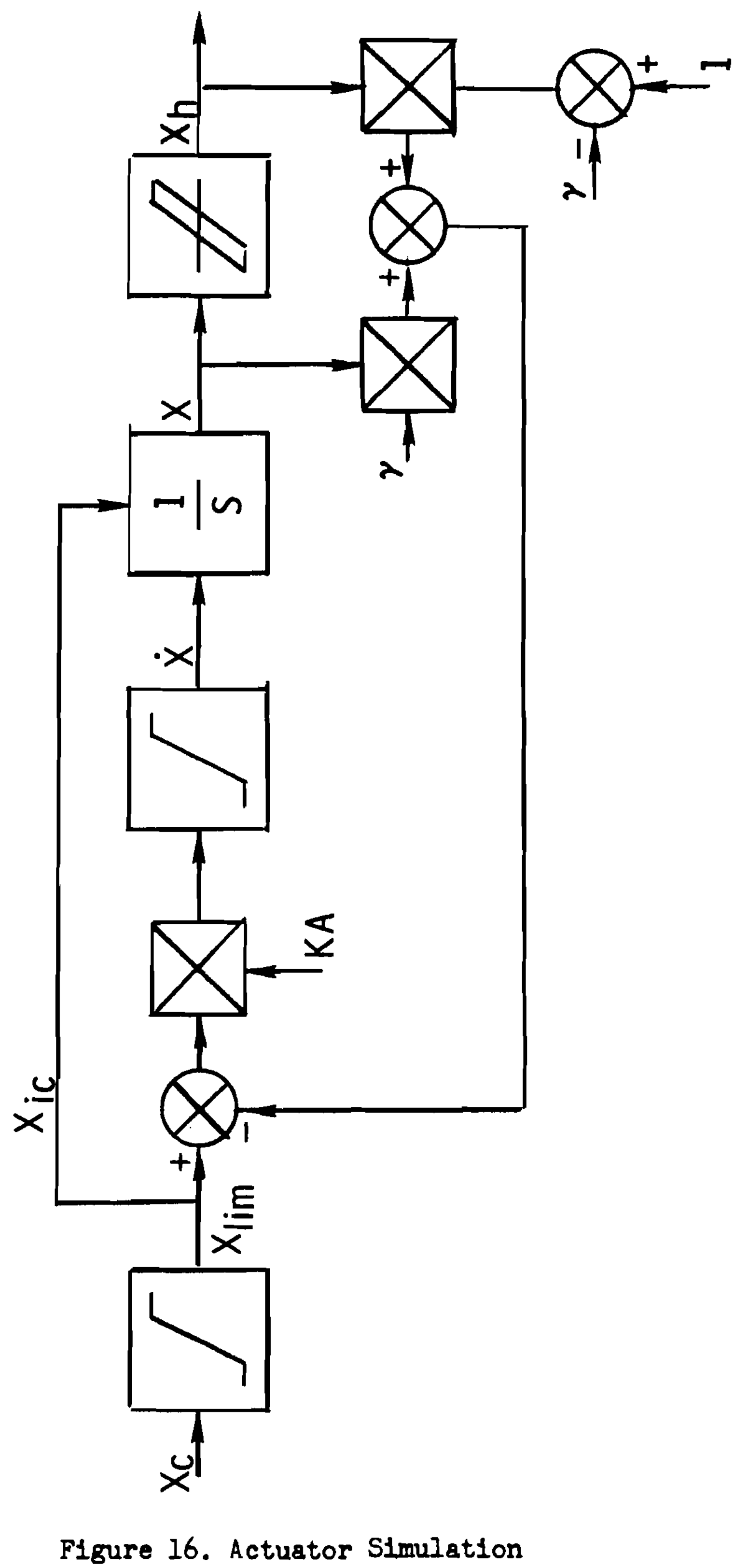


Figure 16. Actuator Simulation

The integrator requires an initial condition value (XIC) and a derivative (XDOT). A parameter, γ , is used to select the feedback signal desired. This parameter is set at either unity or zero with the former selecting feedback from the integrator output.

In order to implement this simulation in DSL/90, a revision to the simulation diagram of figure 16 was made. This revised diagram, shown in figure 17, is functionally identical to the original diagram except in arrangement. The revised diagram allows the components in the dashed box to be placed in the subroutine under discussion (SAACT).

The FORTRAN flow diagram for SAACT is shown on figure 18.

FUNCTION SADSPA

A routine to perform switching operations with provision for a "dead space" is required in the Propulsion Control System (PCS). The standard DSL/90 dead space routine (DEADSP) is limited to a linear output function with unity slope which is not suitable for the discrete function switching requirements of the PCS.

The inputs to SADSPA are three in number, the independent parameter, X, and the left and right limits of the dead space. When the value of X is below the left limit, the function value is a negative one (-1), above the right limit the function value is a positive one (+1), and between the limits the output is zero (0).

This routine has a secondary usage as a switch similar to function SASWCH without hysteresis when either the left or right limits are set above or below any possible value of the independent parameter.

The operation of this function is shown in diagrammatic form and FORTRAN flow form in figure 19.

SUBROUTINE SALIMT

When the limits of a function are computed values there exists the possibility of the minimum limit exceeding the maximum limit. In such an instance in PCS simulation it is desirable to be able to specify which limit has priority. To accomplish this the SALIMT routine was written.

By the use of the argument TYPE the routine will give priority to maximum (TYPE=1.0), minimum (TYPE=0.0) or ignore both limits (TYPE-1.0). When the maximum and minimum limits are in their normal positions the routine functions as a normal limit routine.

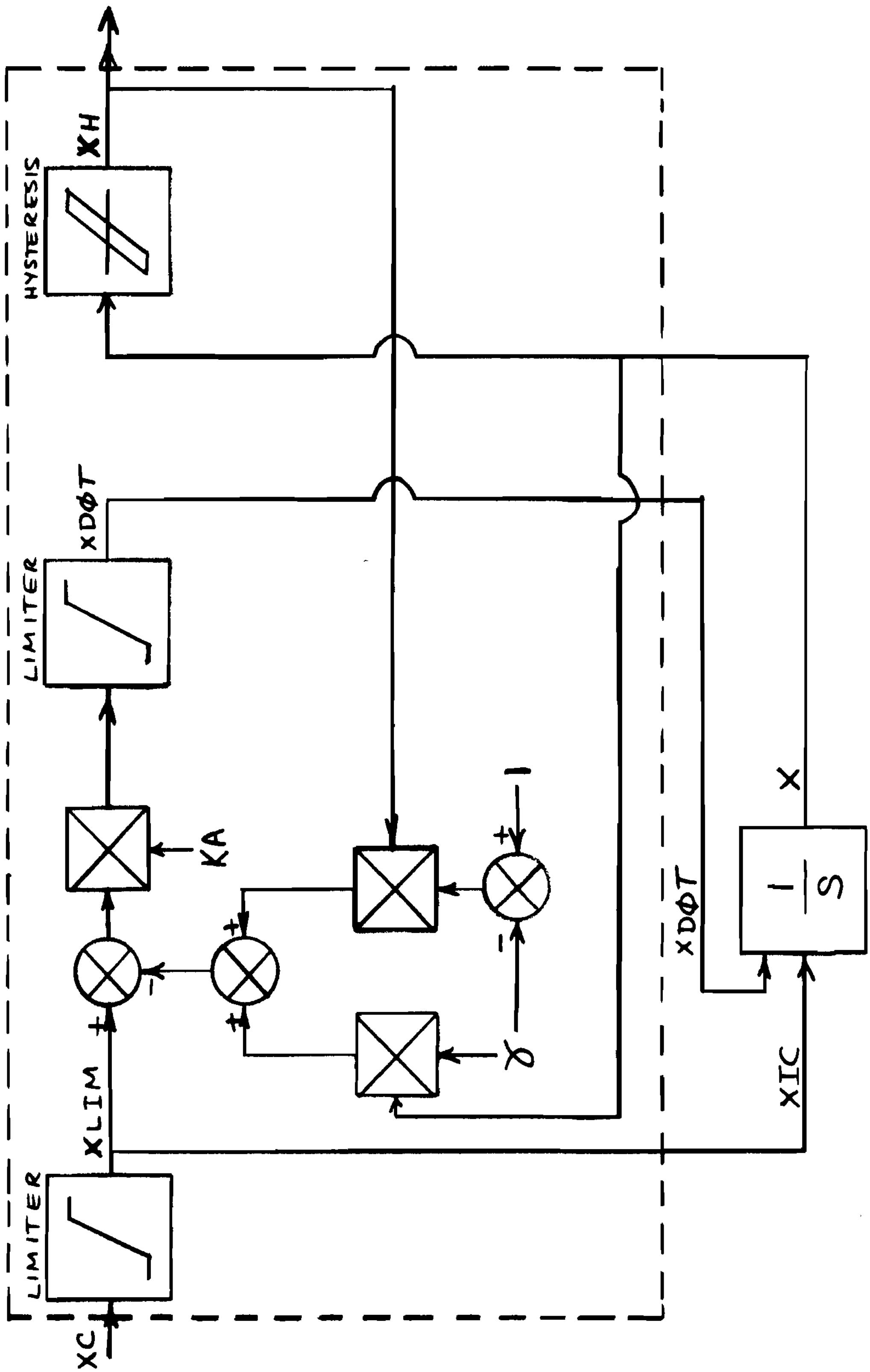


Figure 17. Revised Actuator Simulation

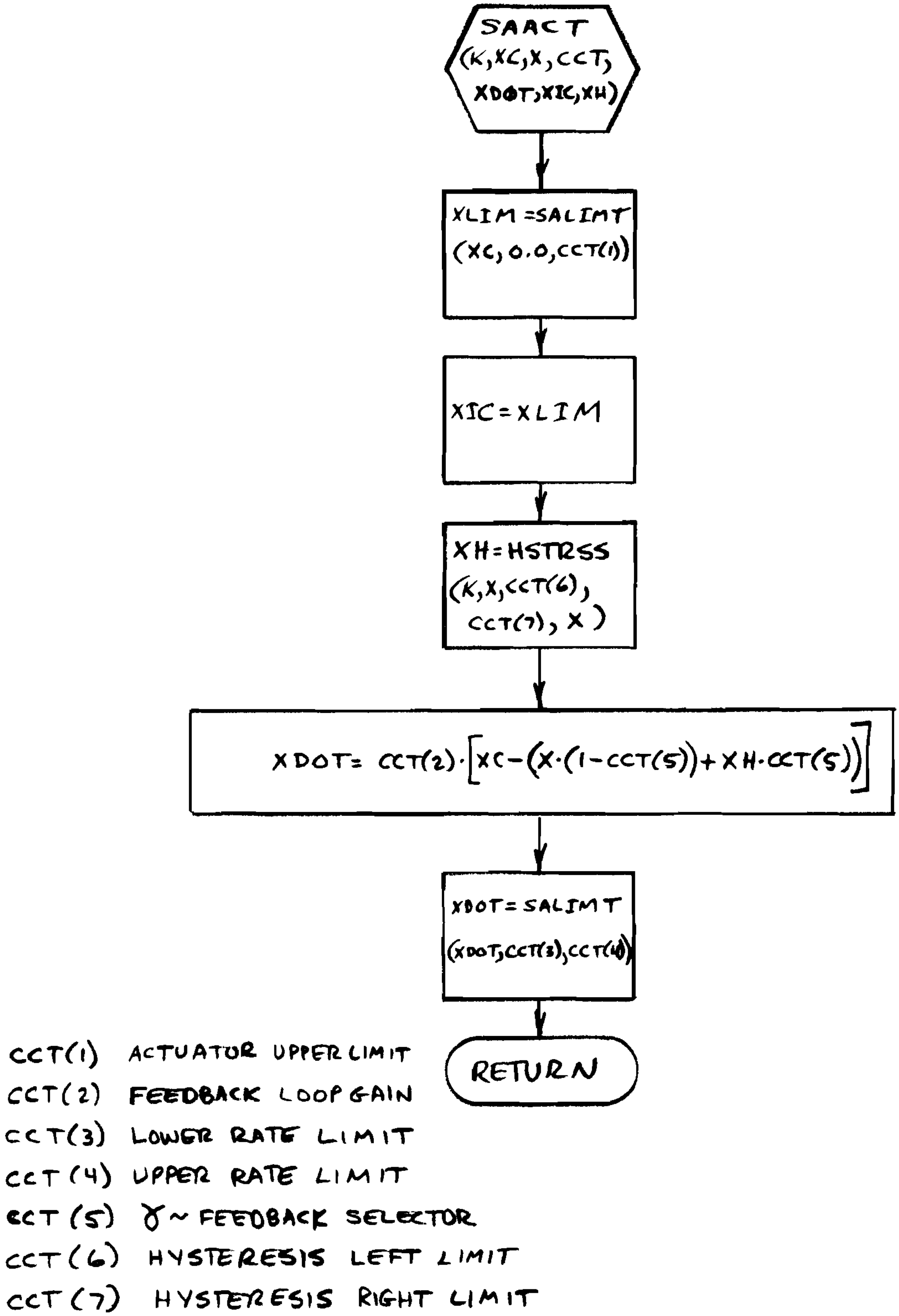
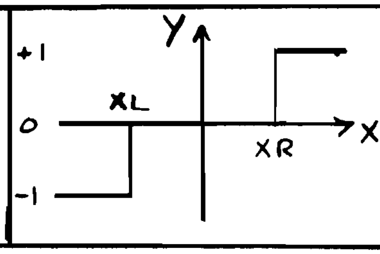


Figure 18. SAACT Subroutine Diagram

GENERAL FORM	FUNCTION		
	WHEN	THEN	
Y = SADSPA(X, XL, XR) RELAY (DISCRETE OUTPUT)	X > XR XL ≤ X ≤ XR X < XL	Y = 1 Y = 0 Y = −1	

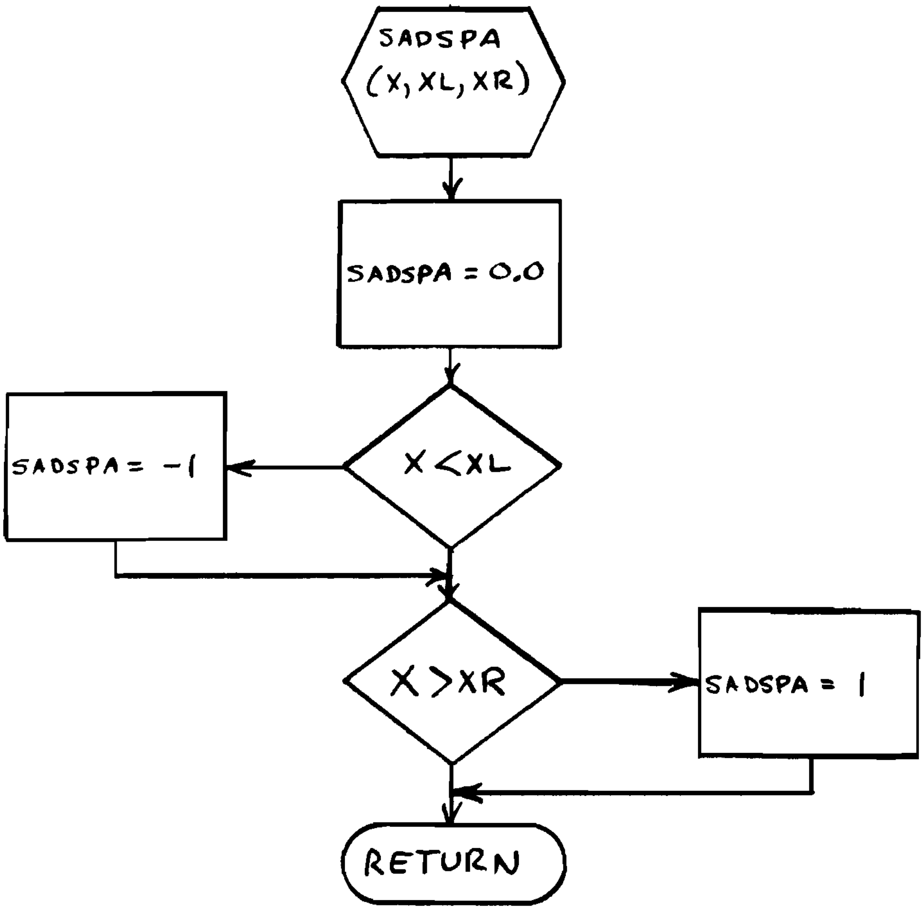


Figure 19. Function SADSPA - Dead Space

The basic limiter operation is illustrated in figure 20. The special priority feature is not shown on the upper diagram but is shown on the FORTRAN flow representation.

FUNCTION SAMOIN

To perform the integral plus proportional function of the PCS, a specialized integrator is required. This integrator must be capable of three modes of operation; normal integration, holding at a limiting value with immediate change when the derivative changes sign, and resetting to an initial value not necessarily the same as the original initial value.

This capability is obtained by use of the function SAMOIN in conjunction with a DSL/90 integrator. The operation of this function, shown diagrammatically and in FORTRAN flow form, is presented in figure 21.

FUNCTION SASWCH

A binary (0, 1) switching function that provides hysteresis is required for PCS simulation. The two entries, SAOFON and SAONOF, provide this capability for a normally off and a normally on binary switch, respectively.

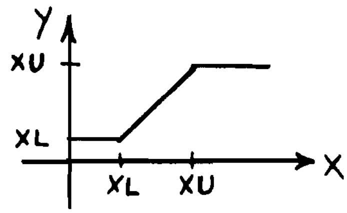
These entries may be used as normally off and normally on switches without hysteresis by setting the upper and lower limits, XL and XU, equal.

Figure 22 illustrates the operation of these entries to the function and the FORTRAN flow diagram.

FUNCTION SAWFAT

This function supplies a binary signal to a mode controlled integrator indicating the need for attenuating the maximum W_f/P_4 limit. When the output of this routine is zero (0) the gain in the W_f/P_4 logic is reduced by a amount determined by input data. When the SAWFAT inputs change, allowing the output to return to unity (1), the integrator returns the fuel flow limit gain to its original value.

Figure 23 shows the flow of FORTRAN logic for this function.

GENERAL FORM	FUNCTION		
	WHEN:	THEN:	
Y = SALIMT(X,XL,XU,TYPE)	X < XL	Y = XL	
SPECIAL PURPOSE LIMITER	X > XU	Y = XH	
	XL ≤ X ≤ XU	Y = X	

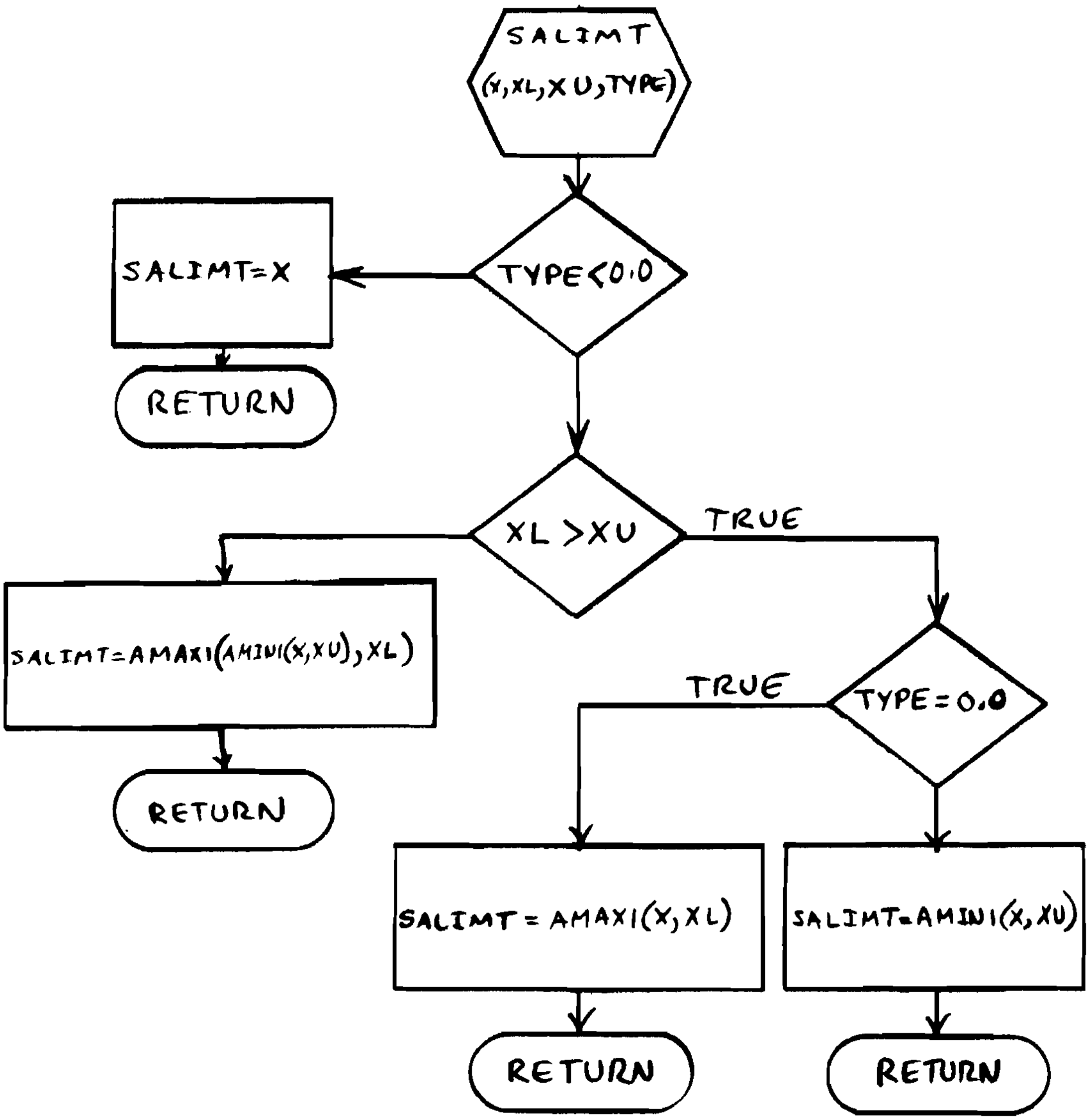


Figure 20. Function SALIMT - Special Purpose Limit Function

GENERAL FORM	FUNCTION
XDØT=SAMØIN(DXDT, X, XL, XU, PATH, XIC, DXDTM)	$\int_0^t XD\emptyset T\,dt + I/C$ PATH ≠ 0
X = INTGRL(I/C, XDØT)	X = XIC PATH = 0
	$X_L \leq \int_0^t XD\emptyset T\,dt + I/C \leq X_U$

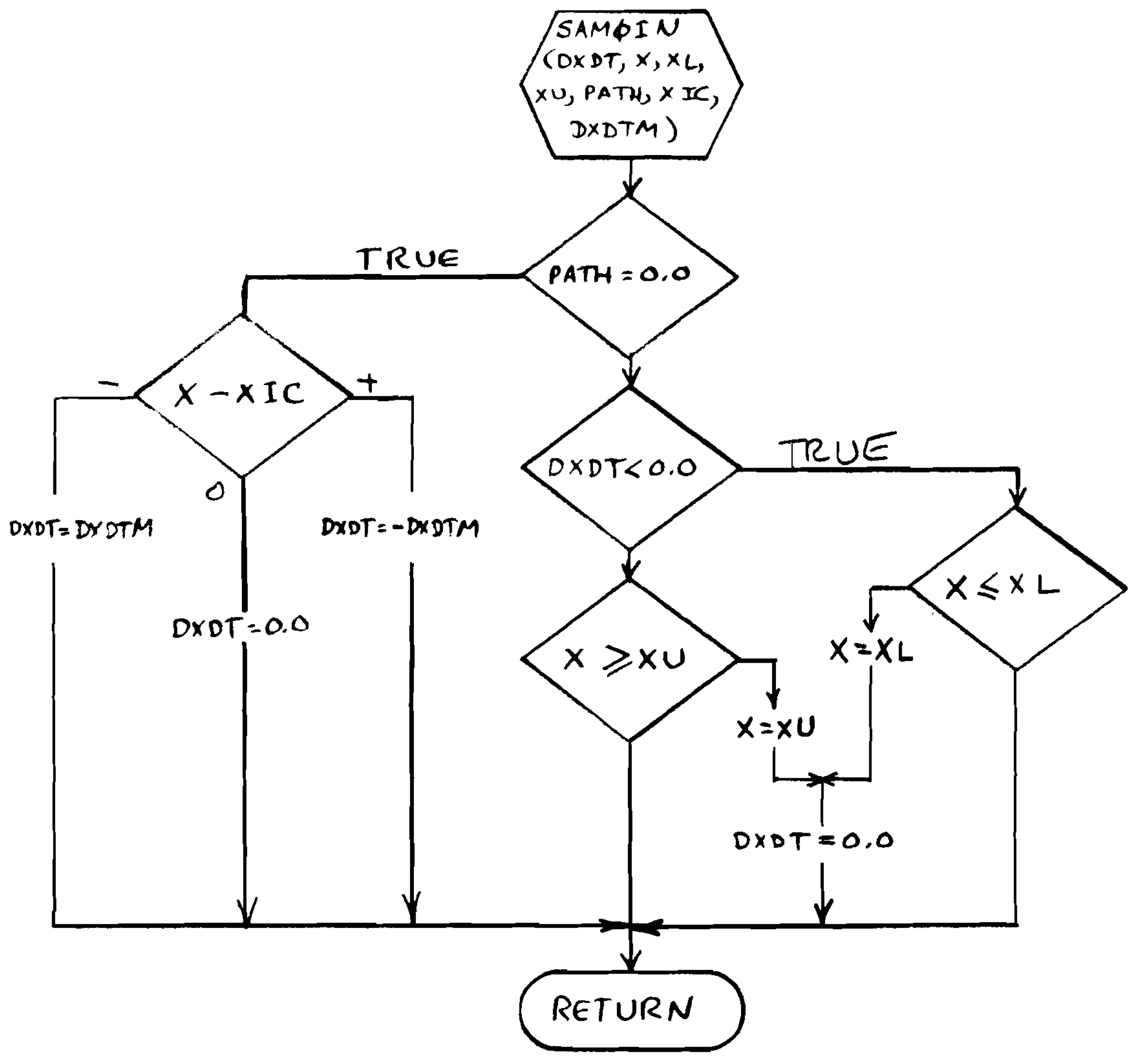
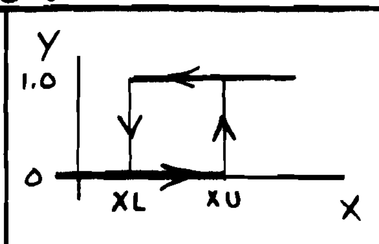
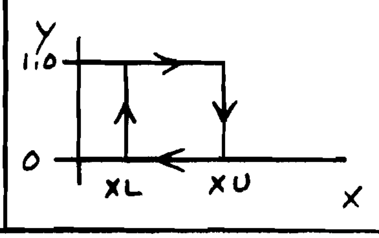


Figure 21. Function SAMOIN - Mode Controlled Integrator

GENERAL FORM	FUNCTION	
Y=SAΦFΦN(X,XL,XU) OFF-ON SWITCH WITH HYSTERESIS	WHEN: THEN: X<XL ; Y=0.0 X>XU ; Y=1.0 X<XU AND Y_{n-1}=0.0; Y=0.0 X>XL AND Y_{n-1}=1.0; Y=1.0	
Y=SAΦNΦF(X,XL,XU) ON-OFF SWITCH WITH HYSTERESIS	WHEN: THEN: X<XL Y=1.0 X>XU Y=0.0 X<XU AND Y_{n-1}=0.0 Y=0.0 X>XL AND Y_{n-1}=1.0 Y=1.0	

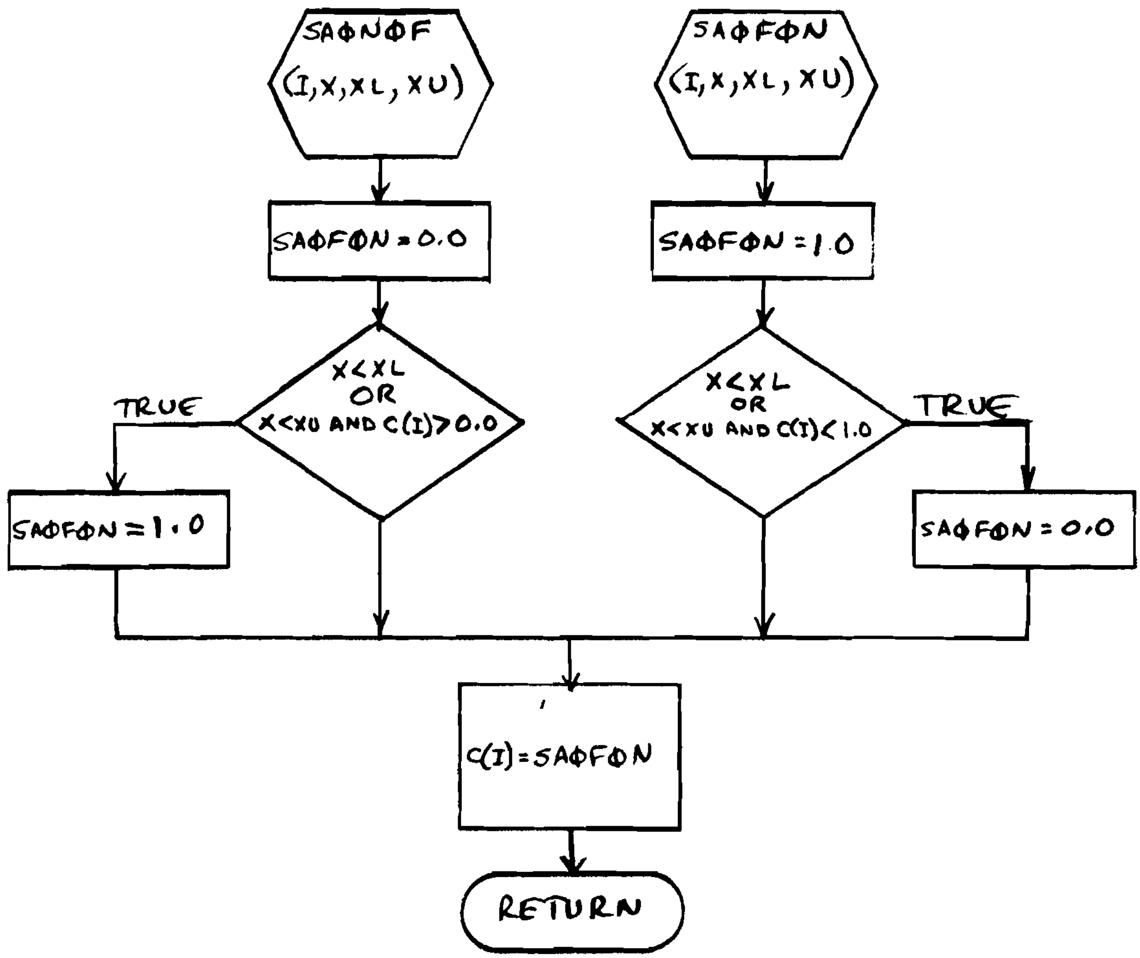


Figure 22. Function SASWCH - Binary Switch Routine With Hysteresis

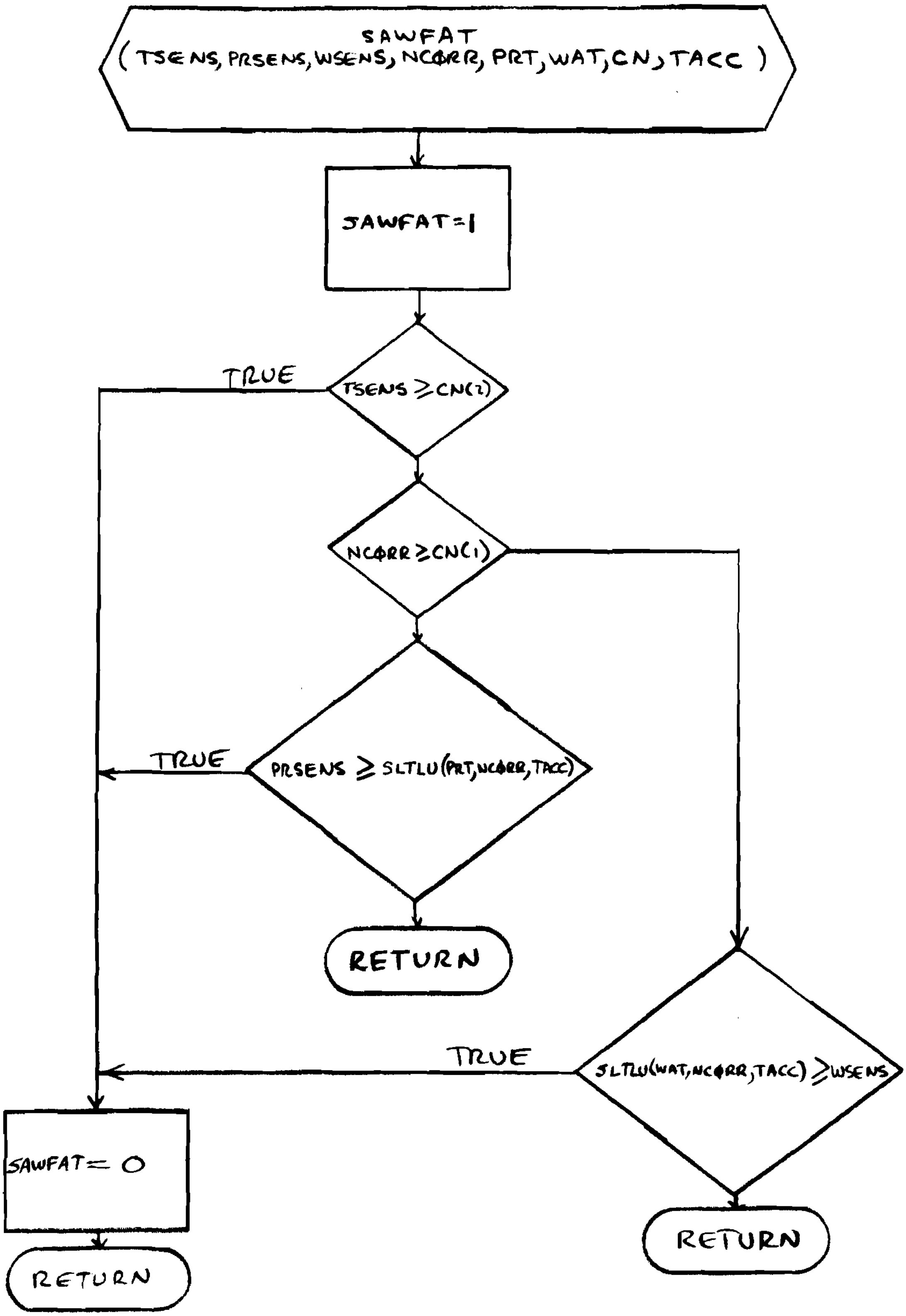


Figure 23. Function SAWFAT

FUNCTION SLFVPG

This utility routine computes the airflow parameter as a function of pressure ratio and gamma (ratio of specific heats).

The derivation of this function is shown below.

W	airflow	lb/sec
T	static temperature	$^{\circ}R$
T_t	total temperature	$^{\circ}R$
P	static pressure	lb/sq. in.
P_t	total pressure	lb/sq. in.
A	area	sq. in.
TR	temperature ratio (T_t/T)	
PR	pressure ratio (P_t/P)	

$$W = AV_p = \frac{AMP}{\sqrt{T}} \sqrt{\frac{\gamma g}{R}}$$

$$\frac{W\sqrt{T}}{AP} = M\sqrt{\frac{\gamma g}{R}}$$

$$\frac{W\sqrt{T_T}}{AP_T} = \frac{W\sqrt{T}}{AP} \cdot \frac{TR^{\frac{1}{2}}}{PR}$$

$$M = \sqrt{2(TR - 1.0)/(\gamma - 1.0)}$$

$$TR = PR^{\frac{\gamma}{1-\gamma}}$$

therefore:

$$\frac{W\sqrt{T_T}}{P_T A} = \sqrt{\frac{2(TR - 1)}{(\gamma - 1)}} \sqrt{\frac{\gamma g}{R}} \Bigg/ TR^{\frac{\gamma+1}{2(\gamma-1)}}$$

Figure 24 shows the FORTRAN flow diagram of this function.

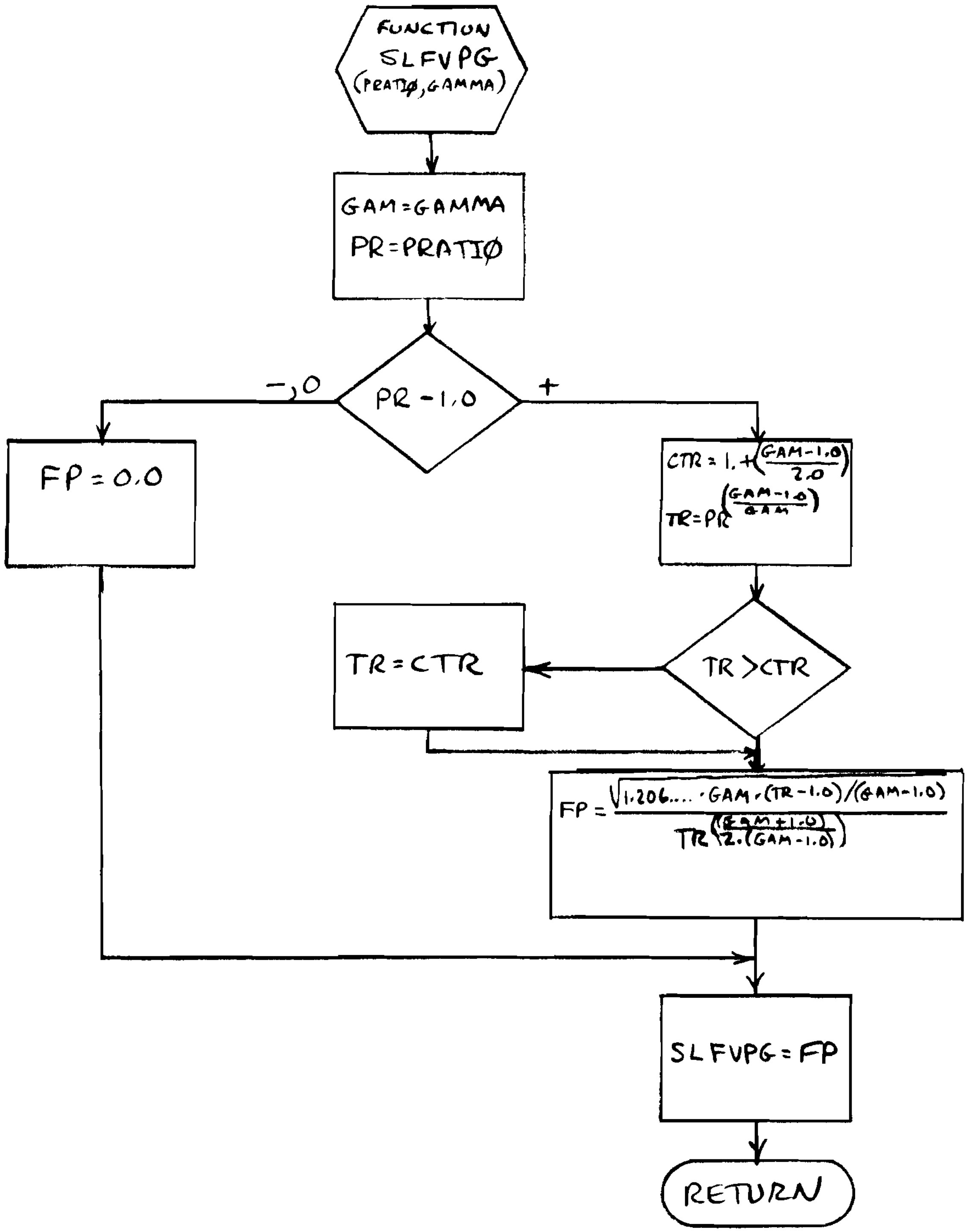


Figure 24. Function SLFVPG

FUNCTION SLGAM

This utility routine computes the ratio of specific heats (γ) as a function of total temperature and fuel-air ratio. The calculations use Grade JP-4 fuel combustion characteristics and are applicable to all of the JP family of fuels.

Two input arguments are used when gamma for a fuel-air mixture is desired. The first argument is temperature, in degrees Rankine, the second is fuel-air ratio, dimensionless. When gamma for air is required, only the temperature argument need be input.

The FORTRAN flow diagram is shown in figure 25.

FUNCTION SLMVFG

This function computes mach number as a function of flow parameter and gamma. The calculation uses a Newton-Raphson iteration using the initial guess for Mach number to determine which solution, subsonic or supersonic, is desired. On successive passes through the routine, the output from the previous pass is used as the initial guess to reduce iteration time.

Inputs to the routine are the initial guess on mach number, XMI, the flow parameter, FLOWP, and the ratio of specific heats, γ . The flow parameter is derived in the description of function SLFVPG. The FORTRAN flow diagram is shown on figure 26 and 27.

FUNCTION SLTLU

This function provides a general purpose table look-up program for use with DSL/90 simulation programs. Functions of one, two and three independent parameters are handled as well as constants. Interpolation in univariant and bivariant tables can be selected as either linear or LaGrangian for each independent parameter. In the trivariant tables, the interpolation between bivariant families is linear.

Figures 27, 28, and 29 illustrate the usage of this routine for tables of one, two, and three independent parameters. Also shown on figure 27 is the usage when the table value is a constant.

FUNCTION SPBLOW

The burner blowout routine incorporates logic which effectively "blows out" a burner when it tries to operate below certain minimum conditions for combustion. A curve plotting minimum burner inlet total pressure (for combustion) versus fuel-air ratio is compared with actual

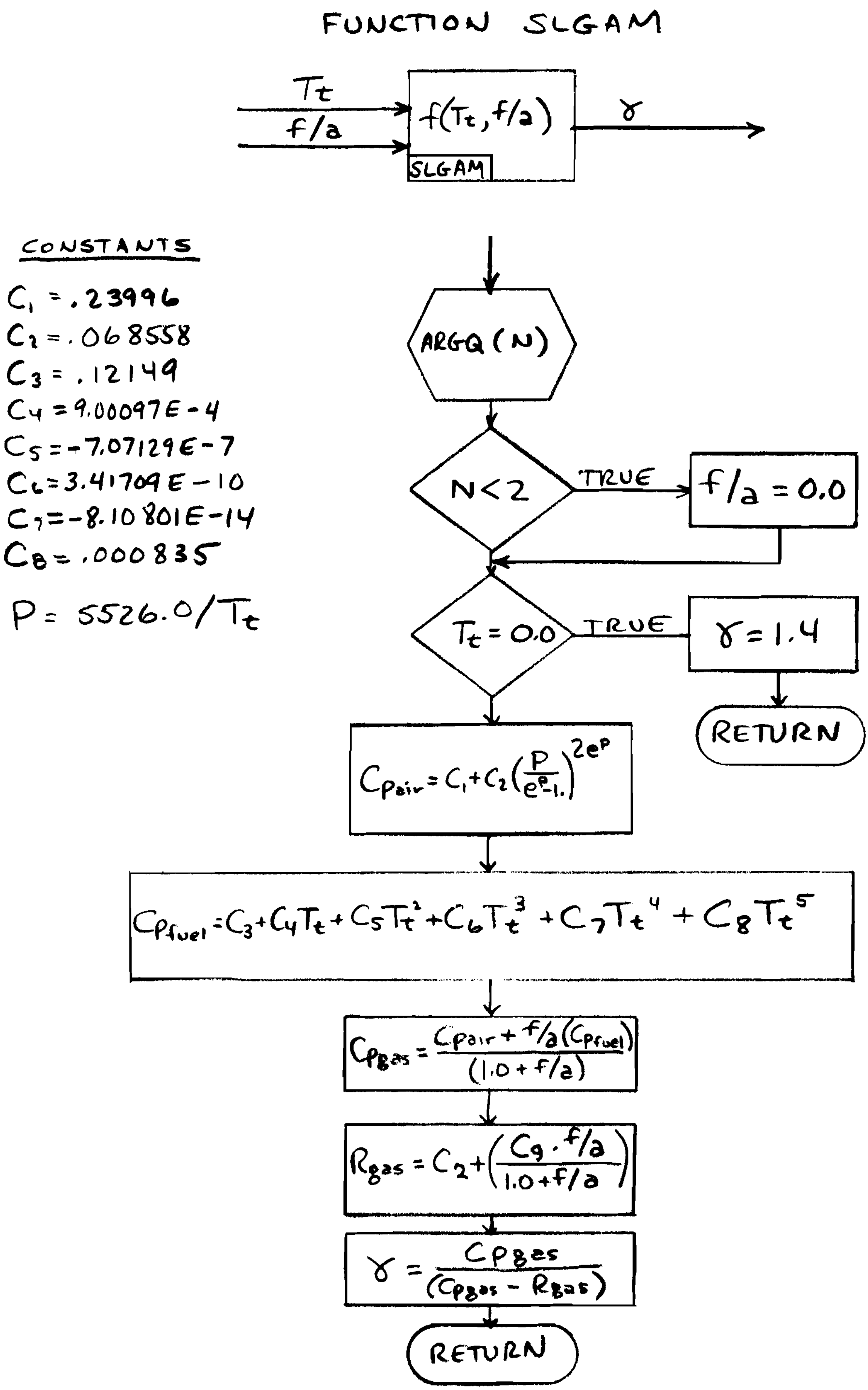


Figure 25. Function SLGAM

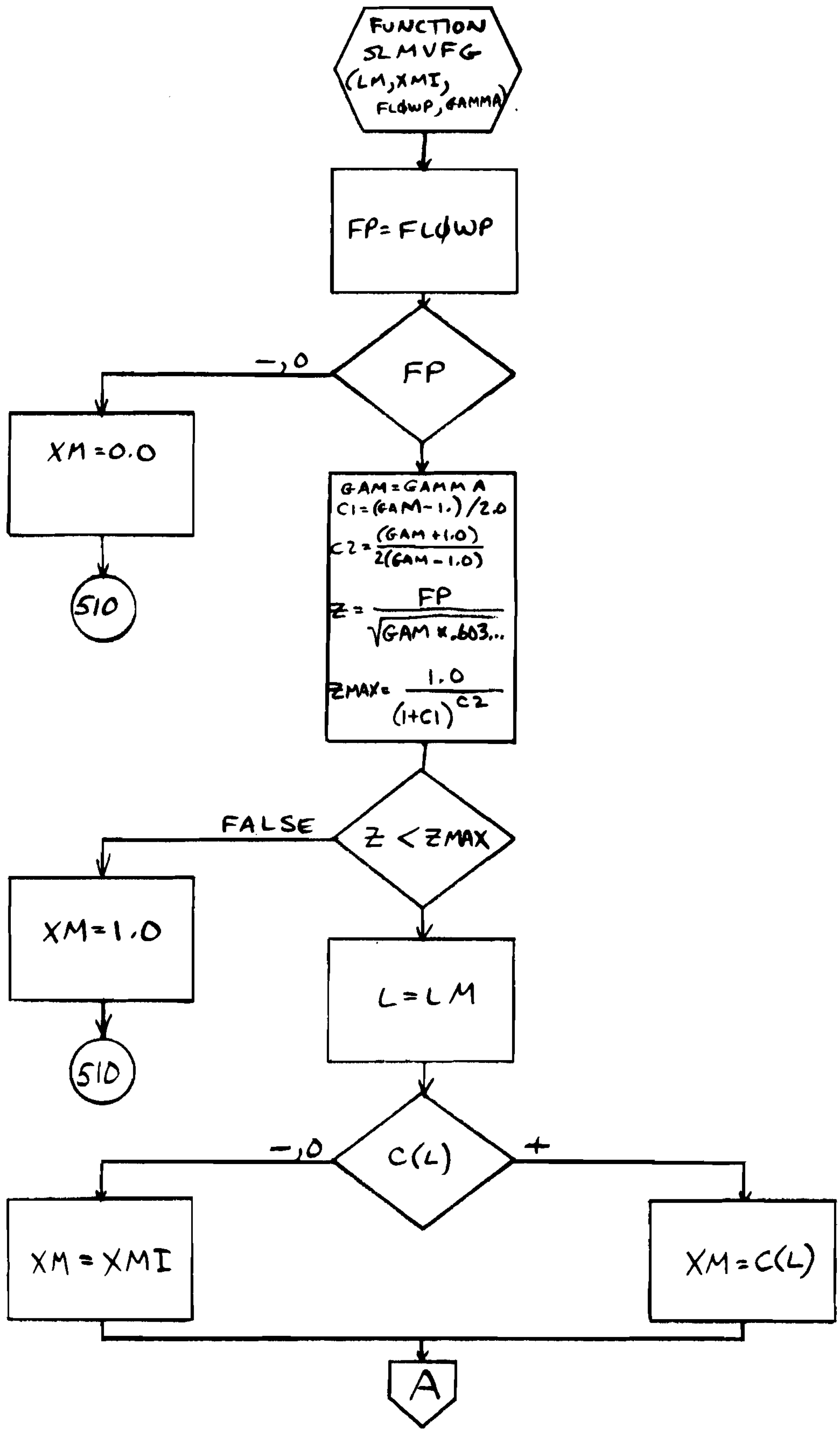


Figure 26. Function SLMVFG

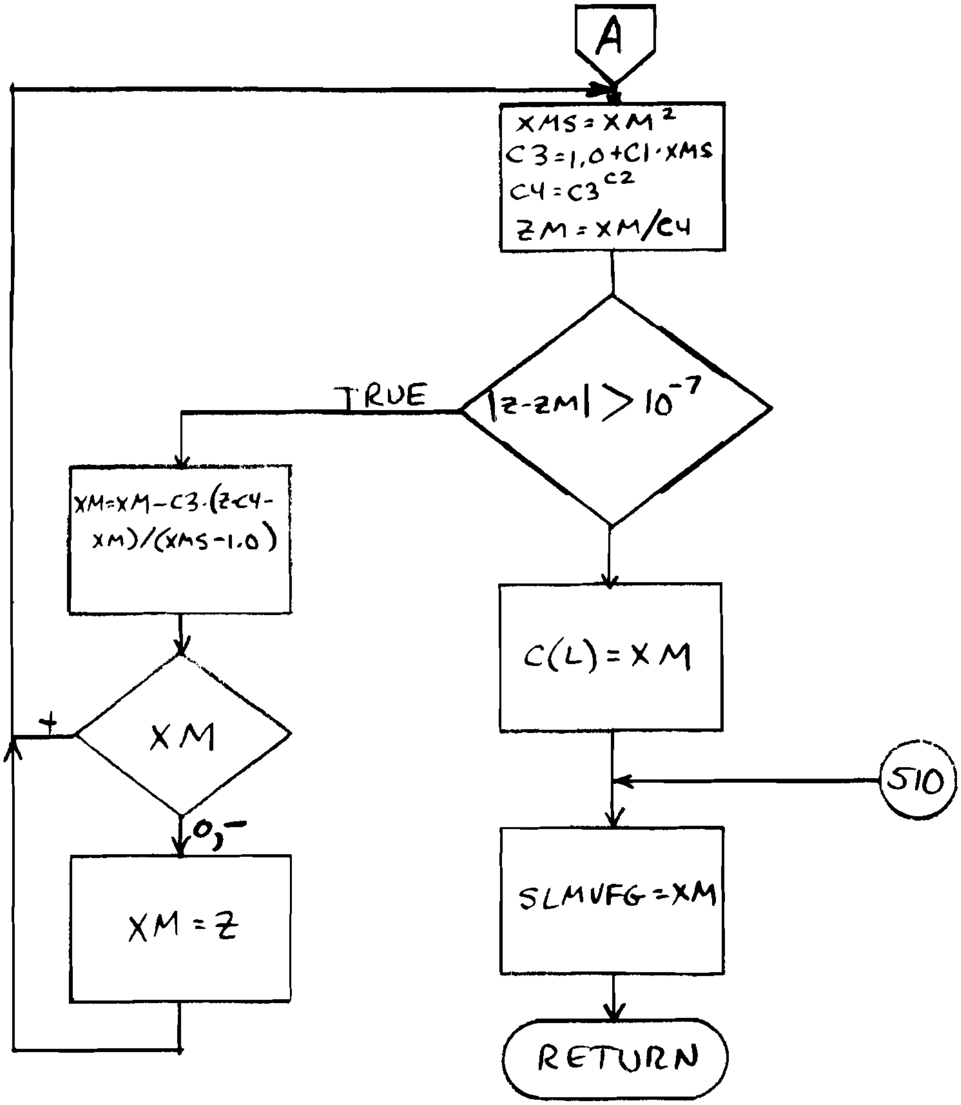
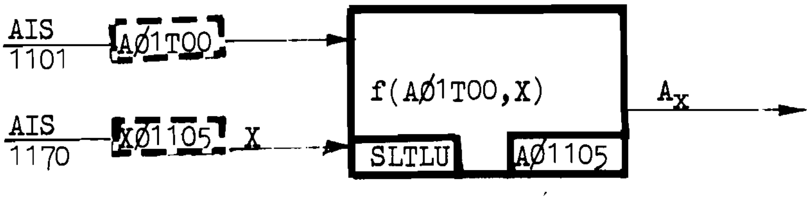


Figure 26. Function SLMVFG (Concluded)

TWO DIMENSIONAL



y = f(X)

STØRAG YA( )

CODING:

y = SLTLU(YA,X)

TABULAR INPUT ARRAY:

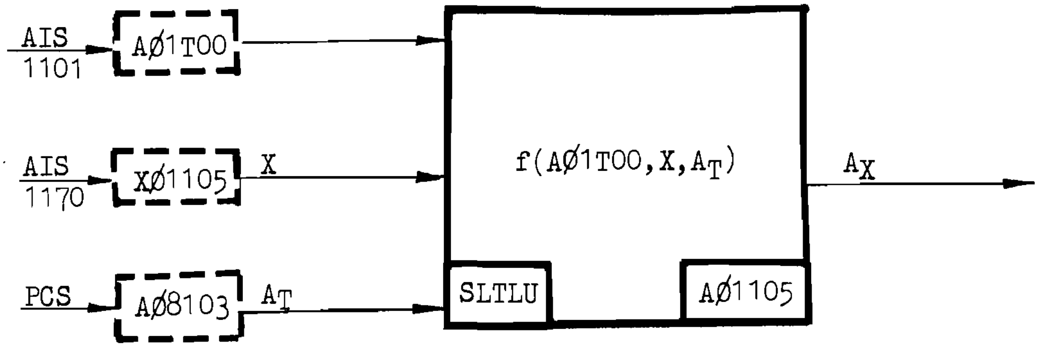
YA(1)	2.0	DENOTES TWO DIMENSIONAL TABLE
(2)		NØ. OF X's
(3)		NØ. OF X POINTS FOR INTERPOLATION
		LIST OF X's
(4)		X_1
(5)		X_2
(6)		X_3
(7)		X_4
(8)		X_5
( )		⋮
( )		⋮
		LIST OF Y's
( )		Y_1
( )		Y_2
( )		Y_3
( )		Y_4
( )		Y_5
( )		:
( )		:

FOR CONSTANT INPUT

YA(1)	1.0	DENOTES CONSTANT FOLLOWING
(2)	C	VALUE OF CONSTANT

Figure 27. Variable Increment Table Look-Up

THREE DIMENSIONAL



Z = f(X,Y)

CODING:

STØRAG ZA( )

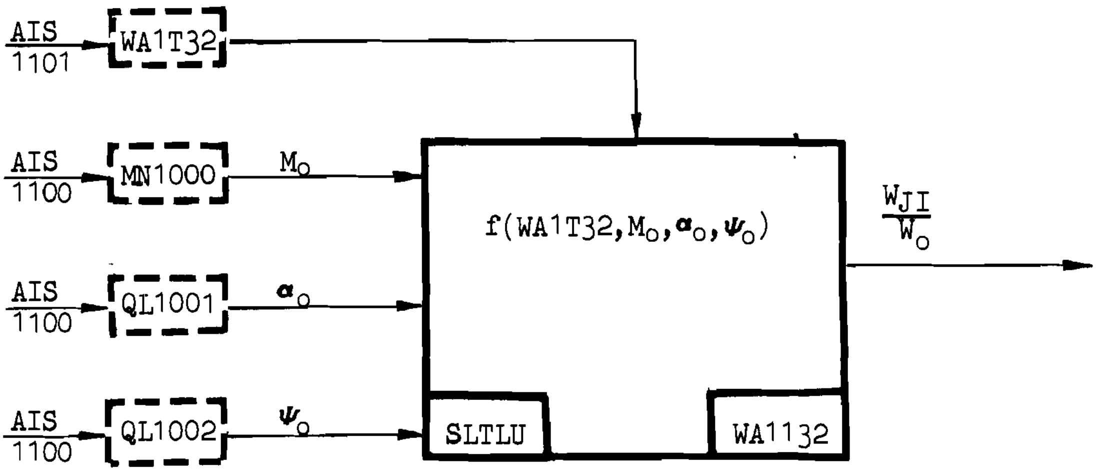
Z = SLTLU(ZA,X,Y)

TABULAR INPUT ARRAY:

ZA(1)	3.0	DENOTES THREE DIMENSIONAL TABLE
(2)		NO. OF X's
(3)		NO. OF X POINTS FOR INTERPOLATION
(4)		NO. OF Y's
(5)		NO. OF Y POINTS FOR INTERPOLATION
		LIST OF X's
(6)		X_1
( )		X_2
( )		X_3
( )		:
		LIST OF Y's
( )		Y_1
( )		Y_2
( )		:
		LIST OF Z's for Y_1
( )		$Z_{1,1}$
( )		$Z_{2,1}$
( )		$Z_{3,1}$
( )		:
( )		:
		LIST OF Z's FOR Y_2
( )		$Z_{1,2}$
( )		$Z_{2,2}$
( )		$Z_{3,2}$
( )		:

Figure 28. Variable Increment Table Look-Up

FOUR-DIMENSIONAL



Z = f(X,Y,W)

STORAG ZA( )

TABULAR INPUT ARRAY:

CODING:

Z = SLTLU(ZA,X,Y,W)

ZA(1)	4.0	DENOTES FOUR DIMENSIONAL TABLE
(2)		NO. OF W's
		LIST OF W's
(3)		W_1
(4)		W_2
(5)		W_3
( )		:
( )		:

THREE DIMENSIONAL FOR W_1
NO. OF X's
NO. OF X POINTS FOR INTERPOLATION
NO. OF Y's
NO. OF Y POINTS FOR INTERPOLATION

LIST OF X's

X_1
X_2

SAME AS THREE DIMENSIONAL INPUT

REPEAT FOR W_2, W_3, ...

Figure 29. Variable Increment Table Look-Up

operating conditions. If the actual inlet total pressure is below the minimum value, the burner efficiency is reduced to zero on a time constant to simulate the blowout. This reduces the burner temperature rise to zero. When inlet total pressure returns to a value above the minimum the efficiency is restored on a time constant.

Figure 30 shows the FORTRAN flow diagram for this routine. The coding has not been accomplished for this routine so the listing will not be found in Appendix I.

SUBROUTINE SPCOMP

The purpose of this routine is to provide a simulation model for fans and compressors. The characteristics of the particular component being used are entered as table names in the argument list of the routine. Other required input data such as the incoming pressure (P_{tin}), the incoming temperature, (T_{tin}), the exit pressure (P_{tout}), and the rotor speed (N) are also entered as arguments. The outputs of the routine are temperature (T_{tout}), change in enthalpy (Δh), airflow (W), and the average ratio of specific heats (γ).

Several versions of the subroutine are documented. The earliest (-01) has the outputs described above, the later version (-02) has added outputs for surge margin (SRGM) and efficiency (η). These added outputs were always computed within SPCOMP but not available externally. They were added to make them available for the SPTACL calculations.

Addition features are shown in the flow diagram of figure 31, such as interstage bleed and variable geometry provisions. These sections have not been incorporated in the coding as of this report.

FUNCTION SPMEMF

This function is to provide a history for a variable in order to break an implicit mathematical loop. The initial value, XIC, must be provided from initialization logic. After the first pass the output value is equal to the computed value, X, from the previous pass.

Figure 32 shows the flow diagram for the FORTRAN logic.

FUNCTION SPTACL

The purpose of the acceleration schedule calculator, SPTACL, is to calculate the maximum fuel flow (Wfe/Pt4) that the engine can hit on an acceleration without exceeding either a set high compressor surge margin or a maximum turbine inlet temperature. The schedule is usually used in

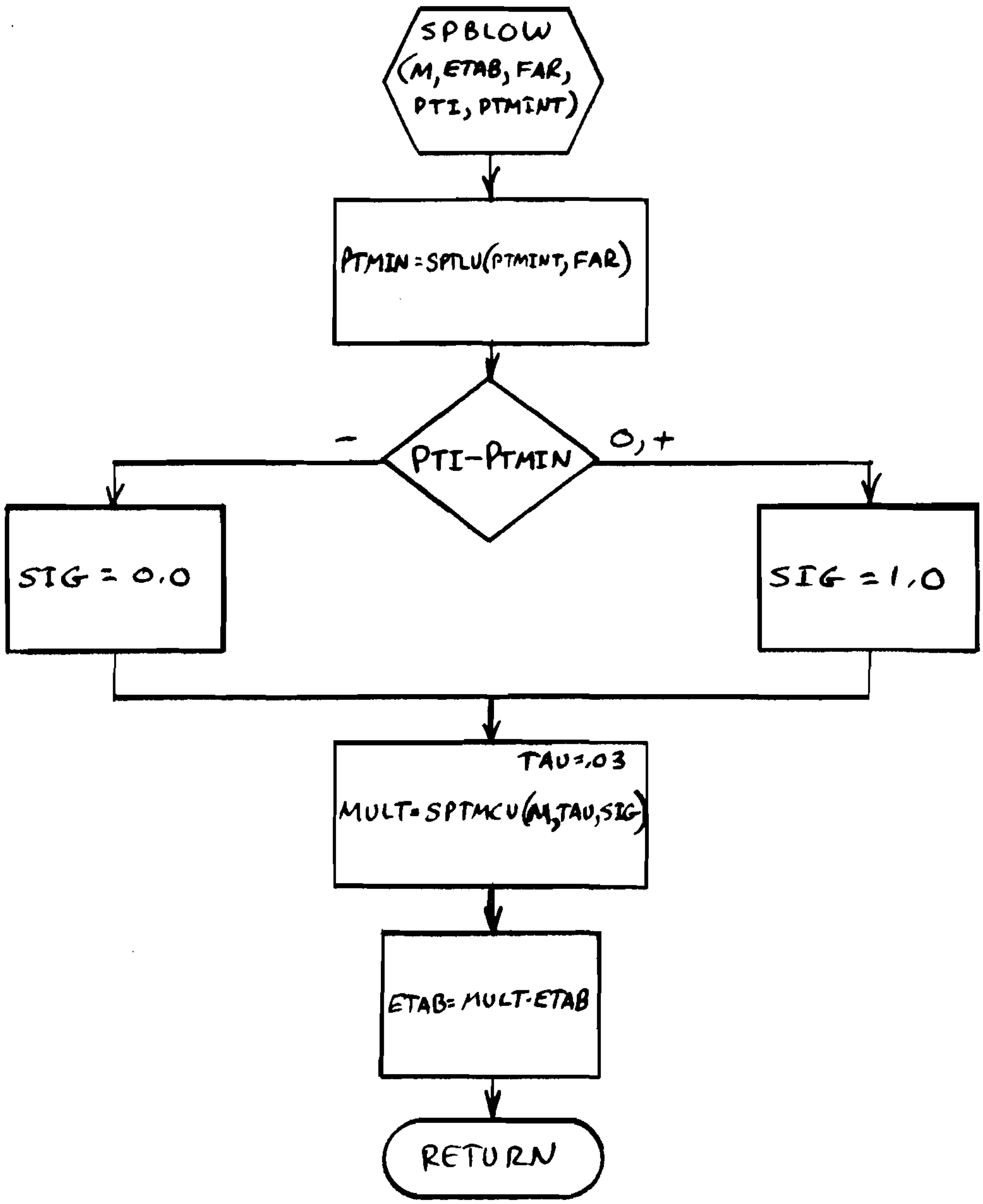
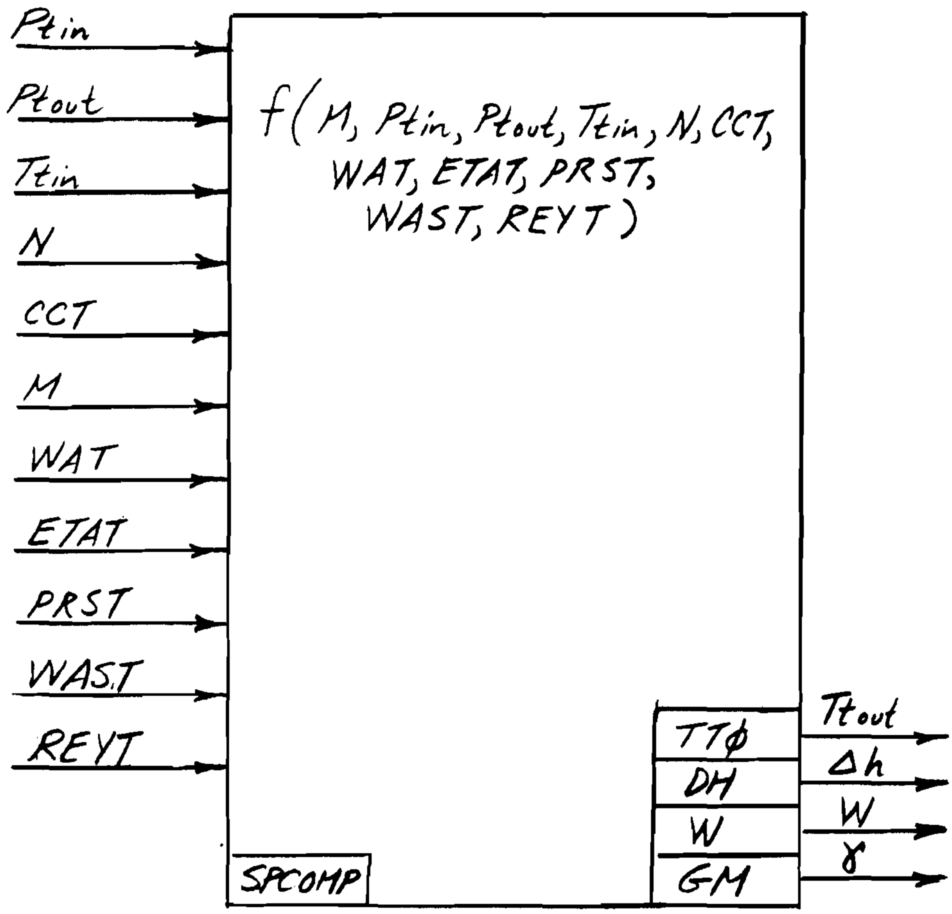


Figure 30. Function SPBLOW - Burner Blowout Limit Routine

FUNCTION SPCOMP



SPCOMP 1-01

Figure 31. Function SPCOMP

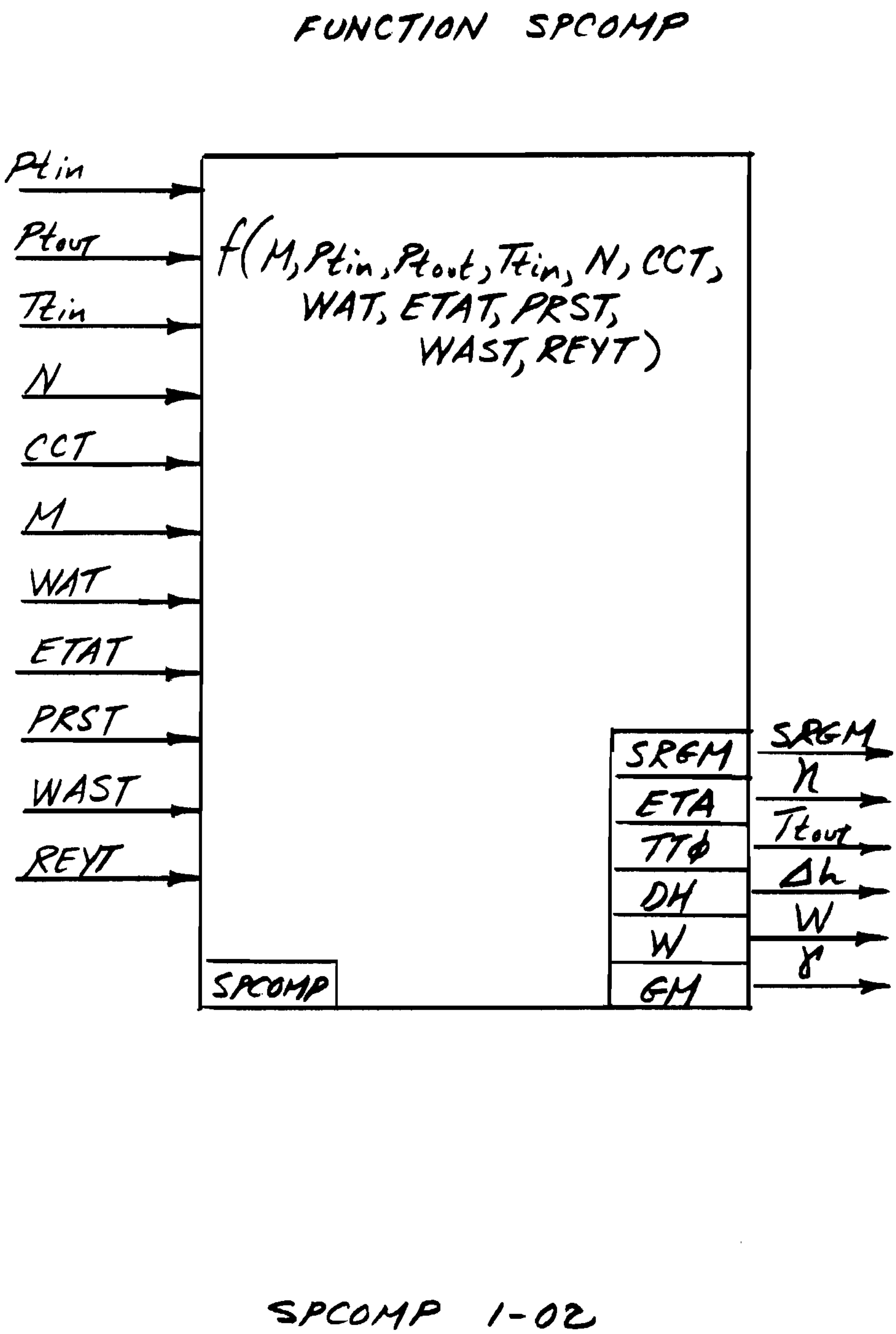


Figure 31. Function SPCOMP (Continued)

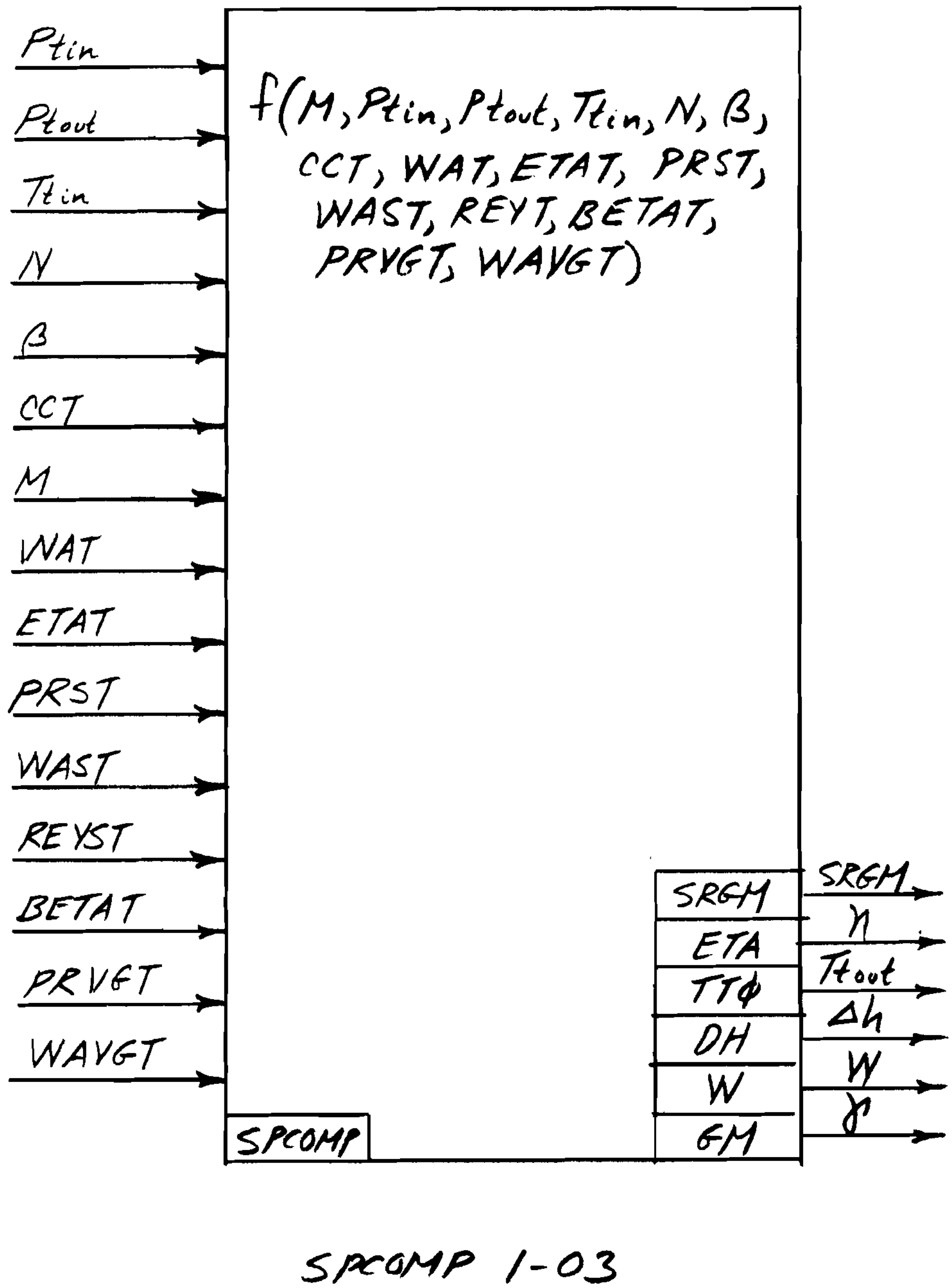


Figure 31. Function SPCOMP (Continued)

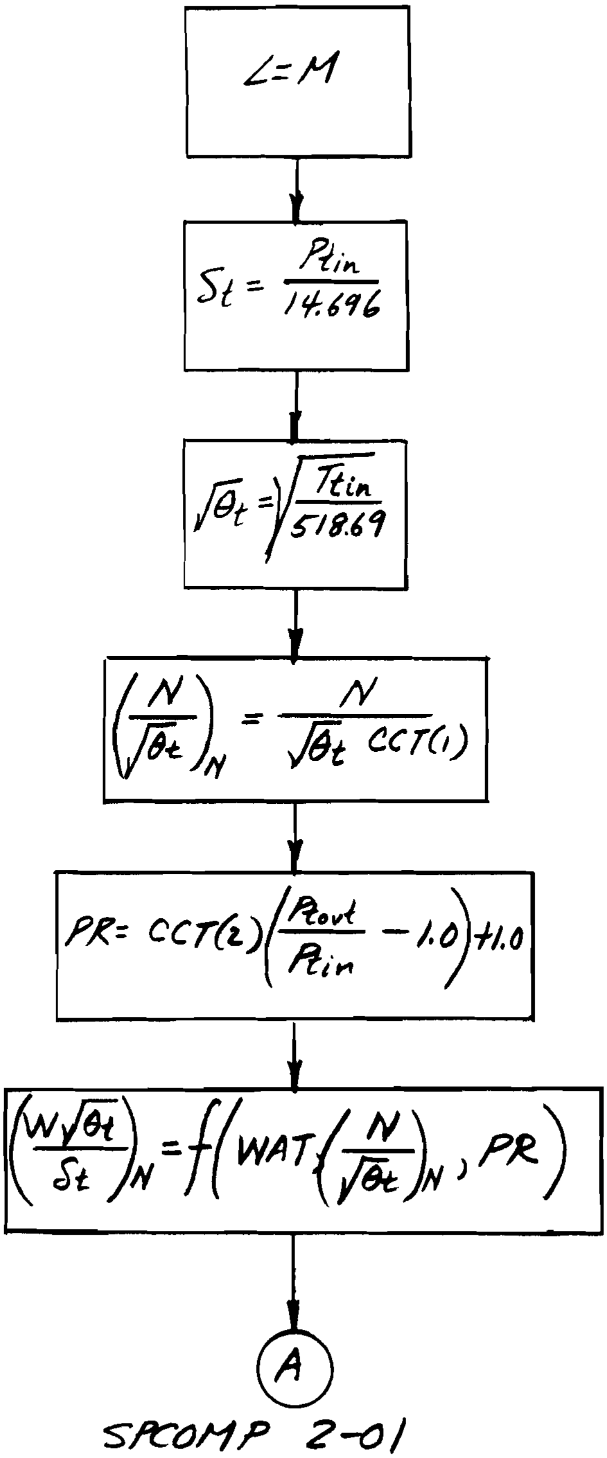
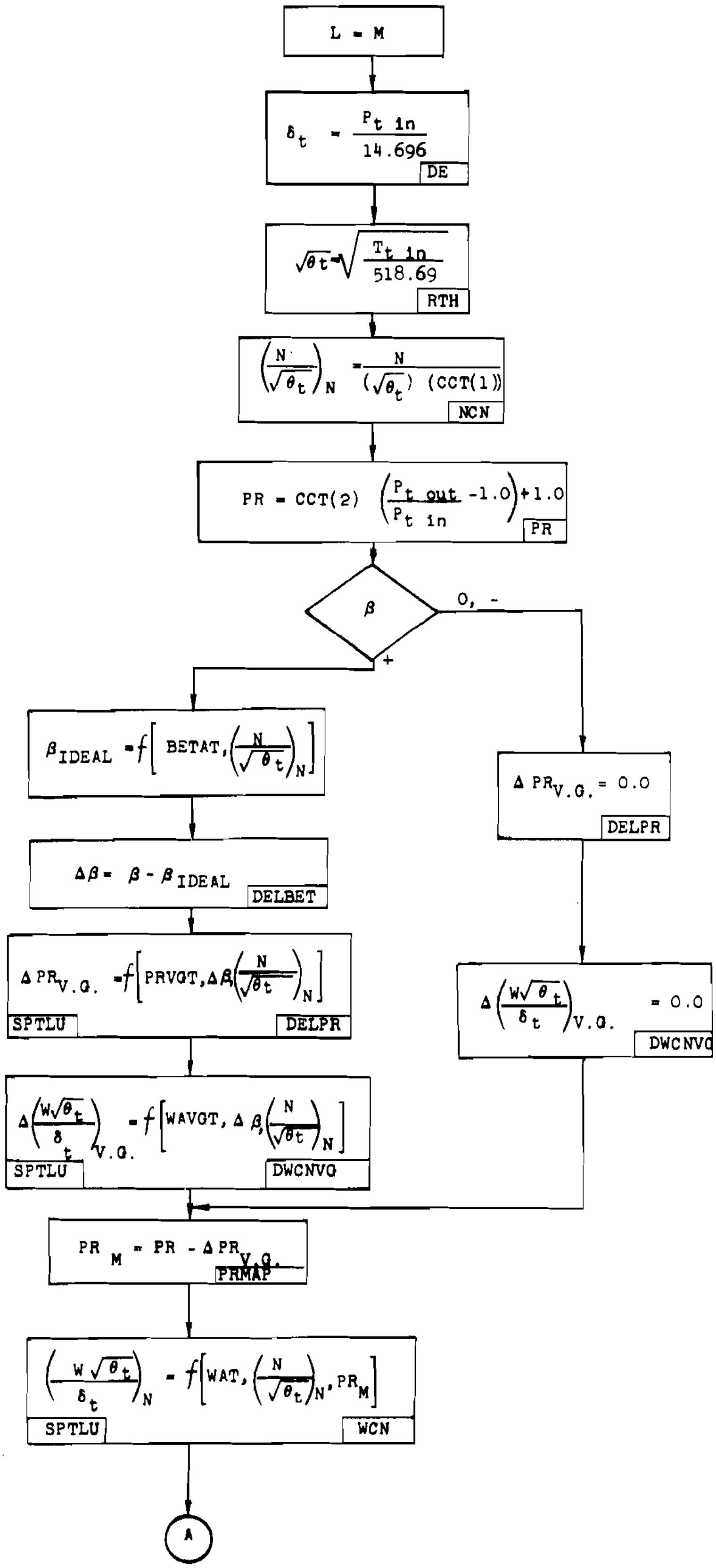


Figure 31. Function SPCOMP (Continued)


L = M
δt = Pt in / 14.696
DE
√θt = √(Tt in / 518.69)
RTH
(N/√θt)N = N / (√θt) (CCT(1))
NCN
PR = CCT(2) (Pt out / Pt in -1.0) + 1.0
PR
β
0, -
+
βIDEAL = f[BETAT, (N/√θt)N]
Δβ = β - βIDEAL
DELBET
ΔPRV.G. = f[PRVGT, Δβ, (N/√θt)N]
SPTLU
DELPR
Δ(W√θt/δt)V.G. = f[WAVGT, Δβ, (N/√θt)N]
SPTLU
DWCNVG
Δ PRV.G. = 0.0
DELPR
Δ(W√θt/δt)V.G. = 0.0
DWCNVG
PRM = PR - ΔPRV.G.
PRMAP
(W√θt/δt)N = f[WAT, (N/√θt)N, PRM]
SPTLU
WCN
A

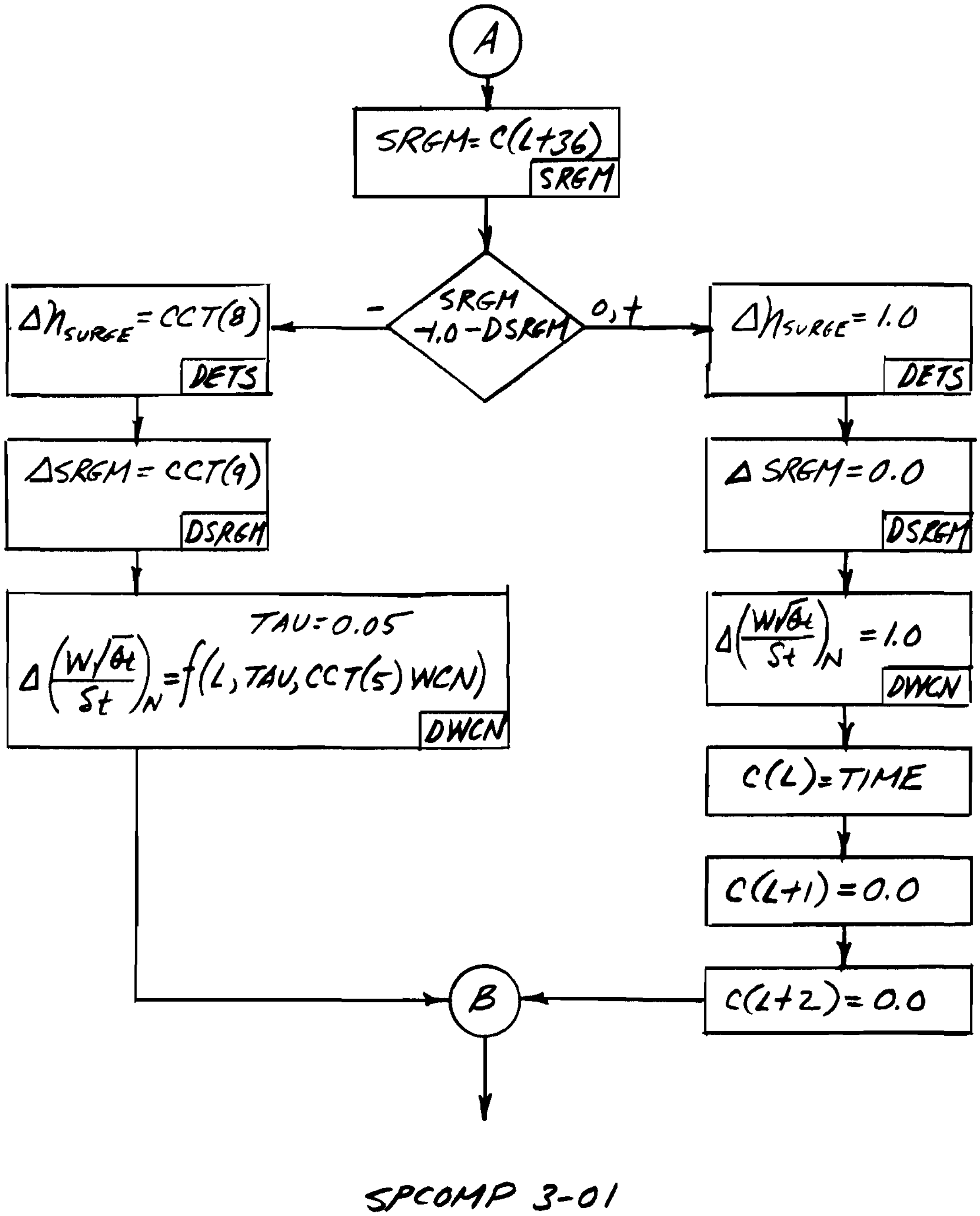


Figure 31. Function SPCOMP (Continued)

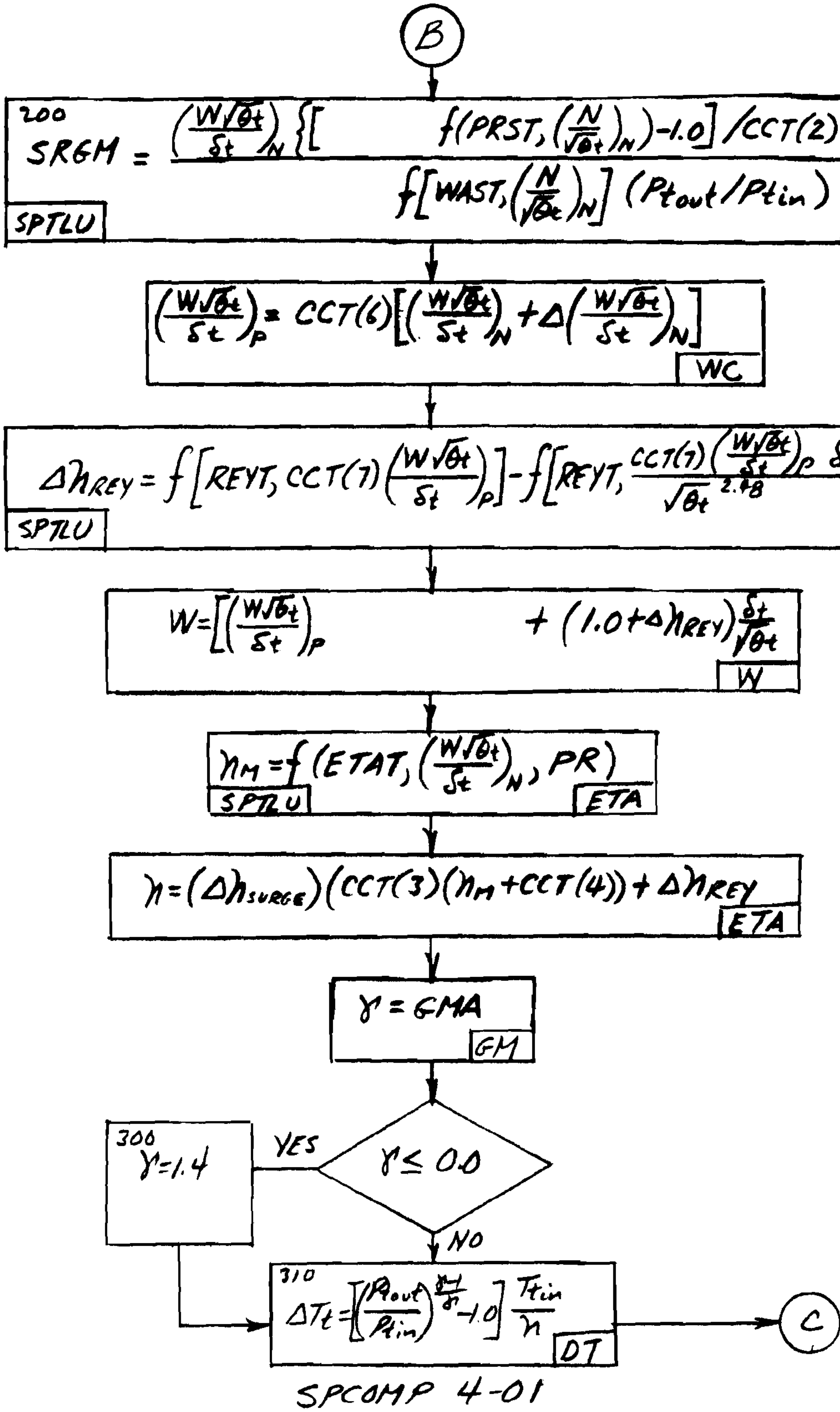
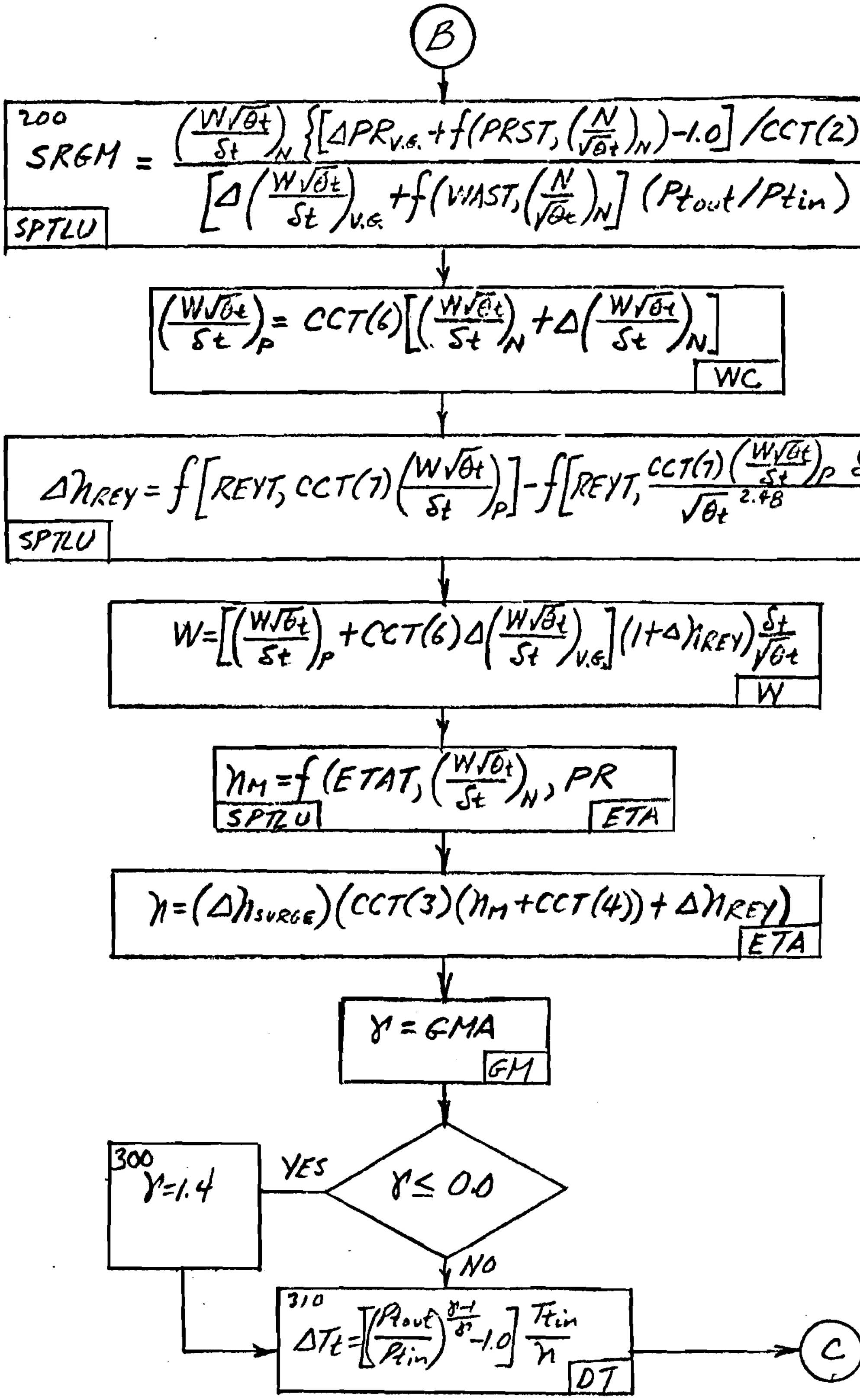


Figure 31. Function SPCOMP (Continued)



SPCOMP 4-02

Figure 31. Function SPCOMP (Continued)

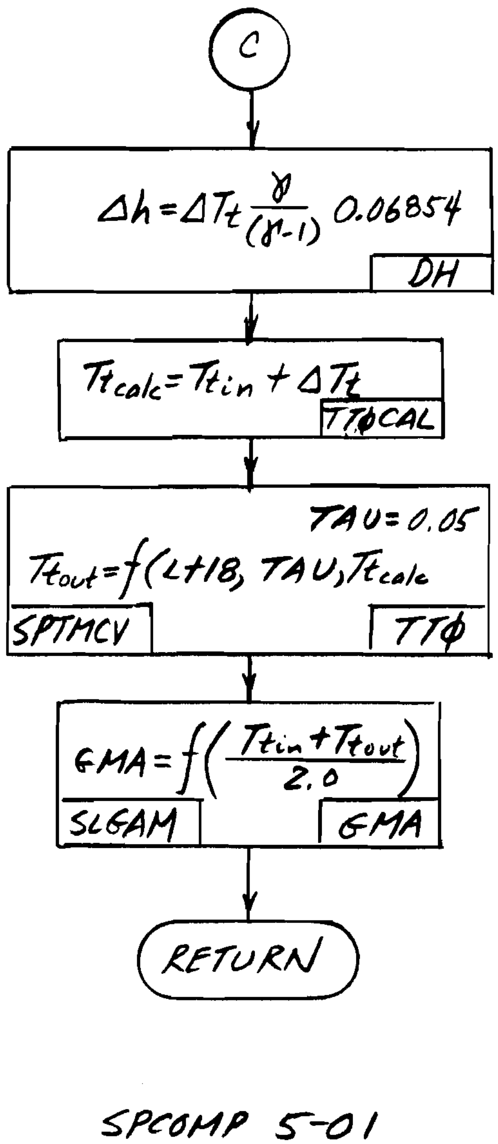


Figure 31. Function SPCOMP (Concluded)

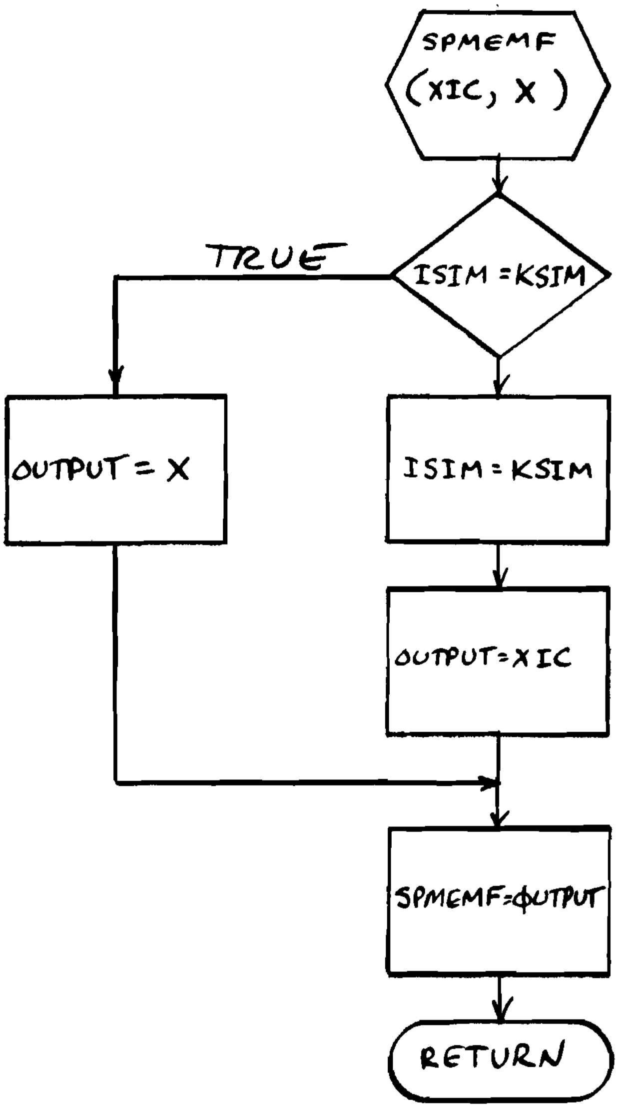


Figure 32. Function SPMEMF

the control as a function of high rotor speed, N_2 (for a turbofan engine).

The routine basically calculates the amount of fuel, Wfs, which would cause the engine to operate at a point corresponding to a set surge margin (CAC_1). It does this from a calculation of the turbine inlet temperature at this point of set surge margin. If this temperature exceeds the maximum allowable turbine inlet temperature (Tt5 MAX), then the fuel flow calculation is limited to the value which gives Tt5 MAX.

The values of maximum fuel flow calculated in this manner are divided by burner pressure Pt4 to obtain the schedule necessary for the control.

Figure 33 shows the FORTRAN flow diagram for this routine.

FUNCTION SPTLU

This routine provides a specialized multi-use table look-up routine for use in DSL/90 simulation. The routine is designed to handle data tabulated at constant increments of each independent parameters. It functions as a general univariant and bivariant table look-up for curves tabulated at constant increments and as a special purpose routine reading multiple tables to obtain a single answer.

The special features read several tables and compute a single value as an output. This provision was made to allow the compressor map presentations to be entered with a single statement although three of a set of four tables are actually required to read the map. Provision was also made to make a similar reading of a thrust table possible. These special features are used by entering key numbers in the first location of the table array to be used. For the thrust table the key number is zero (0.0), for the compressor map a one (1.0).

Figures 34, 35, 36, and 37 show the use of the various forms of the function SPTLU with information on the diagram representation, the coding usage and the method for entering data.

SUBROUTINE SPTURB

This routine provides a simulation model for low and high pressure turbines. The characteristics of the turbine being simulated are entered as table names in the argument list of the routine. Also required are data such as incoming airflow (W_{ein}), cooling airflow (W_{Tc}), temperature of cooling flow (T_{t4}), uncooled turbine inlet temperature (T_{tin}), inlet total pressure (P_{tin}), discharge total pressure (P_{tout}) and rotor speed (N). Outputs of the routine are total gas flow into turbine (W_{EBI}), discharge temperature (T_{tout}), change in enthalpy (Δh), total gas flow out of turbine (W_{EB}) and the ratio of specific heats (γ).

The second (-02) version of this routine has one additional output, the cooled inlet temperature (T_{tb}).

Figure 38 shows the FORTRAN flow diagram of this routine.

FUNCTION SARECT

This function uses the rectangular method for integration and has various options; normal integration, holding at limit values with immediate change when the derivative changes sign, and resetting to a preset value.

Figure 39 shows the various tests that are made.

FUNCTION SPTMCV

This function provides a simplified and accurate representation of a time constant function and does not require the use of a pure integration such as the real pole function in the DSL/90 porgram.

Figure 40 shows the derived equation in block diagram form.

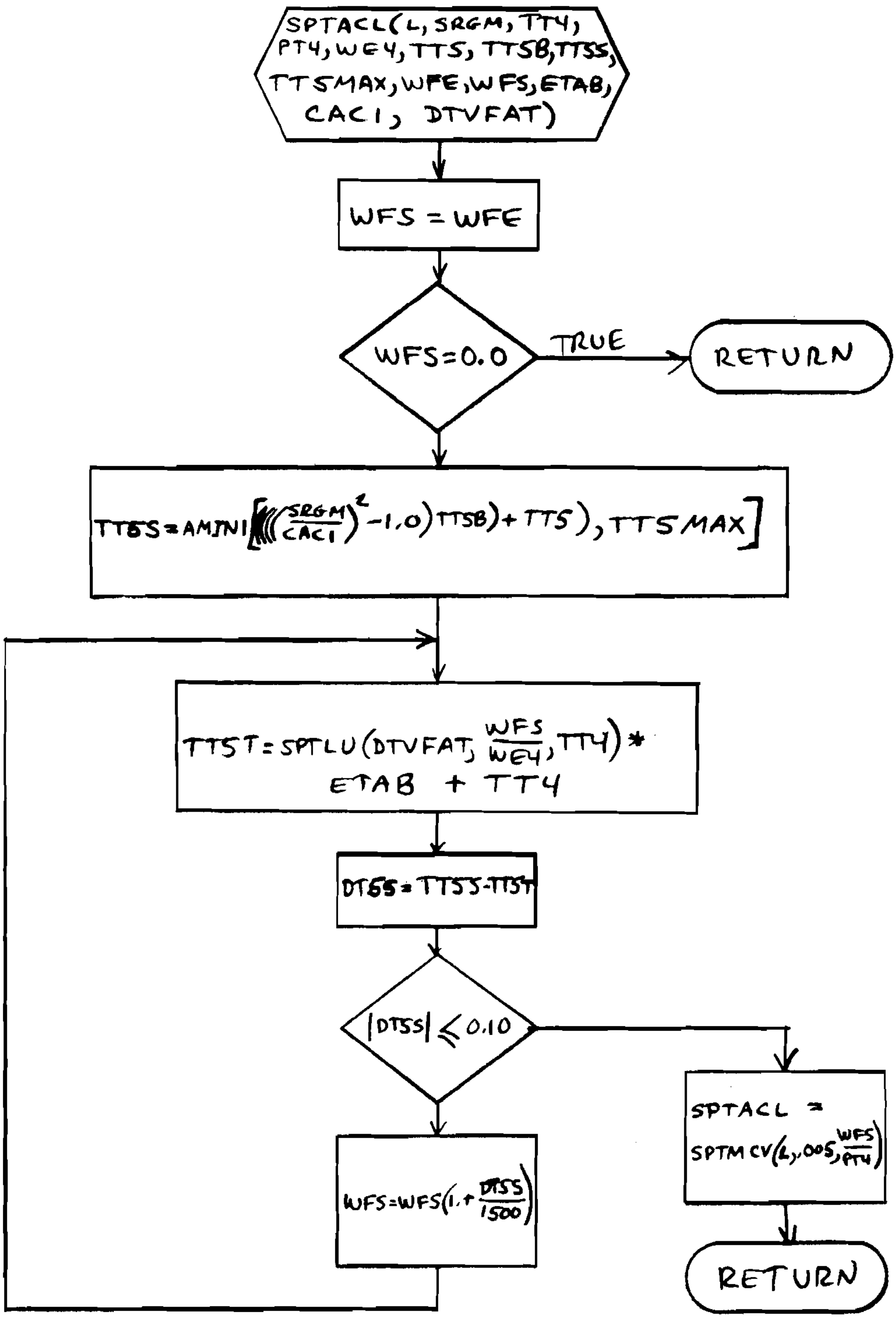
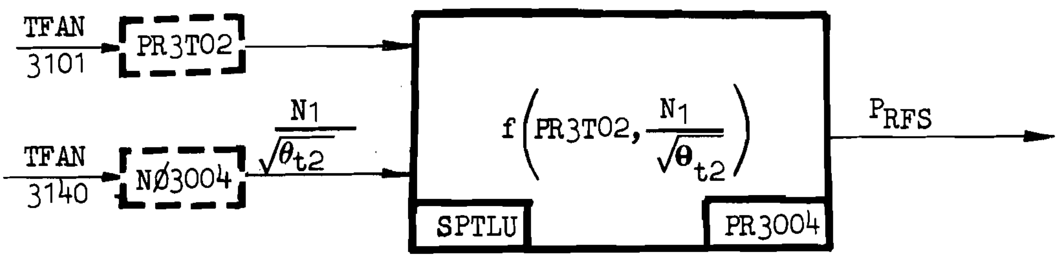


Figure 33. Function SPTACL

GENERAL

TWO DIMENSIONAL



CODING: SPTLU(PR3T02,NØ3004)

example: $Y = f(X)$

$X_1 = 1.1$	$Y_1 = 4.0$
$X_2 = 1.2$	$Y_2 = 4.2$
$X_3 = 1.3$	$Y_3 = 4.8$
$X_4 = 1.4$	$Y_4 = 4.8$
$X_5 = 1.5$	$Y_5 = 4.6$

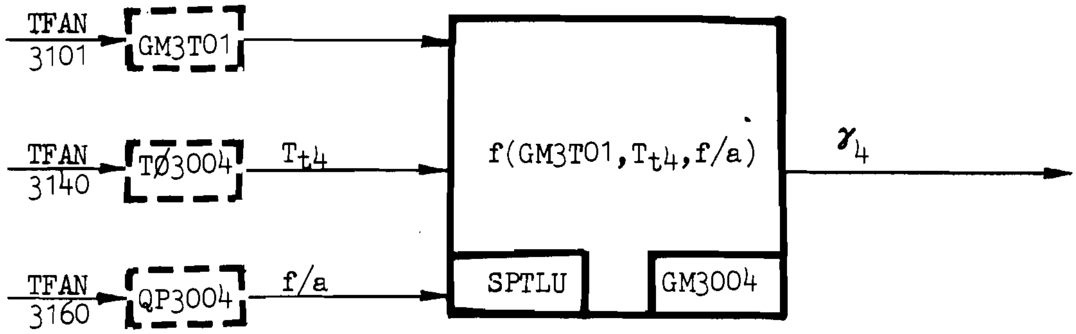
TABULAR INPUT ARRAY:

PR3T02(1)	2.0		denote two dimensional table
" (2)	1.1		minimum value of independent variable
" (3)	1.5		maximum value of independent variable
" (4)	.1		constant increment of independent variable
" (5)	4.0	Y_1	List of dependent values
" (6)	4.2	Y_2	
" (7)	4.8	Y_3	
" (8)	4.8	Y_4	
" (9)	4.6	Y_5	

Figure 34. Constant Increment Table Look-Up

GENERAL

THREE DIMENSIONAL



CODING: SPTLU(GM3T01,TØ3004,QP3004)

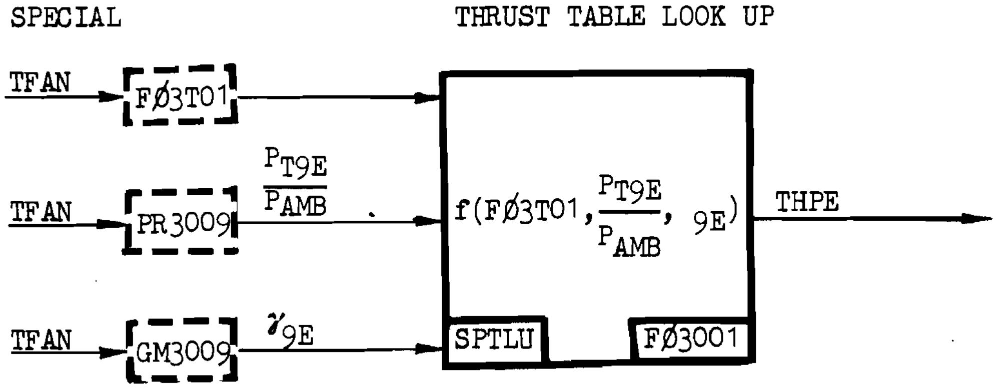
example: $Z = f(X,Y)$

X \ Y	1.2	1.4	1.6
3.0	.1	.2	.4
3.5	.2	.3	.6
4.0	.3	.4	.8
4.5	.4	.5	1.0

TABULAR INPUT ARRAY:

GM3T01(1)	3.0	DENOTE THREE DIMENSIONAL TABLE
GM3T01(2)	3.0	MINIMUM VALUE OF INDEPENDENT VARIABLE X
GM3T01(3)	4.5	MAXIMUM VALUE OF INDEPENDENT VARIABLE X
GM3T01(4)	0.5	INCREMENT VALUE OF INDEPENDENT VARIABLE X
GM3T01(5)	1.2	MINIMUM VALUE OF INDEPENDENT VARIABLE Y
GM3T01(6)	1.6	MAXIMUM VALUE OF INDEPENDENT VARIABLE Y
GM3T01(7)	0.2	INCREMENT VALUE OF INDEPENDENT VARIABLE Y
		LIST OF Z FOR Y_1
GM3T01(8)	.1	$Z_{1,1}$ at X_1
GM3T01(9)	.2	$Z_{2,1}$ X_2
GM3T01(10)	.3	$Z_{3,1}$ X_3
GM3T01(11)	.4	$Z_{4,1}$ X_3
		LIST OF Z FOR Y_2
GM3T01(12)	.2	$Z_{1,2}$ at X_1
GM3T01(13)	.3	$Z_{2,2}$ X_2
GM3T01(14)	.4	$Z_{3,2}$ X_3
GM3T01(15)	.5	$Z_{4,2}$ X_4
		LIST OF Z FOR Y_3
-	-	- -
-	-	- -
-	-	- -

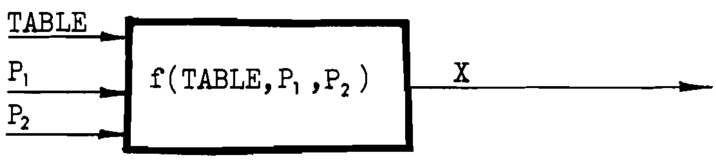
Figure 35. Constant Increment Table Look-Up



FØ3001 = SPTLU(FØ3T01,PR3009,GM3009)

METHOD:

CONTROL INFORMATION AND THREE INDEPENDENT TABLES ARE ENTERED IN ONE ARRAY



IF $P_1 <$ TABLE(4) $\quad X = f(\text{TABLE1}, \log_e P_1, P_2) * f(\text{TABLE2}, P_1)$

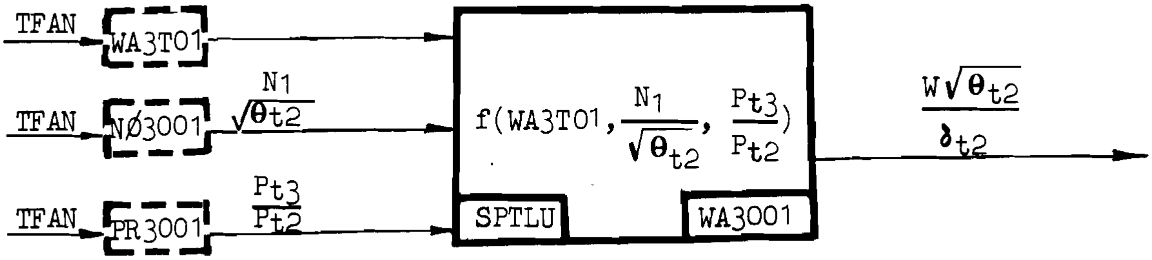
$P_1 \geq$ TABLE(4) $\quad X = f(\text{TABLE2}, \log_e P_1, P_2) * f(\text{TABLE3}, P_1)$

TABULAR INPUT ARRAY

TABLE(1)	0.0	DENOTE SPECIAL TABLE LOOK-UP
(2)	101	1st LOCATION OF TABLE 2 IN ARRAY (INTEGER)
(3)	141	1st LOCATION OF TABLE 3 IN ARRAY (INTEGER)
(4)		VALUE SEPARATING TABLE 2 AND 3
TABLE(5)		1st LOCATION OF THREE DIMENSIONAL TABLE 1 (INPUT SAME AS DESCRIBED UNDER GENERAL TABLE LOOK-UP)
.		
.		
.		
.		
.		
TABLE(101)		1st LOCATION OF TWO DIMENSIONAL TABLE 2
.		
.		
.		
TABLE(141)		1st LOCATION OF TWO DIMENSIONAL TABLE 3
.		
.		
.		

Figure 36. Constant Increment Table Look-Up

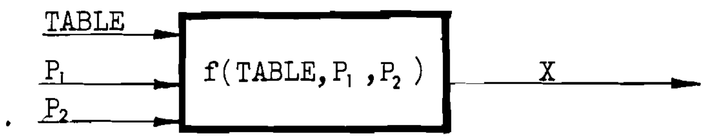
SPECIAL SINGLE VALUE FROM FOUR TABLES



CODING: WA3001 = SPTLU(WA3T01,NØ3001,PR3001)

METHOD:

CONTROL INFORMATION AND FOUR INDEPENDENT TABLES ARE ENTERED IN ONE ARRAY



$$S_1 = f(TABLE1, P\ )$$

IF $P_2 < S_1$: $X = f(TABLE2, P_1) + f(TABLE3, S_1 - P_2, P_1)$

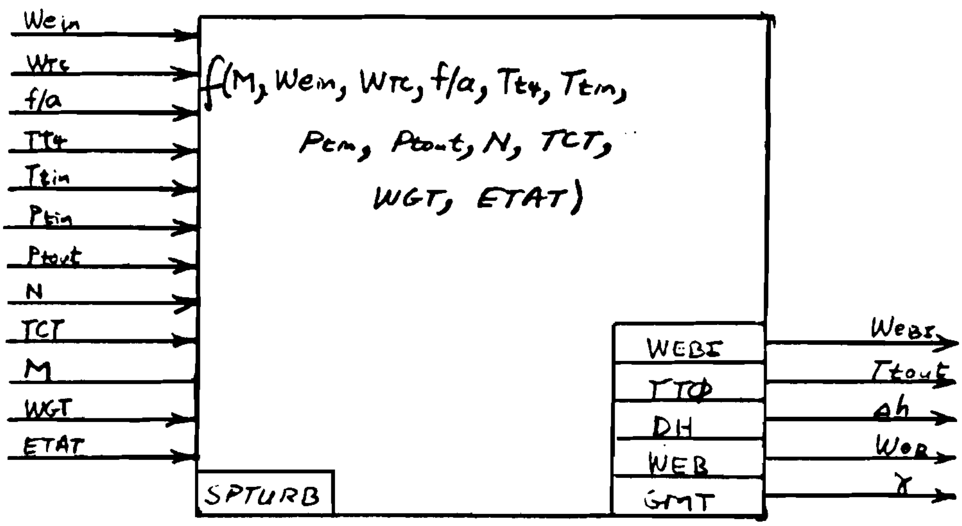
$P_2 \geq S_1$: $X = f(TABLE2, P_1) + f(TABLE4, P_2 - S_1, P_1)$

TABULAR INPUT ARRAY

TABLE (1)	1.0	DENOTE SPECIAL TABLE LOOK UP
(2)	51	1st LOCATION OF TABLE 2 IN ARRAY (INTEGER)
(3)		1st LOCATION OF TABLE 3 IN ARRAY (INTEGER)
(4)		1st LOCATION OF TABLE 4 IN ARRAY (INTEGER)
TABLE (5)		1st LOCATION OF TWO DIMENSIONAL TABLE 1
.		(INPUT SAME AS DESCRIBED UNDER GENERAL TABLE LOOK-UP)
.		
.		
TABLE (51)		1st LOCATION OF THREE DIMENSIONAL TABLE 2
.		
.		
.		

Figure 37. Constant Increment Table Look-Up

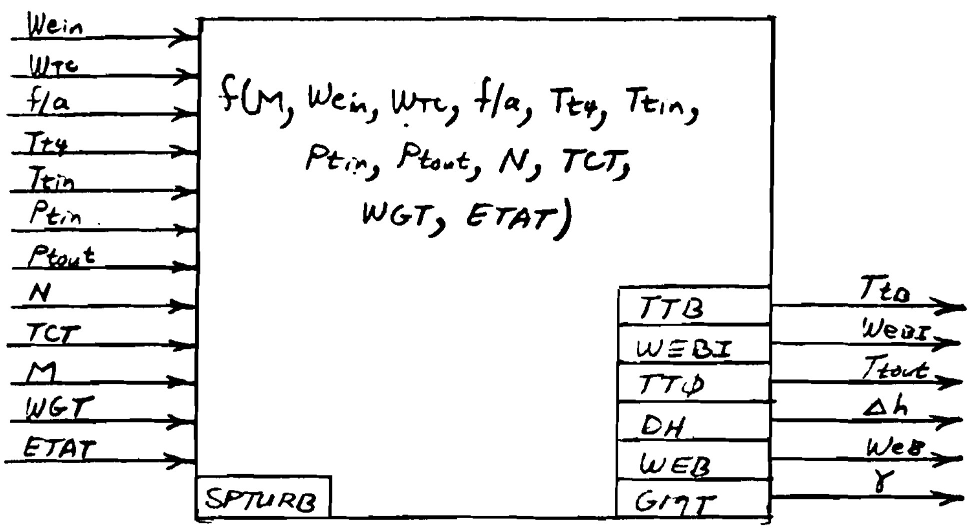
FUNCTION SPTURB



SPTURB 1-01

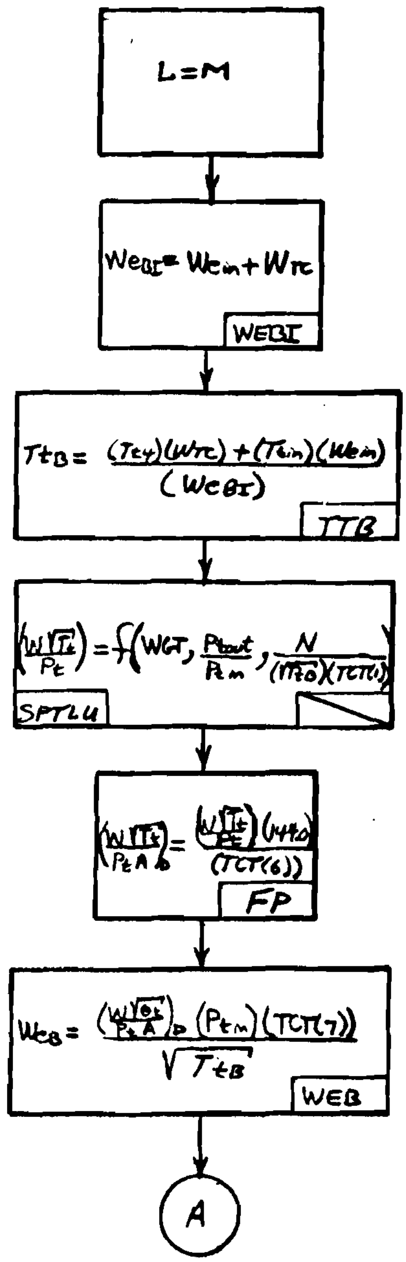
Figure 38. Function SPTURB

FUNCTION SPTURB



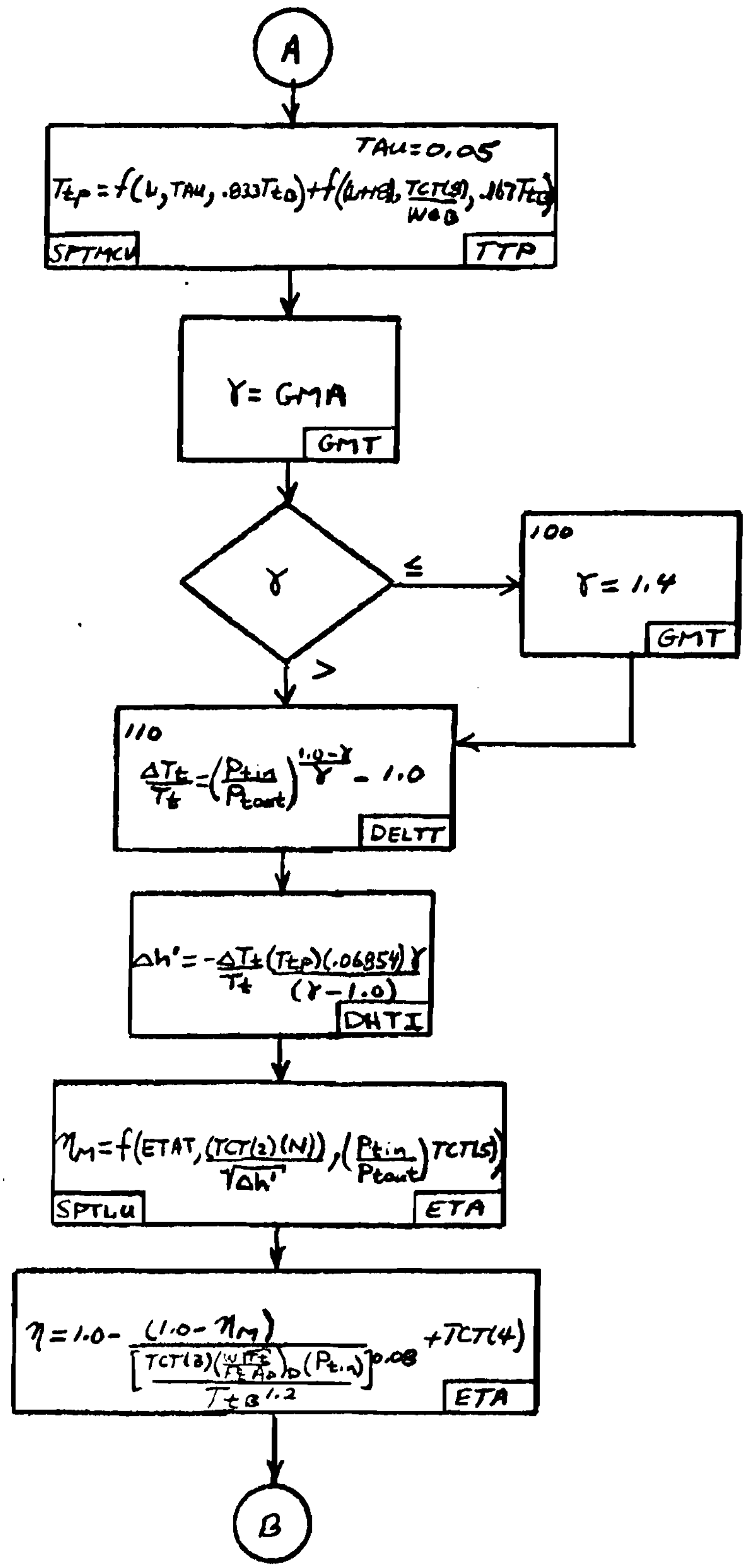
SPTURB 1-02

Figure 38. Function SPTURB (Continued)



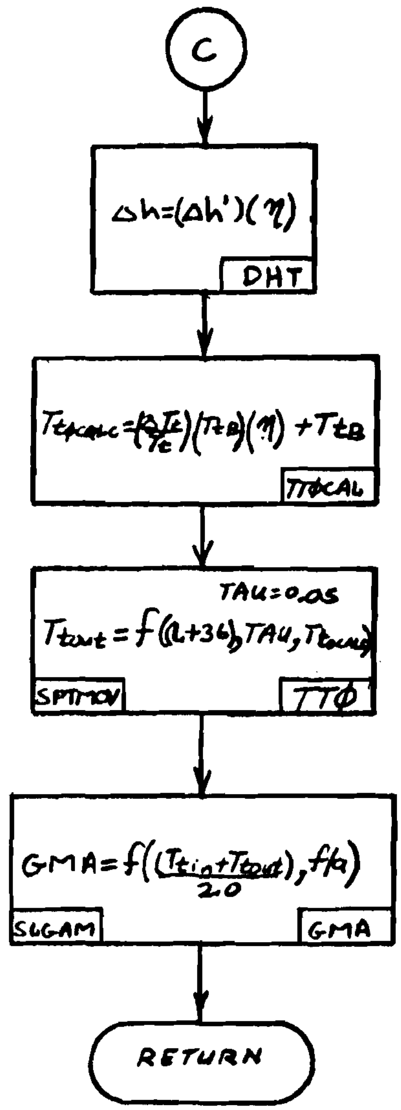
SPTURB 2-01

Figure 38. Function SPTURB (Continued)



SPTURB 3-01

Figure 38. Function SPTURB (Continued)



SPTURB 4-01

Figure 38. Function SPTURB (Concluded)

GENERAL FORM	FUNCTION
X=SARECT(XIC, $\dot{X}$, X_L, X_u, P, XRESET) MODE-CONTROLLED INTEGRATOR USING RECTANGULAR INTEGRATION METHOD	P=1.0 $X_L \le \int_0^t \dot{X}\,dt + X_{IC} \le X_u$ P=0.0 X = XRESECT

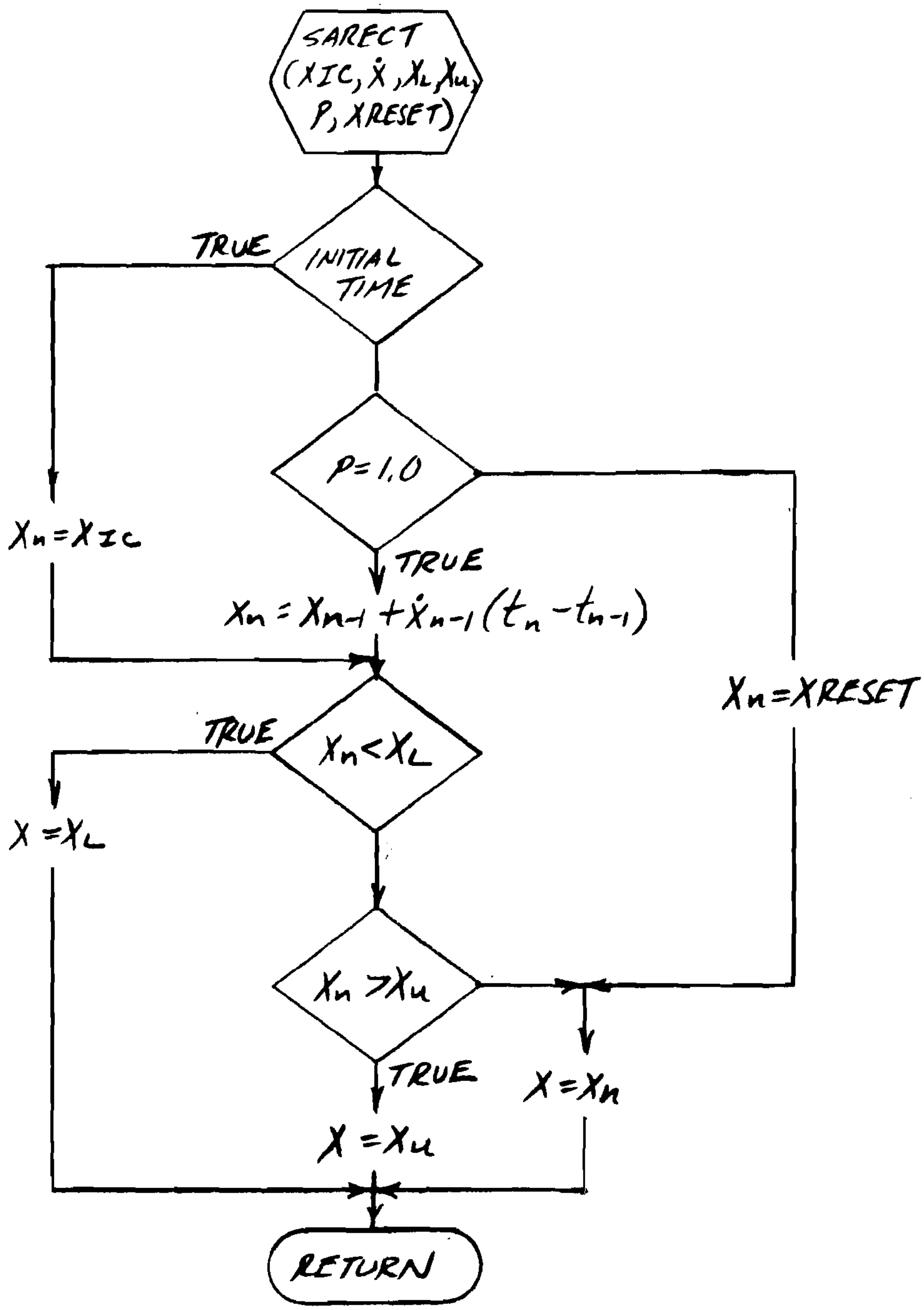


Figure 39. Function SARECT-Mode-Controlled Rectangular Integrator

GENERAL FORM	FUNCTION
$Y = SPTMCV(\tau, X)$ GENERAL TIME CONSTANT FUNCTION	$\tau \dot{Y} + Y = X$ EQUIVALENT LAPLACE TRANSFORM $\frac{1}{\tau S + 1}$

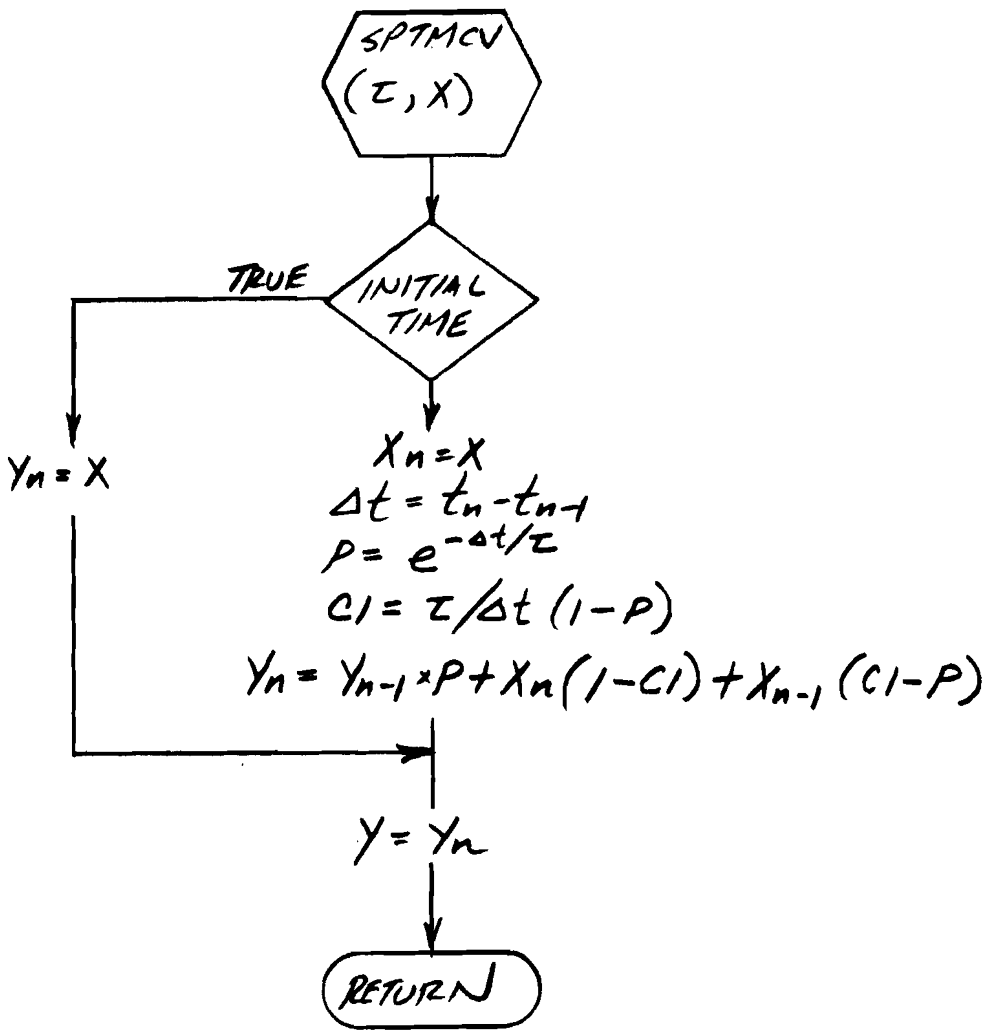


Figure 40. Function SPTMCV-General Time Constant Function

Section V

INITIALIZATION

GENERAL REQUIREMENTS

The simulation of a system experiencing a transient, of necessity begins at a steady-state or quasi-steady-state point. The driving forces, that must be zero to maintain this point, are, in the case of the propulsion system, usually the difference between two large numbers. This effectively means that the initial values which enter into the calculation of the driving forces must be accurate and identical in method of calculation to those within the simulation. The simulation should be stable enough when initialized so that if no transient is introduced, it will maintain its steady-state point.

One method used to initialize a simulation is to provide initial values from steady-state calculations and then run the dynamic program for some time in order to stabilize at the steady-state condition prior to the introduction of the transient. This method is less than satisfactory for several reasons. First, it requires too much preparatory work in that either steady-state programs or hand calculations would have to be executed to provide the initial data. As stated above, accuracy is important which means that the method used for these calculations would have to be compatible to that used in the simulation. Since the simulation logic itself is already available for these calculations its use insures compatibility. Secondly, it does not always work. When initial values are not exact, the start-up transient could drive the simulation into instability.

At the start of this program the ground rule was established that the only initial values to be required were air vehicle Mach number, ambient pressure and temperature, air vehicle angle of attack and yaw, and the position of the power lever. This ground rule is illustrated by the diagram on figure 41.

IMPLEMENTATION

With the ground rule for initialization established several things had to be considered. The most obvious way to insure accuracy and compatibility in computing initial values is to simply use the simulation logic calculations. This would be most efficiently done if sections of the simulation could be executed under the control of an initialization routine. DSL/90 provides a method to do this but unfortunately requires that an invaluable feature of the language be compromised to use it. To explain this anomaly the nature of the DSL/90 system must be lightly touched on.

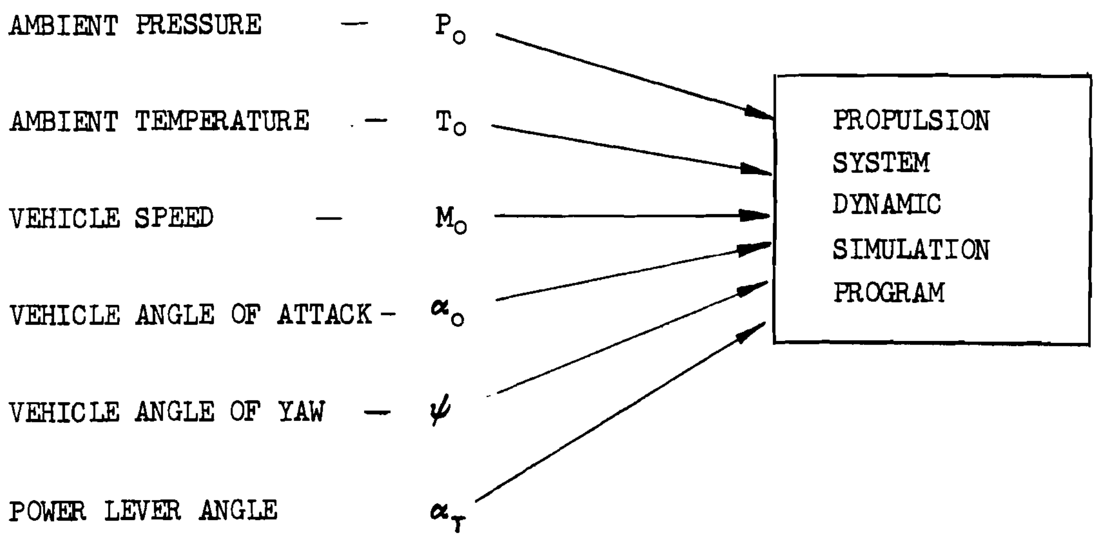


Figure 41. Parameters Required for Initialization

The power of a language such as DSL/90 is its ability to accept the simulation logic equations in any order whatsoever and sort them in such a manner that variables are always available when needed. To accomplish this, the language must prevent the use of logic which transfers the program control from one area in the program to another since after sorting the effect would not likely be the desired one. The language does however, provide a procedure called NØSØRT, which allows the insertion of transfer instructions into the simulation. At first this seems to provide the answer to executing sections of the program under control of an initialization program except that a second look reveals that insertion of a NØSØRT block separates the sorting procedure such that the statements (equations) ahead of the NØSØRT block do not get sorted into the statements after the NØSØRT block. This effectively means that the program must be run through the translation phase which sorts the simulation logic then the original simulation logic rearranged by hand according to the sorted simulation and then the transfer information inserted in NØSØRT blocks in this sorted deck. Since this seems to, at least partially, negate one of the very desirable features of DSL/90 another path seemed advisable.

The basic reason, other than execution time which is negligible, to execute the simulation logic in sections while initializing is the potential for errors which the computer finds unforgiveable such as taking square roots of negative numbers, raising negative numbers to fractional powers, etc. These problems can be overcome in the checkout phase of the initialization by careful choice of initial values of key parameters. In accordance with the ground rule on required inputs stated above, initial values are calculated, not entered as input.

SAMPLE PROGRAM

A sample initialization program used for the initialization of a turbojet engine with a simple integrated propulsion system control and a started inlet phase simulation is presented in Appendix II. This initialization has been checked out only at one power setting at one altitude - Mach condition.

(The reverse of this page is blank)

Appendix I

LIST OF SUPPORTING SUBROUTINES

TABLE I. SAACT

```
$IBFTC SAACT
      SUBROUTINE SAACT (K,XC,X,CCT,XDOT,XIC,XH)
      COMMON/KEYS/NALARM,SKIP(17),KSIM/HMAX/H,KEEP/MEMRY/S(3)
      DIMENSION CCT(1)
      L=K
      XLIM=AMIN1(CCT(1),AMAX1(XC,0.0))
      IF(S(L+2)-KSIM) 100,200,100
  100 IF(KEEP) 110,120,110
  110 S(L+2)=KSIM
      XIC=XLIM
  120 XH=HSTRSS(L,X,CCT(6),CCT(7),X)
      DXDT=CCT(2)*(XLIM-(X*(1.-CCT(5))+CCT(5)*XH))
      XDOT=AMIN1(CCT(4),AMAX1(DXDT,CCT(3)))
      RETURN
      END
```

TABLE II. SADSPA

```
$IBFTC SADSPA                                                    SADSP000
      FUNCTION SADSPA(X,XL,XR)                                   SADSP020
      SADSPA = 0.0                                               SADSP030
      IF(X.LT.XL)SADSPA = -1.0                                   SADSP040
      IF(X.GE.XR)SADSPA =  1.0                                   SADSP050
      RETURN                                                     SADSP060
      END                                                        SADSP070
```

TABLE III. SALIMT

```
$IBFTC SALIMT                                                                           SALIM010
      FUNCTION SALIMT(X,XL,XU,TYPE)                                                     SALIM020
C                                                                                       SALIM025
C     TYPE INDICATES LIMIT PROCEDURE DESIRED                                            SALIM030
C     0 - MIN PRIORITY       1 - MAX PRIORITY       -1 - IGNORE LIMITS                  SALIM040
C                                                                                       SALIM050
      IF(TYPE.LT.0.0)GO TO 400                                                          SALIM060
      IF(XL.GT.XU)GO TO 100                                                             SALIM070
      SALIMT = AMAX1(AMIN1(X,XU),XL)                                                    SALIM080
      RETURN                                                                            SALIM090
  100 IF(TYPE.EQ.0.0)GO TO 300                                                          SALIM100
      SALIMT = AMIN1(X,XU)                                                              SALIM110
      RETURN                                                                            SALIM120
  300 SALIMT = AMAX1(X,XL)                                                              SALIM130
      RETURN                                                                            SALIM140
  400 SALIMT = X                                                                        SALIM150
      RETURN                                                                            SALIM160
      END                                                                               SALIM170
```

TABLE IV. SAMOIN

```
$IBFTC SAMOIN
      FUNCTION SAMOIN (K,DXDT,XIN,XL,XU,PATH,XIC,DXDTM)
      COMMON/CURVAL/TIME                                                TMCV0030
      COMMON/KEYS/NALARM,SKIP(17),KSIM/HMAX/H,KEEP
      COMMON/MEMRY/S(15)
      EQUIVALENCE (KN(1),S(1))
      DIMENSION KN(1)
      I=K                                                               TMCV0040
      L=I                                                               TMCV0050
      XDOT=DXDT
      X=XIN
      TYPE=PATH
      IF(KN(L+14)-KSIM) 100,200,100
  100 IF(KEEP) 110,120,110
  110 KN(L+14)=KSIM
      S(L)=TIME
      S(L+1)=TYPE
      S(L+12)=X
  120 IF(S(L+1)) 960,950,960
  200 DO 300 J=1,5                                                      TMCV0100
      IF(TIME-S(L)) 300,210,310                                         TMCV0110
  210 IF(KEEP) 600,700,600
  300 L=L+2
  310 IF(KEEP) 400,700,400
  400 LL=I+12
      DO 500 JJ=J,5                                                     TMCV0180
      LL=LL-2
      S(LL)=S(LL-2)
  500 S(LL+1)=S(LL-1)
  600 S(L)=TIME                                                         TMCV0230
      S(L+1)=TYPE
  700 IF(TYPE)   900,710,900
  710 IF(S(L+3)) 720,800,720
  720 S(I+12)=XIC
      IF(X-S(I+12)) 730,950,740
  730 XDOT=DXDTM
      GO TO 750
  740 XDOT=-DXDTM
  750 S(I+13)=XDOT
      GO TO 960
  800 IF(X-S(I+12)) 810,950,820
  810 IF(S(I+13)) 840,950,830
  820 IF(S(I+13)) 830,950,840
  830 XDOT=S(I+13)
      GO TO 960
  840 X=S(I+12)
      GO TO 950
  900 IF(XDOT) 910,960,930
  910 IF(X-XL) 920,950,960
  920 X=XL
      GO TO 950
  930 IF(X-XU) 960,950,940
  940 X=XU
  950 XDOT=0.0
      XIN=X
  960 SAMOIN=XDOT
 1000 RETURN
      END
```

TABLE V. SASWCH

```
$IBFTC SASWCH
      FUNCTION SAOFON(I,X,XL,XU)
      COMMON/MEMRY/C(1)
      SAOFON = 1.0
      IF(X.LT.XL.OR.X.LT.XU.AND.C(I).LT.1.0)SAOFON = 0.0
      GO TO 100
      ENTRY SAONOF(I,X,XL,XU)
      SAOFON = 0.0
      IF(X.LT.XL.OR.X.LT.XU.AND.C(I).NE.0.0)SAOFON = 1.0
  100 C(I) = SAOFON
      RETURN
      END
```

TABLE VI. SAWFAT

```
$IBFTC SAWFAT                                                            SAWF0000
C        SUBROUTINE AUTONETICS WF ATTENUATION                            SAWF0015
      INTEGER FUNCTION SAWFAT(TSENS,PRSENS,WSENS,NCORR,PRT,WAT,CN,TACC)  SAWF0020
C                                                                        SAWF0022
C             SIMULATION OF WF/P4 ATTENUATION CONTROL LOOP               SAWF0024
C                                                                        SAWF0026
      REAL NCORR                                                         SAWF0030
      DIMENSION CN(2)                                                    SAWF0040
      SAWFAT = 1                                                         SAWF0050
      IF(TSENS.GE.CN(2))GO TO 200                                        SAWF0060
      IF(NCORR.GE.CN(1))GO TO 100                                        SAWF0070
      IF(PRSENS.GE.SLTLU(PRT,NCORR,TACC))GO TO 200                       SAWF0080
      RETURN                                                             SAWF0085
  100 IF(SLTLU(WAT,NCORR,TACC).GE.WSENS)GO TO 200                        SAWF0090
      RETURN                                                             SAWF0100
  200 SAWFAT = 0                                                         SAWF0110
      RETURN                                                             SAWF0120
      END
```

TABLE VII. SLFVPG

```
$IBFTC SLFVPG                                                                   FVPG0000
C        FLOW PARAMETER AS A FUNCTION OF PRESSURE RATIO AND GAMMA               FVPG0010
         FUNCTION SLFVPG(PRATIO,GAMMA)                                          FVPG0020
C                                                                               FVPG0030
C         G = 32.174049        ,R = 53.34991                                    FVPG0040
         GM = GAMMA                                                             FVPG0050
         PR = PRATIO                                                            FVPG0060
         IF(PR.LE.1.0)GO TO 10                                                  FVPG0070
C                    COMPUTE CRITICAL CONDITIONS FOR MAXIMUM VALUE              FVPG0080
         CTR = 1.0 + (GM - 1.0)/2.0                                             FVPG0090
         TR = PR ** ((GM-1.0)/GM)                                               FVPG0100
         IF(TR.GT.CTR)TR=CTR                                                    FVPG0110
C                    COMPUTE FLOW PARAMETER                                     FVPG0120
         FP = SQRT(1.20615195*GM*(TR-1.)/(GM-1.))/TR**((GM+1.)/2./(GM-1.))      FVPG0130
      20 SLFVPG = FP                                                            FVPG0140
         RETURN                                                                 FVPG0150
      10 FP = 0.0                                                               FVPG0150
         GO TO 20                                                               FVPG0160
         END                                                                    FVPG0170
```

TABLE VIII. SLGAMF

```
$IBFTC SLGAMF
C      GAMMA AS A FUNCTION OF TOTAL TEMP AND FUEL-AIR RATIO
       FUNCTION SLGAM(TT,FARI)
       DIMENSION C(5)
       DATA C/    9.00097E-4,-7.07129E-7,3.41709E-10,-8.10801E-14,
      $           7.27727E-18 /
       T = TT
       FAR = FARI
       CALL ARGQ(N)
       IF(N.LT.2)FAR = 0.0
       IF(T.EQ.0.0)GO TO 1000
       POWER = 5526.0 / T
       CPAIR = .23996+.068558*((POWER/(EXP(POWER)-1.0))**2*EXP(POWER))
       CPFUEL = .12149
       DO 10 I=1,5
    10 CPFUEL = CPFUEL + C(I) * T**(I)
       CPGAS = (CPAIR + FAR * CPFUEL) / (1.0 + FAR)
       RGAS = .068558 + (.000835 * FAR / (1.0 + FAR))
       SLGAM = CPGAS / (CPGAS - RGAS)
       RETURN
  1000 SLGAM = 1.4
       RETURN
       END
```

TABLE IX. SLMVFG

```
$IBFTC SLMVFG
      FUNCTION SLMVFG (LM,XMI,FLOWP,GAMMA)
      COMMON/MEMRY/C(1)
C     MACH NO. AS FUNCTION OF FLOW PARAMETER AND GAMMA
C     G=32.174049
C     R=53.34991
      FP=FLOWP
      IF(FP) 100,100,110
  100 XM=0.0
      GO TO 510
  110 GAM=GAMMA
      C1=(GAM-1.0)/2.0
      C2=(GAM+1.0)/2.0/(GAM-1.0)
      Z=FP/SQRT(GAM*.603075975)
C     COMPUTE MAXIMUM VALUE
      ZMAX =1.0/(1.+C1)**C2
      IF(Z.LT.ZMAX) GO TO 200
      XM=1.0
      GO TO 510
  200 L=LM
      IF(C(L)) 220,220,210
  210 XM=C(L)
      GO TO 320
  220 XM=XMI
      GO TO 320
  300 XM=XM-C3*(Z*C4-XM)/(XMS-1.0)
      IF(XM) 310,310,320
  310 XM=Z
  320 XMS=XM**2
      C3=1.0+C1*XMS
      C4=C3**C2
      ZM=XM/C4
      IF(ABS(Z-ZM).GT. .0000001) GO TO 300
  500 C(L)=XM
  510 SLMVFG=XM
      RETURN
      END
```

TABLE X. SLTLU

```
$IBFTC SLTLU
C      TABLE LOOK-UP CONTROL PROGRAM
       FUNCTION SLTLU (LM,X,Y,W)
       COMMON/CURVAL/A(1)
       COMMON/LCURVE/LOCA(1)
       DATA N/0/
       IPASS=0
       L=LM
       IF(L) 60,60,90
   60  IF(N) 80,70,80
   70  N=LOC(LOCA(1))-LOC(A(1))
       JL=LOCA(1)
       LOCA(1)=LOCA(1)+1+N
       DO  72 J=3,JL
   72  LOCA(J-1)=LOCA(J-1)+LOCA(J-2)
   80  L=-L
       L=LOCA(L)
   90  NX=A(L+1)
       IX=A(L+2)
       IF(A(L)-3.0) 100,270,200
  100  IF(A(L)-1.0) 110,110,120
  110  Z=A(L+1)
       GO TO 400
  120  NY=0
       IY=0
       LX=L+3
       LZ=LX+NX
       GO TO 290
  200  NW=NX
       L1=L+3
       L2=L+1+NW
       L=L2
       NX=A(L+1)
       NY=A(L+3)
       IF(W-A(L1-1)) 280,280,210
  210  DO 230 LW=L1,L2
       IF(W-A(LW)) 240,250,220
  220  L=L+NX+NY+4+NX*NY
       NX=A(L+1)
       NY=A(L+3)
  230  CONTINUE
       GO TO 280
  240  IPASS=1
       RATW=(W-A(LW-1))/(A(LW)-A(LW-1))
       GO TO 280
  250  L=L+NX+NY+4+NX*NY
  260  NX=A(L+1)
  270  NY=A(L+3)
  280  LX=L+5
       LY=LX+NX
```

TABLE X. SLTLU (CONTINUED)

```
      LZ=LY+NY                                                          TLU00340
      IX=A(L+2)                                                         TLU00350
      IY=A(L+4)                                                         TLU00360
  290 IF(IX-2) 300,300,320                                              TLU00370
  300 IF(IY-2) 310,310,320                                              TLU00380
  310 CALL SLTLU2 (A(LX),A(LY),A(LZ),NX,NY,X,Y,Z)                       TLU00390
      GO TO 330                                                         TLU00400
  320 CALL SLTLU3 (A(LX),A(LY),A(LZ),NX,NY,IX,IY,X,Y,Z)                 TLU00410
  330 IF(IPASS) 350,400,340                                             TLU00420
  340 IPASS=-1                                                          TLU00430
      W1=Z                                                              TLU00440
      L=LZ-1+NX*NY                                                      TLU00450
      GO TO 260                                                         TLU00460
  350 Z=W1+RATW*(Z-W1)                                                  TLU00470
  400 SLTLU=Z                                                           TLU00490
  500 RETURN                                                            TLU00500
      END                                                               TLU00510
```

TABLE XI. SLTLU2

```
$IBFTC SLTLU2                                                           LIN30000
C     LINEAR INTERPOLATION FOR THREE DIMENSIONAL TABLE                  LIN30000
      SUBROUTINE SLTLU2 (AX,AY,AZ,NX,NY,X,Y,Z)                          LIN30010
      DIMENSION                                                         LIN30020
     1 AX(1)     ,AY(1)     ,AZ(1)                                      LIN30030
      IF(X-AX(1)) 10,10,20                                              LIN30040
   10 JX=1                                                              LIN30050
      GO TO 40                                                          LIN30060
   20 DO 30 I=2,NX                                                      LIN30070
      JX=I                                                              LIN30080
      IF(X-AX(I)) 50,40,30                                              LIN30090
   30 CONTINUE                                                          LIN30100
   40 RATX=0.0                                                          LIN30110
      GO TO 60                                                          LIN30120
   50 RATX=(AX(JX)-X)/(AX(JX)-AZ(JX-1))                                 LIN30130
   60 IF(NY) 70,70,80                                                   LIN30132
   70 Z=AZ(JX)-RATX*(AZ(JX)-AZ(JX-1))                                   LIN30132
      GO TO 200                                                         LIN30134
   80 IF(Y-AY(1)) 90,90,100                                             LIN30140
   90 JY=1                                                              LIN30150
      GO TO 120                                                         LIN30160
  100 DO 110 J=2,NY                                                     LIN30170
      JY=J                                                              LIN30180
      IF(Y-AY(J)) 130,120,110                                           LIN30190
  110 CONTINUE                                                          LIN30200
  120 RATY=0.0                                                          LIN30210
      GO TO 140                                                         LIN30220
  130 RATY=(AY(JY)-Y)/(AY(JY)-AY(JY-1))                                 LIN30230
  140 JZ=JX+NX*(JY-1)                                                   LIN30240
      Z2=AZ(JZ)-RATX*(AZ(JZ)-AZ(JZ-1))                                  LIN30250
      JZ=JZ-NX                                                          LIN30260
      Z1=AZ(JZ)-RATX*(AZ(JZ)-AZ(JZ-1))                                  LIN30270
      Z=Z2-RATY*(Z2-Z1)                                                 LIN30280
  200 RETURN                                                            LIN30290
      END                                                               LIN30300
```

TABLE XII. SLTLU3

```
$IBFTC SLTLU3                                                          LAG30000
C     LAGRANGE INTERPOLATION FORMULA FOR THREE DIMENSIONAL TABLE       LAG30000
      SUBROUTINE SLTLU3 (AX,AY,AZ,NX,NY,IX,IY,X,Y,Z)                   LAG30010
      DIMENSION                                                        LAG30020
     1 AX(1)     ,AY(1)     ,AZ(1)     ,YY(10)    ,C(10)               LAG30030
      CALL SLTLU4(AX,NX,IX,X,N1,N2)                                    LAG30040
      CALL SLTLU4(AY,NY,IY,Y,M1,M2)                                    LAG30050
      IF(N2) 50,10,50                                                  LAG30060
   10 IF(M2) 30,20,30                                                  LAG30070
   20 JZ=N1+NX*(M1-1)                                                  LAG30080
      Z=AZ(JZ)                                                         LAG30090
      GO TO 200                                                        LAG30100
   30 JY=N1+NX*(M1-2)                                                  LAG30110
      L=0                                                              LAG30112
      DO 40 J=M1,M2                                                    LAG30120
      L=L+1                                                            LAG30122
      JY=JY+NX                                                         LAG30130
      YY(L)=AZ(JY)                                                     LAG30140
   40 CONTINUF                                                         LAG30150
      GO TO 130                                                        LAG30160
   50 P=1.0                                                            LAG30170
      K=0                                                              LAG30172
      DO 80 J=N1,N2                                                    LAG30180
      K=K+1                                                            LAG30182
      C(K)=1.0                                                         LAG30190
      P=P*(X-AX(J))                                                    LAG30200
      DO 80 I=N1,N2                                                    LAG30210
      IF(I-J)  70,80,70                                                LAG30220
   70 C(K)=C(K)/(AX(J)-AX(I))                                          LAG30230
   80 CONTINUE                                                         LAG30240
      IF(M2) 100,90,100                                                LAG30250
   90 M2=M1                                                            LAG30260
  100 L=0                                                              LAG30270
      DO 110 I=M1,M2                                                   LAG30272
      L=L+1                                                            LAG30274
      YY(L)=0.0                                                        LAG30280
      JZ=N1-1+NX*(I-1)                                                 LAG30290
      K=0                                                              LAG30292
      DO 110 M=N1,N2                                                   LAG30300
      K=K+1                                                            LAG30302
      JZ=JZ+1                                                          LAG30310
      YY(L)=YY(L)+P*AZ(JZ)/(X-AX(J))*C(K)                              LAG30320
  110 CONTINUF                                                         LAG30330
      IF(M1-M2) 130,120,130                                            LAG30330
  120 Z=YY(1)                                                          LAG30340
      GO TO 200                                                        LAG30350
  130 P=1.0                                                            LAG30360
      L=0                                                              LAG30362
      DO 150 J=M1,M2                                                   LAG30370
      L=L+1                                                            LAG30372
      C(L)=1.0                                                         LAG30380
      P=P*(Y-AY(J))                                                    LAG30390
      DO 150 I=M1,M2                                                   LAG30400
      IF(I-J) 140,150,140                                              LAG30410
  140 C(L)=C(L)/(AY(J)-AY(I))                                          LAG30420
  150 CONTINUE                                                         LAG30430
```

TABLE XII. SLTLU3 (CONTINUED)

```
      Z=0.0                                                             LAG30440
      L=0                                                               LAG30442
      DO 160 J=M1,M2                                                    LAG30450
      L=L+1                                                             LAG30452
      Z=Z+P*YY(L)/(Y-AY(J))*C(L)                                        LAG30460
  160 CONTINUE                                                          LAG30470
  200 RETURN                                                            LAG30480
      END                                                               LAG30490
```

TABLE XIII. SLTLU4

```
$IBFTC SLTLU4
C     LOCATE RANGE OF POINTS FOR LAGRANGE INTERPOLATION                 LAGRG000
      SUBROUTINE SLTLU4 (AX,N,L,X,N1,N2)                                LAGRG000
      DIMENSION                                                         LAGRG010
     1 AX(1)                                                            LAGRG012
      N2=0                                                              LAGRG014
      IF(N-L) 220,220,10                                                LAGRG020
   10 IF(X-AX(1)) 30,30,40                                              LAGRG030
   30 N1=1                                                              LAGRG040
      GO TO 300                                                         LAGRG080
   40 DO 210 J=2,N                                                      LAGRG090
      IF(X-AX(J)) 60,50,210                                             LAGRG100
   50 N1=J                                                              LAGRG110
      GO TO 300                                                         LAGRG120
   60 JJ=L-1                                                            LAGRG130
      K1=J-JJ                                                           LAGRG140
      IF(K1) 70,70,80                                                   LAGRG150
   70 K1=1                                                              LAGRG160
      GO TO 100                                                         LAGRG170
   80 K3=J+L-2                                                          LAGRG180
      IF(K3-N) 100,100,90                                               LAGRG190
   90 K2=N-JJ                                                           LAGRG200
      GO TO 110                                                         LAGRG210
  100 K2=J-1                                                            LAGRG220
  110 RB=10000.                                                         LAGRG230
      DO 190 K=K1,K2                                                    LAGRG240
      KK=K+JJ                                                           LAGRG250
      C1=X-AX(K)                                                        LAGRG260
      C2=AX(KK)-X                                                       LAGRG270
      IF(C1-C2) 140,120,130                                             LAGRG280
  120 N2=KK                                                             LAGRG290
      GO TO 200                                                         LAGRG300
  130 RA=1.0-C2/C1                                                      LAGRG310
      GO TO 150                                                         LAGRG320
  140 RA=1.0-C1/C2                                                      LAGRG330
  150 IF(RB-RA) 190,160,170                                             LAGRG340
  160 IF(J-L/2-K) 190,180,180                                           LAGRG350
  170 RB=RA                                                             LAGRG360
  180 N2=KK                                                             LAGRG370
  190 CONTINUE                                                          LAGRG380
  200 N1=N2-JJ                                                          LAGRG390
      GO TO 300                                                         LAGRG400
  210 CONTINUE                                                          LAGRG410
      N1=N                                                              LAGRG420
      GO TO 300                                                         LAGRG440
  220 N2=N                                                              LAGRG450
      N1=1                                                              LAGRG460
  300 RETURN                                                            LAGRG470
      END                                                               LAGRG480
                                                                        LAGRG490
```

TABLE XIV. SPCOMP

```
$IBFTC SPCOMP                                                           COMP0000
      SUBROUTINE SPCOMP(M,PTI,PTO,TTI,N,CCT,WAT,ETAT,PRST,WAST,REYT,SRGMCOMP0010
     1,ETA,TTO,DH,W,GMA)                                                COMP0020
      COMMON/CURVAL/TIME                                                COMP0030
      COMMON/KEYS/NALARM,SKIP(17),KSIM/HMAX/H,KEEP/MEMRY/S(40)          COMP0040
      EQUIVALENCE (KN(1),S(1))                                          COMP0042
      REAL N,NCN                                                        COMP0050
      DIMENSION CCT(1),KN(1)                                            COMP0060
      L=M                                                               COMP0070
      DE=PTI/14.696                                                     COMP0080
      RTH=SQRT(TTI/518.69)                                              COMP0090
      NCN=N/(RTH*CCT(1))                                                COMP0100
      PRATIO=PTO/PTI                                                    COMP0110
      PR=CCT(2)*(PRATIO-1.0)+1.0                                        COMP0120
      WCN=SPTLU(WAT,NCN,PR)                                             COMP0130
      IF(KN(L+39)-KSIM) 120,100,120                                     COMP0140
  100 SRGM=S(L+38)                                                      COMP0160
      IF(SRGM-1.0-DSRGM) 110,130,130                                    COMP0170
  110 DETS=CCT(8)                                                       COMP0180
      DSRGM=CCT(9)                                                      COMP0190
      DWCN=SPTMCV(L,0.05,CCT(5)*WCN)                                    COMP0200
      GO TO 200                                                         COMP0210
  120 KN(L+39)=KSIM                                                     COMP0212
  130 DETS=1.0                                                          COMP0220
      DSRGM=0.0                                                         COMP0230
      DWCN=0.0                                                          COMP0240
      KN(L+18)=KSIM                                                     COMP0242
      S(L)=TIME                                                         COMP0250
      S(L+1)=0.0                                                        COMP0260
      S(L+2)=0.0                                                        COMP0270
  200 S(L+38)=WCN*(((SPTLU(PRST,NCN)-1.0)/CCT(2))+1.0)/PRATIO/(SPTLU(WASCOMP0280
     1T,NCN))                                                           COMP0290
      WC=CCT(6)*(WCN+DWCN)                                              COMP0300
      DET=SPTLU(REYT,(CCT(7)*WC))-SPTLU(REYT,(CCT(7)*WC*DE/(RTH**2.8))) COMP0310
      W=WC*(1.0+DET)*DE/RTH                                             COMP0320
      ETA=DETS*(CCT(3)*(SPTLU(ETAT,WCN,PR)+CCT(4))+DET)                 COMP0330
      GM=GMA                                                            COMP0340
      IF(GM) 300,300,310                                                COMP0350
  300 GM=1.4                                                            COMP0360
  310 DT=(PRATIO**((GM-1.0)/GM)-1.0)*TTI/ETA                            COMP0370
      DH=DT*0.06854*GM/(GM-1.0)                                         COMP0380
      TTOCAL=TTI+DT                                                     COMP0390
      TTO=SPTMCV(L+19,0.05,TTOCAL)                                      COMP0400
      GMA = SLGAM((TTI+TTO)/2.0)                                        COMP0410
      RETURN                                                            COMP0420
      END                                                               COMP0430
```

TABLE XV. SPMEMF

```
$IBFTC SPMEMF (K,XIC,X)                                               MEMF0000
      COMMON/CURVAL/TIME                                              MEMF0010
      COMMON/KEYS/NALARM,SKIP(17),KSIM/HMAX/H,KEEP/MEMRY/S(13)        MEMF0020
      EQUIVALENCE (KN(1),S(1))                                        MEMF0022
      DIMENSION KN(1)                                                 MEMF0023
      I=K                                                             MEMF0030
      L=I                                                             MEMF0040
      IF(KN(L+12)-KSIM) 100,200,100                                   MEMF0050
  100 IF(KEEP) 110,120,110                                            MEMF0060
  100 KN(L+12)=KSIM                                                   MEMF0070
  120 OUTPUT=XIC                                                      MEMF0100
      IF(KEEP) 800,900,800                                            MEMF0110
  200 DO 220 J=1,5                                                    MEMF0130
      IF(TIME-S(L)) 220,210,300                                       MEMF0140
  210 OUTPUT=S(L+3)                                                   MEMF0150
      IF(KEEP) 800,900,800                                            MEMF0160
  220 L=L+2                                                           MEMF0180
  300 OUTPUT=S(L+1)                                                   MEMF0190
  700 IF(KEEP) 800,900,720                                            MEMF0200
  720 LL=I+12                                                         MEMF0210
      DO 730 JJ=1,5                                                   MEMF0220
      LL=LL-2                                                         MEMF0230
      S(LL+1)=S(LL-2)                                                 MEMF0240
      S(LL+2)=S(LL-1)                                                 MEMF0250
  730 S(LL+3)=S(LL)                                                   MEMF0260
  800 S(L)=TIME                                                       MEMF0270
      S(L+1)=X                                                        MEMF0280
  900 SPMEMF=OUTPUT                                                   MEMF0290
      RETURN                                                          MEMF0300
      END                                                             MEMF0310
```

TABLE XVI. SPTACL

```
$IBFTC SPTACL                                                            SPTAC010
      FUNCTION SPTACL(L,SRGM,TT4,PT4,WE4,TT5,TT5B,TT5S,TT5MAX,WFE,WFS,   SPTAC020
     $               ETAB,CAC1,DTVFAT)                                    SPTAC030
      WFS = WFE                                                          SPTAC040
      IF(WFS.EQ.0.0)RETURN                                               SPTAC045
      N=0
      TT5S = AMIN1  (((SRGM/CAC1)**2-1.0)*TT5B   + TT5  ,TT5MAX )        SPTAC050
  100 TT5T = SPTLU(DTVFAT,WFS/WE4,TT4)*ETAB + TT4                        SPTAC060
      DT5S = TT5S - TT5T                                                 SPTAC070
      IF(ABS(DT5S).LE.0.10)GO TO 102                                     SPTAC080
      N=N+1
      IF(N.GT.20)WRITE(6,101)N,TT5S,TT5T,DT5S,WFS,WFE
  101 FORMAT(12H  SPTACL, N=,I3, 2X,5HTT5S=,F10.3, 2X,5HTT5T=,F10.3
     $        ,2X,5HDT5S=,F10.5, 2X,4HWFS= ,F10.5, 2X,4HWFE= ,F10.5)
      IF(N.GT.25)GO TO 102
      WFS = WFS*(1.0+(DT5S/1500.0))                                      SPTAC090
      GO TO 100                                                          SPTAC100
  102 SPTACL = SPTMCV(L,0.005,WFS/PT4)                                   SPTAC110
  105 RETURN                                                             SPTAC120
      END                                                                SPTAC130
```

TABLE XVII. SPTLU

```
$IBFTC SPTLU
      FUNCTION SPTLU(LIN,XIN,YIN)
C     TABLE LOOK-UP FOR PRATT AND WHITNEY TABLE (CONTROL)
      COMMON/CURVAL/C(1)
      COMMON/PCURVE/LOCA(1)
      DIMENSION  KC(1)
      EQUIVALENCE (C,KC)
      DATA N/0/
      L=LIN
      X=XIN
      Y=YIN
      IF(L) 100,100,140
  100 IF(N) 130,110,130
  110 N=LOC(LOCA(1))-LOC(C(1))
      JL=LOCA(1)
      LOCA(1)=LOCA(1)+1+N
      DO 120 J=3,JL
  120 LOCA(J-1)=LOCA(J-1)+LOCA(J-2)
  130 L=-L
      L=LOCA(L)
  140 IF(ABS(C(L))-1.0) 200,300,400
  200 IF(X-C(L+3)) 210,220,220
  210 L1=L+1
      GO TO 230
  220 L1=L+2
  230 L2=KC(L1)+L-1
      ANS=SPTLU1(L+4,ALOG(X)-Y)*SPTLU1(L2,X)
      GO TO 500
  300 DPR=Y-SPTLU1(L+4,X)
      IF(DPR) 310,320,320
  310 DPR=-DPR
      L1=L+2
      GO TO 330
  320 L1=L+3
  330 L2=KC(L1)+L-1
      L3=KC(L+1)+L-1
      ANS=SPTLU1(L3,X)+C(L)*SPTLU1(L2,DPR,X)
      GO TO 500
  400 ANS=SPTLU1(L,X,Y)
  500 SPTLU=ANS
      RETURN
      END
```

TABLE XVIII. SPTLU1

```
$IBFTC SPTLU1                                                          PTL10000
      FUNCTION SPTLU1(LOC,XIN,YIN)                                     PTL10010
      COMMON/CURVAL/C(1)                                               PTL10020
      DIMENSION XY(2)                                                  PTL10030
      L=LOC                                                            PTL10040
      X=XIN                                                            PTL10050
      IF(X-C(L+1)) 110,110,120                                         PTL10060
C        X VALUE LESS THAN X MINIMUM                                   PTL10070
  110 X=C(L+1)                                                         PTL10080
      GO TO 140                                                        PTL10090
  120 IF(X-C(L+2)) 140,130,130                                         PTL10100
C        X VALUE GREATER X MAXIMUM                                     PTL10110
  130 X=C(L+2)                                                         PTL10120
  140 N1=(X-C(L+1))/C(L+3)                                             PTL10130
      C1=N1                                                            PTL10140
      RATX=(X-C(L+1))/C(L+3)-C1                                        PTL10150
  200 IF(C(L)-2.0) 210,210,220                                         PTL10160
  210 LN=L+N1+4                                                        PTL10170
      ANS=C(LN)+RATX*(C(LN+1)-C(LN))                                   PTL10180
      GO TO 300                                                        PTL10190
C        THREE DIMENSIONAL TABLE SECTION                               PTL10200
  220 Y=YIN                                                            PTL10210
      M=(C(L+5)-C(L+4))/C(L+6)+1.01                                    PTL10220
      IF(Y-C(L+4)) 230,230,240                                         PTL10230
C        Y VALUE LESS THAN Y MINIMUM                                   PTL10240
  230 Y=C(L+4)                                                         PTL10250
      GO TO 260                                                        PTL10260
  240 IF(C(L+5)-Y) 250,250,260                                         PTL10270
C        Y VALUE GREATER THAN Y MAXIMUM                                PTL10280
  250 Y=C(L+5)                                                         PTL10290
  260 M1=(Y-C(L+4))/C(L+6)                                             PTL10300
      D1=M1                                                            PTL10310
      RATY=(Y-C(L+4))/C(L+6)-D1                                        PTL10320
      N=(C(L+2)-C(L+1))/C(L+3)+1.01                                    PTL10330
      LN=L+7+(N*M1)+N1                                                 PTL10340
      DO 280 K=1,2                                                     PTL10350
      XY(K)=C(LN)+RATX*(C(LN+1)-C(LN))                                 PTL10360
      IF(RATY) 280,270,280                                             PTL10370
  270 ANS=XY(1)                                                        PTL10380
      GO TO 300                                                        PTL10390
  280 LN=LN+N                                                          PTL10400
      ANS=XY(1)+RATY*(XY(2)-XY(1))                                     PTL10410
  300 SPTLU1=ANS                                                       PTL10420
      RETURN                                                           PTL10430
      END                                                              PTL10440
```

TABLE XIX. SPTURB

```
$IBFTC SPTURB
      SUBROUTINE SPTURB(M,WE,WTC,FA,TT4,TTI,PTI,PTO,N,TCT,WGT,ETAT,TTB,
     1 WEBI,TTO,DHT,WEB,GMA)
      REAL N
      DIMENSION TCT(1)
      L=M
      WEBI=WE+WTC
      TTB=(TT4*WTC+TTI*WE)/WEBI
      PRATIO=PTI/PTO
      FP=SPTLU(WGT,1./PRATIO,(N/SQRT(TTB)/TCT(1)))/TCT(6)*144.
      WEB=FP*PTI*TCT(7)/SQRT(TTB)
      TTP=SPTMCV(L,0.05,0.833*TTB)+SPTMCV(L+19,TCT(8)/WEB,0.167*TTB)
      GMT=GMA
      IF(GMT) 100,100,110
  100 GMT=1.4
  110 DELTT=PRATIO**((1.0-GMT)/GMT)-1.0
      DHTI=-DELTT*TTP*0.06854*GMT/(GMT-1.0)
      ETA=SPTLU(ETAT,(TCT(2)*N/SQRT(DHTI)),PRATIO*TCT(5))
      ETA=1.0-(1.0-ETA)/(TCT(3)*FP*PTI/(TTB**1.2))**0.08+TCT(4)
      DHT=DHTI*ETA
      TTOCAL=DELTT*TTB*ETA+TTB
      TTO=SPTMCV(L+38,0.05,TTOCAL)
      GAM=SLGAM((TTI+TTO)/2.0,FA)
      RETURN
      END
```

TABLE XX. SARECT

```
$IBFTC SARECT
      FUNCTION SARECT (I,XIC,DXDT,XL,XU,PATH,XRESET)
      DIMENSION KN(1)
      COMMON/CURVAL/TIME
      COMMON/KEYS/NALARM,SKIP(17),KSIM/HMAX/H,KEEP /MEMRY/S(14)
      EQUIVALENCE (KN(1),S(1))
      L=I
      IF(KN(I+13).NE.KSIM)GO TO 300
      DO 220 J=1,3
      IF(TIME-S(L)) 220,230,600
  220 L=L+4
  230 SARECT=S(L+3)
      RETURN
  300 IF(KEEP.EQ.0)GO TO 320
      KN(I+13)=KSIM
  320 CALL ARGQ(KN(I+12))
      IF(KN(I+12).EQ.7.AND.PATH.LE.0.0)GO TO 330
      SARECT=XIC
      IF(KN(I+12).LT.5)GO TO 340
      IF(SARECT.GT.XU)SARECT=XU
      IF(SARECT.LT.XL)SARECT=XL
      GO TO 340
  330 SARECT=XRESET
  340 IF(KEEP.NE.0)GO TO 890
      RETURN
  600 IF(KN(I+12).LT.7.OR.PATH.NE.0.0)GO TO 800
      DXDT=0.0
      SARECT=XRESET
      GO TO 850
  800 SARECT=S(L+3)+S(L+2)*(TIME-S(L))
      IF(KN(I+12).EQ.3.OR.S(L+2).EQ.0.0)GO TO 850
      IF(S(L+2).GT.0.0.AND.SARECT.GT.XU)SARECT=XU
      IF(S(L+2).LT.0.0.AND.SARECT.LT.XL)SARECT=XL
  850 IF(KEEP.EQ.0)RETURN
      IF(KEEP.LT.0.AND.J.GE.3)GO TO 890
      LL=I+12
      DO 880 JJ=J,3
      LL=LL-4
      S(LL)=S(LL-4)
      S(LL+1)=S(LL-3)
      S(LL+2)=S(LL-2)
  880 S(LL+3)=S(LL-1)
  890 S(L)=TIME
      S(L+1)=PATH
      S(L+2)=DXDT
      S(L+3)=SARECT
      RETURN
      END
```

TABLE XXI. SPTMCV

```
$IBFTC SPTMCV
      FUNCTION SPTMCV(K,TAU,TT)
      COMMON/CURVAL/TIME
      COMMON/KEYS/NALARM,SKIP(17),KSIM/HMAX/H,KEEP/MEMRY/S(10)
      EQUIVALENCE (KN(1),S(1))
      DIMENSION KN(1)
      I=K
      L=I
      IF(KN(L+9)-KSIM) 100,200,100
  100 IF(KEEP) 110,120,110
  110 KN(L+9)=KSIM
  120 OUTPUT=TT
      IF(KEEP) 800,900,800
  200 DO 220 J=1,3
      IF(TIME-S(L)) 220,230,300
  220 L=L+3
  230 OUTPUT=S(L+2)
      GO TO 900
  300 DT=TIME-S(L)
      P=EXP(-DT/TAU)
      C1=TAU/DT*(1.-P)
      OUTPUT=S(L+2)*P+TT*(1.-C1)+S(L+1)*(C1-P)
  400 IF(KEEP) 410,900,420
  410 IF(J-3) 420,800,800
  420 LL=I+9
      DO 430 JJ=J,2
      LL=LL-3
      S(LL)=S(LL-3)
      S(LL+1)=S(LL-2)
  430 S(LL+2)=S(LL-1)
  800 S(L)=TIME
      S(L+1)=TT
      S(L+2)=OUTPUT
  900 SPTMCV=OUTPUT
      RETURN
      END
```

Appendix II

INITIALIZATION PROGRAM

Appendix II

INITIALIZATION PROGRAM

In keeping with the ground rules established for the propulsion system simulation an initialization or initial conditions (I/C) program was developed. Certain elements of the I/C program are general and the example shown in this appendix will suffice to illustrate the general form of such programs. Specific I/C programs must be tailored for the specific system being initialized.

Figure 1 shows the FORTRAN flow diagram for the STEADY program which acts as the controlling element in the I/C phase. The MOVE subroutine referred to in this diagram is a routine which locates the simulation program variables by name in the CURVAL storage areas and places an identifying subscript in the STEADY routine so reference to and from UPDATE, the DSL/90 created routine which actually contains the simulation logic, can be made. The MOVE routine is diagrammed on figure 2. The STEADY program computes initial values for engine and inlet parameters that must have values in order to allow UPDATE to be executed. After execution of the simulation logic (UPDATE) the calculated values are used to compute initial values for several inlet variables which are required for use in SLMASS which computes inlet initial conditions. Figure 3 shows the flow diagram for SLMASS.

After another pass through the UPDATE routine to establish engine face conditions, the SPJENG routine, diagrammed in figure 4, is called to compute the initial conditions which will bring the engine to a balanced condition. After balancing the engine control is returned to STEADY where final calculations are made and UPDATE is called several times to set all initial values. The repeated calling of the UPDATE routine is necessary due to the nature of certain DSL/90 functions such as HSTRSS which must be entered twice before the value appears as an output.

The program listings for the STEADY, MØVE, SLMASS and SPJENG are given in tables I, II, III, and IV. The print statements sprinkled through all three routines are meant for checkout only and in the final version will be removed.

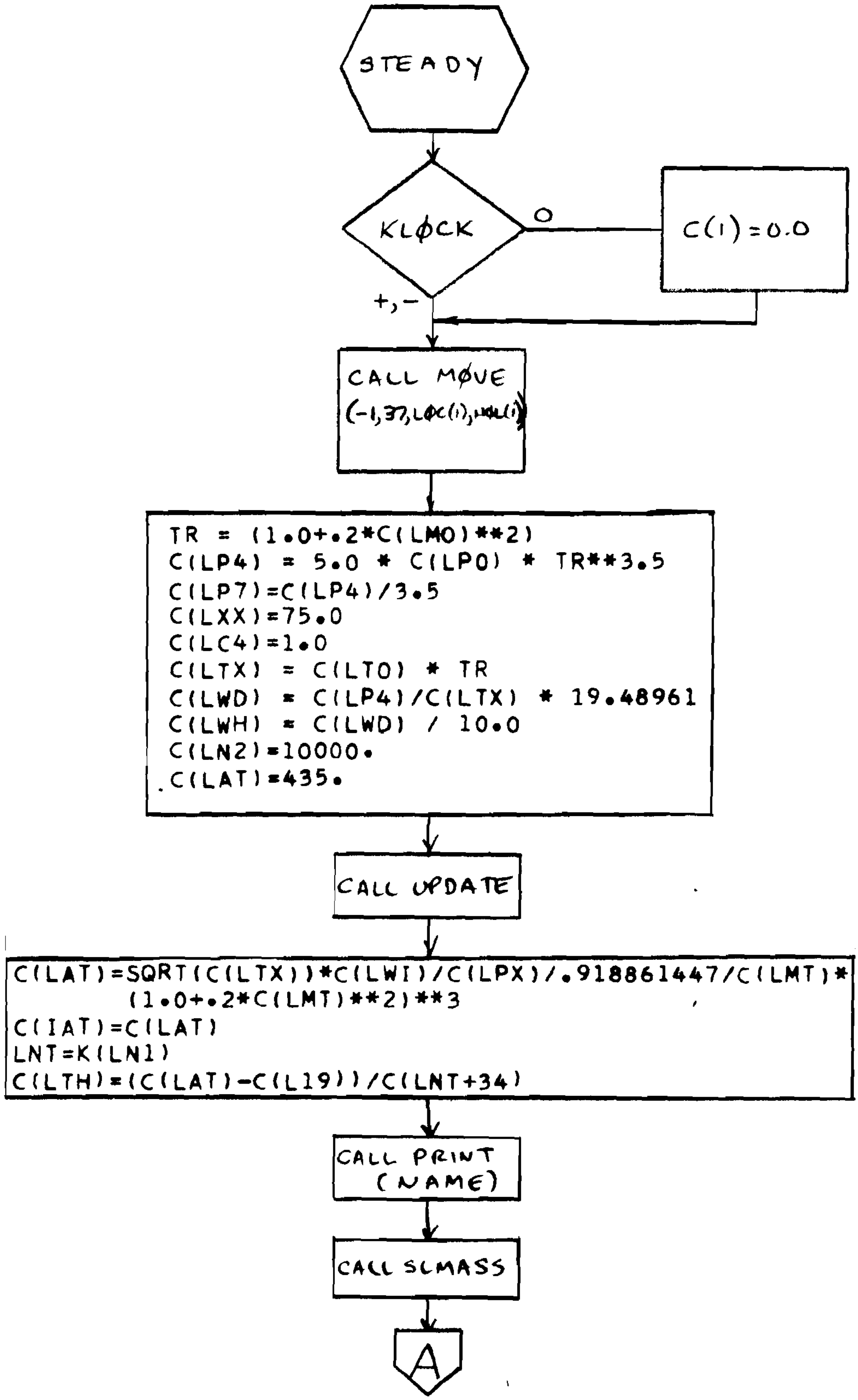


Figure 1. STEADY Routine

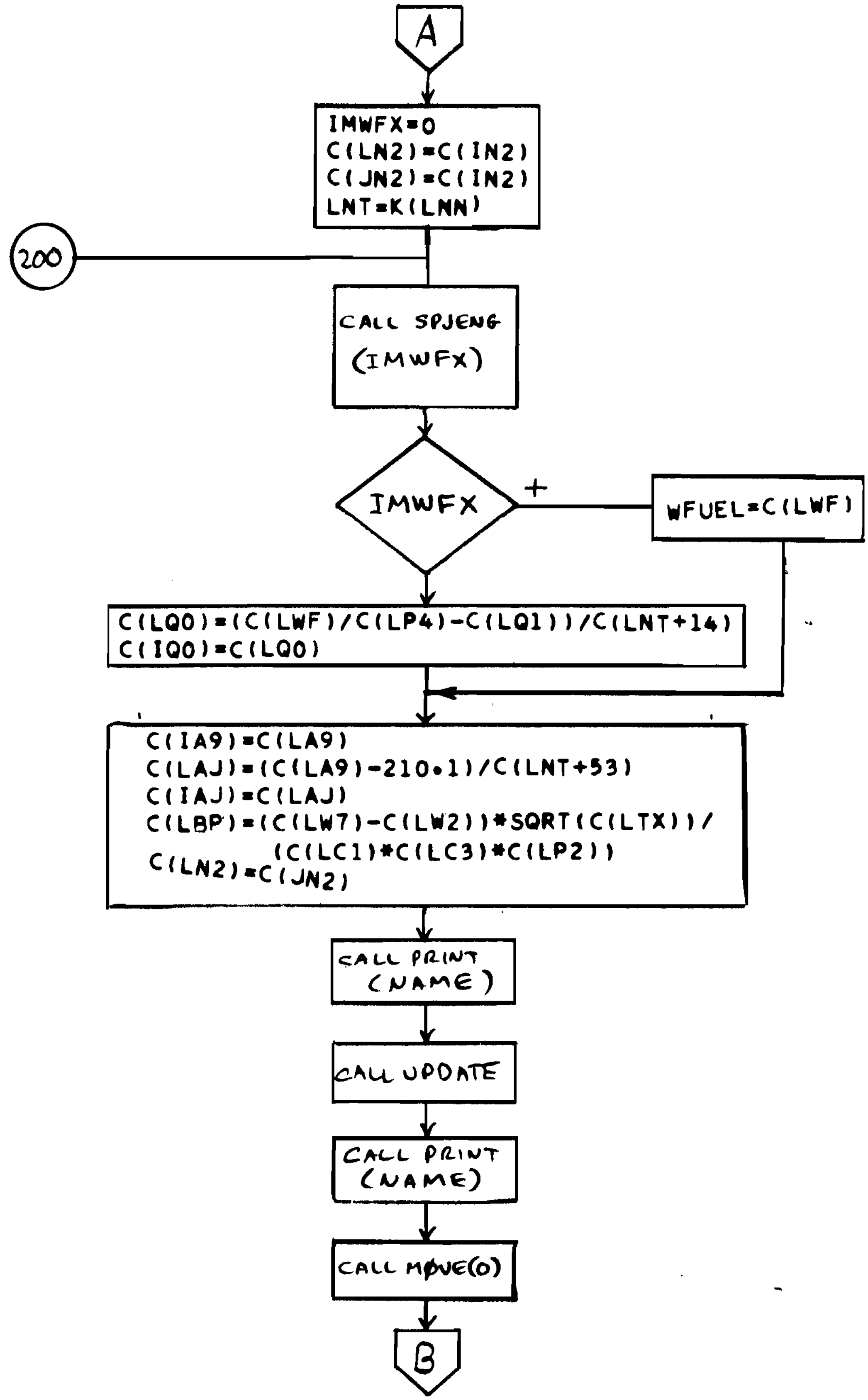


Figure 1. STEADY Routine (Continued)

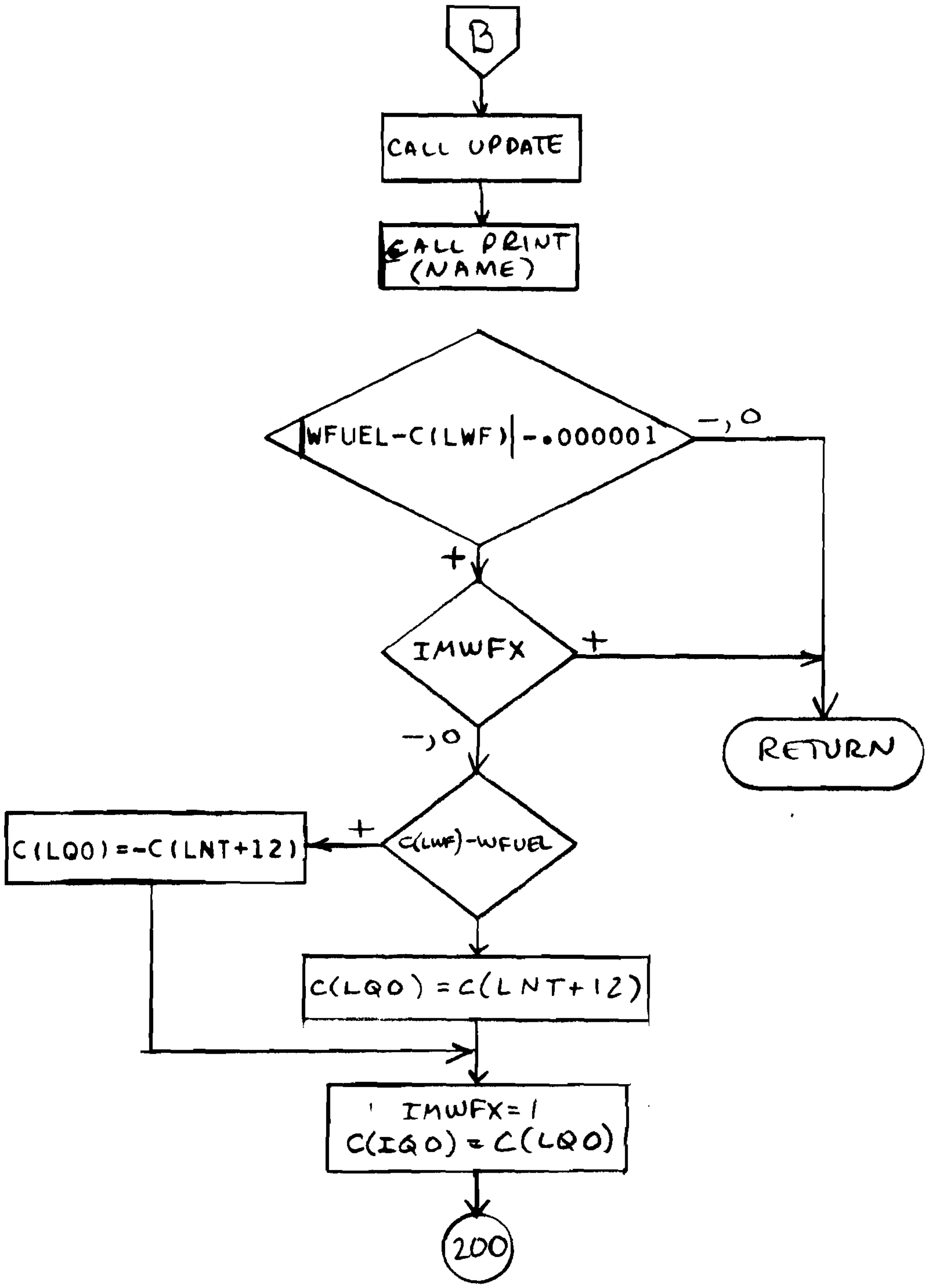


Figure 1. STEADY Routine (Concluded)

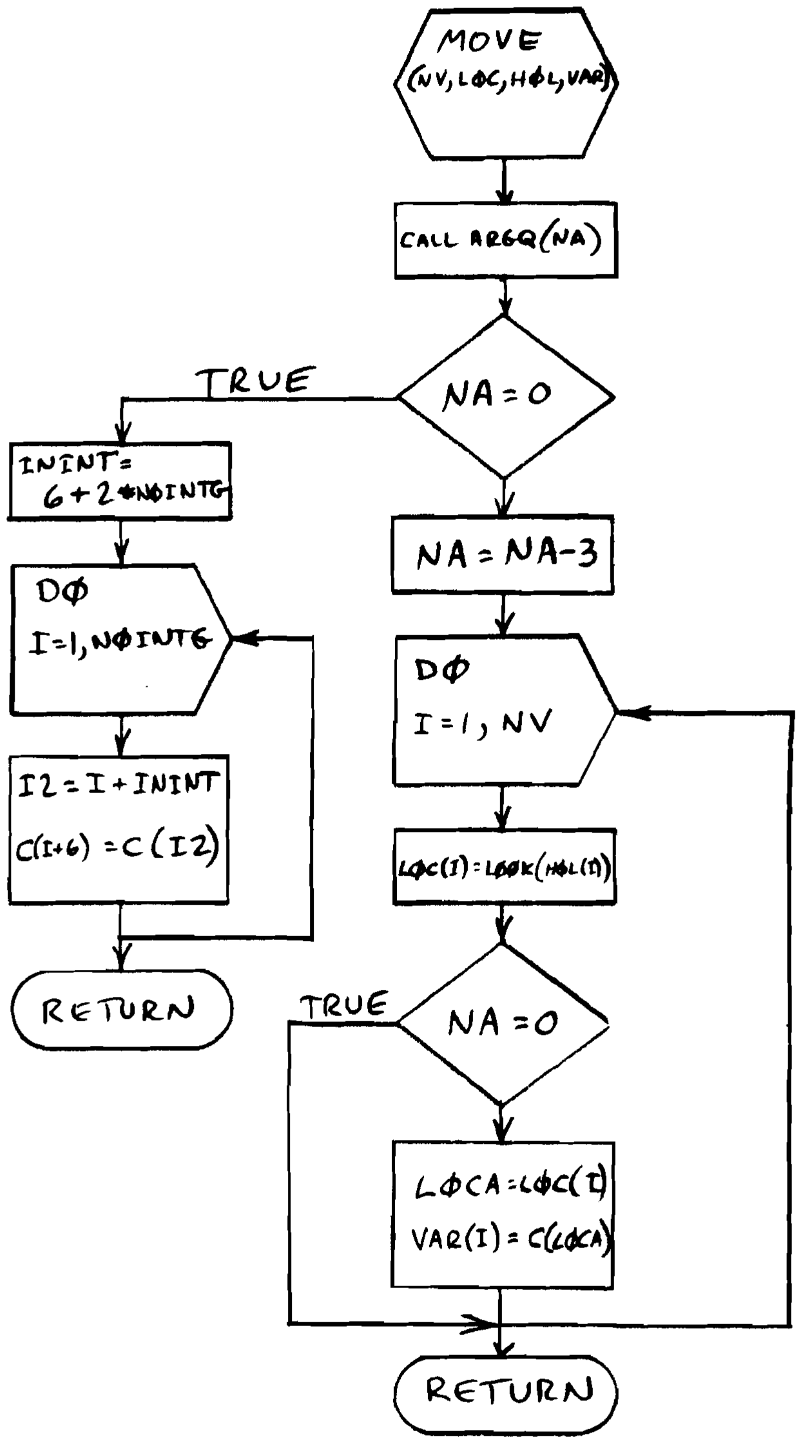


Figure 2. MOVE Routine

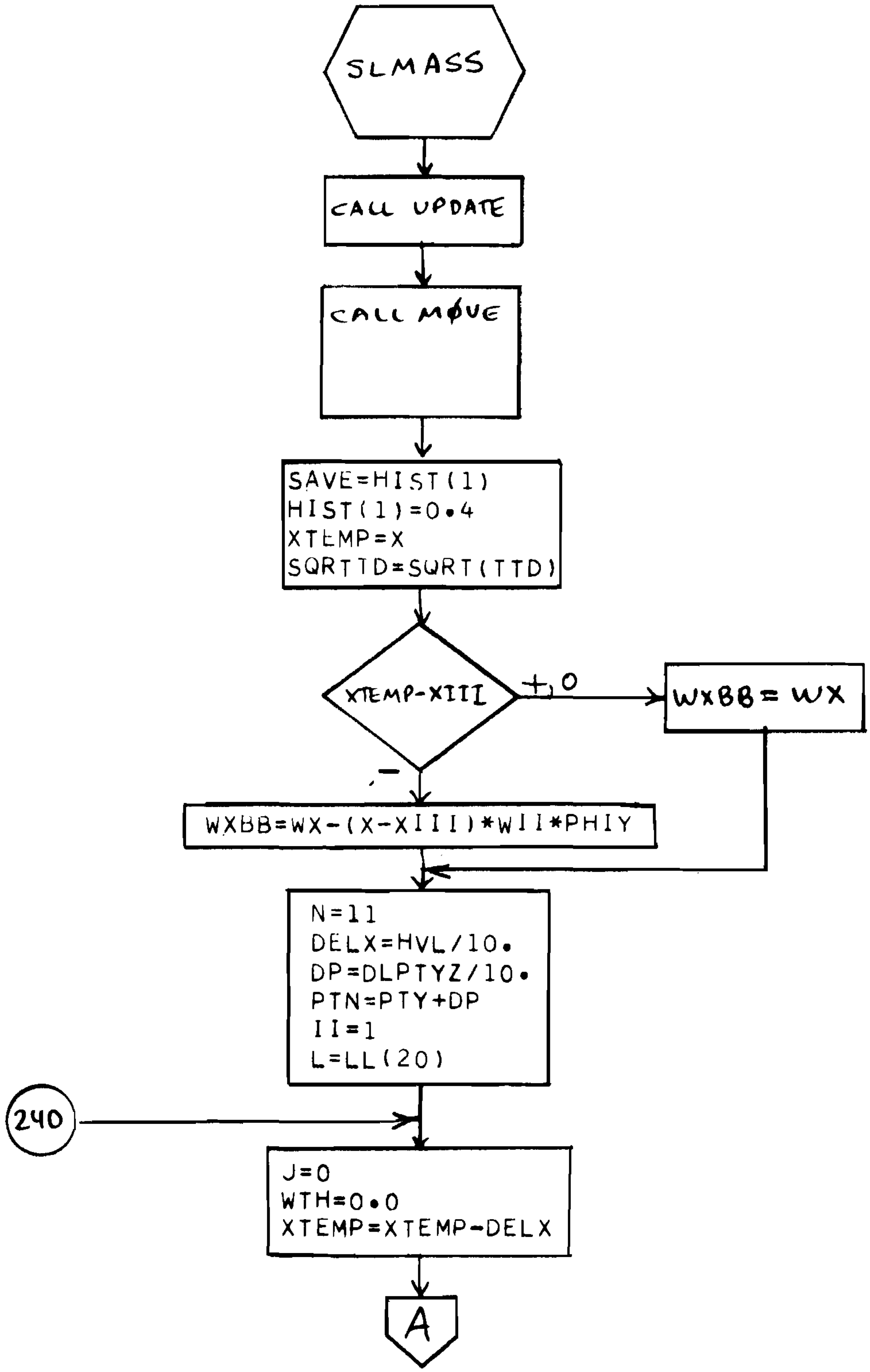


Figure 3. SLMASS Routine

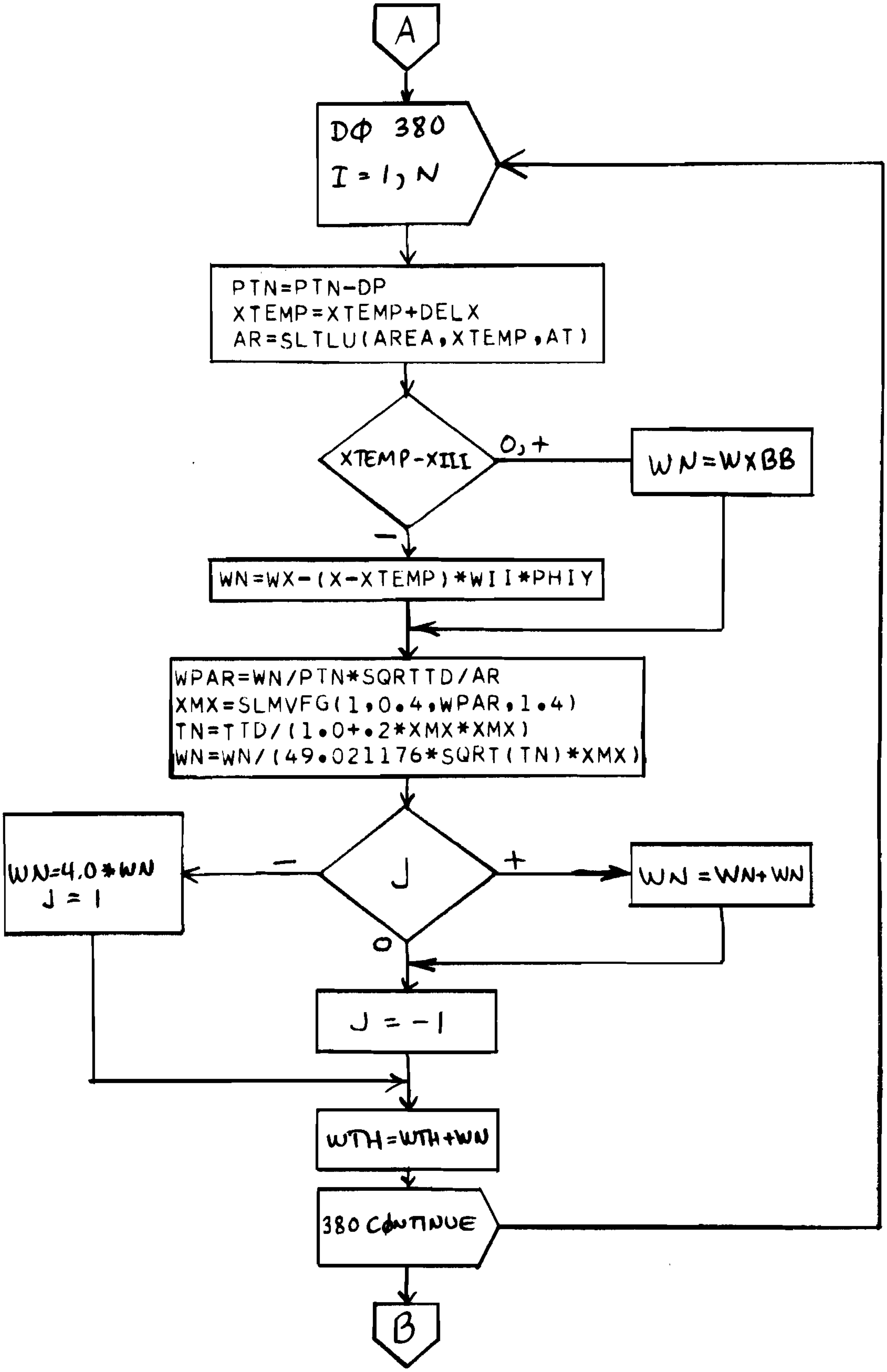


Figure 3. SLMASS Routine (Continued)

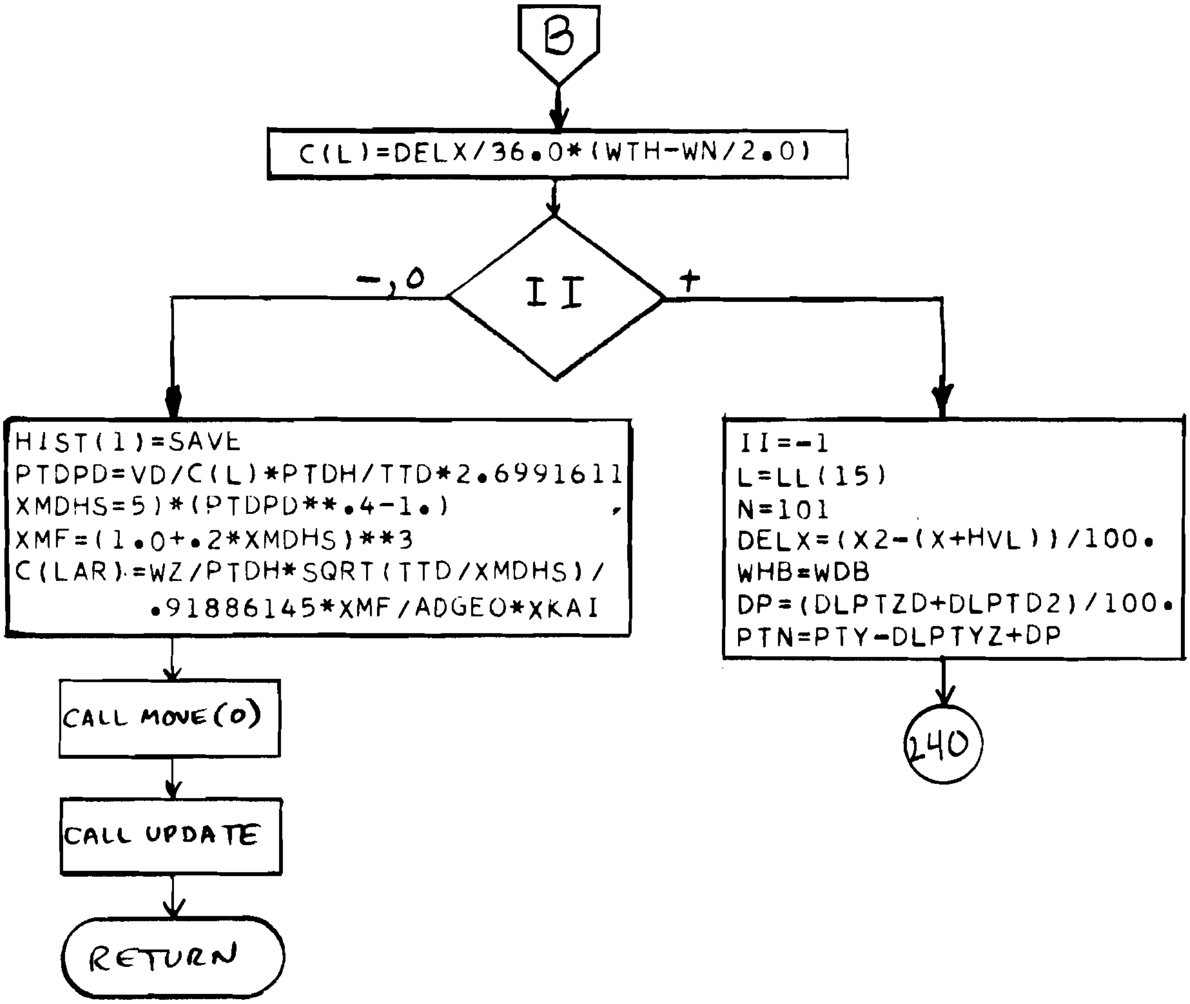


Figure 3. SLMASS Routine (Concluded)

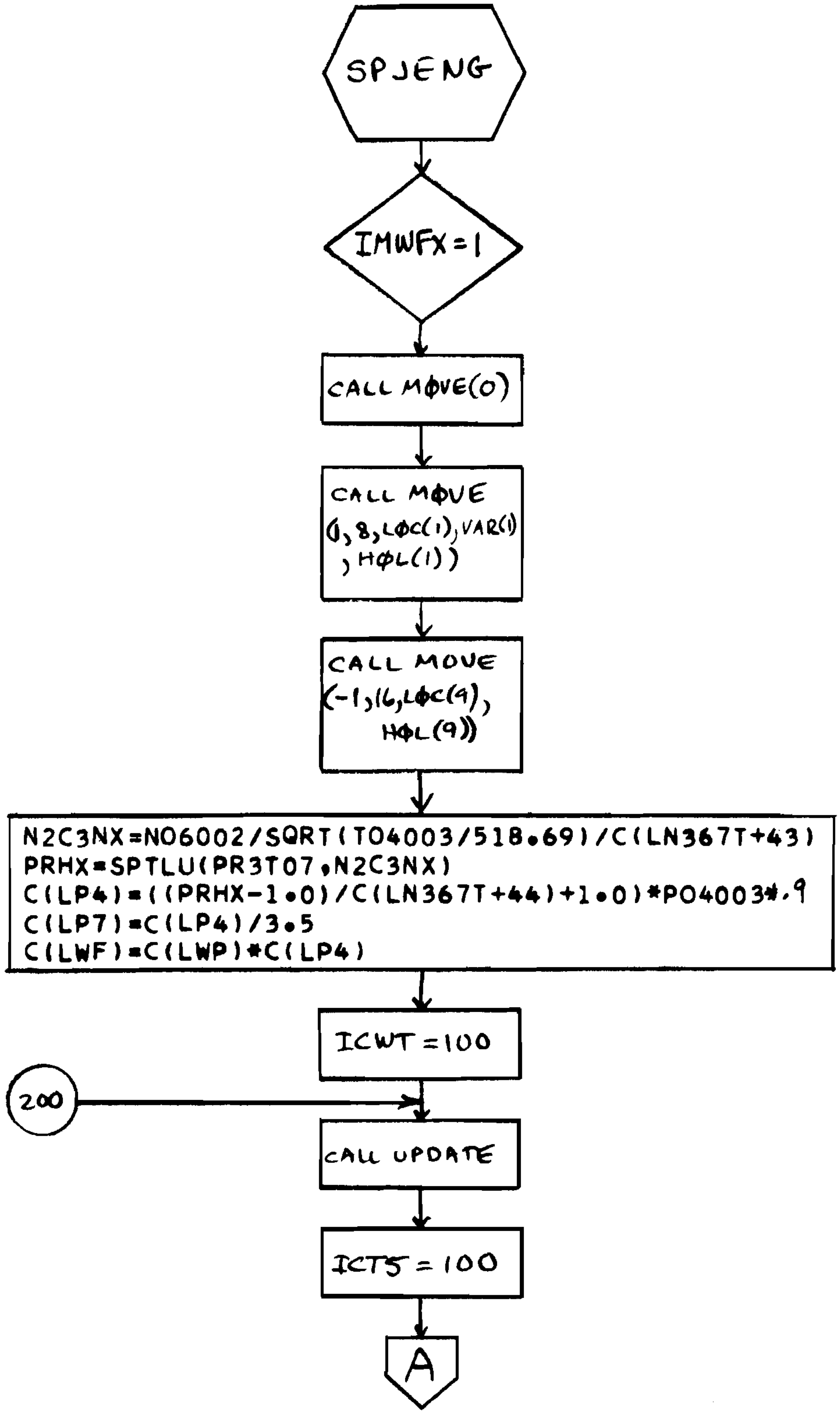


Figure 4. SPJENG Routine

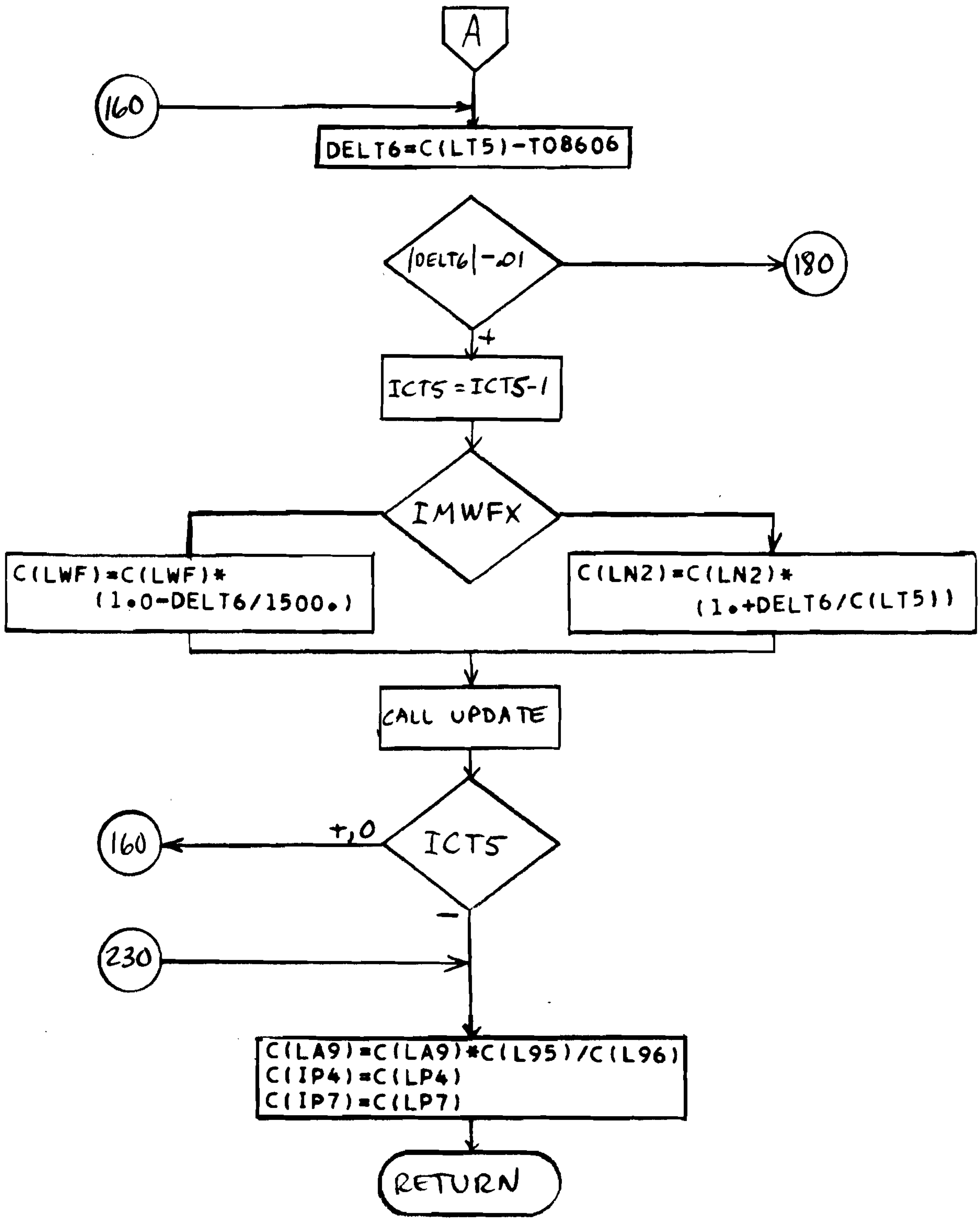


Figure 4. SPJENG Routine (Continued)

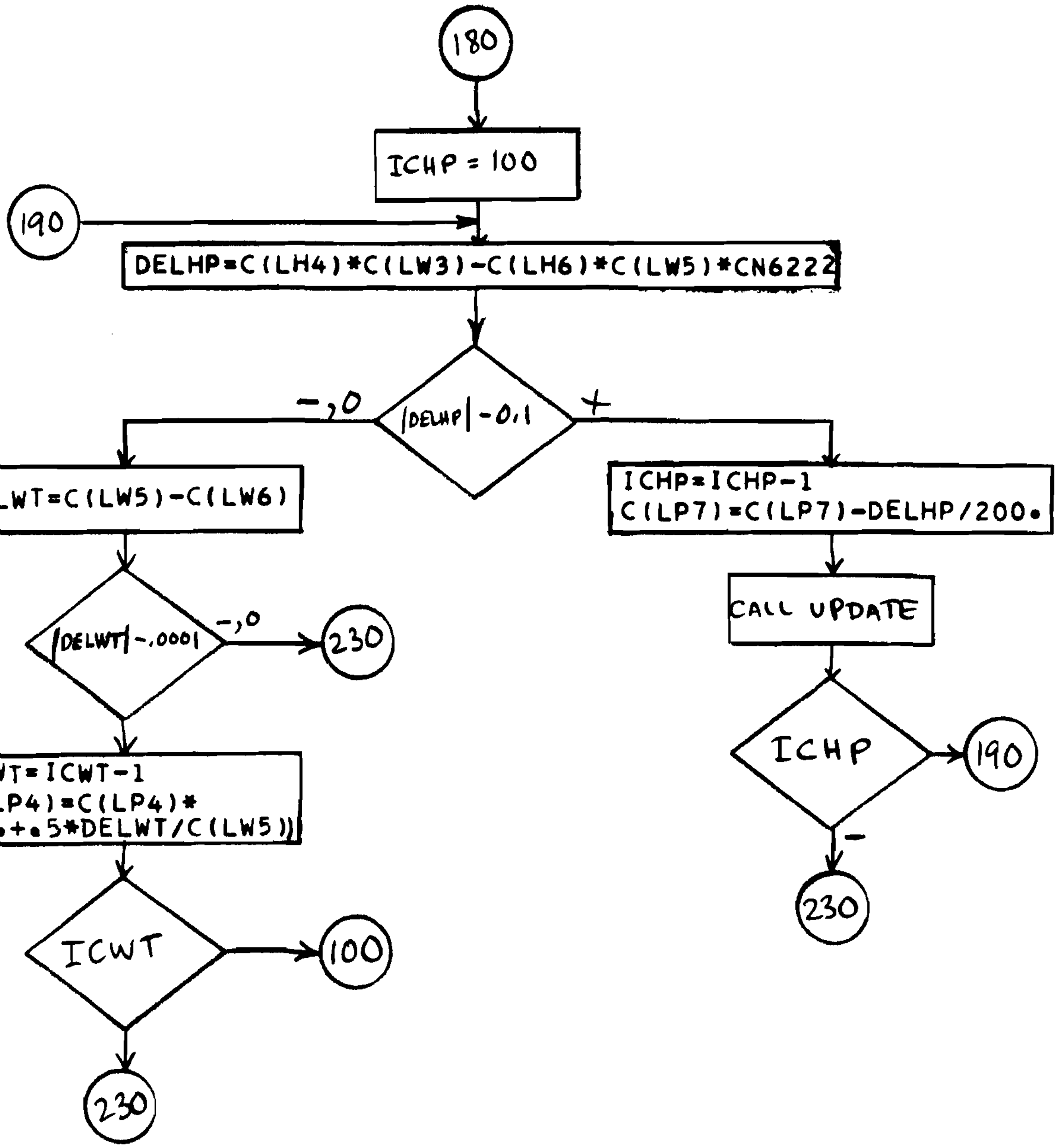


Figure 4. SPJENG Routine (Concluded)

TABLE I. STEADY

```
$IBFTC STEADY
      SUBROUTINE STEADY
      COMMON/CURVAL/C(2)/SYMBLS/NOINTG,NOSYMB,SYMB(2)
      COMMON/KEYS/NALARM,KPOINT,KPRINT,DELTP,KPLOT,KLAB,DELTG,KFINIS,
     1KRANGE,KLOCK,INTYPE,KREL,KABS,KPT1,T,RLAST,ALAST,KTITLE,KSIM
      DIMENSION
     1 HOL(37)    ,LOC(37)
      DIMENSION K(1)
      DATA HOL(1) / 222HA08108X08110P00000CN6004CN6007N06222T01015WA6003
     $P01015MN8110MN8105A08120A08113WF6000N08605QA8611T08608A08602WA1027
     $CN1001CN1003P01012A06009A08105WQ1967WQ1904CN1004MN0000T00000CN86T0
     3CN81T0WA1022A01803N08650N06002N08603T08602 /
      EQUIVALENCE
     1(LOC   (   1),   MAT),(LOC   (   2),   LXX),(LOC   (   3),   LP0)
     2,(LOC  (   4),   LP4),(LOC   (   5),   LP7),(LOC   (   6),   LN2)
     3,(LOC  (   7),   LTX),(LOC   (   8),   LW2),(LOC   (   9),   LPX)
     4,(LOC  (  10),   LMT),(LOC   (  11),   LTH),(LOC   (  12),   L19)
     5,(LOC  (  13),   LBP),(LOC   (  14),   LWF),(LOC   (  15),   LQ0)
     6,(LOC  (  16),   LQ1),(LOC   (  17),   LAJ),(LOC   (  18),   IA9)
     7,(LOC  (  19),   LW7),(LOC   (  20),   LC1),(LOC   (  21),   LC3)
     8,(LOC  (  22),   LP2),(LOC   (  23),   LA9),(LOC   (  24),   LAT)
     9,(LOC  (  25),   LWH),(LOC   (  26),   LWD),(LOC   (  27),   LC4)
     A,(LOC  (  28),   LM0),(LOC   (  29),   LT0),(LOC   (  30),   LNN)
     B,(LOC  (  31),   LN1),(LOC   (  32),   LWI),(LOC   (  33),   IAT)
     D,(K(1),C(1))          ,(LOC   (  34),   IN2),(LOC   (  35),   JN2)
     E,(LOC  (  36),   IQ0),(LOC   (  37),   IAJ)
      DATA NAME/6HSTEADY/
      IF(KLOCK) 30,20,30
   20 C(1)=0.0
   30 CONTINUE
      CALL MOVE (-1,37,LOC(1),DUM,HOL(1))
      DO 10 I=1,1
   10 LOC(I)=LOC(I)
      TR = (1.0+.2*C(LM0)**2)
      C(LP4) = 5.0 * C(LP0) * TR**3.5
      C(LP7)=C(LP4)/3.5
      C(LXX)=75.0
      C(LC4)=1.0
      C(LTX) = C(LT0) * TR
      C(LWD) = C(LP4)/C(LTX) * 19.48961
      C(LWH) = C(LWD) / 10.0
      C(LN2)=10000.
      CALL MOVE(0)
      C(LAT)=435.
      CALL UPDATE
      C(LAT)=SQRT(C(LTX))*C(LWI)/C(LPX)/.918861447/C(LMT)*
     1        (1.0+.2*C(LMT)**2)**3
      C(IAT)=C(LAT)
      LNT=K(LN1)
      C(LTH)=(C(LAT)-C(L19))/C(LNT+34)
      CALL PRINT (NAME)
      CALL SLMASS
```

TABLE I. STEADY (CONCLUDED)

```
      IMWFX=0
      C(LN2)=C(IN2)
      C(JN2)=C(IN2)
      LNT=K(LNN)
  200 CALL SPJENG(IMWFX)
      IF(IMWFX) 210,210,220
  210 C(LQO)=(C(LWF)/C(LP4)-C(LQ1))/C(LNT+14)
      C(IQO)=C(LQO)
  220 WFUEL=C(LWF)
      C(IA9)=C(LA9)
      C(LAJ)=(C(LA9)-210.1)/C(LNT+53)
      C(IAJ)=C(LAJ)
      C(LBP)=(C(LW7)-C(LW2))*SQRT(C(LTX))/(C(LC1)*C(LC3)*C(LP2))
      C(LN2)=C(JN2)
      CALL PRINT (NAME)
      CALL UPDATE
      CALL PRINT (NAME)
      CALL MOVE(0)
      CALL UPDATE
      CALL PRINT (NAME)
      IF(ABS(WFUEL-C(LWF))-.000001) 400,400,300
  300 IF(IMWFX) 310,310,400
  310 IF(C(LWF)-WFUEL) 320,320,330
  320 C(LQO)=C(LNT+12)
      GO TO 340
  330 C(LQO)=-C(LNT+12)
  340 IMWFX=1
      C(IQO)=C(LQO)
      GO TO 200
  400 RETURN
      END
```

TABLE II. MOVE

```
$IBFTC MOVE                                                             MOVE0010
      SUBROUTINE MOVE  (NV,LOC,HOL,VAR)                                 MOVE0020
C *** WHEN M = -1    DSL/90 PARAMETER NAMES IN HOL ARE LOCATED AND THIERMOVE0030
C     LOCATION PLACED IN LOC ARRAY                                      MOVE0040
C *** WHEN M = 0     DSL/90 INTEGRATOR I/C VALUES ARE MOVED TO OUTPUTS  MOVE0050
C *** WHEN M = +1    M = -1 IS EXECUTED THEN VALUES OF C(LOC) ARE MOVED MOVE0060
C     TO VAR ARRAY                                                      MOVE0070
      COMMON /CURVAL/C(1)/SYMBLS/NOINTG                                 MOVE0080
      DIMENSION LOC(1),HOL(1),VAR(1)                                    MOVE0090
      CALL ARGQ(NA)                                                     MOVE0100
      IF(NA.EQ.0)GO TO 500                                              MOVE0110
      NA = NA - 3                                                       MOVE0115
      DO 100 I=1,NV                                                     MOVE0120
      LOC(I) = LOOK(HOL(I))                                             MOVE0130
      IF(NA.EQ.0)GO TO 100                                              MOVE0140
      LOCA = LOC(I)                                                     MOVE0150
      VAR(I) = C(LOCA)                                                  MOVE0160
  100 CONTINUE                                                          MOVE0170
      RETURN                                                            MOVE0180
  500 ININT = 6 + 2 * NOINTG                                            MOVE0190
      DO 600 I=1,NOINTG                                                 MOVE0200
      I2 = I + ININT                                                    MOVE0210
  600 C(I+6) = C(I2)                                                    MOVE0220
      RETURN                                                            MOVE0230
      END                                                               MOVE0240
```

TABLE III. SLMASS

```
$IBFTC SLMASS
      SUBROUTINE SLMASS
      COMMON/CURVAL/C(1)
      COMMON/SYMBLS/NOINTG,NOSYMB,SYMB(1)
      COMMON/MEMRY/HIST(1)
      DIMENSION
     1 HOL(22)    ,VAR(22)    ,LL(22)
      DATA HOL(1)          /132HXO1005XO1021XO1023QL1101AO1TOOCN1004AO1803
     1AO1004VO1004PO1016PO1024PO1467PO1474PO1442WQ1904QL1046QA1022QA1005
     2XO1002WQ1967TO1015WA1027 /
      EQUIVALENCE
     1 (VAR     (    1),       X),(VAR     (    2),     XI),(VAR     (    3),   XIII)
     2,(VAR     (    4),     HVL),(VAR     (    5),   AREA),(VAR     (    6),   XKAI)
     3,(VAR     (    7),      AT),(VAR     (    8),  ADGEO),(VAR     (    9),     VD)
     4,(VAR     (   10),     PTY),(VAR     (   11),   PTDH),(VAR     (   12),DLPTYZ)
     5,(VAR     (   13),DLPTZD),(VAR     (   14),DLPTD2),(VAR     (   15),    WDB)
     6,(VAR     (   16),    PHIY),(VAR     (   17),    WII),(VAR     (   18),     WX)
     7,(VAR     (   19),      X2),(VAR     (   20),    WHB),(VAR     (   21),    TTD)
     8,(LL      (    6),     LAR),(VAR     (   22),     WZ)
      CALL UPDATE
      CALL MOVE (1,22,LL(1),VAR(1),HOL(1))
      SAVE=HIST(1)
      HIST(1)=0.4
      XTEMP=X
      SQRTTD=SQRT(TTD)
      IF(XTEMP-XIII) 210,220,220
  210 WXBB=WX-(X-XIII)*WII*PHIY
      GO TO 230
  220 WXBB=WX
  230 N=11
      DELX=HVL/10.
      DP=DLPTYZ/10.
      PTN=PTY+DP
      II=1
      L=LL(20)
  240 J=0
      WTH=0.0
      XTEMP=XTEMP-DELX
      DO 380 I=1,N
      PTN=PTN-DP
      XTEMP=XTEMP+DELX
      AR=SLTLU(AREA,XTEMP,AT)
      IF(XTEMP-XIII) 250,260,260
  250 WN=WX-(X-XTEMP)*WII*PHIY
      GO TO 270
  260 WN=WXBB
  270 WPAR=WN/PTN*SQRTTD/AR
      XMX=SLMVFG(1,0.4,WPAR,1.4)
  330 TN=TTD/(1.0+.2*XMX*XMX)
      WN=WN/(49.021176*SQRT(TN)*XMX)
      IF(J) 340,360,350
```

```
340 WN=4.0*WN
    J=1
    GO TO 370
350 WN=WN+WN
360 J=-1
370 WTH=WTH+WN
380 CONTINUE
    C(L)=DELX/36.0*(WTH-WN/2.0)
    IF(II) 400,400,390
390 II=-1
    L=LL(15)
    N=101
    DELX=(X2-(X+HVL))/100.
    WHB=WDB
    DP=(DLPTZD+DLPTD2)/100.
    PTN=PTY-DLPTYZ+DP
    GO TO 240
400 HIST(1)=SAVE
    PTDPD=VD/C(L)*PTDH/TTD*2.6991611
    XMDHS=5)*(PTDPD**.4-1.)
    XMF=(1.0+.2*XMDHS)**3
    C(LAR)=WZ/PTDH*SQRT(TTD/XMDHS)/.91886145*XMF/ADGEO*XKAI
    CALL MOVE(0)
    CALL UPDATE
    RETURN
    END
```

TABLE IV. SPJENG

```
$IBFTC SPJENG                                                            JENG0000
C      PROGRAM TO INITIALIZE SIMPLIFIED INTEGRATED PROPULSION SYSTEM     JENG0010
C      5-5-68   TURBOJET ENGINE INITIALIZATION                           JENG0020
       SUBROUTINE SPJENG (IMWFX)                                         JENG0030
       COMMON/CURVAL/C(1)/SYMBLS/NOINTG,NOSYMB,SYMB(1)                   JENG0040
       DIMENSION                                                         JENG0050
     1 HOL(24)    ,LOC(24)    ,VAR(8)
       DATA NAME/6HSPJENG/
       DATA HOL(1) / 144HCN6222N06002T08606T04003P04003CN367TPR3T07WA3T31
     1P06004P06007P06005T06005WF8601WA6003HD6034HD6056WG6055WG6056A06009JENG0080
     2CN6004CN6007WG6095WG6096QA8611 /
       EQUIVALENCE                                                       JENG0100
     1 (LOC    (    9),    LP4),(LOC     (   10),    LP7),(LOC    (   11),    LP5)JENG0110
     2,(LOC    (   12),    LT5),(LOC     (   13),    LWF),(LOC    (   14),    LW3)JENG0120
     3,(LOC    (   15),    LH4),(LOC     (   16),    LH6),(LOC    (   17),    LW5)JENG0130
     4,(LOC    (   18),    LW6),(LOC     (   19),    LA9),(LOC    (   20),    IP4)JENG0140
     5,(LOC    (   21),    IP7),(LOC     (   22),    L95),(LOC    (   23),    L96)JENG0150
     6,(LOC    (   24),    LWP),(LOC     (    2),    LN2)
       EQUIVALENCE                                                       JENG0160
     1 (VAR    (    1),CN6222),(VAR      (    2),N06002),(VAR     (    3),T08606)JENG0170
     2,(VAR    (    4),T04003),(VAR      (    5),P04003),(VAR     (    6),LN367T)JENG0180
     3,(VAR    (    7),PR3T07),(VAR      (    8),WA3T31)                JENG0190
       REAL        N06002      ,N2C3NX                                   JENG0200
       IF(IMWFX.EQ.1)GO TO 15                                            JENG3235
       CALL MOVE(0)                                                      JENG0210
       CALL MOVE (1,8,LOC(1),VAR(1),HOL(1))                              JENG0220
       CALL MOVE (-1,16,LOC(9),VAR(1),HOL(9))
       DO 10 I=1,1
    10 LOC(I)=LOC(I)
       N2C3NX=N06002/SQRT(T04003/518.69)/C(LN367T+43)                    JENG0240
       PRHX=SPTLU(PR3T07,N2C3NX)                                         JENG0250
       C(LP4)=((PRHX-1.0)/C(LN367T+44)+1.0)*P04003     * 0.9             JENG0270
       C(LP7)=C(LP4)/3.5                                                 JENG0290
       C(LWF)=C(LWP)*C(LP4)
    15 ICWT = 100                                                        JENG0310
       WRITE (6,20) N2C3NX,PRHX,            C(LP4),        C(LP7),C(LWF)
    20 FORMAT (1H0/(1H ,5F20.8))
    30    FORMAT (1H ,I5,5F19.5/(1H ,5X,5F19.5))
   100 CALL UPDATE
   150 ICT5=100                                                          JENG0390
   160 DELT6=C(LT5)-T08606                                               JENG0400
       WRITE (6,30) ICT5,C(LWF),C(LT5),C(LN2),DELT6
       IF(ABS(DELT6)-.01) 180,180,170
   170 ICT5=ICT5-1                                                       JENG0420
       IF(IMWFX) 172,174,172
   172 C(LN2)=C(LN2)*(1.+DELT6/C(LT5))
       GO TO 176
   174 C(LWF)=C(LWF)*(1.0-DELT6/1500.)                                   JENG0430
   176 CALL UPDATE                                                       JENG0440
       IF(ICT5) 224,160,160                                              JENG0450
   180 ICHP=100                                                          JENG0460
   190 DELHP=C(LH4)*C(LW3)-C(LH6)*C(LW5)*CN6222                          JENG0470
       WRITE (6,30) ICHP,C(LP7),C(LH4),C(LW3),C(LH6),C(LW5),DELHP
       IF(ABS(DELHP)-0.1) 210,210,200
   200 ICHP=ICHP-1                                                       JENG0490
       C(LP7)=C(LP7)-DELHP/200.                                          JENG0500
       CALL UPDATE                                                       JENG0510
       IF(ICHP) 224,190,190                                              JENG0520
```

TABLE IV. SPJENG (CONCLUDED)

```
210 DELWT=C(LW5)-C(LW6)                                              JENG0530
    WRITE (6,30) ICWT,WA603G,C(LW5),C(LW6),DELWT
    IF(ABS(DELWT)-.0001)230,230,220
220 ICWT=ICWT-1                                                      JENG0550
    C(LP4)=C(LP4)*(1.+.5*DELWT/C(LW5))
    WRITE (6,222) ICCOMP,ICT5,ICHP,ICWT                              JENG0570
222 FORMAT (1H ,4I10)                                                JENG0580
    IF(ICWT) 224,100,100                                             JENG0590
    CALL INTRAN
    CALL PRINT (NAME)
    C(LA9)=C(LA9)*C(L95)/C(L96)                                      JENG0630
    C(IP4)=C(LP4)                                                    JENG0640
    C(IP7)=C(LP7)                                                    JENG0650
    RETURN                                                           JENG0660
    END                                                              JENG0670
```


(The reverse of this page is blank)

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Los Angeles Division Of North American Rockwell Corporation	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP —

3. REPORT TITLE

Propulsion System Flow Stability Program (Dynamic)
Part XVII. Propulsion System Simulation Digital Computer Program Format and Routines

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Phase I Final Technical Report June 1967 to September 1968

5. AUTHOR(S) (First name, middle initial, last name)

Earl H. Kaplan and Heeman W. Wong

6. REPORT DATE December 1968	7a. TOTAL NO. OF PAGES 123 + i through x	7b. NO. OF REFS None
8a. CONTRACT OR GRANT NO. F33615-67-C-1848 b. PROJECT NO.	9a. ORIGINATOR'S REPORT NUMBER(S) NA-68-562 Part XVII	
c. d.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) AFAPL-TR-68-142 Part XVII	

10. DISTRIBUTION STATEMENT This document is subject to special export controls and each transmittal to foreign governments or foreign nationals may be made only with prior approval of the Air Force Aero Propulsion Laboratory (APTA), Air Force Systems Command, Wright-Patterson Air Force Base, Ohio.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Air Force Aero Propulsion Laboratory Air Force Systems Command Wright-Patterson Air Force Base, Ohio 45433

13. ABSTRACT

The primary objective of Task 7 of the "Propulsion System Flow Stability Program" was to develop a simulation program to be used in Phase II for the evaluation of two control systems capable of sensing and accommodating a transient condition.

Since the work on this task was being performed by three companies, every effort was made to insure compatibility in terminology, units, and program documentation as well as to provide means of communicating the myriad details involved in making computer runs of the system. This documentation format is described in Section II of this volume.

An early element of this task was the selection of a simulation language for use in programming the simulation. The choice of IBM's DSL/90 and the factors involved in making that choice are discussed in Section III.

Simulation programs have a natural tendency to be rather voluminous and, when the system being simulated is as complex as a supersonic inlet, turbofan, and an integrated control system can be, computer storage space is rapidly filled. To alleviate this crowding, numerous logic blocks which were repetitive, such as compressor logic, were removed from the simulation logic deck and made into subroutines or functions. These subprograms are discussed in Section IV.

DD FORM 1 NOV 65 1473

UNCLASSIFIED
Security Classification

UNCLASSIFIED

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Propulsion System Computer Program						
Propulsion System Simulation						

ITEM 13. ABSTRACT (continued)

Once the simulation logic is written, the most difficult task of all begins. The job of initialization is usually not given proper emphasis until many hours of work have convinced all concerned that it is really the most important phase. Section V discusses this task and shows an example of an initialization routine.

UNCLASSIFIED

Security Classification

NTRL
NATIONAL TECHNICAL REPORTS LIBRARY
U.S. Department of Commerce

National Technical Reports Library

Help

Advanced Search

AD859282

Only documents with full text

Search Clear

Fields

Title

Add field

Date Published

<1900 TO 2019

Refine

Source

Non Paid Delimited ADS (1)

Keywords

Subject

Document Type

Year

Search Results

Filter results Filter Results

Search Relevance DESC

(1 - 1 of 1) 1 10

Title/Authors	Accession Number	Publication Year	Page Count	Download
Propulsion System Flow Stability Program (Dynamic). Part XVII. Propulsion System Simulation Digital Computer Program Format and Routines. Kaplan, E. H; Wong, H. W.	AD859282	1968	133 pages	

(1 - 1 of 1) 1 10

Accession Number : AD0859282

Title : Propulsion System Flow Stability Program (Dynamic). Part XVII. Propulsion System Simulation Digital Computer Program Format and Routines.

Descriptive Note : Final technical rept. Jun 67-Sep 68 on Phase 1,

Corporate Author : NORTH AMERICAN ROCKWELL CORP LOS ANGELES CA LOS ANGELES DIV

Personal Author(s) : Kaplan, Earl H ; Wong, Heeman W

Report Date : Dec 1968

Pagination or Media Count : 133

Abstract : The object of the study was to develop a simulation program to be used for the evaluation of two control systems capable of sensing and accommodating a transient condition. (Author)

Descriptors : *TURBOJET ENGINES , CONTROL SYSTEMS , MATHEMATICAL MODELS , COMPUTER PROGRAMS , TRANSIENTS , DUCT INLETS , SUPERSONIC CHARACTERISTICS , TURBOFAN ENGINES , FLOW CHARTING , SUBROUTINES , DIGITAL COMPUTERS , INPUT OUTPUT DEVICES

Subject Categories : Jet and Gas Turbine Engines

Distribution Statement : APPROVED FOR PUBLIC RELEASE