Dynetics, Inc.

P. O. Drawer B
Huntsville, Alabama
35814-5050

TR-93-NAS8-38981-061

FINAL REPORT

MASTRE TRAJECTORY CODE UPDATE TO AUTOMATE FLIGHT TRAJECTORY DESIGN, PERFORMANCE PREDICTIONS, AND VEHICLE SIZING FOR SUPPORT OF SHUTTLE AND SHUTTLE DERIVED VEHICLES - PROGRAMMERS MANUAL

SUBCONTRACT NAS8-38981
TASKS 1, 2, 3, AND 4

APRIL 1993

PREPARED FOR:

MARSHALL SPACE FLIGHT CENTER
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
MARSHALL SPACE FLIGHT CENTER, AL 35812

TR-93-NAS8-38981-061

FINAL REPORT

MASTRE TRAJECTORY CODE UPDATE TO AUTOMATE FLIGHT TRAJECTORY DESIGN, PERFORMANCE PREDICTIONS, AND VEHICLE SIZING FOR SUPPORT OF SHUTTLE AND SHUTTLE DERIVED VEHICLES - PROGRAMMERS MANUAL

SUBCONTRACT NAS8-38981
TASKS 1, 2, 3, AND 4

APRIL 1993

PREPARED FOR:

MARSHALL SPACE FLIGHT CENTER
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
MARSHALL SPACE FLIGHT CENTER, AL 35812

FORWARD

This is one of three reports presenting the work performed by Dynetics, Inc. while under contract to the Propulsion Laboratory of the Marshall Space Flight Center (NAS8-38981) to perform a research study entitled "MASTRE Trajectory Code Update to Automate Flight Trajectory Design, Performance Predictions, and Vehicle Sizing for Support of Shuttle and Shuttle Derived Vehicles". Technical coordination was provided by Mr. Dan Adams and Mr. David Anderson of EP-56.

(Reverse Blank)

TABLE OF CONTENTS

Page

1. INTRODUCTION 1

1.1 PROGRAM OPERATIONS OVERVIEW 1

1.1.1 Initiation 2

1.1.2 Forward Trajectory Integration and Evaluation 2

1.1.3 Backward Trajectory Integration 3

1.1.4 Partial Derivative Calculation 4

1.1.5 Parameter Update and Convergence Test 4

1.1.6 Table and Plot Output 5

1.2 OPERATIONAL CONSTRAINTS 5

1.3 USE OF SPECIAL SOFTWARE 5

2. SUBROUTINE DESCRIPTIONS 6

2.1 ALPHABETICALLY ORDERED LIST OF NONSYSTEM SUBROUTINE AND FUNCTION NAMES 6

3. NARRATIVE SUBROUTINE DESCRIPTIONS 13

3.1 SUBROUTINE MASTRE 14

3.2 SUBROUTINE ACNTRO 21

3.3 SUBROUTINE ACSTOP 24

3.4 SUBROUTINE ADER 34

3.5 SUBROUTINE ADER1 73

3.6 SUBROUTINE AEBANK 80

3.7 SUBROUTINE AEOSR 83

3.8 SUBROUTINE AFORND 87

3.9 SUBROUTINE AFORUN 97

3.10 SUBROUTINE AGEO 116

TABLE OF CONTENTS (Continued)

Page

3.11 SUBROUTINE AINIT 119

3.12 SUBROUTINE AMULG 151

3.13 SUBROUTINE AMULG7 154

3.14 SUBROUTINE ANEWCH 157

3.15 SUBROUTINE APRTN 172

3.16 SUBROUTINE AREAIN 192

3.17 SUBROUTINE ATHREV 195

3.18 SUBROUTINE ATILT 215

3.19 SUBROUTINE ATMOSPHERE 228

3.20 SUBROUTINE BOPTBL 237

3.21 SUBROUTINE B1PTBL 240

3.22 SUBROUTINE BADLX 247

3.23 SUBROUTINE BAKRN 259

3.24 SUBROUTINE BDR1I 267

3.25 SUBROUTINE BDRI 341

3.26 SUBROUTINE BKEND 347

3.27 SUBROUTINE BSAVEG 355

3.28 SUBROUTINE BTHREV 362

3.29 SUBROUTINE CAERO 372

3.30 SUBROUTINE CHIPOL 375

3.31 SUBROUTINE COPLDF 378

3.32 SUBROUTINE DBANK 381

3.33 SUBROUTINE DESOLV 384

TABLE OF CONTENTS(Continued)

Page

3.34 SUBROUTINE DISPPRT 392

3.35 SUBROUTINE DPIR 401

3.36 SUBROUTINE FIND 419

3.37 FUNCTION FUNISP 422

3.38 SUBROUTINE GOUT 425

3.39 SUBROUTINE HOLDIT 428

3.40 SUBROUTINE IINRC 431

3.41 SUBROUTINE INITIAL 434

3.42 SUBROUTINE INTER 437

3.43 SUBROUTINE IS_PROCESS_INTERACTIVE 440

3.44 SUBROUTINE LDSWI 443

3.45 SUBROUTINE LDWRIT 446

3.46 SUBROUTINE LLPRT 449

3.47 SUBROUTINE MASSPRO 455

3.48 SUBROUTINE MATINV 461

3.49 SUBROUTINE MENU 465

3.50 SUBROUTINE RRAINT 468

3.51 SUBROUITINE RRASPL 473

3.52 SUBROUTINE RRRATM 476

3.53 SUBROUTINE RTMRK 489

3.54 SUBROUTINE SDATA 492

3.55 SUBROUTINE SEARCH 495

3.56 SUBROUTINE SIMUL 498

TABLE OF CONTENTS (Concluded)

		Page
3.57	SUBROUTINE SPLINE	501
3.58	SUBROUTINE SUM15STG	504
3.59	SUBROUTINE SUMARY	512
3.60	SUBROUTINE SUMHLLV	521
3.61	SUBROUTINE TERMINATE	527
3.62	SUBROUTINE TRANFM	530
3.63	SUBROUTINE WINDIN	533
	APPENDIX A	A-1
	APPENDIX B	B-1
	APPENDIX C	C-1

Section 1.0
Introduction

1.1 Program Operations Overview

This manual furnishes the information required by a programmer using the **M**inimum Hamiltonian **AS**cent **TR**ajectory **E**valuation (MASTRE) Program. This document enables the programmer to either modify the program or convert the program to computers other than the VAX computer. This document includes documentation for each subroutine or function based on providing the definitions of the variables and a source listing. Questions concerning the equations, techniques, or input requirements should be answered by either the Engineering or User's manuals.

Three appendices are also included which provide a listing of the Root-Sum-Square (RSS) program, a listing of subroutine names and definitions used in the MASTRE User Friendly Interface (UFI) Program, and a listing of the subroutine names and definitions used in the Mass Properties Program. All of these programs were developed under this contract. The RSS Program is used to aid in the performance of dispersion analyses. Ths RSS program reads a file generated by the MASTRE Program, calculates dispersion parameters, and generates output tables and output plot files. The UFI Program provides a screen user interface to aid the user in providing input to the model. The Mass Properties Program defines the mass properties data for the MASTRE program through the use of user interface software.

Before a programmer begins work on this program he should acquire an understanding of the general flow of the program placing special emphasis on the integration package subroutines DESOLV and DPIR. Whenever these subroutines are called they control the flow of the integration process until termination (which is caused by calling the subroutine RTMRK). Because both forward and backward integrations are required by the steepest ascent algorithm, these subroutines are the focal points of the program.

The general flow of the MASTRE program consists of the following six primary segments :

- Initiation,
- Forward trajectory integration and evaluation,
- Backward trajectory integration,
- Partial derivative calculation,
- Parameter update and convergence test, and
- Table and plot output.

These six elements are described briefly in the following paragarphs. A general flow diagram of the main program (MASTRE) is shown in **Figure 1-1**.

1.1.1 Initiation

The main program begins operation by calling the subroutines INITIAL, MENU, and AINIT to preset variables and constants used in the program. The subroutine AINIT reads the input data in the Namelist format which includes a description of the vehicle, the type of vehicle to be simulated, the parameters to be optimized, and the output required. The Users Manual defines the input parameters required by the program. Subroutines INITIAL and MENU provide user interface options to control the operation of the program during execution.

1.1.2 Forward Trajectory Integration and Evaluation

The forward trajectory integration and evaluation is performed by calling subroutines AFORUN and AFORND. Subroutine AFORUN defines the setup logic for the forward trajectory and calls the integration subroutine DESOLV. The AFORND subroutine calculates the values of the constraint errors (the difference between the desired and simulated values), the payoff value, and the initial values of the backward trajectory. After the converged run is simulated, the AFORND subroutine prinbts and also calls other subroutines to print out parameter summary tables, orbital element tables, load indicator tables, and weight summary tables. The calling of subroutine AFORUN is the beginning of the iteration cycle.

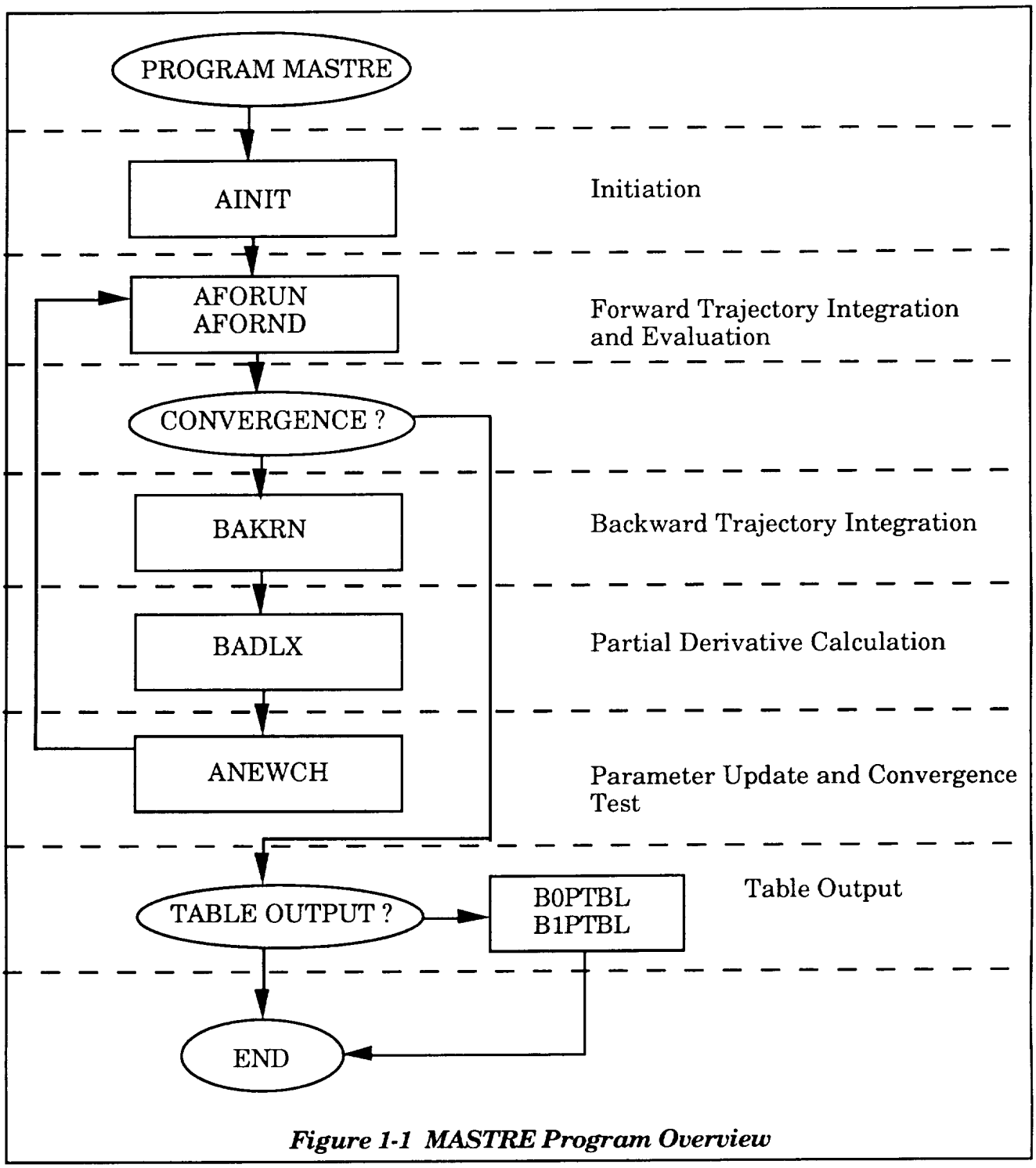

Figure 1-1 MASTRE Program Overview

1.1.3 Backward Trajectory Integration

A backward trajectory integration is simulated from the terminal point of the forward trajectory to the min-H initialization point by calling

subroutine BAKRN. During this time, the adjoint differential equations are integrated based upon interpolated values of the state variables stored during the forward trajectory. Impulse response functions, which consist of the first and second partial derivatives of the state with respect to the attitude angles, are also evaluated to determine the effects of the pitch and yaw attitude angles on the trajectory. Subroutine BSAVEG is also called during the backward integration to integrate the weighting matrix using the Simpson method.

1.1.4 Partial Derivative Calculation

The partial derivatives of the payoff, terminal, and intermediate constraints with respect to the control parameters are determined by calling subroutine BADLX. The control parameters consist of parameters such as lift-off weight, first stage pitch and yaw angles, and thrust time segments. The partial derivatives are defined by using forward differences, or the shooting method, which involves simulating forward trajectories to calculate the changes with respect to the payoff, terminal, and intermediate constraints by varying the control parameters. The partial derivatives are calculated by dividing the difference between the perturbed values of the constraints and the nominal values by the control parameter variances.

1.1.5 Parameter Update and Convergence Test

The final portion of the iteration cycle is the call to subroutine ANEWCH which controls the following items:

- Determination of optimization and restoration step size requirements;
- Controls parameter update requirements;
- Calculates update requirements for the control variables (pitch and yaw attitude angles) during the min-H phase of flight;
- Investigation of the convergence tests; and
- Determination of convergence.

If the convergence tests are met, the iteration cycle is terminated after the next forward trajectory is simulated and evaluated. If the

convergence tests are not met, the program flow is returned to forward trajectory integration and the iteration cycle is continued.

1.1.6 Table and Plot Output

Once a converged trajectory is obtained, or the maximum number of iterations has been reached, special output tables and a plot file are produced based upon the user request. If tables are desired, the subroutine B0PTBL is called to read the $INPUT2 Namelist file and subroutine B1PTBL is called to generate the special output tables. The plot file can be used by the plotting package in the MASTRE User Interface Program.

1.2 Operational Constraints

The best method of running the MASTRE Program is through use of a command file which defines user options and the names of input/output files. The design and construction of the command file is described in the MASTRE User Interface documentation. The user, however, is not limited to this manner of execution and can run the program in an interactive fashion by typing the command "RUN MASTRE". The user is then prompted for inputs.

Currently the MASTRE program is resident on the micro-VAX computer system and utilizes features of the VAX Fortran compiler. To convert this program to another computer system these special features should be investigated for compatiblity.

1.3 Use of Special Software

Although the MASTRE Program uses no special external software, the accompanying MASTRE User Interface Program and the Mass Properties Program require the use of the Screen Management Guideline (SMG) routines which are resident on the VAX computer system.

Section 2.0
Subroutine Descriptions

The following sub-sections give an alphabetic listing of all subroutines and functions and provide a brief statement of the purpose of each.

2.1 Alphabetically Ordered List of Nonsystem Subroutine and Function Names

Subroutine Name	Function
ACNTRO	Determines the values of pitch and yaw attitude based on interpolation of the proper control table. (Only used during the min-H phase).
ACSTOP	Evaluates the payoff and determines the value of the terminal and intermediate equality constraints. Also evaluates the partial derivatives of the payoff and constraints with respect to the state.
ADER	Evaluates the time independent state derivatives during the forward integration.
ADER1	Evaluates the time dependent state derivatives during the forward integration.
AEBANK	Reads aerodynamic data from aerodynamic database.
AEOSR	Stores the state variables during the forward trajectory.
AFORND	Controls the call to ACSTOP to determine the payoff function and terminal and intermediate constraints, computes errors from desired values, sets up initial

	backward integration parameters, and prints terminal summaries.
AFORUN	Controls the setup logic for the forward trajectory integration and calls the integration package.
AGEO	Calculates the gravitational parameters for the forward and backward integration.
AJUMP	Controls storage of state variables for 1st stage jump start. (Entry point in subroutine ATILT).
ALIFT	Controls lift-off simulation based on thrust-to-weight requirement. (Entry point in subroutine ATILT).
AINIT	Reads input data and performs initial setup.
AMULG	Linear interpolation scheme for two dependent parameters.
AMULG7	Linear interpolation scheme for seven dependent parameters
ANEWCH	Evaluates the optimization and restoration equations to determine the required changes in the controlling parameters and variables. Tests for convergence are also made in this subroutine.
APRTN	Computes output parameters, prints data, and writes output files for output tables and plots.
AQMAX	Provides maximum dynamic pressure printout.
AREAIN	Controls user interface to call termination, execution, data input, plot and table output files, and printout menu.

AROLL	Defines parameters and prints roll maneuvers. (Entry point in subroutine ATILT).
ATHREV	Thrust event control routine which sets up vehicle geometry, aerodynamic data, number of engines, thrust limits, and flow rates; and calculates times for events.
ATHRO	Defines parameters and prints MPS throttle events. (Entry point in subroutine ATILT).
ATILT	Controls time related events directly after lift-off. Also contains entry point routines AQMAX, AXPRT, GLIMT, AWDEV, ALIFT, ATHRO, AROLL, and AJUMP which control maximum dynamic pressure, normal print, acceleration, weigth drop, throttle events, roll maneuver, and jump start events, respectively.
ATMOSPHERE	Atmosphere subroutine which contains Patrick AFB nominal, hot and cold and Vandenburg AFB nominal, hot, and cold atmosphere routines.
AWDEV	Defines parameters and prints weight drop events. (Entry point in subroutine ATILT)
AXPRT	Prints trajectory block printout based on print interval. (Entry point in subroutine ATILT).
B0PTBL	Controls input for output tables for subroutine B1PTBL.
B1PTBL	Prints output parameter summary tables.
BADLX	Determines the influence coefficients (partial derivatives) of the payoff, terminal, and intermediate constraints with respect to the control parameters.

BAKRN	Controls the setup logic for the backward trajectory integration and calls the integration routines.
BDR1I	Computes the partial derivatives used in the calculation of the adjoint equations for the backward trajectory.
BDRI	Computes the total adjoint derivatives.
BGLIM	Defines acceleration limit events during backward trajectory. (Entry point in subroutine BTHREV).
BKEND	Terminates adjoint (or backward) integration and adjusts influence coefficients (if required).
BSAVEG	Saves impulse response functions and performs Simpson integration to calculate dynamic weighting matrix.
BTHREV	Controls the thrust event logic during the backward integration. Also contains entry routines BWDEV and BGLIM which controls weight drop and acceleration events during the backward integration.
BWDEV	Controls weight drop events during backward integration. (Entry point in subroutine BTHREV).
CAERO	Evaluates nonlinear aerodynamic data.
CHIPOL	Evaluates pitch and yaw attitude polynomials.
COPLDF	Writes a list directed file based on user requirements.
DBANK	Reads aerodynamic and wind database files.
DESOLV	Controls call to integration subroutine DPIR.

DEVISP	Calculates partial derivative of specific impulse with respect to mass. (Entry point in function FUNISP).
DISPPRT	Controls the output of data to the dispersion file.
DPIR	Integration routine.
FIND	Determines attitude polynomials based on attitude time history information.
FUNISP	Calculates specific impulse based on thrust throttle level.
GLIMT	Defines parameters and prints acceleration limit events. (Entry point in subroutine ATILT).
GOUT	Displays a character string at the terminal and writes it to the print file.
HOLDIT	Creates pause in execution of program in graphic or demand mode.
IINRC	Reads integer values from the aerodynamic database.
INITIAL	Initializes program inputs.
INTER	Performs linear interpolation on nonlinear aerodynamic data.
IS_PROCESS_ INTERACTIVE	Inquires whether the current process is interactive or demand mode.
LDSWI	Creates and writes direct access file (used in interactive mode).
LDWRIT	Writes data to specified direct access file.

LLPRT	Controls user interface to print constraints and optimized updated values, print control variable update, print parameter summary table, and print load indicator table.
MASSPRO	Mass properties routine for calculation of dynamic center of gravity.
MASTRE	Main program which controls overall program flow.
MATINV	Performs matrix inversion.
MENU	Controls calls to area mode, execute mode, print mode, and namelist input selection.
PR63	Nominal 1963 Patrick reference atmosphere routine. Used with Range Reference atmosphere model.
RRAINT	Initilization routine for range reference atmosphere.
RRASPL	Spline interpolation routine for range reference atmosphere.
RRRAIN	Interpolates either the monthly range reference table or the final atmosphere table.
RRRATM	Range Reference atmosphere routine.
RTMRK	Part of integration package which flags termination of the integration.
SDATA	Block data for Space Shuttle weight summary data.
SEARCH	Searches a table of values for a unique value.

SIMUL	Calculates coefficients for quadratic equation used to relate specific impulse to thrust throttle level.
SPLINE	Third order spline interpolation routine.
SUM15STG	Generates output summary tables for Liquid stage configuration.
SUMARY	Generates output summary tables for Space Shuttle configuration.
SUMHLLV	Generates output summary tables for Strap-on vehicle configuration.
TERMINATE	Terminates execution of MASTRE.
TRANFM	Multiplies 3x3 matrix times a 3x1 vector.
VR71	Vandenburg reference atmophere model. Used with Range Reference atmosphere model.
WINDIN	Allows selection of eastern or western test range mean monthly wind.

Section 3.0
Narrative Subroutine Descriptions

The following subsections define detail information about each of the subroutines and functions found in the MASTRE Program. Except for the first subsection, the subsections will be ordered in an alphabetic format of the name of the subroutine or function. Entry routines found in subroutines ATILT, BTHREV, and FUNISP will be included in the descriptions of these subroutines and will not be specified as separate subroutines.

3.1 Subroutine MASTRE

3.1.1 Purpose

Program MASTRE is the main program and controls the logical flow of the program by calling the major subroutines.

3.1.2 Variable Listing

3.1.3 Subroutines Called:

Name	Function
AFORND	Calculates end conditions at the end of the forward trajectory
AFORUN	Controls and sets up the forward trajectory
ANEWCH	Calculates and applies the updates for the control parameters
B0PTBL	Input routine for output parameter summary tables in subroutine B1PTBL
B1PTBL	Prints output parameter summary tables
BADLX	Calculates partial derivatives by using forward differences method
BAKRN	Controls and sets up backward trajectory
FUNISP	Calculates specific impulse based on throttle level
INITIAL	Controls initial inputs from user
MENU	Controls execution modes of operation from user input

3.1.4 Calling Subroutines:

None

3.1.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
char_vel(5)	Characteristic velocity	m/sec	ainit	input	Global	multi
choice	Input flag				Local	
chvel	Characteristic velocity	m/sec	mastre	input	Global	agen1
covl(7)	Lagrangian multipliers			output	Global	covl
delvp	Delta velocity based on characteristics velocity	m/sec		output input	Global	delvp
demand	Logical indicating the operational mode		initial	output	Global	ccc
dold	Date of update name				Local	
engdat(8,15)	Engine data matrix		ainit	input	Global	agen1
fprfac	Flight performance reserve factor		ainit	input	Global	agen1
fpr_fac(5)	Factor for calculation of flight performance reserve (branch option)		ainit	input	Global	multi
gzero	Gravitational constant (9.80665 m/sec)	m/sec^	ainit	input	Global	const
ipr	Denotes the thrust event from which fuel propellant reserve (FPR) is calculated			input output	Global	ipr
iprop(6)	Propulsion cutoff flags (stage dependent)		ainit	input	Global	ipr
istat	Status index				Local	
jo	Output file index		initial	input	Global	ccc
last	Previous index used in atmospheric routine			output	Global	enput
lbmm	Denotes the simulation of liquid booster module			output	Global	lonoff
naltw	Index used when calculating launch azimuth partial derivatives			output	Global	naltw
ncase	Case number			output	Global	bopcom
nmax	Iteration counter		mastre	output input	Global	agen3
nom1	Index indicating first iteration pass			input output	Global	bgen3
noss	Number of constraints		mastre	output input	Global	bgen3
ntcn	Number of intermediate constraints		ainit	input	Global	rest
ntime	Case number identification used in subroutine ainit to reduce input requirements for multiple cases			output	Global	ntime
prk4	Temporary variable used in calculation of performance reserve option		aint	input	Global	ipr

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
prk5(5)	Same as prk4 but stage dependent (used with branch trajectory option)		mastre	output	Global	ipr
t	Time from lift-off	sec	dpir	output	Global	forint
timer_addr	Time address				Local	
timer_data	Time data				Local	
vers	Version name				Local	

```
C*********************************************************************
C
      PROGRAM MASTRE
C
C     MAIN PROGRAM
C
C      VERSION 1.01 8/10/90
C
C       USES WEIGHT NAMELIST
C
C       SP
C
C*********************************************************************
C
C234567890123456789012345678901234567890123456789012345678901234567890123456789012
C         1         2         3         4         5         6         7
C
C*********************************************************************
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      LOGICAL DEMAND,GRAPH,LBMM,GLIMIT,QFLG,VISC
C
      CHARACTER*36 VERS,DOLD
      CHARACTER CHOICE
      CHARACTER*3 MNTH
      CHARACTER*6 MISION,ATMOSN,NAME
      CHARACTER*12 OFFICE,DATE,header
      character*20 srid
      CHARACTER*48 TITLE
      CHARACTER*60 HEAD
C
      COMMON/CCC/GRAPH,JO,DEMAND

      COMMON/AERO/NTALPH(2),NTBETA(2),NTMACH,TALPH(11,2),TALYH(11,2),
     *TXMAC(28),CAA(15,11,6),CNA(15,11,6),CMA(15,11,6),CYA(15,11,6),
     *CXA(15,11,6),CLA(15,11,6)

      COMMON/AERODI/AEROD(30,17),AEROB(50,6,3),TOMACH(25),DELCA(25),
     *PNMONE(15),BKFABT(50,2),BKFTHR(50,2),DELCN(25),DELCM(25)

      COMMON/AGEN1/TIME(2,16),TAUT(15),TAUW(15),TZERO,TLIFT,TTILT,TMINH,
     *DTZ,TQ,TL,XMAUG,TNE(6,15),ENGDAT(8,15),S(15),WD(15),WJET(15),
     *HEAD,PRINT(15),STEP(15),HNOM(15),HMAX(15),PROP(6),TBEGR,TENDR,
     *CHRDOT,CHIRO,FAZ,BO(3,2),CBAXIL,TCHIR,VRCUT,FPRFAC,CHVEL

      COMMON/AGEN2/AA,SA,CA,ALF,ALFY,CHIP,SCHIP,CCHIP,CHIY,SCHIY,CCHIY,
     *CHIR,SCHIR,CCHIR,STH,CTH,STHL,CTHL,DPHIZ,RTHE,R,VR,XM,SW,SU,SV,Q,
     *VIV(25),DVARS(13),DELXDW(15,3),DELXDR(7,5),DELXD(15,7),WZERO,ALT,
     *XMACH,QDOT,FAA,FAN,A(3,3),CASE,CT2,UMF,A12W,A22W,A32W,XJEXT,BEU,
     *BYL,BEL,BHMAX(15),BHMN,BSTEP(15),HNMB,HMXB,TPOLY,XMIAD

      COMMON/AGEN3/ITHR,IWD,IXR,JUMP,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISION,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)
```



```
      COMMON/AGEN4/QALPHA,QBETA,QCN,STH1,CTH1,XJX(3),XJV(3)

      COMMON/AGEN5/CHIBAS,CHPBAS,CHYBAS,CHISAV,TXX,TYY,TZZ,AMX,AMY,AMZ,
     *TMZ,THMFA,SIDE,WMAG,AZW,FANC,FLATC,FOM,VISC

      COMMON/AFPRN/VI,GAM,C1,C3,XINC,XNOD,XMLB,THETG,THETA,PHI,RNGO,RNGA
     *N,AZI,AZRE

      COMMON/ALAT/ALAT,ALONGO,ALTLS,NTABLE

      COMMON/ARDC/PAT(14),ATMOSN,IATMOS,HBIAS,HAERO

      COMMON/ASTOR/ASTOR(4,67)

      COMMON /ATMOS / ATMTBL(110,6),MTHRRA,MATM1,MATM2,MNTH

      COMMON/AUTO/KDB(40),SAVCP(40),XLAMB(40,20),DP2

      COMMON/AVGGP/XGPA,YGPA,ZGPA

      COMMON/BAKE/WISS(20,20),YLF(7,20)

      COMMON/BGEN3/NOSS,JTB,NENT,NACT,NOM1,NP(7)

      COMMON/BLAMB/YLBT(15,20),YLBW(15,20),GAPHI(20),PROD(2,20)

      COMMON/BODY/TBDWT(15,2),TXCG(15,2),TYCG(15,2),XLEN(2),XREF(2),
     *YREF(2),KP(2),KY(2),AP(10,2),AY(10,2)

      COMMON/BOPCOM/NCASE,OFFICE,TITLE,DATE,SRID(30),NOTAB,NERR

      COMMON /CISP  / COISP(3,2)

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM

      COMMON/CONTRO/TTBL(30),CPTBL(30),CYTBL(30),TOBL(210),
     *CPOTBL(210),CYOTBL(210)

      COMMON/COVL/COVL(7)

      COMMON/DELFAB/DELALT(25),DELFAB(25),M30,M31,ITOL(15)

      COMMON/DELPAR/SPRADB,SYRADB

      COMMON/DELVP/DELVP,DELVG

      COMMON/DRHO/DRHO(19),E1,WIPPB,WGNU

      COMMON/ELEVON/EMACH(25,2),ELEVON(25,4),M10,M11

      COMMON/ENPUT/PSIRST(6,5),HEADER(20),NVRST(5),KCDRES(6,5),LAST

      COMMON/FMXX/FMXX,MSW,WG,SPOA
```

```
      COMMON/FORINT/HBANK(2),T,TT,W,U,V,X,Y,Z,XMI,ZAP(18),DVAR(25),FSAVE
     *(625),NTRG1,TV1,NTRG2,TV2,NTRG3,TV3,NTRG4,TV4,NTRG5,TV5,NTRG6,TV6,
     *NTRG7,TV7,NTRG8,TV8,NTRG9,TV9,NTRG10,TV10,NTRG11,TV11,NTRG12,TV12,
     *NTRG13,TV13,KIND,KRDER,EU,EL,AYL,HMX,HMN,HNM

      COMMON /HEADUP/ HDMACH,HDCHRD

      COMMON/IMP/GAPTB(210,20),GSPTB(210,20),AHTB(210),SITB(210),TS(7)
     *,DELTS(7),KJTB

      COMMON/IPR/IPR,WPMX,ILAST,KDT(15),PRK4,PRTOT,iprop(6),prk5(5)

      COMMON/ISTART/JSTART(6),ISTART(2),STARTV(40,2),SVXINC,SVXNOD,SVAZRE

      COMMON/JUMPST/JUMPST,TSAVET,WZERST

      COMMON/LPRINT/LPRINT,INMAX,PREDP,QY,TOL

      COMMON/LONOFF/LBMM

      COMMON/LBM/TIMLBM(30),THRLBM(30),WDTLBM(30),PLBM(2),MLBM1,MLBM2,
     *ALBM(2),BLBM(2),CLBM(2),VLBM(2)

      COMMON/MPS/TIMMPS(15),FTBL(15),THROT(15),GLIM(15),TAUALT(5),
     *TAUTBL(5),MPS1,MPS2,TGLIM(10,15),GLIMIT,tau_const(10,5),jthrot(15)

      common/multi/char_vel(5),fpr_fac(5),w_margin(2)

      COMMON/NAME/NAME(128),WORD(160)

      COMMON/NALTW/NALTW

      COMMON/NTIME/NTIME

      COMMON/OLOW/OLOW,WINT

      COMMON/OMSRCS/FOMS(15),FRCS(15),WDOMS(15),WDRCS(15),WDDUMP(15)

      COMMON/OUTPUT/OUTTH(6),OUTTV(6),OUTISP(6),OUTWD(6)

      COMMON/QMAX/QMAX,QFLG,QPEN,MAXQ

      COMMON/REST/JSTR,NCNRS(5),NTCN,TR(5),LSB,NF8,NOPAR,WIBT(15)

      COMMON/RESTX/RESTX(10,5)

      COMMON/RICONT/TTABLE(20),SIGMA,SIGTAB(20),M15,M16

      COMMON/RTLS/VRINT(2),VSLOPE(2)

      COMMON/SRM/SRMTTB(600),SRMFTB(600),SRMWDT(600),SRMAE(600),MSRB

      COMMON/STAUT/STAUT(15)

      COMMON/TAULIM/TAULIM,NTAU,NBETA,TIMTAU
```



```
      COMMON/TCL/TCL(6),TVL(6),TLL(6),RATIO

      COMMON/VAERO/VAEROD(10,19),M18,M19,MBS18,MBS19

      COMMON/VCHSTG/VCHSTG

      COMMON/VIC/VIC(2),XK1(2),XK2(2),XK3(2),XK4(2),LK

      COMMON/WIND/ALTTBL(100),WTBL(100,2),NWIND

      COMMON/XTRA/XMD(15),CFR(15),F(15),CISP(15)

      COMMON/YLI/YLINT(7,20),PHITE(20),PSIREQ(20),KCDPHI(20),PSTR(20)
C
C       THESE WERE ADDED TO GIVE A CPU TIMER
C       SAM POWELL 7/14/89
C

      INTEGER*4 TIMER_ADDR,TIMER_DATA,LIB$DATE_TIME

      INTEGER*4 LIB$INIT_TIMER,LIB$SHOW_TIMER,LIB$FREE_TIMER

      EXTERNAL LIB$INIT_TIMER,LIB$SHOW_TIMER,LIB$FREE_TIMER
C
      DATA LBMM/.FALSE./
C                                                                       MAIN  92
C
C***********************************************************************
C
C     THIS PROGRAM WAS FORMULATED AND PROGRAMMED                        MAIN  94
C     DURING 1973 AND 1974 BY J.T.LYONS OF
C     NORTHROP SERVICES, INC., HUNTSVILLE, ALABAMA.                     MAIN  96
C                                                                       MAIN  99
C     FURTHER MODICATIONS WERE IMPLEMENTED BY
C     J.T.LYONS OF DYNETICS,INC HUNTSVILLE,ALA
C     DURING THE FALL OF 1980
C
C
C       EXTENSIVE MODIFICATIONS HAVE BEEN ADDED OVER THE
C       YEARS BY SAM POWELL EP55
C
C     Additional modifications by J.T. Lyons of Dynetics
c     Inc. are currently underway as of April 1, 1991.
c
C***********************************************************************
C
C************MASTRE************
C
C     M   MINIMUM HAMILTONIAN (MIN-H)
C     A   ASCENT
C     S   SHUTTLE
C     TR  TRAJECTORY
C     E   EVALUATION
C         PROGRAM
C
C                                                                       MAIN 111
```

```
C
C*********************************************************************
      VERS='   VERSION NUMBER 2.00'
      DOLD='  Date of last update 6/19/92'

      WRITE(6,'(a36)') VERS
      WRITE(6,'(a36)') DOLD

      ISTAT=LIB$INIT_TIMER(TIMER_ADDR)
C
      CALL INITIAL
C
      IF( DEMAND ) THEN

          WRITE(6,*) ' '
          WRITE(6,*) ' DOUBLE PRECISION MASTRE PROGRAM'
          WRITE(6,*) ' ** CHOOSE ONE OF THE FOLLOWING:'
          WRITE(6,*) '   1) BATCH'
          WRITE(6,*) '   2) DEMAND'

          WRITE(6,4444)
 4444     FORMAT(1H ,'CHOICE >',$)

          READ(5,'(a1)') CHOICE
C
          IF( CHOICE.EQ.'2' ) THEN
              JO=31
              DEMAND = .TRUE.
          ELSE
              DEMAND = .FALSE.
          END IF
C
      END IF
C
C*********************************************************************
C
      NTIME=1
      NCASE=1
C
C*********************************************************************
C                                                                      MAIN 203
C
C     OBTAIN INPUT AND INITILIZATION VALUES
C
C     READ THE INPUT                                                   MAIN 204
C
 2    CONTINUE

      ISTAT=LIB$SHOW_TIMER(TIMER_ADDR,,,TIMER_DATA)
C
      CALL MENU
C
C*********************************************************************
C
```



```
C     INTEGRATE THE TRAJECTORY                                         MAIN 224

3     WRITE(31,'(a36)') VERS
      WRITE(31,'(a36)') DOLD
C
      IF(IPR.ne.0.and.NOM1.gt.5) then

         DELVP=FPRFAC*CHVEL
         PRK4=1.-EXP(-DELVP*ENGDAT(7,1)/ENGDAT(6,1))

      endif

      do 10 i=1,5

      if(iprop(i).ne.0) then

         prk5(i)=exp(fpr_fac(i)*char_vel(i)/(gzero*funisp(1.,i)))

      endif

10    continue

      CALL AFORUN

      CALL AFORND                                                      MAIN 226

      IF(NMAX.gt.0) then

   15    CALL BAKRN

         WRITE(JO,30) T,(COVL(I),I=1,7)
30       FORMAT(/1X,11HCOV LAMBDAS/,3X,8E14.6)

         CALL BADLX

         IF(naltw.ne.0) then

4           NOSS=NOSS-NTCN

            CALL AFORUN

            CALL AFORND

            NOSS=NOSS+NTCN

            CALL BADLX

            if(naltw.ne.0) go to 4

         endif

         CALL ANEWCH

         NMAX=NMAX-1

         GO TO 3
```

```
      endif
C
C*************************************************************************
C
      NMAX=1                                                        MAIN 565
      CALL B0PTBL
      CALL B1PTBL
C
C*************************************************************************
C
      IF(DEMAND .OR. LAST.EQ.1) GO TO 2
C
C*************************************************************************
C
      STOP                                                          MAIN 572
C
      END                                                           MAIN 573
```

3.2 Subroutine ACNTRO

3.2.1 Purpose

The purpose of subroutine ACNTRO is to calculate the current values of the pitch and yaw attitude angles during the min-H phase of flight. Subroutine ACNTRO determines, via linear interpolation, the values of the pitch and yaw attitude angles at discrete time points.

3.2.2 Variable Listing

3.2.3 Subroutines Called:

Name	Function
AMULG	Performs linear interpolation

3.2.4 Calling Subroutines:

Name	Function
ADER1	Calculates time dependent portion of equations of motion
BDR1I	Calculates equations of motion for backward trajectory

3.2.5 Fortran Listing

acntro

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
chir	Roll attitude angle	rad		output	Global	agen2
chiy	Yaw attitude angle	rad	acntro	output	Global	agen2
cpotbl(210)	Pitch attitude table during min-H	rad	anewch	input	Global	contro
cyotbl(210)	Yaw attitude table during min-H	rad	anewch	input	Global	contro
i	Index				Local	
ihead	Roll attitude flag			output	Global	agen3
itab	Table index for attitude tables				Local	
kcytab	Attitude control table index		ainit	input	Global	agen3
kwta(7)	Flag indicating pitch only (1) or pitch and yaw (2) attitude control		ainit	input	Global	agen3
nn	Number of points in attitude table				Local	
np(7)	Number of points in attitude tables during min-H phases		ainit	input	Global	bgen3
pi	Pi constant (3.14159265)		ainit	input	Global	const

```
      SUBROUTINE ACNTRO                                                 ACNT   1
c*********************************************************************
C       Determines the values of pitch and yaw attitude angles from
c       attitude tables
c*********************************************************************
C       Only called during MINH phase calls AMULG to do table lookup
C       called by ADER1(forward) & BDR1I(backward)
c*********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      CHARACTER*60 HEAD
      CHARACTER*6 MISION

      COMMON/AGEN2/AA,SA,CA,ALF,ALFY,CHIP,SCHIP,CCHIP,CHIY,SCHIY,CCHIY,
     *CHIR,SCHIR,CCHIR,STH,CTH,STHL,CTHL,DPHIZ,RTHE,R,VR,XM,SW,SU,SV,Q,
     *VIV(25),DVARS(13),DELXDW(15,3),DELXDR(7,5),DELXD(15,7),WZERO,ALT,
     *XMACH,QDOT,FAA,FAN,A(3,3),CASE,CT2,UMF,A12W,A22W,A32W,XJEXT,BEU,
     *BYL,BEL,BHMAX(15),BHMN,BSTEP(15),HNMB,HMXB,TPOLY,XMIAD

      COMMON/AGEN3/ITHR,IWD,IXR,JUMP,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISION,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)

      COMMON/BGEN3/NOSS,JTB,NENT,NACT,NOM1,NP(7)

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM

      COMMON/CONTRO/TTBL(30),CPTBL(30),CYTBL(30),TOBL(210),CPOTBL(210),
     *CYOTBL(210)

      COMMON/FORINT/HBANK(2),T,TT,W,U,V,X,Y,Z,XMI,ZAP(18),DVAR(25),FSAVE
     *(625),NTRG1,TV1,NTRG2,TV2,NTRG3,TV3,NTRG4,TV4,NTRG5,TV5,NTRG6,TV6,
     *NTRG7,TV7,NTRG8,TV8,NTRG9,TV9,NTRG10,TV10,NTRG11,TV11,NTRG12,TV12,
     *NTRG13,TV13,KIND,KRDER,EU,EL,AYL,HMX,HMN,HNM

      common/mult/nchiot(6),tbias(6),nstage,nstg(15),wf(5),ibranch

      COMMON/TABLK/MCON(7)

c*********************************************************************
C     GET CHIP AND CHIY AS FUNCTIONS OF TIME                            ACNT  22
c*********************************************************************
        I=KWTA(KCYTAB)-1
```



```
        NN=NP(KCYTAB)
        IF(NN.EQ.0) RETURN
c*********************************************************************
c       kcytab indicates which group of tables will be used
c*********************************************************************
        itab=30*(kcytab-1)+1

        call amulg(i,mcon(kcytab),nn,t-tbias(nstage),tobl(itab),chip,
     *  cpotbl(itab),chiy,cyotbl(itab))

        if(i.eq.1) CHIY=0.

        chir=0.
        if(ihead.eq.1) chir=pi

        RETURN

        END
```

3.3 Subroutine ACSTOP

3.3.1 Purpose

The purpose of this subroutine is to calculate values and partial derivatives of the terminal and intermediate constraints. The partial derivatives are used as the initial values in the backward integration of the adjoint equations.

3.3.2 Variable Listing

3.3.3 Subroutines Called:

None

3.3.4 Calling Subroutines:

Name	Function
AFORND	Defines parameters at the terminal and at intermediate points on the forward trajectory

3.3.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a(3,3)	Tranformation matrix from equatorial to plumbline coordinate systems		aforun	input	Global	agen2
a12w	Plumbline X component of earth spin axis		aforun	input	Global	agen2
a22w	Plumbline Y component of earth spin axis		aforun	input	Global	agen2
a32w	Plumbline Z component of earth spin axis		aforun	input	Global	agen2
b	Temporary storage for velocity intercept point	m/sec			Local	
c3	Twice the energy				Local	
cenc	Cosine of inclination angle				Local	
cinc	Cosine of inclination angle				Local	
cmue	Gravitiational constant	m^3/se		input	Global	const
cnmm	Conversion from meters to nautical miles	nm/m			Local	
comt	Cosine of earth rotational angular rate times time				Local	
cosb	Cosine of heading angle				Local	
crange	Cosine of range angle				Local	
ct2	Squared value of cth			output input	Global	agen2
cth	Cosine of colatitude			output input	Global	agen2
cthc	Cosine of launch colatitude				Local	
cthl	Cosine of launch colatitude			input	Global	agen2
d	Direction cosine matrix				Local	
dr	Radius magnitude				Local	
drngdt(2)	Time derivative of range from launch site			output input	Global	drngdt
drthe	Partial derivative of altitude with respect to colatitude				Local	
dur	U Component of relative velocity (excluding winds)	m/sec			Local	
dv	Squared value of inertial velocity magnitude	m^2/se			Local	
dvr	V Component of relative velocity (excluding winds)	m/sec			Local	
dwr	W Component of relative velocity	m/sec			Local	

Variable	Variable Definition (excluding winds)	Units	Source	I/O	Status	Common Block
e	Slope coefficient for relative velocity				Local	
flat	Earth flattening coefficient		ainit	input	Global	const
hratec	Aerodynamic heating rate	btu/sec	ader	input	Global	comnew
i	Index for array location				Local	
lk	Index indicating which orbital velocity data to use as terminal or intermediate constraints			input output	Global	vic
m	Constraint index				Local	
maxhat	Index used to indicate the number of the constraint used for the maximum heating constraint			output	Global	heat
maxq	Index for maximum dynamic pressure constraint			output	Global	qmax
n	First order partial derivatives of the velocity components in the local coordinate system				Local	
nd	Temporary array				Local	
omega	Earth's spin rate	rad/sec	ainit	input	Global	const
omt	Earth's rotational rate times time	rad			Local	
p	Partial derivative of the relative velocity components with respect to the state				Local	
phi	Longitude	deg			Local	
qpen	Maximum value of dynamic pressure			input	Global	qmax
r	Radius from earth's center	m	ader	output input	Global	agen2
r2	Squared value of radius	m^2			Local	
rangle	Range angle	rad			Local	
re	Earth's radius	m	ainit	input	Global	const
rsth	Temporary variable				Local	
rthe	Current earth radius			output input	Global	agen2
save_mass	Storage variable for final mass used in branch option		athrev	input	Global	save_mass
senc	Sine of inclination angle				Local	
sinb	Sine of heading angle				Local	
sing	Sine of inertial flight path angle				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
somt	Sine of earth rotation rate times time				Local	
state	State variable array (interpolated)				Local	
sth	Sine of colatitude			output input	Global	agen2
sthc	Sine of launch colatitude				Local	
sthl	Sine of launch colatitude			input	Global	agen2
t	Time from lift-off			input	Global	cstop
ta	Temporary variable				Local	
tb	Temporary variable				Local	
temp	Temporary variable				Local	
temp1	Temporary variable				Local	
temp2	Temporary variable				Local	
u	Y component of plumbline inertial velocity vector	m/sec	dpir	input	Global	cstop
umf	One minus the earth flattening coefficient		ainit	input	Global	agen2
umfc	Temporary variable				Local	
ure	U component of earth relative velocity vector				Local	
us	U component of inertial velocity vector				Local	
v	Z component of plumbline inertial velocity vector	m/sec	dpir	input	Global	cstop
vale	Value of constraint				Local	
vi	Magnitude of inertial velocity vector	m/sec			Local	
vi2	Squared value of inertial velocity magnitude	m^2/se			Local	
vic	Circular velocity used to calculate oblate earth model velocity option (KCDRES and/or KCDPHI=14)	m/sec			Local	
vre	V component of earth relative velocity vector	m/sec			Local	
vrint(2)	Intercept point for velocity versus range constraint		ainit	input	Global	rtls
vs	V component of inertial velocity vector	m/sec			Local	
vslope(2)	Slope coefficient for velocity versus range constraint		ainit	input	Global	rtls
vvic(2)	Cutoff velocity based on two body mechanics			input	Global	vic

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
vvr	Reltive velocity (excluding winds)				Local	
vxk1(2)	Temporary variable for calculation of oblate model corrections		afornd	input	Global	vic
vxk2(2)	Temporary variable for calculation of oblate model corrections		afornd	input	Global	vic
vxk3(2)	Temporary variable for calculation of oblate model corrections		afornd	input	Global	vic
vxk4(2)	Temporary variable for calculation of oblate model corrections		afornd	input	Global	vic
w	X component of plumbline inertial velocity vector			input	Global	cstop
wre	W component of relative velocity vector	m/sec			Local	
ws	W component of inertial velocity vector	m/sec			Local	
x	X component of plumbline position vector			input	Global	cstop
xk1	Coefficient for calculation of oblate effects on velocity				Local	
xk2	Coefficient for calculation of oblate effects on velocity				Local	
xk3	Coefficient for calculation of oblate effects on velocity				Local	
xk4	Coefficient for calculation of oblate effects on velocity				Local	
xlr	X direction cosine of current launch vector				Local	
xmxm	Final mass	kg		input	Global	cstop
y	Y component of plumbline position vector	m	afornd	input	Global	cstop
yl	Initial adjoint variables				Local	
yl4	Temporary variable				Local	
yl5	Temporary variable				Local	
yl6	Temporary variable				Local	
ylr	Y direction cosine of current launch vector				Local	
z	Z component of plumbline position vector	m	dpir	input	Global	cstop
zlr	Z direction cosine of current launch vector				Local	

```
      SUBROUTINE ACSTOP(M,VALE,YL,I)

c*****************************************************************************

C      EVALUATES THE PAYOFF AND CONSTRAINTS AND THE INITIAL ADJOINT VARIABLES
C          M      = CONSTRAINT INDEX
C          VALE   = VALUE OF CONSTRAINT
C          YL     = INITIAL ADJOINT VARIABLES
C          I      = INDEX FOR ARRAY LOCATION

c*****************************************************************************

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      LOGICAL DEMAND,GRAPH,GLIMIT,QFLG

      REAL*8 N(6,6),ND(3)

      CHARACTER*60 HEAD
      CHARACTER*6 MISION

      COMMON/AGEN2/AA,SA,CA,ALF,ALFY,CHIP,SCHIP,CCHIP,CHIY,SCHIY,CCHIY,
     *CHIR,SCHIR,CCHIR,STH,CTH,STHL,CTHL,DPHIZ,RTHE,R,VR,XM,SW,SU,SV,Q,
     *VIV(25),DVARS(13),DELXDW(15,3),DELXDR(7,5),DELXD(15,7),WZERO,ALT,
     *XMACH,QDOT,FAA,FAN,A(3,3),CASE,CT2,UMF,A12W,A22W,A32W,XJEXT,BEU,
     *BYL,BEL,BHMAX(15),BHMN,BSTEP(15),HNMB,HMXB,TPOLY,XMIAD

      common/comnew/hratec

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM

      COMMON/CSTOP/W,U,V,X,Y,Z,XMXM,T,HEAT,ZZZ

      COMMON/DRNGDT/DRNGDT(2)

      COMMON/HEAT/MAXHAT

      COMMON/QMAX/QMAX,QFLG,QPEN,MAXQ

      COMMON/RTLS/VRINT(2),VSLOPE(2)

      common/save_mass/save_mass

      COMMON/VIC/VVIC(2),VXK1(2),VXK2(2),VXK3(2),VXK4(2),LK

      DIMENSION D(3,3),STATE(3)

      EQUIVALENCE (STATE,W)

      DIMENSION YL(7,20),p(3,6)

c*****************************************************************************

      IF(I.EQ.1) MAXQ=0
```



```
      IF(m.eq.0) then

c*****************************************************************************

c       INITIALIZE PARTIAL MATRICES and additional calculations

c*****************************************************************************

        R2=X*X+Y*Y+Z*Z                                                  ACST 270
        R=SQRT(R2)                                                      ACST 271

        VI2=W*W+U*U+V*V                                                 ACST 272
        VI=SQRT(VI2)                                                    ACST 273

        DR=DSQRT(X*X+Y*Y+Z*Z)
        DV=W*W+U*U+V*V

        C3=DV-2.*CMUE/DR                                                ACST 276

c       relative velocity components (excluding winds)

        DWR=W-A22W*Z+A32W*Y
        DUR=U-A32W*X+A12W*Z
        DVR=V-A12W*Y+A22W*X

        VVR=DSQRT(DWR*DWR+DUR*DUR+DVR*DVR)                              ACST 280

c       D matrix is calculated which a direction cosine matrix
c       which defines an orthogonial set of vectors in the local
c       vertical(the radius vector)[D(i,2)], the local east[D(i,1)],
c       and local south[D(i,3)] directions.

        D(1,2)=X/R                                                      ACST 281
        D(2,2)=Y/R                                                      ACST 282
        D(3,2)=Z/R                                                      ACST 283

        CTH=A(1,2)*D(1,2)+A(2,2)*D(2,2)+A(3,2)*D(3,2)                   ACST 284
        STH=SQRT(DMAX1(0.D0,1.D0-CTH*CTH))
        RSTH=R*STH                                                      ACST 286

        D(1,1)=(-Y*A(3,2)+Z*A(2,2))/RSTH                                ACST 287
        D(2,1)=(X*A(3,2)-Z*A(1,2))/RSTH                                 ACST 288
        D(3,1)=(-X*A(2,2)+Y*A(1,2))/RSTH                                ACST 289

        D(1,3)=(D(1,2)*CTH-A(1,2))/STH                                  ACST 290
        D(2,3)=(D(2,2)*CTH-A(2,2))/STH                                  ACST 291
        D(3,3)=(D(3,2)*CTH-A(3,2))/STH                                  ACST 292

c       The inertial velocity vector is now defined in the local
c       coordinate system.

        WS=D(1,1)*W+D(2,1)*U+D(3,1)*V                                   ACST 293
        US=D(1,2)*W+D(2,2)*U+D(3,2)*V                                   ACST 294
        VS=D(1,3)*W+D(2,3)*U+D(3,3)*V                                   ACST 295

C       Calculate the N matrix which a 6X6 matrix of first order
```

```
c       partial derivatives of the velocity components in the local
c       coordinate system and spherical components (longitude, radius,
c       and colatitude) with respect to the state variables. (See
c       Appendix in the Engineering Manual for additional information)

        DO 1002 k=1,3                                                   ACST 298

        ND(k)=D(k,3)*CTH+D(k,2)*STH                                     ACST 299
        N(4,k+3)=D(k,1)/RSTH                                            ACST 300
        N(5,k+3)=D(k,2)                                                 ACST 301
        N(6,k+3)=D(k,3)/R                                               ACST 302

        DO 1002 j=1,3                                                   ACST 303
        N(k,j)=D(j,k)                                                   ACST 304
1002    N(k+3,j)=0.                                                     ACST 305

        TEMP=WS*CTH/STH                                                 ACST 306

        N(1,4)=(A(3,2)*U-A(2,2)*V-WS*ND(1))/RSTH                        ACST 307
        N(1,5)=(A(1,2)*V-A(3,2)*W-WS*ND(2))/RSTH                        ACST 308
        N(1,6)=(A(2,2)*W-A(1,2)*U-WS*ND(3))/RSTH                        ACST 309

        N(2,4)=(W-D(1,2)*US)/R                                          ACST 310
        N(2,5)=(U-D(2,2)*US)/R                                          ACST 311
        N(2,6)=(V-D(3,2)*US)/R                                          ACST 312

        N(3,4)=(D(1,1)*TEMP-D(1,3)*US)/R                                ACST 313
        N(3,5)=(D(2,1)*TEMP-D(2,3)*US)/R                                ACST 314
        N(3,6)=(D(3,1)*TEMP-D(3,3)*US)/R                                ACST 315

        WRE=D(1,1)*DWR+D(2,1)*DUR+D(3,1)*DVR
        URE=D(1,2)*DWR+D(2,2)*DUR+D(3,2)*DVR
        VRE=D(1,3)*DWR+D(2,3)*DUR+D(3,3)*DVR

c       Calculate the P matrix which is the partial derivative of the
c       relative velocity components with respect to the state.

        P(1,1)=D(1,1)
        P(1,2)=D(2,1)
        P(1,3)=D(3,1)

        P(1,4)=-D(2,1)*A32W+D(3,1)*A22W+(A(3,2)*DUR-A(2,2)*DVR-
     *          WRE*ND(1))/RSTH
        P(1,5)=+D(1,1)*A32W-D(3,1)*A12W+(A(1,2)*DVR-A(3,2)*DWR-
     *          WRE*ND(2))/RSTH
        P(1,6)=-D(1,1)*A22W+D(2,1)*A12W+(A(2,2)*DWR-A(1,2)*DUR-
     *          WRE*ND(3))/RSTH

        TEMP=WRE*CTH/STH

        P(3,1)=D(1,3)
        P(3,2)=D(2,3)
        P(3,3)=D(3,3)

        P(3,4)=-D(2,3)*A32W+D(3,3)*A22W+(D(1,1)*TEMP-D(1,3)*URE)/R
        P(3,5)=+D(1,3)*A32W-D(3,3)*A12W+(D(2,1)*TEMP-D(2,3)*URE)/R
```



```
        P(3,6)=-D(1,3)*A22W+D(2,3)*A12W+(D(3,1)*TEMP-D(3,3)*URE)/R

        RETURN

      endif
c*****************************************************************************
C       PAYLOAD (final mass)
c*****************************************************************************
      if(m.eq.1) then

        VALE=XMXM+save_mass

        YL(7,I)=1.

        RETURN

      endif
c*****************************************************************************
C       INERTIAL VELOCITY
c*****************************************************************************
      if(m.eq.2) then

        VALE=DSQRT(W*W+U*U+V*V)

        YL(1,I)=W/VALE
        YL(2,I)=U/VALE
        YL(3,I)=V/VALE

        yl(4,i)=0.
        yl(5,i)=0.
        yl(6,i)=0.

        return

      endif
c*****************************************************************************
C       INERTIAL FLIGHT PATH ANGLE
c*****************************************************************************
      if(m.eq.3) then

        SING=US/VI
        TEMP=RAD/(VI*SQRT(DMAX1(0.D0,1.D0-SING**2)))

        VALE=ASIN(SING)*RAD
```

```
         IF(VALE.GT.180.)VALE=VALE-360.

         YL(1,I)=(N(2,1)-SING*W/VI)*TEMP
         YL(2,I)=(N(2,2)-SING*U/VI)*TEMP
         YL(3,I)=(N(2,3)-SING*V/VI)*TEMP

         DO 103 J=4,6
  103    YL(J,I)=N(2,J)*TEMP

         RETURN

      endif
C*****************************************************************************
C        RADIUS
c*****************************************************************************
      if(m.eq.4) then

         VALE=DSQRT(X*X+Y*Y+Z*Z)

         yl(1,i)=0.
         yl(2,i)=0.
         yl(3,i)=0.

         YL(4,I)=X/VALE
         YL(5,I)=Y/VALE
         YL(6,I)=Z/VALE

         return

      endif
c*****************************************************************************
c        Altitude above the oblate earth
c*****************************************************************************
      if(m.eq.5) then

         r=sqrt(x*x+y*y+z*z)
         cth=(a(1,2)*x+a(2,2)*y+a(3,2)*z)/r
         ct2=cth*cth
         rthe=umf*re/(umf*umf*(1.-ct2)+ct2)**.5

         vale=r-rthe

         umfc=flat*(2.-flat)/(re*re*umf*umf)
         drthe=rthe**3*umfc*cth/(r*r)

         yl(1,i)=0.
         yl(2,i)=0.
         yl(3,i)=0.
```



```
         yl(4,i)=x/r-(x*cth-r*a(1,2))*drthe
         yl(5,i)=y/r-(y*cth-r*a(2,2))*drthe
         yl(6,i)=z/r-(z*cth-r*a(3,2))*drthe

         return

      endif
C*****************************************************************************
C        RELATIVE VELOCITY
c*****************************************************************************
      if(m.eq.7.or.m.eq.18) then

         VALE=VVR

         YL(1,I)=DWR/VVR
         YL(2,I)=DUR/VVR
         YL(3,I)=DVR/VVR

         YL(4,I)=(-DUR*A32W+DVR*A22W)/VVR
         YL(5,I)=(DWR*A32W-DVR*A12W)/VVR
         YL(6,I)=(-DWR*A22W+DUR*A12W)/VVR

         IF(M.EQ.7) RETURN

         YL4=YL(4,I)
         YL5=YL(5,I)
         YL6=YL(6,I)

      endif
C*****************************************************************************
C        EARTH RELATIVE AZIMUTH(HEADING ANGLE)
c*****************************************************************************
      if(m.eq.8..or.m.eq.14) then

         VALE=ATAN2(WRE,-VRE)

         TEMP=VRE/(VRE*VRE+WRE*WRE)
         TEMP1=WRE/(VRE*VRE+WRE*WRE)

         DO 80 J=1,6
   80    YL(J,I)=-(TEMP*P(1,J)-TEMP1*P(3,J))*RAD

         VALE=VALE*RAD

         RETURN

      endif
```

```
c*********************************************************************

C       INERTIAL HEADING ANGLE

c*********************************************************************

      if(m.eq.10.or.m.eq.11) then

         VALE=ATAN2(WS,-VS)

         TEMP1=WS/(VS*VS+WS*WS)
         TEMP=VS/(VS*VS+WS*WS)

   32    DO 108 J=1,6
  108    YL(J,I)=-(TEMP*N(1,J)-TEMP1*N(3,J))*RAD

      endif

c*********************************************************************

C       RELATIVE GAMMA

c*********************************************************************

      if(m.eq.9) then

         SING=URE/VVR

         VALE=ASIN(SING)*RAD
         TEMP=RAD/(VVR*SQRT(DMAX1(0.D0,1.D0-SING*SING)))

         YL(1,I)=TEMP*(D(1,2)-DWR*SING/VVR)
         YL(2,I)=TEMP*(D(2,2)-DUR*SING/VVR)
         YL(3,I)=TEMP*(D(3,2)-DVR*SING/VVR)

         YL(4,I)=TEMP*(W/R-SING*(D(1,2)*VVR/R+(A22W*DVR-A32W*DUR)/VVR))
         YL(5,I)=TEMP*(U/R-SING*(D(2,2)*VVR/R+(A32W*DWR-A12W*DVR)/VVR))
         YL(6,I)=TEMP*(V/R-SING*(D(3,2)*VVR/R+(A12W*DUR-A22W*DWR)/VVR))

         return

      endif

c*********************************************************************

C       INCLINATION

c*********************************************************************

      if(m.eq.10.or.m.eq.14) then

         SINB=SIN(VALE)
         COSB=COS(VALE)
         CINC=STH*SINB
```



```
         VALE=ACOS(CINC)*RAD

         TEMP1=SINB*CTH/SIN(VALE/RAD)*RAD
         TEMP2=COSB*STH/SIN(VALE/RAD)

         DO 110 J=1,6
  110    YL(J,I)=-(TEMP1*N(6,J)+TEMP2*YL(J,I))

         if(m.eq.10) return

      endif

c*********************************************************************

C       LINE OF NODES

c*********************************************************************

      if(m.eq.11) then

         TEMP1=A(1,1)*D(1,2)+A(2,1)*D(2,2)+A(3,1)*D(3,2)
         TEMP2=A(1,3)*D(1,2)+A(2,3)*D(2,2)+A(3,3)*D(3,2)

         PHI=ATAN2(TEMP1,TEMP2)

         VALE=PHI+ATAN2(CTH*WS,VS)
         VALE=VALE*RAD

         TA=WS*VS*STH/(VS**2+(WS*CTH)**2)
         TB=(VS**2+WS**2)*CTH/(VS**2+(WS*CTH)**2)

         DO 111 J=1,6
  111    YL(J,I)=(N(4,J)-TB*YL(J,I)/RAD-TA*N(6,J))*RAD

         RETURN

      endif

c*********************************************************************

C       ORBITAL VELOCITY CORRECTION

c*********************************************************************

      if(m.eq.14) then

         SENC=SIN(VALE/RAD)
         CENC=COS(VALE/RAD)
         VIC=VVIC(LK)

         XK1=VXK1(LK)
         XK2=VXK2(LK)
         XK3=VXK3(LK)
         XK4=VXK4(LK)

         TEMP=1.+XK1*(XK3+XK4*SENC*SENC-CTH*CTH*XK2)
```

```
         TEMP1=XK1*XK2*CTH*STH
         VALE=VIC*SQRT(TEMP)-VI
         TEMP2=XK1*XK4*SENC*CENC

         DO 140 J=1,3
  140    YL(J,I)=VIC*(TEMP2*YL(J,I)/RAD)/SQRT(TEMP)-STATE(J)/VI

         DO 141 J=4,6
  141    YL(J,I)=VIC*(TEMP2*YL(J,I)/RAD+TEMP1*N(6,J))/SQRT(TEMP)

         RETURN

      endif

c*************************************************************************

C        DYNAMIC PRESSURE PENALTY FUNCTION

c*************************************************************************

      if(m.eq.15) then

         VALE=QPEN
         MAXQ=I

         RETURN

      endif

c*************************************************************************

c        heat rate constraint

c*************************************************************************

      if(m.eq.16.or.m.eq.17) then

         VALE=HRATEC
         MAXHAT=I

         RETURN

      endif

C*************************************************************************

C        RANGE FROM LAUNCH SITE

c*************************************************************************

      if(m.eq.18.or.m.eq.19.or.m.eq.20) then

         OMT=OMEGA*T
         SOMT=SIN(OMT)
         COMT=COS(OMT)
         STHC=STHL
```



```
         CTHC=CTHL

         XLR=CTHC*A(1,2)+STHC*(A(1,1)*SOMT+A(1,3)*COMT)
         YLR=CTHC*A(2,2)+STHC*A(2,3)*COMT
         ZLR=CTHC*A(3,2)+STHC*(A(3,1)*SOMT+A(3,3)*COMT)

         CRANGE=XLR*D(1,2)+YLR*D(2,2)+ZLR*D(3,2)
         RANGLE=ACOS(CRANGE)
         CNMM=1./1852.
         VALE=RE*RANGLE*CNMM
         TEMP=-RE*CNMM/(R*SIN(RANGLE))

         YL(4,I)=TEMP*(XLR-D(1,2)*CRANGE)+YL4
         YL(5,I)=TEMP*(YLR-D(2,2)*CRANGE)+YL5
         YL(6,I)=TEMP*(ZLR-D(3,2)*CRANGE)+YL6

         DRNGDT(LK)=-RE*STHC*OMEGA*CNMM/SIN(RANGLE)*(D(1,2)*
     *              (A(1,1)*COMT-A(1,3)*SOMT)-D(2,2)*A(2,3)*
     *              SOMT+D(3,2)*(A(3,1)*COMT-A(3,3)*SOMT))

         IF(M.NE.18) return

C     RTLS VELOCITY REQUIREMENT

         B=VRINT(LK)
         E=VSLOPE(LK)
         VALE=B+E*VALE-VVR

         YL(1,I)=-YL(1,I)
         YL(2,I)=-YL(2,I)
         YL(3,I)=-YL(3,I)
         YL(4,I)=E*YL(4,I)
         YL(5,I)=E*YL(5,I)
         YL(6,I)=E*YL(6,I)

         DRNGDT(LK)=E*DRNGDT(LK)

      endif

      RETURN

      END
```

3.4 Subroutine ADER

3.4.1 Purpose

The purpose of this subroutine is to calculate the portion of the equations of motion which are independent of time. These equations are used for the forward integration only. Subroutine ADER1 is combined with this subroutine to define the complete equations of motion. Information for all stages is contained within these equations.

3.4.2 Variable Listing

3.4.3 Subroutines Called:

Name	Function
AGEO	Calculates gravitational accelerations
AMULG	Linear interpolation routine
ATMOSPHERE	Patrick and Vandenburg atmospheric routines
CAERO	Nonlinear atmosphere routine
FUNISP	Calculates specific impulse based on throttle level
RRASPL	Calls subroutine SPLINE to interpolate Reference atmospheric routine parameters
SPLINE	Interpolation routine using spline technique

3.4.4 Calling Subroutines:

Name	Function
AEOSR	Stores the state variables during the forward integration
AFORUN	Controls forward integration
ATHREV	Controls thrust event logic in forward integration
ATILT	Controls tilt-over logic in forward integration
DISPPRT	Controls writing output file for dispersion analyses

3.4.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a(3,3)	Tranformation matrix from equatorial to plumbline coordinate systems		aforun	input	Global	agen2
a12w	Plumbline X component of earth spin axis		aforun	input	Global	agen2
a22w	Plumbline Y component of earth spin axis		aforun	input	Global	agen2
a32w	Plumbline Z component of earth spin axis		aforun	input	Global	agen2
aa	Constants of spline interpolation of second stage aerodynamic coefficients				Local	
abfact	Factor used to define the effects of angle of attack and sideslip angle on base force				Local	
acacf	Coefficient of continuum flow effects to axial force coefficient				Local	
acamf	Coefficient of molecular flow effects to axial force coefficient				Local	
acmcf	Coefficient of continuum flow effects to pitching moment coefficient				Local	
acmmf	Coefficient of molecular flow effects to pitching moment coefficient				Local	
acncf	Coefficient of continuum flow effects to normal force coefficient				Local	
acnmf	Coefficient of molecular flow effects to normal force coefficient				Local	
aero	Interpolated aerodynamic coefficients				Local	
aerob(50,6,3	Base force tables		ainit	input	Global	aerodi
aerod(30,17)	Aerodynamic coefficient tables		ainit	input	Global	aerodi
aerodd	Array of partial derivatives of aerodynamic coefficients with respect to mach number				Local	
af	Constants of spline interpolation of power-on base force				Local	
afab	Constants of spline interpolation of power-on base force				Local	
alf	Angle of attack	rad	ader	output	Global	agen2
alfy	Sideslip angle	rad	ader	output	Global	agen2
alp	Angle of attack	deg			Local	
alt	Altitude	m	ader	output input	Global	agen2

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
altls	Altitude of launch site	m	ainit	input	Global	alat
alttbl(100)	Altitude table for wind tables	m	ainit	input	Global	wind
amx	Aerodynamic moment about vehicle yaw axis	nt-m	ader	output	Global	agen5
amy	Aerodynamic moment about vehicle roll axis	nt-m	ader	output	Global	agen5
amz	Aerodynamic moment about vehicle pitch axis	nt-m	ader	output input	Global	agen5
asto(4,67)	Storage array for weight summary table	lbs	ainit	input	Global	astor
aw	Spline coefficients for wind parameter interpolation				Local	
azw	Wind azimuth	rad	ader	output	Global	agen5
baero	Spline coefficients for aerodynamic coefficient interpolation				Local	
basthr	Base force with respect to thrust				Local	
bb	Constants of spline interpolation of second stage aerodynamic coefficients				Local	
bcacf	Coefficient of continuum flow effects to axial force coefficient				Local	
bcamf	Coefficient of molecular flow effects to axial force coefficient				Local	
bcmcf	Coefficient of continuum flow effects to pitching moment coefficient				Local	
bcmmf	Coefficient of molecular flow effects to pitching moment coefficient				Local	
bcncf	Coefficient of continuum flow effects to normal force coefficient				Local	
bcnmf	Coefficient of molecular flow effects to normal force coefficient				Local	
beta	Sideslip angle	deg			Local	
bf	Constants of spline interpolation of power-on base force				Local	
bfab	Constants of spline interpolation of power-on base force				Local	
bkfabt(50,2)	Coefficient table for alpha-beta factor for power-on base force		ainit	input	Global	aerodi
bkfthr(50,2)	Coefficient tables for booster base force		ainit	input	Global	aerodi
boost	Flag specifying that the booster portion				Local	

Variable	Variable Definition of the flight is being simulated	Units	Source	I/O	Status	Common Block
bw	Spline coefficient for wind table interpolation				Local	
ca	Cosine of launch azimuth		aforun	input	Global	agen2
caalp	Slope of axial force coefficient with respect to angle of attack (equivalent to AERO(1))				Local	
cacf	Coefficient of axial force coefficient with respect to continuum flow effects				Local	
calp	Cosine of desired angle of attack (alf)				Local	
calp1	Temporary variable				Local	
calp2	Temporary variable				Local	
calp3	Temporary variable				Local	
calpha	Cosine of angle of attack				Local	
camf	Coefficient of axial force coefficient with respect to molecular flow effects				Local	
cao	Zero angle of attack axial force coefficient (equivalent to AERO(2))				Local	
cax	Axial force coefficient		ader	output	Global	aprint
cazw	Cosine of wind azimuth				Local	
cbaxil	Constant value of base axial force		ainit	input	Global	agen1
cbeta	Cosine of desired sideslip angle (alfy)				Local	
cc	Spline coefficient for linear aerodynamic interpolation				Local	
ccacf	Coefficient of continuum flow effects to axial force coefficient				Local	
ccamf	Coefficient of molecular flow effects to axial force coefficient				Local	
cchip	Cosine of pitch attitude angle		ader1	output input	Global	agen2
cchir	Cosine of roll attitude angle		ader1	output input	Global	agen2
cchiy	Cosine of yaw attitude angle		ader1	input	Global	agen2
ccmcf	Coefficient of continuum flow effects to pitching moment coefficient				Local	
ccmmf	Coefficient of molecular flow effects to pitching moment coefficient				Local	
ccncf	Coefficient of continuum flow effects to normal force coefficient				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ccnmf	Coefficient of molecular flow effects to normal force coefficient				Local	
cf	Constants of spline interpolation of power-on base force				Local	
cfab	Constants of spline interpolation of power-on base force				Local	
chip	Pitch attitude angle	rad	ader1	output input	Global	agen2
chiy	Yaw attitude angle	rad	ader1	output input	Global	agen2
clbeta	Slope of roll moment coefficient with respect to sideslip angle (equivalent to AERO(3))	/deg			Local	
clcg	Roll moment coefficient at center of gravity				Local	
clo	Zero sideslip angle value of roll moment coefficent (equivalent to AERO(10))				Local	
cmalp	Slope of pitch moment coefficient with respect to angle of attack (equivalent to AERO(4))	/deg			Local	
cmcf	Coefficient of pitching moment coefficient with respect to continuum flow effects				Local	
cmcg	Pitch moment coefficient at center of gravity				Local	
cmmf	Coefficient of pitching moment coefficient with respect to molecular flow effects				Local	
cmo	Zero angle of attack value of pitch moment coefficient (equivalent to AERO(5))				Local	
cmx	Yaw moment coefficient about center of gravity		ader	output	Global	aprint
cmy	Roll moment coefficient about center of gravity		ader	output	Global	aprint
cmz	Pitch moment coefficient about center of gravity		ader	output	Global	aprint
cnalp	Slope of normal force coefficient with respect to angle of attack (equivalent to AERO(6))	/deg			Local	
cnbeta	Slope of yaw moment coefficient with respect to sideslip angle (equivalent to				Local	

Variable	Variable Definition AERO(7))	Units	Source	I/O	Status	Common Block
cnbo	Zero sideslip angle value of roll moment coefficent (equivalent to AERO(12))				Local	
cncf	Coefficient of normal force coefficient with respect to continuum flow effects				Local	
cncg	Yaw moment coefficient at center of gravity				Local	
cnmf	Coefficient of normal force coefficient with respect to molecular flow effects				Local	
cno	Zero angle of attack value of normal force coefficient (equivalent to AERO(8))				Local	
cnp	Normal force coefficient			output	Global	aprint
colda	Cosine of past value of angle of attack (olda)				Local	
coldb	Cosine of past value of sideslip angle (oldb)				Local	
copga	Cosine of orbiter pitch gimbal angle				Local	
cosdb	Cosine of desired minus past sideslip angle				Local	
cosrho	Cosine of engine gimbal angle				Local	
coss	Cosine of sigma attitude angle				Local	
cost	Cosine of theta attitude angle				Local	
coyga	Cosine of orbiter yaw gimbal angle				Local	
cpsi	Cosine of psi attitude angle				Local	
cp_input	Logical indicating center of pressure input		ainit	input	Global	cp_input
cs	Yaw force coefficient			output	Global	aprint
cspga	Cosine of SRB pitch gimbal angle				Local	
csyga	Cosine of SRB yaw gimbal angle				Local	
ct2	Squared value of cth		ader	output input	Global	agen2
cth	Cosine of colatitude		ader	output input	Global	agen2
cw	Spline coefficients for wind parameter interpolation				Local	
cybeta	Slope of side force coefficient with respect to sideslip angle (equivalent to AERO(9))	/deg			Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
cyo	Zero sideslip angle value of side force coefficent (equivalent to AERO(11))				Local	
dcax	Incremental value of axial force coefficient				Local	
dcmz	Incremental value of pitching moment coefficient				Local	
dcnp	Incremental value of normal force coefficient				Local	
ddfab	Partial derivative of incremental value of base drag force with respect to altitude	nt/m			Local	
dens	Density of LOX tank	lbs/in^			Local	
dfab	Incremental value of base drag force	newton			Local	
dinbd	Inboard elevon deflection angle	deg		output input	Global	aprint
doutbd	Outboard elevon deflection angle	deg		output input	Global	aprint
dqdx	Array of partial derivatives of dynamic pressure with respect to state variables				Local	
drag	Aerodynamic drag force	newton	ader	output	Global	aprint
drdr	Partial derivative of density with respect to altitude				Local	
drthe	Partial derivative of altitude with respect to colatitude				Local	
dthrtl	Delta between 1.09 and current value of throttle level				Local	
dvar(25)	Storage array for the state derivatives		ader	output input	Global	forint
dx	Normal component of thrust moment arm	m		output	Global	garbge
dxdy	dx/dy			output input	Global	garbge
dxpr	Normal component of aerodynamic moment reference arm	m			Local	
dy	Longitudinal component of thrust moment arm	m			Local	
dyi	1/dy	/ m		output input	Global	garbge
dypr	Longitudinal component of aerodynamic moment reference arm	m			Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
dz	Lateral component of aerodynamic moment arm	m			Local	
dzi	1/dz	/ m	ader	output input	Global	garbge
e1	X component of local east vector				Local	
e2	Y component of local east vector				Local	
e3	Z component of local east vector				Local	
elein	Inboard elevon angle	rad	ader	output	Global	aprint
eleout	Outboard elevon angle	rad	ader	output	Global	aprint
elevon(25,4)	Elevon angle tables	rad	ainit	input	Global	elevon
engdat(8,15)	Engine data matrix		ainit	input	Global	agen1
engines(15,1	This array indicates which engine from the ENGDAT array is being used for a certain thrust event. (0- not being used, 1-being used)		ainit	input	Global	engines
etirt	External tank inert weight	lbs			Local	
fa	Aerodynamic axial force (negative)	nt			Local	
faa	Aerodynamic axial force	newton		output input	Global	agen2
fab	Base axial force	newton		output	Global	aprint
faborb	Orbiter base axial base force for 109% thrust level				Local	
fabt(4)	Base axail force	newton	ader	output	Global	aprint
fan	Aerodynamic normal force	newton	ader	output	Global	agen2
fanc	Total normal force	newton	ader	output	Global	agen5
fk1thr	Coefficient for calculation of base force				Local	
fk2thr	Coefficient for calculation of base force				Local	
fkalf	Coefficient of base force with respect to angle of attack				Local	
fkbeta	Coefficient of base force with respect to sideslip angle				Local	
flat	Earth flattening coefficient		ainit	input	Global	const
flatc	Total side force	newton		output input	Global	agen5
fmb	Pitching moment component of base force				Local	
fn	Normal component of base force	newton			Local	
fnb	Normal component of base force	newton			Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
fom	Total vehicle sensed acceleration	g's		output	Global	agen5
foms(15)	Thrust table for orbit maneuver system	lbs	ainit	input	Global	omsrcs
frcs(15)	Thrust table for rocket control system	lbs	ainit	input	Global	omsrcs
ftbl(15)	Thrust table for main proplusion system	lbs	ainit	input	Global	mps
fu	Temporary variable				Local	
fw	Temporary variable				Local	
fx	X component of summation of thrust and aerodynamic forces				Local	
fy	Y component of summation of thrust and aerodynamic forces				Local	
fz	Z component of summation of thrust and aerodynamic forces				Local	
g11	Temporary variable in gravitational equations		ageo	input	Global	grav
glim(15)	Acceleration limit	g's	ainit	input	Global	mps
glimit	Logical variable indicating aceleration limit has been achieved		atilt(gli	output	Global	mps
gm	Projected mass at acceleration limit	kg			Local	
gto	Temporary variable in gravitational equation	m/sec^	ageo	input	Global	grav
gx	X component of graviational acceleration	m/sec^			Local	
gy	Y component of graviational acceleration	m/sec^			Local	
gz	Gravitational constant (9.80665 m/sec)	m/sec	ainit	input	Global	const
gzero	Gravitational constant (9.80665 m/sec)	m/sec^	ainit	input	Global	const
haero	Adjusted altitude for base force table interpolation	m		output	Global	ardc
hatmos	Altitude used to interpolate for atmospheric parameters	m			Local	
hbias	Altitude bias	m	ainit	input	Global	ardc
headt	Height of LOX in propellant tank	in			Local	
hlox	Height of LOX in tank	in			Local	
hratec	Aerodynamic heating rate	btu/sec		output	Global	comnew
iatmos	Atmospheric routine indicator		ainit	input	Global	ardc
iconsw	Flag to indicate termination of SRB moment balance		ainit	output	Global	agen3
ifact	Flag indicating the type of attitude used		atilt	input	Global	agen3

Variable	Variable Definition in the booster stage	Units	Source	I/O	Status	Common Block
ipoly	Flag indicating number of polynominals in booster stage		ainit	input	Global	agen3
ithr	Thrust event index number		athrev	input output	Global	agen3
ithro	Index for SSME throttle table		atilt	input	Global	ithro
itol(15)	Flag indicating thrust event use of delta aerodynamic coefficients and base force		ainit	input	Global	delfab
ivrtsw	1st stage attitude option flag		ainit	input	Global	vrwtbl
jthrot(15)	Index indicating number of times the acceleration limit has been reached within a given thrust event		atilt	input	Global	mps
jump	Jump start flag		ainit	input	Global	agen3
kk	Index				Local	
lift	Total lift force	newton	ader	output	Global	aprint
linear_aero	Flag to denote use of linear aerodynamic tables in the first stage		ainit	input	Global	linear_aero
lstge(15)	Aerodynamic coefficient flag to distinguish stages		ainit	input	Global	agen3
m1	Index to indicate place in aerodynamic tables		athrev	input	Global	
m19	Previous index for continuum and free molecular aerodynamic data		athrev	output	Global	vaero
m2	Previous index for aerodynamic coefficient tables for forward trajectory			output	Global	bodyf
m9	Previous index for base axial force table		athrev	output	Global	bodyf
minh	Flag to indicate thrust event which begins min-h attitude optimization		ainit	input	Global	agen3
mload	Counter index for interpolation of LOX parameters				Local	
mmax	Maximum index for mps engines				Local	
mmin	Minimum index for mps engines				Local	
mombal	Flag to specify the use of moment balance equations		ainit	input	Global	mombal
moment_bal	Flag to indicate simulation of moment balance				Local	
mpsflg(15)	Flag indicating the type of MPS thrust simulation		ainit	output	Global	agen3

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
mpsinp(15)	Array used in conjunction with mpsflg=6 to denote which engines (with respect to the ENGDAT array) will use the throttle table		ainit	input	Global	mps1
n	Index				Local	
nchi	Not used				Local	
neng	Number of engines				Local	
nf8	Number of differential equations to be integrated		aforun	input	Global	rest
nmax	Iteration counter		mastre	input	Global	agen3
nobase	Flag indicating that no base force will be calculated after the thrust event specified by nobase		ainit	input	Global	agen3
noevnt(5)	Number of thrust events per stage		ainit	output input	Global	agen3
nstage	Index number of current stage		athrev	input	Global	mult
nsy	Index which specifies the type of engines that will be used for each thrust event				Local	
nsyst(15)	Index array which specifies the type of engines that will be used for each thrust event		ainit	input	Global	agen3
ntilt	Flag indicating end of tiltover phase of flight		atilt	input	Global	ntilt
ntrg7	Liftoff trigger number		atilt	output	Global	forint
numq	not used				Local	
nwind	Number of tabular values in wind parameter tables		ainit	input	Global	wind
odynsf	Orbiter dynamic scale factor				Local	
olda	Past value of angle of attack				Local	
oldb	Past value of sideslip angle				Local	
omass	One divided by mass				Local	
omx	X component of earth spin vector (same as a12w)				Local	
omy	Y component of earth spin vector (same as a22w)				Local	
omz	Z component of earth spin vector (same as a32w)				Local	
orbast	Orbiter axial base force due to SSME throttling	nt			Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
orbdrg	Darg force on orbiter	nt			Local	
orbdsf	Orbiter drag scale factor	nt			Local	
orbet	Orbiter/external tank shear force	lbs	ader	output	Global	pdome
orbfx	longitudinal component of orbiter thrust vector	lbs			Local	
orbfz	Normal component of orbiter thrust vector	lbs			Local	
orbitr	Flag specifying that the first stage(s) is being simulated				Local	
orbtf	Orbiter thrust scale factor				Local	
orbwt	Current orbiter weight	lbs			Local	
outisp(6)	Specific impulse of engine types	sec		input	Global	output
outth(6)	Actual thrust of engine types	lbs		input	Global	output
outtv(6)	Vacuum thrust of engine types	lbs		input	Global	output
outwd(6)	Flowrate of engine types	lbs/sec		input	Global	output
p1	X component of radius vector projected on equatorial plane				Local	
p2	Y component of radius vector projected on equatorial plane				Local	
p3	Z component of radius vector projected on equatorial plane				Local	
pam	Atmospheric pressure				Local	
pat(14)	Input/output array for atmospheric parameters			input	Global	ardc
pazwdh	Partial derivative of wind azimuth with respect to altitude (equivalent to wargd(2))				Local	
pdome	LOX dome pressure	psi	ader	output	Global	pdome
pe1x	Partial derivative of e1 with respect to X coordinate				Local	
pe1y	Partial derivative of e1 with respect to Y coordinate				Local	
pe1z	Partial derivative of e1 with respect to Z coordinate				Local	
pe2x	Partial derivative of e2 with respect to X coordinate				Local	
pe2y	Partial derivative of e2 with respect to Y coordinate				Local	
pe2z	Partial derivative of e2 with respect to				Local	

Variable	Variable Definition Z coordinate	Units	Source	I/O	Status	Common Block
pe3x	Partial derivative of e3 with respect to X coordinate				Local	
pe3y	Partial derivative of e3 with respect to Y coordinate				Local	
pe3z	Partial derivative of e3 with respect to Z coordinate				Local	
pfkg	Conversion for pounds to kilograms	kg/lbs	ainit	input	Global	const
pfn	Conversion from pounds to newtons	lbs/nt	ainit	input	Global	const
phx	Partial derivative of altitude with respect to X component				Local	
phy	Partial derivative of altitude with respect to Y component				Local	
phz	Partial derivative of altitude with respect to Z component				Local	
plag	temporary variable				Local	
plox	Weight of LOX in tank	lbs			Local	
prxx	Partial derivative of X component of the radius unit vector with respect to the X coordinate				Local	
prxy	Partial derivative of X component of the radius unit vector with respect to the Y coordinate				Local	
prxz	Partial derivative of X component of the radius unit vector with respect to the Z coordinate				Local	
pryy	Partial derivative of Y component of the radius unit vector with respect to the Y coordinate				Local	
pryz	Partial derivative of Y component of the radius unit vector with respect to the Z coordinate				Local	
przz	Partial derivative of Z component of the radius unit vector with respect to the Z coordinate				Local	
psffm	Conversion for pounds per square feet from newtons per square meter		ainit	input	Global	const
psftm	Conversion for pounds per square feet to newtons per square meter		ainit	input	Global	const
psi	Yaw attitude angle (optional reference system)	rad			Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ptkg	Conversion from pounds to kilograms	kg/lbs	ainit	input	Global	const
ptn	Conversion from pounds to newtons	nt/lbs	ainit	input	Global	const
pvru	Partial derivative of relative velocity with respect to Y component of inertial velocity vector				Local	
pvrv	Partial derivative of relative velocity with respect to Z component of inertial velocity vector				Local	
pvrw	Partial derivative of relative velocity with respect to X component of inertial velocity vector				Local	
pvrx	Partial derivative of relative velocity with respect to X component of position vector				Local	
pvry	Partial derivative of relative velocity with respect to Y component of position vector				Local	
pvrz	Partial derivative of relative velocity with respect to Z component of position vector				Local	
pw1x	Partial derivative of x component of wind vector with respect to x position				Local	
pw1y	Partial derivative of y component of wind vector with respect to x position				Local	
pw1z	Partial derivative of z component of wind vector with respect to x position				Local	
pw2x	Partial derivative of Y component of wind vector with respect to X component of position vector				Local	
pw2y	Partial derivative of Y component of wind vector with respect to Y component of position vector				Local	
pw2z	Partial derivative of Y component of wind vector with respect to Z component of position vector				Local	
pw3x	Partial derivative of Z component of wind vector with respect to X component of position vector				Local	
pw3y	Partial derivative of Z component of wind vector with respect to Y component of position vector				Local	
pw3z	Partial derivative of Z component of wind vector with respect to Z				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
	component of position vector					
pwmdh	Partial derivative of wind speed with respect to altitude (equivalent to wargd(1))				Local	
q	Dynamic pressure	nt/m^2	ader	output input	Global	agen2
qalpha	Product of dynamic pressure and angle of attack		ader	output input	Global	agen4
qalpha_max	Maximum value of q alpha		ainit	input	Global	qalpha_ max
qbeta	Produce of dynamic pressure and sideslip angle		ader	output	Global	agen4
qcn	Produce of dynamic pressure and normal force coefficient		ader	output	Global	agen4
qdot	Time derivative of dynamic pressure		ader	output input	Global	agen2
qfact	Dynamic pressure lag pressure				Local	
qflg	Logical variable indicating booster SSME throttle being simulated		ader	output	Global	qmax
qmax	Maximum value of dynamic pressure	psf	ainit	output input	Global	qmax
qratio	Ratio of dynamic pressure to reference value of dynamic pressure				Local	
qref	Reference value of dynamic pressure	nt/m^2			Local	
qvalue(13)	Current values of parameters used in load table outputs	variabl	ader	output input	Global	loads
r	Radius from earth's center	m	ader	output	Global	agen2
r2	Squared value of radius	m^2			Local	
ratio	Propellant mixture ratio		ainit	input	Global	tcl
re	Earth's radius	m	ainit	input	Global	const
reyno	Reynold's number		ader	output	Global	reyno
rho	Atmospheric density	kg/m^3			Local	
rthe	Current earth radius			output	Global	agen2
rxwxr	Local north unit vector				Local	
s(15)	Aerodynamic reference area for each thrust event	m^2	ainit	input	Global	agen1
sa	Sine of launch azimuth		aforun	input	Global	agen2
salp	Sine of desired angle of attack (alf)				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
salpha	Sine of angle of attack				Local	
sazw	Sine of wind azimuth				Local	
sbeta	Sine of desired sideslip angle (alfy)				Local	
schip	Sine of pitch attitude angle		ader1	input	Global	agen2
schir	Sine of roll attitude angle		ader1	input	Global	agen2
schiy	Sine of yaw attitude angle		ader1	input	Global	agen2
sdynsf	SRM dynamic scale factor				Local	
semach	Mach number based on using earth relative velocity			output	Global	aprint
side	Aerodynamic side force	newton		output	Global	agen5
sigma	Roll attitude angle in boost reference coordinate system	deg	ader1	input	Global	ricont
sindb	Sine of desired minus past sideslip angle				Local	
sinrho	Sine of angle between vehicle centerline and center of gravity location			output	Global	garbge
sins	Sine of sigma attitude angle				Local	
sint	Sine of theta attitude angle				Local	
smach	Saved value of mach number				Local	
solda	Sine of past value of angle of attack (olda)				Local	
soldb	Sine of past value of sideslip angle (oldb)				Local	
sopga	Sine of orbiter pitch gimbal angle				Local	
spsi	Sine of psi attitude angle				Local	
sq	Product of aerodynamic reference area and dynamic pressure				Local	
srbdsf	SRB drag scale factor				Local	
srbtf	SRB thrust scale factor				Local	
srbwt	Average weight of one SRB	lbs			Local	
sreyno	Square root of reynold number				Local	
srmdrg	Drag force on SRM				Local	
srmet	SRM/External tank shear force	lbs		output input	Global	pdome
srmext	SRM current exit area	m^2		input	Global	srmext
srmfx	Longitudinal component of SRM thrust	lbs			Local	
srmfz	Normal component of SRM thrust vector	lbs			Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
srmirt	SRB inert weight	lbs			Local	
sspga	Sine of SRB pitch gimbal angle				Local	
su	Y component of relative velocity vector	m/sec		output	Global	agen2
sue	Y component of earth relative velocity vector	m/sec		output input	Global	reyno
sv	Z component of relative velocity vector	m/sec		output	Global	agen2
sve	Z component of earth relative velocity vector	m/sec		output input	Global	reyno
sw	X component of relative velocity vector	m/sec		output	Global	agen2
swe	X component of earth relative velocity vector	m/sec	ader	output	Global	reyno
system	Name of the vehicle system being simulated (SHUTTLE, STAGE15, HLLV)		ainit	input	Global	system
t	Time from lift-off	sec	dpir	output	Global	forint
tandp1	Tangent of pitch gimbal angle of 1st equivalent engine				Local	
tandp2	Tangent of pitch gimbal angle of 2nd equivalent engine				Local	
tandy	Tangent of yaw gimbal angle				Local	
tanp	Tangent of engine pitch angle				Local	
tany	Tangent of engine yaw angle				Local	
tau	Thrust throttle value		ader	output input	Global	garbge
tau2	Interpolated MPS throttle required				Local	
taulim	Not used				Local	
taumin	Minimum value of throttle level				Local	
tau_const(10,	Constant value of throttle used for constant throttle after acceleration limit		ainit	input	Global	mps
tcl(6)	Longitudinal component of thrust vector for each system	newton	ader	output input	Global	tcl
temp	Temporary variable				Local	
temp1	Temporary variable				Local	
temp2	Temporary variable				Local	
temp3	Temporary variable				Local	
temp4	Temporary variable				Local	
temp5	Temporary variable				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
temp6	X component of radius unit vector				Local	
temp7	Y component of radius unit vector				Local	
temp8	Z component of radius unit vector				Local	
tendr	Time to terminate roll maneuver	sec	ainit	output	Global	agen1
test	Logical flag				Local	
test1	Logical flag				Local	
tfxow	Thrust to weight ratio in longitudinal component	nt			Local	
tfzow	Normal component of thrust to weight ratio	nt			Local	
theta	Pitch attitude angle (optional reference system)	rad			Local	
thmfa	Thrust minus axial force	nt	ader	output input	Global	agen5
thr1	Thrust	nt			Local	
thr2	Thrust	nt			Local	
throt(15)	Throttle table for MPS motor		ainit	input	Global	mps
thrust	Thrust	nt			Local	
thrvac	Vacuum thrust	nt			Local	
tlbm(30,2)	Vacuum thrust table for liquid booster motor	nt	ainit	input	Global	lbm
tll(6)	Lateral component of thrust vector for each system	nt		output input	Global	tcl
tmagorb	Thrust magnitude of orbiter engines	lbs			Local	
tmagsrb	Thrust magnitude of solid engines	lbs			Local	
tmz	Z component of thrust moment	nt -m	ader	output input	Global	agen5
tne(6,15)	Array which defines the number of engines per thrust event		ainit	input	Global	agen1
to2	Thrust divided by 2				Local	
ts	Summation of thrust magnitudes	nt			Local	
tv	Vacuum thrust magnitude	nt			Local	
tvl(6)	Vertical component of thrust vector for each system	sec	ader	output input	Global	tcl
tx	Vertical component of moment balanced thrust	newton	ader	output input	Global	agen5
tx1	Vertical component of thrust of 1st equivalent engine	newton	ader	output input	Global	garbge

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
tx2	Vertical component of thrust of 2nd equivalent engine	newton	ader	output input	Global	garbge
txs	Vertical component of non-moment balanced thrust	newton	ader	output input	Global	garbge
txx	Vertical component of moment balanced thrust	newton	ader	output input	Global	agen5
txxx	Normal component of thrust vector				Local	
ty	Longitudinal component of moment balanced thrust	newton	ader	output input	Global	agen5
ty1	Longitudinal component of thrust of 1st equivalent engine	newton	ader	output input	Global	garbge
ty2	Axial thrust component for second equivalent engine	newton	ader	output input	Global	garbge
tys	Longitudinal component of non-moment balanced thrust	nt	ader	output input	Global	garbge
tyy	Longitudinal component of moment balanced thrust	nt	ader	output input	Global	agen5
tyyy	Longitudinal component of thrust vector				Local	
tz	Lateral component of moment balanced thrust	nt	ader	output input	Global	agen5
tzs	Lateral component of solid motor thrust vector	nt			Local	
tzz	Lateral component of moment balanced thrust	newton		output input	Global	agen5
u	Y component of plumbline inertial velocity vector	m/sec	dpir	input	Global	forint
udxdy2	Temporary variable used in calculation of moment balance (1.-(dx/dy)^2)			output input	Global	garbge
ullage	Ullage pressure of LOX tank	psi			Local	
umf	One minus the earth flattening coefficient		ainit	input	Global	agen2
umfc	Temporary variable				Local	
ur	Radius unit vector				Local	
uwxr	Unit vector in local east direction				Local	
v	Z component of plumbline inertial velocity vector	m/sec	dpir	input	Global	forint
va	Array of viscous interaction coefficients				Local	
vaerdd	Array of partial derivatives of viscous interaction coefficients with respect to				Local	

Variable	Variable Definition mach number	Units	Source	I/O	Status	Common Block
vaero	Array of viscous interaction coefficients				Local	
vb	Spline coefficients for viscous interaction coefficient interpolation				Local	
vc	Spline coefficients for viscous interaction coefficient interpolation				Local	
vdotg	Gravity losses				Local	
ve	Magnitude of earth relative velocity	m/sec	ader	output input	Global	reyno
vmag	Magnitude of inertial velocity vector	m/sec			Local	
v r	Magnitude of relative velocity	m/sec	ader	output	Global	agen2
vr2	Squared value of wind relative velocity	m^2/s^			Local	
vwb	Temporary variable				Local	
vxb	Temporary variable				Local	
vyb	Temporary variable				Local	
vzb	Temporary variable				Local	
w	X component of plumbline inertial velocity vector	m/sec	dpir	input	Global	forint
warg	Wind parameters				Local	
wargd	Partial derivative of wind parameters with respect to altitude				Local	
wddump(15)	Amount of weight dumped during OMS/RCS thrust events	kg	ainit	input	Global	omsrcs
wdoms(15)	Weight flowrate of OMS engines	lbs/sec	ainit	input	Global	omsrcs
wdot	Propellant flow rate	kg/sec			Local	
wdot1	Propellant flow rate	kg/sec			Local	
wdot2	Propellant flow rate	kg/sec			Local	
wdrcs(15)	Weight flowrate of RCS engines	lbs/sec	ainit	output input	Global	omsrcs
wind	Wind vector	m/sec			Local	
wloxt	Current weight of LOX in propellant tanks	lbs			Local	
wmag	Wind speed magnitude	m/sec	ader	output	Global	agen5
w x r	Magnitude of WXR vector				Local	
wxrm	Magnitude of WXR vector				Local	
x	X component of plumbline position	m	dpir	input	Global	forint

Variable	Variable Definition vector	Units	Source	I/O	Status	Common Block
xcg	Vertical component of center of gravity	m	ader1	input	Global	garbge
xgpa	Vertical component of gimbal points of equivalent booster engines	m	athrev	input	Global	avggp
xisp	Specific impulse	sec			Local	
xk1	Coefficient to calculate aerodynamic heating indicator				Local	
xk2	Coefficient to calculate aerodynamic heating indicator				Local	
xk3	Coefficient to calculate aerodynamic heating indicator				Local	
xk4	Coefficient to calculate aerodynamic heating indicator				Local	
xk5	Coefficient to calculate aerodynamic heating indicator				Local	
xk6	Coefficient to calculate aerodynamic heating indicator				Local	
xk7	Coefficient to calculate aerodynamic heating indicator				Local	
xknud	Knudsen number		ader	output input	Global	aprint
xlamda	Mean free path		ader	output input	Global	aprint
xlen	Reference length	m	athrev	input	Global	bodyf
xm	Current vehicle mass	kg	ader	output	Global	agen2
xmach	Mach number		ader	output	Global	agen2
xmaug	Auxiliary vehicle mass	kg	athrev	input	Global	agen1
xmdot	Propellant flow rate	kg/sec			Local	
xmiad	Continuous portion of vehicle mass	kg	ader1	input	Global	agen2
xmu	Viscosity coefficient				Local	
xref	Vertical component of moment reference point	m	athrev	input	Global	bodyf
xv(4)	Storage array for sine and cosine values of pitch and yaw attitude angles		ader1	input	Global	x v
xww	Total vehicle weight	lbs			Local	
x x	Temporary variable				Local	
x y	Temporary variable				Local	
xz	Z component of normal axis direction				Local	

Variable	Variable Definition cosine	Units	Source	I/O	Status	Common Block
xzn	Temporary variable				Local	
y	Y component of plumbline position vector	m	dpir	input	Global	forint
ycg	Longitudinal component of centerof gravity	m	ader1	input	Global	garbge
ycp	Longitudinal component of center of pressure	m			Local	
ygpa	Longitudinal component of gimbal point of booster equivalent engine	m	athrev	input	Global	avggp
yref	Longitudinal component of moment reference point	m	athrev	input	Global	bodyf
yring	External tank Y-ring pressure	psi	ader	output input	Global	pdome
y x	X component of longitudinal axis direction cosine				Local	
yxn	Temporary variable				Local	
y y	Y component of longitudinal axis direction cosine				Local	
yyn	Temporary variable				Local	
y z	Z component of longitudinal axis direction cosine				Local	
yzn	Temporary variable				Local	
z	Z component of plumbline position vector	m	dpir	input	Global	forint
zap(18)	Additional integrated variables	variabl	dpir	input	Global	forint
zgpa	Lateral component of gimbal point of booster equivalent engines	m	athrev	input	Global	avggp
z x	Temporary variable				Local	
zy	Temporary variable				Local	
zz	Z component of lateral axis direction cosine				Local	
zzn	Temporary variable				Local	

```
      SUBROUTINE ADER                                                   ADER   1
c*********************************************************************
C     Evaluates the state derivatives for the forward trajectory
c*********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      CHARACTER*60 HEAD
      CHARACTER*6 MISION,ATMOSN
      character*7 system

C     Sam Powell
c     changes for heat rate constraint
c     8/21/90


      REAL*8 LIFT

      integer engines

      LOGICAL GLIMIT,QFLG,BOOST,ORBITR,VISC,moment_bal,linear_aero,
     *cp_input,mombal,test,test1

      common/astor/asto(4,67)

      COMMON/AERODI/AEROD(30,17),AEROB(50,6,3),TOMACH(25),DELCA(25),
     *PNMONE(15),BKFABT(50,2),BKFTHR(50,2),DELCN(25),DELCM(25)

      COMMON/AGEN1/TIME(2,16),TAUT(15),TAUW(15),TZERO,TLIFT,TTILT,TMINH,
     *DTZ,TQ,TL,XMAUG,TNE(6,15),ENGDAT(8,15),S(15),WD(15),WJET(15),
     *HEAD,PRINT(15),STEP(15),HNOM(15),HMAX(15),PROP(6),TBEGR,TENDR,
     *CHRDOT,CHIRO,FAZ,BO(3,2),CBAXIL,TCHIR,VRCUT,FPRFAC,CHVEL

      COMMON/AGEN2/AZ,SA,CA,ALF,ALFY,CHIP,SCHIP,CCHIP,CHIY,SCHIY,CCHIY,
     *CHIR,SCHIR,CCHIR,STH,CTH,STHL,CTHL,DPHIZ,RTHE,R,VR,XM,SW,SU,SV,Q,
     *VIV(25),DVARS(13),DELXDW(15,3),DELXDR(7,5),DELXD(15,7),WZERO,ALT,
     *XMACH,QDOT,FAA,FAN,A(3,3),CASE,CT2,UMF,A12W,A22W,A32W,XJEXT,BEU,
     *BYL,BEL,BHMAX(15),BHMN,BSTEP(15),HNMB,HMXB,TPOLY,XMIAD

      COMMON/AGEN3/ITHR,IWD,IXR,JUMP,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISION,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)

      COMMON/AGEN4/QALPHA,QBETA,QCN,STH1,CTH1,XJX(3),XJV(3)

      COMMON/AGEN5/CHIBAS,CHPBAS,CHYBAS,CHISAV,TXX,TYY,TZZ,AMX,AMY,AMZ,
     *TMZ,THMFA,SIDE,WMAG,AZW,FANC,FLATC,FOM,VISC

      COMMON/ALAT/ALAT,ALONG,ALTLS,NTABLE

      COMMON/APRINT/ELEIN,ELEOUT,DINBD,DOUTBD,FABT(4),CAX,CNP,CMZ,CS,
     *CMX,CMY,SEMACH,DRAG,LIFT,XLAMDA,XKNUD
```



```
      COMMON/ARDC/PAT(14),ATMOSN,IATMOS,HBIAS,HAERO

      COMMON/AVGGP/XGPA,YGPA,ZGPA

      COMMON/BODYF/TBDWT(15),TXCG(15),TYCG(15),XLEN,XREF,YREF,M1,M2,M5,
     *M6,M7,M8,M9,M50,M51,MSRW,AP(10),AY(10)

      COMMON/COMNEW/HRATEC

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM

      COMMON/CONTRO/TTBL(30),CPTBL(30),CYTBL(30),TOBL(210),
     *CPOTBL(210),CYOTBL(210)

      common/cp_input/cp_input

      COMMON/DELFAB/DELALT(25),DELFAB(25),M30,M31,ITOL(15)

      common/engines/engines(15,15)

      COMMON/ELEVON/EMACH(25,2),ELEVON(25,4),M10,M11

      COMMON/FORINT/HBANK(2),T,TT,W,U,V,X,Y,Z,XMI,ZAP(18),DVAR(25),FSAVE
     *(625),NTRG1,TV1,NTRG2,TV2,NTRG3,TV3,NTRG4,TV4,NTRG5,TV5,NTRG6,TV6,
     *NTRG7,TV7,NTRG8,TV8,NTRG9,TV9,NTRG10,TV10,NTRG11,TV11,NTRG12,TV12,
     *NTRG13,TV13,KIND,KRDER,EU,EL,AYL,HMX,HMN,HNM

      COMMON/GARBGE/DX,DYI,DZI,DXDY,UDXDYI,UDXDY2,XCG,YCG,TX1,TX2,TY1,TY
     *2,YHAT(3),TXS,TYS,TAU,SINRHO

      COMMON/GRAV/GTO,G11,G22,G23,G33

      common/ithro/ithro

      COMMON/LBM/TIMLBM(30),TLBM(30,2)              ,PLBM(2),MLBM1,MLBM2,
     *ALBM(2),BLBM(2),CLBM(2),VLBM(2)

      common/linear_aero/linear_aero

      COMMON/M20/M17,M20,M21,M22,M23,M24

      common/mombal/mombal

      COMMON/MPS/TIMMPS(15),FTBL(15),THROT(15),GLIM(15),TAUALT(5),TAUTBL
     *(5),MPS1,MPS2,TGLIM(10,15),GLIMIT,tau_const(10,5),jthrot(15)

      common/mps1/timemps(15),taumps(15),mpsinp(15)

      common/mult/nchiot(6),tbias(6),nstage,nstg(15),wf(5),ibranch

      COMMON/NTILT/NTILT

      COMMON/OMSRCS/FOMS(15),FRCS(15),WDOMS(15),WDRCS(15),WDDUMP(15)
```

```
      COMMON/OUTPUT/OUTTH(6),OUTTV(6),OUTISP(6),OUTWD(6)

      common/pdome/pdome,yring,orbet,srmet

      common/qalpha_max/qalpha_max,xnul

      COMMON/QMAX/QMAX,QFLG,QPEN,MAXQ

      COMMON/LOADS/QLOAD(26),QTIME(26),QVALUE(13),SAVE(20,13)

      COMMON/REST/JSTR,NCNRS(5),NTCN,TR(5),LSB,NF8,NOPAR,WIBT(15)

      COMMON/REYNO/REYNO,VE,SWE,SUE,SVE

      COMMON/RICONT/TTABLE(20),SIGMA,SIGTAB(20),M15,M16

      COMMON/SRM/SRMTTB(600),SRMFTB(600),SRMWDT(600),SRMAE(600),MSRB

      COMMON/SRMEXT/SRMEXT

      common/system/system

      COMMON/TCL/TCL(6),TVL(6),TLL(6),RATIO

      COMMON/VAERO/VAEROD(10,19),M18,M19,MBS18,MBS19

      common/vrwtbl/vrwtbl(30),ivrtsw

      COMMON/WIND/ALTTBL(100),WTBL(100,2),NWIND

      COMMON/XV/XV(4)

      DIMENSION WARG(2),WARGD(2),AW(2),BW(2),CW(2)

      EQUIVALENCE (WMAG,WARG(1)),(AZW,WARG(2))

      DIMENSION AERO(13),WXR(3),WIND(3),RXWXR(3),UR(3),UWXR(3),
     *AERODD(17),DQDX(6),AA(13),BB(13),CC(13)        ,AFAB(4),BFAB(4),
     *CFAB(4)

      EQUIVALENCE (FAB,FABT(1)),(FNB,FABT(2)),(FMB,FABT(3)),
     *(qref,fabt(4))

      EQUIVALENCE (CAALP,AERO(1)),(CAO,AERO(2)),(CLBETA,AERO(3)),      ADER  57
     *(CMALP,AERO(4)),(CMO,AERO(5)),(CNALP,AERO(6)),(CNBETA,AERO(7)),  ADER  58
     *(CNO,AERO(8)),(CYBETA,AERO(9)),(CLO,AERO(10)),(CYO,AERO(11)),
     *(CNBO,AERO(12)),(ycp,aero(13)),(PWMDH,WARGD(1)),(PAZWDH,WARGD(2))

      DIMENSION VAERO(18),VAERDD(18),VA(18),VB(18),VC(18)

      EQUIVALENCE(ACACF,VAERO(1)),(BCACF,VAERO(2)),(CCACF,VAERO(3)),
     *(ACNCF,VAERO(4)),(BCNCF,VAERO(5)),(CCNCF,VAERO(6)),
     *(ACMCF,VAERO(7)),(BCMCF,VAERO(8)),(CCMCF,VAERO(9)),
     *(ACAMF,VAERO(10)),(BCAMF,VAERO(11)),(CCAMF,VAERO(12)),
     *(ACNMF,VAERO(13)),(BCNMF,VAERO(14)),(CCNMF,VAERO(15)),
     *(ACMMF,VAERO(16)),(BCMMF,VAERO(17)),(CCMMF,VAERO(18))
```



```
      DIMENSION BAERO(6)

      EQUIVALENCE (CAX,BAERO(1)),(CNP,BAERO(2)),(CMZ,BAERO(3)),
     *(CS,BAERO(4)),(CMX,BAERO(5)),(CMY,BAERO(6))

      DIMENSION DFAB(1),DDFAB(1),AF(1),BF(1),CF(1),wloxt(21),
     *headt(21)

      DATA XK1,XK2,XK3,XK4,XK5,XK6,XK7/17700.,.0019403,.3048,3048.,
     *110.4,.432,50063./

      data qfact,ullage,dens,odynsf,sdynsf,orbdsf,srbdsf,orbtf,srbtf/
     *.0026,22.,0.041,1.,1.,.349,.103,1.,1./

      data headt/0.,297.8,332.1,338.9,345.9,353.0,360.3,367.8,375.5,
     *383.5,391.5,400.4,409.4,429.0,451.6,479.2,503.1,509.4,515.2,
     *518.7,550.0/

      data wloxt/0.,.9,1.,1.02,1.04,1.06,1.08,1.10,1.12,1.14,1.16,1.18,
     *1.20,1.24,1.28,1.32,1.34651,1.352396,1.357157,1.36,1.37/

      BOOST=ITHR.LE.NOEVNT(1)
      ORBITR=.NOT.BOOST

      NUMQ=1
      R2=X*X+Y*Y+Z*Z                                                   ADER  60
      R=SQRT(R2)                                                       ADER  61

      SCHIP=XV(1)                                                      ADER  62
      CCHIP=XV(2)                                                      ADER  63
      SCHIY=XV(3)                                                      ADER  64
      CCHIY=XV(4)                                                      ADER  65

      CTH=(A(1,2)*X+A(2,2)*Y+A(3,2)*Z)/R
      CT2=CTH*CTH                                                      ADER  74

c**********************************************************************

c       Define Gravitation components in AGEO subroutine

c**********************************************************************

      CALL AGEO(1)

      GX=X*G11-A(1,2)*GTO                                              ADER  76
      GY=Y*G11-A(2,2)*GTO                                              ADER  77
      GZ=Z*G11-A(3,2)*GTO                                              ADER  78

      RTHE=UMF*RE/(UMF*UMF*(1.-CT2)+CT2)**.5                           ADER  79
      ALT=R-RTHE                                                       ADER  80

c**********************************************************************

c       Define components of wind vector
```

```
c*********************************************************************

      DO I=1,3
         WIND(I)=0.
      enddo

      WMAG=0.                                                          ADER  83
      AZW=0.                                                           ADER  84

      WXR(1)=A22W*Z-A32W*Y                                             ADER  85
      WXR(2)=A32W*X-A12W*Z                                             ADER  86
      WXR(3)=A12W*Y-A22W*X                                             ADER  87

      WXRM=SQRT(WXR(1)*WXR(1)+WXR(2)*WXR(2)+WXR(3)*WXR(3))             ADER  90

      DO I=1,3
         UWXR(I)=WXR(I)/WXRM
      enddo

      UR(1)=X/R                                                        ADER  93
      UR(2)=Y/R                                                        ADER  94
      UR(3)=Z/R                                                        ADER  95

      RXWXR(1)=UR(2)*UWXR(3)-UR(3)*UWXR(2)                             ADER  96
      RXWXR(2)=UR(3)*UWXR(1)-UR(1)*UWXR(3)                             ADER  97
      RXWXR(3)=UR(1)*UWXR(2)-UR(2)*UWXR(1)                             ADER  98

      IF(NWIND.ne.0) then

         HATMOS=ALT+HBIAS

         IF(HATMOS.ge.ALTTBL(1).and.HATMOS.le.ALTTBL(NWIND)) then

            CALL SPLINE(2,M7,100,HATMOS,ALTTBL,WARG,WTBL,WARGD,2,MSRW,
     *      AW,BW,CW)

            SAZW=-SIN(AZW)
            CAZW=-COS(AZW)

            DO I=1,3
               WIND(I)=WMAG*(SAZW*UWXR(I)+CAZW*RXWXR(I))
            enddo

         endif

      endif

c*********************************************************************

c       Define relative velocity components and magnitude

c*********************************************************************

      SW=W-WXR(1)-WIND(1)                                              ADER 104
      SU=U-WXR(2)-WIND(2)                                              ADER 105
      SV=V-WXR(3)-WIND(3)                                              ADER 106
```



```
      VR2=SW*SW+SU*SU+SV*SV

C       CHANGE WAS NECESSARY FOR 0 WINDS

      IF(VR2.LT..001) VR2=.1
      VR=SQRT(VR2)                                                     ADER 108

c*********************************************************************

c       Define atmospheric parameters as a function of altitude

c*********************************************************************

      PAT(1)=ALT+HBIAS

      if(iatmos.eq.0) then

         call rraspl(pat)

         pam=pat(2)

      else

         call atmosphere(pat,atmosn,ier,iatmos)

         pam=pat(2)*10000.

      endif

      PAT(1)=ALT
      RHO=PAT(6)                                                       ADER 114
      XMU=PAT(10)
      REYNO=RHO*VR*XLEN/XMU
      XMACH=VR/PAT(9)                                                  ADER 115

c*********************************************************************

C     CALCULATE ALTITUDE FOR BASE FORCE TABLE LOOKUP

c*********************************************************************

      HAERO=ALT+aerob(1,1,1)-altls

c*********************************************************************

C     SPLINE INTERPOLATION FOR SECOND STAGE AERODYNAMIC DATA(MONOVARIATE

c*********************************************************************

      IF(.not.BOOST.or.linear_aero) then

         smach=xmach
         IF(XMACH.ge.10.) xmach=10.

         CALL SPLINE(13,M1,30,XMACH,AEROD(1,14),AERO,AEROD,AERODD,
```

```
     *    1,m2,AA,BB,CC)

          xmach=smach

       endif

c**********************************************************************

C      SPLINE INTERPOLATION FOR POWER-ON BASE FORCE EFFECTS

c**********************************************************************

       IF(ITHR.GE.NOBASE) then

          fab=cbax11
          fnb=0.
          fmb=0.

       else

          L=LSTGE(ITHR)
          IF(L.EQ.2.AND.TNE(1,ITHR).LT.2.9) L=3

          IF(L.EQ.1) THEN
             KK=4
          ELSE
             KK=3
          ENDIF

          CALL SPLINE(KK,M8,50,HAERO,AEROB(1,1,L),FABT,AEROB(1,2,L),
     *    AERODD(14),1,M9,AFAB,BFAB,CFAB)

       endif

c**********************************************************************

c       Define current mass

c**********************************************************************

       XM=XMIAD+XMAUG                                                  ADER 118

c**********************************************************************

C      EARTH RELATIVE VELOCITY (without wind components)

c**********************************************************************

       SWE=W-WXR(1)
       SUE=U-WXR(2)
       SVE=V-WXR(3)

       VE=SQRT(SWE*SWE+SUE*SUE+SVE*SVE)

c**********************************************************************

c       Define dynamic pressure (Q)
```



```
c**********************************************************************

       Q=.5*RHO*VR2                                                    ADER 121

       if(ithr.lt.nobase) then
          if(lstge(ithr).eq.1.and.linear_aero.and.qref.gt.1.) then
             FAB=FAB*Q/QREF
          endif
       endif

c      QPEN=DMAX1(QPEN,Q*PSFFM)

       SQ=S(ITHR)*Q                                                    ADER 122

c**********************************************************************

c       Define attitude angles (chip & chiy) prior to time of minh

c**********************************************************************

       NCHI=0

       if(ifact.eq.1.and.ithr.lt.minh.and.ivrtsw.eq.2) then

          call amulg(2,m15,30,vr/.3048,vrwtbl,chip,cptbl,chiy,cytbl)

          chip=chip/rad
          chiy=chiy/rad
          schip=sin(chip)
          cchip=cos(chip)
          schiy=sin(chiy)
          cchiy=cos(chiy)

       endif

       IF(IFACT.eq.3.and.ITHR.lt.MINH) then

          CALL AMULG(2,M16,30,VE,TTBL,THETA,CPTBL,PSI,CYTBL)

C**********************************************************************

C      COMPUTE ATTITUDE ANGLES IN BOOST REFERENCE COORD SYSTEM FOR PRINT
C      BOOST COORD SYSTEM : X-NORTH
C                           Y-EAST
C                           Z-DOWN LOCAL VERTICAL

C**********************************************************************

          SPSI=SIN(PSI/RAD)
          CPSI=COS(PSI/RAD)
          SINT=SIN(THETA/RAD)
          COST=COS(THETA/RAD)
          SINS=SIN(SIGMA/RAD)
          COSS=COS(SIGMA/RAD)
```

```
         YX=COST*(CPSI*CA+SPSI*SA)
         YY=SINT
         YZ=COST*(SPSI*CA-CPSI*SA)

         XZ=SA*COST*(-SINT*CPSI*COSS-SPSI*SINS)+CA*(SINT*SPSI*
     *      COSS-CPSI*SINS)
         ZZ=SA*COST*(-SINT*CPSI*SINS-SPSI*COSS)+CA*(SINT*SPSI*
     *      SINS+CPSI*COSS)

         CCHIY=SQRT(YX*YX+YY*YY)
         SCHIP=YX/CCHIY
         CCHIP=YY/CCHIY
         SCHIY=YZ
         SCHIR=-XZ/CCHIY
         CCHIR=ZZ/CCHIY

      else IF(TENDR.gt..001.and.T.le.TENDR) then

         YX=-CCHIY*SCHIP*CCHIR-SCHIY*SCHIR
         YY=CCHIY*CCHIP
         YZ=CCHIY*SCHIP*SCHIR-SCHIY*CCHIR

         XZ=-CCHIP*SCHIR
         ZZ=SCHIY*SCHIP*SCHIR+CCHIY*CCHIR

         CCHIY=SQRT(YX*YX+YY*YY)
         SCHIP=YX/CCHIY
         CCHIP=YY/CCHIY
         SCHIY=YZ
         SCHIR=-XZ/CCHIY
         CCHIR=ZZ/CCHIY

      endif


      VWB=SCHIP*SW+CCHIP*SU                                          ADER 135
      VZB=SCHIY*VWB-CCHIY*SV                                         ADER 136
      VYB=CCHIP*SW-SCHIP*SU                                          ADER 137
      VXB=CCHIR*VYB+SCHIR*VZB                                        ADER 138
      VZB=SCHIR*VYB-CCHIR*VZB                                        ADER 139
      VYB=CCHIY*VWB+SCHIY*SV                                         ADER 140

c*********************************************************************

c       Define angle of attack (alf) and sideslip angle (alfy)

c*********************************************************************

      ALF=0.
      ALFY=0.

      IF(T.ge..3) then

         ALF=ATAN2(VXB,VYB)
         ALFY=ASIN(VZB/VR)
```



```
      endif

      OLDA=ALF                                                       ADER 149
      OLDB=ALFY                                                      ADER 150
      QALPHA=ABS(Q*ALF)
      QBETA=ABS(Q*ALFY)

      test=ITHR.lt.MINH.and.IFACT.eq.2.and.NTILT.eq.1.and.ITHR.lt.IPOLY
      test1=qalpha.gt.qalpha_max.and.qalpha_max.gt.0..and.ithr.ge.minh

      if(test.or.test1) then

c*********************************************************************

c       Define attitude angles as a function of angle of attach and
c       sideslip angle

c*********************************************************************
         if(test) then
            if(qalpha_max.eq.0.) then
               CALL AMULG(2,M20,30,XMACH,AEROD(1,16),ALF,AEROD(1,17),
     *         ALFY,AEROD(1,15))
            else
               alf=-qalpha_max/q
            endif
         else
            alf=qalpha_max/q*sign(1.d0,alf)
         endif

         COLDA=COS(OLDA)
         SOLDA=SIN(OLDA)
         COLDB=COS(OLDB)
         SOLDB=SIN(OLDB)
         SALP=SIN(ALF)
         CALP=COS(ALF)
         SBETA=SIN(ALFY)
         CBETA=COS(ALFY)

         SINDB=SOLDB*CBETA-COLDB*SBETA
         COSDB=COLDB*CBETA+SOLDB*SBETA

         XX=CALP*COLDA+SALP*SOLDA*COSDB
         XY=-CALP*SOLDA+SALP*COLDA*COSDB
         XZ=SALP*SINDB

         YX=-SALP*COLDA+CALP*SOLDA*COSDB
         YY=SALP*SOLDA+CALP*COLDA*COSDB
         YZ=CALP*SINDB

         ZX=-SOLDA*SINDB
         ZY=-COLDA*SINDB
         ZZ=COSDB

         YXN=YX*(CCHIR*CCHIP+SCHIR*SCHIY*SCHIP)+YY*(CCHIY*SCHIP)+
     *      YZ*(SCHIR*CCHIP-CCHIR*SCHIY*SCHIP)
         YYN=YX*(-CCHIR*SCHIP+SCHIR*SCHIY*CCHIP)+YY*(CCHIY*CCHIP)+
```

```
     *       YZ*(-SCHIR*SCHIP-CCHIR*SCHIY*CCHIP)

         YZN=YX*(-SCHIR*CCHIY)+YY*SCHIY+YZ*CCHIR*CCHIY
         XZN=XX*(-SCHIR*CCHIY)+XY*SCHIY+XZ*CCHIR*CCHIY
         ZZN=ZX*(-SCHIR*CCHIY)+ZY*SCHIY+ZZ*CCHIR*CCHIY

         CCHIY=SQRT(YXN*YXN+YYN*YYN)
         SCHIP=YXN/CCHIY
         CCHIP=YYN/CCHIY
         SCHIY=YZN
         SCHIR=-XZN/CCHIY
         CCHIR=ZZN/CCHIY

      endif
C
      ALP=ALF*RAD
      BETA=ALFY*RAD

C*********************************************************************

c       Calculate mach number (SEMACH) and Reynolds number (SREYNO)

C*********************************************************************

      SEMACH=VE/PAT(9)
      SREYNO=SQRT(REYNO)

C*********************************************************************

C     DEFINITION OF AERODYNAMIC COEFFICIENTS
C
C     CAX= AXIAL FORCE COEFFICIENT.                                  ADER 216
C     CS=  SIDE FORCE COEFFICIENT.                                   ADER 217
C     CNP=NORMAL FORCE COEFFICIENT.                                  ADER 218
C     CMZ=PITCHING MOMENT COEFFICIENT
C     CMY=ROLLING MOMENT COEFFICIENT
C     CMX=YAWING MOMENT COEFFICIENT

C*********************************************************************

      if(boost.and..not.linear_aero) then

         IF(ITHR.lt.NOBASE) then

C*********************************************************************

C     LINEAR INTERPOLATION FOR SSME THROTTLING,Q AND ALPHA-BETA EFFECTS
C     ON POWER-ON BASE FORCE--BOOSTER ONLY

C*********************************************************************

            CALL AMULG(2,M24,50,HAERO,AEROB(1,1,1),FK1THR,
     *      BKFTHR(1,1),FK2THR,BKFTHR(1,2))
C
            CALL AMULG(2,M23,50,HAERO,AEROB(1,1,1),FABORB,
     *      AEROB(1,6,1),QREF,AEROB(1,5,1))
```



```
C*********************************************************************

C     SSME THROTTLING EFFECT-BASE FORCE

C*********************************************************************

  320       if(mpsflg(ithr).eq.1.or.mpsflg(ithr).eq.6) tau=throt(ithr)

            if(mpsflg(ithr).eq.2) call amulg(1,m51,15,t,timmps,tau,
     *      ftbl,0,0)

            if(mpsflg(ithr).eq.3) then

               if(qflg) then

                  tau=ftbl(ithro)

               else

                  tau=throt(ithr)

               endif

            endif

            if(mpsflg(ithr).eq.4) then

               if(glimit) then

                  mmin=0
                  do i=1,15

                     if(nsyst(i).eq.1.and.engines(i,ithr).eq.1) then
                        if(mmin.eq.0) mmin=i
                     endif

                  enddo

                  gm=xm*glim(ithr)*gzero
                  tau=gm/(engdat(6,mmin)*tne(1,ithr)*ptn)

               else

                  tau=throt(ithr)

               endif

            endif

            if(mpsflg(ithr).eq.5) then

               if(glimit) then

                  tau=tau_const(jthrot(ithr),nstage)
```

```
               else

                  tau=throt(ithr)

               endif

            endif

            DTHRTL=1.09-TAU
            BASTHR=(FK2THR*DTHRTL+FK1THR)*DTHRTL+1.0

C*****************************************************************

C     ORBAST=ORBITER BASE FORCE-AXIAL,DUE TO SSME THROTTLING
C     FABORB=ORBITER BASE FORCE-AXIAL,FOR SSME=109% RPL

C*****************************************************************

            ORBAST=FABORB*BASTHR

C     TOTAL AXIAL BASE FORCE-SSME THROTTLING EFFECT

            FAB=FAB-FABORB+ORBAST

C     TOTAL NORMAL BASE FORCE-SSME THROTTLING EFFECT

            FNB=FNB*BASTHR

C     TOTAL PIT MOMENT BASE FORCE-SSME THROTTLING EFFECT

            FMB=FMB*BASTHR
C
C     Q EFFECT-BASE FORCE
C
            IF(QREF.gt.48.) then

               QRATIO=Q/QREF

C     AXIAL BASE FORCE-Q EFFECT

               FAB=FAB*QRATIO

C     NORMAL BASE FORCE-Q EFFECT

               FNB=FNB*QRATIO

C     PIT MOMENT BASE FORCE-Q EFFECT

               FMB=FMB*QRATIO

            endif

C*****************************************************************

C     LINEAR INTERPOLATION FOR ALPHA-BETA FACTOR FOR POWER-ON BASE
C     FORCE EFFECT
```



```
C*****************************************************************

            IF(T.ge.2.0) then

               CALL AMULG(2,M17,50,HAERO,AEROB(1,1,1),FKALF,
     *         BKFABT(1,1),FKBETA,BKFABT(1,2))

               ABFACT=1.0+FKALF*ALP+FKBETA*BETA**2

               FAB=FAB*ABFACT
               FNB=FNB*ABFACT
               FMB=FMB*ABFACT

            endif

         endif

C*****************************************************************

C     FIRST STAGE AERODYNAMIC COEFFICIENTS

C*****************************************************************

C*****************************************************************

C       Determine elevon deflection angle

C*****************************************************************

         CALL AMULG(2,M10,25,VE/.3048,EMACH(1,1),ELEIN,ELEVON(1,1),
     *   ELEOUT,ELEVON(1,2))

         CALL AMULG(2,M11,25,SEMACH,EMACH(1,2),DINBD,ELEVON(1,3),
     *   DOUTBD,ELEVON(1,4))

C*****************************************************************

C     DELTA(1)=INBOARD ELEVON DEFLECTION  (SCHEDULE 6)
C     DELTA(2)=OUTBOARD ELEVON DEFLECTION (SCHEDULE 6)
C     DELTA(3)=INBOARD ELEVON DEFLECTION  (JSC/RI AERO DATA BASE)
C     DELTA(4)=OUTBOARD ELEVON DEFLECTION (JSC/RI AERO DATA BASE)

C*****************************************************************

         DINBD=ELEIN-DINBD
         DOUTBD=ELEOUT-DOUTBD
C
         CALL CAERO(BAERO,XMACH,ALP,BETA,DINBD,DOUTBD)

      else

C*****************************************************************

C     SECOND STAGE AERODYNAMIC COEFFICIENTS
```

```
C*********************************************************************

         CAX=CAO+CAALP*ALF
         CNP=CNO+CNALP*ALF
         CMZ=CMO+CMALP*ALF
         CS=CYO+CYBETA*ALFY
         CMX=CNBO+CNBETA*ALFY
         CMY=CLO+CLBETA*ALFY
C
c        IF(XMACH.gt.10.0) then
         if(xmach.gt.100.) then

C*********************************************************************

C     HYPERSONIC VISCOUS INTERACTION EFFECTS (ORBITER STAGE ONLY)
C
C              REYNO - REYNOLDS NUMBER
C              XLAMDA- MEAN FREE PATH (PAGE 13,U.S. STD ATMOS, 1962)
C              XKNUD  - KNUDSEN NUMBER (AERO DATA BOOK 4.1.5)
C              CF     - CONTINUUM FLOW
C              MF     - FREE MOLECULAR FLOW

C*********************************************************************

            XLAMDA=2.332353E-05*PAT(12)/PAM
            XKNUD=.1407*XLAMDA/VR*(PAM/RHO)**.5

            CALL SPLINE(18,M18,10,XMACH,VAEROD(1,19),VAERO,VAEROD,
     *      VAERDD,1,M19,VA,VB,VC)

            CACF=ACACF+(BCACF+CCACF*ALP)*ALP
            CNCF=ACNCF+(BCNCF+CCNCF*ALP)*ALP
            CMCF=ACMCF+(BCMCF+CCMCF*ALP)*ALP
            CAMF=ACAMF+(BCAMF+CCAMF*ALP)*ALP
            CNMF=ACNMF+(BCNMF+CCNMF*ALP)*ALP
            CMMF=ACMMF+(BCMMF+CCMMF*ALP)*ALP

            IF(XKNUD.LT..01) XKNUD=.01
            IF(XKNUD.GT.10.) XKNUD=10.

            TEMP=SIN(9.009*(XKNUD-.01)/RAD)

            CAX=(CAMF-CACF)*TEMP+CACF
            CNP=(CNMF-CNCF)*TEMP+CNCF
            CMZ=(CMMF-CMCF)*TEMP+CMCF

            QCN=Q*CNP
C                                                                    ADER 223
         endif

      endif

C*********************************************************************

C     ADD AERO COEFF TOLERANCES=F(MACH NBR) IF ITOL NOT EQUAL TO ZERO
```



```
C*********************************************************************

      IF(ITOL(ITHR).ne.0) then

         CALL AMULG(2,M21,25,XMACH,TOMACH,DCAX,DELCA,DCNP,DELCN)

         CALL AMULG(1,M21,25,XMACH,TOMACH,DCMZ,DELCM,0,0)

         CAX=CAX+DCAX
         CNP=CNP+DCNP
         CMZ=CMZ+DCMZ

         CALL SPLINE(1,M30,25,HAERO,DELALT,DFAB,DELFAB,DDFAB,1,M31,
     *   AF,BF,CF)

         FAB=FAB+DFAB(1)

      endif

C*********************************************************************

C       MOMENT COEFFICIENTS ABOUT C.G. NOW FORMED

C*********************************************************************

      TEMP=1./XLEN

      if(cp_input) then
         dxpr=0.
         dypr=(ycg-ycp)*temp
      else
         DXPR=(XCG-XREF)*TEMP
         DYPR=(YCG-YREF)*TEMP
      endif

      CMCG=CMZ-CNP*DYPR+CAX*DXPR
      CNCG=CMX-CS*DYPR
      CLCG=CMY+CS*DXPR

      FAA=CAX*SQ+FAB
      FAN=CNP*SQ+FNB
      SIDE=CS*SQ                                                     ADER 243

      AMX=CNCG*SQ*XLEN                                               ADER 244
      AMY=CLCG*SQ*XLEN                                               ADER 245
      AMZ=CMCG*SQ*XLEN+FMB-FNB*DYPR*XLEN+FAB*DXPR*XLEN

C*********************************************************************

C     THRUST AND MOMENT CALCULATIONS FOR MULTIPLE PROPULSION SYSTEMS

C*********************************************************************

      TXX=0.
      TYY=0.
      TZZ=0.
```

```
      TXS=0.
      TYS=0.
      TZS=0.
      XMDOT=0.
      TMZ=0.

      DO 500 NSY=1,7

      N=NSY

      IF(ORBITR.AND.N.EQ.1) GO TO 500
      IF(ORBITR.AND.N.EQ.7) N=1

      IF(BOOST.AND.ITHR.LT.ICONSW.AND.N.EQ.2) GO TO 500
      IF(BOOST.AND.ITHR.LT.ICONSW.AND.N.EQ.7) N=2

      IF(BOOST.AND.ITHR.GE.ICONSW.AND.N.EQ.1) GO TO 500
      IF(BOOST.AND.ITHR.GE.ICONSW.AND.N.EQ.7) N=1

      DVAR(N+7)=0.
      OUTISP(N)=0.
      OUTTH(N)=0.
      OUTTV(N)=0.
      OUTWD(N)=0.

      TCL(N)=0.
      TVL(N)=0.
      TLL(N)=0.

      IF(TNE(N,ITHR).LT..001) GO TO 500

      MMIN=0

      do i=1,15

         IF(NSYST(I).eq.N.and.engines(i,ithr).eq.1) then
            IF(MMIN.EQ.0) MMIN=I
            MMAX=I
         endif

      enddo

C*********************************************************************

C     MPS SYSTEM (n=1,throttable;n=6,fixed)

C*********************************************************************

      if(n.eq.1.or.n.eq.6) then

c       define minimum throttle level as a function of altitude

         CALL AMULG(1,M50,5,ALT,TAUALT,TAUMIN,TAUTBL,0,0)

         tau=1.
```



```
C*********************************************************************

C     NORMAL OPERATIONS

C*********************************************************************

         if(mpsflg(ithr).eq.1) then

            TAU=THROT(ITHR)

            IF(QFLG) QFLG=.FALSE.

            THRVAC=TAU*ENGDAT(6,MMIN)*PTN

C*********************************************************************

C     TABLE OF THROTTLE VS TIME

C*********************************************************************

         else if(mpsflg(ithr).eq.2) then

            CALL AMULG(1,M51,15,T,TIMMPS,TAU,FTBL,0,0)

            IF(TAU.LT.TAUMIN) TAU=TAUMIN
            THRVAC=TAU*ENGDAT(6,MMIN)*PTN

C*********************************************************************

C     QMAX THROTTLE

C*********************************************************************

         else if(mpsflg(ithr).eq.3) then

            IF(.NOT.QFLG) then

               TAU=THROT(ITHR)

               IF(QFLG) QFLG=.FALSE.

               THRVAC=TAU*ENGDAT(6,MMIN)*PTN

            else

               if(taulim.gt.0.) then

                  CALL AMULG(1,M51,15,T,TIMMPS,TAU,FTBL,0,0)

               else

                  tau=ftbl(ithro)

               endif

               IF(TAU.LT.TAUMIN) TAU=TAUMIN
```

```
               THRVAC=TAU*ENGDAT(6,MMIN)*PTN

            endif

C*****************************************************************

C     G LIMIT THROTTLE

C*****************************************************************

         else if(mpsflg(ithr).eq.4.or.mpsflg(ithr).eq.5) then

            IF(.NOT.GLIMIT) then

               TAU=THROT(ITHR)

               IF(QFLG) QFLG=.FALSE.

               THRVAC=TAU*ENGDAT(6,MMIN)*PTN

            else

               if(mpsflg(ithr).eq.4) then

                  GM=XM*GLIM(ITHR)*GZERO
                  TAU=GM/(ENGDAT(6,MMIN)*(TNE(1,ITHR)+
     *                tne(6,ithr))*PTN)

               else

                  tau=tau_const(jthrot(ithr),nstage)

               endif

c              IF(TAU.LT.TAUMIN) TAU=TAUMIN
               THRVAC=TAU*ENGDAT(6,MMIN)*PTN

            endif

         endif

         XISP=FUNISP(TAU,nstage)

         WDOT=THRVAC/(GZERO*XISP)*TNE(N,ITHR)

c*****************************************************************

c       mps thrust and flowrate table on specific engines

c*****************************************************************

         if(mpsflg(ithr).eq.6) then

            call amulg(1,m51,15,t-time(1,ithr),timemps,tau2,taumps,
     *       0.,0.)
```



```
            thr2=tau2*engdat(6,mmin)*ptn
            xisp=funisp(tau2,nstage)
            wdot2=thr2/(gzero*xisp)

            thr1=throt(ithr)*engdat(6,mmin)*ptn
            xisp=funisp(throt(ithr),nstage)
            wdot1=thr1/(gzero*xisp)

            thrvac=0.
            wdot=0.

            do i=1,15

               if(nsyst(i).eq.1.and.engines(i,ithr).eq.1) then

                  if(mpsinp(i).eq.0) then

                     thrvac=thrvac+thr1
                     wdot=wdot+wdot1

                  else

                     thrvac=thrvac+thr2
                     wdot=wdot+wdot2

                  endif

               endif

            enddo

            xisp=thrvac/(gzero*wdot)
            thrvac=thrvac/tne(1,ithr)

         endif
C*****************************************************************

C     SRB SYSTEM

C*****************************************************************

      else if(n.eq.2) then

         CALL AMULG(2,MSRB,600,T,SRMTTB,THRVAC,SRMFTB,WDOT,SRMWDT)

         CALL AMULG(1,MSRB,600,T,SRMTTB,SRMEXT,SRMAE,0,0)

         XISP=0.
         IF(WDOT.GT.0.) XISP=THRVAC/WDOT

         THRVAC=THRVAC*PTN

         ENGDAT(8,MMIN)=SRMEXT

         WDOT=WDOT*TNE(2,ITHR)*PTKG
```

```
C*********************************************************************
C     LBM SYSTEM
C*********************************************************************
      else if(n.eq.3) then

         CALL AMULG(2,MLBM1,30,T,TIMLBM,THRVAC,TLBM(1,1),WDOT,
     *   TLBM(1,2))

         XISP=0.
         IF(WDOT.GT.0.) XISP=THRVAC/WDOT

         THRVAC=THRVAC*PTN

         WDOT=WDOT*TNE(3,ITHR)*PTKG
C*********************************************************************
C     OMS SYSTEM
C*********************************************************************
      else if(n.eq.4) then

         THRVAC=FOMS(ITHR)*PTN
         XISP=THRVAC/(WDOMS(ITHR)*GZERO*PTKG)
         WDOT=(WDOMS(ITHR)*TNE(4,ITHR)+WDDUMP(ITHR))*PTKG
C*********************************************************************
C     RCS SYSTEM
C*********************************************************************
      else if(n.eq.5) then

         THRVAC=FRCS(ITHR)*PTN
         XISP=0.

         IF(WDRCS(ITHR).EQ.0..OR.GZERO.EQ.0..OR.PTKG.EQ.0.) THEN

            XISP = 0.0

         ELSE

            IF(WDRCS(ITHR).LT..001) XISP=THRVAC/(WDRCS(ITHR)*GZERO*
     *      PTKG)

         ENDIF

         WDOT=WDRCS(ITHR)*TNE(5,ITHR)*PTKG
C*********************************************************************
```



```
C     reserved for FUTURE SYSTEM
C*********************************************************************
      else

         GO TO 500

      endif

      OUTISP(N)=XISP
      OUTWD(N)=WDOT

      THRUST=THRVAC-ENGDAT(8,MMIN)*PAM
      IF(THRUST.LT.0.) THRUST=0.

      OUTTH(N)=THRUST*TNE(N,ITHR)
      OUTTV(N)=THRVAC*TNE(N,ITHR)

      XMDOT=XMDOT+WDOT

      DVAR(N+7)=-WDOT

      moment_bal=.false.

      IF(ORBITR.AND.N.EQ.1) moment_bal=.true.

      IF(BOOST.AND.N.EQ.2.AND.ITHR.LT.ICONSW)  moment_bal=.true.
      IF(BOOST.AND.N.EQ.1.AND.ITHR.GE.ICONSW)  moment_bal=.true.

      IF(OUTTH(N).LT.10000..OR.NTRG7.GT.0)  moment_bal=.false.

      if(.not.mombal) moment_bal=.false.
C*********************************************************************
C     NON-GIMBALLED THRUST AND PITCH MOMENT COMPONENTS
C*********************************************************************
      if(.not.moment_bal) then

         NENG=MMAX-MMIN+1
         IF(INT(TNE(N,ITHR)+.1).NE.NENG) MMAX=INT(TNE(N,ITHR)+.1)+
     *   MMIN-1

         DO 475 I=1,15

         if(nsyst(i).ne.n.or.engines(i,ithr).eq.0) go to 475

         if(n.ne.6) then

            TANP=TAN(ENGDAT(4,I)/RAD)
            TANY=TAN(ENGDAT(5,I)/RAD)

         else
```

```
            tanp=0.
            if(mombal) tanp=-(xcg-engdat(3,i))/(ycg-engdat(1,i))
            tany=0.

         endif

         TEMP=THRUST/SQRT(1.+TANP*TANP+TANY*TANY)

         TX=-TEMP*TANP
         TY=TEMP
         TZ=TEMP*TANY

         TCL(N)=TCL(N)+TY
         TVL(N)=TVL(N)+TX
         TLL(N)=TLL(N)+TZ

         if(mombal) then
            TMZ=-TY*(XCG-ENGDAT(3,I))+TX*(YCG-ENGDAT(1,I))+TMZ
         else
            tmz=0.
         endif

         IF(QFLG.and.N.eq.1) then

            TXS=TXS+TX
            TYS=TYS+TY
            TZS=TZS+TZ

         endif

         TXX=TXX+TX
         TYY=TYY+TY
         TZZ=TZZ+TZ

  475    CONTINUE

C*****************************************************************

C     MOMENT BALANCE EQUATIONS

C*****************************************************************

      else

         THRUST=OUTTH(N)

         DX=XGPA-XCG
         DY=YGPA-YCG
         DZ=ABS(ZGPA)

         DYI=1./DY
         DZI=1./DZ
         DXDY=DX*DYI
         UDXDY2=1.+DXDY*DXDY

         COSRHO=1./SQRT(UDXDY2)
```



```
         SINRHO=DXDY*COSRHO

         TMZ=AMZ+TMZ
         TANDY=-(AMX-AMY*DXDY)*DYI/(THRUST*UDXDY2)
         TEMP=(AMX*DXDY+AMY)*DZI/THRUST
         TEMP1=-TMZ*DYI/THRUST
         TANDP1=(TEMP1-TEMP)*COSRHO
         TANDP2=(TEMP1+TEMP)*COSRHO
         TO2=THRUST/2.
         TEMP2=TANDY*TANDY

         TY1=TO2/SQRT(1.+TANDP1*TANDP1+TEMP2)
         TY2=TO2/SQRT(1.+TANDP2*TANDP2+TEMP2)
         TX2=-TY2*TANDP2

         TX1=TX2-TY1*TANDP1
         TY1=TY1+TY2

         TXX=TX1*COSRHO+TY1*SINRHO+TXX
         TYY=-TX1*SINRHO+TY1*COSRHO+TYY
         TZZ=TY1*TANDY+TZZ

         TCL(N)=-TX1*SINRHO+TY1*COSRHO
         TVL(N)=TX1*COSRHO+TY1*SINRHO
         TLL(N)=TY1*TANDY

      endif

  500 CONTINUE

      THMFA=TYY-FAA
      FANC=TXX-FAN
      FLATC=TZZ+SIDE

      FOM=SQRT(THMFA*THMFA+FANC*FANC+FLATC*FLATC)/XM/GZERO

      TXXX=TXX
      TYYY=TYY

      DVAR(7)=-XMDOT

C*****************************************************************

C     COMPUTE LIFT AND DRAG

C*****************************************************************

      CALPHA=COS(ALF)
      SALPHA=SIN(ALF)
      CBETA=COS(ALFY)
      SBETA=SIN(ALFY)

      DRAG=FAA*CALPHA+FAN*SALPHA
      LIFT=-FAA*SALPHA+FAN*CALPHA

C*****************************************************************
```

```
c       Calculate force components
c*********************************************************************
      FZ=TZZ+SIDE                                                    ADER 353
      FW=SCHIR*FANC-CCHIR*FZ                                         ADER 354
      FY=SCHIY*FW+CCHIY*THMFA                                        ADER 355
      FU=CCHIR*FANC+SCHIR*FZ                                         ADER 356
      FX=CCHIP*FU+SCHIP*FY                                           ADER 357
      FY=-SCHIP*FU+CCHIP*FY                                          ADER 358
      FZ=-CCHIY*FW+SCHIY*THMFA                                       ADER 359
c*********************************************************************
c       Calculate acceleration components
c*********************************************************************
      DVAR(1)=FX/XM+GX                                               ADER 360
      DVAR(2)=FY/XM+GY                                               ADER 361
      DVAR(3)=FZ/XM+GZ                                               ADER 362
c*********************************************************************
c       Set components to zero if Liftoff has not occurred
c*********************************************************************
      IF(NTRG7.ge.0) then
        DVAR(1)=0.
        DVAR(2)=0.
        DVAR(3)=0.
      endif
c*********************************************************************
c       Calculate velocity components
c*********************************************************************
      DVAR(4)=W                                                      ADER 363
      DVAR(5)=U                                                      ADER 364
      DVAR(6)=V                                                      ADER 365
      IF(NF8.gt.13) then
         VMAG=SQRT(W*W+U*U+V*V)
         VDOTG=(W*GX+U*GY+V*GZ)/VMAG
         CALP1=(W*CCHIP-U*SCHIP)*CCHIR/VMAG
         CALP3=W*SCHIP+U*CCHIP
         CALP2=(CALP3*CCHIY+V*SCHIY)/VMAG
         CALP3=(V*CCHIY-CALP3*SCHIY)*CCHIR/VMAG
```



```
         TS=OUTTH(1)+OUTTH(2)+OUTTH(3)+OUTTH(4)+OUTTH(5)+OUTTH(6)
         TV=SQRT(TXXX*TXXX+TYYY*TYYY+TZZ*TZZ)
c*********************************************************************
c       Calculate heating indicator rate
c*********************************************************************
         DVAR(14)=XK1*(XK2*RHO)**.5*(VR/XK4)**3.07*(1.-XK5/
     *            (XK6*PAT(12)+((VR/XK3)**2)/XK7))
C
         IF(T.LT.5.) HRATEC=0.
         IF(T.LT.150.) THEN
            IF(HRATEC.LE.DVAR(14)) HRATEC=DVAR(14)
         ENDIF
         OMASS=1./XM
         DVAR(15)=OMASS*(OUTTV(1)+OUTTV(2)+OUTTV(3)+OUTTV(4)+OUTTV(5)+
     *            OUTTV(6))
         DVAR(16)=DVAR(15)-TS*OMASS
         DVAR(17)=(TS-TV)*OMASS
         IF(NF8.gt.17) then
            DVAR(18)=-VDOTG
            DVAR(19)=(-TXXX*CALP1-TYYY*CALP2-TZZ*CALP3+TV)*OMASS
            DVAR(20)=(FAN*CALP1+FAA*CALP2-SIDE*CALP3)*OMASS
            DVAR(21)=0.
            IF(Q.GT.QMAX*PSFTM) DVAR(21)=Q*PSFFM-QMAX
            DVAR(22)=WDDUMP(ITHR)
         endif
       endif
c*********************************************************************
c       Store first stage values during final trajectory iteration
c*********************************************************************
      IF(NMAX.eq.0.and.JUMP.eq.1) then
         QVALUE(1)=Q*ALF*PSFFM*RAD
         QVALUE(2)=Q*ALFY*PSFFM*RAD
         QVALUE(3)=Q*PSFFM
         QVALUE(4)=THMFA/XM/GZERO
         QVALUE(5)=(TZZ+SIDE)/XM/GZERO
         QVALUE(6)=FANC/XM/GZERO
         QVALUE(7)=SQRT(QVALUE(4)**2+QVALUE(5)**2+QVALUE(6)**2)
         QVALUE(8)=ZAP(7)
         QVALUE(9)=DVAR(14)
```

```
c       load indicator calculations
c
c       variable                description                         units
c
c       qfact           dynamic pressure lag pressure               -
c       ullage          ullage pressure of lox tank                 psi
c       dens            density of lox tank                         lbs/in^3
c       odynsf          orbiter dynamic scale factor                -
c       sdynsf          srm dynamic scale factor                    -
c       orbdsf          orbiter drag scale factor                   -
c       srbdsf          srb drag scale factor                       -
c       orbtf           orbiter thrust scale factor                 -
c       srbtf           srb thrust scale factor                     -
c       q               dynamic pressure                            psf
c       sopga           sine of orbiter pitch gimbal angle          -
c       copga           cosine of orbiter pitch gimbal angle        -
c       sspga           sine of srb pitch gimbal angle              -
c       cspga           cosine of srb pitch gimbal angle            -
c       wloxt           current weight of lox in tank               lbs
c       xww             total current vehicle weight                lbs
c       tmagorb         total resultant ssme thrust(all 3 ssme)     lbs
c       tmagsrb         total resultant srb thrust(both motors)     lbs
c       fa              aerodynamic axial force (negative)          lbs
c       fn              aerodynamic normal force (negative)         lbs
c       orbwt           current orbiter weight                      lbs
c       pdome           lox dome pressure                           psi
c       yring           et y-ring pressure                          psi
c       orbet           orbite/et shear force                       lbs
c       srmet           srm/et shear force                          lbs

c       updated prerequisite equations needed for load indicator
c       equations, provided by RI/Downey ascent performance 8/88

        mload=1
        plox=ratio*zap(1)*pfkg/1.d+06

        call amulg(1,mload,21,plox,wloxt,hlox,headt,0,0)

c       orbiter gimbal angles
c       (if-then block accounts for orbiter gimbal angles at meco)

        if(tvl(1).lt.1.d-04.and.tcl(1).lt.1.d-04.and.
     *  tll(1).lt.1.d-04) then

           sopga=0.
           copga=1.
           coyga=1.
        else
           sopga=sin(atan2(-tvl(1),tcl(1)))
           copga=cos(atan2(-tvl(1),tcl(1)))
           coyga=cos(atan2(tll(1),tcl(1)))
        endif

c       srb gimbal angles
c       (if-then block accounts for srb gimbal angles after separation)
```



```
        if(tvl(2).lt.1.d-04.and.tcl(2).lt.1.d-04.and.
     *  tll(2).lt.1.d-04) then

           sspga=0.
           cspga=1.
           csyga=1.
        else
           sspga=sin(atan2(-tvl(2),tcl(2)))
           cspga=cos(atan2(-tvl(2),tcl(2)))
           csyga=cos(atan2(tll(2),tcl(2)))
        endif

c       thrust magnitudes

        tmagorb=outth(1)*pfn
        tmagsrb=outth(2)*pfn

c       x and z components of total ssme thrust

        orbfx=copga*coyga*tmagorb*orbtf
        orbfz=sopga*coyga*tmagorb*orbtf

c       x and z components of total srm thrust

        srmfx=cspga*csyga*tmagsrb*srbtf
        srmfz=sspga*csyga*tmagsrb*srbtf

c       total vehicle weight

        xww=xm*pfkg

c       aerodynamic forces

        fa=-faa*pfn
        fn=-fan*pfn

c       thrust to weight factors

        tfxow=(odynsf*orbfx+sdynsf*srmfx+fa)/xww
        tfzow=abs((odynsf*orbfz+sdynsf*srmfz+fn)/xww)

c       drag on orbiter and one srb

        orbdrg=fa*orbdsf
        srmdrg=fa*srbdsf

c       average weight of one srb

        srbwt=0.
        if(system.eq.'SHUTTLE'.or.system.eq.'HLLV    ') then
           srmirt=asto(1,47)
           srbwt=(zap(2)*pfkg+srmirt)/2.0
        endif

c       calculate load indicators
```

```
         pdome=0.
         yring=0.
         orbet=0.
         srmet=0.

c        lox dome pressure

         plag=qfact*q*psffm
         pdome=ullage+dens*tfxow*hlox+plag

c        ET y-ring pressure

         yring=ullage+dens*(165.5*tfzow+(hlox-110.625)*tfxow)+plag

         if(ithr.le.noevnt(1)) then

c        SRM/ET shear force

             srmet=0.
             if(system.eq.'SHUTTLE'.or.system.eq.'HLLV   ') then
                srmet=(srmfx/2.0)*sdynsf+srmdrg-srbwt*tfxow
             endif

c        Orbiter/ET shear force

             etirt=asto(1,25)
             orbwt=xww-2.0*srbwt-zap(1)*pfkg-etirt

c        the above formula is correct for both first and second stages

         endif

         orbet=orbfx*odynsf+orbdrg-orbwt*tfxow

         qvalue(10)=pdome
         qvalue(11)=yring
         qvalue(12)=orbet
         qvalue(13)=srmet

       endif

       IF(ORBITR) RETURN

C*********************************************************************

C      CALCULATE Q DOT (first stage only)

C*********************************************************************

C      CALCULATE P(RHO)WRT ALT                                          ADER 386

       DRDR=PAT(4)                                                      ADER 387

C      CALCULATE DQDX                                                   ADER 388

       UMFC=FLAT*(2.-FLAT)/(RE*RE*UMF*UMF)
```



```
       DRTHE=RTHE**3*UMFC*CTH/R2

       PHX=X/R-(X*CTH-R*A(1,2))*DRTHE
       PHY=Y/R-(Y*CTH-R*A(2,2))*DRTHE
       PHZ=Z/R-(Z*CTH-R*A(3,2))*DRTHE
C
       OMX=A12W
       OMY=A22W
       OMZ=A32W

       TEMP=WXR(1)
       TEMP1=WXR(2)
       TEMP2=WXR(3)

       TEMP3=TEMP*TEMP+TEMP1*TEMP1+TEMP2*TEMP2

       P1=(TEMP1*OMZ-TEMP2*OMY)/TEMP3
       P2=(TEMP2*OMX-TEMP *OMZ)/TEMP3
       P3=(TEMP *OMY-TEMP1*OMX)/TEMP3

       TEMP4=1./SQRT(TEMP3)
C
       E1=TEMP *TEMP4
       E2=TEMP1*TEMP4
       E3=TEMP2*TEMP4

       PE1X=-E1*P1
       PE1Y=-E1*P2-OMZ*TEMP4
       PE1Z=-E1*P3+OMY*TEMP4

       PE2X=-E2*P1+OMZ*TEMP4
       PE2Y=-E2*P2
       PE2Z=-E2*P3-OMX*TEMP4

       PE3X=-E3*P1-OMY*TEMP4
       PE3Y=-E3*P2+OMX*TEMP4
       PE3Z=-E3*P3
C
       TEMP5=1./R
       TEMP6=X*TEMP5
       TEMP7=Y*TEMP5
       TEMP8=Z*TEMP5

       PRXX=TEMP5*(1.-TEMP6*TEMP6)
       PRXY=TEMP5*(  -TEMP6*TEMP7)
       PRXZ=TEMP5*(  -TEMP6*TEMP8)

       PRYY=TEMP5*(1.-TEMP7*TEMP7)
       PRYZ=TEMP5*(  -TEMP7*TEMP8)
       PRZZ=TEMP5*(1.-TEMP8*TEMP8)
C
       TEMP4=E3*TEMP7-E2*TEMP8
       TEMP3=SAZW*E1+CAZW*TEMP4

       PW1X=PWMDH*PHX*TEMP3+WMAG*(SAZW*(PE1X-PAZWDH*PHX*TEMP4)+CAZW*
     *      ((PE3X*Y-PE2X*Z)*TEMP5+(E3*PRXY-E2*PRXZ)+PAZWDH*PHX*E1))
```

```
      PW1Y=PWMDH*PHY*TEMP3+WMAG*(SAZW*(PE1Y-PAZWDH*PHY*TEMP4)+CAZW*
     *      ((PE3Y*Y-PE2Y*Z)*TEMP5+(E3*PRYY-E2*PRYZ)+PAZWDH*PHY*E1))
      PW1Z=PWMDH*PHZ*TEMP3+WMAG*(SAZW*(PE1Z-PAZWDH*PHZ*TEMP4)+CAZW*
     *      ((PE3Z*Y-PE2Z*Z)*TEMP5+(E3*PRYZ-E2*PRZZ)+PAZWDH*PHZ*E1))
C
      TEMP4=E1*TEMP8-E3*TEMP6
      TEMP3=SAZW*E2+CAZW*TEMP4

      PW2X=PWMDH*PHX*TEMP3+WMAG*(SAZW*(PE2X-PAZWDH*PHX*TEMP4)+CAZW*
     *      ((PE1X*Z-PE3X*X)*TEMP5+(E1*PRXZ-E3*PRXX)+PAZWDH*PHX*E2))
      PW2Y=PWMDH*PHY*TEMP3+WMAG*(SAZW*(PE2Y-PAZWDH*PHY*TEMP4)+CAZW*
     *      ((PE1Y*Z-PE3Y*X)*TEMP5+(E1*PRYZ-E3*PRXY)+PAZWDH*PHY*E2))
      PW2Z=PWMDH*PHZ*TEMP3+WMAG*(SAZW*(PE2Z-PAZWDH*PHZ*TEMP4)+CAZW*
     *      ((PE1Z*Z-PE3Z*X)*TEMP5+(E1*PRZZ-E3*PRXZ)+PAZWDH*PHZ*E2))
C
      TEMP4=E2*TEMP6-E1*TEMP7
      TEMP3=SAZW*E3+CAZW*TEMP4

      PW3X=PWMDH*PHX*TEMP3+WMAG*(SAZW*(PE3X-PAZWDH*PHX*TEMP4)+CAZW*
     *      ((PE2X*X-PE1X*Y)*TEMP5+(E2*PRXX-E1*PRXY)+PAZWDH*PHX*E3))
      PW3Y=PWMDH*PHY*TEMP3+WMAG*(SAZW*(PE3Y-PAZWDH*PHY*TEMP4)+CAZW*
     *      ((PE2Y*X-PE1Y*Y)*TEMP5+(E2*PRXY-E1*PRYY)+PAZWDH*PHY*E3))
      PW3Z=PWMDH*PHZ*TEMP3+WMAG*(SAZW*(PE3Z-PAZWDH*PHZ*TEMP4)+CAZW*
     *      ((PE2Z*X-PE1Z*Y)*TEMP5+(E2*PRXZ-E1*PRYZ)+PAZWDH*PHZ*E3))
C
      TEMP=1./VR

      PVRX=(SV*OMY-SU*OMZ)*TEMP
      PVRY=(SW*OMZ-SV*OMX)*TEMP
      PVRZ=(SU*OMX-SW*OMY)*TEMP

      IF(NWIND.ne.0) then

         PVRX=PVRX+(-SW*PW1X-SU*PW2X-SV*PW3X)*TEMP
         PVRY=PVRY+(-SW*PW1Y-SU*PW2Y-SV*PW3Y)*TEMP
         PVRZ=PVRZ+(-SW*PW1Z-SU*PW2Z-SV*PW3Z)*TEMP

      endif

      PVRW=SW*TEMP
      PVRU=SU*TEMP
      PVRV=SV*TEMP
C
      TEMP1=RHO*VR
      TEMP2=.5*VR*VR*DRDR

      DQDX(1)=TEMP1*PVRW
      DQDX(2)=TEMP1*PVRU
      DQDX(3)=TEMP1*PVRV
      DQDX(4)=TEMP1*PVRX+TEMP2*PHX
      DQDX(5)=TEMP1*PVRY+TEMP2*PHY
      DQDX(6)=TEMP1*PVRZ+TEMP2*PHZ

      QDOT=0.                                                    ADER 405

      DO 11 I=1,6                                                ADER 406
```



```
   11 QDOT=QDOT+DQDX(I)*DVAR(I)                                  ADER 407

      RETURN
      END                                                        ADER 409
```

3.5 Subroutine ADER1

3.5.1 Purpose

The purpose of this subroutine is to calculate the time dependent portion of the equations of motion. During the forward integration both subroutines ADER and ADER1 are called to evaluate the equations of motion. Subroutine ADER1 is only called when the time is updated.

3.5.2 Variable Listing

3.5.3 Subroutines Called:

Name	Function
ACNTRO	Defines vehicle attitude during min-H phase
AMULG	Linear interpolation routine
CHIPOL	Calculates pitch and yaw attitude from polynomial
MASSPRO	Calculates dynamic mass properties
SPLINE	Interpolation routine using spline method

3.5.4 Calling Subroutines:

Name	Function
AEOSR	Stores the state variables during the forward integration
AFORUN	Controls forward integration
ATHREV	Controls thrust event logic
ATILT	Controls tilt-over logic in forward integration
DISPPRT	Controls writing output file for dispersion analyses

3.5.5 Fortran Listing

ader1

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a(3,3)	Tranformation matrix from equatorial to plumbline coordinate systems		aforun	input	Global	agen2
acg	Spline coefficients for center of gravity interpolation				Local	
bcg	Spline coefficients for center of gravity interpolation				Local	
calcpar	Logical flag to indicate calculation of partials				Local	
ccg	Spline coefficients for center of gravity interpolation				Local	
cchip	Cosine of pitch attitude angle		ader1	output input	Global	agen2
cchir	Cosine of roll attitude angle		ader1	output input	Global	agen2
cchiy	Cosine of yaw attitude angle		ader1	output input	Global	agen2
cg	Interpolated values of center of gravity components				Local	
cgbl	Center of gravity tables				Local	
chip	Pitch attitude angle	rad	ader1	output input	Global	agen2
chipa	Pitch attitude angle (output from interpolation)	deg			Local	
chir	Roll attitude angle	rad		output input	Global	agen2
chiro	Initial value of roll attitude angle		atilt	input	Global	agen1
chisav	Saved value of chip	rad	ader1	output	Global	agen5
chiy	Yaw attitude angle	rad	ader1	output input	Global	agen2
chiya	Yaw attitude angle (output from interpolation)				Local	
chpbas	Pitch attitude bias angle	rad	ader1	output	Global	agen5
chrdot	Roll attitude rate	rad/sec	aforun	input	Global	agen1
chybas	Yaw attitude bias angle	rad	ader1	output	Global	agen5
cpst	Cosine of psi attitude angle				Local	
cth1	Cosine of launch colatitude			input	Global	agen4
delwt	Incremental weight overboard	kg			Local	
dphiz	Longitudinal displacement due to GRR			input	Global	agen2
eventnum	Number of event				Local	

ader1

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
hdmach	Mach number lower limit for second roll maneuver			output	Global	headup
ifact	Flag indicating the type of attitude used in the booster stage		atilt	input	Global	agen3
ihead	Roll attitude flag			output	Global	agen3
ipoly	Flag indicating number of polynominals in booster stage		ainit	input	Global	agen3
ithr	Thrust event index number		athrev	input	Global	agen3
ivrtsw	1st stage attitude option flag		ainit	input	Global	vrwtbl
jump	Jump start flag		ainit	input	Global	agen3
ks1	Booster attitude indicator		atilt	input	Global	agen3
latcg	Lateral center of gravity component	m			Local	
latcgp	Partial derivative of lateral CG with respect to mass				Local	
loncg	Total longitudinal center of gravity	m			Local	
loncgp	Partial derivative of total longitudinal CG with respect to mass				Local	
minh	Flag to indicate thrust event which begins min-h attitude optimization		ainit		Global	agen3
moboos	Booster mass				Local	
momain	Main engine mass				Local	
mombal	Flag to specify the use of moment balance equations		ainit	input	Global	mombal
mpflag	Alternate mass properties flag				Global	mpropin
noevnt(5)	Number of thrust events per stage		ainit	input	Global	agen3
norcg	Total normal component of center of gravity	m			Local	
norcgp	Derivative of normal CG with respect to mass	in/lbs			Local	
nroll	Flag indicating booster roll program after launch		aforun	input	Global	agen3
olow	Orbiter liftoff weight	kg	athrev	input	Global	olow
omega	Earth's spin rate	rad/sec	ainit	input	Global	const
pcg	Derivative of center of gravity components with respect to mass				Local	
pfkg	Conversion for pounds to kilograms	kg/lbs	ainit	input	Global	const
phil	Longitude at launch	rad			Local	
pi	Pi constant (3.14159265)		ainit	input	Global	const

ader1

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
prop(6)	Propellant weight for each engine type	kg	ainit	input	Global	agen1
schip	Sine of pitch attitude angle			output	Global	agen2
schir	Sine of roll attitude angle		ader1	output	Global	agen2
schiy	Sine of yaw attitude angle		ader1	output input	Global	agen2
sth1	Sine of launch colatitude			input	Global	agen4
stsp	Temporary variable				Local	
t	Time from lift-off			output input	Global	forint
t	Time from lift-off	sec	dpir	input	Global	forint
tbegr	Time to begin roll maneuver	sec	aforun	output input	Global	agen1
tendr	Time to terminate roll maneuver	sec	ainit	input	Global	agen1
ttilt	Time to begin tiltover maneuver	sec	atilt	input	Global	agen1
txcg(15)	Vertical center of gravity table	m	athrev	input	Global	bodyf
tycg(15)	Longitudinal center of gravity table	m	athrev	input	Global	bodyf
t_liftoff	Time of lift-off	sec	ainit	input	Global	t_liftoff
wint	Initial vehicle weight	kg	ainit	input	Global	olow
xcg	Vertical component of center of gravity				Global	garbge
xhat	Temporary variable				Local	
xm	Current vehicle mass	kg	ader1	output input	Global	agen2
xmach	Mach number		ader	input	Global	agen2
xmaug	Auxiliary vehicle mass	kg	athrev	input	Global	agen1
xmi	Integrated vehicle mass less jettisoned inerts.	kg	dpir	input	Global	forint
xmiad	Continuous portion of vehicle mass	kg	ader1	output input	Global	agen2
xv(4)	Storage array for sine and cosine values of pitch and yaw attitude angles			output	Global	x v
ycg	Longitudinal component of centerof gravity	m		output	Global	garbge
yhat	Temporary variable				Local	
y x	Temporary variable				Local	
zap(18)	Additional integrated variables	variabl	dpir	input	Global	forint

```
      SUBROUTINE ADER1                                                  ADER   1
C*********************************************************************
C      Evaluates the time dependent variables for the forward trajectory
C*********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      INTEGER EVENTNUM
      REAL*8 MOBOOS,MOMAIN,BOMDOT,MNMDOT,LONCG,LATCG,NORCG,LONCGP
      REAL*8 LATCGP,NORCGP

      LOGICAL MPFLAG,CALCPAR
      CHARACTER*30 MPNAME

      LOGICAL VISC,mombal
C
      CHARACTER*60 HEAD
      CHARACTER*6 MISION

      COMMON/CONTRO/TTBL(30),CPTBL(30),CYTBL(30),TOBL(210),
     *CPOTBL(210),CYOTBL(210)

      COMMON/AGEN1/TIME(2,16),TAUT(15),TAUW(15),TZERO,TLIFT,TTILT,TMINH,
     *DTZ,TQ,TL,XMAUG,TNE(6,15),ENGDAT(8,15),S(15),WD(15),WJET(15),
     *HEAD,PRINT(15),STEP(15),HNOM(15),HMAX(15),PROP(6),TBEGR,TENDR,
     *CHRDOT,CHIRO,FAZ,BO(3,2),CBAXIL,TCHIR,VRCUT,FPRFAC,CHVEL

      COMMON/AGEN2/AA,SA,CA,ALF,ALFY,CHIP,SCHIP,CCHIP,CHIY,SCHIY,CCHIY,
     *CHIR,SCHIR,CCHIR,STH,CTH,STHL,CTHL,DPHIZ,RTHE,R,VR,XM,SW,SU,SV,Q,
     *VIV(25),DVARS(13),DELXDW(15,3),DELXDR(7,5),DELXD(15,7),WZERO,ALT,
     *XMACH,QDOT,FAA,FAN,A(3,3),CASE,CT2,UMF,A12W,A22W,A32W,XJEXT,BEU,
     *BYL,BEL,BHMAX(15),BHMN,BSTEP(15),HNMB,HMXB,TPOLY,XMIAD

      COMMON/AGEN3/ITHR,IWD,IXR,JUMP,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISION,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)

      COMMON/AGEN4/QALPHA,QBETA,QCN,STH1,CTH1,XJX(3),XJV(3)

      COMMON/AGEN5/CHIBAS,CHPBAS,CHYBAS,CHISAV,TXX,TYY,TZZ,AMX,AMY,AMZ,
     *TMZ,THMFA,SIDE,WMAG,AZW,FANC,FLATC,FOM,VISC

      COMMON/BODYF/TBDWT(15),TXCG(15),TYCG(15),XLEN,XREF,YREF,M1,M2,M5,
     *M6,M7,M8,M9,M50,M51,MSRW,AP(10),AY(10)

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM

      COMMON/FORINT/HBANK(2),T,TT,W,U,V,X,Y,Z,XMI,ZAP(18),DVAR(25),FSAVE
     *(625),NTRG1,TV1,NTRG2,TV2,NTRG3,TV3,NTRG4,TV4,NTRG5,TV5,NTRG6,TV6,
     *NTRG7,TV7,NTRG8,TV8,NTRG9,TV9,NTRG10,TV10,NTRG11,TV11,NTRG12,TV12,
     *NTRG13,TV13,KIND,KRDER,EU,EL,AYL,HMX,HMN,HNM
```



```
      COMMON/GARBGE/DX,DYI,DZI,DXDY,UDXDYI,UDXDY2,XCG,YCG,TX1,TX2,TY1,TY
     *2,ZHAT(3),TXS,TYS,TAU,SINRHO

      COMMON/HEADUP/HDMACH,HDCHRD

      COMMON/KP/KP,KY

      common/mombal/mombal

      COMMON/OLOW/OLOW,WINT

      COMMON/RICONT/TTABLE(20),SIGMA,SIGTAB(20),M15,M16

      common/t_liftoff/t_liftoff

      common/vrwtbl/vrwtbl(30),ivrtsw

      COMMON/XV/XV(4)

      COMMON/MPROPIN/ MPFLAG

      COMMON/MPROPNM/ MPNAME

      COMMON/OUTPUT/OUTTH(6),OUTTV(6),OUTISP(6),OUTWD(6)

      DIMENSION CG(2),CGBL(15,2),PCG(2),ACG(2),BCG(2),CCG(2)

      EQUIVALENCE (XCG,CG(1)),(YCG,CG(2)),(TXCG(1),CGBL(1,1)),
     *(TYCG(1),CGBL(1,2))

c*********************************************************************
c       Initializes the pitch and yaw attitude angles and calculates
c       the roll attitude angle for the booster rolling maneuver
c*********************************************************************

      CHIP=0.
      CHIY=0.

      SCHIY=0.
      CCHIY=1.

      chir=0.

      IF(IHEAD.EQ.1) CHIR=PI
      IF(IHEAD.EQ.0 .AND. XMACH.GT.( HDMACH + .2)) CHIR=PI

      IF(T.LE.TBEGR) CHIR=CHIRO
      IF(NROLL.EQ.1) CHIR=CHIRO+CHRDOT*(T-TBEGR)                        ADER  50
      IF(JUMP.NE.1) CHIR=CHIRO+CHRDOT*(TENDR-TBEGR)                     ADER  51

      SCHIR=SIN(CHIR)                                                   ADER  52
      CCHIR=COS(CHIR)                                                   ADER  54
```

```
      XMIAD=XMI
c*********************************************************************
C**   GET HERE DURING INITIAL LIFT OFF                                ADER  76
c*********************************************************************
      IF(KS1.lt.2) then

         PHIL=DPHIZ+OMEGA*(T-t_liftoff)

         STSP=STH1*SIN(PHIL)
         CPST=STH1*COS(PHIL)

         XHAT=A(1,1)*STSP+A(1,2)*CTH1+A(1,3)*CPST
         YHAT=           A(2,2)*CTH1+A(2,3)*CPST

         SCHIY=A(3,1)*STSP+A(3,2)*CTH1+A(3,3)*CPST
         YX=XHAT*CCHIR-SCHIY*SCHIR
         SCHIY=-XHAT*SCHIR-SCHIY*CCHIR

         CHIP=ATAN2(-YX,YHAT)
         CHIY=ASIN(SCHIY)

         CHPBAS=CHIP
         CHYBAS=CHIY
         CHISAV=CHIP

         IF(IFACT.EQ.3) then

            CALL AMULG(1,M15,20,T,TTABLE,SIGMA,SIGTAB,0,0)

            GO TO 100

         endif

         if(schiy.le.1..and.schiy.ge.-1.) then

            chiy=asin(schiy)

         else if(schiy.le.1.) then

            chiy=1.5*pi

         else

            chiy=pi/2.

         endif

         go to 6

      endif
c*********************************************************************
```



```
c        During the minh phase the attitude angles are calculated by
c        calling subroutine ACNTRO which interpolates from attitude
c        tables
c*********************************************************************
      IF(ITHR.ge.MINH) then

         CALL ACNTRO

         GO TO 6

      endif
c*********************************************************************
c        If shuttle attitude tables are supplied for the first stage, the
c        roll angle (a function of time) is interpolated in this subroutine
c*********************************************************************
      IF(IFACT.eq.3) then

         CALL AMULG(1,M15,20,T,TTABLE,SIGMA,SIGTAB,0,0)

         GO TO 100

      endif
c*********************************************************************
c        If polynomials are used to define the first stage attitude angles
c        (after tiltover), the attitude angles are calculated in subroutine
c        chipol
c*********************************************************************
      IF(ITHR.GE.IPOLY) then

         CALL CHIPOL(KP,KY,T-TPOLY,CHIP,AP,CHIY,AY)

         GO TO 6

      endif

      IF(IFACT.GT.1.AND.T.GT.TTILT) GO TO 101
c*********************************************************************
c        When ifact = 1, the user wishes to calculate the pitch and
c        yaw attitude angles based on the interpolation of the cptbl
c        and cytbl input tables
c*********************************************************************
```

```
      if(ifact.eq.1) then

         if(ivrtsw.eq.1) then

            CALL AMULG(2,M15,30,T,TTBL,CHIPA,CPTBL,chiya,cytbl)

            CHIP=CHIPA/rad
            CHIY=CHIYA/rad

         else

            chip=0.
            chiy=0.

         endif

      endif

c*****************************************************************************

c       When ifact = 2, the attitude is based on a polynomial expression

c*****************************************************************************

      if(ifact.eq.2) then

         call chipol(kp,ky,t-tlift,chip,ap,chiy,ay)

      endif

c*****************************************************************************

    6 SCHIY=SIN(CHIY)                                                   ADER 104
      CCHIY=COS(CHIY)                                                   ADER 105

      SCHIP=SIN(CHIP)                                                   ADER 106
      CCHIP=COS(CHIP)                                                   ADER 107
      CHISAV=CHIP                                                       ADER 108

  101 XV(1)=SCHIP                                                       ADER 109
      XV(2)=CCHIP                                                       ADER 110
      XV(3)=SCHIY                                                       ADER 111
      XV(4)=CCHIY                                                       ADER 112

  100 XM=XMIAD+XMAUG                                                    ADER 113

      if(.not.mombal) return
c*****************************************************************************

c       Delta mass is calculated so that the center of gravity locations
c       can be interpolated

c*****************************************************************************
      IF(.NOT.MPFLAG) THEN
         DELWT=WINT-XM
         IF(ITHR.GT.NOEVNT(1)) DELWT=OLOW-XM
```



```
         IF(DELWT.LT..01)DELWT=.01                                      ADER 116

         CALL SPLINE(2,M5,15,DELWT,TBDWI,CG,CGBL,PCG,1,M6,ACG,BCG,CCG)
      ELSE
         FTTM=.3048
         EVENTNUM=ITHR
         MOBOOS=PROP(2)-ZAP(2)*PFKG
         MOMAIN=PROP(1)-ZAP(1)*PFKG
         CALCPAR=.FALSE.
         CALL MASSPRO(EVENTNUM,MOBOOS,MOMAIN,BOMDOT,MNMDOT,LONCG,
     &                LATCG,NORCG,CALCPAR,LONCGP,LATCGP,NORCGP)
         CG(1)=-(NORCG/12.)*FTTM
         CG(2)=-(LONCG/12.)*FTTM
      ENDIF

      RETURN                                                            ADER 142
      END                                                               ADER 143
```

3.6 Subroutine AEBANK

3.6.1 Purpose

The purpose of this subroutine is to read and store the nonlinear aerodynamic database.

3.6.2 Variable Listing

3.6.3 Subroutines Called:

None

3.6.4 Calling Subroutines:

Name	Function
AINIT	Input routine

3.6.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a	Temporary storage array				Local	
block(1500)	Output storage block			input	Global	sblock
i1	Number of elements in array				Local	
i2	Index				Local	
im	Number of mach points				Local	
in	Index				Local	
kk	Index				Local	
l	Index				Local	
m	Index				Local	
n	Index				Local	
nf	Record number				Local	
nr	Number of record on direct access file				Local	

```
      SUBROUTINE AEBANK(A,NF,IM,I1,I2,NTMACH)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      LOGICAL DEMAND,GRAPH
C
      REAL*4 BLOCK
C
      COMMON/CCC/GRAPH,JO,DEMAND

      COMMON/SBLOCK/BLOCK(1500)

      DIMENSION A(1)
C
      NR=NF
      IN=0

      DO 1 N=1,6

      READ(8'NR,ERR=2) BLOCK

      NR=NR+1
      I=I2
      L=IM*I1*(I-1)

      DO 1 J=1,I1

      M=L+IM*(J-1)

      DO 1 K=1,15

      KK=K+M
      IN=IN+1

    1 A(IN)=DBLE(BLOCK(KK))

      RETURN
C
    2 WRITE(JO,3) NR
    3 FORMAT(1X,'ERROR IN READING RECORD NO.',I6,' IN AEBANK')

      STOP
      END
```

3.7 Subroutine AEOSR

3.7.1 Purpose

The purpose of this subroutine is to store the time and state variables during the forward integration. These stored values are interpolated in the backward integration for determination of the state at discrete times.

3.7.2 Variable Listing

3.7.3 Subroutines Called:

Name	Function
ADER	Calculates the time independent portion of the equations of motion during forward integration
ADER1	Calculates the time dependent portion of the equations of motion during forward integration

3.7.4 Calling Subroutines:

Name	Function
ATHREV	Controls thrust event logic
ATILT	Controls tilt-over logic in forward integration
DPIR	Integration routine defined via AFORUN

3.7.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
dvar(25)	Storage array for the state derivatives		ader	input	Global	forint
ithr	Thrust event index number		athrev	input	Global	agen3
ixr	Index for state variable tables		dpir	output input	Global	agen3
jump	Jump start flag		ainit	inut	Global	agen3
ktbl	Parameter for Tabs common array				Local	
minh	Flag to indicate thrust event which begins min-h attitude optimization		ainit		Global	agen3
nmax	Iteration counter		mastre	input	Global	agen3
nstage	Index number of current stage		athrev	input	Global	mult
qload(26)	Storage array for parameters printed in final loads output tables			output	Global	loads
qtime(26)	Storage array for times associated with parameters in qload array			output	Global	loads
qvalue(13)	Current values of parameters used in load table outputs			input	Global	loads
step(15)	Integration step size		ainit	input	Global	agen1
t	Time from lift-off			input	Global	forint
tbias(6)	Bias between simulated time and reported time (used for branch option)	sec	athrev	input	Global	mult
tbl(ktbl)	Time table for stored state variables	sec		output	Global	tabs
tl	Trigger time for storage of state during forward integration	sec		output	Global	agen1
tne(6,15)	Array which defines the number of engines per thrust event		ainit	input	Global	agen1
u	Y component of plumbline inertial velocity vector	m/sec	dpir	input	Global	forint
ubl(201)	Array for storage of Y component of inertial velocity vector during forward integration	m/sec		output	Global	tabs
v	Z component of plumbline inertial velocity vector	m/sec	dpir	input	Global	forint
vbl(201)	Array for storage of Z component of inertial velocity vector during forward integration	m/sec		input	Global	tabs
w	X component of plumbline inertial velocity vector			input	Global	forint
wbl(201)	Array for storage of X component of inertial velocity vector during forward integration	m/sec		input	Global	tabs

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
wdot_mps	Main propulsion system flowrate tables	kg/sec			Local	
wgt_boost(21	Stored booster mass tables	kg		output	Global	tabs
wgt_mps(201	Stored MPS mass tables	kg		output	Global	tabs
x	X component of plumbline position vector			input	Global	forint
xbl(201)	Array for storage of X component of position vector during forward integration	m		output	Global	tabs
xmbl(201)	Array for storage of vehicle mass (continuous) during forward integration			output	Global	tabs
xmiad	Continuous portion of vehicle mass	kg	ader1	input	Global	agen2
y	Y component of plumbline position vector	m	dpir	input	Global	forint
ybl(201)	Array for storage of Y component of position vector during forward integration	m		output	Global	tabs
z	Z component of plumbline position vector	m	dpir	input	Global	forint
zap(18)	Additional integrated variables	variabl	dpir	input	Global	forint
zbl(201)	Array for storage of Z position component during forward integration	m		output	Global	tabs

```
      SUBROUTINE AEOSR                                                  AEOS   1
C
C      STORES THE STATE DURING THE FORWARD TRAJECTORY
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      parameter (ktbl=201)

      CHARACTER*60 HEAD
      CHARACTER*6 MISION

      COMMON/AGEN1/TIME(2,16),TAUT(15),TAUW(15),TZERO,TLIFT,TTILT,TMINH,
     *DTZ,TQ,TL,XMAUG,TNE(6,15),ENGDAT(8,15),S(15),WD(15),WJET(15),
     *HEAD,PRINT(15),STEP(15),HNOM(15),HMAX(15),PROP(6),TBEGR,TENDR,
     *CHRDOT,CHIRO,FAZ,BO(3,2),CBAXIL,TCHIR,VRCUT,FPRFAC,CHVEL

      COMMON/AGEN2/AA,SA,CA,ALF,ALFY,CHIP,SCHIP,CCHIP,CHIY,SCHIY,CCHIY,
     *CHIR,SCHIR,CCHIR,STH,CTH,STHL,CTHL,DPHIZ,RTHE,R,VR,XM,SW,SU,SV,Q,
     *VIV(25),DVARS(13),DELXDW(15,3),DELXDR(7,5),DELXD(15,7),WZERO,ALT,
     *XMACH,QDOT,FAA,FAN,A(3,3),CASE,CT2,UMF,A12W,A22W,A32W,XJEXT,BEU,
     *BYL,BEL,BHMAX(15),BHMN,BSTEP(15),HNMB,HMXB,TPOLY,XMIAD

      COMMON/AGEN3/ITHR,IWD,IXR,JUMP,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISION,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)

      COMMON/FORINT/HBANK(2),T,TT,W,U,V,X,Y,Z,XMI,ZAP(18),DVAR(25),FSAVE
     *(625),NTRG1,TV1,NTRG2,TV2,NTRG3,TV3,NTRG4,TV4,NTRG5,TV5,NTRG6,TV6,
     *NTRG7,TV7,NTRG8,TV8,NTRG9,TV9,NTRG10,TV10,NTRG11,TV11,NTRG12,TV12,
     *NTRG13,TV13,KIND,KRDER,EU,EL,AYL,HMX,HMN,HNM

      COMMON/LOADS/QLOAD(26),QTIME(26),QVALUE(13),SAVE(20,13)

      common/mult/nchiot(6),tbias(6),nstage,nstg(15),wf(5),ibranch

      COMMON/TABS/TBL(ktbl),WBL(ktbl),UBL(ktbl),VBL(ktbl),XBL(ktbl),
     *YBL(ktbl),ZBL(ktbl),XMBL(ktbl),wgt_mps(ktbl),wgt_boost(ktbl),
     *  wdot_mps(ktbl),wdot_boost(ktbl)
C                                                                       AEOS   16
C     SAVES STATE VARIABLE PTS. FROM FORWARD INTEGRATION                AEOS   17

      IF(NMAX.eq.0.and.JUMP.eq.1) then

         CALL ADER1

         CALL ADER

         DO 11 I=1,13

         IF(QVALUE(I).le.QLOAD(I)) then
            QLOAD(I)=QVALUE(I)
            QTIME(I)=T-tbias(nstage)
         endif

         K=I+13
```



```
         IF(QVALUE(I).ge.QLOAD(K)) then
            QLOAD(K)=QVALUE(I)
            QTIME(K)=T-tbias(nstage)
         endif

   11    CONTINUE

      endif

      IF(ITHR.LT.MINH) RETURN

      IF(TL.GT.T)RETURN                                                 AEOS   18

      IXR=IXR+1                                                         AEOS   19

C**   SET 7 STATES AND TIME IN TABLES                                   AEOS   20

      TBL(IXR)=T                                                        AEOS   21
      WBL(IXR)=W                                                        AEOS   26
      UBL(IXR)=U                                                        AEOS   27
      VBL(IXR)=V                                                        AEOS   28
      XBL(IXR)=X                                                        AEOS   22
      YBL(IXR)=Y                                                        AEOS   23
      ZBL(IXR)=Z                                                        AEOS   24
      XMBL(IXR)=XMIAD                                                   AEOS   25
        wgt_mps(ixr)=zap(1)
        wdot_mps(ixr)=-dvar(8)
        if(tne(2,ithr).ne.0.) then
           wgt_boost(ixr)=zap(2)
           wdot_boost(ixr)=-dvar(9)
        else if(tne(3,ithr).ne.0.) then
           wgt_boost(ixr)=zap(3)
           wdot_boost(ixr)=-dvar(10)
        else
           wgt_boost(ixr)=0.
           wdot_boost(ixr)=0.
        endif

    1 TL=T+STEP(ITHR)-.2                                                AEOS   29

      IF(IXR.LT.ktbl-1) RETURN

C     WRITE OUT SAVED VALUES (4 STATES)                                 AEOS   31

      WRITE(3)(TBL(I),WBL(I),UBL(I),VBL(I),XBL(I),YBL(I),ZBL(I),XMBL(I),
     *wgt_mps(i),wgt_boost(i),wdot_mps(i),wdot_boost(i),I=1,ktbl-1)

      IXR=0                                                             AEOS   34

      RETURN                                                            AEOS   35
      END                                                               AEOS   36
```

3.8 Subroutine AFORND

3.8.1 Purpose

The purpose of this subroutine is to compute constraint errors, set up initial values for the backward integration adjoint equations, and print terminal summary tables.

3.8.2 Variable Listing

3.8.3 Subroutines Called:

Name	Function
ACSTOP	Calculates intermediate and terminal constraints
SUM15STG	Provides weight summary assuming 1.5 stage configuration
SUMARY	Provides weight summary assuming Shuttle configuration
SUMHLLV	Provides weight summary assuming Heavy Lift configuration

3.8.4 Calling Subroutines:

Name	Function
MASTRE	Controls program flow

3.8.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
aa	Launch azimuth	rad	ainit	input	Global	agen2
aerod(30,17)	Aerodynamic coefficient tables		ainit	input	Global	aerodi
alt	Altitude	m	ader	output	Global	agen2
ap(10,2)	Coefficients for booster pitch attitude		atilt	input	Global	body
ask	Change in constraints desired in predicting control variables and parameters				Local	
ay(10,2)	Coefficients for booster yaw attitude		atilt	input	Global	body
azi	Inertial azimuth	deg	aprtn	output	Global	afprn
b	Storage array of printed variables	variabl			Local	
c1	Angular momentum	m^2/se	afornd	output	Global	afprn
c3	Twice the energy	m^2/se	afornd	output	Global	afprn
case	Case number		ainit	output	Global	agen2
cjd	Semi-major axis	m			Local	
cmue	Gravitiational constant	m^3/se	ainit	input	Global	const
cptbl(30)	Booster pitch attitude angle table	rad	athrev	input	Global	contro
cstvar(10)	State storage array	variabl		output	Global	cstop
cytbl(30)	Booster yaw attitude angle table	rad	athrev	input	Global	contro
demand	Logical indicating the operational mode		initial	input	Global	ccc
drho(19)	Difference between current and desired constraint values	variabl	anewch	input	Global	drho
dsireq	Desired value of constraint				Local	
e1	Fraction of optimization step to be made when updating optimization variables		anewch	input	Global	drho
ftbl(15)	Thrust table for main proplusion system	lbs	ainit	input	Global	mps
gam	Inertial flight path angle	rad	ader	output	Global	afprn
gamr	Relative flight path angle	rad			Local	
got	Change in constraints obtained (actual - desired)				Local	
ha	Altitude of apogee	nm			Local	
head	Storage array for headings in printout		ainit	input	Global	agen1
header(20)	Array defining constraint titles		ainit	input	Global	enput
hp	Altitude of perigee	nm			Local	
ii	Number of lines to use in load table				Local	

Variable	Variable Definition output	Units	Source	I/O	Status	Common Block
iunit	Name of units of parameters in load table				Local	
ja	Number of constraints				Local	
jj	Index counter				Local	
jump	Jump start flag		ainit	input	Global	agen3
k1	Index				Local	
k2	Index				Local	
kcdphi(20)	Terminal constraint selection flag array		ainit	input	Global	yli
kcdres(6,5)	Array for intermediate constraint function code		ainit	input	Global	enput
kk	Index				Local	
kl	Index				Local	
lk	Index indicating which orbital velocity data to use as terminal or intermediate constraints			output	Global	vic
lsb	Internal flag to indicate partial derivative runs		badlx	input	Global	rest
mission(2)	Character indicating name of mission		ainit	input	Global	mission
na	Name of parameters in load table output				Local	
name	Name of parameters in load table output				Local	
nbeta	Index used for booster sideslip attitude control		ainit	input	Global	taulim
ncnrs(5)	Number of constraints at restart		ainit	input	Global	rest
nmax	Iteration counter		mastre	input	Global	agen3
noss	Number of constraints		mastre	input	Global	bgen3
ntau	Index used in conjunction with timmps to indicate a throttle event in the booster stage		ainit	input	Global	taulim
ntcn	Number of intermediate constraints		ainit	input	Global	rest
nvnt	Number of thrust events		ainit	input	Global	agen3
nvrst(5)	Array specifying the thrust events where intermediate constraints will be imposed		ainit	input	Global	enput
perc	Percentage of predictability (GOT/ASK*100)				Local	
pfkg	Conversion for pounds to kilograms	kg/lbs	ainit	input	Global	const
phi	Longitude	deg		output	Global	afprn

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
phite(20)	Values of payoff and constraints	variabl		output	Global	yli
predp	Predicted value of payoff function	variabl	anewch	input	Global	lprint
psireq(20)	Desired values of terminal constraints	variabl	ainit	input	Global	yli
psirst(6,5)	Desired values of intermediate constraints	variabl	ainit	input	Global	enput
pstr(20)	Previous values of phite array	variabl	afornd	input output	Global	yli
q	Dynamic pressure	nt/m^2	afornd	output	Global	agen2
r	Radius from earth's center	m	ader	input output	Global	agen2
ra	Radius of apogee	m			Local	
re	Earth's radius	m	ainit	input	Global	const
restx(10,5)	Variable storage array for jump start option		athrev	input	Global	restx
rngan	Range angle	deg		output	Global	afprn
rngo	Ground range	m		output	Global	afprn
rp	Radius of perigee	m			Local	
r r	Saved value of radius	m			Local	
savcp(40)	Storage array for optimization parameters	variabl	afornd	output input	Global	auto
save(20,13)	Storage array	variabl	athrev	input	Global	loads
system	Name of the vehicle system being simulated (SHUTTLE, STAGE15, HLLV)		ainit	input	Global	system
t	Time from lift-off	sec	dpir	input	Global	forint
taut(15)	Time increments for thrust events	sec	anewch	input	Global	agen1
theta	Geocentric latitude of subvehicle point	deg	afornd	output	Global	afprn
thetg	Geodetic latitude of subvehicle point	deg	afornd	output	Global	afprn
time(2,16)	Actual times for thrust events measured from launch	sec	afornd	output	Global	agen1
timmps(15)	Time table for MPS motor	sec	ainit	input	Global	mps
timtau	Time for throttle event in first stage	sec	ainit	input	Global	taulim
u	Y component of plumbline inertial velocity vector	m/sec	dpir	input	Global	forint
v	Z component of plumbline inertial velocity vector	m/sec	dpir	input	Global	forint
vale	Value of constraint from ASCTOP				Local	
vi	Magnitude of inertial velocity vector	m/sec	ader	output	Global	afprn

Variable	Variable Definition	Units	Source	I/O input	Status	Common Block
w	X component of plumbline inertial velocity vector	m/sec	dpir	input	Global	forint
wgnu	Change in optimization parameter with respect to control parameters		anewch	input	Global	drho
wippb	Change in optimization parameter with respect to control variables	variabl	anewch	input	Global	drho
wzero	Initial vehicle weight	lbs	ainit	input	Global	agen2
w_margin(2)	Required value of weight margin at the end of selected stage	kg	ainit	input	Global	multi
x	X component of plumbline position vector	m	dpir	input	Global	forint
xinc	Inclination angle	rad	afornd	output	Global	afprn
xjext	Flag indicating that payoff will be maximized (=1) or minimized (=-1)		ainit	input	Global	agen2
xm	Current vehicle mass	kg	ader1	input	Global	agen2
xmach	Mach number		afornd	output	Global	agen2
xmlb	Vehicle mass	lbs	afornd	output	Global	afprn
xmlb_after	Vehicle weight after event	lbs			Local	
xmlb_before	Vehicle weight before event	lbs			Local	
xnod	Angle between launch longitude and descending node	rad	aprtn	input	Global	afprn
y	Y component of plumbline position vector	m	dpir	input	Global	forint
ylint(7,20)	Initial state variables of the adjoint equations of motions for backward integration		afornd	output	Global	yli
z	Z component of plumbline position vector	m	dpir	input	Global	forint
zap(18)	Additional integrated variables	variabl	dpir	input	Global	forint

```
      SUBROUTINE AFORND                                                 AFOR   1
C*********************************************************************
c       COMPUTES CONSTRAINT ERRORS;SETS UP BACKWARD TRAJECTORY
c       INITIALIZATION; AND PRINTS TERMINAL SUMMARY
C*********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      CHARACTER*12  NAME(13),IUNIT(13)
      character*12 HEADER
      character*20 na
      CHARACTER*6 MISION,mission,nb
      CHARACTER*60 HEAD
      character*7 system
      LOGICAL DEMAND,GRAPH,GLIMIT,QFLG
      COMMON/CCC/GRAPH,JO,DEMAND
      COMMON/AERODI/AEROD(30,17),AEROB(50,6,3),TOMACH(25),DELCA(25),
     *PNMONE(15),BKFABT(50,2),BKFTHR(50,2),DELCN(25),DELCM(25)
      COMMON/AGEN1/TIME(2,16),TAUT(15),TAUW(15),TZERO,TLIFT,TTILT,TMINH,
     *DTZ,TQ,TL,XMAUG,TNE(6,15),ENGDAT(8,15),S(15),WD(15),WJET(15),
     *HEAD,PRINT(15),STEP(15),HNOM(15),HMAX(15),PROP(6),TBEGR,TENDR,
     *CHRDOT,CHIRO,FAZ,BO(3,2),CBAXIL,TCHIR,VRCUT,FPRFAC,CHVEL
      COMMON/AGEN2/AA,SA,CA,ALF,ALFY,CHIP,SCHIP,CCHIP,CHIY,SCHIY,CCHIY,
     *CHIR,SCHIR,CCHIR,STH,CTH,STHL,CTHL,DPHIZ,RTHE,R,VR,XM,SW,SU,SV,Q,
     *VIV(25),DVARS(13),DELXDW(15,3),DELXDR(7,5),DELXD(15,7),WZERO,ALT,
     *XMACH,QDOT,FAA,FAN,A(3,3),CASE,CT2,UMF,A12W,A22W,A32W,XJEXT,BEU,
     *BYL,BEL,BHMAX(15),BHMN,BSTEP(15),HNMB,HMXB,TPOLY,XMIAD
      COMMON/AGEN3/ITHR,IWD,IXR,JUMP,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISION,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)
      COMMON/AFPRN/VI,GAM,C1,C3,XINC,XNOD,XMLB,THETG,THETA,PHI,RNGO,RNGA
     *N,AZI,AZRE
      COMMON/AUTO/KDB(40),SAVCP(40),XLAMB(40,20),DP2
      COMMON/BGEN3/NOSS,JTB,NENT,NACT,NOM1,NP(7)
      COMMON/BODY/TBDWT(15,2),TXCG(15,2),TYCG(15,2),XLEN(2),XREF(2),
     *YREF(2),KP(2),KY(2),AP(10,2),AY(10,2)
      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM
      COMMON/CONTRO/TTBL(30),CPTBL(30),CYTBL(30),TOBL(210),
     *CPOTBL(210),CYOTBL(210)
```



```
      COMMON/CSTOP/CSTVAR(10)
      COMMON/DRHO/DRHO(19),E1,WIPPB,WGNU
      COMMON/ENPUT/PSIRST(6,5),HEADER(20),NVRST(5),KCDRES(6,5),LAST
      COMMON/FORINT/HBANK(2),T,TT,W,U,V,X,Y,Z,XMI,ZAP(18),DVAR(25),
     *FSAVE(625),NTRG1,TV1,NTRG2,TV2,NTRG3,TV3,NTRG4,TV4,NTRG5,TV5,
     *NTRG6,TV6,NTRG7,TV7,NTRG8,TV8,NTRG9,TV9,NTRG10,TV10,NTRG11,TV11,
     *NTRG12,TV12,NTRG13,TV13,KIND,KRDER,EU,EL,AYL,HMX,HMN,HNM
      COMMON/LOADS/QLOAD(26),QTIME(26),QVALUE(13),SAVE(20,13)
      COMMON/LPRINT/LPRINT,INMAX,PREDP,QY,TOL
      common/mission/mission(2)
      COMMON/MPS/TIMMPS(15),FTBL(15),THROT(15),GLIM(15),TAUALT(5),
     *TAUTBL(5),MPS1,MPS2,TGLIM(10,15),GLIMIT,tau_const(10,5),jthrot(15)
      common/multi/char_vel(5),fpr_fac(5),w_margin(2)
      COMMON/QMAX/QMAX,QFLG,QPEN,MAXQ
      COMMON/REST/JSTR,NCNRS(5),NTCN,TR(5),LSB,NF8,NOPAR,WIBT(15)
      COMMON/RESTX/RESTX(10,5)
      common/system/system
      COMMON/TAULIM/TAULIM,NTAU,NBETA,TIMTAU
      COMMON/VIC/VIC(2),XK1(2),XK2(2),XK3(2),XK4(2),LK
      COMMON/YLI/YLINT(7,20),PHITE(20),PSIREQ(20),KCDPHI(20),PSTR(20)
      DIMENSION GOT(20),ASK(20),PERC(20),b(160)
      DATA NAME/' Q ALPHA    ',' Q BETA     ',' Q          ',
     *' LONG ACC   ',' LAT ACC    ',' NORM ACC   ',' TOTAL ACC  ',
     *' STAG HEAT  ',' HEAT RATE  ',' PDOME      ',' YRING      ',
     *' ORBET      ',' SRMET      '/
      DATA IUNIT/'LB-DEG/FT**2','LB-DEG/FT**2',' LB/FT**2  ',
     *'           ','    G''S    ','    G''S    ','    G''S    ',
     *'           ','    G''S    ',' BTU/FT**2  ',' BTU/FT**2/S',
     *'           ','    PSI     ','    PSI     ','    LBS     ',
     *'           ','    LBS     '/
c*********************************************************************
C**   SET LAST STATE POINT IN TABLES                                   AFOR  40
C     EVALUATE TERMINAL CONDITIONS AND INITIALIZE LAMBDAS              AFOR  41
C*****NOSS AT THIS POINT IS EQUAL TO REGULAR SETS-SET BY ANEWCH        AFOR  42
```

```
c*********************************************************************

      TIME(1,NVNT+1)=T

      CSTVAR(1)=W                                                    AFOR  43
      CSTVAR(2)=U                                                    AFOR  44
      CSTVAR(3)=V                                                    AFOR  45
      CSTVAR(4)=X                                                    AFOR  46
      CSTVAR(5)=Y                                                    AFOR  47
      CSTVAR(6)=Z                                                    AFOR  48
      CSTVAR(7)=XM                                                   AFOR  49
      CSTVAR(8)=T                                                    AFOR  51
      CSTVAR(9)=ZAP(7)
      CSTVAR(10)=0.

      CALL ACSTOP(0,VALE,YLINT,I)

      LK=0
      DO I=1,7
         DO J=1,20
            YLINT(I,J)=0.
         enddo
      enddo

      DO I=1,NOSS
         KL=KCDPHI(I)
         IF(KL.EQ.14.OR.KL.GE.18.AND.KL.LE.20) LK=1

         CALL ACSTOP(KL,VALE,YLINT,I)

         IF(I.ne.1) then
            DSIREQ=PSIREQ(I-1)
            PHITE(I)=VALE-DSIREQ
         else
            PHITE(I)=VALE
         endif
      enddo

      JA=NOSS
      RR=R

      DO L=1,5

         K=6-L

         IF(NVRST(K).ne.0) then

c*********************************************************************

C**** IF NVRST.NE.0 EVALUATE RESTART TERMINAL COND AND INIT LAMS        AFOR  64

c*********************************************************************

c           rr=r

            DO I=1,10
```



```
               CSTVAR(I)=RESTX(I,K)
            enddo

            CALL ACSTOP(0,VALE,YLINT,I)

            NCN=NCNRS(K)

            DO I=1,NCN
               JA=JA+1
               KL=KCDRES(I,K)

               IF(KL.EQ.14.OR.KL.GE.18.AND.KL.LE.20) LK=2

               CALL ACSTOP(KL,VALE,YLINT,JA)

               DSIREQ=PSIRST(I,K)
               PHITE(JA)=VALE-DSIREQ
            enddo
         endif
      enddo

      R=RR                                                           AFOR  75

c*********************************************************************

C**   WRITE OUT TEST MATRIX                                          AFOR  76

c*********************************************************************

   15 IF(LSB.EQ.1) RETURN                                            AFOR  77

      K=NOSS+NTCN

      IF(LSB.eq.0) then

         WRITE(JO,56) CASE,HEAD
   56    FORMAT(1H1,////' MASTRE CASE=',F11.4,10X,A60//)

      else

         WRITE(JO,17) CASE,HEAD
   17    FORMAT(////' MASTRE CASE=',F11.4,10X,A60//)

      endif

      ASK(1)=E1*XJEXT*(WIPPB-WGNU)                                   AFOR  88
      JA=NOSS+NTCN-1
      ASK(1)=ASK(1)-PREDP                                            AFOR  90

      DO I=1,JA
         ASK(I+1)=DRHO(I)
      enddo

      JA=JA+1                                                        AFOR  93

      DO I=1,JA
```

```
         GOT(I)=PHITE(I)-PSTR(I)
         IF(ASK(I).EQ.0..OR.GOT(I).EQ.0.)THEN
            PERC(I) = 0.
         else
            PERC(I)=GOT(I)/ASK(I)*100.
         endif
      enddo

      kk=(ja-1)/10+1

      do j=1,kk

         k1=(j-1)*10+1
         k2=k1+9
         if(k2.gt.ja) k2=ja

         WRITE(JO,'(///,7x,10a12)')(HEADER(I),I=k1,k2)

         WRITE(JO,8)(PSTR(I),I=k1,k2)
         WRITE(JO,9)(PHITE(I),I=k1,k2)
         WRITE(JO,10)(ASK(I),I=k1,k2)
         WRITE(JO,11)(GOT(I),I=k1,k2)
         WRITE(JO,12)(PERC(I),I=k1,k2)
      enddo

    8 FORMAT(1X,'OLD    ',10E12.5)
    9 FORMAT(1X,'NEW    ',10E12.5)
   10 FORMAT(1X,'ASK    ',10E12.5)
   11 FORMAT(1X,'GOT    ',10E12.5)
   12 FORMAT(1X,'PCNT   ',10E12.5)

      IF(DEMAND) then

         kk=(ja-1)/5+1

         do j=1,kk

            k1=(j-1)*5+1
            k2=k1+4
            if(k2.gt.ja) k2=ja

            if(j.eq.1) then
               write(*,'(//,4x,a,5a12)') 'case  ',(header(i),i=k1,k2)
               write(*,'(2x,f6.4,2x,5e12.5)') case,(phite(i),i=k1,k2)
            else
               write(*,'(/,10x,5a12)') (header(i),i=k1,k2)
               write(*,'(10x,5e12.5)') (phite(i),i=k1,k2)
            endif

         enddo

C-------------------------------------------------------------------------
         WRITE(10) JA,K,CASE,header,phite
C-------------------------------------------------------------------------
      endif
```



```
      DO I=1,JA
         PSTR(I)=PHITE(I)
      enddo

      IF(LSB.NE.0) RETURN

      WRITE(JO,31) AP,AY                                                AFOR 11

   31 FORMAT(//' POLYNOMIAL COEFFICIENTS'/1X,'AP',5E15.8,3(/3X,5E15.8)
     */1X,'AY',5E15.8,3(/3X,5E15.8))

      IF(NMAX.NE.0) RETURN                                              AFOR 113

      na='ENDPRT              '

      write(9) na,b,nb

      DO I=1,15
         SAVCP(I)=TAUT(I)
      enddo

      SAVCP(16)=WZERO/.45359237                                         AFOR 116
      SAVCP(17)=AA*RAD                                                  AFOR 117

      DO I=1,5
         SAVCP(I+17)=CPTBL(I)
         SAVCP(I+25)=CYTBL(I)
      enddo

      SAVCP(23)=CPTBL(6)                                                AFOR 124
      SAVCP(24)=CPTBL(7)                                                AFOR 125
      SAVCP(25)=CPTBL(8)                                                AFOR 126
      SAVCP(31)=TIMMPS(NTAU)
      SAVCP(32)=TIMTAU
      SAVCP(33)=FTBL(NTAU+1)
      SAVCP(34)=AEROD(NBETA,15)
      SAVCP(35)=AEROD(NBETA+1,15)
      SAVCP(36)=AEROD(NBETA+2,15)
      savcp(37)=w_margin(1)*pfkg
      savcp(38)=w_margin(2)*pfkg

      WRITE(JO,30) (SAVCP(I),I=1,38)

   30 FORMAT(/1X,'PARAMETERS'/4X,'TAUT'/8X,8E15.8/8X,7E15.8/5X,'WO1'/
     *8X,E15.8/6X,'AZ'/8X,E15.8/3X,'CPTBL'/8X,8E15.8/3X,'CYTBL'/8X,5E15.
     *8/3X,'THROTTLE PARAMETERS'/8X,3E15.8/3X,'SIDESLIP CONTROL'/8X,
     *3E15.8/3x,'MARGIN'/8x,2e15.8)

      rewind 9

      jj=0

50    read(9,end=51) na,b,nb

         if(na.eq.'INJECTION           ') then
            t=b(1)
```

```
            rngo=b(35)
            rngan=b(36)
            vi=b(8)*.3048
            r=b(7)*.3048
            gam=b(9)
            xinc=b(34)
            wlo=SAVCP(16)*.45359237
            c1=r*vi*cos(gam/rad)
            xmlb=b(6)
            thetg=b(17)
            theta=b(11)
            phi=b(12)
            azi=b(10)
            xnod=b(18)

            c3=vi*vi-2.*cmue/r
            p=c1**2/cmue
            CJD=CMUE/ABS(C3)
            E=SQRT(ABS(1.-(P/CJD)))
            IF(E.le..00005) e=0.

            RA=CJD*(1.+E)
            RP=CJD*(1.-E)
            HA=(RA-RE)/1852.
            HP=(RP-RE)/1852.

            WRITE(JO,'(1h1)')
            write(21,'(1h1)')

            WRITE(14) HEAD,T,RNGO,RNGAN,VI,R,GAM,XINC,XNOD,
     *      SAVCP(17),THETG,THETA,PHI,AZI,C1,XMLB,C3,E,RA,RP,HA,HP

            WRITE(JO,38) HEAD
            WRITE(21,38) HEAD

   38       FORMAT(////29X,A60//)

            jj=jj+1

            write(jo,'(a,a)') ' case =',mission(jj)
            write(21,'(a,a)') ' case =',mission(jj)

            WRITE(JO,36)T,RNGO,RNGAN,VI,R,GAM,XINC,XNOD,SAVCP(17),
     *      THETG,THETA,PHI,AZI,C1,XMLB,C3

            WRITE(21,36)T,RNGO,RNGAN,VI,R,GAM,XINC,XNOD,SAVCP(17),
     *      THETG,THETA,PHI,AZI,C1,XMLB,C3

            WRITE(JO,40) E,RA,RP,HA,HP
            WRITE(21,40) E,RA,RP,HA,HP

         endif

         go to 50

51     continue
```



```
      SAVCP(17)=AA

   36 FORMAT(//5X,'TIME',f15.6,' SEC',12X,'RANGE',f15.2,' NM',9X,
     *'RANGE ANGLE',f15.6,' DEG'///' INERTIAL',f15.4,' M/SEC',9X,
     *'RADIUS',f15.2,' M',11X,'FLIGHT',4X,f15.6,' DEG'/' VELOCITY'
     *,62X,'PATH ANGLE'///5X,'INCL',f15.6,' DEG',13X,'DES.',f15.6,
     *' DEG',12X,'FLIGHT',1X,f15.6,' DEG'/40X,'NODE',30X,'AZIMUTH'///
     *4X,'GDLAT',f15.6,' DEG',12X,'GCLAT',f15.6,' DEG',15X,'LONG',
     *f15.6,' DEG'///' INERTIAL',f15.6,' DEG',14X,'C1',1X,E15.8,
     *' M**2/SEC'/2X,'AZIMUTH'///3X,'WEIGHT',f15.2,' LBS',14X,'C3',
     *1X,E15.8,' M**2/SEC**2'///)

   40 FORMAT(1X,'KEPLERIAN ORBITAL PARAMETERS'//1X,'ECNTRICY',f15.6,15X
     *,'APOGEE',f15.2,' M',14X,'PERIGEE',f15.2,' M'/38X,'RADIUS',31X
     *,'RADIUS'///3X,'APOGEE',f15.6,' NM',11X,'PERIGEE',f15.6,' NM'/
     *3X,6HHEIGHT,29X,6HHEIGHT)

      if(system.eq.'STAGE15') then
         call sum15stg
      else if(system.eq.'HLLV') then
         call sumhllv
      else
         call sumary
      endif

      IF(JUMP.NE.1) RETURN

      rewind 9

   55 read(9,end=54) na,b,nb

      if(na.ne.'SEPARATION          ') then

         t=b(1)
         alt=b(13)
         q=b(103)
         axacc=b(124)
         xmach=b(97)
         vsubr=b(14)
         gamr=b(15)
         vsubi=b(8)
         gami=b(9)
         xmlb_before=b(6)

         go to 55

      endif

      xmlb_after=b(6)

      do 45 j=1,2

      if(mission(j).eq.'      ') go to 45

      WRITE(JO,'(1h1)')
```

```
      WRITE(JO,38) HEAD

      WRITE(21,'(1h1)')
      WRITE(21,38) HEAD

      write(jo,'(1x,a,5x,a6)') ' case =',mission(j)
      write(21,'(1x,a,5x,a6)') ' case =',mission(j)

      WRITE(JO,41)
      WRITE(21,41)

      if(system.eq.'SHUTTLE') then

         ii=13

      else

         ii=11

      endif

      if(j.eq.1) then

         DO I=1,ii

         WRITE(JO,42) NAME(I),(SAVE(K,I),K=1,8),IUNIT(I)
         WRITE(21,42) NAME(I),(SAVE(K,I),K=1,8),IUNIT(I)

         enddo

      else

         do i=1,ii

         write(jo,42) name(i),(save(k,i),k=1,4),
     *   (save(k,i),k=9,12),iunit(i)
         write(21,42) name(i),(save(k,i),k=1,4),
     *   (save(k,i),k=9,12),iunit(i)

         enddo

      endif

      write(jo,53) t,xmlb_before,xmlb_after,alt,vsubi,gami,q,
     *vsubr,gamr,axacc,xmach
      write(21,53) t,xmlb_before,xmlb_after,alt,vsubi,gami,q,
     *vsubr,gamr,axacc,xmach

45    continue

      WRITE(2) HEAD,NAME,SAVE,IUNIT

   41 FORMAT(///33X,'FIRST STAGE',36X,'SECOND STAGE'/24X,'MINIMUM',14X,
     *'MAXIMUM',20X,'MINIMUM',14X,'MAXIMUM'/2X,'LOAD INDICATOR',2X,
     *2(2X,'TIME',6X,'VALUE',4X),6X,2(2X,'TIME',6X,'VALUE',4X),8X,
     *'UNITS'//)
```



```
   42 FORMAT(2X,A12,4X,2(F6.2,F11.2,4X),6X,2(F6.2,F11.2,4X),4X,
     *A12/)

53    format(///1x,'STAGING CONDITIONS'//
     *1x,'Time    ',f12.3,' sec',5x,'Wgt-pre ',f12.1,' lbs',
     *5x,'Wgt-post',f12.1,' lbs'//
     *1x,'Altitude',f12.1,' ft ',5x,'Vel Iner',f12.2,' fps',
     *5x,'Gamma I ',f12.5,' deg'//
     *1x,'Dyn Pres',f12.3,' psf',5x,'Vel Rel ',f12.2,' fps',
     *5x,'Gamma R ',f12.5,' deg'//
     *1x,'Ax Acc  ',f12.3,' gs ',5x,'Mach No.',f12.2)

      RETURN                                                          AFOR 165
54    write(*,*) ' error in reading tape 9'
      END                                                             AFOR 166
```

3.9 Subroutine AFORUN

3.9.1 Purpose

The purpose of this subroutine is to control the setup logic for the forward trajectory and call the integration subroutines DESOLV and DPIR.

3.9.2 Variable Listing

3.9.3 Subroutines Called:

Name	Function
ADER	Calculates the time independent portion of the equations of motion during forward integration
ADER1	Calculates the time dependent portion of the equations of motion during forward integration
AEOSR	Stores state variables during forward integration
AJUMP	Controls first stage jump start logic
ALIFT	Controls liftoff based on thrust-to-weight
APRTN	Prints block trajectory format and output files
AQMAX	Controls printing of maximum dynamic pressure
AROLL	Controls roll timing logic
ATHREV	Controls thrust event logic during forward integration
ATHRO	Controls throttle phase
ATILT	Controls tiltover phase of trajectory
AWDEV	Controls weight drop logic
AXPRT	Controls normal printout
DESOLV	Initializes integration parameters and calls integration subroutine
DISPPRT	Controls printing of output to dispersion file
FIND	Calculates pitch and yaw attitude using polynomial
GLIMT	Controls and calculates acceleration limit timing during forward integration

3.9.4 Calling Subroutines:

Name	Function

BADLX	Calculates partial derivatives for control parameters
MASTRE	Controls total program logic

3.9.5 Fortran Listing

aforun

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a(3,3)	Tranformation matrix from equatorial to plumbline coordinate systems		aforun	output	Global	agen2
a12w	Plumbline X component of earth spin axis		aforun	output	Global	agen2
a22w	Plumbline Y component of earth spin axis		aforun	output	Global	agen2
a32w	Plumbline Z component of earth spin axis		aforun	output	Global	agen2
aa	Launch azimuth	rad	ainit	input	Global	agen2
ap(10)	Coefficients for booster pitch attitude		aforun	output input	Global	bodyf
api(10,2)	Coefficients for booster pitch attitude		aforun	output	Global	body
astor(4,67)	Storage array for weight output table	lbs	ainit	input	Global	astor
ay(10)	Coefficients for booster yaw attitude		aforun	output input	Global	bodyf
ayi(10,2)	Coefficients for booster yaw attitude		aforun	output	Global	body
ayl	Lower dependent variable value for auto step size logic	sec	ainit	input	Global	forint
blow	Storage variable for blow (booster liftoff weight)	kg			Local	
blows	Storage variable for blow (booster liftoff weight)	kg			Local	
ca	Cosine of launch azimuth		aforun	output	Global	agen2
cchip	Cosine of pitch attitude angle		aforun	output	Global	agen2
chibas	Total vehicle attitude bias at launch	rad	aforun	output	Global	agen5
chip	Pitch attitude angle	rad	aforun	output	Global	agen2
chir	Roll attitude angle	rad	aforun	output	Global	agen2
chirf	Final value of roll attitude after first stage roll maneuver	rad			Local	
chiro	Initial value of roll attitude angle	rad	atilt	output input	Global	agen1
chiy	Yaw attitude angle	rad	aforun	output	Global	agen2
chpbas	Pitch attitude bias angle	rad	aforun	output	Global	agen5
chrdot	Roll attitude rate	rad/sec	aforun	output input	Global	agen1
chybas	Yaw attitude bias angle	rad	aforun	output	Global	agen5
cpotbl(210)	Pitch attitude table during min-H	rad	anewch	input	Global	contro
cptbl(30)	Booster pitch attitude angle table	deg	ainit	input	Global	contro

aforun

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
cth1	Cosine of launch colatitude		ainit	input	Global	agen4
cyotbl(210)	Yaw attitude table during min-H	rad	anewch	input	Global	contro
cytbl(30)	Booster yaw attitude angle table	deg	ainit	input	Global	contro
disp_var	Array of output variables used in dispersion analysis				Local	
el	Lower limit for stepsize control	sec	ainit	input	Global	forint
eu	Upper limit for stepsize control	sec	ainit	input	Global	forint
faz	Initial pointing direction of orbiter tail	rad	ainit	input	Global	agen1
fom	Designation that thrust-to-weight will be used as trigger value in integration	g's		output	Global	agen5
fsave(625)	Storage array for integration routine		aforun	output	Global	forint
ftbl(15)	Thrust table for main proplusion system	lbs	ainit	output input	Global	mps
glimit	Logical variable indicating aceleration limit has been achieved		aforun	output	Global	mps
hdmach	Mach number lower limit for second roll maneuver		ainit	input	Global	headup
hmn	Minimum step size when using variable step integration routine	sec	ainit	input	Global	forint
hmx	Maximum step size	sec	athrev	input	Global	forint
ifact	Flag indicating the type of attitude used in the booster stage		atilt	input	Global	agen3
ihead	Roll attitude flag		aforun	output	Global	agen3
ipr	Denotes the thrust event from which fuel propellant reserve (FPR) is calculated		ainit	input	Global	ipr
iprop(6)	Propulsion cutoff flags (stage dependent)		ainit	input	Global	ipr
irec	Number of record on direct access storage file for dispersion data file		dispprt	output input	Global	disp
ithr	Thrust event index number		aforun	output	Global	agen3
ivar	Index used to denote the type of dispersion case		ainit	input	Global	disp
iwd	Index for weight drop events		aforun	output	Global	agen3
iwdj	Index for weight drop events				Local	
ixr	Index for state variable tables		aforun	output	Global	agen3
jjump	Flag indicating that all runs between the first iteration and the converged iteration can be simulated by jump start		ainit	input	Global	jjump

Variable	Variable Definition at the beginning of the minh phase	Units	Source	I/O	Status	Common Block
jump	Jump start flag		ainit	output input	Global	agen3
jumps	Saved value of jump				Local	
jumpst	Saved value of jump		athrev	input	Global	jumpst
kcytab	Attitude control table index		ainit	input output	Global	agen3
kerr	Integration error indicator				Local	
key	Mode switch to indicate the portion of the routine to execute		aforun	output	Global	disp
kind	Integration type flag for forward trajectory		ainit	input	Global	forint
kjum	Flag to indicate internal jump start				Local	
kkkk	Index for interpolation routine in atmosphere routines		aforun	output	Global	mlast
kp	Order of booster pitch attitude polynomial		aforun	output	Global	kp
kpi(2)	Order of booster pitch attitude polynomials		ainit	input	Global	body
krder	Order of differences for forward integration		ainit	input	Global	forint
ks1	Booster attitude indicator		aforun	input output	Global	agen3
ks2	Storage variable for ks1 (attitude flag)				Local	
ky	Order of booster yaw attitude polynomial		aforun	output	Global	kp
kyi(2)	Order of booster yaw attitude polynomials		ainit	input	Global	body
lines	Line counter for printout		aforun	output	Global	agen3
lldsol	Variable dimension used in integration routine				Local	
load	Integration option				Local	
lsb	Internal flag to indicate partial derivative runs		badlx	input	Global	rest
lvar	Index to terminate read				Local	
m15	Pointer index for roll angle (sigma) lookup		aforun	output	Global	ricont
m16	Pointer index for pitch and yaw angle (theta & phi) lookup		aforun	output	Global	ricont

aforun

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
minh	Flag to indicate thrust event which begins min-h attitude optimization		ainit	input	Global	agen3
mision	Character indicating name of mission		aforun	output	Global	agen3
mission(2)	Character indicating name of mission		ainit	input	Global	mission
mlast	Previous index in atmospheric interpolation		aforun	output	Global	mlast
mmmm	Pointer index for atmospheric parameters interpolation		aforun	output	Global	mlast
n	Index				Local	
naltw	Index used when calculating launch azimuth partial derivatives		badlx	input	Global	naltw
nchiot(6)	Optimization flag used with branch trajectory option		ainit	input	Global	mult
ncoast	Thrust event to begin coast on final case		ainit	input	Global	agen3
nendct(7)	Index indicating end of min-H phases		ainit	input	Global	agen3
nf8	Number of differential equations to be integrated		aforun	output	Global	rest
njump	Number of jump start times to be saved		aforun	output	Global	time_jump
nmax	Iteration counter		aforun	output	Global	agen3
noevnt(5)	Number of thrust events per stage		ainit	input	Global	agen3
notrg	Number of triggers to be used in integration				Local	
nowd(15)	Index indicating thrust event where weight drop event will occur		ainit	input	Global	agen3
np(7)	Number of points in attitude tables during min-H phases		ainit	input	Global	bgen3
nroll	Flag indicating booster roll program after launch		aforun	output	Global	agen3
nstage	Index number of current stage		aforun	output	Global	mult
nstg(15)	Internal index which relates thrust event to stage number		ainit	input	Global	mult
ntau	Index used in conjunction with timmps to indicate a throttle event in the booster stage		ainit	input	Global	taulim
ntilt	Flag indicating end of tiltover phase of flight		aforun	output	Global	ntilt
ntrg1	MPS throttle trigger number		aforun	output	Global	forint
ntrg12	Weight drop event trigger for forward		aforun	output	Global	forint

Variable	Variable Definition trajectory	Units	Source	I/O	Status	Common Block
ntrg13	MPS throttle trigger number		aforun	output	Global	forint
ntrg14	Trigger for roll maneuver		aforun	output	Global	trigc
ntrg15	Trigger flag for dispersion output	sec			Local	
ntrg2	Thrust event trigger number		aforun	output	Global	forint
ntrg3	Relative velocity cutoff trigger for forward trajectory		aforun	output	Global	forint
ntrg4	Booster attitude trigger for forward trajectory		aforun	output	Global	forint
ntrg6	Maximum dynamic pressure trigger for forward trajectory		aforun	output	Global	forint
ntrg7	Liftoff trigger number		aforun	output	Global	forint
ntrg9	Acceleration limit trigger number		aforun	output	Global	forint
nvnt	Number of thrust events		ainit	input	Global	agen3
nwvnt	Number of weight drop events		ainit	input	Global	agen3
olow	Orbiter liftoff weight	kg	athrev	input output	Global	olow
olows	Storage variable for olow (orbiter liftoff weight)	kg			Local	
omega	Earth's spin rate	rad/sec	ainit	input	Global	const
pi	Pi constant (3.14159265)		ainit	input	Global	const
prop(6)	Propellant weight for each engine type	kg	ainit	input	Global	agen1
ptkg	Conversion from pounds to kilograms	kg/lbs	ainit	input	Global	const
qdot	Designation that the time derivative of dynamic pressure will be used as a trigger value		aforun	output	Global	agen2
qflg	Logical variable indicating booster MPS throttle being simulated		aforun	output	Global	qmax
qload(26)	Storage array for parameters printed in final loads output tables		aforun	output	Global	loads
qmax	Maximum value of dynamic pressure	psf	atilt	input	Global	qmax
qnom	Storage variable for qmax (maximum dynamic pressure)				Local	
qpen	Maximum value of dynamic pressure	kg/m^2	aforun	output	Global	qmax
qtime(26)	Storage array for times associated with parameters in qload array	sec	aforun	output	Global	loads
sa	Sine of launch azimuth		aforun	output	Global	agen2

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
save_mass	Storage variable for final mass used in branch option		aforun	output	Global	save_mass
schip	Sine of pitch attitude angle		aforun	output	Global	agen2
sth1	Sine of launch colatitude		ainit	input	Global	agen4
system	Name of the vehicle system being simulated (SHUTTLE, STAGE15, HLLV)		ainit	input	Global	system
t	Time from lift-off	sec	aforun	input	Global	forint
taulim	Rate of change of throttle limit	/sec	ainit	input	Global	taulim
taut(15)	Time increments for thrust events	sec	anewch	input	Global	agen1
tauw(15)	Time increment for weight drop event	sec	ainit	input	Global	agen1
tbegr	Time to begin roll maneuver	sec	aforun	output	Global	agen1
tchir	Time to begin roll maneuver	sec	ainit	input	Global	agen1
tendo	Time required for booster roll program	sec			Local	
tendr	Time to terminate roll maneuver	sec	ainit	output input	Global	agen1
time(2,16)	Actual times for thrust events measured from launch	sec	aforun	output	Global	agen1
time_jump(5)	Time of jump start	sec	ainit	input	Global	time_jump
timmps(15)	Time table for MPS motor	sec	ainit	output input	Global	mps
tl	Trigger time for storage of state during forward integration	sec	aforun	output	Global	agen1
tlift	Time to begin tiltover maneuver	sec	ainit	input output	Global	agen1
tlifts	Storage variable for tlift (time to begin tiltover maneuver)	sec			Local	
tq	Time that the minh phase is initiated	sec	aforun	output	Global	agen1
tsavet	Time storage variable for restart	sec	athrev	input	Global	jumpst
ttilt	Time to begin tiltover maneuver	sec	aforun	output	Global	agen1
tv1	Time to activate print trigger	sec	aforun	output	Global	forint
tv12	Time to activate weight drop event	sec	aforun	output	Global	forint
tv13	Time to activate booster MPS throttle trigger	sec	aforun	output	Global	forint
tv14	Mach number to start upper stage roll maneuver		aforun	output	Global	trigc
tv15	Time to activate dispersion output file creation	sec	aforun	output	Global	disp

aforun

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
tv2	Time to activate thrust events	sec	aforun	output	Global	forint
tv3	Value of relative velocity to activate trigger	m/sec	aforun	output	Global	forint
tv4	Time to activate booster attitude phase	sec	aforun	output	Global	forint
tv6	Value of qdot to activate maximum dynamic pressure trigger (0)	kg/m^2	aforun	output	Global	forint
tv7	Value of thrust-to-weight to activate liftoff trigger (1.)	g's	aforun	output	Global	forint
tv9	Time to activate acceleration limit trigger	sec	aforun	output	Global	forint
tzero	Time of simulation initiation	sec	ainit	input	Global	agen1
tzeros	Storage variable for tzero (initial time)				Local	
t_liftoff	Time of lift-off	sec	ainit	input	Global	t_liftoff
t_liftoffs	Storage variable for t_liftoff (liftoff time)				Local	
u	Y component of plumbline inertial velocity vector	m/sec	aforun	output	Global	forint
uz	Y plumbline component of launch site inertial velocity vector	m/sec			Local	
v	Z component of plumbline inertial velocity vector	m/sec	aforun	output	Global	forint
viv(25)	Initial condition array for jump start		ainit	input	Global	agen2
viv_sav(26)	Saved values of state variable for first stage restart	variabl	ainit	input	Global	viv_sav
vz	Z plumbline component of launch site inertial velocity vector	m/sec			Local	
w	X component of plumbline inertial velocity vector	m/sec	aforun	output	Global	forint
wd(15)	Weight dropped during weight drop event	kg	ainit	input	Global	agen1
wint	Initial vehicle weight	kg	ainit	input output	Global	olow
wjet(15)	Jettison weight per thrust event	kg	ainit	input	Global	agen1
wz	Initial w velocity component	lbs			Local	
wzero	Initial vehicle weight	lbs	ainit	input	Global	agen2
wzeros	Storage variable for wzero (initial weight)	lbs			Local	
wzerst	Stored value of wzero	lbs	athrev	input	Global	jumpst
x	X component of plumbline position	m	aforun	output	Global	forint

Variable	Variable Definition vector	Units	Source	I/O	Status	Common Block
xjv(3)	Components of plumbline inertial velocity at launch	m/sec	ainit	input	Global	agen4
xjx(3)	Components of plumbline position vector at launch	m	ainit	input	Global	agen4
xm	Current vehicle mass	kg	aforun	output	Global	agen2
xmach	Designation that mach number will be used for trigger		ader	input	Global	agen2
xmaug	Auxiliary vehicle mass	kg	athrev	output input	Global	agen1
xmi	Integrated vehicle mass less jettisoned inerts.	kg	aforun	input	Global	forint
xmiad	Continuous portion of vehicle mass	kg	aforun	output	Global	agen2
xz	X plumbline component of launch site radius vector	m			Local	
y	Y component of plumbline position vector	m	aforun	output	Global	forint
yz	Y plumbline component of launch site radius vector	m			Local	
z	Z component of plumbline position vector	m	aforun	output	Global	forint
zap(18)	Additional integrated variables	variabl	aforun	output	Global	forint
zz	Z plumbline component of launch site radius vector	m			Local	

```
      SUBROUTINE AFORUN                                                 AFOR   1
C*********************************************************************
C       Controls the setup logic for the forward trajectory and calls
C       the integration package desolv
C*********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      LOGICAL DEMAND,GRAPH,GLIMIT,QFLG,VISC

      CHARACTER*60 HEAD
      CHARACTER*6 MISION,MISSION
      character*7 system

      COMMON/CCC/GRAPH,JO,DEMAND

      COMMON/AGEN1/TIME(2,16),TAUT(15),TAUW(15),TZERO,TLIFT,TTILT,TMINH,
     *DTZ,TQ,TL,XMAUG,TNE(6,15),ENGDAT(8,15),S(15),WD(15),WJET(15),
     *HEAD,PRINT(15),STEP(15),HNOM(15),HMAX(15),PROP(6),TBEGR,TENDR,
     *CHRDOT,CHIRO,FAZ,BO(3,2),CBAXIL,TCHIR,VRCUT,FPRFAC,CHVEL

      COMMON/AGEN2/AA,SA,CA,ALF,ALFY,CHIP,SCHIP,CCHIP,CHIY,SCHIY,CCHIY,
     *CHIR,SCHIR,CCHIR,STH,CTH,STHL,CTHL,DPHIZ,RTHE,R,VR,XM,SW,SU,SV,Q,
     *VIV(25),DVARS(13),DELXDW(15,3),DELXDR(7,5),DELXD(15,7),WZERO,ALT,
     *XMACH,QDOT,FAA,FAN,A(3,3),CASE,CT2,UMF,A12W,A22W,A32W,XJEXT,BEU,
     *BYL,BEL,BHMAX(15),BHMN,BSTEP(15),HNMB,HMXB,TPOLY,XMIAD

      COMMON/AGEN3/ITHR,IWD,IXR,JUMP,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISION,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)

      COMMON/AGEN4/QALPHA,QBETA,QCN,STH1,CTH1,XJX(3),XJV(3)

      COMMON/AGEN5/CHIBAS,CHPBAS,CHYBAS,CHISAV,TXX,TYY,TZZ,AMX,AMY,AMZ,
     *TMZ,THMFA,SIDE,WMAG,AZW,FANC,FLATC,FOM,VISC

      COMMON/ASTOR/ASTOR(4,67)

      COMMON/AUTO/KDB(40),SAVCP(40),XLAMB(40,20),DP2

      COMMON/BGEN3/NOSS,JTB,NENT,NACT,NOM1,NP(7)

      COMMON/BODY/ATBDWT(15,2),ATXCG(15,2),ATYCG(15,2),AXLEN(2),AXREF(2)
     *,AYREF(2),KPI(2),KYI(2),API(10,2),AYI(10,2)

      COMMON/BODYF/TBDWT(15),TXCG(15),TYCG(15),XLEN,XREF,YREF,M1,M2,M5,
     *M6,M7,M8,M9,M50,M51,MSRW,AP(10),AY(10)

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM

      COMMON/CONTRO/TTBL(30),CPTBL(30),CYTBL(30),TOBL(210),
```



```
     *CPOTBL(210),CYOTBL(210)

      common/disp/key,tv15,ivar,tdisp,dispstep(2),irec

      COMMON/FORINT/HBANK(2),T,TT,W,U,V,X,Y,Z,XMI,ZAP(18),DVAR(25),FSAVE
     *(625),NTRG1,TV1,NTRG2,TV2,NTRG3,TV3,NTRG4,TV4,NTRG5,TV5,NTRG6,TV6,
     *NTRG7,TV7,NTRG8,TV8,NTRG9,TV9,NTRG10,TV10,NTRG11,TV11,NTRG12,TV12,
     *NTRG13,TV13,KIND,KRDER,EU,EL,AYL,HMX,HMN,HNM

      COMMON/JJUMP/JJUMP

      COMMON/HEADUP/ HDMACH,HDCHRD

      COMMON/HEAT/MAXHAT

      COMMON/IPR/IPR,WPMX,ILAST,KDT(15),PRK4,PRTOT,iprop(6),prk5(5)

      COMMON/ITHRO/ITHRO

      COMMON/JUMPST/JUMPST,TSAVET,WZERST

      common/kp/kp,ky

      COMMON/LOADS/QLOAD(26),QTIME(26),QVALUE(13),SAVE(20,13)

      COMMON/LBM/TIMLBM(30),THRLBM(30),WDTLBM(30),PLBM(2),MLBM1,MLBM2,
     *ALBM(2),BLBM(2),CLBM(2),VLBM(2)

      COMMON/MISSION/MISSION(2)

      COMMON/MPS/TIMMPS(15),FTBL(15),THROT(15),GLIM(15),TAUALT(5),
     *TAUTBL(5),MPS1,MPS2,TGLIM(10,15),GLIMIT,tau_const(10,5),jthrot(15)

      COMMON/MLAST/MLAST,MMMM,KKKK

      common/mult/nchiot(6),tbias(6),nstage,nstg(15),wf(5),ibranch

      COMMON/NALTW/NALTW

      COMMON/NTILT/NTILT

      COMMON/OLOW/OLOW,WINT

      COMMON/QMAX/QMAX,QFLG,QPEN,MAXQ

      COMMON/REST/JSTR,NCNRS(5),NTCN,TR(5),LSB,NF8,NOPAR,WIBT(15)

      COMMON/RICONT/TTABLE(20),SIGMA,SIGTAB(20),M15,M16

      common/save_mass/save_mass

      COMMON/SRM/SRMTTB(600),SRMFTB(600),SRMWDT(600),SRMAE(600),MSRB

      COMMON/STAUT/STAUT(15)

      common/system/system
```

```
      COMMON/TAULIM/TAULIM,NTAU,NBETA,TIMTAU
      common/time_jump/time_jump(5),njump
      COMMON /TRIGC / NTRG14,TV14
      common/t_liftoff/t_liftoff
      common/viv_sav/viv_sav(26)
      dimension disp_var(25)
c*********************************************************************
c       external statement is needed since subroutine names are included
c       in the call to desolv
c*********************************************************************
      EXTERNAL AXPRT,ATILT,AQMAX,GLIMT,AWDEV,AEOSR,ALIFT,ATHRO,AROLL,
     *ader,ader1,ajump,dispprt
c*********************************************************************
c       initialize parameters
c*********************************************************************
C      MAXHAT=0
      QPEN=0.
      IWD=0
      KCYTAB=1
      CHIBAS=0.
      CHPBAS=0.
      CHYBAS=0.
      NTILT=0                                                    AFOR  66
      M15=1
      M16=1
      save_mass=0.

      mision=mission(1)

      if(jump.eq.1) nstage=1

      do i=1,26

         if(i.le.13) then
            qload(i)=100000000000.
         else
            qload(i)=-100000000000.
         endif

         qtime(i)=0.

      enddo
```



```
      KJUM=0                                                     AFOR  83
c*********************************************************************
C     CALCULATE A MATRIX (conversion from plumbline to equatorial)
c*********************************************************************
      SA=SIN(AA)                                                 AFOR  68
      CA=COS(AA)                                                 AFOR  69

      A(1,1)=SA                                                  AFOR  70
      A(2,1)=0.                                                  AFOR  71
      A(3,1)=CA                                                  AFOR  72

      A(1,2)=CA*STH1                                             AFOR  73
      A(2,2)=CTH1                                                AFOR  74
      A(3,2)=-SA*STH1                                            AFOR  75

      A(1,3)=-CA*CTH1                                            AFOR  76
      A(2,3)=STH1                                                AFOR  77
      A(3,3)=SA*CTH1                                             AFOR  78
c*********************************************************************
c       Get here for special internal jump start when running
c       cases that require no first stage simulation during
c       the iteration cycle
c*********************************************************************
      IF(JJUMP.ne.0.and.NMAX.ne.0.and.JUMPST.ne.0.and.LSB.ne.1) then AFOR  84

         JUMPS=JUMP                                              AFOR  85
         JUMP=JUMPST                                             AFOR  86
         KS2=KS1                                                 AFOR  87
         KS1=2                                                   AFOR  88
         WZEROS=WZERO                                            AFOR  89
         TZEROS=TZERO
         t_liftoffs=t_liftoff
         TZERO=TSAVET                                            AFOR  91
         TLIFTS=TLIFT                                            AFOR  92
         TLIFT=TLIFT+TZERO                                       AFOR  93
         WZERO=WZERST                                            AFOR  94
         QNOM=QMAX                                               AFOR  95
         BLOWS=BLOW                                              AFOR  96
         OLOWS=OLOW                                              AFOR  97

         CHIP=CPOTBL(1)
         CHIY=CYOTBL(1)

         KJUM=1                                                  AFOR 100

      endif
```

```
c*********************************************************************

c       Define earth spin axis vector(north direction)

c*********************************************************************

      A12W=A(1,2)*OMEGA                                                 AFOR 103
      A22W=A(2,2)*OMEGA                                                 AFOR 104
      A32W=A(3,2)*OMEGA                                                 AFOR 105

c*********************************************************************

c       Either calculate the initial state parameters or use the input
c       values (viv)

c*********************************************************************

      IF((JUMP.eq.1..or.viv(7).ge.1.).and.tzero.lt..1) then

         XZ=A(1,1)*XJX(1)+A(1,2)*XJX(2)+A(1,3)*XJX(3)
         YZ=              A(2,2)*XJX(2)+A(2,3)*XJX(3)
         ZZ=A(3,1)*XJX(1)+A(3,2)*XJX(2)+A(3,3)*XJX(3)

         WZ=A(1,1)*XJV(1)+A(1,2)*XJV(2)+A(1,3)*XJV(3)
         UZ=              A(2,2)*XJV(2)+A(2,3)*XJV(3)
         VZ=A(3,1)*XJV(1)+A(3,2)*XJV(2)+A(3,3)*XJV(3)

         t_liftoff=tzero

      else if(jump.eq.1.and.tzero.gt..1) then

         w=viv_sav(1)
         u=viv_sav(2)
         v=viv_sav(3)
         x=viv_sav(4)
         y=viv_sav(5)
         z=viv_sav(6)

         do i=1,18
            zap(i)=viv_sav(i+7)
         enddo

         chip=viv_sav(26)
         schip=sin(chip)
         cchip=cos(chip)

      else

         W=VIV(1)
         U=VIV(2)
         V=VIV(3)
         X=VIV(4)
         Y=VIV(5)
         Z=VIV(6)

         if(jump.gt.1) t_liftoff=tzero
```



```
         do i=1,18
            zap(i)=viv(i+7)
         enddo
      endif

c*********************************************************************

c       Set variable dimensioned array to be used in the integration
c       subroutines

c*********************************************************************

      LLDSOL=5                                                          AFOR 126
      IF(KIND.NE.2) LLDSOL=3*(KRDER+2)+4                                AFOR 127

C **  KRDER CAN NOT BE GREATER THAN 5                                   AFOR 128

c*********************************************************************

C     INITIALIZE TABLE LOOK-UP INDICES                                  AFOR 129

c*********************************************************************

      MLAST=0                                                           AFOR 130
      MMMM=1                                                            AFOR 131
      KKKK=2                                                            AFOR 132

c*********************************************************************

c       ks1 controls whether the liftoff logic will be initiated
c       ks1=1, logic is used; ks1=2, logic is not used

c*********************************************************************

      KS1=1                                                             AFOR 134
      IF(JUMP.GT.1) KS1=2                                               AFOR 135

c*********************************************************************

      LINES=0                                                           AFOR 136
      TL=0.                                                             AFOR 137
      QDOT=.1                                                           AFOR 138
      IWD=0                                                             AFOR 139

      IF(JUMP.eq.1.and.tzero.lt.1.) then

         W=WZ
         U=UZ
         V=VZ
         X=XZ
         Y=YZ
         Z=ZZ

         do i=1,6
            zap(i)=prop(i)*ptkg
```

```
            zap(i+6)=0.
            zap(i+12)=0.
         enddo

         if(system.eq.'SHUTTLE') then
            ZAP(1)=ZAP(1)-(ASTOR(1,64)+ASTOR(1,65))*PTKG
         else if(system.eq.'STAGE15') then
            zap(1)=zap(1)-(astor(1,32)+astor(1,33))*ptkg
         else if(system.eq.'HLLV') then
            zap(1)=zap(1)-(astor(1,30)+astor(1,31))*ptkg
         endif

      else

         tq=tzero

      endif

c*********************************************************************

c       Calculate the total jettison weight(xmiad) over the entire
c       trajectory

c*********************************************************************

c       wint is initial mass at initiation

      if(wint.lt.1.) wint=wzero


C     TAUT IS DURATION TIME OF THRUST EVENTS


      xmaug=wjet(jump)

      time(1,jump)=t_liftoff

      do i=jump,nvnt
         time(1,i+1)=time(1,i)+taut(i)
         if(nchiot(nstg(i)).eq.0)  XMAUG=XMAUG+WJET(I)
      enddo

c*********************************************************************

c       Calculates the total amount of mass dropped(xmaud) (minh only)

c*********************************************************************

      IWDJ=0
      I=1

    2 J=NOWD(I)

      IF(J.lt.JUMP) then

         IF(I.GE.NWVNT) GO TO 5
```



```
         I=I+1

         GO TO 2

      endif

      IWDJ=I

      do i=iwdj,nwvnt
         j=nowd(i)
         time(2,i)=time(1,j)+tauw(i)
         xmaug=xmaug+wd(i)
      enddo

c*********************************************************************

c       Calculate initial mass(xm) and time(t)

c*********************************************************************

    5 XM=WZERO                                                        AFOR 174
      XMI=XM-XMAUG
      XMIAD=XMI                                                       AFOR 177

      T=TZERO                                                         AFOR 178

c*********************************************************************

c       Set option flags and define initial roll attitude angle for
c       initial roll maneuver based on user input

c*********************************************************************

      NROLL=0
      IF(ABS(CHRDOT).LT.1.E-04) NROLL=-1

      IF(NROLL.EQ.-1.AND.IHEAD.EQ.0) FAZ=AA+PI                        AFOR 179
      IF(NROLL.EQ.-1.AND.IHEAD.NE.0) FAZ=AA                           AFOR 180
      IF(FAZ.GT.2.*PI) FAZ=FAZ-2.*PI                                  AFOR 181

      CHIRO=PI-FAZ+AA                                                 AFOR 182
      IF(CHIRO.GE.2.*PI) CHIRO=CHIRO-2.*PI                            AFOR 183

      IF(NROLL.eq.0) then

         CHIRF=0.
         IF(IHEAD.NE.0) CHIRF=PI

         N=0
         TBEGR=TCHIR
         TENDR=0.
         TENDO=TENDR-TBEGR

   19    IF(CHIRO.le.CHIRF) then
            IF(CHRDOT.LT.0.) TENDR=(-2.*PI+CHIRF-CHIRO)/CHRDOT
```

```
            IF(CHRDOT.GT.0.) TENDR=(CHIRF-CHIRO)/CHRDOT
         else
            IF(CHRDOT.LT.0.) TENDR=(CHIRF-CHIRO)/CHRDOT
            IF(CHRDOT.GT.0.) TENDR=(2.*PI+CHIRF-CHIRO)/CHRDOT
         endif

         IF(TENDO.gt.0..and.TENDR.ge.TENDO+PI/(2.*ABS(CHRDOT))) then
            N=N+1
            CHRDOT=-CHRDOT

            IF(N.EQ.2) STOP

            GO TO 19
         endif

         TENDR=TENDR+TBEGR

      endif

      CHIR=CHIRO                                                        AFOR 203

c*****************************************************************

c       initialize coefficients for pitch and yaw attitude angle
c       calculations in chipol subroutine

c*****************************************************************

      do i=1,10
         ap(i)=0.
         ay(i)=0.
         do j=1,2
            api(i,j)=0.
            ayi(i,j)=0.
         enddo
      enddo

C*****************************************************************

C       forward integration triggers

c       number  description                              routine parameter
c                                                        called
c       *****   ****************************         ******* ********
c       1       trajectory printout                      axprt   time
c       2       thrust event                             athrev  time
c       3       first jump start option                  ajump   time
c       4       tiltover and roll maneuver events        atilt   time
c       5       not used                                 -----   ----
c       6       maximum dynamic pressure                 aqmax   qdot
c       7       liftoff on thrust-to-mass = 1            alift   fom
c       9       acceleration limit                       glimt   fom/time
c       12      drop weight                              awdev   time
c       13      mps throttle                             athro   time
c       14      roll maneuver initiation                 aroll   mach
c       15      dispersion analysis printout on file     dispprt time
```



```
c*****************************************************************

      notrg=10
      if(ivar.ne.0) notrg=11

C*****************************************************************

C *1*   PRINT TRIGGER

c*****************************************************************

      NTRG1=-1                                                          AFOR 217

      IF(NMAX.EQ.0) NTRG1=1                                             AFOR 218

      TV1=AINT(T)+PRINT(JUMP)                                           AFOR 219

c*****************************************************************

c *3*   First stage jump start times

c*****************************************************************

      ntrg3=-2

      if(nmax.eq.0.and.time_jump(1).gt.0..and.tzero.lt..1) ntrg3=2

      tv3=time_jump(1)

      njump=1

C*****************************************************************

C *4*   TILT TRIGGER

c*****************************************************************

      NTRG4=2                                                           AFOR 222

      IF(JUMP.GT.1) NTRG4=-2                                            AFOR 223

c       The following logic has been implemented to account for a
c       first stage jump start option

      if(ntrg4.gt.0) then

         tv4=0.
         if(tlift.ge.tzero) tv4=tlift
         if(nroll.eq.0.and.tbegr.le.tv4.and.tbegr.ge.tzero) tv4=tbegr
         if(ifact.eq.3.and.tlift.ge.tzero) tv4=tlift

         if(tv4.lt..001) then

            if(nroll.eq.0.and.tendr.ge.tzero) tv4=tendr
            if(ttilt.lt.tendr.and.tlift.ge.tzero) tv4=ttilt
```

```
            if(tzero.ge.tendr) then
               nroll=-1
               if(tzero.lt.ttilt) tv4=ttilt
            endif
            if(tv4.lt..001) ntrg4=-2

         endif

         if(tzero.ge.tbegr.and.nroll.eq.0.and.ntrg4.gt.0) nroll=1

         if(tzero.gt.tlift.and.tzero.lt.ttilt) then

            ks1=2

            cptbl(1)=chpbas*rad
            cytbl(1)=chybas*rad
            kp=kpi(1)
            ky=kyi(1)

            call find(cptbl,ap,ttbl,kp)
            call find(cytbl,ay,ttbl(16),ky)

            do i=1,10
               api(i,1)=ap(i)
               ayi(i,1)=ay(i)
            enddo

         endif

         if(tzero.ge.ttilt) then
            ntilt=1
            ks1=2
         endif

      endif

C*********************************************************************

C *6*   MAX Q TRIGGER

C*********************************************************************

      NTRG6=-1

      if(jump.eq.1.and.tzero.gt.tlift) ntrg6=1

      TV6=0.

C*********************************************************************

c *9*   G limit trigger

C*********************************************************************

      ntrg9=-2
```



```
      tv9=0.

C*********************************************************************

C*12*   WEIGHT DROP EVENT TRIGGER (required in minh)

C*********************************************************************

      NTRG12=2

      IF(IWDJ.LE.0) NTRG12=-2

      TV12=TIME(2,IWDJ)

C*********************************************************************

C*13*   SSME THROTTLE TABLE TRIGGER

C*********************************************************************

      NTRG13=-2

      IF(NTAU.ne.0) then

         FTBL(NTAU+2)=FTBL(NTAU+1)

         if(taulim.gt.0.) then
            TIMMPS(NTAU+1)=TIMMPS(NTAU)+(FTBL(NTAU)-FTBL(NTAU+1))/
     *      TAULIM
         else
            timmps(ntau+1)=timmps(ntau)
         endif

         TIMMPS(NTAU+2)=TIMMPS(NTAU+1)+50.
         TIMMPS(NTAU+3)=TIMMPS(NTAU+2)+4.

      endif

C*********************************************************************

C *14*  HEADS-UP TO HEADS-DOWN ROLL TRIGGER

C*********************************************************************

      NTRG14=-1

      IF(IHEAD.EQ.0) THEN
         NTRG14=1
         TV14=HDMACH
      ENDIF

C*********************************************************************

c *15*  special output for dispersion analysis runs

C*********************************************************************
```

```
      ntrg15=-1
      if(ivar.ne.0) ntrg15=1
C*********************************************************************
      GLIMIT=.FALSE.
      QFLG=.FALSE.
C*********************************************************************
C       nf8 is the number of differential equations to be integrated
C*********************************************************************
      NF8=14
      IF(IPR.NE.0) NF8=17
      if(iprop(1)+iprop(2)+iprop(3)+iprop(4)+iprop(5).gt.0) nf8=20
      IF(NMAX.EQ.0) NF8=22
C*********************************************************************
C     Initialize thrust, step, mdot, etc. by calling subroutine athrev
C*********************************************************************
      ITHR=JUMP-1
      NTRG7=-2
      CALL ATHREV
      if(jump.eq.1.and.tv2.le.tlift.and.minh.eq.ithr+1) then
         ntrg4=-2
      endif
C*********************************************************************
C       If a dispersion run is being made, initialize print routine
C       dispprt
C*********************************************************************
      if(ivar.ne.0) then
         key=0
         tv15=0.
         call dispprt
      endif
C*********************************************************************
C       Simulate trajectory prior to minh phase
```



```
C*********************************************************************
      IF(ITHR.ne.MINH.and.JUMP.le.NOEVNT(1)) then
C                                                                  AFOR 259
         CALL ADER1
         CALL ADER
C*********************************************************************
C       if thrust-to-weight is less than 1 turn on liftoff trigger
C*********************************************************************
         IF(FOM.lt.1.) then
            TV7=1.
            NTRG7=2
            CALL APRTN(3)
         else
            CALL APRTN(2)
         endif
         IF(JUMP.NE.1) NTRG7=-2
C*********************************************************************
C       Subroutine desolv is called with different triggers depending
C       on the mode of operation.  In normal mode the value of lsb
C       would be not equal to 1 and the first call would be used.
C       The second call is used when runs associated with pre-minh
C       (first stage) parameters are being simulated to define
C       partial derivatives.
C*********************************************************************
         IF(LSB.ne.1) then
            LOAD=6
            IF(NMAX.EQ.0.AND.JUMP.EQ.1) LOAD=2
            CALL DESOLV(25,NF8,ADER1,ADER,AEOSR,LOAD,KERR,HNM,HBANK,
     *      FSAVE,KIND,LLDSOL,EU,EL,HMX,HMN,AYL,NOTRG,
     *      NTRG1,AXPRT,T,TV1, NTRG2,ATHREV,T,TV2, NTRG4,ATILT,T,TV4,
     *      NTRG6,AQMAX,QDOT,TV6, NTRG12,AWDEV,T,TV12,
     *      NTRG7,ALIFT,FOM,TV7, NTRG13,ATHRO,T,TV13,
     *      NTRG14,AROLL,XMACH,TV14, ntrg9,GLIMT,t,tv9,
     *      ntrg3,ajump,t,tv3, ntrg15,dispprt,t,tv15)
C
         else
            CALL DESOLV(25,NF8,ADER1,ADER,0,6,KERR,HNM,HBANK,FSAVE,
```

```
     *       KIND,LLDSOL,EU,EL,HMX,HMN,AYL,10,
     *       NTRG2,ATHREV,T,TV2,  NTRG4,ATILT,T,TV4,
     *       NTRG6,AQMAX,QDOT,TV6, NTRG12,AWDEV,T,TV12,
     *       NTRG7,ALIFT,FOM,TV7, NTRG13,ATHRO,T,TV13,
     *       NTRG14,AROLL,XMACH,TV14, ntrg9,GLIMT,t,tv9,
     *       ntrg3,ajump,t,tv3, ntrg15,dispprt,t,tv15, 0,0,0,0)

         endif

C*****************************************************************

c       a kerr not equal to zero indicates an integration error,
c       the simulation is stopped if this occurs

c*****************************************************************

         IF(KERR.NE.0) then

            WRITE(JO,12)

            stop

         endif

      endif

c*****************************************************************

c       If partial derivative runs are being made on all parameters
c       except launch azimuth, only the first stage (pre-minh)
c       portion of the simulation is run.

c*****************************************************************

      IF(LSB.EQ.1.AND.NALTW.EQ.0) RETURN

c*****************************************************************

c       If jump start begins after the first stage (noevnt(1)),
c       attitude table flag must be adjusted.

c*****************************************************************

      IF(JUMP.gt.NOEVNT(1)) then
         DO I=1,7
            IF(NP(I).ne.0) then
               IF(JUMP.GE.NENDCT(I)) KCYTAB=I+1
               IF(KCYTAB.GT.7) KCYTAB=7
            endif
         enddo
      endif

c*****************************************************************

c       Integration logic for upper stage (begin minh phase)
```



```
c*****************************************************************

10    IF(LSB.ne.1) then
C
         IXR=0

         NOTRG=5
         if(ivar.ne.0) notrg=6

         ntrg6=-1
         if(ithr.lt.noevnt(1).and.t.le.70.) ntrg6=1

         CALL DESOLV(25,NF8,ADER1,ADER,AEOSR,2,KERR,HNM,HBANK,
     *   FSAVE,KIND,LLDSOL,EU,EL,HMX,HMN,AYL,NOTRG,
     *   NTRG1,AXPRT,T,TV1, NTRG2,ATHREV,T,TV2, NTRG9,GLIMT,T,TV9,
     *   NTRG12,AWDEV,T,TV12, ntrg6,aqmax,qdot,tv6,
     *   ntrg15,dispprt,t,tv15,   0,0,0,0,    0,0,0,0,  0,0,0,0,
     *   0,0,0,0,   0,0,0,0)

c*****************************************************************

C       Special logic for coast phase following engine shutdown

c*****************************************************************

         IF(NMAX.eq.0.and.ITHR.eq.NCOAST) then

            CALL ADER1

            CALL ADER

            CALL APRTN(6)

            NVNT=NVNT+1

            GO TO 10

         endif

c*****************************************************************

c       Logic to reset jump start for internal jump start cases

c*****************************************************************

         IF(KJUM.ne.0) then
            JUMP=JUMPS
            KS1=KS2
            WZERO=WZEROS
            TZERO=TZEROS
            TLIFT=TLIFTS
            t_liftoff=t_liftoffs
            BLOW=BLOWS
            OLOW=OLOWS
         endif
```

```
      else

         ntrg6=-1
         if(ithr.lt.noevnt(1).and.t.le.70.) ntrg6=1

         CALL DESOLV(25,NF8,ADER1,ADER,0,6,KERR,HNM,HBANK,FSAVE,
     *   KIND,LLDSOL,EU,EL,HMX,HMN,AYL,5,
     *   NTRG2,ATHREV,T,TV2, NTRG9,GLIMT,T,TV9, NTRG12,AWDEV,T,TV12,
     *   ntrg6,aqmax,qdot,tv6, ntrg15,dispprt,t,tv15,   0,0,0,0,
     *   0,0,0,0,   0,0,0,0,  0,0,0,0,  0,0,0,0,   0,0,0,0)

      endif

c**********************************************************************

c       If a dispersion run is being made (ivar ne 0), then write last
c       record on trajectory file by marking with a 9999

c**********************************************************************

      if(ivar.ne.0) then
         irec=irec+1
         lvar=9999

         write(80,rec=irec) disp_var,lvar

      endif

c**********************************************************************

c       a kerr not equal to zero indicates an integration error,
c       the simulation is stopped if this occurs

c**********************************************************************

      IF(KERR.eq.0) return

      WRITE(JO,12)
   12 FORMAT(//1X,25HINTEGRATION ERROR. DUMPED)                    AFOR 284

      STOP                                                         AFOR 285
      END                                                          AFOR 286
```

3.10 Subroutine AGEO

3.10.1 Purpose

The purpose of this subroutine is to calculate the gravitational parameters for forward and backward trajectories. Gravitational acceleration is calculated for both forward and backward trajectories and partial derivatives of the gravitational acceleration components with respect to the state variables are calculated for the backward trajectory.

3.10.2 Variable Listing

3.10.3 Subroutines Called:

None

3.10.4 Calling Subroutines:

Name	Function
ADER	Calculates the time independent portion of the equations of motion during forward integration
BDR1I	Calculates equations of motion for backward integration

3.10.5 Fortran Listing

ageo

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
cj	J harmonic for earth model		ainit	input	Global	const
cjr	Temporary variable				Local	
cmt	Temporary variable				Local	
cmue	Gravitiational constant	m^3/se	ainit	input	Global	const
ct2	Squared value of cth		ader	input	Global	agen2
cth	Cosine of colatitude		ader	input	Global	agen2
dj	Second harmonic for earth model			input	Global	const
djr	Temporary variable				Local	
g11	Temporary variable in gravitational equations		ageo	output	Global	grav
g22	Temporary variable in gravitational partial derivative equations		ageo	output	Global	grav
g23	Temporary variable in gravitational partial derivative equations		ageo	output	Global	grav
g33	Temporary variable in gravitational partial derivative equations		ageo	output	Global	grav
gto	Temporary variable in gravitational equation	m/sec^	ageo	output	Global	grav
h	Third gravitational harmonic		ainit	input	Global	const
hr	Temporary variable				Local	
n	Flag indicating forward or backward trajectory being simulated (=1, forward; =2, backward)				Local	
r	Radius from earth's center	m	ader	input	Global	agen2
r2	Radius squared	m^2			Local	
re	Earth's radius	m	ainit	input	Global	const
rer	Earth radius divided by radius of vehicle				Local	
rer2	Squared value of rer				Local	

```
      SUBROUTINE AGEO(N)                                                AGEO   1
C*********************************************************************
C       Calculates the gravitation parameters for forward and backward
C       trajectories
C          N      =     1       forward trajectory (accelerations terms)
c                       2       backward trajectory (accelerations and
c                               partial derivative terms)
C*********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      COMMON/AGEN2/AA,SA,CA,ALF,ALFY,CHIP,SCHIP,CCHIP,CHIY,SCHIY,CCHIY,
     *CHIR,SCHIR,CCHIR,STH,CTH,STHL,CTHL,DPHIZ,RTHE,R,VR,XM,SW,SU,SV,Q,
     *VIV(25),DVARS(13),DELXDW(15,3),DELXDR(7,5),DELXD(15,7),WZERO,ALT,
     *XMACH,QDOT,FAA,FAN,A(3,3),CASE,CT2,UMF,A12W,A22W,A32W,XJEXT,BEU,
     *BYL,BEL,BHMAX(15),BHMN,BSTEP(15),HNMB,HMXB,TPOLY,XMIAD

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM

      COMMON/GRAV/GTO,G11,G22,G23,G33
C*********************************************************************
C     NEEDS CTH,CT2,     RE,R,CMUE,J,H,D,R2                             AGEO  12
C*********************************************************************
C     GET OBLATE EARTH GRAVITY TERMS                                    AGEO  14
C*********************************************************************
      R2=R*R
      RER=RE/R                                                          AGEO  15
      RER2=RER*RER                                                      AGEO  16

      CJR=CJ*RER2                                                       AGEO  17
      HR=H*RER2*RER                                                     AGEO  18
      DJR=DJ*RER2*RER2                                                  AGEO  19
      CMT=CMUE/(R2*R)                                                   AGEO  20

      G11=-CMT*(1.+CJR*(1.-5.*CT2)+HR*(3.-7.*CT2)*CTH                   AGEO  21
     *    +DJR*(.42857143-(6.-9.*CT2)*CT2))                             AGEO  22

      IF(N.eq.1) then                                                   AGEO  23

         GTO=R*CMT*(2.*CJR*CTH-HR*(.6-3.*CT2)+DJR*(1.7142857-4.*CT2)*
     *       CTH)

         RETURN

      endif
```



```
      G22=-(3./R2)*(G11-CMT*(CJR*(.66666667-6.66666667*CT2)             AGEO  26
     *      +HR*(4.-14.*CT2)*CTH+DJR*(.57142857-(12.-24.*CT2)*CT2)))    AGEO  27

      G23=(CMT/R)*(10.*CJR*CTH-HR*(3.-21.*CT2)+DJR*(12.-36.*CT2)*CTH)   AGEO  28

      G33=-CMT*     (2.*CJR+6. *HR*CTH          +DJR*(1.7142857-12.*CT2)) AGEO  29

      RETURN                                                            AGEO  30
      END                                                               AGEO  31
```

3.11 Subroutine AINIT

3.11.1 Purpose

The purpose of this subroutine is to read the input namelist files and initialize program parameters.

3.11.2 Variable Listing

3.11.3 Subroutines Called:

Name	Function
AEBANK	Retrieves nonlinear aerodynamic data from database
ATMOSPHERE	Patrick and Vandenburg atmospheric routines
DBANK	Retrieves tabular data from database
FUNISP	Calculates specific impulse based on throttle level
IINRC	Retrieves integer data from database
MASSPRO	Calculate dynamic mass properties
RRAINT	Initializes range reference atmospheric model
RRASPL	Range reference atmospheric model
SIMUL	Calculates coefficients for specific impulse versus throttle level 2nd order polynomial
WINDIN	Reads wind data from database

3.11.4 Calling Subroutines:

Name	Function
MENU	Controls operation of program

3.11.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
aa	Launch azimuth	rad	ainit	output	Global	agen2
aerob(50,6,3	Base force tables		ainit	output	Global	aerodi
aerod(30,17)	Aerodynamic coefficient tables		ainit	input	Global	aerodi
alat	Launch geodetic latitude	deg	ainit	input	Global	alat
along	Launch longitude	rad	ainit	output	Global	alat
alt	Altitude	m	ader	output	Global	agen2
alt1	First discrete altitude for printout (10km) (not used)	m	ainit	output	Global	const
alt2	Second discrete altitude for printout (14km) (not used)	m	ainit	output	Global	const
altbas	Altitude table for base force table	m			Local	
altls	Altitude of launch site	m	ainit	output	Global	alat
answer	Character variable defining user answer to questions asked during execution				Local	
astor(4,67)	Storage array for weight output table	lbs	ainit	output	Global	astor
ayl	Lower dependent variable value for auto step size logic	sec		output	Global	forint
az	Wind azimuth table	rad	ainit	output input	Global	initp
aztbl(100)	Wind azimuth table	rad	ainit	output	Global	wind
azwtbl(100)	Wind azimuth table	rad	ainit	input	Global	initp
baxial	Base axial force table				Local	
bcd(22)	Temporary variable		ainit	input	Global	initp
bel	Lower error limit for backwards integration	sec		output	Global	agen2
beu	Upper limit for backwards integration	sec		output	Global	agen2
bhmax(15)	Maximum stepsize for backwards integration	sec		output	Global	agen2
bhmn	Minimum stepsize for backwards integration	sec		output	Global	agen2
bnorm	Normal base force table				Local	
bo(3,2)	Direction cosines of position and velocity vectors at launch			output input	Global	agen1
bpit	Base force pitching moment table	nt			Local	
bstep(15)	Backward integration stepsize	sec		output	Global	agen2
byl	Lower dependent variable value for auto stepsize	sec		output	Global	agen2

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
caa(15,11,6)	Axial force coefficient		ainit	output	Global	aero
caalp	Slope of axial force coefficient with respect to angle of attack table				Local	
cao	Zero angle of attack axial coefficient table				Local	
case	Case number		ainit	output input	Global	agen2
cbaxil	Constant value of base axial force		ainit	output	Global	agen1
cdphi	Cosine of launch longitude bias				Local	
char_vel(5)	Characteristic velocity	m/sec	ainit	input	Global	multi
chip	Pitch attitude angle	rad	ader1	output input	Global	agen2
chiy	Yaw attitude angle	rad	ader1	output input	Global	agen2
chpbas	Pitch attitude bias angle	rad	ader1	output	Global	agen5
chrd	Roll attitude rate	rad/sec	ainit	output	Global	agen1
chrdhd	Roll rate upper stage roll maneuver	deg/sec			Local	
chrdot	Roll attitude rate	rad/sec			Local	
chybas	Yaw attitude bias angle	rad	ainit	output	Global	agen5
cj	J harmonic for earth model		ainit	output	Global	const
cla(15,11,6)	Yaw force coefficient		ainit	output	Global	aero
clbeta	Rolling moment slope coefficient	/deg			Local	
clo	Zero sideslip angle roll moment cefficient table				Local	
cma(15,11,6)	Pitching moment coefficient			output	Global	aero
cmalp	Slope of pitching moment coefficient with respect to angle of attack table	/deg			Local	
cmo	Zero angle of attack pitching moment coefficient table				Local	
cmue	Gravitiational constant	m^3/se	ainit	output	Global	const
cna(15,11,6)	Normal force coefficient		ainit	output	Global	aero
cnalp	Slope of normal force coefficient with rspect to angle of attack table	/deg			Local	
cnbeta	Slope of yawing moment coefficient with respecty to sideslip angle table	/deg			Local	
cnbo	Zero sideslip angle yawing moment coefficient table				Local	
cno	Zero angle of attack normal force				Local	

Variable	Variable Definition coefficient table	Units	Source	I/O	Status	Common Block
coment	Character string for comments in weights namelist				Local	
cpotbl(210)	Pitch attitude table used during min-H phases	rad	ainit	output	Global	contro
cp_input	Logical indicating center of pressure input		ainit	output	Global	cp_input
croll	Cosine of adjusted roll angle				Local	
cth	Cosine of colatitude			output input	Global	agen2
cth1	Cosine of launch colatitude		ainit	output	Global	agen4
cxa(15,11,6)	Axial force coefficient		ainit	output	Global	aero
cya(15,11,6)	Side force coefficient		ainit	output	Global	aero
cybeta	Slope of yaw force coefficient with respect to sideslip angle table	/deg			Local	
cyo	Zero sideslip angle value of side force coefficent				Local	
cyotbl(210)	Yaw attitude table used during min-H phases	rad	ainit	output	Global	contro
dalt(25)	Independent altitude table for base force table dfab	m		output	Global	delfab
dbname	Name of aerodynamic data base				Local	
delalt	Independent altitude table for incremental base force table				Local	
delfab	Incremental base force table				Local	
delvg	Delta velocity for geometry reserves	m/sec		output	Global	delvp
demand	Logical indicating the operational mode		initial	input	Global	ccc
dfab(25)	Base force table			output	Global	delfab
dispstep(2)	Increment steps used in output of dispersion trajectories	sec	ainit	output	Global	disp
dj	Second harmonic for earth model			output	Global	const
dp2	Constraint scale factor		ainit	output	Global	auto
dphiz	Longitudinal displacement due to GRR			output input	Global	agen2
dtz	Time difference between GRR and launch time	sec	ainit	output	Global	agen1
el	Lower limit on step size	sec	ainit	output	Global	forint
engdat(8,15)	Engine data matrix		ainit	output	Global	agen1

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
engines(15,1	This array indicates which engine from the ENGDAT array is being used for a certain thrust event. (0- not being used, 1-being used)		ainit	input	Global	engines
eu	Upper limit for stepsize control	sec	ainit	output	Global	forint
eventnum	Number of event				Local	
faz	Stored value of pointing azimuth of orbiter tail fin	rad			Local	
fazz	Stored value of pointing azimuth of orbiter tail fin	rad	ainit	output	Global	agen1
file_name	Genertic file name				Local	
flat	Earth flattening coefficient		ainit	output	Global	const
fpr_fac(5)	Factor for calculation of flight performance reserve (branch option)		ainit	output	Global	multi
ftm	Feet to meter conversion				Local	
glim(15)	Acceleration limit	g's	ainit	output	Global	mps
gzero	Gravitational constant (9.80665 m/sec)	m/sec^	ainit	output	Global	const
h	Third gravitational harmonic		ainit	output	Global	const
hbias	Altitude bias	m	ainit	input	Global	ardc
hdchrd	Heads-up to head-down roll rate	rad/sec	ainit	output	Global	headup
hdmach	Mach number lower limit for second roll maneuver		ainit	output	Global	headup
head	Storage array for headings in printout		ainit	output	Global	agen1
header(20)	Array defining constraint titles		ainit	output	Global	enput
hfuel(200)	LH2 height table	in		output	Global	tanks
hlox(100)	LOX height table	in		output	Global	tanks
hmax(15)	Maximum step size when using variable step integration routine	sec		output	Global	agen1
hmn	Minimum step size when using variable step integration routine	sec		output	Global	forint
ia	Index				Local	
iatmos	Atmospheric routine indicator		ainit	output	Global	ardc
ib	Index				Local	
iconsw	Flag to indicate termination of SRB moment balance		ainit	output	Global	agen3
ifact	Flag indicating the type of attitude used in the booster stage		atilt	output	Global	agen3
ihead	Roll attitude flag			output	Global	agen3

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ii	index				Local	
ij	Index				Local	
ij1	Index				Local	
ijk	Index				Local	
ik	Index				Local	
ik1	Index				Local	
ilast	Flag indicating the first thrust event in the last stage			output	Global	ipr
inmax	Temporary variable for nmax			output	Global	Iprint
ipoly	Flag indicating number of polynominals in booster stage		ainit	output	Global	agen3
ipr	Denotes the thrust event from which fuel propellant reserve (FPR) is calculated			output	Global	ipr
iprop(6)	Propulsion cutoff flags (stage dependent)		ainit	output	Global	ipr
istart(2)	Index used for jump start for branch trajectory		ainit	output	Global	istart
itol(15)	Flag indicating thrust event use of delta aerodynamic coefficients and base force		ainit	output	Global	delfab
ivar	Index used to denote the type of dispersion case		ainit	output	Global	disp
iwnum	Wind table type index		ainit	output	Global	iwnc
ja	Index indicating number of thrust events used in the simulation				Local	
jeng	Number of MPS engines				Local	
jetet	Thrust event indicator of jettison of external tank				Local	
jjump	Flag indicating that all runs between the first iteration and the converged iteration can be simulated by jump start at the beginning of the minh phase		ainit	output	Global	jjump
jstart(6)	Index for use with branch trajectory option		ainit	input	Global	istart
jump	Jump start flag		ainit	output	Global	agen3
jumpst	Saved value of jump		ainit	output	Global	jumpst
kcdphi(20)	Terminal constraint selection flag array		ainit	output	Global	yli
kcdres(6,5)	Array for intermediate constraint function code		ainit	output	Global	enput

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
kdb(40)	Array for control parameter flags		ainit	output	Global	auto
kdt(15)	Array used in confunction with the ipr option		ainit	output	Global	ipr
kind	Integration type flag for forward trajectory		ainit	output	Global	forint
kindb	Integration type flag for backwards trajectory		ainit	output	Global	agen3
kk	Index				Local	
kp(2)	Order of booster pitch attitude polynomial		ainit	output	Global	body
kpage	Page number index		ainit	output	Global	agen3
kprt	Index				Local	
krder	Order of differences for forward integration		ainit	output	Global	forint
krderb	Order of differences for backwards integration		ainit	output	Global	agen3
kv	Index for terminal and intermediate constraints				Local	
kwta(7)	Flag indicating pitch only (1) or pitch and yaw (2) attitude control		ainit	output	Global	agen3
ky(2)	Order of booster yaw attitude polynomial		ainit	output	Global	body
last	Previous index used in atmospheric routine			output	Global	enput
latcg	Lateral center of gravity component	m			Local	
latcgp	Partial derivative of lateral CG with respect to mass				Local	
level	Flag specifying propulsion summary output required				Global	tanks
linear_aero	Flag to denote use of linear aerodynamic tables in the first stage		ainit	output	Global	linear_aero
lk	Index indicating which orbital velocity data to use as terminal or intermediate constraints			output input	Global	vic
loncgp	Partial derivative of total longitudinal CG with respect to mass				Local	
lprint	Print flag			output	Global	lprint
lsb	Internal flag to indicate partial derivative runs			output	Global	rest
lstge(15)	Aerodynamic coefficient flag to		ainit	output	Global	agen3

Variable	Variable Definition distinguish stages	Units	Source	I/O input	Status	Common Block
m	Index				Local	
minh	Flag to indicate thrust event which begins min-h attitude optimization		ainit	input	Global	agen3
mision	Character indicating name of mission			output	Global	agen3
mission(2)	Character indicating name of mission				Global	mission
misson(6)	Character indicating name of mission			input output	Global	initp
mombal	Flag to indicate moment balance calculations are desired		ainit	output	Global	mombal
mpflag	Alternate mass properties flag				Global	mpropin
mpname	Name of alternate mass properties input file			output	Global	mpropn m
mpsflg(15)	Flag indicating the type of MPS thrust simulation		ainit	output	Global	agen3
mpsinp(15)	Array used in conjunction with mpsflg=6 to denote which engines (with respect to the ENGDAT array) will use the throttle table		ainit	output	Global	mps1
mswch(15)	Flag indicating which thrust events are considered critical			output	Global	agen3
n	Index				Local	
namlst	Name of Namelist file				Local	
nbeta	Index used for booster sideslip attitude control		ainit	output	Global	taulim
ncnrs(5)	Number of constraints at restart		ainit	output	Global	rest
ncoast	Thrust event to begin coast on final case		ainit	output	Global	agen3
neng_out	Index defining which engine will considered out for a first jump start				Local	
nfixed	Number of fixed engines				Local	
nfuel	Number of points in LH2 height table			output input	Global	tanks
nlines	Index used for output page control				Local	
nlox	Number of points in LOX height table			output input	Global	tanks
nmax	Iteration counter		ainit	output	Global	agen3
nobase	Flag indicating that no base force will be calculated after the thrust event specified by nobase		ainit	output	Global	agen3

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
noevnt(5)	Number of thrust events per stage		ainit	output input	Global	agen3
nom1	Index indicating first iteration pass			output	Global	bgen3
nopar	Number of optimization parameters			output input	Global	rest
norcg	Total normal component of center of gravity	in			Local	
norcgp	Derivative of normal CG with respect to mass	in/lbs			Local	
norder	Temporary index array				Local	
noss	Number of constraints		ainit	output input	Global	bgen3
nowd(15)	Index indicating thrust event where weight drop event will occur		ainit	output	Global	agen3
np(7)	Number of points in attitude tables during min-H phases		ainit	output	Global	bgen3
nroll	Flag indicating booster roll program after launch		ainit	output	Global	agen3
nstage	Index number of current stage		athrev	output	Global	mult
nstg(15)	Internal index which relates thrust event to stage number		ainit	output	Global	mult
nsyst(15)	Index array which specifies the type of engines that will be used for each thrust event		ainit	output	Global	agen3
nta	Number of points in angle of attack table		ainit	output	Global	aero
ntable	Flag indicating that output tables will (>0) or will not (=0) be generated		ainit	output	Global	alat
ntalph(2)	Number of points in angle of attack table			input	Global	aero
ntau	Index used in conjunction with timmps to indicate a throttle event in the booster stage			output	Global	taulim
ntb	Number of tabular values in sideslip angle tables			input	Global	aero
ntbeta(2)	Number of tabular values in sideslip angle tables			input	Global	aero
ntcn	Number of intermediate constraints		ainit	output input	Global	rest
ntime	Case number identification used in subroutine ainit to reduce input requirements for multiple cases			output input	Global	ntime

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ntmach	Number of tabular values in mach number table		ainit	output	Global	aero
nvnt	Number of thrust events			output	Global	agen3
nvrst(5)	Array specifying the thrust events where intermediate constraints will be imposed		ainit	output	Global	enput
nwind	Number of tabular values in wind parameter tables		ainit	output	Global	wind
nwvnt	Number of weight drop events			output	Global	agen3
olow	Orbiter liftoff weight	kg	ainit	output	Global	olow
omega	Earth's spin rate	rad/sec		output	Global	const
orbase	Orbiter base force table	nt			Local	
out	Index defining which engine will considered out for a first jump start				Local	
pat(14)	Input/output array for atmospheric parameters			output input	Global	ardc
pfkg	Conversion for pounds to kilograms	kg/lbs	ainit	output input	Global	const
pfn	Conversion from pounds to newtons	lbs/nt	ainit	output input	Global	const
pi	Pi constant (3.14159265)		ainit	output	Global	const
plot	Logical variable used to indicate plot output file is to be generated			output	Global	plot
pnm	Independent mach number table used for aerodynamic coefficients				Local	
prk4	Temporary variable used in calculation of performance reserve option		ainit	output	Global	ipr
prk5(5)	Same as prk4 but stage dependent (used with branch trajectory option)		ainit	output	Global	ipr
prtot	Total propellant weight used for performance reserve option	kg	ainit	output	Global	ipr
psffm	Conversion for pounds per square feet from newtons per square meter		ainit	output	Global	const
psftm	Conversion for pounds per square feet to newtons per square meter		ainit	output	Global	const
psireq(20)	Desired values of terminal constraints	variabl	ainit	output	Global	yli
psirst(6,5)	Desired values of intermediate constraints	variabl	ainit	output	Global	enput
psl	Sea level pressure	newton	ainit	output	Global	const

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ptkg	Conversion from pounds to kilograms	kg/lbs	ainit	output	Global	const
ptn	Conversion from pounds to newtons	nt/lbs	ainit	output	Global	const
qalpha_max	Maximum value of q alpha		ainit	output input	Global	qalpha_ max
qreft	Reference dynamic pressure table	kg/m^2			Local	
qy	Optimization step size control constant		ainit	output	Global	lprint
r	Radius from earth's center	m	ainit	output input	Global	agen2
ratio	Propellant mixture ratio		ainit	output	Global	tcl
re	Earth's radius	m	ainit	output	Global	const
rho	Atmospheric density	kg/m^3			Local	
ri	Temporary radius value	m			Local	
rm	Temporary radius value	m			Local	
roll	Adjusted roll angle to compensate for engine out	rad			Local	
s(15)	Aerodynamic reference area for each thrust event	m^2	ainit	output	Global	agen1
sdphi	Sine of launch longitude bias				Local	
spradb	Increment in forward difference to evaluate partial derivative of payoff and constraints with respect to first stage pitch attitude variations	rad		output	Global	delpar
srbnam	Name of SRB input file				Local	
srmae(600)	SRB exit area table	m^2		output input	Global	srm
srmftb(600)	SRM vacuum thrust table	kg	ainit	output	Global	srm
srmttb(600)	SRM time table	sec	ainit	output	Global	srm
srmv1	Temporary storage for SRB time table variables (1st field of SRB input data file)				Local	
srmv2	Temporary storage for 2nd field of SRB input data file				Local	
srmv3	Temporary storage for 3rd field of SRB input data file				Local	
srmv4	Temporary storage for SRB vacuum thrust (4th field of SRB input data file)				Local	
srmv5	Temporary storage for SRB flowrate (5th field of SRB input data file)				Local	
srmv6	Temporary storage for SRB exit area				Local	

Variable	Variable Definition (6th field of SRB input data file)	Units	Source	I/O	Status	Common Block
srmwdt(600)	SRM flowrate table	lbs/sec	ainit	output	Global	srm
sroll	Sine of adjusted roll angle				Local	
startv(40,2)	Saved values of state for jump start			input	Global	istart
staut(15)	Stored value of TAUT	sec		output	Global	staut
step(15)	Integration step size	sec		output	Global	agen1
sth	Sine of colatitude			output input	Global	agen2
sth1	Sine of launch colatitude			output input	Global	agen4
sthl	Sine of launch colatitude			output	Global	agen2
svazre	Azimuth angle for RTLS mission phase	rad			Local	
svxinc	Saved value of inclination angle for AOA trajectory constraint for variable IY targeting	rad		input	Global	istart
svxnod	Saved value of descending nodal angle for AOA trajectory constraint for variable IY targeting			input	Global	istart
syradb	Increment in forward difference to evaluate partial derivative of payoff and constraints with respect to first stage yaw attitude variations	rad	ainit	output	Global	delpar
system	Name of the vehicle system being simulated (SHUTTLE, STAGE15, HLLV)		ainit	output	Global	system
talfp	Angle of attack attitude angle table	deg			Local	
talph(11,2)	Independent angle of attack tables for aerodynamic tables	deg	ainit	output	Global	aero
talyh(11,2)	Independent sideslip tables for aerodynamic tables	deg	ainit	output	Global	aero
taulim	Rate of change of throttle limit	/sec	ainit	output	Global	taulim
taumps(15)	Throttle level table for thrust tailoff option (mpsflg=6)		ainit	output	Global	mps1
taut(15)	Time increments for thrust events	sec	ainit	output	Global	agen1
tauw(15)	Time increment for weight drop event	sec	ainit	output	Global	agen1
tbegr	Time to begin roll maneuver	sec	aforun	output	Global	agen1
tbeta	Desired sideslip angle attitude angle table	deg			Local	
tblisp	Specific impulse table				Local	
tblpl	Power level table for specific impulse				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
tdisp	Time to begine the print increment from the 2nd array of the dispstep array times	sec	ainit	output	Global	disp
temp	Temporary variable				Local	
temp1	Temporary variable				Local	
tendr	Time to terminate roll maneuver	sec	ainit	output	Global	agen1
thet1	Colatitude of launch site	rad			Local	
time_jump(5)	Time of jump start	sec	ainit	output	Global	time_jump
tjetwt	Jettison weight table	lbs			Local	
tlift	Time to begin tiltover maneuver	sec	ainit	output	Global	agen1
tmach	Independent mach number table for angle of attack and sideslip tables (talfp and tbeta)				Local	
tne(6,15)	Array which defines the number of engines per thrust event		ainit	output	Global	agen1
tol	Convergence tolerance on scaled total derivatives		ainit	output	Global	lprint
tr(5)	Intermediate constraint absolute time	sec	ainit	output	Global	rest
ttt	Initial time from jump start file for first stage jump start option				Local	
txmac(28)	Independent mach number table for aerodynamic tables		ainit	output	Global	aero
tzero	Time of simulation initiation	sec	ainit	output	Global	agen1
t_liftoff	Time of lift-off	sec	ainit	output	Global	t_liftoff
umf	One minus the earth flattening coefficient		ainit	output input	Global	agen2
vic(2)	Circular velocity used to calculate oblate earth model velocity option (KCDRES and/or KCDPHI=14)			output	Global	vic
viv(25)	Initial condition array for jump start		ainit	output	Global	agen2
viv_sav(26)	Saved values of state variable for first stage restart		ainit	input	Global	viv_sav
wd(15)	Weight dropped during weight drop event	kg	ainit	output	Global	agen1
wdlbs(15)	MPS weight flowrate	lbs/sec	ainit	output	Global	initp
wfuel(200)	LH2 weight overboard tables	kg		output	Global	tanks
wh2l	LH2 mass overboard tables	kg	ainit	output	Global	tanks
wibt(15)	Dynamic parameter weight factor		ainit	output	Global	rest

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
wint	Initial vehicle weight	kg	ainit	output	Global	olow
wjet(15)	Jettison weight per thrust event	kg	ainit	output	Global	agen1
wlox(100)	LOX tank mass overboard table	kg	ainit	output	Global	tanks
wo1	Liftoff weight	lbs			Local	
woxl	LOX propellant weight	lbs	ainit	output	Global	tanks
wrz	Earth spin velocity at launch latitude	m/sec			Local	
wtjet(15)	Jettison weight per thrust event	lbs	ainit	output	Global	initp
wtnl	Name of weight namelist				Local	
wzero	Initial vehicle weight	lbs	ainit	output	Global	agen2
w_marg(2)	Guessed value of weight margin	lbs	ainit	input	Global	multi
w_margin	Weight margin input	lbs	ainit	input	Global	
x	X component of plumbline position vector	m	dpir	input	Global	forint
xcp	Center of pressure table	m			Local	
xisp	Specific impulse	sec			Local	
xjext	Flag indicating that payoff will be maximized (=1) or minimized (=-1)		ainit	output	Global	agen2
xjv(3)	Components of plumbline inertial velocity at launch	m/sec	ainit	output	Global	agen4
xjx(3)	Components of plumbline position vector at launch	m	ainit	output	Global	agen4
xk1(2)	Temporary variable use with calculation of oblate velocity option		ainit	output	Global	vic
xk2(2)	Temporary variable use with calculation of oblate velocity option		ainit	output	Global	vic
xk3(2)	Temporary variable use with calculation of oblate velocity option		ainit	output	Global	vic
xk4(2)	Temporary variable use with calculation of oblate velocity option		ainit	output	Global	vic
xlamb(40,20)	Partial derivatives of payoff and constraints with respect to the optimization parameters		ainit	output	Global	auto
xmass	Vehicle weight from jump start file	kg			Local	
xmaug	Auxiliary vehicle mass	kg	athrev	output	Global	agen1
ygp	Longitudinal component of gimbal point of engine considered out by using the neng_out variable for first stage jump start	m			Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ygpout	Longitudinal component of gimbal point of engine considered out by using the neng_out variable for first stage jump start	m			Local	
zgp	Normal component of gimbal point of engine considered out by using the neng_out variable for first stage jump start	m			Local	
zgpout	Normal component of gimbal point of engine considered out by using the neng_out variable for first stage jump start	m			Local	

```
      SUBROUTINE AINIT                                                  AINI   1
C*********************************************************************
C       READS INPUT DATA; PERFORMS INITIALIZATION
C*********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

        REAL*4 SRMV1,SRMV2,SRMV3,SRMV4,SRMV5,SRMV6
        real*8 loncg,latcg,norcg,loncgp,latcgp,norcgp
        real*8 moboos,momain,bomdot,mnmdot

        integer eventnum
        integer engines

        LOGICAL DEMAND,GRAPH,GLIMIT,QFLG,VISC,plot,linear_aero,
     *  cp_input,mombal,mpflag


        CHARACTER*24 DBNAME
        CHARACTER*80 COMENT
      CHARACTER*3 MNTH
      CHARACTER*6 MISSON,MISION,ATMOSN,mission
      character*12 bcd,header
      CHARACTER*30 SRBNAM,NAMLST
C     CHARACTER*36 IAR
      CHARACTER*60 HEAD,DN2
      CHARACTER*24 WTNL
      character*30 file_name
      character*1 answer
      character*7 system
        character*30 mpname
C
C       COMMON/VMR/VMRINT,VMRFIN,VMRSWH
C
      COMMON/CCC/GRAPH,JO,DEMAND

      COMMON/AERO/NTALPH(2),NTBETA(2),NTMACH,TALPH(11,2),TALYH(11,2),
     *TXMAC(28),CAA(15,11,6),CNA(15,11,6),CMA(15,11,6),CYA(15,11,6),
     *CXA(15,11,6),CLA(15,11,6)

      COMMON/AERODI/AEROD(30,17),AEROB(50,6,3),TOMACH(25),DELCA(25),
     *PNMONE(15),BKFABT(50,2),BKFTHR(50,2),DELCN(25),DELCM(25)

      COMMON/AGEN1/TIME(2,16),TAUT(15),TAUW(15),TZERO,TLIFT,TTILT,TMINH,
     *DTZ,TQ,TL,XMAUG,TNE(6,15),ENGDAT(8,15),S(15),WD(15),WJET(15),
     *HEAD,PRINT(15),STEP(15),HNOM(15),HMAX(15),PROP(6),TBEGR,TENDR,
     *CHRD ,CHIRO,FAZZ,BO(3,2),CBAXIL,TCHIR,VRCUT,FPRFAC,CHVEL

      COMMON/AGEN2/AA,SA,CA,ALF,ALFY,CHIP,SCHIP,CCHIP,CHIY,SCHIY,CCHIY,
     *CHIR,SCHIR,CCHIR,STH,CTH,STHL,CTHL,DPHIZ,RTHE,R,VR,XM,SW,SU,SV,Q,
     *VIV(25),DVARS(13),DELXDW(15,3),DELXDR(7,5),DELXD(15,7),WZERO,ALT,
     *XMACH,QDOT,FAA,FAN,A(3,3),CASE,CT2,UMF,A12W,A22W,A32W,XJEXT,BEU,
     *BYL,BEL,BHMAX(15),BHMN,BSTEP(15),HNMB,HMXB,TPOLY,XMIAD
```



```
      COMMON/AGEN3/ITHR,IWD,IXR,JUMP,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISION,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)

      COMMON/AGEN4/QALPHA,QBETA,QCN,STH1,CTH1,XJX(3),XJV(3)

      COMMON/AGEN5/CHIBAS,CHPBAS,CHYBAS,CHISAV,TXX,TYY,TZZ,AMX,AMY,AMZ,
     *TMZ,THMFA,SIDE,WMAG,AZW,FANC,FLATC,FOM,VISC

      COMMON/ALAT/ALAT,ALONG,ALTLS,NTABLE

      COMMON/ARDC/PAT(14),ATMOSN,IATMOS,HBIAS,HAERO

      COMMON/ASTOR/ASTOR(4,67)

      COMMON/ATMOS/ATMTBL(110,6),MTHRRA,MATM1,MATM2,MNTH

      COMMON/AUTO/KDB(40),SAVCP(40),XLAMB(40,20),DP2

      COMMON/BODY/TBDWT(15,2),TZCG(15,2),TXCG(15,2),XLEN(2),ZREF(2),
     *XREF(2),KP(2),KY(2),AP(10,2),AY(10,2)

      COMMON/BGEN3/NOSS,JTB,NENT,NACT,NOM1,NP(7)

      COMMON/CISP/COISP(3,2)

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM

      COMMON/CONTRO/TTBL(30),CPTBL(30),CYTBL(30),TOBL(210),
     *CPOTBL(210),CYOTBL(210)

      common/cp_input/cp_input

      COMMON/DELFAB/DALT(25),DFAB(25),M30,M31,ITOL(15)

      COMMON/DELPAR/SPRADB,SYRADB

      COMMON/DELVP/DELVP,DELVG

      common/disp/key,tv15,ivar,tdisp,dispstep(2),irec

      COMMON/ELEVON/EMACH(25,2),ELEVON(25,4),M10,M11

      common/engines/engines(15,15)

      COMMON/ENPUT/PSIRST(6,5),HEADER(20),NVRST(5),KCDRES(6,5),LAST

      COMMON/FORINT/HBANK(2),T,TT,W,U,V,X,Y,Z,XMI,ZAP(18),DVAR(25),FSAVE
     *(625),NTRG1,TV1,NTRG2,TV2,NTRG3,TV3,NTRG4,TV4,NTRG5,TV5,NTRG6,TV6,
     *NTRG7,TV7,NTRG8,TV8,NTRG9,TV9,NTRG10,TV10,NTRG11,TV11,NTRG12,TV12,
     *NTRG13,TV13,KIND,KRDER,EU,EL,AYL,HMX,HMN,HNM

      COMMON/HEADUP/ HDMACH,HDCHRD
```

```
      COMMON/IUSE/IUSE

      COMMON/IIELE/IIELE,DN2(2)

      COMMON/INITP/MISSON(6),WTJET(15),WDLBS(15),AZWTBL(100),BCD(22)

      COMMON/IPR/IPR,WPMX,ILAST,KDT(15),PRK4,PRTOT,iprop(6),prk5(5)

      COMMON/IRTAOA/IRTAOA

      COMMON/ISTART/JSTART(6),ISTART(2),STARTV(40,2),SVXINC,SVXNOD,SVAZRE

      COMMON/IWNC/IWNUM

      COMMON/JJUMP/JJUMP

      common/linear_aero/linear_aero

      COMMON/JUMPST/JUMPST,TSAVET,WZERST

      COMMON/LPRINT/LPRINT,INMAX,PREDP,QY,TOL

      COMMON/LBM/TIMLBM(30),THRLBM(30),WDTLBM(30),PLBM(2),MLBM1,MLBM2,
     *ALBM(2),BLBM(2),CLBM(2),VLBM(2)

      common/mission/mission(2)

      common/mombal/mombal

        common/mpropin/mpflag

        common/mpropnm/mpname

      COMMON/MPS/TIMMPS(15),FTBL(15),THROT(15),GLIM(15),TAUALT(5),
     *TAUTBL(5),MPS1,MPS2,TGLIM(10,15),GLIMIT,tau_const(10,5),jthrot(15)

      common/mps1/timemps(15),taumps(15),mpsinp(15)

      common/mult/nchiot(6),tbias(6),nstage,nstg(15),wf(5),ibranch

      common/multi/char_vel(5),fpr_fac(5),w_marg(2)

      COMMON/NTIME/NTIME

      COMMON/OMSRCS/FOMS(15),FRCS(15),WDOMS(15),WDRCS(15),WDDUMP(15)

      COMMON/OLOW/OLOW,WINT

      common/plot/plot

      common/qalpha_max/qalpha_max,xnu1

      COMMON/QMAX/QMAX,QFLG,QPEN,MAXQ

      COMMON/REST/JSTR,NCNRS(5),NTCN,TR(5),LSB,NF8,NOPAR,WIBT(15)
```



```
      COMMON/RICONT/TTABLE(20),SIGMA,SIGTAB(20),M15,M16

      COMMON/RRRAIN/INIT,PRNTTB,SITEMO,NCHG,ALTSIG(7,2)

      COMMON/RTLS/VRINT(2),VSLOPE(2)

      COMMON/SRM/SRMTTB(600),SRMFTB(600),SRMWDT(600),SRMAE(600),MSRB

      COMMON/STAUT/STAUT(15)

      common/system/system

      common/tanks/level,mlox,nlox,wlox(100),hlox(100),mh2,nfuel,
     *wfuel(200),hfuel(200),wh2l,woxl

      COMMON/TAULIM/TAULIM,NTAU,NBETA,TIMTAU

      COMMON/TCL/TCL(6),TVL(6),TLL(6),RATIO

      common/time_jump/time_jump(5),njump

      common/t_liftoff/t_liftoff

      COMMON/VAERO/VAEROD(10,19),M18,M19,MBS18,MBS19

      COMMON/VIC/VIC(2),XK1(2),XK2(2),XK3(2),XK4(2),LK

      common/viv_sav/viv_sav(26)

      common/vrwtbl/vrwtbl(30),ivrtsw

      COMMON/WIND/ALTTBL(100),WTBL(100),AZTBL(100),NWIND

      COMMON/YLI/YLINT(7,20),PHITE(20),PSIREQ(20),KCDPHI(20),PSTR(20)

      DIMENSION CAO(30),CAALP(30),CNO(30),CNALP(30),CMO(30),CMALP(30),
     *CYO(30),CYBETA(30),CNBO(30),CNBETA(30),CLO(30),CLBETA(30),PNM(30),
     *TALFP(30),TBETA(30),TMACH(30),ALTBAS(50,3),BAXIAL(50,3),xcp(30),
     *BNORM(50,3),BPIT(50,3),QREFT(50),ORBASE(50),temp1(8,15),norder(15)

      DIMENSION DELALT(25),DELFAB(25),TBLPL(3,2),TBLISP(3,2),w_margin(2)

      data cp_input/.false./,mombal/.true./,mpflag/.false./
      data wh2l,woxl/591.0,17539.456/
      data ratio/.857142857/
c*********************************************************************

c               General Namelist Input Blocks

c*********************************************************************

      NAMELIST/INPUT/ CASE,HEAD,LAST,MISION,system,mission,ivar,tdisp,
     *dispstep,

c*********************************************************************
```

```
C                GENERAL PROPULSION
c*********************************************************************
      *NSYST,ENGDAT,TNE,PROP,engines,level,wh2l,woxl,
c*********************************************************************
C                MPS DATA
c*********************************************************************
      *MPSFLG,TIMMPS,FTBL,THROT,GLIM,TAUALT,TAUTBL,IPR,DELVG,TAULIM,NTAU,
      *TIMTAU,TBLPL,TBLISP,RATIO,tau_const,neng_out,timemps,taumps,
      *mpsinp,
c*********************************************************************
C                LBM DATA
c*********************************************************************
      *TIMLBM,THRLBM,WDTLBM,
c*********************************************************************
C                OMS AND RCS DATA
c*********************************************************************
      *FOMS,WDOMS,WDDUMP,   FRCS,WDRCS,
c*********************************************************************
C                THRUST EVENT DEPENDENT DATA
c*********************************************************************
      *TAUT,TAUW,WTJET,WDLBS,NOWD,PRINT,NOEVNT,
c*********************************************************************
C                ATTITUDE CONTROL DATA
c*********************************************************************
      *TTBL,CPTBL,CYTBL,KWTA,TOBL,CPOTBL,CYOTBL,IFACT,TMACH,TALFP,TBETA,
      *CHRDOT,KP,KY,TTABLE,SIGTAB,SPRADB,SYRADB,IPOLY,IHEAD,CHRDHD,
      *HDMACH,NBETA,ivrtsw,vrwtbl,
c*********************************************************************
C                EXTRA AERO DATA
c*********************************************************************
```



```
      *S,XLEN,XREF,ZREF,TOMACH,DELCA,DELCN,DELCM,NOBASE,CBAXIL,ITOL,
      *DELALT,DELFAB,
c*********************************************************************
C       CG ,WIND DATA, ATMOSPHERE SELECTION, AND ALTITUDE BIAS
c*********************************************************************
      *TBDWT,TXCG,TZCG,mpflag,     IWNUM,NWIND,ALTTBL,AZWTBL,WTBL,
      *IATMOS,MTHRRA,NCHG,ALTSIG,    HBIAS,
c*********************************************************************
C                TIMES
c*********************************************************************
      *DTZ,TZERO,TLIFT,TTILT,TCHIR,time_jump,
c*********************************************************************
C                CONSTANTS
c*********************************************************************
      *CJ,DJ,H,CMUE,OMEGA,GZERO,RE,FLAT,PTN,PTKG,PSFTM,
c*********************************************************************
C                INTEGRATION DATA
c*********************************************************************
      *AYL,BYL,EU,BEU,HMN,BHMN,KRDER,KRDERB,KIND,KINDB,STEP,BSTEP,
c*********************************************************************
C                LAUNCH AND INITIAL CONDITIONS
c*********************************************************************
      *ALAT,ALONG,ALTLS,AZ,FAZ,JUMP,VIV,WO1,
c*********************************************************************
C                OPTIONS
c*********************************************************************
      *ISTART,JSTART,JETET,LPRINT,NCOAST,NMAX,NTABLE,KCDPHI,KCDRES,KDB,
      *KDT,PSIREQ,PSIRST,NVRST,VRCUT,VRINT,VSLOPE,ICONSW,FPRFAC,mombal,
      *IRTLS,MSWCH,QMAX,IRTAOA,fpr_fac,nchiot,w_margin,char_vel,iprop,
c*********************************************************************
```

```
C               OPTIMIZATION DATA
c*********************************************************************
     *XJEXT,DP2,QY,TOL,MINH,NBGCT,NENDCT,NP,WIBT,qalpha_max
c*********************************************************************
c               Aerodynamic data input Namelist
c*********************************************************************
      NAMELIST/AEROIN/linear_aero,cp_input,
c*********************************************************************
C               FIRST STAGE AERO
c*********************************************************************
     *NTALPH,NTBETA,NTMACH,TALPH,TALYH,TXMAC,CAA,CNA,CMA,CYA,CXA,CLA,
c*********************************************************************
C               SECOND STAGE AERO
c*********************************************************************
     *PNM,CAO,CAALP,CNO,CNALP,CMO,CMALP,CYO,CYBETA,CNBO,CNBETA,CLO,
     *CLBETA,xcp,
c*********************************************************************
C               BASE DRAG DATA
c*********************************************************************
     *ALTBAS,ORBASE,BAXIAL,BNORM,BPIT,BKFABT,BKFTHR,QREFT,
c*********************************************************************
C               Elevon Data
c*********************************************************************
     *EMACH,ELEVON,VAEROD
c*********************************************************************
      NAMELIST/WEIGHTS/ASTOR,COMENT
c*********************************************************************
      DATA MISSON/'NOM   ','AOA  ','RTLS  ','POST N','POST A','VAR IY'/
      DATA MISION/'AOA   '/
```



```
      data mission/'ENGOUT','NOM   '/

      DATA BCD/'    Mass    ',' Velocity I ',' Gamma Iner ',
     *         '   Radius   ',' Altitude  ',              ',
     *         ' Velocity E ',' Azimuth E ',' Gamma Eart ',
     *         ' Inclination',' Desc Node ',              ',
     *         '            ',' Delta V   ','   Q Max    ',
     *         '  Stg Heat  ','            ','   RTLS DV  ',
     *         '            ','    Range   ','            ',
     *         '            '/

      IF(NTIME.eq.1) then
C *   REWIND BOPTBL TAPE                                          AINI  82
         REWIND 3
c*********************************************************************
c       Initialize internal parameters (if first case)
c*********************************************************************
         NOBASE=20
         CBAXIL=0.
         IHEAD=1
         CHRDHD=0.
         HDMACH=1000.
         IPR=0
         IFACT=1
         KIND=1
         KINDB=3
         KRDER=3
         KRDERB=3
         SPRADB=.1
         SYRADB=.1
         ALTLS=0.
         TLIFT=8.
         ISTART(1)=0
         ISTART(2)=0
         TOL=.005
         IPOLY=20
         ICONSW=20
         TBEGR=0.
         TENDR=0.
         NROLL=0
         LPRINT=0
         LAST=0
         NMAX=0
         NTABLE=0
         NWIND=0
         ivar=0
         tdisp=0.
         dispstep(1)=1.
         dispstep(2)=10.
```

```
      DO 1 I=1,15

      ITOL(I)=0
      WJET(I)=0.
      KDB(I)=0
      KDT(I)=0
      KDB(I+15)=0
      NOWD(I)=0.
      S(I)=0.
      TAUT(I)=0.
      TAUW(I)=0.
      MSWCH(I)=0
      MPSFLG(I)=0
      STEP(I)=4.
      BSTEP(I)=16.
      HMAX(I)=100000.
      BHMAX(I)=200000.
      PRINT(I)=8.
      GLIM(I)=.1E20
      WIBT(I)=1.
      nstg(i)=0
      mpsinp(i)=0
      taumps(i)=0.

      IF(I.le.10) then

         KCDPHI(I)=0
         PSIREQ(I)=0.

         IF(I.le.5) then

            NOEVNT(I)=0
            KCDRES(I,1)=0
            KCDRES(I,2)=0
            KCDRES(I,3)=0
            KCDRES(I,4)=0
            kcdres(i,5)=0
            kwta(i)=2
            np(i)=0
            NVRST(I)=0
            KCDRES(6,I)=0

            IF(I.le.2) then

              KP(I)=0
              KY(I)=0

            endif
         endif
      endif

1     CONTINUE

      STEP(1)=1.
      PI=3.14159265
```



```
         RAD=180./PI
         JUMP=1
         AA=90./RAD
         ALAT=28.608422
         ALONG=80.6041334
         DP2=.25
         XJEXT=1.
         QY=.8
         CASE=1.
         DELVG=0.
         EU=1.E-05
         BEU=2.E-05
         AYL=2.E-03
         BYL=4.E-05
         HMN=.25
         BHMN=.5
         CMUE=3.986012E14
         OMEGA=.729211595E-04
         CJ=1.62405E-03
         H=-6.40E-06
         DJ=6.9125E-06
         FLAT=1./298.25
         RE=6378160.
         GZERO=9.80665

      endif

      NCOAST=0
      PRTOT=0.

      do ijk=1,60
         head(ijk:ijk) = ' '
      enddo

      NOM1=0                                                    AINI  86
      LSB=0                                                     AINI  87
      KPRT=1                                                    AINI  88
      KPAGE=0                                                   AINI  89
      wint=0.

      CASE=float(int(case))
      NMAX=INMAX                                                AINI  93

      PTN=4.44822162                                            AINI 109
      PTKG=.45359237                                            AINI 110
      PSFTM=47.8802589
      PFN=1./PTN
      PFKG=1./PTKG
      PSFFM=1./PSFTM

      NTAU=0
      NBETA=0
      TAULIM=.1

      FTM=.3048
```

```
      DO I=1,15
         WDLBS(I)=WD(I)/PTKG
         WTJET(I)=WJET(I)/PTKG
      enddo

      CHRDOT=CHRD*RAD                                          AINI 130
      AZ=AA*RAD                                                AINI 131
      WO1=WZERO/PTKG                                           AINI 139
      FAZ=FAZZ*RAD                                             AINI 149
      w_margin(1)=w_marg(1)*pfkg
      w_margin(2)=w_marg(2)*pfkg

      IF(JSTART(1).ne.0) then

         J=jstart(1)

         DO I=1,25
            VIV(I)=STARTV(I,J)
         enddo

         TZERO=STARTV(26,J)
         WO1=STARTV(27,J)/PTKG
         CHIP=STARTV(28,J)
         CHIY=STARTV(29,J)
         OLOW=STARTV(30,J)
         xmaug=startv(31,j)

         CPOTBL(1)=CHIP
         CYOTBL(1)=CHIY

         JUMP=ISTART(J)

         N=NOEVNT(1)+1
         ISTART(J)=0

      endif

      DO I=1,100
         AZWTBL(I)=AZTBL(I)*RAD
      enddo

      DO I=1,30
         CAALP (I)=AEROD(I, 1)/RAD
         CAO   (I)=AEROD(I, 2)
         CLBETA(I)=AEROD(I, 3)/RAD
         CMALP (I)=AEROD(I, 4)/RAD
         CMO   (I)=AEROD(I, 5)
         CNALP (I)=AEROD(I, 6)/RAD
         CNBETA(I)=AEROD(I, 7)/RAD
         CNO   (I)=AEROD(I, 8)
         CYBETA(I)=AEROD(I, 9)/RAD
         CLO   (I)=AEROD(I,10)
         CYO   (I)=AEROD(I,11)
         CNBO  (I)=AEROD(I,12)
         xcp   (i)=aerod(i,13)
         PNM   (I)=AEROD(I,14)
```



```
         TBETA (I)=AEROD(I,15)*RAD
         TMACH (I)=AEROD(I,16)
         TALFP (I)=AEROD(I,17)*RAD
      enddo

      DO I=1,25
         DELALT(I)=DALT(I)/FTM
         DELFAB(I)=DFAB(I)*PFN
      enddo

      DO I=1,50
         DO J=1,3
            ALTBAS(I,J)=AEROB(I,1,J)/FTM
            BAXIAL(I,J)=AEROB(I,2,J)/PTN
            BNORM(I,J)=AEROB(I,3,J)/PTN
            BPIT(I,J)=AEROB(I,4,J)/PTN/FTM

            IF(J.EQ.1) then
               ORBASE(I)=AEROB(I,6,J)/PTN
               QREFT(I)=AEROB(I,5,J)/PSFTM
            endif
         enddo
      enddo

      DO I=1,210
         CPOTBL(I)=CPOTBL(I)*RAD
         CYOTBL(I)=CYOTBL(I)*RAD
      enddo

C*************************************************************************

C       For first case read input data

C*************************************************************************

      IF(NTIME.eq.1) then

         if(demand)
     *      write(*,*) ' Will an aero data base be input ? (y/n) [n]'

         read(*,'(a1)') answer

         if(answer.eq.'Y'.or.answer.eq.'y') then
C*************************************************************************

C     READ AERO DATA BASE

c*************************************************************************

            WRITE(6,*)' SPECIFY NAME OF DATA BASE E.G. NSC'

            READ(5,'(A24)') DBNAME

            OPEN(UNIT=8,FILE=DBNAME,ACCESS='DIRECT',RECL=1500,
     *      STATUS='OLD')
C
```

```
      CALL DBANK(EMACH(1,2),1,170,194)
      CALL DBANK(ELEVON(1,3),1,-220,244)
      CALL DBANK(ELEVON(1,4),1,-270,294)
      CALL DBANK(EMACH(1,1),1,-1070,1094)
      CALL DBANK(ELEVON(1,1),1,-1100,1124)
      CALL DBANK(ELEVON(1,2),1,-1130,1154)

      CALL IINRC(NTALPH(1),7,1)
      CALL IINRC(NTALPH(2),7,2)
      CALL IINRC(NTBETA(1),7,101)
      CALL IINRC(NTBETA(2),7,102)
      CALL IINRC(NTMACH,7,201)

      WRITE(31,*)'NTALPH NTBETA NTMACH ',NTALPH,NTBETA,NTMACH

      CALL DBANK(TALPH,7, 301,322)
      CALL DBANK(TALYH,7,-401,422)
      CALL DBANK(TXMAC,7,-501,528)

      CALL DBANK(ALTBAS(1,1),7,-701,750)
      CALL DBANK(BNORM(1,1),7,-801,850)
      CALL DBANK(BPIT(1,1),7,-901,950)
      CALL DBANK(BAXIAL(1,1),7,-1001,1050)
      CALL DBANK(BKFABT,7,-1101,1200)
      CALL DBANK(BKFTHR(1,1),7,-1201,1250)
      CALL DBANK(BKFTHR(1,2),7,-1301,1350)
      CALL DBANK(ORBASE,7,-1401,1450)
      CALL DBANK(QREFT,7,-1451,1500)

      CALL DBANK(BAXIAL(1,2),50,1,30)
      CALL DBANK(BAXIAL(1,3),50,-31,60)
      CALL DBANK(ALTBAS(1,2),50,-1350,1379)
      CALL DBANK(ALTBAS(1,3),50,-1350,1379)
      CALL DBANK(BNORM(1,2),51,1,30)
      CALL DBANK(BNORM(1,3),51,-31,60)
      CALL DBANK(BPIT(1,2),52,1,30)
      CALL DBANK(BPIT(1,3),52,-31,60)

      CALL DBANK(CAO,44,1400,1429)
      CALL DBANK(CAALP,44,-1451,1480)
      CALL DBANK(CNO,45,1400,1429)
      CALL DBANK(CNALP,45,-1451,1480)
      CALL DBANK(CMO,46,1400,1429)
      CALL DBANK(CMALP,46,-1451,1480)
      CALL DBANK(CYO,47,1400,1429)
      CALL DBANK(CYBETA,47,-1451,1480)
      CALL DBANK(CNBO,48,1400,1429)
      CALL DBANK(CNBETA,48,-1451,1480)
      CALL DBANK(CLO,49,1400,1429)
      CALL DBANK(CLBETA,49,-1451,1480)
      CALL DBANK(PNM,49,-1350,1379)

      CALL DBANK(VAEROD(1,19),53,1350,1359)
      CALL DBANK(VAEROD(1,1),53,-1401,1430)
      CALL DBANK(VAEROD(1,10),53,-1451,1480)
      CALL DBANK(VAEROD(1,4),54,1401,1430)
```



```
      CALL DBANK(VAEROD(1,13),54,-1451,1480)
      CALL DBANK(VAEROD(1,7),55,1401,1430)
      CALL DBANK(VAEROD(1,16),55,-1451,1480)

      NTB=NTBETA(1)

      DO I=1,NTB
         IF(ABS(TALYH(I,1)).LT.1.E-04) GO TO 322
      enddo

322   IB=I

      CALL AEBANK(CAA, 8,28,11,IB,NTMACH)

      CALL AEBANK(CNA,14,28,11,IB,NTMACH)

      CALL AEBANK(CMA,20,28,11,IB,NTMACH)

      NTA=NTALPH(2)

      DO I=1,NTA
         IF(ABS(TALPH(I,2)).LT.1.E-04) GO TO 325
      enddo

325   IA=I

      CALL AEBANK(CYA,26,28,11,IA,NTMACH)

      CALL AEBANK(CXA,32,28,11,IA,NTMACH)

      CALL AEBANK(CLA,38,28,11,IA,NTMACH)

   endif
c*************************************************************************
      IF(DEMAND) WRITE(6,*)' ENTER AERODYNAMIC DATA UPDATE FILE NAME'

      READ(5,'(a24)') NAMLST

      OPEN(UNIT=17,FILE=NAMLST,STATUS='OLD',READONLY,SHARED)

      write(*,*) ' You are using Aerodynamic namelist ',namlst

      READ(17,AEROIN)

      if(.not.linear_aero) then

         write(31,*)
         write(31,*) '     First stage axial force coefficent data'
         write(31,'(1x,a,11f10.2)') 'mach/alpha',(talph(j,1),j=1,11)

         do i=1,15
            write(31,'(2x,f10.2,11f10.5)') txmac(i),(caa(i,j,1),
     *      j=1,11)
         enddo
```

```
      write(31,*)
      write(31,*) '     First stage normal force coefficent data'
      write(31,'(1x,a,11f10.2)') 'mach/alpha',(talph(j,1),j=1,11)

      do i=1,15
         write(31,'(2x,f10.2,11f10.5)') txmac(i),(cna(i,j,1),
     *      j=1,11)
      enddo

      write(31,*)
      write(31,*) '     First stage pitching moment coefficent data'
      write(31,'(1x,a,11f10.2)') 'mach/alpha',(talph(j,1),j=1,11)

      do i=1,15
         write(31,'(2x,f10.2,11f10.5)') txmac(i),(cma(i,j,1),
     *      j=1,11)
      enddo

      write(31,*)
      write(31,*) '     First stage yaw force coefficent data'
      write(31,'(1x,a,11f10.2)') 'mach/beta ',(talyh(j,2),j=1,11)

      do i=1,15
         write(31,'(2x,f10.2,11f10.5)') txmac(i),(cya(i,j,1),
     *      j=1,11)
      enddo

      write(31,*)
      write(31,*) '     First stage yaw moment coefficent data'
      write(31,'(1x,a,11f10.2)') 'mach/beta ',(talyh(j,2),j=1,11)

      do i=1,15
         write(31,'(2x,f10.2,11f10.5)') txmac(i),(cxa(i,j,1),
     *      j=1,11)
      enddo

      write(31,*)
      write(31,*) '     First stage roll moment coefficent data'
      write(31,'(1x,a,11f10.2)') 'mach/beta ',(talyh(j,2),j=1,11)

      do i=1,15
         write(31,'(2x,f10.2,11f10.5)') txmac(i),(cla(i,j,1),
     *      j=1,11)
      enddo

   endif

   if(linear_aero) then

      write(31,*) ' Linearized Aerodynamic Coefficients'

   else

      write(31,'(46x,a)')' Second Stage Linearized Coefficients'
```



```
      endif

      write(31,'(a,a,a)')'   mach no.       cao   caalpha',
     *  '       cno     cnalp       cmo     cmalp       cyo',
     *  '    cybeta      cnbo    cnbeta       clo    clbeta'

      k=1

317   write(31,'(1x,f10.2,12f10.5)') pnm(k),cao(k),caalp(k),
     *  cno(k),cnalp(k),cmo(k),cmalp(k),cyo(k),cybeta(k),cnbo(k),
     *  cnbeta(k),clo(k),clbeta(k)

      k=k+1

      if(pnm(k).gt..1) go to 317


      write(31,*) ' base force input tables'

      write(31,*) ' booster stage base force input tables'
      write(31,*) '     altitude  base axial base normal ',
     *  ' base pitch       q ref'

      k=1

315   write(31,'(1x,f12.2,f12.3,f12.5,f12.3,f12.3)') altbas(k,1),
     *  baxial(k,1),bnorm(k,1),bpit(k,1),qreft(k)

      k=k+1

      if(altbas(k,1).gt.1.) go to 315

      write(31,*) ' upper stage base force input tables'

      write(31,*) '     altitude  base axial  base axial ',
     *  'base normal  base pitch       q ref'
      write(31,*) '               2 engine    3 engine'

      k=1

316   write(31,'(1x,f12.2,2f12.3,f12.5,f12.3,f12.3)') altbas(k,2),
     *  baxial(k,2),baxial(k,3),bnorm(k,2),bpit(k,1),qreft(k)

      k=k+1

      if(altbas(k,2).gt.1.) go to 316

c     WRITE(31,AEROIN)

c*****************************************************************************

      WRITE(6,*)' ENTER WEIGHT NAMELIST FILE'

      READ(5,'(A)') WTNL

      OPEN(UNIT=22,FILE=WTNL,STATUS='OLD')
```

```
      READ(22,WEIGHTS)

      kk=68
      if(system.eq.'STAGE15') kk=36
      if(system.eq.'HLLV') kk=34

      if(kk.lt.67) then
         do k=kk,67
            do i=1,4
               astor(i,k)=0.d0
            enddo
         enddo
      endif
C*********************************************************************

C     OPEN SRM DATA FILE FOR READING

C*********************************************************************

      if(demand) write(*,*) ' Are SRMs to be simulated ? (y/n)[n]'

      read(*,'(a1)') answer

      if(answer.eq.'Y'.or.answer.eq.'y') then

         IF(DEMAND)
     *   WRITE(6,*)' INPUT THE SRM DATA FILE NAME TO READ.'

         READ(5,'(a24)') SRBNAM

         IF(SRBNAM(1:4).EQ.'    ') THEN

            WRITE(6,*) ' SRB FILE NAME WAS NOT ENTERED , STOP PROGRAM'

            STOP

         ELSE

            IF(DEMAND) WRITE(6,'(a,5x,a24)') ' You are using',SRBNAM

         ENDIF
C
         CLOSE(UNIT=41)

         OPEN(UNIT=41,
     *      FILE=SRBNAM,                !************** SMS CHANGE
     *      STATUS='UNKNOWN',
     *      ACCESS='SEQUENTIAL',
     *      FORM='FORMATTED',READONLY,SHARED,                ! CHANGE
     *      IOSTAT=IOS)

         IF(IOS.GT.0) THEN

            WRITE(JO,*) 'ERROR OPENING SRM FILE'
            IF(DEMAND) WRITE(6,*) ' ERROR OPENING SRM FILE'
```



```
            CLOSE(UNIT=41)

            STOP

         ENDIF

C*********************************************************************

c       READ SRM DATA FILE

C*********************************************************************

         I = 0
 4       I = I + 1

         READ(41,5,END=6,ERR=7) SRMV1,SRMV2,SRMV3,SRMV4,SRMV5,SRMV6

 5       FORMAT(F7.3,2F10.3,F13.1,2F10.3)

         SRMTTB(I) = DBLE(SRMV1)
         SRMFTB(I) = DBLE(SRMV4)
         SRMWDT(I) = DBLE(SRMV5)
         SRMAE(I)  = DBLE(SRMV6)

         IF(I.EQ. 600) GO TO 10

         GO TO 4
C
 6       WRITE(JO,*) 'END-OF-FILE ENCOUNTERED READING SRM DATA'

         IF(DEMAND)
     *   WRITE(6,*)' END-OF-FILE ENCOUNTERED READING SRM DATA'

         GO TO 10
C
 7       WRITE(JO,*) 'ERROR READING SRM DATA FILE'

         IF(DEMAND) WRITE(6,*) ' ERROR READING SRM DATA FILE'

         CLOSE(UNIT=41)

         STOP
C
 10      CONTINUE

         CLOSE(UNIT=41)

C*********************************************************************

C     WRITE SRM DATA ON OUTPUT FILE

c*********************************************************************

         WRITE(JO,21) SRBNAM
```

```
 21         FORMAT(1H1,41X,'SRM THRUST AND FLOWRATE TABLES'/45X,
     *      A24// 2X,'TIME',7X,'VAC THRUST',4X,'FLOW RATE',5X,
     *      'EXIT AREA',4X,'I',5X,'TIME',7X,'VAC THRUST',4X,
     *      'FLOW RATE',5X,'EXIT AREA'/2X,'(SEC)',9X,'(LBS)',6X,
     *      '(LBS/SEC)',7X,'(SQIN)',5X,'I',5X,'(SEC)',9X,'(LBS)',
     *      6X,'(LBS/SEC)',7X,'(SQIN)')

            NLINES=5

            DO I=1,300
               J=I+300

               WRITE(JO,22) SRMTTB(I),SRMFTB(I),SRMWDT(I),SRMAE(I),
     *                      SRMTTB(J),SRMFTB(J),SRMWDT(J),SRMAE(J)

 22            FORMAT(1X,F7.3,5X,F9.1,5X,F9.3,5X,F9.3,4X,'I',
     *                4X,F7.3,5X,F9.1,5X,F9.3,5X,F9.3)
C
               NLINES=NLINES+1
               IF(NLINES.EQ.55) THEN
                  WRITE(JO,21) SRBNAM
                  NLINES=5
               ENDIF
            enddo

C*********************************************************************

C     CONVERT SRM EXIT AREA TO METRIC UNITS

C*********************************************************************

            DO I=1,600
               SRMAE(I)=SRMAE(I)/144.*FTM**2
            enddo

         endif

      endif

      CLOSE (UNIT=15)

C*********************************************************************

C       NEW FOR VMR STUDY
C
C       IF(DEMAND)WRITE(6,*)'ENTER INITIAL MIXTURE RATIO'
C       READ(5,*)VMRINT
C       IF(DEMAND)WRITE(6,*)'ENTER FINAL MIXTURE RATIO'
C       READ(5,*)VMRFIN
C       IF(DEMAND)WRITE(6,*)'ENTER SWITCH TIME'
C       READ(5,*)VMRSWH
C       END OF VMR CHANGES

C*********************************************************************

      IF(DEMAND) WRITE(6,*)' ENTER TRAJECTORY NAMELIST FILE NAME'
```



```
      READ(5,'(a24)') NAMLST

      OPEN(UNIT=15,FILE=NAMLST,STATUS='OLD',READONLY,SHARED)

      write(*,*) ' You are using trajectory namelist ',namlst

C*********************************************************************

      READ(15,INPUT)

      if(ntable.eq.1) then

         CLOSE(UNIT=9)

         OPEN(UNIT=9,FILE=MISION,FORM='UNFORMATTED',
     *   RECL=163,STATUS='NEW')                              !    CHANGE

      endif
      if(ntime.eq.1) then
         if(mpflag) then
            read(*,'(a30)') mpname
            write(*,'(a,a)') ' You are using mass properties file ',
     *      mpname
            call masspro(eventnum,moboos,momain,bomdot,mnmdot,loncg,
     *           latcg,norcg,loncgp,latcgp,norcgp)
         endif
      endif
      if(demand)
     *   write(*,*) 'Is plot file to be generated ? (y/n)[n]'
      read(*,'(a1)') answer

      plot=.false.

      if(answer.eq.'Y'.or.answer.eq.'y') then

         plot=.true.

         if(demand) write(*,*) ' Input name of output plot file'
         read(*,'(a30)') file_name

         open(unit=19,file=file_name,status='new')

         rewind 19

      endif

      if(ivar.ne.0) then

         write(*,*) ' Name of dispersion data file'

         read(*,'(a30)') file_name

         open(unit=80,file=file_name,status='old',access='direct',
     *   recl=104)
```

```
      endif

      WRITE(6,*)'IN AINIT IWNUM= ',IWNUM

      IF(NTIME.EQ.1 .AND. IWNUM.GE.0) CALL WINDIN

C*****************************************************************************

C     CALCULATE SSME ISP COEFFICIENTS AS A FUNCTION OF POWER LEVEL

C*****************************************************************************

      IF(NTIME.EQ.1) CALL SIMUL(TBLPL,TBLISP,COISP)

      REWIND 1
      REWIND 2
      REWIND 4
      REWIND 10
      REWIND 11
      REWIND 12
      REWIND 13

C*****************************************************************************

c       Establish or read files for first stage restart

C*****************************************************************************

      if(jump.eq.1.and.tzero.gt..1) then

         if(demand) write(*,*)' first stage jump start file name'
         read(*,'(a24)') file_name

         open(unit=18,file=file_name,status='old',readonly)

         rewind 18

500      read(18,*,end=501) ttt,viv_sav,xmass,wint,t_liftoff,chpbas,
     *   chybas

         if(abs(ttt-tzero).gt..01) go to 500

         go to 502

501      write(*,*) ' end of file encountered before obtaining data'
         stop

502      tzero=ttt

         wo1=xmass/ptkg

         close (unit=18)

      endif

      if(time_jump(1).gt.0..and.tzero.lt..1) then
```



```
         if(demand) write(*,*)' name of jump start output file'

         read(*,'(a24)') file_name

         open(unit=18,file=file_name,status='new')

         rewind 18

      endif

C*****************************************************************************

      IF(JUMP.ne.1) then

         WO1=WO1-WTJET(JUMP-1)

         IF(MISION.eq.MISSON(3)) then

C*****************************************************************************

C     SET UP Azimuth (AZRE) CONSTRAINT FOR RTLS

C*****************************************************************************

            DO I=1,6
               IF(KCDRES(I,1).eq.8.or.KCDRES(I,1).eq.0) then
                  KCDRES(I,1)=8
                  PSIRST(I,1)=SVAZRE+180.
                  IF(SVAZRE.GE.0.) PSIRST(I,1)=SVAZRE-180.
                  GO TO 350
               endif
            enddo
         endif

         IF(MISION.eq.MISSON(6)) then

C*****************************************************************************

C     SET UP INCLINATION AND NODE CONSTRAINTS FOR AOA(VARIABLE IY)

C*****************************************************************************

            DO I=1,20
               IF(KCDPHI(I).eq.10 .or. KCDPHI(I).eq.0) then
                  KCDPHI(I)=10
                  KCDPHI(I+1)=11
                  PSIREQ(I-1)=SVXINC*RAD
                  PSIREQ(I)=SVXNOD
                  MISION=MISSON(2)
                  GO TO 350
               endif
            enddo
         endif
      endif
```

```
 350 IF(JETET.ne.0) then

        M=1
        IF(MISION.EQ.MISSON(5)) M=2

        WTJET(JETET)=ASTOR(M,30)-ASTOR(M,22)+ASTOR(M,18)+ASTOR(M,19)+
     *                ASTOR(M,20)+ASTOR(M,21)

      endif

      IF(NTABLE.eq.1) REWIND 9

c     IF(JSTART(1).EQ.0) JSTART(1)=ISTART(1)

      DO I=1,15
         IF(I.LT.JUMP.AND.KDB(I).NE.0) KDB(I)=0
         IF(I.le.13) then
            IF(JUMP.GT.NOEVNT(1).AND.KDB(I+15).NE.0) KDB(I+15)=0
         endif
      enddo

      IF(IPR.NE.0.AND.KDB(IPR).NE.0) KDB(IPR)=0                    AINI 177

      IF(ABS(CHRDOT).LT.1.E-04) NROLL=-1                           AINI 195

      FAZZ=FAZ/RAD                                                 AINI 196
      IF(FAZZ.LT.0.) FAZZ=FAZZ+2.*PI                               AINI 197

      CHRD=CHRDOT/RAD                                              AINI 198
      HDCHRD=CHRDHD/RAD
      AA=AZ/RAD                                                    AINI 199
      INMAX=NMAX                                                   AINI 200
      THET1=(90.-ALAT)/RAD                                         AINI 201
      WZERO=WO1*PTKG
      qalpha_max=qalpha_max*psftm/rad

      DO I=1,15
         STAUT(I)=TAUT(I)
         WD(I)=WDLBS(I)*PTKG
         WJET(I)=WTJET(I)*PTKG
      enddo

      EL=EU/2.**(KRDER+2)                                          AINI 378
      BEL=BEU/2.**(KRDERB+2)

      DO I=1,7
         IF(NP(I).ne.0) then
            IF(NP(I).EQ.1) NP(I)=3
            IF(NP(I).GE.30) NP(I)=29
            J=(NP(I)/2)*2
            IF(J.EQ.NP(I)) NP(I)=NP(I)+1
         endif
      enddo

C*********************************************************************
```



```
C     CALCULATE NUMBER OF PARAMETERS                                 AINI 383
c*********************************************************************

      NOPAR=0                                                        AINI 384

      DO I=1,40
         IF(KDB(I).ne.0) NOPAR=NOPAR+1
      enddo

C*********************************************************************

C     CALCULATE NUMBER OF REGULAR SETS                               AINI 395

c*********************************************************************

      NOSS=0                                                         AINI 397

      DO I=1,7
         IF(KCDPHI(I).ne.0) NOSS=NOSS+1
      enddo

C*********************************************************************

C     CALCULATE NUMBER OF RESTART CONSTRAINTS                        AINI 403

c*********************************************************************

      NTCN=0

      DO K=1,5
         NCNRS(K)=0
         TR(K)=1.E19

         IF(NVRST(K).ne.0) then
            DO I=1,6
               IF(KCDRES(I,K).ne.0) NCNRS(K)=NCNRS(K)+1
            enddo
            NTCN=NTCN+NCNRS(K)
         endif
      enddo

      IF((NOSS+NTCN).gt.20) then

         WRITE(JO,*)' TOO MANY CONSTRAINTS'

         WRITE(JO,INPUT)

         STOP

      endif

c*********************************************************************

C     CALCULATE NO OF THRUST EVENTS                                  AINI 418
```

```
c*********************************************************************

        NVNT=0
        k=0

        do i=1,5
           if(noevnt(i).ne.0) then
              nvnt=nvnt+noevnt(i)
              ja=i
              do j=1,noevnt(i)
                 k=k+1
                 nstg(k)=i
              enddo
           endif
        enddo

        nstg(k+1)=nstg(k)+1

        nstage=nstg(jump)

        do i=1,15
           if(kdb(i).ne.0) then
              if(jump.gt.i.or.nvnt.lt.i) then
                 kdb(i)=0
                 nopar=nopar-1
              endif
           endif
        enddo

        ILAST=NVNT-NOEVNT(JA)+1

c*********************************************************************

C     CALCULATE NO OF WEIGHT EVENTS

c*********************************************************************

        NWVNT=0

        do i=1,15
           IF(NOWD(I).ne.0) NWVNT=NWVNT+1
        enddo

c*********************************************************************

c       reconfigure engine locations and associated data for engine out
c       assumes one engine will be directed through the center of
c       gravity and other engines will be used for controllability

c*********************************************************************

        if(neng_out.ne.0) then

           jeng=int(tne(1,1)+.001)

c        define roll adjustment angle
```



```
           ygpout=engdat(2,neng_out)
           zgpout=engdat(3,neng_out)

           roll=atan2(-ygpout,zgpout)
           sroll=sin(roll)
           croll=cos(roll)

           write(*,*) ' The roll angle adjustment for engine out is ',
     *     roll*rad
           write(31,*) ' The roll angle adjustment for engine out is ',
     *     roll*rad

c       define new locations of engines

           do i=1,jeng
              if(engines(i,jump).ne.0) then
                 ygp=engdat(2,i)
                 zgp=engdat(3,i)

                 if(abs(ygp+ygpout).lt..001.and.abs(zgp+zgpout).lt.
     *           .001.and.i.ne.neng_out) nfixed=i

                 engdat(2,i)=ygp*croll+zgp*sroll
                 engdat(3,i)=-ygp*sroll+zgp*croll

                 write(*,*) i,ygp,zgp,engdat(2,i),engdat(3,i)
                 write(31,*) i,ygp,zgp,engdat(2,i),engdat(3,i)

              endif
           enddo

           write(*,*) ' fixed engine is number ',nfixed
           write(*,*) ' engine number ',neng_out,' is out'
           write(31,*) ' fixed engine is number ',nfixed
           write(31,*) ' engine number ',neng_out,' is out'

c       define order of engines such that out engine and fixed engine
c       are placed last

           i=1
           j=0
  220      continue

           if(i.ne.neng_out.and.i.ne.nfixed.and.engines(i,jump).
     *     eq.1) then
              j=j+1
              norder(j)=i
           else if(i.eq.neng_out) then
              norder(jeng-1)=i
           else if(i.eq.nfixed) then
              norder(jeng)=i
           endif

           i=i+1
           if(i.le.jeng) go to 220
```

```
c       relocate engines based on new order

           do i=1,jeng
              do j=1,8
                 templ(j,i)=engdat(j,norder(i))
              enddo
           enddo

           do i=1,jeng
              if(engines(i,jump).eq.1) then
                 do j=1,8
                    engdat(j,i)=templ(j,i)
                 enddo
              endif
           enddo

           write(31,*) ' new engdat array'

           do i=1,jeng
              write(31,'(1x,i3,5f9.3,f9.1,2f9.3)') i,(engdat(j,i),j=1,8)
           enddo

c       update nsyst and tne input parameters

           nsyst(jeng-1)=0
           nsyst(jeng)=6

           write(31,*) ' new nsyst array'
           write(31,'(1x,15i3)') nsyst

           write(31,*) ' new tne array'

           do i=1,15
              if(neng_out.le.int(tne(1,i)+.001)) then
                 tne(1,i)=tne(1,i)-2.
                 tne(6,i)=1.
              endif
              write(31,'(1x,i5,6f9.3)') i,(tne(j,i),j=1,6)
           enddo

        endif

c**********************************************************************
C     CALCULATE PERFORMANCE RESERVE CONSTANTS
c**********************************************************************

        IF(IPR.ne.0) then

           XISP=ENGDAT(6,1)/ENGDAT(7,1)
           PRK4=1.-EXP(-DELVG/(GZERO*XISP))

C***  CHECK KDT SETUP FOR IPR
```



```
           do j=ilast,nvnt
              if(kdb(j).ne.0) then
                 kdt(j)=ipr-j
                 if(kdt(j).le.0) then
                    kdt(j)=0
                    kdb(j)=0
                    nopar=nopar-1
                 endif
              endif
           enddo

           if(kdb(16).ne.1) then
              kdb(16)=1
              nopar=nopar+1
           endif
        endif

        ii=noevnt(1)
        tjetwt=0.

        do i=jump,nvnt
           tjetwt=tjetwt+wjet(i)+wd(i)
        enddo

        do i=1,ii
           lstge(i)=1
        enddo

        ii=ii+1

        do i=ii,nvnt
           lstge(i)=2
        enddo

        IF(NVNT.NE.15) LSTGE(NVNT+1)=LSTGE(NVNT)

   20   CONTINUE

        do i=1,40
           do j=1,20
              xlamb(i,j)=0.
           enddo
        enddo

c**********************************************************************
c       Call appropriate atmospheric model to define initial conditions
c**********************************************************************

      PAT(1)=0. + HBIAS

      if(iatmos.lt.0.or.iatmos.gt.6) iatmos=1

      if(iatmos.eq.0) then
```

```
         if(ntime.eq.1) call rraint(atmosn)

         call rraspl(pat)

      else

         call atmosphere(pat,atmosn,ier,iatmos)

      endif

      pat(1)=0.
      psl=pat(2)*10000.                                              AINI 442
      if(iatmos.eq.0) psl=pat(2)

c*********************************************************************

      STH1=SIN(THET1)                                                AINI 443
      CTH1=COS(THET1)                                                AINI 444
      STHL=STH1                                                      AINI 445
      CTHL=CTH1                                                      AINI 446

c*********************************************************************

C     oblate earth constants

c*********************************************************************

      UMF=1.-FLAT
      TEMP=PI/2.-ATAN2(UMF*UMF*CTH1,STH1)                            AINI 457

      STH=SIN(TEMP)                                                  AINI 458
      CTH=COS(TEMP)                                                  AINI 459

      STHL=STH
      CTHL=CTH

      R=UMF*RE/SQRT((UMF*STH)**2+CTH*CTH)+ALTLS                      AINI 460

   30 WRZ=R*STH *OMEGA                                               AINI 461

      DPHIZ=OMEGA*DTZ                                                AINI 462

      SDPHI=SIN(DPHIZ)                                               AINI 463
      CDPHI=COS(DPHIZ)                                               AINI 464

      IF(JUMP.eq.1) then

         BO(1,1)=CDPHI
         BO(1,2)=SDPHI*STH
         BO(2,1)=0.
         BO(2,2)=CTH
         BO(3,1)=-SDPHI
         BO(3,2)=CDPHI*STH

         XJV(1)=WRZ*BO(1,1)
```



```
         XJV(2)=0.
         XJV(3)=WRZ*BO(3,1)

         XJX(1)=R*BO(1,2)
         XJX(2)=R*BO(2,2)
         XJX(3)=R*BO(3,2)

      endif

      ALT1=10000.020
      ALT2=14000.028                                                 AINI 501
      ALT=0.                                                         AINI 502

c*********************************************************************

c     SET UP HEADERS FOR CONSTRAINTS                                 AINI 503

c*********************************************************************

      KV=1                                                           AINI 513

      DO I=1,20
         IF(KCDPHI(I).ne.0) then
            J=KCDPHI(I)
            HEADER(KV)=BCD(J)
            KV=KV+1
         endif
      enddo

c*********************************************************************

C     PRINT NAMELIST INPUT HERE

c*********************************************************************

      IF(NTIME.EQ.1) THEN

         WRITE(JO,314) HEAD

         WRITE(JO,AEROIN)

      ENDIF

      WRITE(JO,314) HEAD
  314 FORMAT(1H1,33X,A60,//)

      WRITE(JO,INPUT)


      NTIME=NTIME+1

c*********************************************************************

c       Store aero data into storage data tables

c*********************************************************************
```

```
      DO I=1,30
         AEROD(I, 1)=CAALP  (I)*RAD
         AEROD(I, 2)=CAO    (I)
         AEROD(I, 3)=CLBETA(I)*RAD
         AEROD(I, 4)=CMALP  (I)*RAD
         AEROD(I, 5)=CMO    (I)
         AEROD(I, 6)=CNALP  (I)*RAD
         AEROD(I, 7)=CNBETA(I)*RAD
         AEROD(I, 8)=CNO    (I)
         AEROD(I, 9)=CYBETA(I)*RAD
         AEROD(I,10)=CLO    (I)
         AEROD(I,11)=CYO    (I)
         AEROD(I,12)=CNBO   (I)
         aerod(i,13)=xcp    (i)
         AEROD(I,14)=PNM    (I)
         AEROD(I,15)=TBETA  (I)/RAD
         AEROD(I,16)=TMACH  (I)
         AEROD(I,17)=TALFP  (I)/RAD
      enddo

      DO I=1,25
         DALT(I)=DELALT(I)*FTM
         DFAB(I)=DELFAB(I)*PTN
      enddo

      DO I=1,50
         DO J=1,3
            AEROB(I,1,J)=ALTBAS(I,J)*FTM
            AEROB(I,2,J)=BAXIAL(I,J)*PTN
            AEROB(I,3,J)=BNORM(I,J)*PTN
            AEROB(I,4,J)=BPIT(I,J)*PTN*FTM
            IF(J.EQ.1) then
               AEROB(I,6,J)=ORBASE(I)*PTN
               AEROB(I,5,J)=QREFT(I)*PSFTM
            endif
         enddo
      enddo

      DO I=1,210
         CPOTBL(I)=CPOTBL(I)/RAD
         CYOTBL(I)=CYOTBL(I)/RAD
      enddo

      DO I=1,100
         AZTBL(I)=AZWTBL(I)/RAD
      enddo

      w_marg(1)=w_margin(1)*ptkg
      w_marg(2)=w_margin(2)*ptkg

c       Define initial guesses for flight performance reserve for each
c       stage based on input estimate of characteristic velocity (Char_vel)
c       and fpr coefficient (Fpr_fac) (The iprop flag must be set to the number
c       of the thrust event where the fpr will be used in the calculation of
c       the final mass of the stage
```



```
      do i=1,5
         if(iprop(i).ne.0) prk5(i)=exp(char_vel(i)*fpr_fac(i)/
     *   (gzero*funisp(1.,i)))
      enddo

      JJUMP=0

      IF(JUMP.eq.1.and.MINH.gt.NOEVNT(1).and.istart(1).eq.0) then

         J=1
         K=NOEVNT(1)
         L=1

         DO I=J,K
            IF(KDB(I).EQ.1) GO TO 64
         enddo

         J=16
         K=28
         L=2

         DO I=J,K
            IF(KDB(I).EQ.1) GO TO 64
         enddo

         JJUMP=1

      endif

   64 CONTINUE

      JUMPST=0

      DO L=1,4
         K=5-L
         IF(NVRST(K).ne.0) then
            DO I=1,6
               IF(KCDRES(I,K).ne.0) then
                  J=KCDRES(I,K)
                  HEADER(KV)=BCD(J)
                  KV=KV+1
               endif
            enddo
         endif
      enddo

      IK1=0
      IJ1=0
      LK=0
      K=0

      DO I=2,20
         IF(KCDPHI(I).EQ.4) IJ=I-1
         IF(KCDPHI(I).EQ.14) IJ1=I-1
         IF(I.le.7) then
```

```
            DO L=1,5
               IF(KCDRES(I-1,L).EQ.4) IK=I-1
               IF(KCDRES(I-1,L).EQ.14) IK1=I-1
               IF(IK1.NE.0.AND.K.EQ.0) K=L
            enddo
         endif
      enddo

      DO 70 I=1,2

      IF(I.EQ.1) then

         IF(IJ1.eq.0) GO TO 70

         LK=LK+1
         RM=PSIREQ(IJ1)
         RI=PSIREQ(IJ)
         PSIREQ(IJ1)=0.

      else

         IF(IK1.EQ.0) GO TO 70

         LK=LK+1
         RM=PSIRST(IK1,K)
         RI=PSIRST(IK,K)
         PSIRST(IK1,K)=0.

      endif

150   VIC(LK)=SQRT(2.*CMUE*RM/(RI*(RI+RM)))
      RHO=RI/RM

      XK1(LK)=CJ/3.*RE*RE/(RI*RI)
      XK2(LK)=3.+RHO*(3.-RHO)
      XK3(LK)=1.+RHO*(1.+RHO)
      XK4(LK)=-2.*RHO*RHO

   70 CONTINUE

      if(level.ne.0) then

         if(demand) write(*,*) 'name of LOX table file'
         read(*,'(a)') file_name

         open(unit=23,file=file_name,form='formatted',status='old')

         nlox=1
80       read(23,'(f9.3,f13.2)',end=90) hlox(nlox),wlox(nlox)
         nlox=nlox+1
         go to 80
90       nlox=nlox-1

         close(23)

         if(demand) write(*,*) 'name of LH2 table file'
```



```
         read(*,'(a)') file_name

         open(unit=24,file=file_name,form='formatted',status='old')

         nfuel=1
100      read(24,'(f9.3,f13.2)',end=110) hfuel(nfuel),wfuel(nfuel)
         nfuel=nfuel+1
         go to 100
110      nfuel=nfuel-1

         close(24)

         write(7,120) head

120      format(36x,a60/
     *   33x,'Tables of Tank Height, Propellant Mass, Total Thrust and'
     *   ,' Flowrate'/4x,'Time',6x,'Acel',4x,'LOX Level',7x,'LOX Mass',
     *   2x,'Fuel Level',6x,'Fuel Mass',7x,'Flowrate',5x,'Vac Thrust',
     *   5x,'Propellant',4x,'Inlet Pres'/3x,'(sec)',5x,'(G''s)',2(4x,
     *   '(inches)',7x,'(pounds)'),6x,'(lbs/sec)',2(7x,'(pounds)'),3x,
     *   '(lbs/inch^2)')

      endif

      RETURN                                                          AINI 557
      END                                                             AINI 558
```

3.12 Subroutine AMULG

3.12.1 Purpose

This generic subroutine is used for linear interpolation. The subroutine can be used to interpolate two dependent variables for a single independent variable.

3.12.2 Variable Listing

3.12.3 Subroutines Called:

None

3.12.4 Calling Subroutines:

Name	Function
ACNTRO	Interpolates pitch and yaw attitude angles during min-H portion of simulation
ADER	Calculates the time independent portion of the equations of motion during forward integration
ADER1	Calculates the time dependent portion of the equations of motion during forward integration
APRTN	Prints trajectory block output and files
BDR1I	Calculates components of the equations of motion for the backward trajectory

3.12.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
del	Temporary variable				Local	
diff	Difference between current independent variable and values in the independent table				Local	
m	Index defining place in table for current interpolation				Local	
mm1	m-1				Local	
mp1	m+1				Local	
n	Number of points in independent table				Local	
x	Independent variable table				Local	
x t	Independent variable table				Local	
y 1	Dependent variable table for first parameter in argument list				Local	
y 1 t	Dependent variable table for first parameter in argument list				Local	
y 2	Dependent variable table for second parameter in argument list				Local	
y 2 t	Dependent variable table for second parameter in argument list				Local	

```
      SUBROUTINE AMULG(L,M,N,X,XT,Y1,Y1T,Y2,Y2T)
C
C      LINEAR INTERPOLATION SCHEME
C          L      = NUMBER OF DEPENDENT VARIABLES(1 OR 2)
C          M      = PLACE IN TABLE(SET=0 INITIALLY;UPDATED BY ROUTINE)
C          N      = NUMBER OF POINTS IN INDEPENDENT TABLES
C          X      = VALUE OF INDEPENDENT VARIABLE
C          XT     = INDEPENDENT VARIABLE TABLE
C          Y1     = VALUE OF DEPENDENT VARIABLE 1
C          Y1T    = DEPENDENT VARIABLE TABLE FOR 1
C          Y2     = VALUE OF DEPENDENT VARIABLE 2
C          Y2T    = DEPENDENT VARIABLE TABLE FOR 2
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION XT(1),Y1T(1),Y2T(1)
C
      DIFF=X-XT(M)

      IF(DIFF.lt.0.) then

C     GET HERE IF X.LT.XT(M)

1        IF(M.EQ.1) then

            Y1=Y1T(1)
            IF(L.EQ.2) Y2=Y2T(1)

            RETURN

         endif

         M=M-1

         IF(X.lt.XT(M)) then

            go to 1

         else if(x.eq.xt(m)) then

            y1=y1t(m)
            if(l.ne.1) y2=y2t(m)

            return

         endif

         m=m+1

      else if(diff.gt.0.) then

C     GET HERE IF X.GT.XT(M)

2        IF(M.EQ.N) then

            y1=y1t(m)
```



```
            if(l.ne.1) y2=y2t(m)

            return

         endif

         MP1=M+1

         IF(XT(MP1).LT.XT(M)) then

            y1=y1t(m)
            if(l.ne.1) y2=y2t(m)

            return

         endif

         M=MP1

         IF(X.gt.XT(M)) then

            go to 2

         else if(x.eq.xt(m)) then

            y1=y1t(m)
            if(l.ne.1) y2=y2t(m)

            return

         endif

      else if(diff.eq.0.) then

         Y1=Y1T(M)
         IF(L.NE.1)Y2=Y2T(M)

         RETURN

      endif

      MM1=M-1

      DEL=(X-XT(MM1))/(XT(M)-XT(MM1))

      Y1=Y1T(MM1)+DEL*(Y1T(M)-Y1T(MM1))

      IF(L.NE.1)Y2=Y2T(MM1)+DEL*(Y2T(M)-Y2T(MM1))

      RETURN

      END
```

3.13 Subroutine AMULG7

3.13.1 Purpose

This generic subroutine is used for linear interpolation of seven dependent variables for a single independent variable. This subroutine is used to interpolate the state variables during the backward integration.

3.13.2 Variable Listing

3.13.3 Subroutines Called:

None

3.13.4 Calling Subroutines:

Name	Function
BDR1I	Calculates equations of motion for backward integration

3.13.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
del	Temporary variable				Local	
diff	Difference in current independent parameter and places in independent table				Local	
m	Index place in table				Local	
mm1	m minus 1				Local	
mp1	m plus 1				Local	
n	Independent variable table				Local	
x	Independent variable table				Local	
x t	Independent variable table				Local	
y	Value of dependent variable				Local	
y t	Dependent variable tables				Local	

```
      SUBROUTINE AMULG7(M,N,X,XT,Y,YT)
C
C      LINEAR INTERPOLATION SCHEME
C          L      = NUMBER OF DEPENDENT VARIABLES(1 OR 2)
C          M      = PLACE IN TABLE(SET=0 INITIALLY;UPDATED BY ROUTINE)
C          N      = NUMBER OF POINTS IN INDEPENDENT TABLES
C          X      = VALUE OF INDEPENDENT VARIABLE
C          XT     = INDEPENDENT VARIABLE TABLE
C          Y1     = VALUE OF DEPENDENT VARIABLE 1
C          Y1T    = DEPENDENT VARIABLE TABLE FOR 1
C          Y2     = VALUE OF DEPENDENT VARIABLE 2
C          Y2T    = DEPENDENT VARIABLE TABLE FOR 2
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION XT(n),y(11),YT(n,11)
C
      DIFF=X-XT(M)

      IF(DIFF.lt.0.) then

C     GET HERE IF X.LT.XT(M)

1        IF(M.EQ.1) then

            do 5 i=1,11
5           y(i)=yt(1,i)

            RETURN

         endif

         M=M-1

         IF(X.lt.XT(M)) then

            go to 1

         else if(x.eq.xt(m)) then

            do 10 i=1,11

10          y(i)=yt(m,i)

            return

         endif

         m=m+1

      else if(diff.gt.0.) then

C     GET HERE IF X.GT.XT(M)

2        IF(M.EQ.N) then
```



```
            do 20 i=1,11
20          y(i)=yt(m,i)

            return

         endif

         MP1=M+1

         IF(XT(MP1).LT.XT(M)) then

            do 30 i=1,11
30          y(i)=yt(m,i)
            return

         endif

         M=MP1

         IF(X.gt.XT(M)) then

            go to 2

         else if(x.eq.xt(m)) then

            do 40 i=1,11
40          y(i)=yt(m,i)

            return

         endif

      else if(diff.eq.0.) then

         do 50 i=1,11
50       y(i)=yt(m,i)

         RETURN

      endif

      MM1=M-1

      DEL=(X-XT(MM1))/(XT(M)-XT(MM1))

      do 60 i=1,11
60    y(i)=yt(mm1,i)+del*(yt(m,i)-yt(mm1,i))

      RETURN

      END
```

3.14 Subroutine ANEWCH

3.14.1 Purpose

The purpose of this subroutine is to evaluate the optimization and restoration equations to determine the required changes in the controlling parameters and variables. Tests for convergence are also made in this subroutine.

3.14.2 Variable Listing

3.14.3 Subroutines Called:

Name	Function
MATINV	Calculates matrix inverse

3.14.4 Calling Subroutines:

Name	Function
MASTRE	Controls total program logic

3.14.5 Fortran Listing

anewch

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
aa	Launch azimuth	rad	ainit	output input	Global	agen2
ad	Launch azimuth	deg			Local	
aerod(30,17)	Aerodynamic coefficient tables		ainit	output input	Global	aerodi
ah	Previous value of pitch attitude angle	deg			Local	
ahtb(210)	Previous pitch attitude tables	rad	bsaveg	input	Global	imp
betcon	Logical value to indicate parameter convergence				Local	
case	Case number		ainit	output input	Global	agen2
cfr(15)	Critical flowrate	kg/sec	ainit	input	Global	xtra
convk(15,3)	Conversion criteria			output input	Global	convk
convp	Convergence factor for parameters				Local	
cp	Current value of pitch attitude	deg			Local	
cpotbl(210)	Pitch attitude table during min-H	rad	anewch	output input	Global	contro
cptbl(30)	Booster pitch attitude angle table	rad	anewch	output input	Global	contro
cy	Current value of yaw attitude	deg			Local	
cyotbl(210)	Yaw attitude table during min-H	rad	anewch	output input	Global	contro
cytbl(30)	Booster yaw attitude angle table	rad	anewch	output input	Global	contro
dbeta	Calculated change in control parameters	variabl			Local	
delcp	Increment for pitch attitude angle	rad			Local	
delcy	Increment for yaw attitude angle	rad			Local	
delts(7)	Time increment used to build pitch and yaw attitude tables during min-H	sec	bsaveg	input	Global	imp
demand	Logical indicating the operational mode		initial	input	Global	ccc
dep	Change in pitch attitude betwen iterations	deg			Local	
dey	Change in yaw attitude betwen iterations	deg			Local	
dp2	Constraint scale factor		ainit	input	Global	auto
drho(19)	Difference between current and desired constraint values		anewch	output input	Global	drho

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
dtb4	Stepsize for updating time when incrementing the attitude control tables during the minh phase	sec			Local	
e1	Fraction of optimization step to be made when updating optimization variables		anewch	input output	Global	drho
ftbl(15)	Thrust table for main proplusion system	lbs	ainit	input output	Global	mps
gaptb(210,20	Partial derivatives of constraints with respect to pitch attitude		bsaveg	input	Global	imp
gnu(20)	Partial derivative of payoff with respect to constraints		anewch	output input	Global	gnu
gsptb(210,20	Partial derivatives of constraints with respect to yaw attitude		bsaveg	input	Global	imp
humax	Attitude angle increment maximumvalues				Local	
ii	Index				Local	
ijwc	Index				Local	
ik	Index				Local	
im	Index				Local	
inmax	Temporary variable for nmax			input output	Global	Iprint
ipr	Denotes the thrust event from which fuel propellant reserve (FPR) is calculated			input output	Global	ipr
ja	Number of constraints				Local	
jj	Index				Local	
jm	Index				Local	
jtb	Index for control table points		anewch	output input	Global	bgen3
jump	Jump start flag		ainit	input	Global	agen3
kat	Indicator for positive definite values of huu matrix		bsaveg	input	Global	agen3
kcdphi(20)	Terminal constraint selection flag array		ainit	input	Global	yli
kdb(40)	Array for control parameter flags		ainit	input	Global	auto
kdt(15)	Array used in confunction with the ipr option		ainit	input	Global	ipr
kjtb	Index for time and pitch and yaw control table index		anewch	output input	Global	imp
kk	Number of constriants minus one				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
kv	Index				Local	
kwta(7)	Flag indicating pitch only (1) or pitch and yaw (2) attitude control		ainit	input	Global	agen3
l3	Aerodynamic coefficient tables				Local	
lp	Index				Local	
lprint	Print flag			output	Global	lprint
lsb	Internal flag to indicate partial derivative runs			output	Global	rest
msw	Temporay variable		athrev	input	Global	fmxx
nbeta	Index used for booster sideslip attitude control		ainit	input	Global	taulim
nbgct(7)	Index indicating beginning of min-H phases		ainit	input	Global	agen3
ncnrs(5)	Number of constraints at restart		ainit	input	Global	rest
nmax	Iteration counter		mastre	output	Global	agen3
nom1	Index indicating first iteration pass			input output	Global	bgen3
nom2	Flag indicating current position in optimization steps				Local	
nopar	Number of optimization parameters			input output	Global	rest
noss	Number of constraints		mastre	input output	Global	bgen3
np(7)	Number of points in attitude tables during min-H phases		ainit	input	Global	bgen3
nstage	Index number of current stage		athrev	output input	Global	mult
nstg(15)	Internal index which relates thrust event to stage number		ainit	input	Global	mult
ntau	Index used in conjunction with timmps to indicate a throttle event in the booster stage			input output	Global	taulim
ntcn	Number of intermediate constraints		ainit	input	Global	rest
ntinv	Index to denote when first stage throttle constraint time goes negative				Local	
ntnv	Index to denote when throttle value goes below limit for first stage throttle optimization				Local	
nv	Index				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
par	Temporary variable				Local	
pfkg	Conversion for pounds to kilograms	kg/lbs	ainit	input	Global	const
phite(20)	Values of payoff and constraints	variabl	afornd	input	Global	yli
predp	Predicted value of payoff function	variabl	anewch	output input	Global	lprint
pstr(20)	Difference between current and desired constraint values	variabl	afornd	input	Global	yli
qy	Optimization step size control constant		ainit	input	Global	lprint
si	Previous value of yaw attitude angle	deg			Local	
sitb(210)	Previous yaw attitude table during min-H		bsaveg	input	Global	imp
spoa	Specific impulse divided by acceleration limit		athrev	input	Global	fmxx
srmprp	SRM propellant weight	kg			Local	
swibt	Weighting array used in normalizing the partial derivatives				Local	
t	Time from lift-off	sec	dpir	output input	Global	forint
taut(15)	Time increments for thrust events	sec	anewch	input output	Global	agen1
tbias(6)	Bias between simulated time and reported time (used for branch option)	sec	athrev	input	Global	mult
temp	Temporary variable				Local	
temp1	Temporary variable				Local	
timmps(15)	Time table for MPS motor	sec	ainit	input output	Global	mps
timtau	Time for throttle event in first stage	sec	ainit	input output	Global	taulim
tne(6,15)	Array which defines the number of engines per thrust event		ainit	input	Global	agen1
tobl(210)	Independent time table used for attitude tables during min-H		anewch	output	Global	contro
tol	Convergence tolerance on scaled total derivatives		ainit	input	Global	lprint
tr(5)	Intermediate constraint absolute time	sec	anewch	output	Global	rest
ts(7)	Start times to begin min-H phases		bsaveg	input	Global	imp
ttol	Tolerance on steps in taut				Local	
waps	First column of matrix wiss (l-phi-psi for control variables)				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
wass	I-psi-psi for control variables				Local	
wd(15)	Weight dropped during weight drop event	kg	ainit	input	Global	agen1
wds	Matrix product of constraint dispersions and weighting				Local	
wg	Mass at acceleration limit	kg	athrev	input	Global	fmxx
wgnu	Change in optimization parameter with respect to control parameters		anewch	output input	Global	drho
wibt(15)	Dynamic parameter weight factor		anewch	input output	Global	rest
wipp	Change in optimization (payoff) parameter as a function of control variables				Local	
wippb	Change in optimization parameter with respect to control variables	variabl	anewch	output input	Global	drho
wipsb	Total I-phi-psi vector				Local	
wiss(20,20)	Weighting matrix		bsaveg	input	Global	bake
wissb	Total I-psi-psi vector				Local	
wjet(15)	Jettison weight per thrust event	kg	ainit	input	Global	agen1
wo1	Liftoff weight	lbs			Local	
wpmx	Total amount of usable propellant	kg	athrev	input	Global	ipr
w y	Temporary variable				Local	
wzero	Initial vehicle weight	lbs	anewch	output input	Global	agen2
w_margin(2)	Required value of weight margin at the end of selected stage	kg	anewch	output input	Global	multi
x	X component of plumbline position vector	m	dpir	input	Global	forint
xjext	Flag indicating that payoff will be maximized (=1) or minimized (=-1)		ainit	input	Global	agen2
xkay	Decimal fraction of constraint error to remove				Local	
xlamb(40,20)	Partial derivatives of payoff and constraints with respect to the optimization parameters		bkend	input	Global	auto
xlbgnu	Matrix product				Local	
xlbwds	Matrix product				Local	
xmd(15)	Propellant flowrate	kg/sec	ainit	input	Global	xtra

```
      SUBROUTINE ANEWCH                                                 ANEW   1
C
C      DETERMINES THE STEPSIZE AND EVALUATES THE REQUIRED CHANGES IN
C     THE PARAMETERS.CONVERGENCE TESTS ARE MADE FOR ITERATION TERMINATION
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      LOGICAL DEMAND,GRAPH,GLIMIT,QFLG
C
      CHARACTER*60 HEAD
      CHARACTER*6 MISION

      COMMON/CCC/GRAPH,JO,DEMAND

      COMMON/AERODI/AEROD(30,17),AEROB(50,6,3),TOMACH(25),DELCA(25),
     *PNMONE(15),BKFABT(50,2),BKFTHR(50,2),DELCN(25),DELCM(25)

      COMMON/AGEN1/TIME(2,16),TAUT(15),TAUW(15),TZERO,TLIFT,TTILT,TMINH,
     *DTZ,TQ,TL,XMAUG,TNE(6,15),ENGDAT(8,15),S(15),WD(15),WJET(15),
     *HEAD,PRINT(15),STEP(15),HNOM(15),HMAX(15),PROP(6),TBEGR,TENDR,
     *CHRDOT,CHIRO,FAZ,BO(3,2),CBAXIL,TCHIR,VRCUT,FPRFAC,CHVEL

      COMMON/AGEN2/AA,SA,CA,ALF,ALFY,CHIP,SCHIP,CCHIP,CHIY,SCHIY,CCHIY,
     *CHIR,SCHIR,CCHIR,STH,CTH,STHL,CTHL,DPHIZ,RTHE,R,VR,XM,SW,SU,SV,Q,
     *VIV(25),DVARS(13),DELXDW(15,3),DELXDR(7,5),DELXD(15,7),WZERO,ALT,
     *XMACH,QDOT,FAA,FAN,A(3,3),CASE,CT2,UMF,A12W,A22W,A32W,XJEXT,BEU,
     *BYL,BEL,BHMAX(15),BHMN,BSTEP(15),HNMB,HMXB,TPOLY,XMIAD

      COMMON/AGEN3/ITHR,IWD,IXR,JUMP,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISION,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)

      COMMON/AUTO/KDB(40),SAVCP(40),XLAMB(40,20),DP2

      COMMON/BAKE/WISS(20,20),YLF(7,20)

      COMMON/BGEN3/NOSS,JTB,NENT,NACT,NOM1,NP(7)

      COMMON/BLAMB/YLBT(15,20),YLBW(15,20),GAPHI(20),PROD(2,20)

      COMMON/BODY/TBDWT(15,2),TXCG(15,2),TYCG(15,2),XLEN(2),XREF(2),
     *YREF(2),KP(2),KY(2),AP(10,2),AY(10,2)

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM

      COMMON/CONTRO/TTBL(30),CPTBL(30),CYTBL(30),TOBL(210),
     *CPOTBL(210),CYOTBL(210)

      COMMON/CONVK/CONVK(15,3)

      COMMON/DRHO/DRHO(19),E1,WIPPB,WGNU

      COMMON/FMXX/FMXX,MSW,WG,SPOA
```



```
      COMMON/FORINT/HBANK(2),T,TT,W,U,V,X,Y,Z,XMI,ZAP(18),DVAR(25),FSAVE
     *(625),NTRG1,TV1,NTRG2,TV2,NTRG3,TV3,NTRG4,TV4,NTRG5,TV5,NTRG6,TV6,
     *NTRG7,TV7,NTRG8,TV8,NTRG9,TV9,NTRG10,TV10,NTRG11,TV11,NTRG12,TV12,
     *NTRG13,TV13,KIND,KRDER,EU,EL,AYL,HMX,HMN,HNM

      COMMON/GNU/GNU(20)

      COMMON/IMP/GAPTB(210,20),GSPTB(210,20),AHTB(210),SITB(210),TS(7)
     *,DELTS(7),KJTB

      COMMON/IPR/IPR,WPMX,ILAST,KDT(15),PRK4,PRTOT,iprop(6),prk5(5)

      COMMON/LPRINT/LPRINT,INMAX,PREDP,QY,TOL

      COMMON/LBM/TIMLBM(30),THRLBM(30),WDTLBM(30),PLBM(2),MLBM1,MLBM2,
     *ALBM(2),BLBM(2),CLBM(2),VLBM(2)

      COMMON/MPS/TIMMPS(15),FTBL(15),THROT(15),GLIM(15),TAUALT(5),
     *TAUTBL(5),MPS1,MPS2,TGLIM(10,15),GLIMIT,tau_const(10,5),jthrot(15)

      common/multi/char_vel(5),fpr_fac(5),w_margin(2)

      common/mult/nchiot(6),tbias(6),nstage,nstg(15),wf(5),ibranch

      COMMON/OMSRCS/FOMS(15),FRCS(15),WDOMS(15),WDRCS(15),WDDUMP(15)

      COMMON/REST/JSTR,NCNRS(5),NTCN,TR(5),LSB,NF8,NOPAR,WIBT(15)

      COMMON/TAULIM/TAULIM,NTAU,NBETA,TIMTAU

      COMMON/XTRA/XMD(15),CFR(15),F(15),CISP(15)

      COMMON/YLI/YLINT(7,20),PHITE(20),PSIREQ(20),KCDPHI(20),PSTR(20)

      DIMENSION SWIBT(17),DBETA(17),WAPS(19),WASS(20,20),WDS(19)

      DIMENSION HUMAX(2),WIPSB(19),WISSB(19,19),XLBGNU(19),XLBWDS(19)

      LOGICAL BETCON                                                    ANEW  41

C*****                                                                  ANEW  44
C     BEGIN MAIN NEWCH                                                  ANEW  45
C*****                                                                  ANEW  46

   88 XKAY=1.                                                           ANEW  47

      LP=0
      IF(LPRINT.eq.1) then
         LP=1
         LPRINT=2
      endif

      CONVP=1.-.5*E1                                                    ANEW  48
      CASE=CASE+.0001                                                   ANEW  49
      BETCON=.TRUE.                                                     ANEW  50
      LSB=0                                                             ANEW  51
```

```
      NV=1                                                              ANEW  76

        do i=1,40

           IF(KDB(I).eq.1) then

              DO J=1,NOSS
                 XLAMB(NV,J)=XLAMB(I,J)
              enddo

           NV=NV+1

           endif
        enddo

        WRITE(JO,'(/,a)') ' INFLUENCE COEFFICIENTS'

        DO I=1,NOPAR
           WRITE(JO,'(1x,8e15.8)') (XLAMB(I,J),J=1,NOSS)
        enddo

C*****                                                                  ANEW  89
C     CALCULATE I-SY-SY MATRIX                                          ANEW  90

   71   KK=NOSS-1
        WIPP=WISS(1,1)

        DO I=1,19
           WAPS(I)=WISS(I+1,1)
        enddo

        DO I=2,20
           DO J=2,I
              IM=I-1
              JM=J-1
              WASS(IM,JM)=WISS(I,J)
              WASS(JM,IM)=WASS(IM,JM)
           enddo
        enddo

      IF(LPRINT.eq.2) then

         WRITE(JO,*) '     I PSI PSI'

         WRITE(JO,'(1x,e15.8)') WIPP

         DO I=1,KK
            WRITE(JO,'(1x,8e15.8)')WAPS(I),(WASS(I,J),J=1,KK)
         enddo

      endif

C*****GET WIBT                                                          ANEW 108
C*****                                                                  ANEW 109

      DO I=1,NOPAR
```



```
         SWIBT(I)=0.

         DO J=2,NOSS

            TEMP=0.
            IF(ABS(WASS(J-1,J-1)).GT.0.) TEMP=XLAMB(I,J)*XLAMB(I,J)/
     *      WASS(J-1,J-1)
            SWIBT(I)=DMAX1(SWIBT(I),TEMP)
         enddo

         SWIBT(I)=WIBT(I)/SWIBT(I)

      enddo
C*****                                                                  ANEW 116

      IF(NOPAR.EQ.0)SWIBT(1)=0.                                         ANEW 117

C*****GOT WIBT                                                          ANEW 118

      IF(LPRINT.eq.2) then

         WRITE(JO,'(/,a)') ' PSUBI'

   15    WRITE(JO,'(1x,8e15.8)') (SWIBT(I),I=1,NOPAR)

         WRITE(JO,'(/,a)') ' WIBT'

         WRITE(JO,'(1x,8e15.8)') (WIBT(I),I=1,NOPAR)

      endif

      WIPPB=WIPP                                                        ANEW 126

      DO I=1,NOPAR
         WIPPB=WIPPB+XLAMB(I,1)*SWIBT(I)*XLAMB(I,1)
      enddo

      DO I=1,KK
         WIPSB(I)=WAPS(I)
         DO J=1,NOPAR
            WIPSB(I)=WIPSB(I)+XLAMB(J,1)*SWIBT(J)*XLAMB(J,I+1)
         enddo
      enddo

      DO I=1,KK
         DO J=1,KK
            WISSB(I,J)=WASS(I,J)
            DO K=1,NOPAR
               WISSB(I,J)=WISSB(I,J)+XLAMB(K,I+1)*SWIBT(K)*XLAMB(K,J+1)
            enddo
         enddo
      enddo

C     PRINT TOTAL I MATRIX                                              ANEW 138
```

```
      IF(LPRINT.eq.2) then

         WRITE(JO,*) '    TOTAL I SY SY MATRIX'

         WRITE(JO,'(1x,e15.8)') WIPPB

         DO I=1,KK
            WRITE(JO,'(1x,8e15.8)') WIPSB(I),(WISSB(I,J),J=1,KK)
         enddo

      endif

      IF(KK.eq.1) then

         WISSB(1,1)=1./WISSB(1,1)

      else

         CALL MATINV(WISSB(1,1),KK)

         DO I=1,KK
            DO J=1,I
               WISSB(I,J)=(WISSB(I,J)+WISSB(J,I))/2.
               WISSB(J,I)=WISSB(I,J)
            enddo
         enddo

      endif                                                    ANEW 153
C*****                                                         ANEW 154
C     CALCULATE MATRIX PRODUCTS                                ANEW 155
C*****

      DO I=1,KK
         DRHO(I)=-PSTR(I+1)
      enddo

      DO I=1,KK
         WDS(I)=0.
         DO J=1,KK
            WDS(I)=WDS(I)+WISSB(I,J)*DRHO(J)
         enddo
      enddo

      DO I=1,KK
         GNU(I)=0.
         DO J=1,KK
            GNU(I)=GNU(I)-WISSB(I,J)*WIPSB(J)
         enddo
      enddo

      WGNU=0.                                                  ANEW 166
      DO I=1,KK
         WGNU=WGNU-GNU(I)*WIPSB(I)
      enddo

      IF(LPRINT.eq.2) then
```



```
         WRITE(JO,'(a,e15.8)') ' WGNU',WGNU

      endif

C*****                                                         ANEW 172
C     BEGIN E1 LOGIC                                           ANEW 173
C*****                                                         ANEW 174

      NOM2=NOM1+1

      if(nom2.eq.1) then

         e1=0.
         xkay=dmax1(dp2,.25d0)

         if(xkay.ge..4999.and.xkay.le..9999) then

            nom1=1
            xkay=.5

         else if(xkay.gt..9999) then

            nom1=2

         endif

      endif

      if(nom2.eq.2) then

         e1=0.
         xkay=.5

      endif

      if(nom2.eq.3) e1=0.

      if(nom2.eq.4) e1=qy/2.

      if(nom2.eq.5) e1=qy

      if(nom2.le.2.and.xkay.le..5) then

         do i=1,kk
            drho(i)=drho(i)*xkay
            wds(i)=xkay*wds(i)
         enddo

      endif

C*****                                                         ANEW 183
C     CALCULATE DBETA                                          ANEW 184
C*****                                                         ANEW 185

      IF(NOPAR.EQ.0) GO TO 50
```

```
      DO I=1,NOPAR
         CONVK(I,3)=ABS(XLAMB(I,1))
         XLBGNU(I)=0.
         XLBWDS(I)=0.
         DO J=1,KK
            TEMP=XLAMB(I,J+1)*GNU(J)
            XLBGNU(I)=XLBGNU(I)+TEMP
            XLBWDS(I)=XLBWDS(I)+XLAMB(I,J+1)*WDS(J)
            TEMP=TEMP+XJEXT*XLBWDS(I)
            CONVK(I,3)=DMAX1(CONVK(I,3),ABS(TEMP))
         enddo
      enddo

      IF(LPRINT.eq.2) then

         WRITE(JO,'(a,e15.8,a,e15.8)')' xkay',XKAY,' E1',E1

         WRITE(JO,*) ' DRHO,WDS,GNU'

         DO I=1,KK
            WRITE(JO,'(1x,8e15.8)')DRHO(I),WDS(I),GNU(I)
         enddo

      endif

      DO I=1,NOPAR
         TEMP=XLAMB(I,1)+XLBGNU(I)
         CONVK(I,2)=TEMP/CONVK(I,3)
         DBETA(I)=SWIBT(I)*(TEMP*E1*XJEXT+XLBWDS(I))
      enddo

      WRITE(JO,'(/,7x,a,9x,a,7x,a,8x,a)') 'DPAR','OLD PCON','NEW PCON',
     *'P SCALE'

      IK=0

      DO 1002 I=1,NOPAR

      IF(ABS(CONVK(I,2)).GT..1)THEN

         IF(BETCON)THEN

            WRITE(31,*)'GOING FALSE ON CONVK I ',I,CONVK(I,2)
            BETCON=.FALSE.

            ENDIF

         ENDIF

      WRITE(31,*)'ANEWCH I BETCON ',I,BETCON

      IF(.not.BETCON) then

         IK=IK+1
```



```
         IF(IK.le.1.and.ABS(CONVK(I,2)*CONVK(I,3)).le.1.) then

            BETCON=.TRUE.
            IK=0

         endif

      endif

      WRITE(JO,'(1x,8e15.8)') DBETA(I),(CONVK(I,J),J=1,3)

1002  continue

C*****
C     ADD DBETA TO BETA
C*****

      IF(LP.EQ.1) LPRINT=1

      NV=1

      DO 45 I=1,40

      IF(KDB(I).EQ.0)GO TO 45

      IF(i.le.15) then

c        TTOL=1.*TAUT(I)

         ttol=.1*taut(i)

c        IF(TTOL.GT.3.) TTOL=3.

         IF(ABS(DBETA(NV)).GT.TTOL) DBETA(NV)=SIGN(TTOL,DBETA(NV))

         TAUT(I )=TAUT(I )+DBETA(NV)
C                                                                    ANEW 222
C     IF(I.LT.NVNT)TAUT(I+1)=TAUT(I+1)-DBETA(NV)                     ANEW 223
C                                                                    ANEW 224
         IF(MSW.ne.1.and.KDT(I).ne.0) then

            J=I +KDT(I)
            TEMP=TNE(1,I)*CFR(I)/(TNE(1,J)*CFR(J))
            TAUT(J)=TAUT(J)-TEMP*DBETA(NV)
            IF(TAUT(J).LE..01)TAUT(J)=.01
         endif

         IF(ABS(DBETA(NV)/(TAUT(I)-DBETA(NV))).GT..0025) THEN

            IF(BETCON)THEN

               WRITE(31,*)'GOING FALSE ON TAUT DBETA ',TAUT(I),DBETA(NV)
               BETCON=.FALSE.

            ENDIF
```

```
      ENDIF
      IF(TAUT(I ).LE..01)TAUT(I  )=.01
      WRITE(JO,'(a,i2,a,e15.8)') ' TAUT(',I,')=',TAUT(I)
      IF(DEMAND) then
         WRITE(*,'(a,i2,a,e15.8)') ' TAUT(',I,')=',TAUT(I)
         IJWC=1
         WRITE(11)IJWC,I,TAUT(I)
      endif

   else if(i.eq.16) then
      WZERO=WZERO+DBETA(NV)
      IF(ABS(DBETA(NV)).GT.100.)THEN
         IF(BETCON)THEN
            WRITE(31,*)'going false on wzero dbeta ',wzero,dbeta(nv)
            BETCON=.FALSE.
         endif
      endif
      WO1=WZERO/.45359237
      WRITE(JO,'(a,e15.8)') ' WO1=',WO1
      IF(DEMAND) then
         IJWC=2
         WRITE(*,'(a,e15.8)') ' WO1=',WO1
         WRITE(11)IJWC,IJWC,WO1
      endif

   else if(i.eq.17) then
      AA=AA+DBETA(NV)/RAD
      AD=AA*RAD
      IF(ABS(DBETA(NV)/(AD-DBETA(NV))).GT..005) then
         if(betcon)then
            WRITE(31,*)'GOING FALSE ON AD DBETA ',AD,DBETA(NV)
            BETCON=.FALSE.
```



```
         ENDIF
      ENDIF
      WRITE(JO,'(a,e15.8)') ' AZ=',AD
      IF(DEMAND) then
         IJWC=3
         WRITE(*,'(a,e15.8)') ' AZ=',AD
         WRITE(11)IJWC,IJWC,AD
      endif
   else if(i.ge.18.and.i.le.24) then
      JJ=I-16
      WRITE(31,*)'AFTER 143 BETCON ',BETCON
      IF(ABS(DBETA(NV)).GT.2.D0) DBETA(NV)=SIGN(2.D0,DBETA(NV))
      CPTBL(JJ)=CPTBL(JJ)+DBETA(NV)
      IF(ABS(DBETA(NV)/(CPTBL(JJ)-DBETA(NV))).GT..01)THEN
         IF(BETCON)THEN
            WRITE(31,*)'FALSE ON CPTBL DBETA ',CPTBL(JJ),DBETA(NV)
            BETCON=.FALSE.
          ENDIF
      ENDIF
      WRITE(31,*)'CPTBL ',BETCON
      WRITE(JO,'(a,i2,a,e15.8)') ' CPTBL(',JJ,')=',CPTBL(JJ)
      IF(DEMAND)THEN
         WRITE(*,'(a,i2,a,e15.8)') ' CPTBL(',JJ,')=',CPTBL(JJ)
         IJWC=4
         WRITE(11)IJWC,JJ,CPTBL(JJ)
      ENDIF
   else if(i.ge.25.and.i.le.28) then
      JJ=I-23
      IF(ABS(DBETA(NV)).GT.2.D0) DBETA(NV)=SIGN(2.D0,DBETA(NV))
```

```
      CYTBL(JJ)=CYTBL(JJ)+DBETA(NV)

      IF(ABS(DBETA(NV)).GT..05)THEN

C     IF(ABS(DBETA(NV)/(CYTBL(JJ)-DBETA(NV))).GT..2) THEN

         IF(BETCON)THEN

            WRITE(31,*)'FALSE ON CYTBL DBETA ',CYTBL(JJ),DBETA(NV)

            BETCON=.FALSE.

         ENDIF

      ENDIF

      WRITE(31,*)'CYTBL ',BETCON

      WRITE(JO,'(a,i2,a,e15.8)') ' CYTBL(',JJ,')=',CYTBL(JJ)

      IF(DEMAND) then

         WRITE(*,'(a,i2,a,e15.8)') ' CYTBL(',JJ,')=',CYTBL(JJ)

         IJWC=5

         WRITE(11)IJWC,JJ,CYTBL(JJ)

      endif

   else if(i.ge.29.and.i.le.31) then

      JJ=I-28

      IF(ABS(DBETA(NV)).GT.5.D0) DBETA(NV)=SIGN(5.D0,DBETA(NV))

      if(jj.eq.1) then

         PAR=TIMMPS(NTAU)
         TIMMPS(NTAU)=PAR+DBETA(NV)
         TEMP=TIMMPS(NTAU)

      else if(jj.eq.2) then

         PAR=TIMTAU
         TIMTAU=PAR+DBETA(NV)
         NTINV=0
         IF(TIMTAU.LT.0.) then
            NTINV=NV
            TIMTAU=.01
         endif
         TEMP=TIMTAU

      else
```



```
         PAR=FTBL(NTAU+1)*100.
         FTBL(NTAU+1)=(PAR+DBETA(NV))/100.
         NTNV=0

         IF(FTBL(NTAU+1).LT..65) then
            NTNV=NV
            FTBL(NTAU+1)=.65
         endif

        TEMP=FTBL(NTAU+1)

      endif

      IF(ABS(DBETA(NV)/PAR).GT..01) BETCON=.FALSE.

      WRITE(JO,'(a,i2,a,e15.8)') ' THROTTLE PAR(',JJ,')=',TEMP

      IF(DEMAND) then

         WRITE(*,'(a,i2,a,e15.8)') ' THROTTLE PAR(',JJ,')=',TEMP

         IJWC=6

         WRITE(11) IJWC,JJ,TEMP

      endif

   else if(i.ge.32.and.i.le.34) then

      JJ=I-31
      L=NBETA+JJ-1
      AEROD(L,15)=AEROD(L,15)+DBETA(NV)/RAD

      IF(ABS(DBETA(NV)/(AEROD(L,15)*RAD-DBETA(NV))).GT..01)
     *BETCON=.FALSE.

      WRITE(31,*)'AEROD ',BETCON

      PAR=AEROD(L,15)*RAD

      WRITE(JO,'(a,i2,a,e15.8)') ' SIDESLIP PAR(',JJ,')=',PAR

      IF(DEMAND) then

         WRITE(*,'(a,i2,a,e15.8)') ' SIDESLIP PAR(',JJ,')=',PAR

         IJWC=7

         WRITE(11) IJWC,JJ,PAR

      endif

   else if(i.eq.35.or.i.eq.36) then

      if(i.eq.36.and.abs(dbeta(nv)).gt.4000.) then
         dbeta(nv)=sign(1.d0,dbeta(nv))*4000.
```

```
        endif

        l=1
        if(i.eq.36) l=2

        w_margin(l)=w_margin(l)+dbeta(nv)

        write(jo,'(1x,a,i1,a,e15.8)') 'margin(',l,')=',
     *  w_margin(l)*pfkg

        if(demand) then

           write(*,'(1x,a,i1,a,e15.8)') 'margin(',l,')=',
     *     w_margin(l)*pfkg

           ijwc=8

           write(11) ijwc,l,w_margin(l)*pfkg

        endif

      endif

      NV=NV+1

   45 CONTINUE

      IJWC=6

      IF(DEMAND) WRITE(11)IJWC,IJWC,IJWC

      IF(MSW.ne.0) then

         WY=0.
         K=IPR-1
         TEMP=WZERO

         DO 200 I=JUMP,K
         TEMP=TEMP-TNE(1,I)*XMD(I)*TAUT(I)-WJET(I)-WD(I)
         WY=WY+TNE(1,I)*CFR(I)*TAUT(I)
  200    CONTINUE

         TEMP=TEMP-SRMPRP
         WY=WY+TNE(1,IPR)*CFR(IPR)*(TEMP-WG)/(TNE(1,IPR)*XMD(IPR))
         NV=1

         DO 202 I=1,15

         IF(KDB(I).ne.0) then
            IF(KDT(I).ne.0) then

               TEMP=TNE(1,I)*XMD(I)/(TNE(1,IPR)*XMD(IPR))-SPOA*
     *              (TNE(1,I)*CFR(I)-CFR(IPR)/XMD(IPR)*TNE(1,I)*XMD(I))
     *              /(WPMX-WY-CFR(IPR)/XMD(IPR)*WG)
               TAUT(IPR)=TAUT(IPR)-TEMP*DBETA(NV)
```



```
            endif

            NV=NV+1

         endif

  202    CONTINUE

      endif

C*****
C     CALCULATE CHANGE IN CHIP AND CHIY
C*****

   50 HUMAX(1)=0.
      HUMAX(2)=0.
      KV=1

      WRITE(JO,95)
   95 FORMAT(/,7X,4HTIME,9X,6HNEW CP,9X,6HOLD CP,8X,6HNEW CY,8X,
     1 6HOLD CY,10X,3HDCP,10X,3HDCY)

 2020 DO 60 II=1,7

      DELCP=0.
      DELCY=0.

      IF(NP(KV).ne.0) then

         KJTB=(KV-1)*30
         T=TS(KV)
         DTB4=DELTS(KV)
         JTB=NP(KV)
         JA=NOSS
         nstage=nstg(nbgct(kv))

         IF(T+.1.GT.TR(1).AND.TR(1).GT.0.) JA=NOSS-NCNRS(1)
         IF(T+.1.GT.TR(2).AND.TR(2).GT.0.) JA=NOSS-NCNRS(1)-NCNRS(2)
         IF(T+.1.GT.TR(3).AND.TR(3).GT.0.) JA=NOSS-NCNRS(1)-NCNRS(2)-
     *   NCNRS(3)
         IF(T+.1.GT.TR(4).AND.TR(4).GT.0.) JA=NOSS-NCNRS(1)-NCNRS(2)-
     *   NCNRS(3)-ncnrs(4)
         IF(T+.1.GT.TR(5).AND.TR(5).GT.0.) JA=NOSS-NTCN
         SRMPRP=0.

         DO 46 I=1,JTB

         JJ=KJTB+I
         TEMP=0.
         TEMP1=0.

         DO 47 J=2,JA

         TEMP=TEMP+GAPTB(JJ,J)*GNU(J-1)
   47    TEMP1=TEMP1+GAPTB(JJ,J)*WDS(J-1)
```

```
         DELCP=(GAPTB(JJ,1)+TEMP)*XJEXT
         HUMAX(1)=DMAX1(ABS(DELCP+TEMP1),HUMAX(1))
         DELCP=DELCP*E1+TEMP1

C***  END OF CHIP

         IF(KWTA(KV).ne.2) then

            TEMP=0.
            TEMP1=0.

            DO 48 J=2,JA

            TEMP=TEMP+GSPTB(JJ,J)*GNU(J-1)
   48       TEMP1=TEMP1+GSPTB(JJ,J)*WDS(J-1)

            DELCY=(GSPTB(JJ,1)+TEMP)*XJEXT
            HUMAX(2)=DMAX1(ABS(DELCY+TEMP1),HUMAX(2))
            DELCY=DELCY*E1+TEMP1

         endif

C***  END OF CHIY
C     UPDATE TTBL,CPTBL,CYTBL

         TOBL(JJ)=T-tbias(nstage)
         CPOTBL(JJ)=AHTB(JJ)+DELCP
         CYOTBL(JJ)=SITB(JJ)+DELCY

         IF(CYOTBL(JJ).GT.1.5) CYOTBL(JJ)=(1.5+SITB(JJ))/2.

         IF(CYOTBL(JJ).LT.-1.5) CYOTBL(JJ)=-(1.5-SITB(JJ))/2.

         IF(LPRINT.eq.2.or.NMAX.eq.INMAX.or.(I.eq.1.or.I.eq.JTB)) then

            CP=CPOTBL(JJ)*RAD
            AH=AHTB(JJ)*RAD
            CY=CYOTBL(JJ)*RAD
            SI=SITB(JJ)*RAD
            DEP=DELCP*RAD
            DEY=DELCY*RAD

            IF(DEMAND) WRITE(12) T-tbias(nstage),CP,AH,CY,SI,DEP,
     *       DEY,JJ

            WRITE(JO,'(7e14.7,i3)') T-tbias(nstage),CP,AH,CY,SI,DEP,
     *       DEY,JJ

         endif

   46    T=T+DTB4

      endif

      KV=KV+1
```



```
   60 CONTINUE

       WRITE(31,*)'BOTTOM OF ANEWCH NOPAR ',NOPAR

c      DO 6969 KKK=1,4

C       WRITE(31,*)'KKK PHITE PSIREQ ',KKK,PHITE(KKK),PSIREQ(KKK)
C       TSTDEL=PHITE(KKK+1)/PSIREQ(KKK)
C       WRITE(31,*)'TSTDEL ',TSTDEL

c      IF(abs(PHITE(KKK+1)).GT.4.0)THEN

c         IF(BETCON)THEN

c           KKP1=KKK+1

c           WRITE(31,*)'FALSE ON PHITE ',KKP1,PHITE(KKP1)

c           BETCON=.FALSE.

c         ENDIF

c     ENDIF

c6969 CONTINUE

C*****RESTORE NOSS FOR FORWARD RUN

      NOSS=NOSS-NTCN
      NOM1=NOM1+1

      IF(NOM1.LT.5) BETCON=.FALSE.

      IF(IPR.NE.0.AND.NOM1.LE.6) BETCON=.FALSE.

        do i=2,noss

           if(kcdphi(i).eq.2.and.abs(phite(i)).gt.1.) then

              if(betcon) then
                 write(jo,*) ' velocity constraint gt tolerance'
                 betcon=.false.
              endif
           endif
        enddo

        IF(HUMAX(1).LT..005.AND.HUMAX(2).LT..005.AND.BETCON) then
           NMAX=1
        else
           betcon=.false.
        endif

      IF(DEMAND) then

         WRITE(13)HUMAX(1),HUMAX(2),BETCON,KAT
         WRITE(*,61) HUMAX(1),HUMAX(2),BETCON,KAT
```

```
      endif

      WRITE(JO,61)HUMAX(1),HUMAX(2),BETCON,KAT

   61  FORMAT(1X,13HDEL CHIP MAX=,E11.4,16H   DEL CHIY MAX=,E11.4,
     1 10H   BETCON=,L3,6H  KAT=,I3)

C*****UPDATE CONVERGENCE SCALE FACTORS*****

      IF(NOM1.gt.5) then

         DO 63 I=1,NOPAR

         IF(ABS(CONVK(I,2)).ge.TOL.and.ABS(CONVK(I,2)).ge.CONVP*
     *   ABS(CONVK(I,1))) then

            IF(ABS(CONVK(I,2)-CONVK(I,1)).ge.ABS(CONVK(I,2))) then

               WIBT(I)=WIBT(I)/2.

            else

               WIBT(I)=WIBT(I)*2.

            endif

         endif

   63    CONTINUE

         IF(NTINV.NE.0) WIBT(NTINV)=WIBT(NTINV)/6.

         IF(NTNV.NE.0) WIBT(NTNV)=WIBT(NTNV)/6.

      endif

      DO 65 I=1,NOPAR
   65 CONVK(I,1)=CONVK(I,2)

      PREDP=0.

      DO 66 I=1,KK
      PREDP=PREDP+GNU(I)*DRHO(I)

      IF(NOM1.GT.2) GNU(I)=GNU(I)+XJEXT*WDS(I)

   66 CONTINUE

      JA=NOSS+NTCN

      RETURN
      END
```

3.15 Subroutine APRTN

3.15.1 Purpose

The purpose of this subroutine is to compute output parameters, print output data, and write output files for various output tables.

3.15.2 Variable Listing

3.15.3 Subroutines Called:

Name	Function
AMULG	Linear intepolation routine
TRNSFM	Multiplies 3x3 matrix times vector

3.15.4 Calling Subroutines:

Name	Function
AFORUN	Controls logic for forward trajectory and calls integration routine
ATHREV	Controls thrust event logic during forward trajectory
ATILT	Controls tiltover maneuver
AJUMP	Controls 1st stage jump start logic
ALIFT	Controls liftoff based on thrust-to-weight
AQMAX	Controls maximum dynamic pressure printout
ATHRO	Controls acceleration limit when using constant throttle
AWDEV	Controls weight drop printout
AXPRT	Controls printout based on print interval
GLIMT	Controls acceleration limit when using variable throttle

3.15.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a(3,3)	Tranformation matrix from equatorial to plumbline coordinate systems		aforun	input	Global	agen2
a1	Semi-major axis of current position	m			Local	
aa	Launch azimuth	rad	ainit	input	Global	agen2
aeronm	Name of aerodynamic data base				Local	
alf	Angle of attack	rad	ader	input	Global	agen2
alfy	Sideslip angle	rad	ader	input	Global	agen2
alongo	Launch longitude	rad	ainit	input	Global	alat
alt	Altitude	m	ader	input	Global	agen2
altp	Current altitude	m			Local	
amx	Aerodynamic moment about vehicle yaw axis	nt-m	ader	input	Global	agen5
amy	Aerodynamic moment about vehicle roll axis	nt-m	ader	input	Global	agen5
amz	Aerodynamic moment about vehicle pitch axis	nt-m	ader	input	Global	agen5
apogee	Magnitude of inertial velocity	m/sec		output input	Global	afprn
asto(4,67)	Storage array for weight summary table	lbs	ainit	input	Global	astor
at	Transpose of A matrix				Local	
atmosn	Name of atmospheric routine being used		atmosph	input	Global	ardc
azi	Inertial azimuth	rad	aprtn	output input	Global	afprn
azr	Wind relative azimuth	rad	aprtn	output input	Global	afprn
azre	Wind relative azimuth	rad	aprtn	output input	Global	afprn
azw	Wind azimuth	rad	ader	input	Global	agen5
bgenow	Eccentric anomaly of cuurent radius vector				Local	
bgethn	Eccentric anomaly of earth radius vector				Local	
c1	Angular momentum	m^2/se	aprtn	output input	Global	afprn
c3	Twice the energy	m^2/se	aprtn	output input	Global	afprn
ca	Cosine of launch azimuth		aforun	input	Global	agen2

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
case	Case number		ainit	output	Global	agen2
cbgenw	Cosine of eccentric anomaly of radius vector				Local	
cbgetn	Cosine of eccentric anomaly of earth radius vector				Local	
cchip	Cosine of pitch attitude angle		ader1	input	Global	agen2
cchir	Cosine of roll attitude angle		ader1	input	Global	agen2
cchiy	Cosine of yaw attitude angle		ader1	input	Global	agen2
cetanw	Cosine of Eccentric anomaly				Local	
cetatn	Cosine of eccentric anomaly of radius vector				Local	
chip	Pitch attitude angle	rad	ader1	output input	Global	agen2
chir	Roll attitude angle	rad	ader1	output input	Global	agen2
chiy	Yaw attitude angle	rad	ader1	output input	Global	agen2
cinc	Cosine of inclination angle				Local	
cmue	Gravitiational constant	m^3/se	ainit	input	Global	const
cphi	Cosine of longitude				Local	
cter1	Matrix of position, velocity, and aceleration in equatorial coordinate system				Local	
cter2	Temporary variable				Local	
cth	Cosine of colatitude		aprtn	output input	Global	agen2
cthl	Cosine of launch colatitude		ainit	input	Global	agen2
d	Direction cosine matrix				local	
ddwet	Second derivative of wet matrix				Local	
delbge	Delta eccentric anomaly				Local	
deleta	Increment between eccentric anomalies				Local	
dlphi	Delta inertial longitude between current position and impact point				Local	
dltl	Time bias				Local	
drag	Aerodynamic drag force	newton	ader	input	Global	aprint
dti	Delta time between current position and impact point	sec			Local	
dtx	Temporary variable				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
dtxp	Temporary variable				Local	
dty	Temporary variable				Local	
dtyp	Temporary variable				Local	
dtz	Time difference between GRR and launch time	sec	ainit	input	Global	agen1
dummy	Temporary variable				Local	
dvar(25)	Storage array for the state derivatives		ader	input output	Global	forint
dwet	First derivative of wet matrix				Local	
dx	Vertical component of thrust moment arm	m	ader	input	Global	garbge
dyi	1/dy	/ m		input	Global	garbge
dzi	1/dz	/ m	ader	input	Global	garbge
ecc	Eccentricity of current position				Local	
elein	Inboard elevon angle	rad	ader	input	Global	aprint
eleout	Outboard elevon angle	rad	ader	input	Global	aprint
etanw	Eccentric anomaly	rad			Local	
etatn	Eccentric anomaly of earth radius in projected orbit	rad			Local	
etirt	External tank inert weight	lbs			Local	
faa	Aerodynamic axial force	newton		input	Global	agen2
fabt(4)	Base axail force	newton	ader	input	Global	aprint
fan	Aerodynamic normal force	newton	ader	input	Global	agen2
fanc	Total normal force	newton	ader	input	Global	agen5
flatc	Total side force	newton		input	Global	agen5
fom	Total vehicle sensed acceleration	g's		output	Global	agen5
fomn	Output name				Local	
ftm	Feet to meter conversion				Local	
gam	Inertial flight path angle	rad	aprtn	output	Global	afprn
gamr	Flight path angle with respect to earth relative velocity	rad			Local	
gamre	Flight path angle with respect to earth relative velocity	rad			Local	
gzero	Gravitational constant (9.80665 m/sec)	m/sec^	ainit	input	Global	const
h	Third gravitational harmonic		ainit	input	Global	const
h2ht	Height of LH2 propellant				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
haero	Adjusted altitude for base force table interpolation	m		output	Global	ardc
hatm	Character string for output				Local	
hatmos	Altitude used for atmospheric lookup				Local	
hbias	Altitude bias	m	ainit	input	Global	ardc
head	Storage array for headings in printout		ainit	input	Global	agen1
headt	Not used				Local	
hoxt	Height of LOX propellant				Local	
hratec	Aerodynamic heating rate	btu/sec		output	Global	comnew
iatmos	Atmospheric routine indicator		ainit	input	Global	ardc
inmax	Temporary variable for nmax			input output	Global	lprint
is	Index				Local	
iss	Index				Local	
kpage	Page number index		aprtn	output input	Global	agen3
level	Flag specifying propulsion summary output required			output	Global	tanks
lift	Total lift force	newton	ader	input	Global	aprint
lines	Line counter for printout			output input	Global	agen3
lprint	Print flag			output	Global	lprint
lsb	Internal flag to indicate partial derivative runs			output	Global	rest
mision	Character indicating name of mission			output	Global	agen3
mnth	Character variable representing the name of the month used for range reference atmosphere calculations		atmosph	input	Global	atmos
moments	Output name				Local	
name(128)	Parameter names used in print output			output	Global	name
nmax	Iteration counter		mastre	input	Global	agen3
nn	Index for event				Local	
nstage	Index number of current stage		athrev	input	Global	mult
olow	Orbiter liftoff weight	kg	athrev	input	Global	olow
omega	Earth's spin rate	rad/sec	ainit	input	Global	const
orbet	Orbiter/external tank shear force	lbs	ader	input	Global	pdome
outisp(6)	Specific impulse of engine types	sec		input	Global	output

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
outth(6)	Actual thrust of engine types	lbs		input	Global	output
outtv(6)	Vacuum thrust of engine types	lbs		input	Global	output
outwd(6)	Flowrate of engine types	lbs/sec		input	Global	output
p	Longitude	rad		output input	Global	afprn
pat(14)	Input/output array for atmospheric parameters			input	Global	ardc
pdome	LOX dome pressure	psi	ader	input	Global	pdome
perige	Perigee altitude	nm			Local	
pfkg	Conversion for pounds to kilograms	kg/lbs	ainit	input	Global	const
pfn	Conversion from pounds to newtons	lbs/nt	ainit	input	Global	const
phi	Longitude	deg		output	Global	afprn
phimp	Longitude of impact point	deg			Local	
phipp	Geocentric latitude of current position	rad			Local	
phiri	Roll attitude angle of alternate coordinate system	deg			Local	
pi	Pi constant (3.14159265)		ainit	input	Global	const
plot	Logical variable used to indicate plot output file is to be generated			output	Global	plot
prop(6)	Propellant weight for each engine type	kg	ainit	input	Global	agen1
psffm	Conversion for pounds per square feet from newtons per square meter		ainit	input	Global	const
psiri	Yaw attitude angle of alternate coordinate system				Local	
q	Dynamic pressure	nt/m^2	ader	input	Global	agen2
qpenal	Value of dynamic pressure penalty function	psf			Local	
qqq	Output name				Local	
r	Radius from earth's center	m	aprtn	output input	Global	agen2
ratio	Propellant mixture ratio		ainit	input	Global	tcl
re	Earth's radius	m	ainit	input	Global	const
resave	Saved value of earth radius	m			Local	
reyno	Reynold's number		ader	input	Global	reyno
rid	Name of event used on output				Local	
rngan	Range angle			output input	Global	afprn

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
rngo	Ground range	m		output input	Global	afprn
rsth	Temporary variable	m			Local	
sa	Sine of launch azimuth		aforun	input	Global	agen2
schip	Sine of pitch attitude angle		ader1	input	Global	agen2
schir	Sine of roll attitude angle		ader1	input	Global	agen2
schiy	Sine of yaw attitude angle		ader1	input	Global	agen2
semach	Mach number based on using earth relative velocity			input	Global	aprint
side	Aerodynamic side force	newton	ader	input	Global	agen5
sinrho	Sine of angle between vehicle centerline and center of gravity location		ader	input	Global	garbge
srmet	SRM/External tank shear force	lbs		input	Global	pdome
srmirt	Magnitude of inertial velocity	m/sec		output input	Global	afprn
sth	Sine of colatitude			output input	Global	agen2
sthl	Sine of launch colatitude			input	Global	agen2
su	Y component of relative velocity vector	m/sec	ader	input	Global	agen2
sue	Y component of earth relative velocity vector	m/sec		input	Global	reyno
sv	Z component of relative velocity vector	m/sec	ader	input	Global	agen2
sve	Z component of earth relative velocity vector	m/sec		input	Global	reyno
sw	X component of relative velocity vector	m/sec	ader	input	Global	agen2
swe	X component of earth relative velocity vector	m/sec	ader	input	Global	reyno
t	Time from lift-off	sec	dpir	output input	Global	forint
tau	Thrust throttle value		ader	input	Global	garbge
tbias(6)	Bias between simulated time and reported time (used for branch option)	sec	athrev	input	Global	mult
tcl(6)	Longitudinal component of thrust vector for each system	newton	ader	input	Global	tcl
theri	Pitch attitude angle in alternate coordinate system	rad			Local	
thet	Geocentric latitude	rad			Local	
theta	Geocentric latitude of subvehicle point	deg	aprtn	output	Global	afprn

Variable	Variable Definition	Units	Source	I/O input	Status	Common Block
thetg	Geodetic latitude of subvehicle point	deg	aprtn	output input	Global	afprn
thimp	Latitude of impact point	deg			Local	
thmfa	Thrust minus axial force	nt	ader	input	Global	agen5
tht1	Colatitude angle of predicted impact point				Local	
tlift	Time to begin tiltover maneuver	sec	ainit	input	Global	agen1
tll(6)	Lateral component of thrust vector for each system	nt		input	Global	tcl
tmox	Yaw component of total moment in vehicle system				Local	
tmoy	Roll component of total moment in vehicle system				Local	
tmoz	Pitch component of total moment in vehicle system				Local	
tmx	Yaw component of thrust moment in vehicle system				Local	
tmy	Roll component of thrust moment in vehicle system				Local	
tmz	Z component of thrust moment	nt -m	ader	output input	Global	agen5
tvl(6)	Vertical component of thrust vector for each system	sec	ader	input	Global	tcl
tx1	Vertical component of thrust of 1st equivalent engine	newton	ader	input	Global	garbge
tx2	Vertical component of thrust of 2nd equivalent engine	newton	ader	input	Global	garbge
txx	Vertical component of moment balanced thrust	newton	ader	input	Global	agen5
ty1	Longitudinal component of thrust of 1st equivalent engine	newton	ader	input	Global	garbge
ty2	Axial thrust component for second equivalent engine	newton	ader	input	Global	garbge
tyy	Longitudinal component of moment balanced thrust	nt	ader	input	Global	agen5
tzz	Lateral component of moment balanced thrust	newton		input	Global	agen5
u	Y component of plumbline inertial velocity vector	m/sec	dpir	input	Global	forint

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
udxdyi	Temporary variable in moment balance calculation (1./sqrt(udxdy2))			input	Global	garbge
umf	One minus the earth flattening coefficient		ainit	input	Global	agen2
v	Z component of plumbline inertial velocity vector	m/sec	dpir	input	Global	forint
ve	Magnitude of earth relative velocity	m/sec	aprtn	output input	Global	reyno
vi	Inertial velocity			output input	Global	afprn
vph	Sine of relative azimuth				Local	
vphe	Magnitude of inertial velocity	m/sec		output input	Global	afprn
vr	Magnitude of relative velocity	m/sec	ader	output input	Global	agen2
vth	Cosine of relative azimuth				Local	
vthe	Magnitude of inertial velocity	m/sec		output input	Global	afprn
w	X component of plumbline inertial velocity vector	m/sec	dpir	input	Global	forint
wet	Rotation matrix of wt angle about y axis				Local	
wh2l	LH2 mass overboard tables	kg	ainit	input	Global	tanks
wh2t	LH2 fuel mass	lbs			Local	
wint	Initial vehicle weight	kg	aprtn	output	Global	olow
wloxt	not used	lbs			Local	
wmag	Wind speed magnitude	m/sec	ader	input	Global	agen5
word(160)	Storage array for output parameters	variabl	aprtn	output input	Global	name
wordrs	Storage array for relative coordinates				Local	
woxl	LOX propellant weight	lbs	ainit	input	Global	tanks
woxt	LOX fuel mass	lbs			Local	
wt	Product of earth spin rate and time	rad			Local	
wtot	Total vehicle weight	lbs			Local	
x	X component of plumbline position vector	m	dpir	input	Global	forint
xcg	Vertical component of center of gravity	m	ader1	input	Global	garbge
xddec	Relative acceleration vector				Local	
xdds	Acceleration vector	m/s^2			Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
xdec	Relative velocity vector				Local	
xds	Inertial velocity vector	m/sec			Local	
xec	Relative position vector	m			Local	
xinc	Inclination angle	rad	aprtn	output input	Global	afprn
xm	Current vehicle mass	kg	aprtn	output	Global	agen2
xmach	Mach number		ader	input	Global	agen2
xmaug	Auxiliary vehicle mass	kg	athrev	input	Global	agen1
xmiad	Continuous portion of vehicle mass	kg	ader1	input	Global	agen2
xnod	Angle between launch longitude and descending node	rad	aprtn	output	Global	afprn
xs	Position vector	m			Local	
y	Y component of plumbline position vector	m	dpir	input	Global	forint
ycg	Longitudinal component of centerof gravity	m	ader1	input	Global	garbge
yring	External tank Y-ring pressure	psi	ader	input	Global	pdome
z	Z component of plumbline position vector	m	dpir	input	Global	forint
zap(18)	Additional integrated variables	variabl	desolv	input	Global	forint

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
sphi	Sine of longitude				Local	

```
      SUBROUTINE APRTN(NN)                                              APRT   1
c**********************************************************************

c       This subroutine provides output for the program during the run
c       and generates output files for other uses

c**********************************************************************

C     Sam Powell
c     includes heat rate constraints

C     MODIFIED BY SINHA AND WEATHERS, JULY, 1989
C      COMPUTES OUTPUT PARAMETERS,PRINTS DATA,AND WRITES OUTPUT TAPE FOR
C     OUTPUT TABLES
C          NN     = INDEX FOR PRINTOUT IDENTIFICATION

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      LOGICAL DEMAND,GRAPH,GLIMIT,VISC

      COMMON/COMNEW/HRATEC

      CHARACTER*3 MNTH
      CHARACTER*6 MISION,AERONM(2),NAME,QQQ,ATMOSN,HAER,HATM,fomn,
     *moments(6)
      character*20 rid(25)
      CHARACTER*60 HEAD

      REAL*8 LIFT

      LOGICAL STAGE2,plot

      COMMON/CCC/GRAPH,JO,DEMAND

      COMMON/AGEN1/TIME(2,16),TAUT(15),TAUW(15),TZERO,TLIFT,TTILT,TMINH,
     *DTZ,TQ,TL,XMAUG,TNE(6,15),ENGDAT(8,15),S(15),WD(15),WJET(15),
     *HEAD,PRINT(15),STEP(15),HNOM(15),HMAX(15),PROP(6),TBEGR,TENDR,
     *CHRDOT,CHIRO,FAZ,BO(3,2),CBAXIL,TCHIR,VRCUT,FPRFAC,CHVEL

      COMMON/AGEN2/AA,SA,CA,ALF,ALFY,CHIP,SCHIP,CCHIP,CHIY,SCHIY,CCHIY,
     *CHIR,SCHIR,CCHIR,STH,CTH,STHL,CTHL,DPHIZ,RTHE,R,VR,XM,SW,SU,SV,Q,
     *VIV(25),DVARS(13),DELXDW(15,3),DELXDR(7,5),DELXD(15,7),WZERO,ALT,
     *XMACH,QDOT,FAA,FAN,A(3,3),CASE,CT2,UMF,A12W,A22W,A32W,XJEXT,BEU,
     *BYL,BEL,BHMAX(15),BHMN,BSTEP(15),HNMB,HMXB,TPOLY,XMIAD

      COMMON/AGEN3/ITHR,IWD,IXR,JUMP,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISION,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)

      COMMON/AGEN4/QALPHA,QBETA,QCN,STH1,CTH1,XJX(3),XJV(3)

      COMMON/AGEN5/CHIBAS,CHPBAS,CHYBAS,CHISAV,TXX,TYY,TZZ,AMX,AMY,AMZ,
     *TM2,THMFA,SIDE,WMAG,AZW,FANC,FLATC,FOM,VISC
```



```
      COMMON/AFPRN/VI,GAM,C1,C3,XINC,XNOD,XMLB,THETG,THETA,PHI,RNGO,RNGA
     *N,AZI,AZRE

      COMMON/ALAT/ALAT,ALONGC,ALTLS,NTABLE

      COMMON/APRINT/ELEIN,ELEOUT,DINBD,DOUTBD,FABT(4),CAX,CNP,CMZ,CS,
     *CMX,CMY,SEMACH,DRAG,LIFT,XLAMDA,XKNUD

      COMMON/ARDC/PAT(14),ATMOSN,IATMOS,HBIAS,HAERO

      COMMON/ASTOR/ASTO(4,67)

      COMMON /ATMOS / ATMTBL(110,6),MTHRRA,MATM1,MATM2,MNTH

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM

      COMMON/FORINT/HBANK(2),T,TT,W,U,V,X,Y,Z,XMI,ZAP(18),DVAR(25),FSAVE
     *(625),NTRG1,TV1,NTRG2,TV2,NTRG3,TV3,NTRG4,TV4,NTRG5,TV5,NTRG6,TV6,
     *NTRG7,TV7,NTRG8,TV8,NTRG9,TV9,NTRG10,TV10,NTRG11,TV11,NTRG12,TV12,
     *NTRG13,TV13,KIND,KRDER,EU,EL,AYL,HMX,HMN,HNM

      COMMON/GARBGE/DX,DYI,DZI,DXDY,UDXDYI,UDXDY2,XCG,YCG,TX1,TX2,TY1,TY
     *2,YHAT(3),TXS,TYS,TAU,SINRHO

      COMMON/JT/JT

      COMMON/LPRINT/LPRINT,INMAX,PREDP,QY,TOL

      COMMON/MPS/TIMMPS(15),FTBL(15),THROT(15),GLIM(15),TAUALT(5),
     *TAUTBL(5),MPS1,MPS2,TGLIM(10,15),GLIMIT,tau_const(10,5),jthrot(15)

      common/mult/nchiot(6),tbias(6),nstage,nstg(15),wf(5),ibranch

      COMMON/NAME/NAME(128),WORD(160)

      common/olow/olow,wint

      COMMON/OUTPUT/OUTTH(6),OUTTV(6),OUTISP(6),OUTWD(6)

      common/pdome/pdome,yring,orbet,srmet

      common/plot/plot

      COMMON/REST/JSTR,NCNRS(5),NTCN,TR(5),LSB,NF8,NOPAR,WIBT(15)

      COMMON/REYNO/REYNO,VE,SWE,SUE,SVE

      COMMON/SRMEXT/SRMEXT

      common/tanks/level,mlox,nlox,wlox(100),hlox(100),mh2,nfuel,
     *wfuel(200),hfuel(200),wh2l,woxl

      COMMON/TCL/TCL(6),TVL(6),TLL(6),RATIO

      COMMON/VISC/XMU
```

```
      DIMENSION D(3,3)

      DIMENSION XS(3),XDS(3),XDDS(3),XEC(3),XDEC(3),XDDEC(3),AT(3,3),
     *WET(3,3),DWET(3,3),DDWET(3,3),CTER1(9),CTER2(9),WORDRS(12)

      DIMENSION WLOXT(21),HEADT(21)

      EQUIVALENCE (SRMIRT,ASTO(1,47))

      EQUIVALENCE (ETIRT,ASTO(1,24))

      DATA(RID(I),I=1,25)/
     *     '                    ','LIFT-OFF           ',
     *     'IGNITION            ','                   ',
     *     'THRUST EVENT        ','BEGIN COAST        ',
     *     'BEGIN TILT          ','END TILT           ',
     *     'SRB SEPARATION      ','BEGIN MINH         ',
     *     'END COAST           ','LBM SEPARATION     ',
     *     'LBM IGNITION        ','MAX DYN PRES       ',
     *     'INJECTION           ','BEGIN CHIFRZ       ',
     *     'WEIGHT DROP         ','BEGIN ROLL         ',
     *     'END ROLL            ','INTERMEDIATE       ',
     *     'SEPARATION          ','G-LIMIT            ',
     *     'CORE SEPARATION     ','SHROUD JETTISON    ',
     *     'MPS THROTTLE        '/

      DATA AERONM/'  REV ','IVBC-3'/

      DATA (NAME(I),I=1,114)/
     *'  TIME','  VSUBE','  GAME','   AZE',' THRST','  XMLB',
     *'     R','  VSUBI','  GAMI','   AZI',' GCLAT','  LONG',
     *'   ALT','  VSUBR','  GAMR','   AZR',' GDLAT','  NODE',
     *'  Z8-X','  X8-Y','  Y8-Z',' ZD8-W',' XD8-U',' YD8-V',
     *'  CHIP','  CHIY','  CHIR','  TIMP',' LTIMP',' LNGMP',
     *' THERI',' PSIRI',' PHIRI','  INCL',' RANGE',' RNGAN',
     *' THMPS',' THSRM',' THLBM',' THOMS',' THRCS','    TH',
     *' TVMPS',' TVSRM',' TVLBM',' TVOMS',' TVRCS','  TVAC',
     *' ISPMP',' ISPSR',' ISPLB',' ISPOM',' ISPRC','   ISP',
     *' WDMPS',' WDSRM',' WDLBM',' WDOMS',' WDRCS','  WDOT',
     *' MPS F',' SRM F',' LBM F',' OMS F',' RCS F','  FUEL',
     *' MPS D',' SRM D',' LBM D',' OMS D',' RCS D',' SPENT',
     *' DPMPS',' DPSRM',' DPLBM',' DPOMS',' DPRCS','  DELP',
     *' DYMPS',' DYSRM',' DYLBM',' DYOMS',' DYRCS','  DELY',
     *'   TAU',' YRING',' PDOME',' SRMET',' ORBET',' CH VL',
     *'  TN L','  GV L','  DR L',' BKP L',' GIM L',' ID VL',
     *'  MACH',' ALPHA','  BETA','   FAA','   FAS','   FAN',
     *'     Q',' QALPH',' QBETA','   FAB','   FMB','   FNB',
     *'  DRAG','  LIFT',' VWIND','   AZW','  HEAT',' HRATE'/
      DATA (NAME(I),I=115,128)/
     *' PRESS','   RHO','  TEMP',' REYNO',' DELPC',' DELYC',
     *' EMACH','  INBD',' OUTBD',' AXACC',' NRACC',' LTACC',
     *'   XCG','   ZCG'/
C
      DATA QQQ,HAER,HATM,fomn/'  QPEN',' HAERO','HATMOS','   FOM'/
      data moments/
```



```
     *'   AMX','   AMY','   AMZ','   TMX','   TMY','   TMZ'/

      IF(LSB.EQ.1) RETURN                                              APRT  63

      IF(LPRINT.le.1) then

         IF(NMAX.NE.0.AND.NMAX.NE.INMAX) RETURN

      endif

      XM=XMIAD+XMAUG                                                   APRT  66

      R=SQRT(X*X+Y*Y+Z*Z)                                              APRT  67

      D(1,2)=X/R                                                       APRT  68
      D(2,2)=Y/R                                                       APRT  69
      D(3,2)=Z/R                                                       APRT  70

      CTH=A(1,2)*D(1,2)+A(2,2)*D(2,2)+A(3,2)*D(3,2)                    APRT  71
      SPHI=A(1,1)*D(1,2)+A(2,1)*D(2,2)+A(3,1)*D(3,2)                   APRT  72
      CPHI=A(1,3)*D(1,2)+A(2,3)*D(2,2)+A(3,3)*D(3,2)                   APRT  73

      PHI=ATAN2(SPHI,CPHI)                                             APRT  74
      IF(T.LT..1)PHI=DTZ*OMEGA                                         APRT  75
      XNOD=PHI                                                         APRT  76
      PHI=(PHI-OMEGA*(T-tbias(nstage)+DTZ))*RAD-ALONGO

      VI=(W*W+U*U+V*V)**.5                                             APRT  78

      GAM=0.                                                           APRT  79
      IF(ABS(VI).GT..1E-05) GAM=ASIN((W*X+U*Y+V*Z)/(R*VI))             APRT  80

      C1=VI*R*COS(GAM)                                                 APRT  81
      C3=VI*VI-2.*CMUE/R                                               APRT  82

      CHIP=ATAN2(SCHIP,CCHIP)                                          APRT  83
      CHIY=ATAN2(SCHIY,CCHIY)                                          APRT  84
      CHIR=ATAN2(SCHIR,CCHIR)
      IF(CHIY.GT.PI) CHIY=CHIY-2.*PI                                   APRT  85

      IF(T .le. TLIFT) then

         PSIRI=CHIY*RAD+AA*RAD
         THERI=90.
         PHIRI=CHIR*RAD

      else

C     YAW

         PSIRI=ATAN2(SA*SCHIP*CCHIY+SCHIY*CA,CA*SCHIP*CCHIY-
     *   SCHIY*SA)*RAD

         IF(PSIRI.LT. 0.) PSIRI=PSIRI+360.

C     PITCH
```

```
         THERI=ASIN(CCHIP*CCHIY)*RAD

C        ROLL

         PHIRI=ATAN2(SCHIP*SCHIR+CCHIR*SCHIY*CCHIP,CCHIR*SCHIP-
     *   SCHIR*SCHIY*CCHIP)*RAD

        endif

        ALTP=ALT                                                    APRT  86
        STH=(1.-CTH*CTH)**.5                                        APRT  87
        RSTH=R*STH                                                  APRT  88

        D(1,3)=(D(1,2)*CTH-A(1,2))/STH                              APRT  89
        D(2,3)=(D(2,2)*CTH-A(2,2))/STH                              APRT  90
        D(3,3)=(D(3,2)*CTH-A(3,2))/STH                              APRT  91

        D(1,1)=(-Y*A(3,2)+Z*A(2,2))/RSTH                            APRT  92
        D(2,1)=(X*A(3,2)-Z*A(1,2))/RSTH                             APRT  93
        D(3,1)=(-X*A(2,2)+Y*A(1,2))/RSTH                            APRT  94

        GAMR=0.                                                     APRT  95
        IF(ABS(VR).GT..1E-05) GAMR=ASIN((SW*X+SU*Y+SV*Z)/(R*VR))*RAD APRT  96

        GAMRE=0.
        IF(ABS(VE).GT..1E-05) GAMRE=ASIN((SWE*X+SUE*Y+SVE*Z)/(R*VE))*RAD

        VTH=D(1,3)*SW+D(2,3)*SU+D(3,3)*SV                           APRT  97
        VPH=D(1,1)*SW+D(2,1)*SU+D(3,1)*SV                           APRT  98

        AZR=ATAN2(VPH,-VTH)*RAD                                     APRT  99

        VTHE=D(1,3)*SWE+D(2,3)*SUE+D(3,3)*SVE
        VPHE=D(1,1)*SWE+D(2,1)*SUE+D(3,1)*SVE

        AZRE=ATAN2(VPHE,-VTHE)*RAD

        VTH=D(1,3)*W+D(2,3)*U+D(3,3)*V                              APRT 100
        VPH=D(1,1)*W+D(2,1)*U+D(3,1)*V                              APRT 101

        AZI=ATAN2(VPH,-VTH)                                         APRT 102

        THET=ACOS(CTH)                                              APRT 103
        THETA=90.-THET*RAD                                          APRT 104

        THETG=ATAN2(SIN(THETA/RAD),UMF*UMF*COS(THETA/RAD))*RAD      APRT 105

        CINC=SIN(AZI)*SIN(THET)                                     APRT 106
        XINC=ACOS(CINC)                                             APRT 107

        XNOD=(XNOD+ATAN2(CTH*VPH,VTH))*RAD                          APRT 108

        A1=ABS(CMUE/C3)                                             APRT 109
        P=C1*C1/CMUE                                                APRT 110
```



```
        ECC=1.+P*C3/CMUE                                            APRT 111

        RESAVE=RE

        IF(ECC.ge.0.) then

           ECC=SQRT(ECC)
           APOGEE=((P/(1.0-ECC))-RE)/1852.0
           PERIGE=((P/(1.0+ECC))-RE)/1852.0

           IF(RE-P/(1.+ECC).ge.RE/100.) then

              CETANW=(P-R)/(R*ECC)
              ETANW=ABS(CETANW)

              IF(ETANW.gt.1.) then

                 ETANW=0.

              else

                 ETANW=ACOS(ETANW)

              endif

              IF(CETANW.LT.0.) ETANW=PI-ETANW
              IF(GAM.LT.0.) ETANW=2.*PI-ETANW
              CETATN=(P-RE)/(RE*ECC)
              ETATN=ABS(CETATN)

              IF(ETATN.gt.1.) then

                 ETATN=0.

              else

                 ETATN=ACOS(ETATN)

              endif

              IF(CETATN.LT.0.) ETATN=PI-ETATN
              DELETA=2.*PI-ETATN-ETANW
              CBGENW=(A1-R)/(A1*ECC)
              BGENOW=ABS(CBGENW)

              IF(BGENOW.gt.1.) then

                 BGENOW=0.

              else

                 BGENOW=ACOS(BGENOW)

              endif

              IF(CBGENW.LT.0.) BGENOW=PI-BGENOW
```

```
            IF(GAM.LT.0.) BGENOW=2.*PI-BGENOW
            CBGETN=(A1-RE)/(A1*ECC)
            BGETHN=ABS(CBGETN)

            IF(BGETHN.gt.1.) then

               BGETHN=0.

            else

               BGETHN=ACOS(BGETHN)

            endif

            IF(CBGETN.LT.0.)BGETHN=PI-BGETHN
            BGETHN=2.*PI-BGETHN
            DELBGE=BGETHN-BGENOW
            DTI=(DELBGE-ECC*(SIN(BGETHN)-SIN(BGENOW)))/
     *           SQRT(CMUE/(A1*A1*A1))
            THT1=ACOS(CTH*COS(DELETA)+STH*SIN(DELETA)*COS(AZI))
            DLPHI=ASIN(SIN(DELETA)*SIN(AZI)/SIN(THT1))
            PHIMP=PHI+(DLPHI-OMEGA*DTI)*RAD
            THIMP=90.-THT1*RAD

         endif

      endif

      RE=RESAVE

   90 PHIPP=ATAN2(CTHL,STHL)

      DLTL=DTZ

      DO 79 I=1,2                                                    APRT 160

      DUMMY = (SIN(PHIPP)*SIN(THETA/RAD)+COS(PHIPP)*COS(THETA/RAD)*
     *         COS((PHI+ALONGO)/RAD+OMEGA*DLTL))

      IF (DUMMY .GT. 1.0) THEN

          DUMMY = 1.0

C         CALL HOLDIT

      ENDIF

      RNGAN=ACOS(DUMMY)

C     RNGAN=ACOS(SIN(PHIPP)*SIN(THETA/RAD)+COS(PHIPP)*COS(THETA/RAD)*
C    $ COS((PHI+ALONGO)/RAD+OMEGA*DLTL))

      IF(I.eq.1) then

         IF(RNGAN.LT..0001) RNGAN=0.
         RNGO=RNGAN*RE
```



```
         DLTL=T-tbias(nstage)+DTZ

      endif

79    continue

      DTX=2.*TX2-TX1                                                 APRT 193
      DTY=TY1-2.*TY2                                                 APRT 194
      DTXP=DTX*UDXDYI+DTY*SINRHO                                     APRT 195
      DTYP=-DTX*SINRHO+DTY*UDXDYI                                    APRT 196

 159  CONTINUE

      IF(ABS(DZI).GT.0.D0 .AND. ABS(DYI).GT.0.D0 .AND.
     *  ABS(DX).GT.0.D0) THEN

         TMX=DTYP/DZI+TZZ/DYI
         TMY=DTXP/DZI-TZZ*DX
         TMZ=-TXX/DYI+TYY*DX

      ELSE

         TMX=0.
         TMY=0.

      ENDIF

      TMOX=AMX+TMX
      TMOY=AMY+TMY
      TMOZ=AMZ+TMZ

      IF(ALTP.LT.0..AND.T.LT.10.) ALTP=0.                            APRT 201


   50 IF(LINES.le.0) then

C     WRITE HEADER                                                   APRT 204

         KPAGE=KPAGE+1

         WRITE(JO,56)CASE,HEAD,KPAGE,AERONM

   56    FORMAT(1H1,////' MASTRE CASE=',F11.4,10X,A60,7X,'PAGE',I3,
     *   '(AERO-2/83',2A6,1H))

         WRITE(JO,57) ATMOSN,MNTH

   57    FORMAT(107X,'(ATMOS - ',A6,',',A3,')')

         write(jo,'(107x,a,i3)') ' stage -',nstage

   41    LINES=35

      endif

      FTM = 0.3048
```

```
      WORD(1)=T-tbias(nstage)
      WORD(2)=VE/FTM
      WORD(3)=GAMRE
      WORD(4)=AZRE
      WORD(5)=(OUTTH(1)+OUTTH(2)+OUTTH(3)+OUTTH(4)+OUTTH(5)+OUTTH(6))*
     *        PFN
      WORD(6)=XM*PFKG
      WORD(7)=R/FTM
      WORD(8)=VI/FTM
      WORD(9)=GAM*RAD
      WORD(10)=AZI*RAD

      WORD(11)=THETA
      WORD(12)=PHI
      WORD(13)=ALT/FTM
      WORD(14)=VR/FTM
      WORD(15)=GAMR
      WORD(16)=AZR
      WORD(17)=THETG
      WORD(18)=XNOD
      WORD(19)=X/FTM
      WORD(20)=Y/FTM

      WORD(21)=Z/FTM
      WORD(22)=W/FTM
      WORD(23)=U/FTM
      WORD(24)=V/FTM
      WORD(25)=CHIP*RAD
      WORD(26)=CHIY*RAD
      WORD(27)=CHIR*RAD
      WORD(28)=DTI
      WORD(29)=THIMP
      WORD(30)=PHIMP

      WORD(31)=THERI
      WORD(32)=PSIRI
      WORD(33)=PHIRI
      WORD(34)=XINC*RAD
      WORD(35)=RNGO/1852.
      WORD(36)=RNGAN*RAD
      WORD(37)=OUTTH(1)*PFN
      WORD(38)=OUTTH(2)*PFN
      WORD(39)=OUTTH(3)*PFN
      WORD(40)=OUTTH(4)*PFN

      WORD(41)=OUTTH(5)*PFN
      WORD(42)=OUTTH(6)*PFN
      WORD(43)=OUTTV(1)*PFN
      WORD(44)=OUTTV(2)*PFN
      WORD(45)=OUTTV(3)*PFN
      WORD(46)=OUTTV(4)*PFN
      WORD(47)=OUTTV(5)*PFN
      WORD(48)=OUTTV(6)*PFN
      WORD(49)=OUTISP(1)
      WORD(50)=OUTISP(2)
```



```
      WORD(51)=OUTISP(3)
      WORD(52)=OUTISP(4)
      WORD(53)=OUTISP(5)
      WORD(54)=OUTISP(6)
      WORD(55)=OUTWD(1)*PFKG
      WORD(56)=OUTWD(2)*PFKG
      WORD(57)=OUTWD(3)*PFKG
      WORD(58)=OUTWD(4)*PFKG
      WORD(59)=OUTWD(5)*PFKG
      WORD(60)=OUTWD(6)*PFKG

      WORD(61)=ZAP(1)*PFKG
      WORD(62)=ZAP(2)*PFKG
      WORD(63)=ZAP(3)*PFKG
      WORD(64)=ZAP(4)*PFKG
      WORD(65)=ZAP(5)*PFKG
      WORD(66)=ZAP(6)*PFKG
      WORD(67)=PROP(1)-WORD(61)
      WORD(68)=PROP(2)-WORD(62)
      WORD(69)=PROP(3)-WORD(63)
      WORD(70)=PROP(4)-WORD(64)

      WORD(71)=PROP(5)-WORD(65)
      WORD(72)=PROP(6)-WORD(66)

      DO 70 I=1,6

      WORD(I+72)=0.
      WORD(I+78)=0.

      IF(TCL(I).ge.1.E-04) then

         WORD(I+72)=ATAN2(-TVL(I),TCL(I))*RAD
         WORD(I+78)=ATAN2(TLL(I),TCL(I))*RAD

      endif

   70 CONTINUE

      WORD(85)=TAU
      WORD(86)=YRING
      WORD(87)=PDOME
      WORD(88)=SRMET
      WORD(89)=ORBET
      WORD(90)=ZAP(8)-ZAP(9)-ZAP(10)

      WORD(91)=ZAP(12)
      WORD(92)=ZAP(11)
      WORD(93)=ZAP(13)
      WORD(94)=ZAP(9)
      WORD(95)=ZAP(10)
      WORD(96)=ZAP(8)
      WORD(97)=XMACH
      WORD(98)=ALF*RAD
      WORD(99)=ALFY*RAD
```

```
      WORD(100)=FAA*PFN

      WORD(101)=SIDE*PFN
      WORD(102)=FAN*PFN
      WORD(103)=Q*PSFFM
      WORD(104)=WORD(103)*WORD(98)
      WORD(105)=WORD(103)*WORD(99)
      WORD(106)=FABT(1)*PFN
      WORD(107)=FABT(3)*PFN/.3048
      WORD(108)=FABT(2)*PFN
      WORD(109)=DRAG*PFN
      WORD(110)=LIFT*PFN

      WORD(111)=WMAG
      WORD(112)=AZW*RAD
      WORD(113)=ZAP(7)
      WORD(114)=DVAR(14)

      IF(T.LT.5.)HRATEC=0.

      IF(T.LT.150.)THEN

         IF(HRATEC.LE.DVAR(14))HRATEC=DVAR(14)

      ENDIF

      IF(IATMOS.EQ.0) THEN

         WORD(115)=PAT(2)*.020885434

      ELSE

         WORD(115)=PAT(2)*208.85434

      ENDIF

      WORD(116)=PAT(6)
      WORD(117)=PAT(12)
      WORD(118)=REYNO
      WORD(119)=0.
      WORD(120)=0.

      IF(TYY.ge.1.E-04) then

         WORD(119)=ATAN2(-TXX,TYY)*RAD
         WORD(120)=ATAN2(TZZ,TYY)*RAD

      endif

      WORD(121)=SEMACH
      WORD(122)=ELEIN
      WORD(123)=ELEOUT
      WORD(124)=THMFA/XM/GZERO
      WORD(125)=FANC/XM/GZERO
      WORD(126)=FLATC/XM/GZERO
      WORD(127)=YCG*39.3701
```



```
      WORD(128)=XCG*39.3701
      WORD(129)=ZAP(15)
      WORD(130)=REAL(N)
C
C     FILL IN SPARE WORDS WITH ZERO VALUES
C
      DO 72 IS=1,30

      ISS = 130 + IS
 72   WORD(ISS) = 0.0
C
C***************** RANGE SAFETY CALCULATIONS*************************
C
C
C     COMPUTE X,XDOT,XDDOT IN EARTH FIXED,EARTH CENTERED COORD
C     SYSTEM(PACSS11)
C
C     XEC=[A][WET][AT]XS
C     XDEC=[A][DWET][AT]XDS
C     XDDEC=[A][DDWET][AT]XDDS
C
      XS(1)=X
      XS(2)=Y
      XS(3)=Z

      XDS(1)=W
      XDS(2)=U
      XDS(3)=V

      XDDS(1)=DVAR(1)
      XDDS(2)=DVAR(2)
      XDDS(3)=DVAR(3)

      WT=OMEGA*T

C     [WET] MATRIX FORMED BY A POSITIVE OMEGA(WE) ROTATION ABOUT
C     THE YEQ AXIS

      DO 8 I=1,3

      DO 8 J=1,3

      WET(I,J)=0.
      DWET(I,J)=0.
    8 DDWET(I,J)=0.

      WET(1,1)=COS(WT)
      WET(3,1)= SIN(WT)
      WET(2,2)=1.
      WET(1,3)=-SIN(WT)
      WET(3,3)=COS(WT)
C
      DWET(1,1)=-OMEGA*SIN(WT)
      DWET(3,1)= OMEGA*COS(WT)
      DWET(1,3)=-DWET(3,1)
      DWET(3,3)=DWET(1,1)
```

```
C
      DDWET(1,1)=-OMEGA**2*COS(WT)
      DDWET(3,1)=-OMEGA**2*SIN(WT)
      DDWET(1,3)=-DDWET(3,1)
      DDWET(3,3)=DDWET(1,1)
C
      DO 3 I=1,3

      DO 3 J=1,3

    3 AT(J,I)=A(I,J)
C
      CALL TRNSFM(CTER1,AT,XS)
      CALL TRNSFM(CTER1(4),AT,XDS)
      CALL TRNSFM(CTER1(7),AT,XDDS)

C     POSITION

      CALL TRNSFM(CTER2,WET,CTER1)
      CALL TRNSFM(XEC,A,CTER2)

C     VELOCITY

      CALL TRNSFM(CTER2,DWET,CTER1)
      CALL TRNSFM(CTER2(4),WET,CTER1(4))

      DO 9 I=1,3

    9 CTER2(I)=CTER2(I)+CTER2(I+3)

      CALL TRNSFM(XDEC,A,CTER2)

C     ACCELERATION

      CALL TRNSFM(CTER2,DDWET,CTER1)
      CALL TRNSFM(CTER2(4),DWET,CTER1(4))
      CALL TRNSFM(CTER2(7),WET,CTER1(7))

      DO 10 I=1,3

   10 CTER2(I)=CTER2(I)+2.*CTER2(I+3)+CTER2(I+6)

      CALL TRNSFM(XDDEC,A,CTER2)
C
      WORDRS(1)=XEC(2)
      WORDRS(2)=XEC(3)
      WORDRS(3)=XEC(1)

      WORDRS(4)=XDEC(2)
      WORDRS(5)=XDEC(3)
      WORDRS(6)=XDEC(1)

      WORDRS(7)=XDDEC(2)
      WORDRS(8)=XDDEC(3)
      WORDRS(9)=XDDEC(1)
```



```
      WORDRS(10)=0.
      WORDRS(11)=0.
      WORDRS(12)=0.
C
C
C ***************** END RANGE SAFETY CALCULATIONS ********************
C
      if(level.ne.0) then

         wtot=zap(1)
         if(level.eq.2) wtot=zap(1)+zap(2)

c       compute LOX tank mass and tank height

         woxt=ratio*wtot*pfkg-woxl

         call amulg(1,mlox,nlox,woxt,wlox,hoxt,hlox,dum,dum)

c       compute fuel tank level and fuel mass

         wh2t=wtot*pfkg-woxt-woxl-wh2l

         call amulg(1,mh2,nfuel,wh2t,wfuel,h2ht,hfuel,dum,dum)

         word(131)=hoxt
         word(132)=woxt
         word(133)=h2ht
         word(134)=wh2t

      endif

      QPENAL=ZAP(14)*PSFFM
      HATMOS=ALTP+HBIAS

      WRITE(JO,61) RID(nn),(NAME(I),WORD(I),I=1,128)
     *,QQQ,QPENAL,HAER,HAERO,HATM,HATMOS,fomn,fom,
     *moments(1),amx,moments(2),amy,moments(3),amz,moments(4),tmx,
     *moments(5),tmy,moments(6),tmz,'  olow',olow*pfkg,'  wint',
     *wint*pfkg,' tbias',tbias(nstage),' x acc',dvar(1),' y acc',
     *dvar(2),' z acc',dvar(3)

   61 FORMAT(1X,A20/(4X,6(A6,1PE15.8)))

      LINES=LINES-23

      IF(NN.EQ.15) LINES=0
C
      IF(NMAX.ne.0) return
C
C***  WRITE OUTPUT FOR REPORT                                          APRT 290

      WRITE(9) RID(nn),WORD,MISION

      if(plot) write(19,*) word

      RETURN                                                           APRT 293
```

```
C
      END                                                    APRT 294
```

3.16 Subroutine AREAIN

3.16.1 Purpose

The purpose of this subroutine is to display and activate user options pertaining to execution, termination, selecting namelist data, writing plot and table output files, and providing a printout menu.

3.16.2 Variable Listing

3.16.3 Subroutines Called:

Name	Function
COPLDF	Writes a list directed file
GOUT	Displays character strings to the screen
TERMINATE	Terminates execution of program

3.16.4 Calling Subroutines:

Name	Function
MENU	Controls program execution

3.16.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
iuse	Operational mode index			output input	Global	iuse
llflag	File control index			output	Global	llflag

```
      SUBROUTINE AREAIN
C
C
      COMMON/IUSE/IUSE
      COMMON/LLFLAG/LLFLAG
C
      CALL PAGIT
C
      IUSE=0
C
 1    CALL GOUT('  SELECT FROM THE FOLLOWING MENU:',33)
      CALL GOUT('  0 - TERMINATE',15)
      CALL GOUT('  1 - EXECUTE',13)
      CALL GOUT('  2 - SELECT NAMELIST DATA INPUT.',33)
      CALL GOUT('  3 - WRITE FILE FOR PLOTS AND TABLES',37)
      CALL GOUT('  4 - PRINTOUT MENU',19)
C
      WRITE(6,2)
 2    FORMAT(1X,'CHOICE> ',$)
      READ(5,3,ERR=1) IUSE
 3    FORMAT(I1)
      IF(IUSE.LT.0 .OR. IUSE.GT.6)GO TO 1
      IUSE=IUSE+1
      GO TO(10,11,12,13,14),IUSE
C
C---------------------------- TERMINATE -------------------------
C
 10   CALL TERMINATE
      STOP
C
C---------------------------- EXECUTE ---------------------------------
C
 11   CALL GOUT(' YOU ARE NOW EXECUTING.',24)
      LLFLAG=1
C
C---------------------------- DISPLAY NAMELIST INPUT -----------------
C
 12   IUSE=IUSE-1
      RETURN
C
C---------------------------- WRITE FILE FOR PLOTS/TABLES -----------
C
 13   CALL COPLDF
      GO TO 1
C
C---------------------------- PRINTOUT MENU ----------------------
C
 14   CONTINUE
      GO TO 12
C
      END
```

3.17 Subroutine ATHREV

3.17.1 Purpose

The purpose of this subroutine is to control the thrust event logic during the forward trajectory integration. Parameters for each thrust event, as well as stage changes, are defined based on input or calculated.

3.17.2 Variable Listing

3.17.3 Subroutines Called:

Name	Function
ADER	Calculates the time independent portion of the equations of motion during forward integration
ADER1	Calculates the time dependent portion of the equations of motion during forward integration
AEOSR	Stores state variables during forward integration
APRTN	Prints trajectory block output and files
FIND	Calculates attitude polynomials
FUNISP	Calculates specific impulse based on throttle level
RTMRK	Causes termination of integration

3.17.4 Calling Subroutines:

Name	Function
AFORUN	Controls logic for forward trajectory and calls integration routine
DPIR	Integration routine

3.17.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ane	Not used				Local	
ap(10)	Coefficients for booster pitch attitude		atilt	input	Global	bodyf
api(10,2)	Coefficients for booster pitch attitude		atilt	output	Global	body
astor(4,67)	Storage array for weight output table	lbs	ainit	input	Global	astor
atbdwt(15,2)	Weight loss table for center of gravity tables	kg	ainit	input	Global	body
atxcg(15,2)	Storage array for vertical center of gravity tables	m	ainit	input	Global	body
atycg(15,2)	Storage array for longitudinal center of gravity table	m	ainit	input	Global	body
axlen(2)	Aerodynamic reference length	m		input	Global	body
axref(2)	Vertical aerodynamic moment reference point	m		input	Global	body
ay(10)	Coefficients for booster yaw attitude		atilt	input	Global	bodyf
ayi(10,2)	Coefficients for booster yaw attitude		atilt	output	Global	body
ayref(2)	Longitudinal aerodynamic moment reference point	m	aint	input	Global	body
azre	Wind relative azimuth	rad	aprtn	input	Global	afprn
branch_point	Flag indicating branch point thrust event				Local	
cchip	Cosine of pitch attitude angle		ader1	output	Global	agen2
cchiy	Cosine of yaw attitude angle		ader1	input	Global	agen2
cfr(15)	Critical flowrate	kg/sec	ainit	input	Global	xtra
char_vel(5)	Characteristic velocity	m/sec	ainit	output	Global	multi
chip	Pitch attitude angle	rad	ader1	input	Global	agen2
chisav	Saved value of chip	rad	ader1	input	Global	agen5
chiy	Yaw attitude angle	rad	ader1	input	Global	agen2
chvel	Characteristic velocity	m/sec	mastre	output	Global	agen1
cisp(15)	MPS specific impulse	sec	ainit	input	Global	xtra
cptbl(30)	Booster pitch attitude angle table	rad	anewch	output	Global	contro
cytbl(30)	Booster yaw attitude angle table	rad	athrev	output	Global	contro
deltav	Delta velocity required	m/sec			Local	
delx(7)	State variable storage during partial evaluation		athrev	output	Global	delx
delxd(15,7)	State derivative storage at thrust events		athrev	output	Global	agen2
delxdb(7)	State derivative storage for branch trajectory		athrev	output	Global	delxdb

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
delxdr(7,5)	State derivative storage at desired intermediate point		athrev	output	Global	agen2
delxh(5)	Storage array for stagnation heating rate at intermediate constraint thrust events		athrev	output	Global	delxh
dvar(25)	Storage array for the state derivatives		ader	input output	Global	forint
dvars(13)	Derivative storage at the thrust events		athrev	output input	Global	agen2
dvisp	Specific impulse of upper stage	sec			Local	
engdat(8,15)	Engine data matrix		ainit	input	Global	agen1
engines(15,1	Index used to indicate which engines are being simulated for a given thrust event		ainit	input	Global	engines
etfuel	Propellent weight in external tank	kg			Local	
f(15)	MPS engine thrust	newton		input	Global	xtra
fmxx	Temporary variable used for calculation of partial derivatives for tank constraints		athrev	output	Global	fmxx
foms(15)	Thrust table for orbit maneuver system	lbs	ainit	input	Global	omsrcs
frcs(15)	Thrust table for rocket control system	lbs	ainit	input	Global	omsrcs
glim(15)	Acceleration limit	g's	ainit	input	Global	mps
glimit	Logical variable indicating aceleration limit has been achieved		atilt(gli	output	Global	mps
gzero	Gravitational constant (9.80665 m/sec)	m/sec^	ainit	input	Global	const
hmx	Maximum step size	sec		output	Global	forint
hnm	Nominal step size	sec		output	Global	forint
ibranch	Flag indicating the terminal thrust event of the first trajectory of the branch trajectory			output	Global	mult
iconsw	Flag to indicate termination of SRB moment balance		ainit	output	Global	agen3
ipoly	Flag indicating number of polynominals in booster stage		ainit	input	Global	agen3
ipr	Denotes the thrust event from which fuel propellant reserve (FPR) is calculated			input output	Global	ipr
iprop(6)	Propulsion cutoff flags (stage dependent)		ainit	input	Global	ipr
iprp	Index (ipr-1)				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
iprt	Print flag				Local	
irtaoa	Flag indicating thrust event where shuttle rtls-aoa point is located			output	Global	irtaoa
irtflg	Flag indicating FPR calculations			output	Global	agen3
irtls	Return to launch flag			input	Global	agen3
irtls1	Index				Local	
istart(2)	Index used for jump start for branch trajectory		ainit	input	Global	istart
ithr	Thrust event index number		athrev	input output	Global	agen3
ithro	Index for SSME throttle table		atilt(ath	output input	Global	ithro
ithrot	Throttle index				Local	
iwd	Index for weight drop events		aforun	input	Global	agen3
ja	Index				Local	
jstart(6)	Index for use with branch trajectory option		ainit	input	Global	istart
jt	Thrust event number			output input	Global	jt
jthrot(15)	Index indicating number of times the acceleration limit has been reached within a given thrust event		atilt(gli	output	Global	mps
jthrot_save(1	Saved value of jthrot		atilt	output	Global	jthrot_save
jump	Jump start flag		ainit	output	Global	agen3
jumpst	Saved value of jump		athrev	output	Global	jumpst
kcdres(6,5)	Array for intermediate constraint function code		ainit	input	Global	enput
kcytab	Attitude control table index		athrev	output input	Global	agen3
kdb(40)	Array for control parameter flags		ainit	output	Global	auto
kp	Order of booster pitch attitude polynomial		athrev	output	Global	kp
kp1	Order of pitch attitude polynomial				Local	
kpi(2)	Order of booster pitch attitude polynomials		ainit	input	Global	body
ks1	Booster attitude indicator		athrev	output	Global	agen3
ky	Order of booster yaw attitude polynomial		athrev	output	Global	kp

athrev

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ky1	Order of yaw attitude polynomial				Local	
kyi(2)	Order of booster yaw attitude polynomials		ainit	input	Global	body
lsb	Internal flag to indicate partial derivative runs			output	Global	rest
lstg	Aerodynamic coefficient flag to distinguish stages				Local	
lstge(15)	Aerodynamic coefficient flag to distinguish stages		ainit	input	Global	agen3
lvrst	Flag indicating branch point				Local	
m	Index				Local	
m1	Index to indicate place in aerodynamic tables			output	Global	bodyf
m11	Previous index for elevon lookup in forward trajectory			output	Global	elevon
m15	Pointer index for roll angle (sigma) lookup			output	Global	ricont
m16	Pointer index for pitch and yaw angle (theta & phi) lookup			output	Global	ricont
m17	Pointer index for base force tables			output	Global	m20
m18	Pointer index for continuum and free molecular aerodynamic data			output	Global	vaero
m19	Previous index for continuum and free molecular aerodynamic data		athrev	output	Global	vaero
m2	Previous index for aerodynamic coefficient tables for forward trajectory			output	Global	bodyf
m21	Pointer index for aerodynamic coefficient increments			output	Global	m20
m22	Not used			output	Global	m20
m23	Pointer index for base force coefficent tables			output	Global	m20
m24	Pointer index for base force tables			output	Global	m20
m30	Previous index for base force increment tables			output	Global	delfab
m31	Previous index for base force increment tables			output	Global	delfab
m5	Pointer index for center of gravity lookup			output	Global	bodyf
m51	Pointer index for throttle level table			output	Global	bodyf

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
m6	Previous index for center of gravity lookup			output	Global	bodyf
m7	Pointer index for wind table lookup			output	Global	bodyf
m8	Pointer array for base axial force table			output	Global	bodyf
m9	Previous index for base axial force table		athrev	output	Global	bodyf
mcon(7)	Not used			output	Global	tablk
minh	Flag to indicate thrust event which begins min-h attitude optimization		ainit	output	Global	agen3
mision	Character indicating name of mission			output	Global	agen3
mission(2)	Character indicating name of mission			input	Global	mission
mlbm1	Pointer index for LBM thrust and flowrate interpolation in forward integration		athrev/	output	Global	lbm
mlbm2	not used			output	Global	lbm
mombal	Flag to specify the use of moment balance equations		ainit	input	Global	mombal
mps1	Not used			output	Global	mps
mps2	Not used			output	Global	mps
mpsflg(15)	Flag indicating the type of MPS thrust simulation		ainit	output	Global	agen3
msrb	Pointer index for SRB thrust interpolation		athrev/	output	Global	srm
msrw	Previous index for wind parameters interpolation			output	Global	bodyf
msw	Temporay variable		athrev	output	Global	fmxx
mswch(15)	Flag indicating which thrust events are considered critical			output	Global	agen3
n	Index				Local	
nchiot(6)	Optimization flag used with branch trajectory option		ainit	input	Global	mult
ncoast	Thrust event to begin coast on final case		ainit	input	Global	agen3
nendct(7)	Index indicating end of min-H phases		ainit	input	Global	agen3
nf8	Number of differential equations to be integrated		aforun	input	Global	rest
nmax	Iteration counter		mastre	input	Global	agen3
nmps	Index of first MPS engines in nsyst array				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
noevnt(5)	Number of thrust events per stage		ainit	output input	Global	agen3
nowd(15)	Index indicating thrust event where weight drop event will occur		ainit	input	Global	agen3
nstage	Index number of current stage		athrev	output input	Global	mult
nstg(15)	Internal index which relates thrust event to stage number		ainit	input	Global	mult
nsyst(15)	Index array which specifies the type of engines that will be used for each thrust event		ainit	input	Global	agen3
ntrg13	MPS throttle trigger number		athrev	output	Global	forint
ntrg2	Thrust event trigger number		athrev	output	Global	forint
ntrg9	Acceleration limit trigger number		athrev	output	Global	forint
nvnt	Number of thrust events		ainit	input	Global	agen3
nvrst(5)	Array specifying the thrust events where intermediate constraints will be imposed		ainit	output	Global	enput
nwvnt	Number of weight drop events		ainit	input	Global	agen3
ocow	Orbiter cutoff weight	kg			Local	
olow	Orbiter liftoff weight	kg	athrev	output input	Global	olow
pfkg	Conversion for pounds to kilograms	kg/lbs	ainit	input	Global	const
prk4	Temporary variable used in calculation of performance reserve option		ainit	input	Global	ipr
prk5(5)	Same as prk4 but stage dependent (used with branch trajectory option)		athrev	output	Global	ipr
prop(6)	Propellant weight for each engine type	kg	ainit	input	Global	agen1
prtot	Total propellant weight used for performance reserve option	kg	ainit	output input	Global	ipr
ptkg	Conversion from pounds to kilograms	kg/lbs	ainit	input	Global	const
ptn	Conversion from pounds to newtons	nt/lbs	ainit	input	Global	const
qload(26)	Storage array for parameters printed in final loads output tables		aforun	input output	Global	loads
qtime(26)	Storage array for times associated with parameters in qload array	sec	athrev	input output	Global	loads
restx(10,5)	Variable storage array for jump start option		athrev	output	Global	restx
save(20,13)	Storage array	variabl	athrev	output	Global	loads

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
save_mass	Storage variable for final mass used in branch option		athrev	output	Global	save_mass
schi	Sine of pitch attitude angle				Local	
schip	Sine of pitch attitude angle		ader1	output	Global	agen2
schiy	Sine of yaw attitude angle		ader1	input	Global	agen2
spoa	Specific impulse divided by acceleration limit			output	Global	fmxx
startv(40,2)	Saved values of state for jump start			input output	Global	istart
step(15)	Integration step size	sec	ainit	input	Global	agen1
svazre	Saved value of relative azimuth	rad			Local	
svxinc	Saved value of inclination angle for AOA trajectory constraint for variable IY targeting	rad		output	Global	istart
svxnod	Saved value of descending nodal angle for AOA trajectory constraint for variable IY targeting			output	Global	istart
sychi	Saved value of yaw attitude angle				Local	
system	Name of the vehicle system being simulated (SHUTTLE, STAGE15, HLLV)		ainit	input	Global	system
t	Time from lift-off	sec	dpir	output input	Global	forint
taut(15)	Time increments for thrust events	sec	anewch	input output	Global	agen1
tauw(15)	Time increment for weight drop event	sec	ainit	input	Global	agen1
tau_const(10,	Constant value of throttle used for constant throttle after acceleration limit		ainit	input	Global	mps
tbdwt(15)	Weight overboard table for center of gravity tables	kg	athrev	output	Global	bodyf
tbias(6)	Bias between simulated time and reported time (used for branch option)	sec	athrev	output input	Global	mult
temp	Temporary variable				Local	
temp1	Temporary variable				Local	
temp2	Temporary variable				Local	
tg	Time of acceleration limit	sec			Local	
tglim(10,15)	Time of acceleration limit	sec	athrev	output	Global	mps
thr	Vacuum thrust of fixed engines	nt			Local	
throt(15)	Throttle table for MPS motor		ainit	input	Global	mps

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
thrt	Throttle value of thrust	nt			Local	
time(2,16)	Actual times for thrust events measured from launch	sec	aforun	output input	Global	agen1
timmps(15)	Time table for MPS motor	sec	ainit	input	Global	mps
tl	Trigger time for storage of state during forward integration	sec	athrev	output	Global	agen1
tne(6,15)	Array which defines the number of engines per thrust event		ainit	input	Global	agen1
tpoly	Time to begin booster polynomial attitude control	sec	athrev	output	Global	agen2
tq	Time that the minh phase is initiated	sec	athrev	output	Global	agen1
tsavet	Time storage variable for restart	sec	athrev	output	Global	jumpst
tv1	Time to activate print trigger	sec	athrev	output	Global	forint
tv12	Time to activate weight drop event	sec	athrev	output	Global	forint
tv13	Time to activate booster MPS throttle trigger	sec	athrev	output	Global	forint
tv2	Time to activate thrust events	sec	athrev	output	Global	forint
tv9	Time to activate acceleration limit trigger	sec	aforun	output input	Global	forint
txcg(15)	Vertical center of gravity table	m	athrev	output	Global	bodyf
tycg(15)	Longitudinal center of gravity table	m	athrev	output	Global	bodyf
tzero	Time of simulation initiation	sec	ainit	input	Global	agen1
t_liftoff	Time of lift-off	sec	ainit	input	Global	t_liftoff
var(25)	State variable array			input output	Global	forint
vchstg	Characteristic velocity at staging	m/sec		output	Global	vchstg
viv(25)	Initial condition array for jump start		athrev	output	Global	agen2
vrcut	Magnitude of characteristic velocity cutoff	m/sec	ainit	input	Global	agen1
wddump(15)	Amount of weight dumped during OMS/RCS thrust events	kg	ainit	input	Global	omsrcs
wdoms(15)	Weight flowrate of OMS engines	lbs/sec	ainit	input	Global	omsrcs
wdot	MPS propellant flowrate	kg/sec			Local	
wdott	Throttled propellant flowrate	kg/sec			Local	
wdrcs(15)	Weight flowrate of RCS engines	lbs/sec	ainit	input	Global	omsrcs
wf(5)	Predicted final mass of stage	kg		output input	Global	mult

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
wg	Mass at acceleration limit	kg	athrev	output input	Global	fmxx
wg1	Projected mass at acceleration limit in mth thrust event	kg			Local	
wglim	Masses where acceleration limits occur	kg			Local	
wjet(15)	Jettison weight per thrust event	kg	ainit	input	Global	agen1
woms	Propellant weight of OMS engines	kg			Local	
woorb	Initial weight of orbiter	kg			Local	
wpmx	Total amount of usable propellant	kg	athrev	output	Global	ipr
wres	Residual weight	kg			Local	
wt1	Vehicle mass at mth thrust event	kg			Local	
wy	MPS propellant weight	kg			Local	
wzerst	Stored value of wzero	lbs	athrev	output	Global	jumpst
w_margin(2)	Required value of weight margin at the end of selected stage	kg	anewch	input	Global	multi
xgpa	Vertical component of gimbal points of equivalent booster engines	m	athrev	output input	Global	avggp
xinc	Inclination angle	rad	aprtn	input	Global	afprn
xlen	Reference length	m	athrev	output	Global	bodyf
xm	Current vehicle mass	kg	athrev	output	Global	agen2
xm1	Vehicle mass	kg			Local	
xmaug	Auxiliary vehicle mass	kg	athrev	input output	Global	agen1
xmaug_save	Saved value of auxiliary mass used for branch trajectory option	kg	athrev	output	Global	xmaug_save
xmd(15)	Propellant flowrate	kg/sec	ainit	input	Global	xtra
xmdot	Propellant flowrate	kg/sec			Local	
xmiad	Continuous portion of vehicle mass	kg	ader1	input output	Global	agen2
xmtj	Total flowrate of jth thrust event	kg/sec			Local	
xmtl	Total flowrate of lth thrust event	kg/sec			Local	
xnod	Angle between launch longitude and descending node	rad	aprtn	input	Global	afprn
xref	Vertical component of moment reference point	m	athrev	output	Global	bodyf
ygpa	Longitudinal component of gimbal point of booster equivalent engine	m	athrev	output input	Global	avggp

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
yref	Longitudinal component of moment reference point	m	athrev	output	Global	bodyf
zgpa	Lateral component of gimbal point of booster equivalent engines	m	athrev	output input	Global	avggp

```
      SUBROUTINE ATHREV                                                 ATHR   1

C      THRUST EVENT CONTROL ROUTINE;SETS UP VEHICLE GEOMETRY,AERODYNAMICS,
C     NUMBER OF ENGINES,THRUST LEVELS,FLOWRATES,TRIGGER FLAGS

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      LOGICAL GLIMIT,VISC,branch_point,mombal

      integer engines

      CHARACTER*60 HEAD
      CHARACTER*6 MISION,mission
      character*12 HEADER
      character*7 system

      COMMON/AGEN1/TIME(2,16),TAUT(15),TAUW(15),TZERO,TLIFT,TTILT,TMINH,
     *DTZ,TQ,TL,XMAUG,TNE(6,15),ENGDAT(8,15),S(15),WD(15),WJET(15),
     *HEAD,PRINT(15),STEP(15),HNOM(15),HMAX(15),PROP(6),TBEGR,TENDR,
     *CHRDOT,CHIRO,FAZ,BO(3,2),CBAXIL,TCHIR,VRCUT,FPRFAC,CHVEL

      COMMON/AGEN2/AZ,SA,CA,ALF,ALFY,CHIP,SCHIP,CCHIP,CHIY,SCHIY,CCHIY,
     *CHIR,SCHIR,CCHIR,STH,CTH,STHL,CTHL,DPHIZ,RTHE,R,VR,XM,SW,SU,SV,Q,
     *VIV(25),DVARS(13),DELXDW(15,3),DELXDR(7,5),DELXD(15,7),WZERO,ALT,
     *XMACH,QDOT,FAA,FAN,A(3,3),CASE,CT2,UMF,A12W,A22W,A32W,XJEXT,BEU,
     *BYL,BEL,BHMAX(15),BHMN,BSTEP(15),HNMB,HMXB,TPOLY,XMIAD

      COMMON/AGEN3/ITHR,IWD,IXR,JUMP,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISION,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)

      COMMON/AGEN5/CHIBAS,CHPBAS,CHYBAS,CHISAV,TXX,TYY,TZZ,AMX,AMY,AMZ,
     *TMZ,THMFA,SIDE,WMAG,AZW,FANC,FLATC,FOM,VISC

      COMMON/AFPRN/VI,GAM,C1,C3,XINC,XNOD,XMLB,THETG,THETA,PHI,RNGO,RNGA
     *N,AZI,AZRE

      common/astor/astor(4,67)

      COMMON/AUTO/KDB(40),SAVCP(40),XLAMB(40,20),DP2

      COMMON/AVGGP/XGPA,YGPA,ZGPA

      COMMON/BGEN3/NOSS,JTB,NENT,NACT,NOM1,NP(7)

      COMMON/BODY/ATBDWT(15,2),ATXCG(15,2),ATYCG(15,2),AXLEN(2),AXREF(2)
     *,AYREF(2),KPI(2),KYI(2),API(10,2),AYI(10,2)

      COMMON/BODYF/TBDWT(15),TXCG(15),TYCG(15),XLEN,XREF,YREF,M1,M2,M5,
     *M6,M7,M8,M9,M50,M51,MSRW,AP(10),AY(10)

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM

      COMMON/CONTRO/TTBL(30),CPTBL(30),CYTBL(30),TOBL(210),
```



```
     *CPOTBL(210),CYOTBL(210)

      COMMON/DELFAB/DELALT(25),DELFAB(25),M30,M31,ITOL(15)

      COMMON/DELX/DELX(7)

      common/delxdb/delxdb(7)

      COMMON/DELXH/DELXH(5)

      COMMON/ELEVON/EMACH(25,2),ELEVON(25,4),M10,M11

      common/engines/engines(15,15)

      COMMON/ENPUT/PSIRST(6,5),HEADER(20),NVRST(5),KCDRES(6,5),LAST

      COMMON/FMXX/FMXX,MSW,WG,SPOA

      COMMON/FORINT/HBANK(2),T,TT,VAR(25)                    ,DVAR(25),FSAVE
     *(625),NTRG1,TV1,NTRG2,TV2,NTRG3,TV3,NTRG4,TV4,NTRG5,TV5,NTRG6,TV6,
     *NTRG7,TV7,NTRG8,TV8,NTRG9,TV9,NTRG10,TV10,NTRG11,TV11,NTRG12,TV12,
     *NTRG13,TV13,KIND,KRDER,EU,EL,AYL,HMX,HMN,HNM

      COMMON/IPR/IPR,WPMX,ILAST,KDT(15),PRK4,PRTOT,iprop(6),prk5(5)

      COMMON/IRTAOA/IRTAOA

      COMMON/ISTART/JSTART(6),ISTART(2),STARTV(40,2),SVXINC,SVXNOD,SVAZRE

      COMMON/ITHRO/ITHRO

      COMMON/JT/JT

      common/jthrot_save/jthrot_save(15)

      COMMON/JUMPST/JUMPST,TSAVET,WZERST

      COMMON/KP/KP,KY

      COMMON/LOADS/QLOAD(26),QTIME(26),QVALUE(13),SAVE(20,13)

      COMMON/M20/M17,M20,M21,M22,M23,M24

      COMMON/LBM/TIMLBM(30),THRLBM(30),WDTLBM(30),PLBM(2),MLBM1,MLBM2,
     *ALBM(2),BLBM(2),CLBM(2),VLBM(2)

      common/mission/mission(2)

      common/mombal/mombal

      COMMON/MPS/TIMMPS(15),FTBL(15),THROT(15),GLIM(15),TAUALT(5),
     *TAUTBL(5),MPS1,MPS2,TGLIM(10,15),GLIMIT,tau_const(10,5),jthrot(15)

      common/mult/nchiot(6),tbias(6),nstage,nstg(15),wf(5),ibranch

      common/multi/char_vel(5),fpr_fac(5),w_margin(2)
```

```
      COMMON/OLOW/OLOW,WINT
      COMMON/OMSRCS/FOMS(15),FRCS(15),WDOMS(15),WDRCS(15),WDDUMP(15)
      COMMON/REST/JSTR,NCNRS(5),NTCN,TR(5),LSB,NF8,NOPAR,WIBT(15)
      COMMON/RESTX/RESTX(10,5)
      COMMON/RICONT/TTABLE(20),SIGMA,SIGTAB(20),M15,M16
      common/save_mass/save_mass
      COMMON/SRM/SRMTTB(600),SRMFTB(600),SRMWDT(600),SRMAE(600),MSRB
      common/system/system
      COMMON/TABLK/MCON(7)
      common/t_liftoff/t_liftoff
      COMMON/VAERO/VAEROD(10,19),M18,M19,MBS18,MBS19
      COMMON/VCHSTG/VCHSTG
      COMMON/XTRA/XMD(15),CFR(15),F(15),CISP(15)
      common/xmaug_save/xmaug_save
      DIMENSION ANE(2),lvrst(5),wglim(10)
C*****   THIS ROUTINE TAKES SPECIAL FORINT COMMON*****                ATHR  63
      IF(ITHR.NE.JUMP-1) CALL APRTN(1)                                 ATHR  65
   10 JT=ITHR                                                          ATHR  66
      IPRT=5                                                           ATHR  74
      IF(JT.EQ.JUMP-1)T=TZERO                                          ATHR  75
      ITHR=ITHR+1                                                      ATHR  77
      IF(ITHR.ne.1) then
         IF(TNE(3,ITHR).GT.0..AND.TNE(3,ITHR-1).LT.1.E-05) IPRT=13
         IF(TNE(3,ITHR).LT.1.E-05.AND.TNE(3,ITHR-1).GT.0.) IPRT=12
         if(tne(4,ithr).ne.0..and.tne(4,jt).lt.1.e-05) iprt=23
      endif
      L=1
      IF(ITHR.LT.ICONSW) L=2
      xgpa=0.
      ygpa=0.
      zgpa=0.
      if(mombal) then
```



```
         do i=1,15
            if(nsyst(i).eq.1.and.engines(i,ithr).eq.1) then
               XGPA=XGPA+ENGDAT(3,I)
               YGPA=YGPA+ENGDAT(1,I)
               ZGPA=ZGPA+ABS(ENGDAT(2,I))
            endif
         enddo
         if(tne(1,ithr).eq.0.) then
            xgpa=0.
            ygpa=0.
            zgpa=0.
         else
            xgpa=xgpa/tne(1,ithr)
            ygpa=ygpa/tne(1,ithr)
            zgpa=zgpa/tne(1,ithr)
         endif
      endif
      NTRG13=-2
      IF(MPSFLG(ITHR).eq.2.or.mpsflg(ithr).eq.3) then
         NTRG13=2
         DO I=1,15
            IF(TIMMPS(I).GT.T) GO TO 209
         enddo
  209    ITHRO=I
         TV13=TIMMPS(ITHRO)
      endif
      char_vel(nstage)=var(15)-var(16)-var(17)
      IF(ITHR .EQ. 16) GO TO 786
      TIME(1,ITHR)=T
      TIME(1,ITHR+1)=T+TAUT(ITHR)
      IF(NWVNT.gt.0) then
         DO I=1,NWVNT
            J=NOWD(I)
            TIME(2,I)=TIME(1,J)+TAUW(I)
```

```
         enddo

         TV12=TIME(2,IWD+1)

      endif

      IF(ITHR.EQ.1) GO TO 785

      IF(ITHR.eq.IPOLY) then

         TPOLY=T

         KP1=KPI(1)

         IF(KDB(KP1+18).EQ.0.AND.KDB(KP1+19).EQ.1) CPTBL(KP1+2)=
     *   ATAN2(SCHIP,CCHIP)*RAD

         KP=KPI(2)
         KY1=KYI(1)

         IF(KDB(KY1+25).EQ.0.AND.KDB(KY1+26).EQ.1) CYTBL(KY1+2)=
     *   ATAN2(SCHIY,CCHIY)*RAD

         KY=KYI(2)
         KP1=KP1+2
         KY1=KY1+2

         CALL FIND(CPTBL(KP1),AP,TTBL(KP1),KP)
         CALL FIND(CYTBL(KY1),AY,TTBL(KY1+15),KY)

         DO I=1,10
            API(I,2)=AP(I)
            AYI(I,2)=AY(I)
         enddo

      endif

      branch_point=.false.

      if(nstg(jt).ne.nstg(ithr)) then

         nstage=nstg(ithr)

         if(jstart(nstage).ne.0) then

            do k=1,5

               lvrst(k)=0

               if(nvrst(k).eq.jt) then

                  do j=1,7
                     delxd(jt,j)=dvar(j)
                     delxdr(j,k)=dvar(j)
                     restx(j,k)=var(j)
                  enddo
```



```
                  restx(7,k)=xmiad+xmaug
                  restx(8,k)=t
                  restx(9,k)=var(14)
                  restx(10,k)=0.
                  lvrst(k)=1

                  branch_point=.true.

               endif
            enddo

            tbias(nstage)=0.

            save_mass=xmiad+xmaug

            call aprtn(15)

            ithr=ithr-1
            t1=0.
            call aeosr
            ithr=ithr+1

            J=4*(nstg(jt)-1)+1

            DO I=1,13

               SAVE(J,I)=QTIME(I)
               SAVE(J+1,I)=QLOAD(I)
               QLOAD(I)=1000000000.
               QTIME(I)=t
               K=I+13
               SAVE(J+2,I)=QTIME(K)
               SAVE(J+3,I)=QLOAD(K)
               QLOAD(K)=-10000000000.
               QTIME(K)=t

            enddo

            j=jstart(nstage)

            do k=1,25
               var(k)=startv(k,j)
            enddo

            do k=1,7
               dvar(k)=startv(k+31,j)
            enddo

            tbias(nstage)=t-startv(26,j)

            if(nstage.le.5) then

               do i=nstage+1,6
                  tbias(i)=tbias(nstage)
               enddo
```

```
            endif

            olow=startv(30,j)
            xmaug_save=xmaug
            xmaug=startv(31,j)
            xmiad=startv(27,j)-xmaug
            var(7)=xmiad
            mision=mission(2)

            do i=1,7
               mcon(i)=1
            enddo

            kcytab=kcytab+1

            tl=0.

            call aeosr

            kcytab=kcytab-1

            iprt=4

            go to 205

         endif
      endif

      IF(JUMP.GT.NOEVNT(1).AND.ITHR.EQ.JUMP) GO TO 7979

      IF(LSTGE(ITHR).NE.LSTGE(ITHR-1)) GO TO 7979

      GO TO 786

 7979 CONTINUE

      IPRT=21
      IF(ITHR.EQ.JUMP) IPRT=3

  205 IF(LSB.ne.1) then

         do i=1,nf8
            viv(i)=var(i)
         enddo

         viv(7)=0.
         TSAVET=T
         SCHI=CHISAV
         SYCHI=CHIY
         JUMPST=ITHR

      endif

  785 LSTG=LSTGE(ITHR)                                              ATHR 171
```



```
      IF(ITHR.EQ.NOEVNT(1)+1) VCHSTG=VAR(15)-VAR(16)-VAR(17)

      do i=1,7
         MCON(i)=1
      enddo

      M1=1                                                          ATHR 174
      M2=0                                                          ATHR 175
      M5=1                                                          ATHR 176
      M6=0
      M7=1                                                          ATHR 177
      M8=1                                                          ATHR 178
      M9=0
      M10=1
      M11=1
      M15=1
      M16=1
      M17=1
      M18=1
      M19=0
      M20=1
      M21=1
      M22=1
      M23=1
      M24=1
      M30=1
      M31=0
      ETFUEL=0.
      ANE(1)=0.
      ANE(2)=0.
      MSRB=1
      M50=1
      M51=1
      MSRW=0
      MLBM1=1
      MLBM2=0
      MPS1=1
      MPS2=1

      XLEN=AXLEN(LSTG)                                              ATHR 182
      XREF=AXREF(LSTG)                                              ATHR 183
      YREF=AYREF(LSTG)                                              ATHR 184

      do i=1,15
         TBDWT(I)=ATBDWT(I,LSTG)*PTKG
         TXCG(I)=ATXCG(I,LSTG)
         TYCG(I)=ATYCG(I,LSTG)
      enddo

  786 CONTINUE                                                      ATHR 194

      IF((NMAX.eq.0.and.JUMP.eq.1).and.(nstg(jt).ne.nstg(ithr)).and.
     * jstart(nstage).eq.0) then

         J=4*(nstg(jt)-1)+1
```

```
          do i=1,13
             SAVE(J,I)=QTIME(I)
             SAVE(J+1,I)=QLOAD(I)
             QLOAD(I)=1000000000.
             QTIME(I)=T
             K=I+13
             SAVE(J+2,I)=QTIME(K)
             SAVE(J+3,I)=QLOAD(K)
             QLOAD(K)=-10000000000.
             QTIME(K)=T
          enddo

       endif

c      IF(ITHR .EQ. 16) GO TO 40

       NTRG2=2

       TV2=T+TAUT(ITHR)

       if(jump.eq.1.and.tzero.gt..1.and.ithr.eq.jump) then

          tv2=t_liftoff+taut(ithr)

       endif

       NTRG9=-2
       IF(MPSFLG(ITHR).EQ.4.or.mpsflg(ithr).eq.5) NTRG9=2

       glimit=.false.

       HNM=STEP(ITHR)                                          ATHR 262
       HMX=PRINT(ITHR)                                         ATHR 264

       TV1=AINT(T+1.E-03)+PRINT(ITHR)

       IF(ITHR.NE.JUMP) then
          XMAUG=XMAUG-WJET(JT)
          if(jt.eq.noevnt(1).and.abs(var(9)).gt.0.) then
             xmaug=xmaug-var(9)
             var(9)=0.
          endif
       endif

       if(lsb.eq.0) then
          do i=1,10
             tglim(i,ithr)=0.
          enddo
          jthrot_save(ithr)=0
       endif

       jthrot(ithr)=0

       do i=1,15
          IF(nsyst(i).eq.1.and.engines(i,ithr).ne.0) GO TO 39
       enddo
```



```
   39 nmps=i

      IF(NTRG9.gt.0) then

         THR=THROT(ITHR)*(TNE(1,ITHR)+tne(6,ithr))*
     *       ENGDAT(6,NMPS)*PTN
         WDOT=THR/(GZERO*FUNISP(THROT(ITHR),nstage))

         TV9=T+(XMIAD+XMAUG THR/(GZERO*GLIM(ITHR)))/WDOT

         IF(TV9.GT.t) then

            tglim(1,ithr)=tv9
         else

            NTRG9=-2

            tv9=0.

         endif

      endif

      if(jt.eq.noevnt(1)) then

         woorb=xmiad+xmaug
         if(lsb.ne.1) wzerst=woorb
         olow=woorb

      endif

      if(ithr.eq.istart(1).or.ithr.eq.istart(2)) then

         J=2
         IF(ITHR.EQ.ISTART (1)) J=1

         do i=1,25
            STARTV(I,J)=VAR(I)
         enddo

         STARTV(7,J)=0.
         STARTV(26,J)=T
         STARTV(27,J)=XMIAD+XMAUG
         STARTV(28,J)=CHIP
         STARTV(29,J)=CHIY
         STARTV(30,J)=OLOW
         startv(31,j)=xmaug

         do i=1,7
            startv(i+31,j)=dvar(i)
         enddo

         if(j.eq.1) iprt=21

         ibranch=jt
```

```
      endif

      IF(NMAX.EQ.0.AND.JT.EQ.NCOAST.AND.NCOAST.NE.0) GO TO 116

      IF(JT.EQ.NVNT) GO TO 9
C                                                                     ATHR 282
C***  CHECK FOR FIRST POINT                                           ATHR 283

      IF(JUMP.le.NOEVNT(1).or.ITHR.ne.JUMP) then

    1    IF(JT.EQ.JUMP-1)RETURN

         IF(ITHR.EQ.NENDCT(KCYTAB)) KCYTAB=KCYTAB+1

      endif

      do i=1,7
         DVARS(I)=DVAR(I)
      enddo

      DO K=1,5

         IF(NVRST(K).eq.JT.and.lvrst(k).ne.1) then

            IF(JT.GT.NOEVNT(1)) IPRT=20

            IF(JUMP.EQ.1.AND.JT.GE.MINH) IPRT=24

            DO I=1,7
               DELXDR(I,K)=DVAR(I)
               RESTX(I,K)=VAR(I)
            enddo

            RESTX(7,K)=XMIAD+XMAUG+WJET(JT)
            RESTX(8,K)=T
            RESTX(9,K)=VAR(14)
            RESTX(10,K)=0.

            DO I=1,6
               IF(KCDRES(I,K).EQ.16) DELXH(K)=DVAR(14)
            enddo

         endif

         lvrst(k)=0
      enddo

      IF(NVRST(1).EQ.0.AND.JT.EQ.IRTAOA) IPRT=21

      CALL ADER1                                                      ATHR 313

      CALL ADER                                                       ATHR 314

      if(.not.branch_point) then

         do i=1,7
```



```
            delxd(jt,i)=dvars(i)-dvar(i)
         enddo

      else

         do i=1,7
            delxdb(i)=dvars(i)-dvar(i)
         enddo

      endif

      IF(JUMP.GT.NOEVNT(1).AND.ITHR.EQ.JUMP) GO TO 103
C                                                                     ATHR 317
C***  CALCULATE PERF RES AND TAUT(IPR)                                ATHR 318
C                                                                     ATHR 319
      if(iprop(nstage).eq.ithr.and.tne(1,ithr).ne.0.) then

         if(system.eq.'STAGE15') then

            wres=(astor(1,15)+astor(1,16)+astor(1,17)+astor(1,18)+
     *           astor(1,20))*ptkg

         else

            wres=(astor(1,17)+astor(1,18)+astor(1,24)+astor(1,26)+
     *           astor(1,28)+astor(1,34))*ptkg

         endif

         if(nchiot(nstage).eq.0) then
            wf(nstage)=(xmiad+xmaug+wres+w_margin(1)-var(8))*
     *                 prk5(nstage)
         else
            wf(nstage)=(xmiad+xmaug+wres+w_margin(2)-var(8))*
     *                 prk5(nstage)
         endif

         thr=throt(ithr)*(tne(1,ithr)+tne(6,ithr))*
     *        engdat(6,nmps)*ptn
         wdot=thr/(gzero*funisp(throt(ithr),nstage))

         wglim(1)=thr/(gzero*glim(ithr))

         if(wglim(1).gt.wf(nstage).and.wglim(1).lt.xmiad+xmaug) then

            ithrot=1
            tg=(xmiad+xmaug-wglim(1))/wdot
            taut(ithr)=0.

            if(mpsflg(ithr).eq.5) then
94             thrt=tau_const(ithrot,nstage)*thr
               wdott=thrt/(gzero*funisp(tau_const(ithrot,nstage),
     *                nstage))
               taut(ithr)=taut(ithr)+tg
               wglim(ithrot+1)=thrt/(gzero*glim(ithr))
```

```
            if(wglim(ithrot+1).gt.wf(nstage)) then

               ithrot=ithrot+1
               tg=(wglim(ithrot-1)-wglim(ithrot))/wdott
               go to 94

            else

               taut(ithr)=taut(ithr)+(wglim(ithrot)-wf(nstage))/
     *                     wdott

            endif

         endif

      else

         taut(ithr)=(xmiad+xmaug-wf(nstage))/wdot

      endif

      tv2=t+taut(ithr)

      do i=ithr,nvnt
         time(1,i+1)=time(1,i)+taut(i)
      enddo

   else if(iprop(nstage).eq.ithr.and.tne(4,ithr).ne.0.) then

      deltav=astor(3,5)*.3048
      dvisp=astor(4,5)
      temp=exp(-deltav/(gzero*dvisp))

      taut(ithr)=(var(11)-astor(1,4)*ptkg-(xmiad+xmaug)*
     *           (1.-temp))/(wdoms(ithr)*tne(4,ithr)*ptkg*temp)

      tv2=t+taut(ithr)
      if(taut(ithr).lt.0.) then
         write(31,*) ' taut =',taut(ithr)
         write(31,*) ' var(11)=',var(11)*pfkg
         write(31,*) ' astor(1,4)=',astor(1,4)
         write(31,*) ' xmiad+xmaug=',(xmiad+xmaug)*pfkg
         write(31,*) ' temp=',temp
         taut(ithr)=0.01
         tv2=t+.01
      endif
      do i=ithr,nvnt
         time(1,i+1)=time(1,i)+taut(i)
      enddo

   endif

   if(tv9.gt.tv2) then

      ntrg9=-2
      tglim(1,ithr)=0.
```



```
      tv9=0.

   endif

   IF(IRTLS.EQ.0.OR.ITHR.NE.IRTLS) GO TO 102

   XM=XMIAD+XMAUG
   XM1=XM
   WY=PROP(1)*PTKG
   WOMS=PROP(4)*PTKG
   IRTLS1=IRTLS-1
   K=NOEVNT(1)+1

   DO I=K,IRTLS1
      WOMS=WOMS-TAUT(I)*(WDOMS(I)+WDDUMP(I))
      WY=WY+TNE(1,I)*CFR(I)*TAUT(I)
   enddo

   J=IRTLS
   L=IRTLS+1

   IF(MSWCH(L).ne.1) then

      DO I=L,NVNT

         IF(MSWCH(I).EQ.1) GO TO 149

         WY=WY+TNE(1,I)*CFR(I)*TAUT(I)
         XM1=XM1-TAUT(I)*(TNE(1,I)*XMD(I)+WDOMS(I)+WDDUMP(I)+
     *          WDRCS(I))
         WOMS=WOMS-TAUT(I)*(WDOMS(I)+WDDUMP(I))
      enddo
149   L=I

   endif

   WG=(TNE(1,L)*F(L)+FOMS(L)+FRCS(L))/(GZERO*GLIM(L))
   XMTJ=TNE(1,J)*XMD(J)+WDOMS(J)+WDRCS(J)+WDDUMP(J)
   XMTL=TNE(1,L)*XMD(L)+WDOMS(L)+WDRCS(L)+WDDUMP(L)

   TAUT(J)=(XMTL*WOMS-WDOMS(L)*(XM1-WG))/(XMTL*(WDOMS(J)+WDDUMP(J))-
  *        WDOMS(L)*XMTJ)

   N=J+1
   M=L-1
   WT1=XM-XMTJ*TAUT(J)

   IF(N.le.M) then

      DO I=N,M
         WT1=WT1-(TNE(1,I)*XMD(I)+WDOMS(I)+WDRCS(I)+WDDUMP(I))*
     *          TAUT(I)
      enddo

   endif
```

```
      WG1=(TNE(1,M)*F(M)+FOMS(M)+FRCS(M))/(GZERO*GLIM(L))

      IRTFLG=0

      IF(WT1.le.WG1) then

         IRTFLG=1
         WT1=WG1
         TAUT(J)=XM-WG1

         IF(N.le.M) then

            DO I=N,M
               TAUT(J)=TAUT(J)-(TNE(1,I)*XMD(I)+WDOMS(I)+WDRCS(I)+
     *              WDDUMP(I))*TAUT(I)
            enddo
         endif

         TAUT(J)=TAUT(J)/XMTJ

      endif

      TG=(WT1-WG)/XMTL

      WY=WY+TNE(1,J)*CFR(J)*TAUT(J)+TNE(1,L)*CFR(L)*TG
      TV2=T+TAUT(J)
      SPOA=CISP(L)/GLIM(L)

      IF(IPR.eq.0) then

  102    IF(ITHR.NE.IPR.OR.IPR.EQ.0.OR.IRTLS.NE.0) GO TO 103

         XM=XMIAD+XMAUG
         TEMP=WPMX-PRK4*XM*CFR(IPR)/XMD(IPR)
         K=NOEVNT(1)+1

         IF(GLIM(IPR).GT.5.) GO TO 105

         SPOA=CISP(IPR)/GLIM(IPR)
         WG=THR/(GLIM(IPR)*GZERO)
         TG=(XM-WG)/XMDOT
         WY=TNE(1,IPR)*CFR(IPR)*TG
         IPRP=IPR-1
         WY=WY+ETFUEL

         DO JA=K,IPRP
            WY=WY+TNE(1,JA)*CFR(JA)*TAUT(JA)
         enddo

         IF(WY.GE.WPMX) GO TO 105

      endif

      TEMP1=1.

      IF(MSWCH(IPR+1).EQ.2) TEMP1=EXP(-VRCUT/(GZERO*CISP(IPR+1)))
```



```
      TEMP2=WPMX-WY-CFR(IPR)/XMD(IPR)*WG-TNE(1,IPR+2)*XMD(IPR+2)*
     *       TAUT(IPR+2)*(1.-PRK4)

      TAUT(IPR)=TG+SPOA*LOG(WG*(PRK4*TEMP1-CFR(IPR)/XMD(IPR)+
     *          (1.-TEMP1))/TEMP2)

      FMXX=SPOA/TEMP2

      MSW=1

      TIME(1,IPR+1)=TIME(1,IPR)+TAUT(IPR)

      PRTOT=PRK4*(WG*EXP(-GLIM(IPR)*(TAUT(IPR)-TG)/CISP(IPR))*TEMP1-
     *       TNE(1,IPR+2)*XMD(IPR+2)*TAUT(IPR+2))

      IF(WY+PRTOT.GE.WPMX) GO TO 105

      GO TO 106

  105 CONTINUE

      MSW=0
      IPRP=IPR-1
      TEMP=TEMP-ETFUEL

      DO JA=K,IPRP
         TEMP=TEMP-TNE(1,JA)*CFR(JA)*TAUT(JA)
      enddo

      TEMP1=TEMP/(1.-PRK4)
      TAUT(IPR)=TEMP1/(ANE(1)*CFR(IPR))
      TIME(1,IPR+1)=TIME(1,IPR)+TAUT(IPR)
      PRTOT=PRK4*(XM-ANE(1)*XMD(IPR)*TAUT(IPR))

  106 CONTINUE

      IF(ITHR.EQ.IPR) TV2=T+TAUT(IPR)
C                                                                       ATHR 365
C***  PERF RESERVES AND TAUT(IPR) CALCULATED                            ATHR 366
C                                                                       ATHR 367
  103 CONTINUE                                                          ATHR 368

      IF(MSWCH(ITHR).eq.2) then

         TAUT(ITHR)=(XMIAD+XMAUG)*(1.-EXP(-VRCUT/(GZERO*CISP(ITHR))))/
     *              (XMD(ITHR)*TNE(1,ITHR))
         TIME(1,ITHR+1)=TIME(1,ITHR)+TAUT(ITHR)
         TV2=T+TAUT(ITHR)

      endif

      IF(IPRT.eq.21) then
C
C     SAVE INCLINATION AND NODE FOR AOA TRAJECTORY CONSTRAINTS FOR
C     VARIABLE IY TARGETING
```

```
C
         SVXINC=XINC
         SVXNOD=XNOD
C
C     SAVE EARTH RELATIVE AZIMUTH(HEADING ANGLE) FOR RTLS CONSTRAINT
C
         SVAZRE=AZRE

      endif

      IF(LSB.EQ.0) CALL APRTN(IPRT)

      IF(ITHR.NE.MINH) RETURN

      IF(LSB.ne.1) then

         DO I=1,7
            DELX(I)=VAR(I)
         enddo

      endif

      TQ=T

      ks1=2

      CALL ADER1

      CALL ADER

      CALL APRTN(10)

      IF(JUMP.EQ.1) CALL RTMRK

      RETURN

    9 CONTINUE                                                    ATHR 372

      IF(LSB.ne.1) then

  116    IF(NMAX.eq.0.and.JT.eq.NCOAST) then

            XM=XMIAD+XMAUG

            CALL ADER1

            CALL ADER

            CALL APRTN(11)

            IF(ITHR.EQ.ISTART(1)) STARTV(22,1)=XM
            IF(ITHR.EQ.ISTART(2)) STARTV(22,2)=XM

            CALL RTMRK

            RETURN
```



```
         endif

         TL=0.

         CALL AEOSR

         CALL APRTN(15)

         DO I=1,7
            DELXD(JT,I)=DVAR(I)
         enddo

      endif

      THR=0.                                                      ATHR 378
      XMDOT=0.                                                    ATHR 379

      IF(LSB.EQ.0) OCOW=XMIAD+XMAUG

      XMAUG=XMAUG-PRTOT
      XM=XMIAD+XMAUG                                              ATHR 381

      CALL ADER1                                                  ATHR 382

      CALL ADER                                                   ATHR 383

      CALL APRTN(1)                                               ATHR 384

      CHVEL=VAR(15)-VAR(16)-VAR(17)

      CALL RTMRK                                                  ATHR 386

      RETURN                                                      ATHR 387
      END                                                         ATHR 388
```

3.18 Subroutine ATILT

3.18.1 Purpose

This subroutine provides trigger routines for time related events directly after liftoff during the forward trajectory integration. The additional trigger routines AQMAX, AXPRT, GLIMT, AWDEV, ALIFT, ATHRO, AROLL,and AJUMP are included as entry points in this subroutine to provide various output functions. These routines have the following functions:

- ATILT - Controls parameters and prints at tilt-over events;
- AQMAX - Prints trajectory block at maximum dynamic pressure;
- AXPRT - Print trajectory block at print intervals;
- GLIMT - Calculates parameters and prints trajectory block at acceleration limits;
- AWDEV - Calculates parameters and prints weight drop events;
- ALIFT - Prints trajectory block when thrust-to-weight is equal to 1 at lift-off;
- ATHRO - Calculates parameters and prints trajectory block at MPS throttle events;
- AROLL - Calculates parameters and prints trajectory block at roll maneuver events; and
- AJUMP - Writes output file at first stage jump start points.

3.18.2 Variable Listing

3.18.3 Subroutines Called:

Name	Function
ADER	Calculates the time independent portion of the equations of motion during forward integration
ADER1	Calculates the time dependent portion of the equations of motion during forward integration

AEOSR	Stores state variables during forward integration
APRTN	Prints trajectory block output and files
FIND	Calculates attitude polynomials
FUNISP	Calculates specific impulse based on throttle level

3.18.4 Calling Subroutines:

Name	Function
DPIR	Integration routine

3.18.5 Fortran Listing

atilt

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ap(10)	Coefficients for booster pitch attitude		atilt	input	Global	bodyf
api(10,2)	Coefficients for booster pitch attitude		atilt	output	Global	body
ay(10)	Coefficients for booster yaw attitude		atilt	input	Global	bodyf
ayi(10,2)	Coefficients for booster yaw attitude		atilt	output	Global	body
chip	Pitch attitude angle	rad	ader1	output input	Global	agen2
chir	Roll attitude angle	rad	ader1	input	Global	agen2
chirf	Final roll attitude angle after roll maneuver	rad			Local	
chiro	Initial value of roll attitude angle	rad	atilt	output input	Global	agen1
chpbas	Pitch attitude bias angle	rad	ader1	input	Global	agen5
chrdot	Roll attitude rate	rad/sec	atilt	output input	Global	agen1
chrdsv	Saved value of roll rate				Local	
chybas	Yaw attitude bias angle	rad	ainit	input	Global	agen5
cptbl(30)	Booster pitch attitude angle table	rad	athrev	output	Global	contro
cytbl(30)	Booster yaw attitude angle table	rad	atilt	output	Global	contro
delxdw(15,3)	State derivative discontinuity storage at weight grop events		atilt	output	Global	agen2
dvar(25)	Storage array for the state derivatives		ader	input	Global	forint
dvars(13)	Derivative storage at the thrust events		atilt	output input	Global	agen2
engdat(8,15)	Engine data matrix		ainit	input	Global	agen1
ftbl(15)	Thrust table for main proplusion system	lbs	ainit	input	Global	mps
glim(15)	Acceleration limit	g's	ainit	input	Global	mps
glimit	Logical variable indicating aceleration limit has been achieved		atilt(gli	output	Global	mps
gzero	Gravitational constant (9.80665 m/sec)	m/sec^	ainit	input	Global	const
hdchrd	Heads-up to head-down roll rate	rad/sec	ainit	input	Global	headup
hdmach	Mach number lower limit for second roll maneuver		ainit	input	Global	headup
hnm	Nominal step size	sec		output	Global	forint
ifact	Flag indicating the type of attitude used in the booster stage		atilt	input	Global	agen3
ithr	Thrust event index number		athrev	input output	Global	agen3

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ithro	Index for SSME throttle table		atilt(ath	output input	Global	ithro
iwd	Index for weight drop events		atilt	output input	Global	agen3
jthrot(15)	Index indicating number of times the acceleration limit has been reached within a given thrust event		atilt(gli	output input	Global	mps
jthrot_save(1	Saved value of jthrot		atilt	output	Global	jthrot_save
kp	Order of booster pitch attitude polynomial		atilt	output	Global	kp
kpi(2)	Order of booster pitch attitude polynomials		ainit	input	Global	body
ks1	Booster attitude indicator		atilt	output	Global	agen3
ky	Order of booster yaw attitude polynomial		atilt	output	Global	kp
kyi(2)	Order of booster yaw attitude polynomials		ainit	input	Global	body
lsb	Internal flag to indicate partial derivative runs			output	Global	rest
mpsflg(15)	Flag indicating the type of MPS thrust simulation		ainit	output	Global	agen3
njump	Number of jump start times to be saved			output input	Global	time_jump
nmax	Iteration counter		mastre	output	Global	agen3
nroll	Flag indicating booster roll program after launch		atilt	output	Global	agen3
nstage	Index number of current stage		athrev	input	Global	mult
ntau	Index used in conjunction with timmps to indicate a throttle event in the booster stage			output input	Global	taulim
ntilt	Flag indicating end of tiltover phase of flight		atilt	output	Global	ntilt
ntrg12	Weight drop event trigger for forward trajectory		atilt	output	Global	forint
ntrg13	MPS throttle trigger number		atilt	output	Global	forint
ntrg14	Trigger for roll maneuver		atilt	output	Global	trigc
ntrg3	Relative velocity cutoff trigger for forward trajectory		atilt	output	Global	forint

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ntrg4	Booster attitude trigger for forward trajectory		atilt	output	Global	forint
ntrg6	Maximum dynamic pressure trigger for forward trajectory		atilt	output	Global	forint
ntrg7	Liftoff trigger number		atilt	output	Global	forint
ntrg9	Acceleration limit trigger number		atilt	output	Global	forint
nwvnt	Number of weight drop events		ainit	input	Global	agen3
pi	Pi constant (3.14159265)		ainit	input	Global	const
psffm	Conversion for pounds per square feet from newtons per square meter		ainit	input	Global	const
ptn	Conversion from pounds to newtons	nt/lbs	ainit	input	Global	const
q	Dynamic pressure	nt/m^2	ader	input	Global	agen2
qflg	Logical variable indicating booster SSME throttle being simulated		atilt	output	Global	qmax
qpen	Maximum value of dynamic pressure	kg/m^2	atilt	output	Global	qmax
step(15)	Integration step size	sec	ainit	input	Global	agen1
t	Time from lift-off	sec	dpir	input	Global	forint
taulim	Rate of change of throttle limit	/sec	ainit	input	Global	taulim
tau_const(10,	Constant value of throttle used for constant throttle after acceleration limit		ainit	input	Global	mps
tbegr	Time to begin roll maneuver	sec	atilt	input output	Global	agen1
tbias(6)	Bias between simulated time and reported time (used for branch option)	sec	athrev	input	Global	mult
tendr	Time to terminate roll maneuver	sec	ainit	input output	Global	agen1
tg	Time of acceleration limits	sec			Local	
tglim(10,15)	Time of acceleration limit	sec	atilt	output	Global	mps
throt(15)	Throttle table for MPS motor		ainit	input	Global	mps
thrt	Vacuum thrust	nt			Local	
time(2,16)	Actual times for thrust events measured from launch	sec	aforun	input	Global	agen1
time_jump(5)	Time of jump start	sec	ainit	input	Global	time_jump
timmps(15)	Time table for MPS motor	sec	ainit	output input	Global	mps
timtau	Time for throttle event in first stage	sec	ainit	input	Global	taulim

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
tl	Trigger time for storage of state during forward integration	sec	atilt	output	Global	agen1
tlift	Time to begin tiltover maneuver	sec	ainit	input	Global	agen1
tne(6,15)	Array which defines the number of engines per thrust event		ainit	input	Global	agen1
ttilt	Time to begin tiltover maneuver	sec	atilt	input output	Global	agen1
tv1	Time to activate print trigger	sec	atilt	output	Global	forint
tv12	Time to activate weight drop event	sec	atilt	output	Global	forint
tv13	Time to activate booster MPS throttle trigger	sec	atilt	output input	Global	forint
tv2	Time to activate thrust events	sec	atilt	output	Global	forint
tv3	Value of relative velocity to activate trigger	m/sec	atilt	output	Global	forint
tv4	Time to activate booster attitude phase	sec	atilt	output	Global	forint
tv9	Time to activate acceleration limit trigger	sec	atilt	output	Global	forint
tzero	Time of simulation initiation	sec	ainit	input output	Global	agen1
u	Y component of plumbline inertial velocity vector	m/sec	dpir	output	Global	forint
v	Z component of plumbline inertial velocity vector	m/sec	dpir	output	Global	forint
wd(15)	Weight dropped during weight drop event	kg	ainit	input	Global	agen1
wdott	Propellant flowrate	kg/sec			Local	
wglim	Weight at acceleration limit	kg			Local	
wzero	Initial vehicle weight	lbs	ainit	input	Global	agen2
x	X component of plumbline position vector	m	dpir	output	Global	forint
xm	Current vehicle mass	kg	ader1	input	Global	agen2
xmach	Mach number		ader	input	Global	agen2
xmaug	Auxiliary vehicle mass	kg	athrev	input output	Global	agen1
xmi	Integrated vehicle mass less jettisoned inerts.	kg	dpir	input	Global	forint
xmiad	Continuous portion of vehicle mass	kg	ader1	input	Global	agen2
y	Y component of plumbline position vector	m	dpir	input	Global	forint

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
z	Z component of plumbline position vector	m	dpir	input	Global	forint
zap(18)	Additional integrated variables	variabl	desolv	output	Global	forint

```
      SUBROUTINE ATILT                                                  ATIL   1
C*********************************************************************
C      TRIGGER ROUTINE FOR TIME RELATED EVENTS DIRECTLY AFTER LIFTOFF
C*********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      CHARACTER*60 HEAD
      CHARACTER*6 MISION

      LOGICAL QFLG,GLIMIT,VISC

      COMMON/AGEN1/TIME(2,16),TAUT(15),TAUW(15),TZERO,TLIFT,TTILT,TMINH,
     *DTZ,TQ,TL,XMAUG,TNE(6,15),ENGDAT(8,15),S(15),WD(15),WJET(15),
     *HEAD,PRINT(15),STEP(15),HNOM(15),HMAX(15),PROP(6),TBEGR,TENDR,
     *CHRDOT,CHIRO,FAZ,BO(3,2),CBAXIL,TCHIR,VRCUT,FPRFAC,CHVEL

      COMMON/AGEN2/AA,SA,CA,ALF,ALFY,CHIP,SCHIP,CCHIP,CHIY,SCHIY,CCHIY,
     *CHIR,SCHIR,CCHIR,STH,CTH,STHL,CTHL,DPHIZ,RTHE,R,VR,XM,SW,SU,SV,Q,
     *VIV(25),DVARS(13),DELXDW(15,3),DELXDR(7,5),DELXD(15,7),WZERO,ALT,
     *XMACH,QDOT,FAA,FAN,A(3,3),CASE,CT2,UMF,A12W,A22W,A32W,XJEXT,BEU,
     *BYL,BEL,BHMAX(15),BHMN,BSTEP(15),HNMB,HMXB,TPOLY,XMIAD

      COMMON/AGEN3/ITHR,IWD,IXR,JUMP,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISION,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)

      COMMON/AGEN5/CHIBAS,CHPBAS,CHYBAS,CHISAV,TXX,TYY,TZZ,AMX,AMY,AMZ,
     *TMZ,THMFA,SIDE,WMAG,AZW,FANC,FLATC,FOM,VISC

      COMMON/BGEN3/NOSS,JTB,NENT,NACT,NOM1,NP(7)

      COMMON/BODY/ATBDWT(15,2),ATXCG(15,2),ATYCG(15,2),AXLEN(2),AXREF(2)
     *,AYREF(2),KPI(2),KYI(2),API(10,2),AYI(10,2)

      COMMON/BODYF/TBDWT(15),TXCG(15),TYCG(15),XLEN,XREF,YREF,M1,M2,M5,
     *M6,M7,M8,M9,M50,M51,MSRW,AP(10),AY(10)

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM

      COMMON/CONTRO/TTBL(30),CPTBL(30),CYTBL(30),TOBL(210),
     *CPOTBL(210),CYOTBL(210)

      COMMON/FORINT/HBANK(2),T,TT,W,U,V,X,Y,Z,XMI,ZAP(18),DVAR(25),FSAVE
     *(625),NTRG1,TV1,NTRG2,TV2,NTRG3,TV3,NTRG4,TV4,NTRG5,TV5,NTRG6,TV6,
     *NTRG7,TV7,NTRG8,TV8,NTRG9,TV9,NTRG10,TV10,NTRG11,TV11,NTRG12,TV12,
     *NTRG13,TV13,KIND,KRDER,EU,EL,AYL,HMX,HMN,HNM

      COMMON /HEADUP/ HDMACH,HDCHRD

      COMMON/ITHRO/ITHRO
```



```
      common/jthrot_save/jthrot_save(15)

      COMMON/KP/KP,KY

      COMMON/MPS/TIMMPS(15),FTBL(15),THROT(15),GLIM(15),TAUALT(5),
     *TAUTBL(5),MPS1,MPS2,TGLIM(10,15),GLIMIT,tau_const(10,5),jthrot(15)

      common/mult/nchiot(6),tbias(6),nstage,nstg(15),wf(5),ibranch

      COMMON/NTILT/NTILT

      COMMON/QMAX/QMAX,QFLG,QPEN,MAXQ

      COMMON/REST/JSTR,NCNRS(5),NTCN,TR(5),LSB,NF8,NOPAR,WIBT(15)

      COMMON/TAULIM/TAULIM,NTAU,NBETA,TIMTAU

      common/time_jump/time_jump(5),njump

      COMMON /TRIGC / NTRG14,TV14
C
      IF(XMACH.LT.HDMACH) NTILT=0
c*********************************************************************

      IF(IFACT.EQ.3) then

         CALL APRTN(7)

         KS1=2
         NTRG6=1
         NTRG4=-2

         RETURN

      endif
c*********************************************************************
C*********************************************************************
C     begin roll maneuver at time = tbegr and set next event time
C*********************************************************************

      IF(ABS(T-TBEGR).le.1.d-04.and.NROLL.ne.-1) then

         CALL APRTN(18)

         NROLL=1

c        if start time of tiltover maneuver (tlift) is greater than the
c        end of the roll maneuver set next trigger to the end of roll
c        maneuver time (tendr)
```

```
         IF(TLIFT.ge.TENDR) then

            TV4=TENDR

            RETURN

         endif

c        if the current time is the same time as the start time of the
c        tiltover maneuver (tlift), then print and set trigger for next
c        event.  Also turn on max dynamic pressure trigger (ntrg6).

         IF(ABS(T-TLIFT).le.1.d-04) then

            CALL APRTN(7)

            KS1=2
            NTRG6=1

            TV4=TTILT
            IF(TTILT.GE.TENDR) TV4=TENDR

            IF(T.GT.TLIFT) RETURN

            GO TO 2

         endif

c        if current time is greater than tlift, set next trigger to
c        closest event

         IF(T.GT.TLIFT) then

            TV4=TTILT
            IF(TTILT.GE.TENDR) TV4=TENDR

            IF(T.GT.TLIFT) RETURN

            GO TO 2

         endif

c        if all above test are not passed, the assumption is that the
c        next event will be the beginning of the tiltover event

         TV4=TLIFT

         RETURN

      endif

C*******************************************************************

C     END VERTICAL RISE , BEGIN TILT

C*******************************************************************
```



```
      IF(ABS(T-TLIFT).le.1.d-04) then

         CALL APRTN(7)

         KS1=2
         NTRG6=1

c        if nroll is equal to -1, the roll maneuver will not be performed
c        therefore, the next event will be the end of the tiltover maneuver

         IF(NROLL.EQ.-1) then

            tv4=ttilt

            go to 2

         endif

c        if the roll maneuver is to be performed, the next event is assummed
c        to be the time to begin roll (tbegr)

         IF(NROLL.EQ.0) then

            tv4=tbegr

            go to 2

         endif

c        if the roll maneuver is in process (nroll=1), the next event
c        is defined based on the closest time

         TV4=TTILT
         IF(TTILT.GE.TENDR) TV4=TENDR

         IF(T.GT.TLIFT) RETURN

         GO TO 2

      endif

C*******************************************************************

C     end roll maneuver at time = tendr

C*******************************************************************

      IF(ABS(T-TENDR).le.1.d-04) then

         IF(TENDR.GE.TTILT) NTILT=1

         CALL APRTN(19)

         NROLL=0
```

```
c       if the end of roll time (tendr) is less than the beginning of
c       the tiltover maneuver (tlift), tlift is selected as the next
c       event

         IF(TENDR.lt.TLIFT) then

            TV4=TLIFT

            RETURN

         endif

c       if the end of roll maneuver time (tendr) is not equal to the
c       end of tiltover time (ttilt), ttilt is selected as the next event

         IF(ABS(T-TTILT).ge.1.d-04) then

            IF(TENDR.lt.TTILT) then

               TV4=TTILT

               RETURN

            endif

c       if tendr is the last event, the trigger flag(ntrg4) is turned
c       off

            NTRG4=-2

C*****************************************************************

C     AT END OF 2ND ROLL (HEADS-UP TO HEADS-DOWN)

C*****************************************************************

            IF(XMACH.GE.HDMACH) THEN

               CHRDOT=CHRDSV

               CALL ADER1

               CALL ADER

               CALL APRTN(1)

            ENDIF

            RETURN

         endif

      else

C*****************************************************************
```



```
C       end tiltover maneuver and turn off trigger

C*****************************************************************

         CALL APRTN(8)

         TV4=TENDR

         IF(TTILT.GE.TENDR) NTRG4=-2

         NTILT=1

         CALL ADER1

         CALL ADER

         CALL APRTN(1)

         RETURN

      endif

C*****************************************************************

c       Define attitude angles for continuation of the first stage

C*****************************************************************

    2 CPTBL(1)=CHPBAS*RAD
      CYTBL(1)=CHYBAS*RAD

      KP=KPI(1)                                                    ATIL  81
      KY=KYI(1)                                                    ATIL  82

      CALL FIND(CPTBL,AP,TTBL,KP)                                  ATIL  83
      CALL FIND(CYTBL,AY,TTBL(16),KY)

      DO 20 I=1,10
      API(I,1)=AP(I)
   20 AYI(I,1)=AY(I)

      RETURN                                                       ATIL  85

C*****************************************************************

      ENTRY AQMAX                                                  ATIL 103

C*****************************************************************

C       TRIGGER ROUTINE FOR MAXIMUM DYNAMIC PRESSURE PRINTOUT

C
      QFLG=.TRUE.
      NTRG6=-1

      CALL ADER1
```

```
      CALL ADER
      CALL APRTN(14)
      qpen=q*psffm
      IF(NTAU.eq.0) return
      TV13=T+TIMTAU
      TIMMPS(NTAU+2)=TV13
      if(taulim.gt.0.) then
         TIMMPS(NTAU+3)=TV13+(FTBL(NTAU+3)-FTBL(NTAU+2))/TAULIM
      else
         timmps(ntau+3)=tv13
      endif
      NTRG13=2
      RETURN                                                          ATIL 108
C*********************************************************************
      ENTRY AXPRT                                                     ATIL 109
C*********************************************************************
C      TRIGGER ROUTINE FOR NORMAL PRINTOUT
C
      IF(NMAX.EQ.0) CALL APRTN(1)                                     ATIL 110
      TV1=AINT(T+.1E-03)+PRINT(ITHR)+tbias(nstage)-int(tbias(nstage))
      RETURN                                                          ATIL 112
C*********************************************************************
      ENTRY GLIMT                                                     ATIL 113
C*********************************************************************
C      TRIGGER ROUTINE FOR ACCELERATION LIMIT PRINTOUT
C
      NTRG9=-2                                                        ATIL 114
      CALL APRTN(22)                                                  ATIL 115
      HNM=STEP(ITHR)                                                  ATIL 117
      GLIMIT=.TRUE.
```



```
      jthrot(ithr)=jthrot(ithr)+1
      if(lsb.eq.0) jthrot_save(ithr)=jthrot(ithr)
      tglim(jthrot(ithr),ithr)=t
      if(mpsflg(ithr).eq.4) return
      thrt=throt(ithr)*(tne(1,ithr)+tne(6,ithr))*
     *     engdat(6,1)*ptn
      thrt=tau_const(jthrot(ithr),nstage)*thrt
      wdott=thrt/(gzero*funisp(tau_const(jthrot(ithr),nstage),
     *      nstage))
      wglim=thrt/(gzero*glim(ithr))
      tg=0.
      if(wglim.lt.xmiad+xmaug) then
         tg=(xmiad+xmaug-wglim)/wdott
         ntrg9=2
      endif
      tv9=t+tg
      if(tg.eq.0..or.tv9.ge.tv2) ntrg9=-2
      call ader
      call ader1
      call aprtn(1)
      t1=0.
      call aeosr
      RETURN                                                          ATIL 118
C*********************************************************************
      ENTRY AWDEV
C*********************************************************************
C      TRIGGER ROUTINE FOR WEIGHT DROP EVENTS
C
      IWD=IWD+1
      IF(LSB.NE.1.AND.WD(IWD).GT.0.) CALL APRTN(1)
      XMAUG=XMAUG-WD(IWD)
```

```
      IF(LSB.ne.1) then

         DO 12 I=1,3
   12    DVARS(I)=DVAR(I)

         CALL ADER1

         CALL ADER

         DO 13 I=1,3
   13    DELXDW(IWD,I)=DVARS(I)-DVAR(I)

         IF(WD(IWD).GT.0.) CALL APRTN(17)

      endif

      IF(IWD.EQ.NWVNT) then

         ntrg12=-2

      else

         TV12=TIME(2,IWD+1)

      endif

      RETURN

C*********************************************************************

      ENTRY ALIFT

C*********************************************************************

C     TRIGGER ROUTINE TO FLAG LIFTOFF BASED ON F/M=1

C
      NTRG7=-2

      CALL ADER1

      CALL ADER

      CALL APRTN(2)

      RETURN

C*********************************************************************

      ENTRY ATHRO

C*********************************************************************

C     TRIGGER ROUTINE FOR SSME THROTTLE TABLE
```



```
C
      CALL APRTN(25)

      IF(TIMMPS(ITHRO+1).LT.TIMMPS(ITHRO).and.
     *ithro.gt.ntau+2) then

         NTRG13=-2
         qflg=.false.

         return

      endif

      ITHRO=ITHRO+1

      if((timmps(ithro)-timmps(ithro-1)).lt..01) ithro=ithro+1

      TV13=TIMMPS(ITHRO)
      IF(ITHRO.ge.NTAU+2) NTRG13=-2

      qflg=.true.
      if(ithro.ge.ntau+3) qflg=.false.

      if(taulim.eq.0.) then

         call ader

         call ader1

         call aprtn(1)

      endif

      RETURN

C*********************************************************************

      ENTRY AROLL

C*********************************************************************

C     TRIGGER ENTRY POINT FOR HEADS-UP TO HEADS-DOWN ROLL MANEUVER

C
      CALL APRTN(18)

      NROLL=1
      NTRG14=-1
      NTRG4=2

      CHIRF=PI
      CHIRO=CHIR
      CHRDSV=CHRDOT
      CHRDOT=HDCHRD

      TBEGR=T
```

```
      IF(CHIRO.GE.0.0) TENDR=(CHIRF-CHIRO)/CHRDOT
      IF(CHIRO.LT.0.0) TENDR=(CHIRF+CHIRO)/CHRDOT
      TENDR=TENDR+TBEGR

      TV4=TENDR
C
      RETURN
c*************************************************************************

      entry ajump

c*************************************************************************

c       trigger routine for first stage jump start option


      call ader1

      call ader

      call aprtn(23)

      if(nmax.eq.0) write(18,*) t,w,u,v,x,y,z,xmi,(zap(i),i=1,18),
     *              chip,xm,wzero,tzero,chpbas,chybas

      njump=njump+1

      if(njump.gt.6.or.time_jump(njump).lt..1) then

         ntrg3=-2
         return

      endif

      tv3=time_jump(njump)

      return

      END                                                    ATIL 129
```

3.19 Subroutine ATMOSPHERE

3.19.1 Purpose

This subroutine contains the reference, hot, and cold models for the 1963 Patrick and 1971 Vandenburg atmosphere models.

3.19.2 Variable Listing

3.19.3 Subroutines Called:

None

3.19.4 Calling Subroutines:

Name	Function
ADER	Calculates the time independent portion of the equations of motion during forward integration
AINIT	Input routine
BDR1I	Calculates components of the equations of motion for the backward trajectory

3.19.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a	Coefficients for spline interpolation				Local	
atbl	Altitude table for atmosphere parameters	m			Local	
atmos	Name of atmospheric subroutine being used for the simulation				Local	
atmosn	Name of atmospheric subroutine being used for the simulation				Local	
b	Coefficients for spline interpolation				Local	
c	Coefficients for spline interpolation				Local	
ctbl	Speed of sound tables	m/sec			Local	
dlx	Temporary variable				Local	
dlxi	Temporary variable (1/dlx)				Local	
dlxmi	Temporary variable				Local	
dlxpi	Temporary variable				Local	
dx	Temporary variable				Local	
ifirst	First time index				Local	
key	Operation index				Local	
last	Previous index used in atmospheric routine			output	Global	mlast
m	Index of current place in interpolated table			output input	Global	mlast
mm1	m-1				Local	
mp1	m+1				Local	
mp2	m+2				Local	
n	Number of data points in tables				Local	
nn	Previous index used in atmospheric routine			output	Global	mlast
pr	Output array of atmospheric parameters	variabl			Local	
ptbl	Atmospheric pressure tables				Local	
r	Temporary variable				Local	
r2	Radius squared	m^2			Local	
rhotbl	Atmospheric density tables				Local	
s	Temporary array				Local	
sm1	Temporary storage array				Local	
sp1	Temporary storage array				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
t	Temperature				Local	
tn	Temperature				Local	
x	Independent table for spline interpolation				Local	
x t	Independent table for spline interpolation				Local	
y	Dependent tables for spline interpolation				Local	
y t	Dependent tables for spline interpolation				Local	

```
      Subroutine Atmosphere(pr,atmosn,error,key)
c*********************************************************************
c       This subroutine combines six atmospheric routines into one
c       subroutine, the desired atmospheric routine will be denoted by
c       the input parameter "key" in the calling argument. Values of
c       key are the following :
c       key     routine
c       1       Patrick Air Force Base Reference (ETR) PRA63
c       2       Vandenburg Air Force Base Reference (WTR) VRA71
c       3       ETR Hot PHA63
c       4       ETR Cold PCA63
c       5       WTR Hot VHA71
c       6       WTR Cold VCA71
c*********************************************************************
c       Output from the atmosphere routine are contained in the parameter
c       "pr" and are defined as the following :
c       pr      Definition
c       1       Altitude (Input)
c       2       Atmospheric Pressure
c       3       Partial derivative of pressure with respect to altitude
c       4       Partial derivative of density with respect to altitude
c       5       Partial derivative of speed of sound with respect to
c               altitude
c       6       Atmospheric density
c       7       Not used
c       8       Not used
c       9       Speed of sound
c       10      Coefficient of Viscosity
c       11      Partial derivative of coefficient of viscosity with
c               respect to altitude
c       12      Temperature
c       13      Partial derivative of temperature with respect to
c               altitude
c*********************************************************************
      implicit double precision (a-h,o-z)

      character*6 atmos(6),atmosn

      common/mlast/last,m,k

      dimension s(9),sml(9),spl(9),a(10),b(10),c(10),
     *  atbl(89,6),ptbl(89,6),rhotbl(89,6),ctbl(89,6),xt(89),
     *  yt(89,3),y(3),pr(*),nn(6)

      data ifirst/0/
      data nn/89,75,60,60,60,60/
```



```
      data atmos/' pra63',' vra71',' pha63',' pca63',' vha71',
     *  ' vca71'/
c*********************************************************************
c       Eastern Test Range (reference)
c*********************************************************************
      data (atbl(i,1),i=1,89)/
     *  0.,500.,1000.,1500.,2000.,2500.,3000.,4000.,5000.,6000.,7000.,
     *  8000.,9000.,10000.,11000.,12000.,13000.,14000.,15000.,16000.,
     *  16500.,17000.,18000.,19000.,20000.,22000.,24000.,26000.,28000.,
     *  30000.,32000.,34000.,36000.,38000.,40000.,44000.,48000.,52000.,
     *  57000.,62000.,67000.,72000.,78000.,80000.,83000.,84000.,86000.,
     *  89000.,90000.,93000.,96000.,98000.,100000.,103000.,105000.,
     *  110000.,114000.,120000.,124000.,130000.,134000.,140000.,
     *  144000.,150000.,160000.,175000.,190000.,210000.,220000.,
     *  240000.,260000.,280000.,300000.,326000.,350000.,376000.,
     *  400000.,426000.,450000.,476000.,500000.,526000.,550000.,
     *  576000.,600000.,626000.,650000.,676000.,700000./

      data (ptbl(i,1),i=1,44)/
     *  .10170147d02, .96022651d01, .90603417d01, .85438572d01,
     *  .80521166d01, .75843000d01, .71395062d01, .63151744d01,
     *  .55714346d01, .49008910d01, .42967958d01, .37532038d01,
     *  .32649867d01, .28277554d01, .24373143d01, .20909280d01,
     *  .17861066d01, .15199026d01, .12892855d01, .10911841d01,
     *  .10033656d01, .92252635d00, .78097360d00, .66260085d00,
     *  .56315646d00, .40899187d00, .29918756d00, .22038158d00,
     *  .16327365d00, .12146274d00, .9090508d-01, .6842991d-01,
     *  .5180718d-01, .3944799d-01, .3020918d-01, .1800451d-01,
     *  .1091056d-01, .6639319d-02, .3497353d-02, .1781846d-02,
     *  .8709643d-03, .4057600d-03, .1511983d-03, .1068430d-03/

      data (ptbl(i,1),i=45,89)/
     *  .6235580d-04, .5187821d-04, .3591471d-04, .2069615d-04,
     *  .1722443d-04, .1006525d-04, .6033042d-05, .4344971d-05,
     *  .3159717d-05, .2005384d-05, .1506007d-05, .7743898d-06,
     *  .4838607d-06, .2659771d-06, .1914191d-06, .1289841d-06,
     *  .1036962d-06, .7834591d-07, .6656633d-07, .5358494d-07,
     *  .3912849d-07, .2594138d-07, .1787156d-07, .1130478d-07,
     *  .9106353d-08, .6040650d-08, .4100777d-08, .2841337d-08,
     *  .2005731d-08, .1304728d-08, .8946879d-09, .6062792d-09,
     *  .4304566d-09, .3014289d-09, .2195289d-09, .1576117d-09,
     *  .1173154d-09, .8597494d-10, .6498292d-10, .4833493d-10,
     *  .3701989d-10, .2789144d-10, .2157540d-10, .1641584d-10,
     *  .1281153d-10/

      data (rhotbl(i,1),i=1,44)/
     *  .11835467d01, .11312044d01, .10793462d01, .10284922d01,
     *  .97902799d00, .93122445d00, .88525680d00, .79915661d00,
     *  .72084273d00, .64983432d00, .58535150d00, .52651816d00,
     *  .47249380d00, .42255460d00, .37638426d00, .33302118d00,
     *  .29232217d00, .25432636d00, .21920325d00, .18717684d00,
     *  .17239239d00, .15845601d00, .13239217d00, .11096236d00,
     *  .9319379d-01, .6619324d-01, .4747889d-01, .3438248d-01,
     *  .25119032d-1, .1833406d-01, .1345779d-01, .9930103d-02,
```

```
     *  .7365417d-02, .5493419d-02, .4122020d-02, .2368456d-02,
     *  .1401576d-02, .8652672d-03, .4766351d-03, .2574523d-03,
     *  .1343447d-03, .6694933d-04, .2706738d-04, .1967746d-04/

        data (rhotbl(i,1),i=45,89)/
     *  .1202377d-04, .1000425d-04, .6925835d-05, .3991071d-05,
     *  .3321580d-05, .1848883d-05, .1057999d-05, .7396272d-06,
     *  .5225459d-06, .3095993d-06, .2226370d-06, .1034998d-06,
     *  .5606565d-07, .2569189d-07, .1513314d-07, .8014608d-08,
     *  .5638713d-08, .3588138d-08, .2758526d-08, .1943190d-08,
     *  .1227308d-08, .7254957d-09, .4609539d-09, .2714797d-09,
     *  .2113989d-09, .1322960d-09, .8551031d-10, .5654070d-10,
     *  .3816850d-10, .2371704d-10, .1561798d-10, .1014715d-10,
     *  .6940371d-11, .4712580d-11, .3338648d-11, .2328286d-11,
     *  .1688344d-11, .1215120d-11, .9034760d-12, .6603653d-12,
     *  .4978106d-12, .3709640d-12, .2840954d-12, .2138449d-12,
     *  .1652610d-12/

        data (ctbl(i,1),i=1,44)/
     *  .34685752d03, .34474101d03, .34281972d03, .34103528d03,
     *  .33933665d03, .33768002d03, .33602847d03, .33262586d03,
     *  .32895941d03, .32494680d03, .32057962d03, .31591022d03,
     *  .31103836d03, .30609732d03, .30115374d03, .29654541d03,
     *  .29250004d03, .28922838d03, .28690521d03, .28565249d03,
     *  .28544720d03, .28552324d03, .28723862d03, .28897076d03,
     *  .29075108d03, .29419972d03, .29721502d03, .29966127d03,
     *  .30166195d03, .30454827d03, .30751834d03, .31060616d03,
     *  .31380529d03, .31706989d03, .32031579d03, .32622855d03,
     *  .33012557d03, .32775595d03, .32050933d03, .31127962d03,
     *  .30126858d03, .29128989d03, .27964963d03, .27571001d03/

        data (ctbl(i,1),i=45,89)/
     *  .26945234d03, .26944122d03, .26944122d03, .26944122d03,
     *  .26944122d03, .27607143d03, .28254611d03, .28678136d03,
     *  .29095497d03, .30113603d03, .30773633d03, .32364874d03,
     *  .34759671d03, .38070451d03, .42081571d03, .47466911d03,
     *  .50740585d03, .55288841d03, .58123621d03, .62133762d03,
     *  .66808799d03, .70752708d03, .73674375d03, .76353047d03,
     *  .77657742d03, .79952559d03, .81938459d03, .83877353d03,
     *  .85772430d03, .87759431d03, .89554465d03, .91459335d03,
     *  .93183123d03, .94629601d03, .95945458d03, .97350900d03,
     *  .98630458d03, .99526859d03, .10034720d04, .10122840d04,
     *  .10203506d04, .10259673d04, .10311248d04, .10366832d04,
     *  .10417876d04/

c*************************************************************************
c       Western Test Range (Reference)
c*************************************************************************

        data (atbl(i,2),i=1,75)/
     *  0.,500.,1000.,1500.,2000.,2500.,3000.,4000.,5000.,6000.,7000.,
     *  8000.,9000.,10000.,11000.,12000.,13000.,14000.,15000.,16000.,
     *  16500.,17000.,18000.,19000.,20000.,22000.,24000.,26000.,28000.,
     *  30000.,32000.,34000.,36000.,38000.,40000.,44000.,48000.,52000.,
     *  57000.,62000.,67000.,72000.,78000.,80000.,83000.,84000.,86000.,
     *  89000.,90000.,93000.,96000.,98000.,100000.,103000.,105000.,
```



```
     *  110000.,114000.,120000.,124000.,130000.,134000.,140000.,
     *  144000.,150000.,160000.,175000.,190000.,210000.,220000.,
     *  240000.,260000.,280000.,300000.,326000.,350000./

        data (ptbl(i,2),i=1,44)/
     *     10.189904,      9.6029909,      9.0477941,      8.5222274,
     *     8.0243502,      7.5523727,      7.1046558,      6.2761809,
     *     5.5286658,      4.8538800,      4.2453086,      3.6977986,
     *     3.2071987,      2.7706815,      2.3796741,      2.0378622,
     *     1.7407523,      1.4839231,      1.2630643,      1.0740324,
     *     .99021667,      .91290847,      .77604645,      .66155133,
     *     .56398303,      .41046343,      .30077469,      .22205890,
     *     .16405777,      .12206691, .91233502d-01, .68532667d-01,
     *  .51770654d-01, .39343655d-01, .30083150d-01, .17895922d-01,
     *  .10838504d-01, .66065708d-02, .35159756d-02, .18156504d-02,
     *  .89950478d-03, .42355436d-03, .15895186d-03, .11243729d-03/

        data (ptbl(i,2),i=45,75)/
     *  .65436167d-04, .54429003d-04, .37664308d-04, .21690308d-04,
     *  .18049200d-04, .10540546d-04, .63141230d-05, .45456466d-05,
     *  .33040799d-05, .20958842d-05, .15734398d-05, .80843999d-06,
     *  .50485583d-06, .27732500d-06, .19950839d-06, .13437216d-06,
     *  .10800000d-06, .81570390d-07, .69292605d-07, .55766306d-07,
     *  .40705497d-07, .26973787d-07, .18574202d-07, .11742803d-07,
     *  .94567372d-08, .62700608d-08, .42545264d-08, .29465578d-08,
     *  .20790502d-08, .13517150d-08, .92648059d-09/

        data (rhotbl(i,2),i=1,44)/
     *     1.2361775,      1.1632456,      1.0983266,      1.0396111,
     *     .98575564,      .93576941,      .88892961,      .80276189,
     *     .72468088,      .65342553,      .58827869,      .52859970,
     *     .47357411,      .42242641,      .37353754,      .32593435,
     *     .28201088,      .24284485,      .20855008,      .17862758,
     *     .16505515,      .15225517,      .12851184,      .10882416,
     *  .92019110d-01, .65810409d-01, .47498948d-01, .34657364d-01,
     *  .25289117d-01, .18453921d-01, .13544035d-01, .99959356d-02,
     *  .74112084d-02, .55182770d-02, .41277733d-02, .23525465d-02,
     *  .13934194d-02, .84993924d-03, .46833031d-03, .25705139d-03,
     *  .13650376d-03, .68824123d-04, .28236629d-04, .20768398d-04/

        data (rhotbl(i,2),i=45,75)/
     *  .12618788d-04, .10496153d-04, .72632298d-05, .41827847d-05,
     *  .34806292d-05, .19361901d-05, .11072921d-05, .77378743d-06,
     *  .54642031d-06, .32357107d-06, .23260574d-06, .10805076d-06,
     *  .58498391d-07, .26788033d-07, .15772667d-07, .83493992d-08,
     *  .58727388d-08, .37358155d-08, .28715040d-08, .20222947d-08,
     *  .12767727d-08, .75436869d-09, .47907675d-09, .28199857d-09,
     *  .21953292d-09, .13732034d-09, .88716316d-10, .58634523d-10,
     *  .39563746d-10, .24571145d-10, .16172967d-10/

        data (ctbl(i,2),i=1,75)/
     *  339.70470,339.97072,339.61291,338.78074,337.59326,336.14343,
     *  334.50165,330.82996,326.80685,322.49129,317.86835,312.95134,
     *  307.89330,303.03224,298.65008,295.84359,293.95944,292.50124,
     *  291.20953,290.13377,289.79387,289.70126,290.77721,291.65773,
     *  292.84225,295.53832,297.80132,299.43738,301.43187,304.31165,
```

```
     *  307.09098,309.81399,312.72402,315.93631,319.42409,326.34116,
     *  329.99523,329.88156,324.19829,314.46338,303.73397,293.52707,
     *  280.73096,275.30724,269.44122,269.44122,269.44122,269.44122,
     *  269.44122,276.07143,282.54611,286.78136,290.95497,301.13603,
     *  307.73633,323.64874,347.59671,380.70451,420.81571,474.66911,
     *  507.40585,552.88841,581.23621,621.33762,668.08799,707.52708,
     *  736.74375,763.53047,776.57742,799.52559,819.38459,838.77353,
     *  857.72430,877.59431,895.54465/

c*************************************************************************
c       Eastern Test Range (Hot)
c*************************************************************************

        data (atbl(i,3),i=1,60)/
     *  0.,500.,1000.,1500.,2000.,2500.,3000.,4000.,5000.,6000.,7000.,
     *  8000.,9000.,10000.,11000.,12000.,13000.,14000.,15000.,16000.,
     *  16500.,17000.,18000.,19000.,20000.,22000.,24000.,26000.,28000.,
     *  30000.,32000.,34000.,36000.,38000.,40000.,44000.,48000.,52000.,
     *  57000.,62000.,67000.,72000.,78000.,80000.,83000.,84000.,86000.,
     *  89000.,90000.,93000.,96000.,98000.,100000.,103000.,105000.,
     *  110000.,114000.,120000.,124000.,130000./

        data (ptbl(i,3),i=1,60)/
     *  10.10,9.5554384,9.0346817,8.5369369,
     *  8.0614294,7.6074017,7.1741153,6.3668963,5.6342022,4.9707288,
     *  4.3714321,3.8315199,3.3464479,2.9119117,2.5238402,2.1780131,
     *  1.8705877,1.5983606,1.3583010,1.1475476,1.0530908,.96617676,
     *  .81369526,.68712577,.58222915,.42201626,.30875086,.22793980,
     *  .16972565,.12732124,.096198730,.073279037,.056145477,.043294535,
     *  .33570537d-01,.20507717d-01,.12777672d-01,.8039993d-02,
     *  .43505454d-02,.22383559d-02,.10846496d-02,.48844814d-03,
     *  .16777992d-03,.11290073d-03,.61855316d-04,.50659180d-04,
     *  .33922195d-04,.18634796d-04,.15134811d-04,.88441608d-05,
     *  .53011287d-05,.38178501d-05,.27763880d-05,.17620961d-05,
     *  .13233024d-05,.68044276d-06,.42516d-06,.23370942d-06,
     *  .16819664d-06,.1133613d-06/

        data (rhotbl(i,3),i=1,60)/
     *  1.13537,1.0860059,1.0382755,.99214261,.94757068,.90452383,
     *  .86296655,.78418111,.71093916,.64297218,.58001870,.52182367,
     *  .46813915,.41872392,.37334363,.33249312,.29499874,.26068215,
     *  .22936894,.20088866,.18489258,.16985547,.14173245,.11724374,
     *  .097358475,.068172810,.048447590,.034775536,
     *  .25199219d-01,.18405121d-01,.13546524d-01,.10065735d-01,
     *  .75227375d-02,.56749344d-02,.43068771d-02,.25237846d-02,
     *  .15109100d-02,.96910328d-03,.56608540d-03,.31631672d-03,
     *  .16770816d-03,.83469629d-04,.32638311d-04,
     *  .23151398d-04,.12674332d-04,.10385513d-04,.69713593d-05,
     *  .38394928d-05,.31070709d-05,.17294816d-05,.98967373d-06,
     *  .69186170d-06,.48879962d-06,.2896052d-06,.20825900d-06,
     *  .9681575d-07,.6064035d-07,.24032692d-07,.1415584d-07,
     *  .749702d-08/

        data (ctbl(i,3),i=1,60)/
     *  352.9034,350.97257,349.03106,347.07869,345.11527,343.14062,
     *  341.15453,337.14728,333.09181,328.98636,324.82903,320.61779,
```



```
     *  316.35049,312.02484,307.63837,302.83277,297.94967,292.98520,
     *  287.93514,282.79491,282.40316,282.19925,283.50456,286.44251,
     *  289.35063,294.39916,298.69059,302.92123,307.09359,311.21002,
     *  315.27271,319.28370,323.24493,326.81155,330.33967,337.28520,
     *  344.09056,340.79560,328.02715,314.74114,300.86901,286.32557,
     *  267.83305,261.37829,261.37829,261.37829,261.37829,261.37829,
     *  261.37829,261.37829,261.37829,261.37829,261.37829,261.37829,
     *  261.37829,261.37829,261.37829,261.37829,261.37829,261.37829/

c*************************************************************************
c       Eastern Test Range (Cold)
c*************************************************************************

        data (atbl(i,4),i=1,60)/
     *  0.,500.,1000.,1500.,2000.,2500.,3000.,4000.,5000.,6000.,7000.,
     *  8000.,9000.,10000.,11000.,12000.,13000.,14000.,15000.,16000.,
     *  16500.,17000.,18000.,19000.,20000.,22000.,24000.,26000.,28000.,
     *  30000.,32000.,34000.,36000.,38000.,40000.,44000.,48000.,52000.,
     *  57000.,62000.,67000.,72000.,78000.,80000.,83000.,84000.,86000.,
     *  89000.,90000.,93000.,96000.,98000.,100000.,103000.,105000.,
     *  110000.,114000.,120000.,124000.,130000./

        data (ptbl(i,4),i=1,60)/
     *  10.27,9.6487231,9.0598171,8.5018719,17.9735277,7.4734710,
     *  7.0004350,6.1305847,5.3547301,4.6646548,4.0522889,3.5107186,
     *  3.0333981,2.6141402,2.2471262,1.9269158,1.6484578,1.4071002,
     *  1.1985998,1.0191326,.93920913,.86531102,.73453635,.62346069,
     *  .52929926,.38218424,.27700539,.20168165,.14748703,.10832073,
     *  .07995773,.05931480,.04411646,.03303959,.02490124,.01439423,
     *  8.5065841d-03,5.0981116d-03,2.6690495d-03,1.3722539d-3,
     *  6.9297552d-4,3.4146309d-4,1.4385223d-4,1.0599136d-4,
     *  6.6890716d-5,5.7106018d-5,4.1339397d-5,2.4955273d-5,
     *  .20947456d-04,.12240884d-04,.73371013d-05,.52841488d-05,
     *  .38426987d-05,.24388538d-05,.18315352d-05,.94177632d-06,
     *  .58844858d-06,.32346879d-06,.23279491d-06,.15686446d-06/

        data (rhotbl(i,4),i=1,60)/
     *  1.300948,1.2335057,1.1689434,1.1072064,1.0481959,.99181543,
     *  .93797119,.83752835,.74616589,.66278387,.58710627,.51842328,
     *  .45620410,.40002236,.34951847,.30436223,.26421523,.22869338,
     *  .19732939,.16953508,.15675060,.14460154,.12283195,.10417007,
     *  .08822921,.06314260,.04516896,.03239635,.02334539,.01691070,
     *  .01230191,8.9853973d-3,6.5724525d-3,4.8153175d-3,3.5523605d-3,
     *  1.9698486d-3,1.1187134d-3,6.6395879d-4,3.5713458d-4,
     *  1.8879223d-4,9.8073482d-5,4.98156555d-5,2.1701813d-5,
     *  1.6231239d-5,1.0700285d-5,9.2763901d-6,6.9222450d-6,
     *  4.3857098d-6,.37453175d-05,.20847472d-05,.11929696d-05,
     *  .83398208d-06,.58920755d-06,.34909519d-06,.25103905d-06,
     *  .11670339d-06,.63217998d-07,.28969425d-07,.17063697d-07,
     *  .903709d-08/

        data (ctbl(i,4),i=1,60)/
     *  332.43856,330.92403,329.40253,327.87397,326.33826,324.79528,
     *  323.24493,320.12172,316.96774,313.93309,310.85280,307.87839,
     *  305.08411,302.48192,300.04569,297.73722,295.53345,293.45514,
     *  291.59633,290.15454,289.68724,289.46291,289.31583,289.44289,
```

```
     *  289.81148,291.13421,293.00379,295.16011,297.39355,299.57159,
     *  301.66304,303.76093,306.55174,309.92758,313.26704,319.84138,
     *  326.28328,327.87397,323.46687,318.99888,314.46741,309.86969,
     *  304.26072,302.36795,295.81768,293.60179,289.11906,282.26151,
     *  279.93832,286.82681,293.55374,297.59399,302.29020,312.86789,
     *  319.72534,336.25767,361.13865,395.53626,437.21013,493.19276/

c*************************************************************************
c       Western Test Range (Hot)
c*************************************************************************

        data (atbl(i,5),i=1,60)/
     *  0.,500.,1000.,1500.,2000.,2500.,3000.,4000.,5000.,6000.,7000.,
     *  8000.,9000.,10000.,11000.,12000.,13000.,14000.,15000.,16000.,
     *  16500.,17000.,18000.,19000.,20000.,22000.,24000.,26000.,28000.,
     *  30000.,32000.,34000.,36000.,38000.,40000.,44000.,48000.,52000.,
     *  57000.,62000.,67000.,72000.,78000.,80000.,83000.,84000.,86000.,
     *  89000.,90000.,93000.,96000.,98000.,100000.,103000.,105000.,
     *  110000.,114000.,120000.,124000.,130000./

        data (ptbl(i,5),i=1,60)/
     *  10.1,9.5601089,9.0433739,8.5490524,8.0764172,
     *  7.6247555,7.1933710,6.3887240,5.6672056,4.9937765,4.3936324,
     *  3.8521925,3.3650957,2.9268444,2.5326658,2.1795295,1.8645095,
     *  1.5847846,1.3376399,1.1241236,1.0303371,.94468907,.79564330,
     *  .67212650,.56943204,.41212652,.30141799,.22262406,.16595099,
     *  .12478311,.094554038,.72140493d-01,.55399955d-01,
     *  .42809272d-01,.33276706d-01,.20446185d-01,.12818158d-01,
     *  .80809533d-02,.44444739d-02,.23327743d-02,.11592365d-02,
     *  .53989683d-03,.19438558d-03,.13417355d-03,.75973197d-04,
     *  .62852839d-04,.43018365d-04,.24358324d-04,.20151710d-04,
     *  .11762549d-04,.70461420d-05,.50726398d-05,.36871338d-05,
     *  .23401629d-05,.17558543d-05,.90216536d-06,.56385584d-06,
     *  .30947628d-06,.22263812d-06,.14995041d-06/

        data (rhotbl(i,5),i=1,60)/
     *  1.1152041,1.0772095,1.0307463,.98573248,.94228637,.90022641,
     *  .85957157,.78235450,.71039274,.64344846,.58128929,.52368784,
     *  .47042152,.42389678,.38051989,.34017810,.30276636,.26817730,
     *  .23630347,.20010642,.18271118,.16688595,.13807836,.11462241,
     *  .95455732d-01,.66810926d-01,.47305779d-01,.33859990d-01,
     *  .24483823d-01,.17874341d-01,.13205656d-01,.98296066d-02,
     *  .73688882d-02,.55617586d-02,.42250443d-02,.24831313d-02,
     *  .15075731d-02,.95041952d-03,.56202945d-03,.31897732d-03,
     *  .17253926d-03,.88159774d-04,.35927203d-04,.25938802d-04,
     *  .14687349d-04,.12150895d-04,.83164297d-05,.47090188d-05,
     *  .38957845d-05,.21671307d-05,.12393653d-05,.86608154d-06,
     *  .61159536d-06,.36216527d-06,.26034998d-06,.12093860d-06,
     *  .65475833d-07,.30095125d-07,.17653964d-07,.934d-08/

        data (ctbl(i,5),i=1,60)/
     *  354.49409,352.48877,350.47198,348.44352,346.40318,344.35076,
     *  342.28602,338.11873,333.89943,329.62613,325.29670,320.90885,
     *  316.46018,310.90857,305.25600,299.49677,293.62461,287.63258,
     *  281.51303,280.44033,280.97719,281.51303,284.02747,286.51984,
     *  288.99072,293.87016,298.66989,303.39370,308.04507,312.62725,
```



```
     *  316.60954,320.54236,324.42751,328.26668,332.06147,339.52382,
     *  345.01468,345.01468,332.73202,319.97822,306.69451,292.80879,
     *  275.22271,269.10542,269.10542,269.10542,269.10542,296.10542,
     *  296.10542,296.10542,296.10542,296.10542,269.10542,269.10542,
     *  269.10542,269.10542,269.10542,269.10542,269.10542,269.10542/

c*************************************************************************
c       Western Test Range (Cold)
c*************************************************************************

        data (atbl(i,6),i=1,60)/
     *  0.,500.,1000.,1500.,2000.,2500.,3000.,4000.,5000.,6000.,7000.,
     *  8000.,9000.,10000.,11000.,12000.,13000.,14000.,15000.,16000.,
     *  16500.,17000.,18000.,19000.,20000.,22000.,24000.,26000.,28000.,
     *  30000.,32000.,34000.,36000.,38000.,40000.,44000.,48000.,52000.,
     *  57000.,62000.,67000.,72000.,78000.,80000.,83000.,84000.,86000.,
     *  89000.,90000.,93000.,96000.,98000.,100000.,103000.,105000.,
     *  110000.,114000.,120000.,124000.,130000./

        data (ptbl(i,6),i=1,60)/
     *  10.18,9.5587276,8.9693280,8.4104842,7.8809163,7.3793908,
     *  6.9047074,6.0312686,5.2517769,4.5580398,3.9423968,3.3976521,
     *  2.9171198,2.4993634,2.1409335,1.8334760,1.5698037,1.3437334,
     *  1.1499482,.98387621,.90998229,.84158803,.71970595,.61551163,
     *  .52659006,.38584043,.28310973,.20802155,.15306042,.11277507,
     *  .83205808d-01,.61610998d-01,.45881753d-01,.34356322d-01,
     *  .25862528d-01,.14878758d-01,.87191499d-02,.51359787d-02,
     *  .26265177d-02,.13182113d-02,.64858662d-03,.31247647d-03,
     *  .12629642d-03,.92675794d-04,.57823921d-04,.49335935d-04,
     *  .35914930d-04,.22307069d-04,.19032610d-04,.11114846d-04,
     *  .66581474d-05,.47933157d-05,.34841023d-05,.22100782d-05,
     *  .16591685d-05,.85248777d-06,.53236849d-06,.29243502d-06,
     *  .21037858d-06,.14169341d-06/

        data (rhotbl(i,6),i=1,60)/
     *  1.3004703,1.2340923,1.1704464,1.1094477,1.0510133,.99506132,
     *  .94151267,.84131448,.74981521,.66633385,.59045721,.52165360,
     *  .45941687,.39421853,.33819466,.29006526,.24872747,.21323105,
     *  .18275763,.15660248,.14495125,.13415895,.11490479,
     *  .98044963d-01,.83689297d-01,.61041925d-01,.4458692d-01,
     *  .32613851d-01,.23889470d-01,.17523275d-01,.12871319d-01,
     *  .93481175d-02,.68306486d-02,.50203988d-02,.37107363d-02,
     *  .20601264d-02,.11764019d-02,.69295459d-03,.36415558d-03,
     *  .18795236d-03,.95178066d-04,.47234922d-04,.19806644d-04,
     *  .14717842d-04,.93605867d-05,.79865441d-05,.58134902d-05,
     *  .36110878d-05,.30810159d-05,.17138945d-05,.98016301d-06,
     *  .68494827d-06,.48368536d-06,.28642162d-06,.20590009d-06,
     *  .9564537d-07,.5178214d-07,.2371248d-07,.139618d-07,.73908d-08/

        data (ctbl(i,6),i=1,60)/
     *  331.04545,329.29882,327.54287,325.77746,324.00243,322.21762,
     *  320.42287,316.80287,313.14103,309.46182,305.73835,301.96896,
     *  298.15192,297.92719,297.70229,297.47722,297.25198,297.02656,
     *  296.80098,296.57523,296.46228,296.34930,296.12320,296.46228,
     *  296.80098,297.47722,298.15192,298.82510,299.49677,300.16694,
     *  300.83561,303.76028,306.65707,309.52674,312.37005,317.98042,
```

```
     *  322.12407,322.12407,317.76835,313.35209,308.87269,304.32737,
     *  298.78172,296.91016,294.08048,294.08047,294.08047,294.08047,
     *  294.08047,294.08047,294.08047,294.08047,294.08047,294.08047,
     *  294.08047,294.08047,294.08047,294.08047,294.08047,294.08047/

c*************************************************************************
c               start spline calculation
c*************************************************************************

        n=nn(key)
        if(m.lt.1.or.m.gt.n) m=1

        if(ifirst.eq.0) then
           do i=1,n
              xt(i)=atbl(i,key)
              yt(i,1)=ptbl(i,key)
              yt(i,2)=rhotbl(i,key)
              yt(i,3)=ctbl(i,key)
           enddo
           ifirst=1
           atmosn=atmos(key)
        endif

        x=pr(1)
        if(x.lt.0.) x=0.
        if(x.ge.atbl(n,key)) then
           y(1)=ptbl(n,key)
           y(2)=rhotbl(n,key)
           y(3)=ctbl(n,key)
           pr(3)=0.
           pr(4)=0.
           pr(5)=0.
           go to 15
        endif

        if(x-xt(m).le.0.) then
    1      if(m.eq.1) go to 4
           m=m-1
           if(x.lt.xt(m)) go to 1
        else
    2      if(m.eq.n) go to 5
           m=m+1
           if(x-xt(m).gt.0.) go to 2
    5      m=m-1
        endif
    4   mp1=m+1

        dx=x-xt(m)
        dlx=xt(mp1)-xt(m)
        dlxi=1./dlx
        r=dx*dlxi
        r2=r*r

        if(m.eq.last) go to 34

        mm1=m-1
```



```
        mp2=mp1+1

        do i=1,3
           s(i)=(yt(mp1,i)-yt(m,i))*dlxi
        enddo

        if(m.ge.n-1) then
           dlxmi=1./(xt(m)-xt(mm1))
           do i=1,3
              sm1(i)=(yt(m,i)-yt(mm1,i))*dlxmi
              sp1(i)=2.*s(i)-sm1(i)
           enddo
        else
           dlxpi=1./(xt(mp2)-xt(mp1))
           do i=1,3
              sp1(i)=(yt(mp2,i)-yt(mp1,i))*dlxpi
           enddo
           if(m.le.1) then
              do i=1,3
                 sm1(i)=2.*s(i)-sp1(i)
              enddo
           else
              dlxmi=1./(xt(m)-xt(mm1))
              do i=1,3
                 sm1(i)=(yt(m,i)-y t(mm1,i))*dlxmi
              enddo
           endif
        endif

        do i=1,3
           a(i)=.5*(sp1(i)+sm1(i))-s(i)
           b(i)=.5*(s(i)-sp1(i))+s(i)-sm1(i)
           c(i)=.5*(s(i)+sm1(i))
        enddo

   34   if(k.eq.1) then
           do i=1,3
              y(i)=yt(m,i)+(a(i)*r2+b(i)*r+c(i))*dx
           enddo
        else
           do i=1,3
              y(i)=yt(m,i)+(a(i)*r2+b(i)*r+c(i))*dx
              pr(i+2)=3.*a(i)*r2+2.*b(i)*r+c(i)
           enddo
        endif

15      last=m

c*************************************************************************
c               end of spline calculation
c*************************************************************************

        pr(2)=y(1)
        pr(6)=y(2)
        pr(9)=y(3)
```

```
      tn=(y(3)/20.046707)**2
      pr(10)=1.458d-06*(tn**1.5)/(tn+110.4)
      pr(12)=tn
      if(k.eq.2) then
         pr(11)=pr(10)/y(3)*(3.-2.*tn/(tn+110.4))*pr(5)
         pr(13)=2.*tn/y(3)*pr(5)
      endif

      return
      end
```

3.20 Subroutine B0PTBL

3.20.1 Purpose

This subroutine reads the input namelist for the table output subroutine B1PTBL.

3.20.2 Variable Listing

3.20.3 Subroutines Called:

None

3.20.4 Calling Subroutines:

Name	Function
MASTRE	Controls total program logic

3.20.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
blk1	Blank character array				Local	
blk2	Blank character array				Local	
blk3	Blank character array				Local	
date	Date for output			output	Global	bopcom
ncase	Case number			output	Global	bopcom
nerr	Error index			output	Global	bopcom
notab	Table number				Global	bopcom
ntable	Flag indicating that output tables will (>0) or will not (=0) be generated			output	Global	alat
office	Character variable specifying name of originator			output	Global	bopcom
srid(30)	Character variable used for user input description of output tables from subroutine B1PTBL			output	Global	bopcom
title	Table title		b0ptbl	output	Global	bopcom

```
      SUBROUTINE B0PTBL
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      LOGICAL DEMAND,GRAPH
C
      CHARACTER*6 BLK1
      CHARACTER*12 OFFICE,DATE,at
      character*20 srid,blk2
      CHARACTER*48 TITLE,BLK3
C
      COMMON/CCC/GRAPH,JOUT,DEMAND

      COMMON/BOPCOM/NCASE,OFFICE,TITLE,DATE,SRID(30),NOTAB,NERR

      COMMON/ALAT/ALAT,ALONGO,ALTLS,NTABLE

      NAMELIST/INPUT2/NCASE,OFFICE,TITLE,DATE,SRID,NOTAB

      DATA BLK1/'      '/,BLK2/'                    '/
      DATA BLK3/'                                              '/

      IF(NTABLE.EQ.0) RETURN

      DO I=1,30
         SRID(I)=BLK2
      enddo

      OFFICE='            '
      TITLE=BLK3
      DATE='            '
      NERR=10

      WRITE(JOUT,'(1h1)')

      CLOSE(UNIT=16)

      OPEN(UNIT=16,FILE='ELTTBL.NL',STATUS='OLD',
     1  READONLY,SHARED)

      READ(16,INPUT2)

      WRITE(JOUT,6) NCASE
      WRITE(JOUT,7) OFFICE
      WRITE(JOUT,8) TITLE
      WRITE(JOUT,9) DATE
      WRITE(JOUT,10) (SRID(I),I=1,30)
      WRITE(JOUT,11) NOTAB
C
    6 FORMAT(1X,'NCASE=',I3)
    7 FORMAT(1X,'OFFICE=',A12)
    8 FORMAT(1X,'TITLE=',A48)
    9 FORMAT(1X,'DATE=',A12)
   10 FORMAT(1X,'SRID='/30(1X,A20/))
   11 FORMAT(1X,'NOTAB=',I3)
```



```
      RETURN
       END
```

3.21 Subroutine B1PTBL

3.21.1 Purpose

This subroutine outputs various publishable output tables based on a predefined set of requirements.

3.21.2 Variable Listing

3.21.3 Subroutines Called:

None

3.21.4 Calling Subroutines:

Name	Function
MASTRE	Control total program logic

3.21.5 Fortran Listing

b1ptbl

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
al	Tolerance on time between events	sec			Local	
aold	Previous value of aceleration				Local	
b	Parameters from output file				Local	
bk1	Blank format for variable output				Local	
bk2	Blank format for variable format output				Local	
date	Date for output			output	Global	bopcom
def	Name of parameters for use in output tables				Local	
flo	Propellant flowrate	lbs/sec			Local	
imax	Maximum number of parameters per output table				Local	
iunit	Name of units of output parameters				Local	
jfor	Temporary index				Local	
jpar	Parameter index				Local	
junit	Flag index				Local	
kt	Index to count number of thrust events				Local	
kunit	Index for unit				Local	
kw	Index to count number of weight drop events				Local	
level	Flag specifying propulsion summary output required				Global	tanks
lines	Index to specify number of lines on a page				Local	
mt	Number of output tables				Local	
na	Parameter names used in print output				Local	
name(128)	Parameter names used in print output			input	Global	name
nb	Formats used in variable format printouts				Local	
nbk	Formats used in variable format printouts				Local	
ncase	Case number			output	Global	bopcom
nfor	Variable format for parameter value printouts				Local	
nhead	Format used for variable output				Local	
nname	Index to specify the number used to specify the output parameters				Local	
nstop1	Format to end variable format for				Local	

Variable	Variable Definition output	Units	Source	I/O	Status	Common Block
nstop2	Format to end variable format for output				Local	
nstr1	Variable for start of variable format				Local	
nstr2	Variable for start of variable format				Local	
ntable	Flag indicating that output tables will (>0) or will not (=0) be generated			output	Global	alat
ntbl	Variable format for parameter names				Local	
nunit	Index for units used for output tables				Local	
nword	Number of parameters for output tables				Local	
office	Character variable specifying name of originator			output	Global	bopcom
prnt	Names of parameters on output tables				Local	
srid(30)	Character variable used for user input description of output tables from subroutine B1PTBL		b0ptbl	input	Global	bopcom
title	Table title		b0ptbl	input	Global	bopcom
tl	Time tolerance	sec			Local	
told	Previous time				Local	
trus	Local thrust	lbs			Local	
value	Value of parameters in output tables				Local	
wtot	Total vehicle weight	lbs			Local	

```
      SUBROUTINE B1PTBL
C
C     OUTPUT TABLE GENERATOR
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      LOGICAL DEMAND,GRAPH
C
      PARAMETER MT=14,ML=10,ML2=ML+2,ML3=ML+3

      CHARACTER*6 NBK(4),NSTR2(2),NSTOP2,NFOR(ML3),BK2,PRNT,NAME,nb
      CHARACTER*12 NHEAD,NSTR1,NTBL(ML2),NSTOP1,BK1,OFFICE,DATE
      CHARACTER*12 IUNIT(2,20)
      character*20 srid,na
      CHARACTER*48 DEF(128)
      CHARACTER*48 TITLE

      DIMENSION NWORD(128),NNAME(ML,MT),VALUE(ML),PRNT(ML),B(160),
     *NUNIT(128)

      DIMENSION UNIT(2,20)

      COMMON/ALAT/ALAT,ALONGO,ALTLS,NTABLE

      COMMON/CCC/GRAPH,JO,DEMAND

      COMMON/BOPCOM/NCASE,OFFICE,TITLE,DATE,SRID(30),NOTAB,NERR

      COMMON/NAME/NAME(128),WORD(160)

      common/tanks/level,mlox,nlox,wlox(100),hlox(100),mh2,nfuel,
     *wfuel(200),hfuel(200),wh21,woxl

      DATA NWORD/
C       0   1   2   3   4   5   6   7   8   9
     *       2,  2,  3,  2,  0,  0,  0,  2,  3,
     1  2,   2,  3,  0,  2,  3,  2,  3,  3,  0,
     2  0,   0,  2,  2,  2,  3,  3,  3,  3,  3,
     3  3,   3,  3,  3,  3,  3,  3,  0,  0,  0,
     4  0,   0,  0,  0,  0,  0,  0,  0,  0,  2,
     5  2,   2,  2,  2,  2,  2,  2,  2,  2,  2,
     6  2,   0,  0,  0,  0,  0,  0,  0,  0,  0,
     7  0,   0,  0,  2,  2,  2,  2,  2,  2,  2,
     8  2,   2,  2,  2,  2,  2,  2,  2,  0,  0,
     9  2,   2,  2,  2,  2,  2,  2,  3,  3,  3,
     *  1,   1,  1,  2,  0,  0,  1,  1,  1,  1,
     1  1,   2,  2,  2,  2,  1,  3,  2,  0,  3,
     2  3,   2,  2,  2,  3,  3,  3,  2,  2/

      DATA NSTR1/'(12X        '/,NSTR2/'(1X,A1','2     '/
      DATA NSTOP1/')           '/,NSTOP2/')     '/
      DATA NHEAD/',5X,A6,1X   '/
      DATA BK1/',12X        '/,BK2/',12X  '/
      DATA NBK/',F12.0',',F12.1',',F12.2',',F12.3'/

      DATA NNAME/
```



```
     *   1,  -7, -13,  -8,   9,  10,  11,  17,  12,  34,
     *   1,  -2,   3,   4, -14,  15,  16,-111, 112,  35,
     *   1, -19, -20, -21, -22, -23, -24, 124, 125, 126,
     *   1,  25,  26,  27,  31,  32,  33, 119, 120,   0,
     *   1, -37,  85, -38, -39, -40, -41,  -5,   0,   0,
     *   1, -37, -43,  49, -55, -61, -67,  73,  79,  85,
     *   1, -38, -44,  50, -56, -62, -68,  74,  80, -86,
     *   1, - 6,-127,-128,  18,  36,-116, 118,   0,   0,
     *   1, -40, -46,  52, -58, -64, -70,  76,  82,   0,
     *   1, -41, -47,  53, -59, -65, -71,  77,  83,   0,
     *   1,  97,  98,  99,-103,-104,-105, 121, 122, 123,
     *   1, -90, -96, -91, -92, -93, -94, -95,   0,   0,
     *   1,-100,-106,-109,-102,-108,-110,-101,-107,-115,
     *   1,  28,  29,  30, 113, 114, 117, -87, -88, -89/
C
C        0      1      2      3      4      5      6      7      8      9
C
C   0 ------  TIME   VSUBE   GAME   AZE    THRST  XMLB     R    VSUBI  GAMI
C
C   1   AZI   GCLAT  LONG   ALT    VSUBR  GAMR   AZR    GDLAT  NODE   Z8-X
C
C   2   X8-Y   Y8-Z  ZD8-W  XD8-U  YD8-V  CHIP   CHIY   CHIR   TIMP   LTIMP
C
C   3  LNGMP  THERI  PSIRI  PHIRI   INCL  RANGE  RNGAN  THMPS  THSRM  THLBM
C
C   4  THOMS  THRCS   TH    TVMPS  TVSRM  TVLBM  TVOMS  TVRCS   TVAC  ISPMP
C
C   5  ISPSR  ISPLB  ISPOM  ISPRC   ISP   WDMPS  WDSRM  WDLBM  WDOMS  WDRCS
C
C   6   WDOT  MPS F  SRM F  LBM F  OMS F  RCS F   FUEL  MPS D  SRM D  LBM D
C
C   7  OMS D  RCS D  SPENT  DPMPS  DPSRM  DPLBM  DPOMS  DPRCS   DELP   DYMPS
C
C   8  DYSRM  DYLBM  DYOMS  DYRCS   DELY   TAU   YRING  PDOME  SRMET  ORBET
C
C   9  CH VL   TN L   GV L   DR L  BKP L  GIM L  ID VL   MACH   ALPHA   BETA
C
C  10   FAA    FAS    FAN     Q    QALPH  QBETA   FAB    FMB    FNB    DRAG
C
C  11   LIFT  VWIND   AZW    HEAT  HRATE  PRESS   RHO    TEMP  REYNO  DELPC
C
C  12  DELYC  EMACH   INBD  OUTBD  AXACC  NRACC  LTACC   XCG    ZCG
C
      DATA DEF(  1)/'INSTANTANEOUS TIME FROM SRB IGNITION           '/
      DATA DEF(  2)/'EARTH-FIXED VELOCITY                           '/
      DATA DEF(  3)/'EARTH-FIXED FLIGHT PATH ANGLE                  '/
      DATA DEF(  4)/'EARTH-FIXED AZIMUTH                            '/
      DATA DEF(  5)/'TOTAL LOCAL THRUST                             '/
      DATA DEF(  6)/'TOTAL VEHICLE WEIGHT                           '/
      DATA DEF(  7)/'RADIUS FROM EARTH CENTER                       '/
      DATA DEF(  8)/'INERTIAL VELOCITY                              '/
      DATA DEF(  9)/'INERTIAL FLIGHT PATH ANGLE                     '/
      DATA DEF( 10)/'INERTIAL AZIMUTH                               '/
      DATA DEF( 11)/'GEOCENTRIC LATITUDE                            '/
      DATA DEF( 12)/'LONGITUDE                                      '/
      DATA DEF( 13)/'ALTITUDE ABOVE REFERENCE ELLIPSOID             '/
```

```
      DATA DEF( 14)/'WIND RELATIVE VELOCITY                                '/
      DATA DEF( 15)/'WIND RELATIVE FLIGHT PATH ANGLE                       '/
      DATA DEF( 16)/'WIND RELATIVE AZIMUTH                                 '/
      DATA DEF( 17)/'GEODETIC LATITUDE                                     '/
      DATA DEF( 18)/'DESCENDING NODE-MEASURED CCW FROM LAUNCH SITE         '/
      DATA DEF( 19)/'DOWNRANGE POSITION COMPONENT(INERTIAL)                '/
      DATA DEF( 20)/'VERTICAL POSITION COMPONENT(INERTIAL)                 '/
      DATA DEF( 21)/'CROSSRANGE POSITION COMPONENT(INERTIAL)               '/
      DATA DEF( 22)/'DOWNRANGE VELOCITY COMPONENT(INERTIAL)                '/
      DATA DEF( 23)/'VERTICAL VELOCITY COMPONENT(INERTIAL)                 '/
      DATA DEF( 24)/'CROSSRANGE VELOCITY COMPONENT(INERTIAL)               '/
      DATA DEF( 25)/'PITCH ATTITUDE (INERTIAL)                             '/
      DATA DEF( 26)/'YAW ATTITUDE (INERTIAL)                               '/
      DATA DEF( 27)/'ROLL ATTITUDE (INERTIAL)                              '/
      DATA DEF( 28)/'TIME TO VACUUM IMPACT POINT                           '/
      DATA DEF( 29)/'LATITUDE OF VACUUM IMPACT POINT                       '/
      DATA DEF( 30)/'LONGITUDE OF VACUUM IMPACT POINT                      '/
      DATA DEF( 31)/'PITCH ANGLE IN BOOSTER REF SYSTEM                     '/
      DATA DEF( 32)/'YAW(HEADING) ANGLE IN BOOSTER REF SYSTEM              '/
      DATA DEF( 33)/'ROLL ANGLE IN BOOSTER REF SYSTEM                      '/
      DATA DEF( 34)/'ORBITAL INCLINATION ANGLE                             '/
      DATA DEF( 35)/'RELATIVE SURFACE RANGE                                '/
      DATA DEF( 36)/'RANGE ANGLE                                           '/
      DATA DEF( 37)/'MPS LOCAL THRUST                                      '/
      DATA DEF( 38)/'SRM LOCAL THRUST                                      '/
      DATA DEF( 39)/'LBM LOCAL THRUST                                      '/
      DATA DEF( 40)/'OMS LOCAL THRUST                                      '/
      DATA DEF( 41)/'RCS LOCAL THRUST                                      '/
      DATA DEF( 42)/''                                                     '/
      DATA DEF( 43)/'MPS VACUUM THRUST                                     '/
      DATA DEF( 44)/'SRM VACUUM THRUST                                     '/
      DATA DEF( 45)/'LBM VACUUM THRUST                                     '/
      DATA DEF( 46)/'OMS VACUUM THRUST                                     '/
      DATA DEF( 47)/'RCS VACUUM THRUST                                     '/
      DATA DEF( 48)/''                                                     '/
      DATA DEF( 49)/'MPS SPECIFIC IMPULSE                                  '/
      DATA DEF( 50)/'SRM SPECIFIC IMPULSE                                  '/
      DATA DEF( 51)/'LBM SPECIFIC IMPULSE                                  '/
      DATA DEF( 52)/'OMS SPECIFIC IMPULSE                                  '/
      DATA DEF( 53)/'RCS SPECIFIC IMPULSE                                  '/
      DATA DEF( 54)/''                                                     '/
      DATA DEF( 55)/'MPS FLOWRATE                                          '/
      DATA DEF( 56)/'SRM FLOWRATE                                          '/
      DATA DEF( 57)/'LBM FLOWRATE                                          '/
      DATA DEF( 58)/'OMS FLOWRATE                                          '/
      DATA DEF( 59)/'RCS FLOWRATE                                          '/
      DATA DEF( 60)/''                                                     '/
      DATA DEF( 61)/'MPS FUEL ONBOARD                                      '/
      DATA DEF( 62)/'SRM FUEL ONBOARD                                      '/
      DATA DEF( 63)/'LBM FUEL ONBOARD                                      '/
      DATA DEF( 64)/'OMS FUEL ONBOARD                                      '/
      DATA DEF( 65)/'RCS FUEL ONBOARD                                      '/
      DATA DEF( 66)/''                                                     '/
      DATA DEF( 67)/'MPS FUEL SPENT                                        '/
      DATA DEF( 68)/'SRM FUEL SPENT                                        '/
      DATA DEF( 69)/'LBM FUEL SPENT                                        '/
```



```
      DATA DEF( 70)/'OMS FUEL SPENT                                        '/
      DATA DEF( 71)/'RCS FUEL SPENT                                        '/
      DATA DEF( 72)/''                                                     '/
      DATA DEF( 73)/'MPS PITCH GIMBAL ANGLE                                '/
      DATA DEF( 74)/'SRM PITCH GIMBAL ANGLE                                '/
      DATA DEF( 75)/'LBM PITCH GIMBAL ANGLE                                '/
      DATA DEF( 76)/'OMS PITCH GIMBAL ANGLE                                '/
      DATA DEF( 77)/'RCS PITCH GIMBAL ANGLE                                '/
      DATA DEF( 78)/''                                                     '/
      DATA DEF( 79)/'MPS YAW GIMBAL ANGLE                                  '/
      DATA DEF( 80)/'SRM YAW GIMBAL ANGLE                                  '/
      DATA DEF( 81)/'LBM YAW GIMBAL ANGLE                                  '/
      DATA DEF( 82)/'OMS YAW GIMBAL ANGLE                                  '/
      DATA DEF( 83)/'RCS YAW GIMBAL ANGLE                                  '/
      DATA DEF( 84)/''                                                     '/
      DATA DEF( 85)/'MPS THROTTLE FACTOR                                   '/
      DATA DEF( 86)/'Y RING PRESSURE                                       '/
      DATA DEF( 87)/'LOX BULKHEAD PRESSURE INDICATOR                       '/
      DATA DEF( 88)/'SRM-ET SHEAR LOAD INDICATOR                           '/
      DATA DEF( 89)/'ORBITER-ET ATTACH LOAD INDICATOR                      '/
      DATA DEF( 90)/'CHARACTERISTIC VELOCITY                               '/
      DATA DEF( 91)/'TURNING LOSSES                                        '/
      DATA DEF( 92)/'GRAVITY LOSSES                                        '/
      DATA DEF( 93)/'DRAG LOSSES                                           '/
      DATA DEF( 94)/'BACKPRESSURE LOSSES                                   '/
      DATA DEF( 95)/'GIMBAL LOSSES                                         '/
      DATA DEF( 96)/'IDEAL VELOCITY                                        '/
      DATA DEF( 97)/'MACH NUMBER                                           '/
      DATA DEF( 98)/'PITCH ANGLE OF ATTACK                                 '/
      DATA DEF( 99)/'SIDESLIP ANGLE                                        '/
      DATA DEF(100)/'AXIAL AERODYNAMIC FORCE                               '/
      DATA DEF(101)/'SIDE AERODYNAMIC FORCE                                '/
      DATA DEF(102)/'NORMAL AERODYNAMIC FORCE                              '/
      DATA DEF(103)/'DYNAMIC PRESSURE                                      '/
      DATA DEF(104)/'Q * ANGLE OF ATTACK                                   '/
      DATA DEF(105)/'Q * SIDESLIP ANGLE                                    '/
      DATA DEF(106)/'AXIAL BASE FORCE                                      '/
      DATA DEF(107)/'BASE PITCHING MOMENT                                  '/
      DATA DEF(108)/'NORMAL BASE FORCE                                     '/
      DATA DEF(109)/'AERODYNAMIC DRAG                                      '/
      DATA DEF(110)/'AERODYNAMIC LIFT                                      '/
      DATA DEF(111)/'WIND VELOCITY                                         '/
      DATA DEF(112)/'WIND AZIMUTH                                          '/
      DATA DEF(113)/'STAGNATION HEATING INDICATOR                          '/
      DATA DEF(114)/'STAGNATION HEATING RATE                               '/
      DATA DEF(115)/'ATMOSPHERIC PRESSURE                                  '/
      DATA DEF(116)/'ATMOSPHERIC DENSITY                                   '/
      DATA DEF(117)/'ATMOSPHERIC TEMPERATURE                               '/
      DATA DEF(118)/'REYNOLDS NUMBER                                       '/
      DATA DEF(119)/'COMPOSITE PITCH GIMBAL ANGLE                          '/
      DATA DEF(120)/'COMPOSITE YAW GIMBAL ANGLE                            '/
      DATA DEF(121)/'EARTH-RELATIVE MACH NUMBER                            '/
      DATA DEF(122)/'INBOARD ELEVON ANGLE                                  '/
      DATA DEF(123)/'OUTBOARD ELEVON ANGLE                                 '/
      DATA DEF(124)/'TOTAL AXIAL ACCELERATION                              '/
      DATA DEF(125)/'TOTAL NORMAL ACCELERATION                             '/
```

```
      DATA DEF(126)/'TOTAL LATERAL ACCELERATION                    '/
      DATA DEF(127)/'LONGITUDINAL CENTER OF GRAVITY LOCATION       '/
      DATA DEF(128)/'VERTICAL CENTER OF GRAVITY LOCATION           '/

      DATA IUNIT/
     *'SECONDS     ','SECONDS     ','METERS      ','FEET        ',
     *'METERS/SEC  ','FEET/SEC    ','KILOGRAMS   ','POUNDS      ',
     *'DEGREES     ','DEGREES     ','NEWTONS     ','POUNDS      ',
     *'KILOMETERS  ','NAUT MILES  ','KG/SEC      ','LBS/SEC     ',
     *'NEW/M**2    ','LBS/FT**2   ','N-DEG/M**2  ','LB-DEG/FT**2',
     *'BTU         ','BTU         ','BTU/SEC     ','BTU/SEC     ',
     *'METERS**2   ','FEET**2     ','KG/M**3     ','SLUGS/FT**3 ',
     *'KELVIN      ','RANKIN      ','G''S        ','G''S        ',
     *'CENTIMETERS ','INCHES      ','NEWTON-METER','FOOT-POUND  ',
     *'UNITLESS    ','UNITLESS    ','N/M**2      ','LBS/IN**2   '/

      DATA NUNIT/
C        0    1    2    3    4    5    6    7    8    9
     *        1,  -3,   5,   5,  -6,  -4,  -2,  -3,   5,
     1   5,   5,   5,  -2,  -3,   5,   5,   5,   5,  -2,
     2  -2,  -2,  -3,  -3,  -3,   5,   5,   5,   1,   5,
     3   5,   5,   5,   5,   5,  -7,   5,  -6,  -6,  -6,
     4  -6,  -6,  -6,  -6,  -6,  -6,  -6,  -6,  -6,   1,
     5   1,   1,   1,   1,   1,  -8,  -8,  -8,  -8,  -8,
     6  -8,  -4,  -4,  -4,  -4,  -4,  -4,  -4,  -4,  -4,
     7  -4,  -4,  -4,   5,   5,   5,   5,   5,   5,   5,
     8   5,   5,   5,   5,   5,  19, -20, -20,  -6,  -6,
     9   3,   3,   3,   3,   3,   3,   3,  19,   5,   5,
     *  -6,  -6,  -6,  -9, -10, -10,  -6, -18,  -6,  -6,
     1  -6,  -3,   5,  11,  12,  -9,  14,  15,  19,   5,
     2   5,  19,   5,   5,  16,  16,  16, -17, -17/

      DATA UNIT/1.,1.,.3048,3.28084,.3048,3.28084,2.2046223,.453592436,
     *1.,1.,.22480894,4.44822162,.5399568,1.852,2.2046223,.453592436,
     *.02088543,47.8802589,.02088543,47.8802589,1.,1.,1.,1.,
     *.09290304,10.76391,515.3842189,.0019403,1.,1.,1.,1.,
     *2.54,.393701,.73756218,1.3558179,1.,1.,.00014504,6894.7572/
      data al/5.0d-06/,tl/5.0d-04/

      IF(NTABLE.EQ.0.and.level.eq.0) RETURN

      aold=-10.0
      told=-1000.0
      kt=0

      DO 10 K=1,MT

      if(ntable.eq.0) then
         if(k.eq.1) go to 12
         return
      endif

      NTBL(1)=NSTR1
      NFOR(1)=NSTR2(1)
      NFOR(2)=NSTR2(2)
      IMAX=0
```



```
      DO 2 I=1,ML

      JPAR=IABS(NNAME(I,K))

      IF(JPAR.ne.0) then

         NTBL(I+1)=NHEAD
         JFOR=NWORD(JPAR)+1
         NFOR(I+2)=NBK(JFOR)
         PRNT(I)=NAME(JPAR)
         IMAX=IMAX+1

      else

         NTBL(I+1)=BK1
         NFOR(I+2)=BK2

      endif

    2 CONTINUE

      NTBL(ML2)=NSTOP1
      NFOR(ML3)=NSTOP2

12    REWIND 9

      KT=0
      KW=0
      LINES=1

    3 READ(9,end=10) NA,B,NB

      LINES=LINES-1

      IF(NA.EQ.'ENDPRT              ') GO TO 3
      if(ntable.eq.0) go to 13
      IF(NA.eq.'THRUST EVENT        ') then

         KT=KT+1

         NA=SRID(KT)

      endif

      IF(NA.eq.'WEIGHT DROP         ') then

         KW=KW+1
         NA=SRID(KW+15)

      endif

      IF(LINES.eq.0) then

         WRITE(21,100) OFFICE,DATE,TITLE,NCASE,K
         WRITE(21,NTBL)(PRNT(I),I=1,IMAX)
```

```
         WRITE(21,101)
         LINES=45

      endif

      DO 7 I=1,ML

      jpar=iabs(nname(i,k))

      if(jpar.ne.0) then

         kunit=iabs(nunit(jpar))

         if(nunit(jpar)*nname(i,k).gt.0) then

            value(i)=b(jpar)

         else

            junit=2
            if(nunit(jpar).lt.0) junit=1

            value(i)=b(jpar)*unit(junit,kunit)

         endif
      endif

    7 CONTINUE

      WRITE(21,NFOR) NA,(VALUE(I),I=1,IMAX)

   13 if(level.eq.0) GO TO 3

      if(k.gt.1) go to 3
      if(b(1)-told.le.t1) then
         if(abs(b(124)-aold).le.a1.or.b(124).le.a1) go to 3
c       set indicators for number of "same time" events
         kt=kt+1
         if(kt.gt.1) then
            backspace 7
         endif
      else
         kt=0
      endif

      told=b(1)
      aold=b(124)
      trus=b(43)
      if(level.eq.1) then
         flo=b(55)
         wtot=b(61)
      else
         flo=b(55)+b(56)
         wtot=b(61)+b(62)
      endif
```



```
      write(7,9) b(1),b(124),(b(i),i=131,134),flo,trus,wtot,b(135)
    9 format(f9.3,f10.4,2(f12.2,f15.2),3f15.2,f14.2)

      go to 3

   10 CONTINUE

      LINES=0

      DO 20 K=1,MT

      WRITE(21,101)

      DO 20 I=1,ML

      IF(LINES.le.0) then

         WRITE(21,200)
         LINES=40

      endif

      JPAR=IABS(NNAME(I,K))

      IF(JPAR.ne.0) then

         JUNIT=1
         IF(NNAME(I,K).LT.0) JUNIT=2

         KUNIT=IABS(NUNIT(JPAR))

         WRITE(21,201) NAME(JPAR),K,IUNIT(JUNIT,KUNIT),DEF(JPAR)

         LINES=LINES-1

      endif

   20 CONTINUE

  100 FORMAT(1H1/5X,A12,6X,A12,6X,A48,32X,6H  CASE,I3//47X,9HTABLE NO.,
     *I3/)
  101 FORMAT(/)
  200 FORMAT(1H1//41X,'DEFINITION OF TABLE OUTPUT'//9X,'PARAMETER',7X,
     *'TABLE',12X,'UNITS',16X,'DEFINITION'//)
  201 FORMAT(10X,A6,10X,I3,10X,A12,10X,A48)

      RETURN
      END
```

3.22 Subroutine BADLX

3.22.1 Purpose

This subroutine determines the influence coefficients (partial derivatives) of the objective function and the terminal and intermediate constraints with respect to the optimization parameters.

3.22.2 Variable Listing

3.22.3 Subroutines Called:

Name	Function
AFORUN	Controls forward trajectory integration
FUNISP	Calculates specific impulse based on throttle level

3.22.4 Calling Subroutines:

Name	Function
MASTRE	Control total program logic

3.22.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
aa	Launch azimuth	rad	ainit	output input	Global	agen2
aerod(30,17)	Aerodynamic coefficient tables		ainit	output input	Global	aerodi
astor(4,67)	Storage array for weight output table	lbs	ainit	input	Global	astor
cfr(15)	Critical flowrate	kg/sec	ainit	input	Global	xtra
cisp(15)	MPS specific impulse	sec	ainit	input	Global	xtra
cptbl(30)	Booster pitch attitude angle table	rad	badlx	output input	Global	contro
cytbl(30)	Booster yaw attitude angle table	rad	badlx	output input	Global	contro
daa	Increment value of launch azimuth used for calculation of partial derivatives	deg			Local	
dbeta	Incremental sideslip angle for calculation of partial derivatives	deg			Local	
ddelx(20)	Partial derivatives of the payoff and constraints with respect to the control parameters			output	Global	ddelx
del	State variable storage during partial evaluation		ainit	input	Global	delx
deltav	Delta velocity required	m/sec			Local	
delwz	Incremental weight used to vary liftoff weight	kg			Local	
delx(7)	State variable storage during partial evaluation		athrev	input	Global	delx
dtau	Incremental value of throttle for calculation of partial derivatives				Local	
dtime	Incremental time for calculation of partial derivatives	sec			Local	
dvisp	Specific impulse of upper stage	sec			Local	
engdat(8,15)	Engine data matrix		ainit	input	Global	agen1
fmxx	Temporary variable used for calculation of partial derivatives for tank constraints		athrev	input	Global	fmxx
ftbl(15)	Thrust table for main proplusion system	lbs	ainit	output input	Global	mps
glim(15)	Acceleration limit	g's	ainit	input	Global	mps
gzero	Gravitational constant (9.80665 m/sec)	m/sec^	ainit	input	Global	const
hratec	Aerodynamic heating rate	btu/sec		input	Global	comnew

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
hrates	Stored value of heat rate				Local	
ibranch	Flag indicating the terminal thrust event of the first trajectory of the branch trajectory			output	Global	mult
ipr	Denotes the thrust event from which fuel propellant reserve (FPR) is calculated			input output	Global	ipr
iprop(6)	Propulsion cutoff flags (stage dependent)		ainit	input	Global	ipr
irtflg	Flag indicating FPR calculations			output	Global	agen3
irtls	Return to launch flag			input	Global	agen3
it	Thrust event index used to calculate cutoff time based on margin				Local	
j1	Index				Local	
ja	Index				Local	
jj	Index				Local	
jthrot_save(1	Saved value of jthrot		atilt	input	Global	jthrot_save
jump	Jump start flag		ainit	input	Global	agen3
k1	Index				Local	
k2	Index				Local	
kcdphi(20)	Terminal constraint selection flag array		ainit	input	Global	yli
kdb(40)	Array for control parameter flags		ainit	output	Global	auto
lsb	Internal flag to indicate partial derivative runs			output	Global	rest
m	Index				Local	
maxhat	Index used to indicate the number of the constraint used for the maximum heating constraint			output	Global	heat
maxq	Index for maximum dynamic pressure constraint			output	Global	qmax
minh	Flag to indicate thrust event which begins min-h attitude optimization		ainit	input	Global	agen3
mpsflg(15)	Flag indicating the type of MPS thrust simulation		ainit	output	Global	agen3
msw	Temporay variable		athrev	input	Global	fmxx
mswch(15)	Flag indicating which thrust events are considered critical			output	Global	agen3
naltw	Index used when calculating launch			output	Global	naltw

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
	azimuth partial derivatives			input		
nbeta	Index used for booster sideslip attitude control		ainit	input	Global	taulim
nchiot(6)	Optimization flag used with branch trajectory option		ainit	input	Global	mult
ncnrs(5)	Number of constraints at restart		ainit	input	Global	rest
noss	Number of constraints		mastre	input	Global	bgen3
nstg(15)	Internal index which relates thrust event to stage number		ainit	input	Global	mult
ntau	Index used in conjunction with timmps to indicate a throttle event in the booster stage			output input	Global	taulim
ntcn	Number of intermediate constraints		ainit	input	Global	rest
phite(20)	Values of payoff and constraints	variabl		input output	Global	yli
prk4	Temporary variable used in calculation of performance reserve option		ainit	input	Global	ipr
prk5(5)	Same as prk4 but stage dependent (used with branch trajectory option)		athrev	input	Global	ipr
ptfmarg	Partial derivative of final time with respect to margin				Local	
ptfwlo	Partial derivative of final time with respect to liftoff weight				Local	
ptgwo	Partial derivative of time to acceleration limit with respect to liftoff weight				Local	
ptjwo	Partial derivative of duration time of jth thrust event with respect to liftoff weight				Local	
ptkg	Conversion from pounds to kilograms	kg/lbs	ainit	input	Global	const
ptlwo	Partial derivative of duration time of lth thrust event with respect to liftoff weight				Local	
ptn	Conversion from pounds to newtons	nt/lbs	ainit	input	Global	const
qpen	Maximum value of dynamic pressure	kg/m^2	aforun	input	Global	qmax
qpens	Stored value of maximum dynamic pressure				Local	
spradb	Increment in forward difference to evaluate partial derivative of payoff and constraints with respect to first stage pitch attitude variations	rad	ainit	input	Global	delpar

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
syradb	Increment in forward difference to evaluate partial derivative of payoff and constraints with respect to first stage yaw attitude variations	rad	ainit	input	Global	delpar
taut(15)	Time increments for thrust events	sec	anewch	output input	Global	agen1
tau_const(10,	Constant value of throttle used for constant throttle after acceleration limit		ainit	input	Global	mps
tbias(6)	Bias between simulated time and reported time (used for branch option)	sec	athrev	input output	Global	mult
tbias_save	Saved value of tbias	sec			Local	
temp	Temporary variable				Local	
temp1	Temporary variable				Local	
temp2	Temporary variable				Local	
thr	Vacuum thrust of fixed engines	nt			Local	
thrm	Thrust of mth trottle setting	nt			Local	
throt(15)	Throttle table for MPS motor		ainit	input	Global	mps
timmps(15)	Time table for MPS motor	sec	ainit	output input	Global	mps
timtau	Time for throttle event in first stage	sec	ainit	output input	Global	taulim
tne(6,15)	Array which defines the number of engines per thrust event		ainit	input	Global	agen1
var(25)	State variable array			input	Global	forint
vrcut	Magnitude of characteristic velocity cutoff	m/sec	ainit	input	Global	agen1
wddump(15)	Amount of weight dumped during OMS/RCS thrust events	kg	ainit	input	Global	omsrcs
wdoms(15)	Weight flowrate of OMS engines	lbs/sec	ainit	input	Global	omsrcs
wdot	Propellant flowrate of "it" thrust event	kg/sec			Local	
wdotit	Propellant flowrate of "it" thrust event	kg/sec			Local	
wdotm	Propellant flowrate of throttle	kg/sec			Local	
wdrcs(15)	Weight flowrate of RCS engines	lbs/sec	ainit	input	Global	omsrcs
wzero	Initial vehicle weight	lbs	badlx	output input	Global	agen2
xlamb(40,20)	Partial derivatives of payoff and constraints with respect to the optimization parameters		bkend	input	Global	auto

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
xmd(15)	Propellant flowrate	kg/sec	ainit	input	Global	xtra
xmtj	Flowrate of jth thrust event	kg/sec			Local	
xmtl	Flowrate of lth thrust event	kg/sec			Local	
yl(7,20)	Integrated values of adjoint equations of motion		dpir	input	Global	bake
ylbt(15,20)	Partial derivatives of the constraints with respect to the thrust event time intervals		bkend	input	Global	blamb

```
      SUBROUTINE BADLX
C
C
C       Sam Powell
c       changes for heat rate constraint
c
C      DETERMINES THE INFLUENCE COEFFICIENTS(PARTIAL DERIVATIVES)OF THE
C     PAYOFF AND CONSTRAINTS WITH RESPECT TO THE PARAMETERS
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      LOGICAL GLIMIT,QFLG
C
      COMMON/COMNEW/HRATEC

      COMMON/HEAT/MAXHAT

      CHARACTER*60 HEAD
      CHARACTER*6 MISION

      COMMON/AERODI/AEROD(30,17),AEROB(50,6,3),TOMACH(25),DELCA(25),
     *PNMONE(15),BKFABT(50,2),BKFTHR(50,2),DELCN(25),DELCM(25)

      COMMON/AGEN1/TIME(2,16),TAUT(15),TAUW(15),TZERO,TLIFT,TTILT,TMINH,
     *DTZ,TQ,TL,XMAUG,TNE(6,15),ENGDAT(8,15),S(15),WD(15),WJET(15),
     *HEAD,PRINT(15),STEP(15),HNOM(15),HMAX(15),PROP(6),TBEGR,TENDR,
     *CHRDOT,CHIRO,FAZ,BO(3,2),CBAXIL,TCHIR,VRCUT,FPRFAC,CHVEL

      COMMON/AGEN2/AA,SA,CA,ALF,ALFY,CHIP,SCHIP,CCHIP,CHIY,SCHIY,CCHIY,
     *CHIR,SCHIR,CCHIR,STH,CTH,STHL,CTHL,DPHIZ,RTHE,R,VR,XM,SW,SU,SV,Q,
     *VIV(25),DVARS(13),DELXDW(15,3),DELXDR(7,5),DELXD(15,7),WZERO,ALT,
     *XMACH,QDOT,FAA,FAN,A(3,3),CASE,CT2,UMF,A12W,A22W,A32W,XJEXT,BEU,
     *BYL,BEL,BHMAX(15),BHMN,BSTEP(15),HNMB,HMXB,TPOLY,XMIAD

      COMMON/AGEN3/ITHR,IWD,IXR,JUMP,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISION,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)

      COMMON/AUTO/KDB(40),SAVCP(40),XLAMB(40,20),DP2

      common/astor/astor(4,67)

      COMMON/BAKE/WISS(20,20),YL(7,20)

      COMMON/BGEN3/NOSS,JTB,NENT,NACT,NOM1,NP(7)

      COMMON/BLAMB/YLBT(15,20),YLBW(15,20),GAPHI(20),PROD(2,20)

      COMMON/BODY/TBDWT(15,2),TXCG(15,2),TYCG(15,2),XLEN(2),XREF(2),
     *YREF(2),KP(2),KY(2),AP(10,2),AY(10,2)

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM

      COMMON/CONTRO/TTBL(30),CPTBL(30),CYTBL(30),TOBL(210),
```



```
     *CPOTBL(210),CYOTBL(210)

      COMMON/DDELX/DDELX(20)

      COMMON/DELPAR/SPRADB,SYRADB

      COMMON/DELX/DELX(7)

      COMMON/FMXX/FMXX,MSW,WG,SPOA

      COMMON/FORINT/HBANK(2),T,TT,VAR(25)                    ,DVAR(25),FSAVE
     *(625),NTRG1,TV1,NTRG2,TV2,NTRG3,TV3,NTRG4,TV4,NTRG5,TV5,NTRG6,TV6,
     *NTRG7,TV7,NTRG8,TV8,NTRG9,TV9,NTRG10,TV10,NTRG11,TV11,NTRG12,TV12,
     *NTRG13,TV13,KIND,KRDER,EU,EL,AYL,HMX,HMN,HNM

      COMMON/IPR/IPR,WPMX,ILAST,KDT(15),PRK4,PRTOT,iprop(6),prk5(5)

      common/jthrot_save/jthrot_save(15)

      COMMON/LBM/TIMLBM(30),THRLBM(30),WDTLBM(30),PLBM(2),MLBM1,MLBM2,
     *ALBM(2),BLBM(2),CLBM(2),VLBM(2)

      COMMON/MPS/TIMMPS(15),FTBL(15),THROT(15),GLIM(15),TAUALT(5),
     *TAUTBL(5),MPS1,MPS2,TGLIM(10,15),GLIMIT,tau_const(10,5),jthrot(15)

      common/mult/nchiot(6),tbias(6),nstage,nstg(15),wf(5),ibranch

      common/multi/char_vel(5),fpr_fac(5),w_margin(2)

      COMMON/OMSRCS/FOMS(15),FRCS(15),WDOMS(15),WDRCS(15),WDDUMP(15)

      COMMON/QMAX/QMAX,QFLG,QPEN,MAXQ

      COMMON/NALTW/NALTW

      COMMON/REST/JSTR,NCNRS(5),NTCN,TR(5),LSB,NF8,NOPAR,WIBT(15)

      COMMON/TAULIM/TAULIM,NTAU,NBETA,TIMTAU

      COMMON/XTRA/XMD(15),CFR(15),F(15),CISP(15)

      COMMON/YLI/YLINT(7,20),PHITE(20),PSIREQ(20),KCDPHI(20),PSTR(20)

      dimension tbias_save(6)

      DATA DTIME/.1/,DTAU/.1/,DBETA/.1/

      DATA DELWZ/100./,DAA/.125/
C
      LSB=1

      QPENS=QPEN

c       Get lambda beta for tauts where variable weight flowrate are used

        if(naltw.eq.0) then
```

```
            jj=minh-1
            j1=1

            do i=1,15
               if((mpsflg(i).eq.5.or.mpsflg(i).eq.6).and.i.gt.jj) jj=i
            enddo

c10         do i=j1,jj
10          do i=j1,15

               if(kdb(i).eq.1) then

                  naltw=i
                  taut(i)=taut(i)+dtime

                  do k=1,20
                     ddelx(k)=phite(k)
                  enddo

                  do k=1,6
                     tbias_save(k)=tbias(k)
                  enddo

                  return

               endif
            enddo

         else

            if(naltw.lt.15) then

               do i=1,noss
                  xlamb(naltw,i)=(ddelx(i)-phite(i))/(-dtime)
               enddo

               do i=1,20
                  phite(i)=ddelx(i)
               enddo

               taut(naltw)=taut(naltw)-dtime

               j1=naltw+1
               naltw=0

               do k=1,6
                  tbias(k)=tbias_save(k)
               enddo

               go to 10

            endif
         endif
```



```
C     GET LAMBDA BETA 16 FOR LIFTOFF WEIGHT

      IF(KDB(16).ne.0.and.naltw.eq.0) then

         IF(JUMP.ne.1) then

            DO I=1,NOSS
               XLAMB(16,I)=YL(7,I)
            enddo

         else

            WZERO=WZERO+DELWZ

            CALL AFORUN

            WZERO=WZERO-DELWZ

            DO I=1,7
               DDELX(I)=(DELX(I)-VAR(I))/(-DELWZ)
            enddo

            DO I=1,NOSS
               XLAMB(16,I)=0.
               DO J=1,7
                  XLAMB(16,I)=XLAMB(16,I)+DDELX(J)*YL(J,I)
               enddo
            enddo

            if(ibranch.ne.0) xlamb(16,1)=2.0
            IF(MAXQ.NE.0) XLAMB(16,MAXQ)=(QPEN-QPENS)/DELWZ

         endif
      endif
C
C     GET LAMBDA BETAS 18-24 FOR CPTBL
C
      DO 75 K=1,7

      IF(KDB(K+17).ne.0.and.naltw.eq.0) then

         CPTBL(K+1)=CPTBL(K+1)+SPRADB

         CALL AFORUN

         CPTBL(K+1)=CPTBL(K+1)-SPRADB

         DO I=1,7
            DDELX(I)=(DELX(I)-VAR(I))/(-SPRADB)
         enddo

         DO I=1,NOSS
            XLAMB(K+17,I)=0.
            DO J=1,7
               XLAMB(K+17,I)=XLAMB(K+17,I)+DDELX(J)*YL(J,I)
            enddo
```

```
         enddo

         IF(MAXQ.NE.0)THEN
            XLAMB(K+17,MAXQ)=0.

            IF(PHITE(MAXQ).GT.0.)THEN

               XLAMB(K+17,MAXQ)=(QPEN-QPENS)/SPRADB
               WRITE(6,*)'OPTIMIZING PITCH FOR Q'

            ELSE

               WRITE(6,*)'NOT OPTIMIZING PITCH FOR Q'

            ENDIF
         ENDIF
C
         IF(MAXHAT.NE.0)THEN

            XLAMB(K+17,MAXHAT)=0.

            IF(PHITE(MAXHAT).GT.0.)THEN
               XLAMB(K+17,MAXHAT)=.03*(HRATEC-HRATES)/SPRADB
               WRITE(6,*)'OPTIMIZING PITCH FOR HRATE'
C        WRITE(6,*)'PHITE XLAMB ',PHITE(MAXHAT),XLAMB(K+17,MAXHAT)
            ELSE
               WRITE(6,*)'NOT OPTIMIZING PITCH FOR HRATE'
            ENDIF

         ENDIF

      endif

75    continue
C
C
C     GET LAMBDA BETAS 25-28 FOR CYTBL
C
      DO 76 K=1,4

      IF(KDB(K+24).ne.0.and.naltw.eq.0) then

         CYTBL(K+1)=CYTBL(K+1)+SYRADB

         CALL AFORUN

         CYTBL(K+1)=CYTBL(K+1)-SYRADB

         DO I=1,7
            DDELX(I)=(DELX(I)-VAR(I))/(-SYRADB)
         enddo

         DO I=1,NOSS
            XLAMB(K+24,I)=0.
            DO J=1,7
               XLAMB(K+24,I)=XLAMB(K+24,I)+DDELX(J)*YL(J,I)
```



```
            enddo
         enddo

         IF(MAXQ.NE.0)THEN

            XLAMB(K+24,MAXQ)=0.

            IF(PHITE(MAXQ).GT.0.)THEN

               XLAMB(K+24,MAXQ)=(QPEN-QPENS)/SYRADB
               WRITE(6,*)'OPTIMIZING YAW FOR Q'

            ELSE

               WRITE(6,*)'NOT OPTIMIZING YAW FOR Q'

            ENDIF

         ENDIF

      endif

   76 CONTINUE
C
C     GET LAMBDA BETA 17 FOR AZIMUTH
C
      IF(KDB(17).ne.0) then

         if(naltw.eq.0) then

            NALTW=17
            AA=AA+DAA/RAD

            DO I=1,20
               DDELX(I)=PHITE(I)
            enddo

            RETURN

         else

            DO I=1,NOSS
               XLAMB(17,I)=(DDELX(I)-PHITE(I))/(-DAA)
            enddo

            IF(MAXQ.NE.0) XLAMB(17,MAXQ)=(QPEN-QPENS)/DAA

            DO I=1,20
               PHITE(I)=DDELX(I)
            enddo

            AA=AA-DAA/RAD

            NALTW=0

         endif
```

```
      endif
C
C     GET LAMBDA BETAS 29-31 FOR THROTTLE CONTROL
C
      IF(NTAU.ne.0) then

         DO 172 K=1,3

         IF(KDB(K+28).ne.0) then

            IF(K.EQ.1) TIMMPS(NTAU)=TIMMPS(NTAU)+DTIME
            IF(K.EQ.2) TIMTAU=TIMTAU+DTIME
            IF(K.EQ.3) FTBL(NTAU+1)=FTBL(NTAU+1)+DTAU/100.

            CALL AFORUN

            IF(K.EQ.1) TIMMPS(NTAU)=TIMMPS(NTAU)-DTIME
            IF(K.EQ.2) TIMTAU=TIMTAU-DTIME
            IF(K.EQ.3) FTBL(NTAU+1)=FTBL(NTAU+1)-DTAU/100.

            DEL=DTIME
            IF(K.EQ.3) DEL=DTAU

            DO I=1,7
               DDELX(I)=(DELX(I)-VAR(I))/(-DEL)
            enddo

            DO I=1,NOSS
               XLAMB(K+28,I)=0.
               DO J=1,7
                  XLAMB(K+28,I)=XLAMB(K+28,I)+DDELX(J)*YL(J,I)
               enddo
            enddo

            IF(MAXQ.NE.0) XLAMB(K+28,MAXQ)=(QPEN-QPENS)/DEL

         endif

172      continue

      endif
C
C     GET LAMBDA BETA 32-34 FOR SIDESLIP CONTROL
C
      IF(NBETA.ne.0) then

         DO 156 K=1,3

         IF(KDB(K+31).ne.0) then

            L=NBETA+K-1
            AEROD(L,15)=AEROD(L,15)+DBETA/RAD

            CALL AFORUN
```



```
            AEROD(L,15)=AEROD(L,15)-DBETA/RAD

            DO I=1,7
               DDELX(I)=(DELX(I)-VAR(I))/(-DBETA)
            enddo

            DO I=1,NOSS
               XLAMB(K+31,I)=0.
               DO J=1,7
                  XLAMB(K+31,I)=XLAMB(K+31,I)+DDELX(J)*YL(J,I)
               enddo
            enddo

            IF(MAXQ.NE.0) XLAMB(K+31,MAXQ)=(QPEN-QPENS)/DBETA

         endif

156      continue

      endif

c     Get lambda betas 35 and 36 for margin

      ja=noss-ntcn

      do 158 i=5,1,-1

      if(iprop(i).ne.0.and.(kdb(35).eq.1.or.kdb(36).eq.1)) then

         it=iprop(i)
         thr=(tne(1,it)+tne(6,it))*engdat(6,1)*ptn
         wdotit=throt(it)*thr/(gzero*funisp(throt(it),i))

         if(mpsflg(it).eq.5.and.jthrot_save(it).gt.0) then

            wdot=tau_const(jthrot_save(it),i)*thr/(gzero*
     *           funisp(tau_const(jthrot_save(it),i),i))

            wdotm=throt(it)*thr/(gzero*funisp(throt(it),i))

            ptfwlo=1./wdotm-prk5(i)/wdot

         else

            wdot=wdotit
            ptfwlo=(1.-prk5(i))/wdot

         endif

         ptfmarg=-prk5(i)/wdot

         k1=1
         k2=ja

         j=nchiot(i)
```

```
      if(j.ne.0) then

         k1=ja+1
         k2=ja+ncnrs(j)

      endif

      do 258 l=k1,k2

      if(kdb(35).ne.0.and.j.eq.0) then
         if(l.eq.1) then
            xlamb(35,l)=prk5(i)
         else
            xlamb(35,l)=xlamb(it,l)*ptfmarg
         endif
      else
         xlamb(35,l)=0.
      endif

      if(kdb(36).ne.0.and.j.ne.0) then
         xlamb(36,l)=xlamb(it,l)*ptfmarg
      else
         xlamb(36,l)=0.
      endif

      if(kdb(16).ne.0) xlamb(16,l)=xlamb(16,l)+xlamb(it,l)*ptfwlo

      do 258 m=1,it-1

      if(kdb(m).ne.0.and.(m.le.ibranch.or.nstg(m).eq.nstg(it))) then

         if(jthrot_save(m).gt.0) then

            thrm=tau_const(jthrot_save(m),nstg(m))*(tne(1,m)+
     *           tne(6,m))*engdat(6,1)*ptn

            wdotm=thrm/(gzero*funisp(tau_const(jthrot_save(m),
     *             nstg(m)),nstg(m)))

         else

            thrm=throt(m)*(tne(1,m)+tne(6,m))*engdat(6,1)*ptn

            wdotm=thrm/(gzero*funisp(throt(m),nstg(m)))

         endif

         if(l.eq.1) then
            xlamb(m,l)=0.
         else
            xlamb(m,l)=xlamb(m,l)+xlamb(it,l)*(1.-wdotm/wdotit)
         endif

      endif

258   continue
```

257


```
         if(maxq.ne.0) then
            xlamb(35,maxq)=0.
            xlamb(36,maxq)=0.
         endif

      endif

      if(iprop(i).ne.0.and.tne(4,iprop(i)).ne.0.and.kdb(16).eq.1) then

         deltav=astor(3,5)*.3048
         dvisp=astor(4,5)
         temp=exp(-deltav/(gzero*dvisp))
         it=iprop(i)

         ptfwlo=-(1.-temp)/(wdoms(it)*ptkg*temp)

         k1=1
         k2=ja

         j=nchiot(i)

         if(j.ne.0) then

            k1=ja+1
            k2=ja+ncnrs(j)

         endif

         do l=1,k2
            if(l.eq.1.or.l.ge.k1) then
               if(l.eq.1) then
                  xlamb(16,1)=xlamb(16,1)-wdoms(it)*ptkg*ptfwlo
               else
                  xlamb(16,1)=xlamb(16,1)+xlamb(it,1)*ptfwlo
               endif
            endif
         enddo
      endif

158   continue

      IF(IPR.ne.0.and.IRTLS.ne.0) then

         IF(KDB(16).EQ.0) return

         J=IRTLS
         L=IPR

         XMTL=TNE(1,L)*XMD(L)+WDOMS(L)+WDRCS(L)+WDDUMP(L)
         XMTJ=TNE(1,J)*XMD(J)+WDOMS(J)+WDRCS(J)+WDDUMP(J)

         PTJWO=-WDOMS(L)/(XMTL*(WDOMS(J)+WDDUMP(J))-WDOMS(L)*XMTJ)
         IF(IRTFLG.EQ.1) PTJWO=1./XMTJ

         PTGWO=1./XMTL*(1.-XMTJ*PTJWO)
```

```
      TEMP=TNE(1,J)*CFR(J)*PTJWO+TNE(1,L)*CFR(L)*PTGWO
      PTLWO=PTGWO+FMXX*TEMP

      DO 161 I=1,NOSS

      IF(KCDPHI(I).ne.15) then

         IF(KCDPHI(I).eq.1) then

            XLAMB(16,I)=(1.-PRK4)/(CFR(L)/XMD(L)-PRK4)*TEMP

         else

            XLAMB(16,I)=XLAMB(16,I)+PTJWO*YLBT(J,I)+PTLWO*YLBT(L,I)

         endif

      endif

 161  CONTINUE

    else IF(IPR.ne.0) then

C***  ADJUST LAMB1 FOR PERF RES

      IF(MSW.ne.0) then

         TEMP=1./(TNE(1,IPR)*XMD(IPR))+FMXX*(CFR(IPR)/XMD(IPR))
         TEMP1=1.
         IF(MSWCH(IPR+1).EQ.2) TEMP1=EXP(-VRCUT/(GZERO*CISP(IPR+1)))
         TEMP2=0.
         IF(MSWCH(IPR+1).EQ.2) TEMP2=-TAUT(IPR+1)*GLIM(IPR)/
     *   CISP(IPR)*(TEMP-1./(TNE(1,IPR)*XMD(IPR)))

         DO 150 I=1,NOSS

         IF(KCDPHI(I).ne.15) then

            IF(KCDPHI(I).eq.1) then

               XLAMB(16,I)=(1.-PRK4)/(CFR(IPR)/XMD(IPR)-PRK4*
     *                     TEMP1-(1.-TEMP1))*CFR(IPR)/XMD(IPR)*
     *                     TEMP1
            else

               XLAMB(16,I)=XLAMB(16,I)+TEMP*YLBT(IPR,I)+TEMP2*
     *                      YLBT(IPR+1,I)

            endif

         endif

 150     CONTINUE

       else
```



```
         TEMP=PRK4/((1.-PRK4)*XMD(IPR)*TNE(1,IPR))

         DO 16 I=1,NOSS

         IF(KCDPHI(I).EQ.1)YLBT(IPR,I)=0.

         IF(KCDPHI(I).ne.15) XLAMB(16,I)=XLAMB(16,I)-TEMP*YLBT(IPR,I)

 16      CONTINUE

       endif

    endif

C*****
C     END OF LAMB1 AND LAMB2

      RETURN
      END
```

3.23 Subroutine BAKRN

3.23.1 Purpose

This subroutine controls the setup logic for the backward trajectory integration and calls the integration subroutines.

3.23.2 Variable Listing

3.23.3 Subroutines Called:

Name	Function
BDR1I	Calculates components of the equations of motion for the backward trajectory
BDRI	Calculates equations of motion for the backward trajectory
BGLIM	Controls acceleration limit events during backward integration
BKEND	Terminates backward integration
BSAVEG	Calculates impulse response functions and integrates weighting matrix
BTHREV	Controls thrust event logic during backward integration
BWDEV	Controls weight drop event logic during backward integration
DESOLV	Initializes integration constants and calls integration routine

3.23.4 Calling Subroutines:

Name	Function
MASTRE	Control total program logic

3.23.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
bel	Lower error limit for backwards integration	sec		output	Global	agen2
beu	Upper limit for backwards integration	sec		output input	Global	agen2
bglim	Integration type flag for backwards trajectory			output	Global	agen3
bhmn	Minimum stepsize for backwards integration	sec		output	Global	agen2
bkend	Integration type flag for backwards trajectory			output	Global	agen3
byl	Lower dependent variable value for auto stepsize	sec		output	Global	agen2
delxdr(7,5)	State derivative storage at desired intermediate point		athrev	input	Global	agen2
delxh(5)	Storage array for stagnation heating rate at intermediate constraint thrust events		athrev	output	Global	delxh
drngdt(2)	Time derivative of range from launch site		acstop	input	Global	drngdt
hmxb	Maximum step size for backward integration	sec		output	Global	agen2
ithr	Thrust event index number		athrev	output input	Global	agen3
iwd	Index for weight drop events		bakrn	output	Global	agen3
ixr	Index for state variable tables		bakrn	input output	Global	agen3
jtb	Index for control table points		bakrn	output	Global	bgen3
jthrot(15)	Index indicating number of times the acceleration limit has been reached within a given thrust event		atilt(gli	output input	Global	mps
kat	Indicator for positive definite values of huu matrix		bakrn	output	Global	agen3
kcdres(6,5)	Array for intermediate constraint function code		ainit	input	Global	enput
kcytab	Attitude control table index		bakrn	output input	Global	agen3
kerr	Integration error indicator				Local	
kindb	Integration type flag for backwards trajectory		ainit	input	Global	agen3
krderb	Order of differences for backwards integration		ainit	input	Global	agen3

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ll	Dimension of internal array in integration routine				Local	
lldsol	Dimension of internal array in integration routine				Local	
maxhat	Index used to indicate the number of the constraint used for the maximum heating constraint			output	Global	heat
mbs1	Pointer index for aerodynamic lookup in backwards integration			output	Global	bodyb
mbs18	Pointer index for viscous drag interpolation in backwards integration			output	Global	vaero
mbs19	Previous index for viscous drag interpolation in backwards integration			output	Global	vaero
mbs2	Previous index for aerodynamic interpolation in backwards integration			output	Global	bodyb
mbs3	Pointer index for base force interpolation in backwards integration			output	Global	bodyb
mbs4	Previous index for base force interpolation in backwards integration			output	Global	bodyb
mbs5	Index for state interpolation			output	Global	bodyb
mbs6	Not used			output	Global	bodyb
mbs7	Pointer index for center of gravity interpolation in backwards integration			output	Global	bodyb
mbs8	Previous index for center of gravity interpolation in backwards integration			output	Global	bodyb
minh	Flag to indicate thrust event which begins min-h attitude optimization		ainit	output	Global	agen3
nact	Actual number of differential equations to solve			output	Global	bgen3
nbtrg1	Termination trigger for backward integration			output	Global	bakint
nbtrg2	Thrust event trigger for backward integration			output	Global	bakint
nbtrg3	Weight drop event trigger for backward integration			output	Global	bakint
nbtrg4	Impulse response function trigger for backwards integration			output	Global	bakint
nbtrg5	Acceleration limit trigger for backwards integration			output	Global	bakint
ncnrs(5)	Number of constraints at restart		ainit	input	Global	rest

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
nctn	Integration type flag for backwards trajectory			output	Global	agen3
nent	Number of points in state variable tables		bakrn	output	Global	bgen3
nomor	Index indicating end of integration			output	Global	nomor
noss	Number of constraints		bakrn	input output	Global	bgen3
nvnt	Number of thrust events		ainit	input	Global	agen3
nvrst(5)	Array specifying the thrust events where intermediate constraints will be imposed		ainit	output	Global	enput
nwvnt	Number of weight drop events		ainit	input	Global	agen3
prtot	Total propellant weight used for performance reserve option	kg	ainit	input	Global	ipr
t	Time from lift-off	sec	dpir	input	Global	forint
tbk	Time used in backward integration	sec		output input	Global	bakint
tbkd	Time used in backward integration			output	Global	bakint
tbv1	Termination event trigger time during backward integration		bakrn	output	Global	bakint
tbv2	Thrust event trigger time during backward integration		bakrn	output	Global	bakint
tbv3	Weight drop event trigger time during backward integration		bakrn	output	Global	bakint
tbv4	Time to next determination of impulse response function	sec	bakrn	output	Global	bakint
tbv5	Time of acceleration limit event	sec	bakrn	output	Global	bakint
tend	Termination time of forward trajectory (start of backward intgration)	sec	bakrn	output input	Global	bakint
tglim(10,15)	Time of acceleration limit	sec	atilt	input	Global	mps
time(2,16)	Actual times for thrust events measured from launch	sec	aforun	input	Global	agen1
tq	Time that the minh phase is initiated	sec	aforun	input	Global	agen1
tr(5)	Intermediate constraint absolute time	sec	bakrn	output	Global	rest
wiss(20,20)	Weighting matrix		bsaveg	output	Global	bake
xmaug	Auxiliary vehicle mass	kg	athrev	output input	Global	agen1
yl(7,20)	Integrated values of adjoint equations of motion		afornd	output input	Global	bakint

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ylbt(15,20)	Partial derivatives of the constraints with respect to the thrust event time intervals		bakrn	output	Global	blamb
ylint(7,20)	Initial state variables of the adjoint equations of motions for backward integration		afornd	input	Global	yli

```
      SUBROUTINE BAKRN                                                  BAKR   1
C
C      CONTROLS THE SETUP LOGIC FOR THE BACKWARD TRAJECTORY INTEGRATION
C     AND CALLS THE INTEGRATION PACKAGE
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      LOGICAL GLIMIT
C
      CHARACTER*60 HEAD
      CHARACTER*6 MISION
      character*12 HEADER

      COMMON/AGEN1/TIME(2,16),TAUT(15),TAUW(15),TZERO,TLIFT,TTILT,TMINH,
     *DTZ,TQ,TL,XMAUG,TNE(6,15),ENGDAT(8,15),S(15),WD(15),WJET(15),
     *HEAD,PRINT(15),STEP(15),HNOM(15),HMAX(15),PROP(6),TBEGR,TENDR,
     *CHRDOT,CHIRO,FAZ,BO(3,2),CBAXIL,TCHIR,VRCUT,FPRFAC,CHVEL

      COMMON/AGEN2/AA,SA,CA,ALF,ALFY,CHIP,SCHIP,CCHIP,CHIY,SCHIY,CCHIY,
     *CHIR,SCHIR,CCHIR,STH,CTH,STHL,CTHL,DPHIZ,RTHE,R,VR,XM,SW,SU,SV,Q,
     *VIV(25),DVARS(13),DELXDW(15,3),DELXDR(7,5),DELXD(15,7),WZERO,ALT,
     *XMACH,QDOT,FAA,FAN,A(3,3),CASE,CT2,UMF,A12W,A22W,A32W,XJEXT,BEU,
     *BYL,BEL,BHMAX(15),BHMN,BSTEP(15),HNMB,HMXB,TPOLY,XMIAD

      COMMON/AGEN3/ITHR,IWD,IXR,JUMP,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISION,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)

      COMMON/AGEN4/QALPHA,QBETA,QCN,STH1,CTH1,XJX(3),XJV(3)

      COMMON/BAKE/WISS(20,20),YLF(7,20)

      COMMON/BAKINT/BBANK(2),TBK,TBKD,YL(7,20),YLD(7,20),BSAVE(2700),
     *NBTRG1,TBV1,NBTRG2,TBV2,NBTRG3,TBV3,NBTRG4,TBV4,NBTRG5,TBV5,TEND,
     *DTB4,DU3,D23,D43,WISSD(20,20)

      COMMON/BGEN3/NOSS,JTB,NENT,NACT,NOM1,NP(7)

      COMMON/BLAMB/YLBT(15,20),YLBW(15,20),GAPHI(20),PROD(2,20)

      COMMON/BODYB/TBDWT(15),TXCG(15),TYCG(15),XLEN,XREF,YREF,MBS1,MBS2,
     *MBS3,MBS4,MBS5,MBS6,MBS7,MBS8,MBS9,MBS10

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM

      COMMON/CONTRO/TTBL(30),CPTBL(30),CYTBL(30),TOBL(210),
     *CPOTBL(210),CYOTBL(210)

      COMMON/DELXH/DELXH(5)

      COMMON/DRNGDT/DRNGDT(2)

      COMMON/ENPUT/PSIRST(6,5),HEADER(20),NVRST(5),KCDRES(6,5),LAST
```



```
      COMMON/FORINT/HBANK(2),T,TT,W,U,V,X,Y,Z,XMI,ZAP(18),DVAR(25),FSAVE
     *(625),NTRG1,TV1,NTRG2,TV2,NTRG3,TV3,NTRG4,TV4,NTRG5,TV5,NTRG6,TV6,
     *NTRG7,TV7,NTRG8,TV8,NTRG9,TV9,NTRG10,TV10,NTRG11,TV11,NTRG12,TV12,
     *NTRG13,TV13,KIND,KRDER,EU,EL,AYL,HMX,HMN,HNM

      COMMON/HEAT/MAXHAT

      COMMON/IPR/IPR,WPMX,ILAST,KDT(15),PRK4,PRTOT,iprop(6),prk5(5)

      COMMON/MPS/TIMMPS(15),FTBL(15),THROT(15),GLIM(15),TAUALT(5),
     *TAUTBL(5),MPS1,MPS2,TGLIM(10,15),GLIMIT,tau_const(10,5),jthrot(15)

      common/mult/nchiot(6),tbias(6),nstage,nstg(15),wf(5),ibranch

      COMMON/NOMOR/NOMOR

      COMMON/REST/JSTR,NCNRS(5),NTCN,TR(5),LSB,NF8,NOPAR,WIBT(15)

      COMMON/TABLK/mcon(7)

      COMMON/VAERO/VAEROD(10,19),M18,M19,MBS18,MBS19

      COMMON/YLI/YLINT(7,20),PHITE(20),PSIREQ(20),KCDPHI(20),PSTR(20)

C     INITIALIZE BACKWARD RUN                                           BAKR  35

      EXTERNAL BDR1I,BDRI,BWDEV,BSAVEG,BKEND,BGLIM

      MBS1=1
      MBS2=0
      MBS3=1
      MBS4=0
      MBS7=1
      MBS8=0
      MBS18=1
      MBS19=0
      LLDSOL=5                                                          BAKR  39

      IF(KINDB.NE.2)LLDSOL=3*(KRDERB+2)+4                               BAKR  40

      TEND=T                                                            BAKR  41
      TBK=0.                                                            BAKR  42
      TBKD=0.                                                           BAKR  43
      NOMOR=0                                                           BAKR  44
      KAT=0                                                             BAKR  45
      NCTN=1

      DO 30 I=1,5

      IF(NVRST(I).NE.0) NCTN=NCTN+1
      TR(I)=0.

   30 CONTINUE

      KCYTAB=KCYTAB+1                                                   BAKR  47
      JTB=1                                                             BAKR  48
```

```
      ITHR=NVNT+1                                                       BAKR  49
      IWD=NWVNT+1                                                       BAKR  50

      NENT=IXR                                                          BAKR  51
      IF(IXR.EQ.0) NENT=100

      MBS5=NENT
      MBS6=0
      XMAUG=XMAUG+PRTOT                                                 BAKR  54

C     ZERO OUT LAMBDAS                                                  BAKR  55

      do i=1,7
         do j=1,20
            yl(i,j)=0.
         enddo
      enddo

      do j=1,7
         do i=1,noss
            YL(J,I)=YLINT(J,I)
         enddo
      enddo

      do i=1,20
         do j=1,i
            WISS(I,J)=0.
         enddo
      enddo

      do i=1,15
         do j=1,20
            YLBT(I,J)=0.
         enddo
      enddo

C*********************************************************************

c       Backward integration triggers

c       number  description                     routine parameter
c                                               called
c       *****   **********************          ******* ********
c       1       integration termination         bkend   time
c       2       thrust events                   bthrev  time
c       3       weight drop events              bwdev   time
c       4       save impulse responce functions bsaveg  time
c       5       acceleration limit              bglim   time

c*********************************************************************

C     STOP TRIGGER                                                      BAKR  67

c*********************************************************************

      NBTRG1=1
```



```
      TBV1=TEND-TQ                                                      BAKR  69

C*********************************************************************

C*2*  THRUST EVENT TRIGGER  (BTHREV)                                    BAKR  71

c*********************************************************************

      NBTRG2=2                                                          BAKR  72

      TBV2=0.

C*********************************************************************

C*3*  WEIGHT DROP EVENT TRIGGER  (BWDEV)                               BAKR  75

c*********************************************************************

      NBTRG3=2                                                          BAKR  76
      IF(NWVNT.EQ.0) NBTRG3=-2

      TBV3=TEND-TIME(2,NWVNT)                                           BAKR  77

c*********************************************************************

C*4*  SAVE IMPULSE RESPONSE FUNCTIONS  (BSAVEG)                        BAKR  79

c*********************************************************************

      NBTRG4=-1

      TBV4=0.                                                           BAKR  81

C*********************************************************************

C*5*  G LIMIT

c*********************************************************************

      NBTRG5=-2                                                         BAKR  84

      tbv5=0.

      if(jthrot(ithr-1).gt.0) TBV5=TEND-TGLIM(jthrot(ithr-1),ithr-1)

C*********************************************************************

C     INITIALIZE CHI SPACING                                            BAKR  85
C     INITIALIZE THRUST,STEP,HNOM,HMAX,MDOT,ETC.                        BAKR  86

c*********************************************************************

      CALL BTHREV                                                       BAKR  87

c*********************************************************************
```

```
C       Perform Backward integration
c*********************************************************************
    3 NACT=7*NOSS                                                    BAKR  89
      NOMOR=0
      CALL DESOLV(140,NACT,BDR1I,BDRI,0,6,KERR,HNMB,BBANK,BSAVE,
     *KINDB,LLDSOL,BEU,BEL,HMXB,BHMN,BYL,5,
     *NBTRG4,BSAVEG,TBK,TBV4,NBTRG3,BWDEV,TBK,TBV3,
     *NBTRG2,BTHREV,TBK,TBV2,NBTRG5,BGLIM,TBK,TBV5,
     *NBTRG1,BKEND,TBK,TBV1,  0,0,0,0,  0,0,0,0,  0,0,0,0,
     *0,0,0,0,  0,0,0,0,  0,0,0,0)
      IF(KERR.NE.0) then
         WRITE(6,9310)
 9310    FORMAT(//1X,24HINTEGRATION ERROR.DUMPED)
         STOP
      endif
      IF(NOMOR.ne.1) then
c*********************************************************************
C****  NOMOR IS SET TO 1 IN BTHREV BEFORE CALL TO RTMRK IF RESTART     BAKR  98
c*********************************************************************
         IF(ITHR.EQ.MINH.AND.ITHR-1.EQ.NVRST(1)) NOMOR=2
         IF(NOMOR.EQ.0) RETURN
         IF(NOMOR.EQ.2) ITHR=ITHR-1
      endif
      NCTN=NCTN-1
      TR(NCTN)=TEND-TBK
      K=NOSS+1                                                       BAKR 102
      NOSS=NOSS+NCNRS(NCTN)
      do j=1,7
         do i=K,NOSS
            YL(J,I)=YLINT(J,I)
         enddo
      enddo
      DO 6 I=K,NOSS                                                  BAKR 107
      YLBT(ITHR,I)=0.                                                BAKR 108
```



```
      L=I-K+1
      LL=KCDRES(L,NCTN)
      IF(LL.GE.18.AND.LL.LE.20) YLBT(ITHR,I)=DRNGDT(2)
      DO 6 J=1,7                                                     BAKR 109
    6 YLBT(ITHR,I)=YLBT(ITHR,I)+DELXDR(J,NCTN)*YL(J,I)
      IF(MAXHAT.NE.0.AND.ABS(DELXH(NCTN)).GT.0.) YLBT(ITHR,MAXHAT)=
     * YLBT(ITHR,MAXHAT)+DELXH(NCTN)
c       write(31,*) ' from bakrn'
c       write(31,*) ' ylbt(',ithr,')='
c       write(31,'(1x,9e14.7)') (ylbt(ithr,i),i=1,noss)
      IF(NOMOR.ne.2) GO TO 3
      CALL BKEND
      RETURN
      END                                                            BAKR 115
```

3.24 Subroutine BDR1I

3.24.1 Purpose

This subroutine computes the first partial derivatives of the equations of motion with respect to the state variables and the first and second partial derivatives with respect to the attitude angles.

3.24.2 Variable Listing

3.24.3 Subroutines Called:

Name	Function
ACNTRO	Interpolates pitch and yaw attitude angles during min-H phase of simulation
AGEO	Calculates gravitational accelerations and partial derivatives with respect to the state variables
AMULG	Interpolates two dependent variables based on linear technique
AMULG7	Interpolates seven dependent variables based on linear technique
ATMOSPHERE	Calculates atmospheric parameters using either Patrick AFB or Vandenburg AFB models
MASSPRO	Calculates dynamic mass properties
RRASPL	Interpolates atmospheric parameters from Range Reference model using spline method
SPLINE	Interpolates data based on 3rd order spline technique. Partial derivatives are also calculated by the spline routine

3.24.4 Calling Subroutines:

Name	Function
DPIR	Integration routine

3.24.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a(3,3)	Tranformation matrix from equatorial to plumbline coordinate systems		aforun	input	Global	agen2
a12w	Plumbline X component of earth spin axis		aforun	input	Global	agen2
a22w	Plumbline Y component of earth spin axis		aforun	input	Global	agen2
a32w	Plumbline Z component of earth spin axis		aforun	input	Global	agen2
aa	Not used				Local	
aaero	Spline coefficients for aerodynamic coefficient interpolation				Local	
acacf	Continuous flow coefficient for axial force				Local	
acamf	Molecular flow coefficient for axial force				Local	
acg	Spline coefficients for center of gravity interpolation				Local	
acmcf	Continuous flow coefficient for pitching moment				Local	
acmmf	Molecular flow coefficient for pitching moment				Local	
acncf	Continuous flow coefficient for normal force				Local	
acnmf	Molecular flow coefficient for normal force				Local	
aero	Interpolated aerodynamic coefficients				Local	
aerob(50,6,3	Base force tables		ainit	input	Global	aerodi
aerodd	Derivatives of aerodynamic coefficients with respect to mach number				Local	
af	Spline coefficients for incremental base force interpolation				Local	
afab	Spline coefficients for base force interpolation				Local	
alfy	Sideslip angle	rad	ader	output input	Global	agen2
alp	Angle of attack	deg			Local	
alpha	Angle of attack	deg			Local	
alp_new	Angle of attach based on Q alpha	rad	bdr1i	output	Global	q_partials
alt	Altitude	m	bdr1i	output	Global	agen2

Variable	Variable Definition	Units	Source	I/O input	Status	Common Block
altls	Altitude of launch site	m	ainit	input	Global	alat
alttbl(100)	Altitude table for wind tables	m	ainit	input	Global	wind
amx	Aerodynamic moment about vehicle yaw axis	nt-m	bdr1i	output	Global	agen5
amy	Aerodynamic moment about vehicle roll axis	nt-m	bdr1i	output	Global	agen5
amz	Aerodynamic moment about vehicle pitch axis	nt-m	ader	output	Global	agen5
amz_new	Aerodynamic pitching moment (new value based on Qalpha orientation)	nt-m			Local	
aw	Spline coefficients for wind parameter interpolation				Local	
azw	Wind azimuth	rad	bdr1i	output	Global	agen5
baero	Spline coefficients for aerodynamic coefficient interpolation				Local	
bb	Not used				Local	
bcacf	Continuous flow coefficient for axial force				Local	
bcamf	Molecular flow coefficient for axial force				Local	
bcg	Spline coefficients for center of gravity interpolation				Local	
bcmcf	Continuous flow coefficient for pitching moment				Local	
bcmmf	Molecular flow coefficient for pitching moment				Local	
bcncf	Continuous flow coefficient for normal force				Local	
bcnmf	Molecular flow coefficient for normal force				Local	
bf	Spline coefficients for incremental base force interpolation				Local	
bfab	Spline coefficients for base force interpolation				Local	
bomdot	Booster mass flowrate	lbs/sec			Local	
bw	Spline coefficients for wind parameter interpolation				Local	
ca	Axial force coefficient				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
caalp	Slope of axial force coefficient with respect to angle of attack				Local	
cacf	Axial force continuum flow coefficient				Local	
cadele	Coefficient of axial force coefficient with respect to elevon deflection (currently set to 0)				Local	
caero	Spline coefficients for aerodynamic coefficient interpolation				Local	
calcpar	Logical flag to indicate calculation of partials				Local	
calp	Cosine of angle of attack				Local	
calp_new	Base force tables				Local	
camf	Axial force molecular flow coefficient				Local	
cao	Axial force coefficient at zero angle of attack				Local	
cazw	Cosine of the wind azimuth angle				Local	
ca_new	Axial force coefficient based new value of alpha				Local	
cbaxil	Constant value of base axial force		ainit	input	Global	agen1
cc	Not used				Local	
ccacf	Continuous flow coefficient for axial force				Local	
ccamf	Molecular flow coefficient for axial force				Local	
ccg	Spline coefficients for center of gravity interpolation				Local	
cchip	Cosine of pitch attitude angle		ader1	output input	Global	agen2
cchir	Cosine of roll attitude angle		ader1	output input	Global	agen2
cchiy	Cosine of yaw attitude angle		ader1	output input	Global	agen2
ccmcf	Continuous flow coefficient for pitching moment				Local	
ccmmf	Molecular flow coefficient for pitching moment				Local	
ccncf	Continuous flow coefficient for normal force				Local	
ccnmf	Molecular flow coefficient for normal force				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
cf	Spline coefficients for incremental base force interpolation				Local	
cfab	Spline coefficients for base force interpolation				Local	
cg	Interpolated values of center of gravity components	m			Local	
cgbl	Center of gravity tables	m			Local	
chip	Pitch attitude angle	rad	bdr1i	input	Global	agen2
chir	Roll attitude angle	rad	ader1	input	Global	agen2
chiy	Yaw attitude angle	rad	bdr1i	input	Global	agen2
clbeta	Rolling moment slope coefficient	/deg			Local	
clo	Rolling moment coefficient at zero sideslip angle				Local	
cmalp	Pitching moment slope coefficient	/deg			Local	
cmcf	Pitching moment continuum flow coefficient				Local	
cmdele	Coefficient of pitching moment coefficient with respect to elevon deflection (currently set to 0)				Local	
cmmf	Pitching moment molecular flow coefficient				Local	
cmo	Pitching moment coefficient at zero alpha				Local	
cn	Slope of normal force coefficient with respect to angle of attack				Local	
cnalp	Slope of normal force coefficient with respect to angle of attack	/deg			Local	
cnbeta	Yawing moment slope coefficient	/deg			Local	
cnbo	Yawing moment coefficient at zero sideslip angle				Local	
cncf	Normal force continuum flow coefficient				Local	
cndele	Coefficient of normal force coefficient with respect to elevon deflection (currently set to 0)				Local	
cnmf	Normal force molecular flow coefficient				Local	
cno	Normal force coefficient at zero angle of attack				Local	
cn_new	Normal force coefficent based on angle of attack				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
colda	Cosine of previous value of angle of attack				Local	
covl(7)	Lagrangian multipliers				Local	
cp_input	Logical indicating center of pressure input		ainit	input	Global	cp_input
crho	Cosine of angle between aerodynamic reference arm and longitudinal axis of vehicle				Local	
cs	Side force coefficient				Local	
ct2	Squared value of cth		bdr1i	output input	Global	agen2
cth	Cosine of colatitude		bdr1i	output input	Global	agen2
cw	Spline coefficients for wind parameter interpolation				Local	
cxx	Temporary variable used to expedient the calculation of the moment balance gimbal angles				Local	
cxy	Temporary variable used to expedient the calculation of the moment balance gimbal angles				Local	
cxz	Temporary variable used to expedient the calculation of the moment balance gimbal angles				Local	
cybeta	Slope of side force coefficient with respect to sideslip angle	/deg			Local	
cyx	Temporary variable used to expedient the calculation of the moment balance gimbal angles				Local	
cyy	Temporary variable used to expedient the calculation of the moment balance gimbal angles				Local	
cyz	Temporary variable used to expedient the calculation of the moment balance gimbal angles				Local	
czx	Temporary variable used to expedient the calculation of the moment balance gimbal angles				Local	
czy	Temporary variable used to expedient the calculation of the moment balance gimbal angles				Local	
daltb	Difference between base force sea level	m			Local	

Variable	Variable Definition altitude and reference altitude	Units	Source	I/O	Status	Common Block
dcax	Incremental value of aerodynamic axial force coefficient				Local	
dcmz	Incremental value of pitching moment coefficient				Local	
dcnp	Incremental value of aerodynamic normal force coefficient				Local	
ddfab	Derivative of incremental base force with respect to altitude	nt/m			Local	
delte	Effective elevon deflection angle (currently set to 0)				Local	
delwt	Incremental weight overboard	kg			Local	
dfab	Incremental base force	nt			Local	
dmuh	Partial derivative of coefficient of viscosity with respect to altitude				Local	
dpdh	Partial derivative of atmospheric pressure with respect to altitude				Local	
drdh	Partial derivative of atmospheric pressure with respect to altitude				Local	
drthe	Partial derivative of altitude with respect to colatitude				Local	
dsdh	Partial derivative of speed of sound with respect to altitude				Local	
dx	Normal component of aerodynamic moment arm				Local	
dxdy	dx/dy				Local	
dxpr	Normal component of thrust moment arm divided by reference length	m			Local	
dy	Axial component of aerodynamic moment arm	m			Local	
dyi	1/dy	/ m			Local	
dypr	Axial component of thrust moment arm divided by reference length	m			Local	
dz	Lateral component of aerodynamic moment arm	m			Local	
dzi	1/dz	/ m			Local	
e1	X component of local east direction cosine				Local	
e2	Y component of local east direction cosine				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
e3	Z component of local east direction cosine				Local	
engdat(8,15)	Engine data matrix		ainit	input	Global	agen1
engines(15,1	This array indicates which engine from the ENGDAT array is being used for a certain thrust event. (0- not being used, 1-being used)		ainit	input	Global	engines
eventnum	Number of event				Local	
exx	Partial derivative of yawing moment with respect to mach number				Local	
exz	Temporary variable				Local	
eyx	Partial derivative of rolling moment with respect to mach number				Local	
eyz	Temporary variable				Local	
ezx	Partial derivative of pitching moment with respect to mach number				Local	
ezy	Temporary variable				Local	
ezyy	Temporary variable				Local	
fab	Base axial force	nt			Local	
fabdq	Base axial force divided by dynamic pressure				Local	
fabdsq	Base axial force divided by product of area and Q				Local	
fabt	Base force values	nt			Local	
flat	Earth flattening coefficient		ainit	input	Global	const
flb	Rolling moment due to base force effects (currently set to 0)				Local	
fmb	Base force pitch moment				Local	
fnb	Base normal force divided by dynamic pressure	newton			Local	
fnbdq	Base normal force divided by dynamic pressure				Local	
fnbdsq	Base normal force divided by the product of area and Q				Local	
fnmb	Yawing moment due to base force effects (currently set to 0)				Local	
foms(15)	Thrust table for orbit maneuver system	lbs	ainit	input	Global	omsrcs
frcs(15)	Thrust table for rocket control system	lbs	ainit	input	Global	omsrcs
fttm	Feet to meter conversion				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
fyb	Side force caused by base force effects (currently set to 0)				Local	
f_alp	Temporary variable				Local	
g11	Temporary variable in gravitational equations		ageo	input	Global	grav
g22	Temporary variable in gravitational partial derivative equations		ageo	input	Global	grav
g23	Temporary variable in gravitational partial derivative equations		ageo	input	Global	grav
g33	Temporary variable in gravitational partial derivative equations		ageo	input	Global	grav
glim(15)	Acceleration limit	g's	ainit	input	Global	mps
gm	Required level of thrust to provide constant acceleration	newton			Local	
gnu(20)	Partial derivative of payoff with respect to constraints		anewch	input	Global	gnu
gxx	Partial derivative of x component of gravitational vector with respect to plumbline x position coordinate				Local	
gyx	Partial derivative of y component of gravitational vector with respect to plumbline x position coordinate				Local	
gyy	Partial derivative of y component of gravitational vector with respect to plumbline y position coordinate				Local	
gzero	Gravitational constant (9.80665 m/sec)	m/sec^	ainit	input	Global	const
gzx	Partial derivative of z component of gravitational vector with respect to plumbline x position coordinate				Local	
gzy	Partial derivative of z component of gravitational vector with respect to plumbline y position coordinate				Local	
gzz	Partial derivative of z component of gravitational vector with respect to plumbline z position coordinate				Local	
hatmos	Altitude used to interpolate for atmospheric parameters				Local	
hbias	Altitude bias	m	ainit	input	Global	ardc
hgt	Altitude used to interpolate for base force effects	m			Local	
hrate	Temporary variable used in heating				Local	

Variable	Variable Definition equation	Units	Source	I/O	Status	Common Block
hrate1	Temporary variable used in heating equation				Local	
hrate2	Temporary variable used in stagnation heating equation				Local	
hrate3	Temporary variable used in calculation of heating rate partial derivative				Local	
hrate4	Temporary variable used in calculation of heating rate partial derivative				Local	
hrate5	Temporary variable used in calculation of heating rate partial derivative				Local	
iatmos	Atmospheric routine indicator		ainit	input	Global	ardc
ithr	Thrust event index number		athrev	input output	Global	agen3
itol(15)	Flag indicating thrust event use of delta aerodynamic coefficients and base force		ainit	input	Global	delfab
jthrot(15)	Index indicating number of times the acceleration limit has been reached within a given thrust event		atilt(gli	input	Global	mps
kcytab	Attitude control table index		bthrev	input	Global	agen3
kk	Index to indicate number of tables to interpolate				Local	
km1	index				Local	
ktbl	Parameter for Tabs common array				Local	
kwta(7)	Flag indicating pitch only (1) or pitch and yaw (2) attitude control		ainit	input	Global	agen3
latcg	Lateral center of gravity component	m			Local	
latcgp	Partial derivative of lateral CG with respect to mass				Local	
loncg	Total longitudinal center of gravity	m			Local	
loncgp	Partial derivative of total longitudinal CG with respect to mass				Local	
lstge(15)	Aerodynamic coefficient flag to distinguish stages		ainit	input	Global	agen3
m1	Index used for linear interpolation for thrust tail option				Local	
maxhat	Index used to indicate the number of the constraint used for the maximum heating constraint			output	Global	heat
max_qalp	Base force tables				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
mbs19	Previous index for viscous drag interpolation in backwards integration			output	Global	vaero
mbs2	Previous index for aerodynamic interpolation in backwards integration			output	Global	bodyb
mbs31	Previous index for incremental base force interpolation			input	Global	delmbs
mbs4	Previous index for base force interpolation in backwards integration			output	Global	bodyb
mbs5	Index for state interpolation		bdr1i	input	Global	bodyb
mbs6	Not used			output	Global	bodyb
mmax	Maximum index for mps engines				Local	
mmin	Index of first active MPS engine				Local	
mnmdot	Main engine flowrate	lbs/sec			Local	
moboos	Booster mass	lbs			Local	
momain	Main engine mass				Local	
mombal	Flag to specify the use of moment balance equations		ainit	input	Global	mombal
mpflag	Alternate mass properties flag				Global	mpropin
mpsflg(15)	Flag indicating the type of MPS thrust simulation		ainit	output	Global	agen3
mpsinp(15)	Array used in conjunction with mpsflg=6 to denote which engines (with respect to the ENGDAT array) will use the throttle table		ainit	input	Global	mps1
n	Index				Local	
nbtrg5	Acceleration limit trigger for backwards integration		bakrn	input	Global	bakint
nchiot(6)	Optimization flag used with branch trajectory option		ainit	input	Global	mult
ncnrs(5)	Number of constraints at restart		ainit	input	Global	rest
nent	Number of points in state variable tables		bakrn	input	Global	bgen3
nobase	Flag indicating that no base force will be calculated after the thrust event specified by nobase		ainit	input	Global	agen3
noevnt(5)	Number of thrust events per stage		ainit	output	Global	agen3
norcg	Total normal component of center of gravity	in			Local	
norcgp	Derivative of normal CG with respect to	in/lbs			Local	

Variable	Variable Definition mass	Units	Source	I/O	Status	Common Block
noss	Number of constraints		bakrn	input	Global	bgen3
nossm1	Index (noss - 1)				Local	
nowind	Logical flag indicating no wind calculation				Local	
nstage	Index number of current stage		athrev	input	Global	mult
nsyst(15)	Index array which specifies the type of engines that will be used for each thrust event		ainit	input	Global	agen3
nvrst(5)	Array specifying the thrust events where intermediate constraints will be imposed		ainit	output	Global	enput
nwind	Number of tabular values in wind parameter tables		ainit	input	Global	wind
olda	Past value of angle of attack				Local	
olow	Orbiter liftoff weight	kg	athrev	input	Global	olow
omass	One divided by vehicle mass				Local	
omx	X component of earth spin vector (same as a12w)				Local	
omy	Y component of earth spin vector				Local	
omz	Z component of earth spin vector (same as a12w)				Local	
othr	1./thrust				Local	
p1	Temporary variable				Local	
p11	Temporary variable				Local	
p12	Temporary variable				Local	
p2	Temporary variable				Local	
p3	Temporary variable				Local	
p4	Temporary variable				Local	
p5	Temporary variable				Local	
p6	Temporary variable				Local	
p7	Temporary variable				Local	
p8	Temporary variable				Local	
p9	Temporary variable				Local	
paacp	Partial derivative of total normal force component (in body system) with respect to the pitch attitude angle				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
paacpp	Second partial derivative of total normal force component (in body system) with respect to the pitch attitude angle				Local	
paacpy	Second partial derivative of total normal force component (in body system) with respect to the pitch and yaw attitude angles				Local	
paacy	Partial derivative of total normal force component (in body system) with respect to the yaw attitude angle				Local	
paacyy	Second partial derivative of total normal force component (in body system) with respect to the yaw attitude angle				Local	
pabcp	Partial derivative of total axial force component (in body system) with respect to the pitch attitude angle				Local	
pabcpp	Second partial derivative of total axial force component (in body system) with respect to the pitch attitude angle				Local	
pabcpy	Second partial derivative of total axial force component (in body system) with respect to the pitch and yaw attitude angles				Local	
pabcy	Partial derivative of total axial force component (in body system) with respect to the yaw attitude angle				Local	
pabcyy	Second partial derivative of total axial force component (in body system) with respect to the yaw attitude angle				Local	
paccp	Partial derivative of total lateral force component (in body system) with respect to the pitch attitude angle				Local	
paccpp	Second partial derivative of total lateral force component (in body system) with respect to the pitch attitude angle				Local	
paccpy	Second partial derivative of total lateral force component (in body system) with respect to the pitch and yaw attitude angles				Local	
paccy	Partial derivative of total lateral force component (in body system) with respect to the yaw attitude angle				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
paccyy	Second partial derivative of total lateral force component (in body system) with respect to the yaw attitude angle				Local	
pacp	Second partial derivative of angle of attack with respect to the pitch and yaw attitude angles				Local	
pacpcp	Second partial derivative ofangle of attack with respect to pitch attitude				Local	
pacpcy	Second partial derivative of angle of attack with respect to the pitch and yaw attitude angles				Local	
pacy	Partial derivative of angle of attack with respect to yaw attitude angle				Local	
pacycy	Second partial derivative of angle of attack with respect to the yaw attitude angle				Local	
pam	Atmospheric pressure				Local	
pamz_alp	Partial derivative of amz with respect to angle of attack				Local	
pat(14)	Input/output array for atmospheric parameters			output input	Global	ardc
pau	Partial derivative ofangle of attack with respect to Plumbline y velocity component				Local	
pav	Partial derivative ofangle of attack with respect to Plumbline z velocity component				Local	
paw	Partial derivative ofangle of attack with respect to Plumbline x velocity component				Local	
pax	Partial derivative of angle of attack with respect to Plumbline x position component				Local	
pay	Partial derivative of angle of attack with respect to Plumbline y position component				Local	
paz	Partial derivative of the wind azimuth angle with respect to altitude				Local	
pazwdh	Partial derivative of the wind azimuth angle with respect to altitude				Local	
pbcp	Second partial derivative of sideslip angle with respect to pitch attitude				Local	

Variable	Variable Definition angle	Units	Source	I/O	Status	Common Block
pbcpcp	Second partial derivative of sideslip angle with respect to pitch attitude angle				Local	
pbcpcy	Second partial derivative of sideslip angle with respect to pitch and yaw attitude				Local	
pbcy	Partial derivative of sideslip angle with respect to yaw attitude				Local	
pbcycy	Second partial derivative of sideslip angle with respect to yaw attitude				Local	
pbu	Partial derivative of sideslip angle with respect toPlumbline y velocity component				Local	
pbv	Partial derivative of sideslip angle with respect toPlumbline z velocity component				Local	
pbw	Partial derivative of sideslip angle with respect toPlumbline x velocity component				Local	
pbx	Partial derivative of sideslip angle with respect toPlumbline x position component				Local	
pby	Partial derivative of sideslip angle with respect toPlumbline y position component				Local	
pbz	Partial derivative of sideslip angle with respect toPlumbline x position component				Local	
pcaam	Partial derivative of axial force slope coefficients with respect to mach number				Local	
pcacfu	Partial derivative of "cacf" with respect toPlumbline y velocity component				Local	
pcacfv	Partial derivative of "cacf" with respect toPlumbline z velocity component				Local	
pcacfw	Partial derivative of "cacf" with respect toPlumbline x velocity component				Local	
pcacfx	Partial derivative of "cacf" with respect toPlumbline x position				Local	

Variable	Variable Definition component	Units	Source	I/O	Status	Common Block
pcacfy	Partial derivative of "cacf" with respect toPlumbline y position component				Local	
pcacfz	Partial derivative of "cacf" with respect toPlumbline z position component				Local	
pcadm	Partial derivative of base axial force with respect to mach number (currently set to 0)				Local	
pcamfu	Partial derivative of "camf" with respect toPlumbline y velocity component				Local	
pcamfv	Partial derivative of "camf" with respect toPlumbline z velocity component				Local	
pcamfw	Partial derivative of "camf" with respect toPlumbline x velocity component				Local	
pcamfx	Partial derivative of "camf" with respect toPlumbline x position component				Local	
pcamfy	Partial derivative of "camf" with respect toPlumbline y position component				Local	
pcamfz	Partial derivative of "camf" with respect toPlumbline z position component				Local	
pcaom	Partial derivative of zero alpha axial force coefficients with respect to mach number				Local	
pcau	Partial derivative of axial force coefficient with respect toPlumbline y velocity component				Local	
pcav	Partial derivative of axial force coefficient with respect toPlumbline z velocity component				Local	
pcaw	Partial derivative of axial force coefficient with respect toPlumbline x velocity component				Local	
pcax	Partial derivative of axial force coefficient with respect toPlumbline x position component				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
pcay	Partial derivative of axial force coefficient with respect toPlumbline y position component				Local	
pcaz	Partial derivative of axial force coefficient with respect toPlumbline z position component				Local	
pcg	Derivative of center of gravity components with respect to mass				Local	
pclbm	Partial derivative of rolling moment slope coefficients with respect to mach number				Local	
pclom	Partial derivative of zero beta rolling moment coefficients with respect to mach number				Local	
pcmam	Partial derivative of pitching moment slope coefficients with respect to mach number				Local	
pcmcfu	Partial derivative of "cmcf" with respect toPlumbline y velocity component				Local	
pcmcfv	Partial derivative of "cmcf" with respect toPlumbline z velocity component				Local	
pcmcfw	Partial derivative of "cmcf" with respect toPlumbline x velocity component				Local	
pcmcfx	Partial derivative of "cmcf" with respect toPlumbline x position component				Local	
pcmcfy	Partial derivative of "cmcf" with respect toPlumbline y position component				Local	
pcmcfz	Partial derivative of "cmcf" with respect toPlumbline z position component				Local	
pcmdm	Partial derivative of elevon pitching moment with respect to mach number (currently set to 0)				Local	
pcmmfu	Partial derivative of "cmmf" with respect toPlumbline y velocity component				Local	
pcmmfv	Partial derivative of "cmmf" with respect toPlumbline z velocity component				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
pcmmfw	Partial derivative of "cmmf" with respect toPlumbline x velocity component				Local	
pcmmfx	Partial derivative of "cmmf" with respect toPlumbline x position component				Local	
pcmmfy	Partial derivative of "cmmf" with respect toPlumbline y position component				Local	
pcmmfz	Partial derivative of "cmmf" with respect toPlumbline z position component				Local	
pcmom	Partial derivative of zero alpha pitching moment coefficients with respect to mach number				Local	
pcmu	Partial derivative of pitching moment coefficient with respect toPlumbline y velocity component				Local	
pcmv	Partial derivative of pitching moment coefficient with respect toPlumbline z velocity component				Local	
pcmw	Partial derivative of pitching moment coefficient with respect toPlumbline x velocity component				Local	
pcmx	Partial derivative of pitching moment coefficient with respect toPlumbline x position component				Local	
pcmy	Partial derivative of pitching moment coefficient with respect toPlumbline y position component				Local	
pcmz	Partial derivative of pitching moment coefficient with respect toPlumbline z position component				Local	
pcnam	Partial derivative of normal force slope coefficients with respect to mach number				Local	
pcnbm	Partial derivative of yawing moment slope coefficients with respect to mach number				Local	
pcnbom	Partial derivative of zero beta yawing moment coefficients with respect to mach number				Local	
pcncfu	Partial derivative of "cncf" with respect toPlumbline y velocity				Local	

Variable	Variable Definition component	Units	Source	I/O	Status	Common Block
pcncfv	Partial derivative of "cncf" with respect toPlumbline z velocity component				Local	
pcncfw	Partial derivative of "cncf" with respect toPlumbline x velocity component				Local	
pcncfx	Partial derivative of "cncf" with respect toPlumbline x position component				Local	
pcncfy	Partial derivative of "cncf" with respect toPlumbline y position component				Local	
pcncfz	Partial derivative of "cncf" with respect toPlumbline z position component				Local	
pcndm	Partial derivative of elevon normal force with respect to mach number (currently set to 0)				Local	
pcnmfu	Partial derivative of "cnmf" with respect toPlumbline y velocity component				Local	
pcnmfv	Partial derivative of "cnmf" with respect toPlumbline z velocity component				Local	
pcnmfw	Partial derivative of "cnmf" with respect toPlumbline x velocity component				Local	
pcnmfx	Partial derivative of "cnmf" with respect toPlumbline x position component				Local	
pcnmfy	Partial derivative of "cnmf" with respect toPlumbline y position component				Local	
pcnmfz	Partial derivative of "cnmf" with respect toPlumbline z position component				Local	
pcnom	Partial derivative of zero alpha normal force coefficients with respect to mach number				Local	
pcnu	Partial derivative of normal force coefficient with respect toPlumbline y velocity component				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
pcnv	Partial derivative of normal force coefficient with respect toPlumbline z velocity component				Local	
pcnw	Partial derivative of normal force coefficient with respect toPlumbline x velocity component				Local	
pcnx	Partial derivative of normal force coefficient with respect toPlumbline x position component				Local	
pcny	Partial derivative of normal force coefficient with respect toPlumbline y position component				Local	
pcnz	Partial derivative of normal force coefficient with respect toPlumbline z position component				Local	
pcrhom	Partial derivative of "crho" with respect to vehicle mass				Local	
pcsu	Partial derivative of side force coefficient with respect toPlumbline y velocity component				Local	
pcsv	Partial derivative of side force coefficient with respect toPlumbline z velocity component				Local	
pcsw	Partial derivative of side force coefficient with respect toPlumbline x velocity component				Local	
pcsx	Partial derivative of side force coefficient with respect toPlumbline x position component				Local	
pcsy	Partial derivative of side force coefficient with respect toPlumbline y position component				Local	
pcsz	Partial derivative of side force coefficient with respect toPlumbline z position component				Local	
pcxxm	Partial derivative of "cxx" with respect to vehicle mass				Local	
pcxym	Partial derivative of "cxy" with respect to vehicle mass				Local	
pcxzm	Partial derivative of "cxz" with respect to vehicle mass				Local	
pcybm	Partial derivative of side force slope coefficients with respect to mach				Local	

Variable	Variable Definition number	Units	Source	I/O	Status	Common Block
pcyom	Partial derivative of zero beta side force coefficients with respect to mach number				Local	
pcyxm	Partial derivative of "cyx" with respect to vehicle mass				Local	
pcyym	Partial derivative of "cyy" with respect to vehicle mass				Local	
pcyzm	Partial derivative of "cyz" with respect to vehicle mass				Local	
pczxm	Partial derivative of "czx" with respect to vehicle mass				Local	
pczym	Partial derivative of "czy" with respect to vehicle mass				Local	
pdelm	Partial derivative of elevon deflection angle with respect to vehicle mass (currently set to 0)				Local	
pdxcp	Partial derivative of X acceleration component with respect to pitch attitude angle			output	Global	calif
pdxcpp	Second partial derivative of X acceleration component with respect to pitch attitude angle			output	Global	calif
pdxcpy	Second partial derivative of X acceleration component with respect to pitch and yaw attitude angles			output	Global	calif
pdxcy	Partial derivative of X acceleration component with respect to yaw attitude angle			output	Global	calif
pdxcyy	Second partial derivative of X acceleration component with respect to yaw attitude angle			output	Global	calif
pdycp	Partial derivative of Y acceleration component with respect to pitch attitude angle			output	Global	calif
pdycpp	Second partial derivative of Y acceleration component with respect to pitch attitude angle			output	Global	calif
pdycpy	Second partial derivative of Y acceleration component with respect to pitch and yaw attitude angles			output	Global	calif
pdycy	Partial derivative of Y acceleration component with respect to yaw attitude			output	Global	calif

Variable	Variable Definition angle	Units	Source	I/O	Status	Common Block
pdycyy	Second partial derivative of Y acceleration component with respect to yaw attitude angle			output	Global	calif
pdyprm	Partial derivative of longitudinal component of moment arm with respect to mass				Local	
pdzcp	Partial derivative of Z acceleration component with respect to pitch attitude angle			output	Global	calif
pdzcpp	Second partial derivative of Z acceleration component with respect to pitch attitude angle			output	Global	calif
pdzcpy	Second partial derivative of Z acceleration component with respect to pitch and yaw attitude angles			output	Global	calif
pdzcy	Partial derivative of Z acceleration component with respect to yaw attitude angle			output	Global	calif
pdzcyy	Partial derivative of Z acceleration component with respect to yaw attitude angle			output	Global	calif
pe1x	Partial derivative of the x component of the east direction cosine vector with respect to the Plumbline x position component				Local	
pe1y	Partial derivative of the x component of the east direction cosine vector with respect to the Plumbline y position component				Local	
pe1z	Partial derivative of the x component of the east direction cosine vector with respect to the Plumbline z position component				Local	
pe2x	Partial derivative of the y component of the east direction cosine vector with respect to the Plumbline x position component				Local	
pe2y	Partial derivative of the y component of the east direction cosine vector with respect to the Plumbline y position component				Local	
pe2z	Partial derivative of the y component of the east direction cosine vector with				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
	respect to the Plumbline z position component					
pe3x	Partial derivative of the z component of the east direction cosine vector with respect to the Plumbline x position component				Local	
pe3y	Partial derivative of the z component of the east direction cosine vector with respect to the Plumbline y position component				Local	
pe3z	Partial derivative of the z component of the east direction cosine vector with respect to the Plumbline z position component				Local	
pfabh	Partial derivative of axial base force with respect to altitude				Local	
pfabq	Partial derivative of axial base force with respect to dynamic pressure				Local	
pfkg	Conversion for pounds to kilograms	kg/lbs	ainit	input	Global	const
pflbh	Partial derivative of "flb" with respect to altitude (currently set to 0)				Local	
pfmbh	Partial derivative of base force pitching moment with respect to altitude				Local	
pfnbh	Partial derivative of normal base force with respect to altitude				Local	
pfnmbh	Partial derivative of "fnmb" with respect to altitude (currently set to 0)				Local	
pfnmbh	Partial derivative of "fnmb" with respect to altitude (currently set to 0)				Local	
pfxx	Partial derivative of normal component of fixed thrust vector with respect to plumbline x position component				Local	
pfxy	Partial derivative of normal component of fixed thrust vector with respect to plumbline y position component				Local	
pfxz	Partial derivative of normal component of fixed thrust vector with respect to plumbline z position component				Local	
pfybh	Partial derivative of "fyb" with respect to altitude (currently set to 0)				Local	
pfyx	Partial derivative of longitudinal component of fixed thrust vector with respect to plumbline x position				Local	

Variable	Variable Definition component	Units	Source	I/O	Status	Common Block
pfyy	Partial derivative of longitudinal component of fixed thrust vector with respect to plumbline y position component				Local	
pfyz	Partial derivative of longitudinal component of fixed thrust vector with respect to plumbline z position component				Local	
pfzx	Partial derivative of lateral component of thrust vector with respect to Plumbline x position component				Local	
pfzy	Partial derivative of lateral component of thrust vector with respect to Plumbline y position component				Local	
pfzz	Partial derivative of lateral component of thrust vector with respect to Plumbline z position component				Local	
pg1_alp	Temporary variable				Local	
phru	Partial derivative of heat rate with respect u velocity component			output	Global	phrs
phrv	Partial derivative of heat rate with respect v velocity component			output	Global	phrs
phrw	Partial derivative of heat rate with respect w velocity component			output	Global	phrs
phrx	Partial derivative of heat rate with respect x position component			output	Global	phrs
phry	Partial derivative of heat rate with respect y position component			output	Global	phrs
phrz	Partial derivative of heat rate with respect z position component			output	Global	phrs
phx	Partial derivative of vehicle altitude with respect to Plumbline x position component				Local	
phy	Partial derivative ofvehicle altitude with respect to Plumbline y position component				Local	
phz	Partial derivative ofvehicle altitude with respect to Plumbline z position component				Local	
pknudu	Partial derivative of Knudsen's number with respect toPlumbline y velocity component				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
pknudv	Partial derivative of Knudsen's number with respect toPlumbline z velocity component				Local	
pknudw	Partial derivative of Knudsen's number with respect toPlumbline x velocity component				Local	
pknudx	Partial derivative of Knudsen's number with respect toPlumbline x position component				Local	
pknudy	Partial derivative of Knudsen's number with respect toPlumbline y position component				Local	
pknudz	Partial derivative of Knudsen's number with respect toPlumbline z position component				Local	
pmaxcp	Partial derivative of aerodynamic yawing moment with respect to pitch attitude				Local	
pmaxcy	Partial derivative of aerodynamic yawing moment with respect to yaw attitude				Local	
pmaxm	Partial derivative of aerodynamic yawing moment with respect to vehicle mass				Local	
pmaxu	Partial derivative of aerodynamic yawing moment with respect to Y velocity component				Local	
pmaxv	Partial derivative of aerodynamic yawing moment with respect to Z velocity component				Local	
pmaxw	Partial derivative of aerodynamic yawing moment with respect to X velocity component				Local	
pmaxx	Partial derivative of aerodynamic yawing moment with respect to X position component				Local	
pmaxy	Partial derivative of aerodynamic yawing moment with respect to Y position component				Local	
pmaxz	Partial derivative of aerodynamic yawing moment with respect to Z position component				Local	
pmaycp	Partial derivative of aerodynamic rolling moment with respect to pitch				Local	

Variable	Variable Definition attitude	Units	Source	I/O	Status	Common Block
pmaycy	Partial derivative of aerodynamic rolling moment with respect to yaw attitude				Local	
pmaym	Partial derivative of aerodynamic rolling moment with respect to vehicle mass				Local	
pmayu	Partial derivative of aerodynamic rolling moment with respect to Y velocity component				Local	
pmayv	Partial derivative of aerodynamic rolling moment with respect to Z velocity component				Local	
pmayw	Partial derivative of aerodynamic rolling moment with respect to X velocity component				Local	
pmayx	Partial derivative of aerodynamic rolling moment with respect to X position component				Local	
pmayy	Partial derivative of aerodynamic rolling moment with respect to Y position component				Local	
pmayz	Partial derivative of aerodynamic rolling moment with respect to Z position component				Local	
pmazcp	Partial derivative of aerodynamic pitching moment with respect to pitch attitude				Local	
pmazcy	Partial derivative of aerodynamic pitching moment with respect to yaw attitude				Local	
pmazm	Partial derivative of aerodynamic pitching moment with respect to vehicle mass				Local	
pmazu	Partial derivative of aerodynamic pitching moment with respect to Plumbline Y velocity component				Local	
pmazv	Partial derivative of aerodynamic pitching moment with respect to Plumbline Z velocity component				Local	
pmazw	Partial derivative of aerodynamic pitching moment with respect to Plumbline X velocity component				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
pmazx	Partial derivative of aerodynamic pitching moment with respect to Plumbline X position component				Local	
pmazy	Partial derivative of aerodynamic pitching moment with respect to Plumbline Y position component				Local	
pmazz	Partial derivative of aerodynamic pitching moment with respect to Plumbline Z position component				Local	
pmu	Partial derivative of mach number with respect to Plumbline Y velocity component				Local	
pmv	Partial derivative of mach number with respect to Plumbline z velocity component				Local	
pmw	Partial derivative of mach number with respect to Plumbline x velocity component				Local	
pmx	Partial derivative of mach number with respect to Plumbline x position component				Local	
pmy	Partial derivative of mach number with respect to Plumbline y position component				Local	
pmz	Partial derivative of mach number with respect to Plumbline z position component				Local	
ponly	Logical pitch denoting only pitch attitude being simulated				Local	
pp1cpp	Second partial derivative of "tandp1" with respect to pitch attitude				Local	
pp1cpy	Second partial derivative of ty1 with respect to pitch and yaw attitude angle				Local	
pp1cyy	Second partial derivative of "tandp1" with respect to yaw attitude				Local	
pp2cpp	Second partial derivative of "tandp2" with respect to pitch attitude				Local	
pp2cpy	Second partial derivative of ty2 with respect to pitch and yaw attitude angles				Local	
pp2cyy	Second partial derivative of "tandp2" with respect to yaw attitude				Local	
pqrefh	Partial derivative of reference Q with respect to altitude				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
pqu	Partial derivative of dynamic pressure with respect to u velocity component			output	Global	q_partials
pqv	Partial derivative of dynamic pressure with respect to v velocity component			output	Global	q_partials
pqw	Partial derivative of dynamic pressure with respect to w velocity component			output	Global	q_partials
pqx	Partial derivative of dynamic pressure with respect to x position component			output	Global	q_partials
pqy	Partial derivative of dynamic pressure with respect to y position component			output	Global	q_partials
pqz	Partial derivative of dynamic pressure with respect to z position component			output	Global	q_partials
prop(6)	Propellant weight for each engine type	kg	ainit	input	Global	agen1
prxx	Partial derivative of X component of the radius unit vector with respect to the X coordinate				Local	
prxy	Partial derivative of the x component of the radius vector with respect to the Plumbline y position component				Local	
prxz	Partial derivative of the x component of the radius vector with respect to the Plumbline z position component				Local	
pryy	Partial derivative of the y component of the radius vector with respect to the Plumbline y position component				Local	
pryz	Partial derivative of the y component of the radius vector with respect to the Plumbline z position component				Local	
przz	Partial derivative of the z component of the radius vector with respect to the Plumbline z position component				Local	
psrhom	Partial derivative of "srho" with respect to vehicle mass				Local	
ptandp1_alp	Partial derivative of tandp1 with respect to angle of attack				Local	
ptandp2_alp	Partial derivative of tandp2 with respect to angle of attack				Local	
ptm	Partial derivative of thrust magnitude with respect to vehicle mass				Local	
ptmzm	Partial derivative of lateral thrust moment component with respect to mass				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ptmzx	Partial derivative of lateral thrust moment component with respect to X position component				Local	
ptmzy	Partial derivative of lateral thrust moment component with respect to Y position component				Local	
ptmzz	Partial derivative of lateral thrust moment component with respect to Z position component				Local	
ptn	Conversion from pounds to newtons	nt/lbs	ainit	input	Global	const
ptp1cp	Partial derivative of "tandp1" with respect to pitch attitude				Local	
ptp1cy	Partial derivative of "tandp1" with respect to yaw attitude				Local	
ptp1m	Partial derivative of "tandp1" with respect to vehicle mass				Local	
ptp1u	Partial derivative of "tandp1" with respect to Plumbline y velocity component				Local	
ptp1v	Partial derivative of "tandp1" with respect to Plumbline z velocity component				Local	
ptp1w	Partial derivative of "tandp1" with respect to Plumbline x velocity component				Local	
ptp1x	Partial derivative of "tandp1" with respect to Plumbline x position component				Local	
ptp1y	Partial derivative of "tandp1" with respect to Plumbline y position component				Local	
ptp1z	Partial derivative of "tandp1" with respect to Plumbline z position component				Local	
ptp2cp	Partial derivative of "tandp2" with respect to pitch attitude				Local	
ptp2cy	Partial derivative of "tandp2" with respect to yaw attitude				Local	
ptp2m	Partial derivative of "tandp2" with respect to vehicle mass				Local	
ptp2u	Partial derivative of "tandp2" with respect to Plumbline y velocity component				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ptp2v	Partial derivative of "tandp2" with respect to Plumbline z velocity component				Local	
ptp2w	Partial derivative of "tandp2" with respect to Plumbline x velocity component				Local	
ptp2x	Partial derivative of "tandp2" with respect to Plumbline x position component				Local	
ptp2y	Partial derivative of "tandp2" with respect to Plumbline y position component				Local	
ptp2z	Partial derivative of "tandp2" with respect to Plumbline z position component				Local	
ptx	Partial derivative of vertical thrust component in body coordinate system with respect to plumbline z velocity component				Local	
ptxcp	Second partial derivative of normal thrust component in body coordinate system with respect to pitch and yaw attitude angles				Local	
ptxcpp	Second partial derivative of vertical thrust component in body coordinate system with respect to pitch attitude angle				Local	
ptxcpy	Second partial derivative of normal thrust component in body coordinate system with respect to pitch and yaw attitude angles				Local	
ptxcy	Second partial derivative of normal thrust component in body coordinate system with respect to yaw attitude angle				Local	
ptxcyy	Second partial derivative of normal thrust component in body coordinate system with respect to yaw attitude angle				Local	
ptxfm	Temporary variable				Local	
ptxm	Partial derivative of normal component of total thrust vector with respect to vehicle mass				Local	
ptxp_alp	Partial derivative of normal component				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
	of thrust in alternate system with respect to angle of attack					
ptxu	Partial derivative of vertical thrust component in body coordinate system with respect to plumbline y velocity component				Local	
ptxv	Partial derivative of vertical thrust component in body coordinate system with respect to plumbline z velocity component				Local	
ptxw	Partial derivative of vertical thrust component in body coordinate system with respect to plumbline x velocity component				Local	
ptxx	Partial derivative of normal component of total thrust vector with respect to plumbline x position coordinate				Local	
ptxxt	Partial derivative of normal component of thrust vector with respect to Plumbline x position component				Local	
ptxy	Partial derivative of normal component of total thrust vector with respect to plumbline y position coordinate				Local	
ptxyt	Partial derivative of normal component of thrust vector with respect to Plumbline y position component				Local	
ptxz	Partial derivative of normal component of total thrust vector with respect to plumbline z position coordinate				Local	
ptxzt	Partial derivative of normal component of thrust vector with respect to Plumbline z position component				Local	
ptx_alp	Partial derivative of normal component of thrust with respect to angle of attack				Local	
pty	Partial derivative of ty2 with respect to plumbline z velocity component				Local	
pty1cp	Partial derivative of ty1 with respect to pitch attitude angle				Local	
pty1cy	Partial derivative of ty1 with respect to yaw attitude angle				Local	
pty1m	Partial derivative of "ty1" with respect to vehicle mass				Local	
pty1u	Partial derivative of ty1 with respect				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
	to plumbline y velocity component					
pty1v	Partial derivative of ty1 with respect to plumbline z velocity component				Local	
pty1w	Partial derivative of ty1 with respect to plumbline x velocity component				Local	
pty1x	Partial derivative of ty1 with respect to plumbline x position component				Local	
pty1y	Partial derivative of ty1 with respect to plumbline y position component				Local	
pty1z	Partial derivative of ty1 with respect to plumbline z position component				Local	
pty1_alp	Partial derivative of longitudinal component of thrust for engine 1 with respect to angle of attack				Local	
pty2cp	Partial derivative of ty2 with respect to pitch attitude angle				Local	
pty2cy	Partial derivative of ty2 with respect to yaw attitude angle				Local	
pty2m	Partial derivative of "ty2" with respect to vehicle mass				Local	
pty2u	Partial derivative of ty2 with respect to plumbline y velocity component				Local	
pty2v	Partial derivative of ty2 with respect to plumbline z velocity component				Local	
pty2w	Partial derivative of ty2 with respect to plumbline x velocity component				Local	
pty2x	Partial derivative of ty2 with respect to plumbline x position component				Local	
pty2y	Partial derivative of ty2 with respect to plumbline y position component				Local	
pty2z	Partial derivative of ty2 with respect to plumbline z position component				Local	
pty2_alp	Partial derivative of longitudinal component of thrust for engine 2 with respect to angle of attack				Local	
ptycp	Second partial derivative of longitudinal component of thrust vector in body coordinate system with respect to pitch and yaw attitude angles				Local	
ptycpp	Second partial derivative of longitudinal thrust component in body coordinate system with respect to pitch attitude				Local	

Variable	Variable Definition angle	Units	Source	I/O	Status	Common Block
ptycpy	Second partial derivative of longitudinal component of thrust vector in body coordinate system with respect to pitch and yaw attitude angles				Local	
ptycy	Second partial derivative of longitudinal component of thrust vector in body coordinate system with respect to yaw attitude angle				Local	
ptycyy	Second partial derivative of longitudinal component of thrust vector in body coordinate system with respect to yaw attitude angle				Local	
ptyfm	Partial derivative of longitudinal component of thrust with respect to mass				Local	
ptym	Partial derivative of "tandy" with respect to vehicle mass				Local	
ptyp_alp	Partial derivative of longitudinal component of thrust in alternate system with respect to angle of attack				Local	
ptyu	Partial derivative of "tandy" with respect to Plumbline Y velocity component				Local	
ptyv	Partial derivative of "tandy" with respect to Plumbline Z velocity component				Local	
ptyw	Partial derivative of "tandy" with respect to Plumbline X velocity component				Local	
ptyx	Partial derivative of "tandy" with respect to Plumbline X position component				Local	
ptyxt	Partial derivative of thrust magnitude with respect to X position component				Local	
ptyy	Partial derivative of "tandy" with respect to Plumbline Y position component				Local	
ptyyt	Partial derivative of axial component of thrust vector with respect to Plumbline y position component				Local	
ptyz	Partial derivative of "tandy" with respect to Plumbline Z position component				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ptyzt	Partial derivative of thrust magnitude with respect to Z position component				Local	
pty_alp	Partial derivative of longitudinal component of thrust with respect to angle of attack				Local	
ptz	Second partial derivative of lateral thrust component in body coordinate system with respect to pitch and yaw attitude angles				Local	
ptzcp	Second partial derivative of lateral thrust component in body coordinate system with respect to pitch and yaw attitude angles				Local	
ptzcpp	Second partial derivative of lateral thrust component in body coordinate system with respect to pitch attitude angle				Local	
ptzcpy	Second partial derivative of lateral thrust component in body coordinate system with respect to pitch and yaw attitude angles				Local	
ptzcy	Second partial derivative of lateral thrust component in body coordinate system with respect to yaw attitude angle				Local	
ptzcyy	Second partial derivative of lateral thrust component in body coordinate system with respect to yaw attitude angle				Local	
ptzfm	Partial derivative of lateral component of thrust with respect to mass				Local	
ptzm	Partial derivative of lateral component of total thrust vector with respect to vehicle mass				Local	
ptzu	Partial derivative of lateral component of total thrust vector with respect to plumbline y velocity coordinate				Local	
ptzv	Partial derivative of lateral component of total thrust vector with respect to plumbline z velocity coordinate				Local	
ptzw	Partial derivative of lateral component of total thrust vector with respect to plumbline x velocity coordinate				Local	
ptzx	Partial derivative of lateral component				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
	of total thrust vector with respect to plumbline x position coordinate					
ptzy	Partial derivative of lateral component of total thrust vector with respect to plumbline y position coordinate				Local	
ptzz	Partial derivative of lateral component of total thrust vector with respect to plumbline z position coordinate				Local	
ptz_alp	Partial derivative of lateral component of thrust with respect to angle of attack				Local	
pvru	Partial derivative of relative velocity magnitude with respect to the Plumbline y velocity component				Local	
pvrv	Partial derivative of relative velocity magnitude with respect to the Plumbline z velocity component				Local	
pvrw	Partial derivative of relative velocity magnitude with respect to the Plumbline x velocity component				Local	
pvrx	Partial derivative of relative velocity (including winds) with respect to Plumbline x position component				Local	
pvry	Partial derivative of relative velocity magnitude with respect to the Plumbline y position component				Local	
pvrz	Partial derivative of relative velocity (including winds) with respect to Plumbline z position component				Local	
pw1x	Partial derivative of the x component of the wind vector with respect to the Plumbline x position component				Local	
pw1y	Partial derivative of the x component of the wind vector with respect to the Plumbline y position component				Local	
pw1z	Partial derivative of the x component of the wind vector with respect to the Plumbline z position component				Local	
pw2x	Partial derivative of y component of the wind vector with respect to Plumbline x position component				Local	
pw2y	Partial derivative of the y component of the wind vector with respect to the Plumbline y position component				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
pw2z	Partial derivative of y component of the wind vector with respect to Plumbline z position component				Local	
pw3x	Partial derivative of the z component of the wind vector with respect to the Plumbline x position component				Local	
pw3y	Partial derivative of the z component of the wind vector with respect to the Plumbline y position component				Local	
pw3z	Partial derivative of the z component of the wind vector with respect to the Plumbline z position component				Local	
pwmdh	Partial derivative of the wind magnitude with respect to altitude				Local	
px1cp	Second partial derivative of x component of direction cosine of the normal component of the vehicle (xhat(1)) with respect to the pitch and yaw attitude angles				Local	
px1cpp	Second partial derivative of x component of direction cosine of the normal component of the vehicle (xhat(1)) with respect to the pitch attitude angle				Local	
px1cpy	Second partial derivative of x component of direction cosine of the normal component of the vehicle (xhat(1)) with respect to the pitch and yaw attitude angles				Local	
px1cy	Second partial derivative of x component of direction cosine of the normal component of the vehicle (xhat(1)) with respect to the yaw attitude angle				Local	
px1cyy	Second partial derivative of x component of direction cosine of the normal component of the vehicle (xhat(1)) with respect to the yaw attitude angle				Local	
px2cp	Second partial derivative of y component of direction cosine of the normal component of the vehicle (xhat(2)) with respect to the pitch and yaw attitude angles				Local	
px2cpp	Second partial derivative of y				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
	component of direction cosine of the normal component of the vehicle (xhat(2)) with respect to the pitch attitude angle					
px2cpy	Second partial derivative of y component of direction cosine of the normal component of the vehicle (xhat(2)) with respect to the pitch and yaw attitude angles				Local	
px2cy	Second partial derivative of y component of direction cosine of the normal component of the vehicle (xhat(2)) with respect to the yaw attitude angle				Local	
px2cyy	Second partial derivative of y component of direction cosine of the normal component of the vehicle (xhat(2)) with respect to the yaw attitude angle				Local	
px3cp	Partial derivative of xhat(3) with respect to pitch attitude				Local	
px3cpp	Second partial derivative of xhat(3) with respect to pitch attitude				Local	
px3cpy	Second partial derivative of xhat(3) with respect to pitch and yaw attitude				Local	
px3cy	Partial derivative of xhat(3) with respect to yaw attitude				Local	
px3cyy	Second partial derivative of xhat(3) with respect to yaw attitude				Local	
pxcgm	Partial derivative of normal CG with respect to mass				Local	
pxcpcp	Second partial derivative of aerodynamic yawing moment with respect to pitch attitude				Local	
pxcpcy	Second partial derivative of aerodynamic yawing moment with respect to pitch and yaw attitude				Local	
pxcycy	Second partial derivative of aerodynamic yawing moment with respect to yaw attitude				Local	
pxhat_alp	Partial derivative of xhat with respect to angle of attack				Local	
pxm	Partial derivative of X acceleration component with respect to mass			output	Global	calif

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
pxu	Partial derivative of X acceleration component with respect to Y velocity component			output	Global	calif
pxv	Partial derivative of X acceleration component with respect to Z velocity component			output	Global	calif
pxw	Partial derivative of X acceleration component with respect to Z velocity component			output	Global	calif
pxx	Partial derivative of X acceleration component with respect to X position component			output	Global	calif
pxy	Partial derivative of X acceleration component with respect to Y position component			output	Global	calif
pxz	Partial derivative of X acceleration component with respect to Z position component			output	Global	calif
px_alp	Partial derivative of X acceleration component with respect to angle of attack				Local	
py1cp	Second partial derivative of x component of direction cosine of the axial component of the vehicle (yhat(1)) with respect to the pitch and yaw attitude angles				Local	
py1cpp	Second partial derivative of x component of direction cosine of the axial component of the vehicle (yhat(1)) with respect to the pitch attitude angle				Local	
py1cpy	Second partial derivative of x component of direction cosine of the axial component of the vehicle (yhat(1)) with respect to the pitch and yaw attitude angles				Local	
py1cy	Second partial derivative of x component of direction cosine of the axial component of the vehicle (yhat(1)) with respect to the yaw attitude angle				Local	
py1cyy	Second partial derivative of x component of direction cosine of the axial component of the vehicle (yhat(1)) with respect to the yaw attitude angle				Local	
py2cp	Second partial derivative of y component of direction cosine of the				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
	axial component of the vehicle (yhat(2)) with respect to the pitch and yaw attitude angles					
py2cpp	Second partial derivative of y component of direction cosine of the axial component of the vehicle (yhat(2)) with respect to the pitch attitude angle				Local	
py2cpy	Second partial derivative of y component of direction cosine of the axial component of the vehicle (yhat(2)) with respect to the pitch and yaw attitude angles				Local	
py2cy	Second partial derivative of y component of direction cosine of the axial component of the vehicle (yhat(2)) with respect to the yaw attitude angle				Local	
py2cyy	Second partial derivative of y component of direction cosine of the axial component of the vehicle (yhat(2)) with respect to the yaw attitude angle				Local	
py3cp	Second partial derivative of z component of direction cosine of the axial component of the vehicle (yhat(3)) with respect to the pitch and yaw attitude angles				Local	
py3cpp	Second partial derivative of z component of direction cosine of the axial component of the vehicle (yhat(3)) with respect to the pitch attitude angle				Local	
py3cpy	Second partial derivative of z component of direction cosine of the axial component of the vehicle (yhat(3)) with respect to the pitch and yaw attitude angles				Local	
py3cy	Second partial derivative of z component of direction cosine of the axial component of the vehicle (yhat(3)) with respect to the yaw attitude angle				Local	
py3cyy	Second partial derivative of z component of direction cosine of the axial component of the vehicle (yhat(3)) with respect to the yaw attitude angle				Local	
pycgm	Partial derivative of longitudinal component of CG with respect to mass				Local	
pycpcp	Second partial derivative of				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
	aerodynamic rolling moment with respect to pitch attitude					
pycpcy	Second partial derivative of aerodynamic rolling moment with respect to pitch and yaw attitude				Local	
pycpm	Partial derivative of longitudinal component of center of pressure with respect to mass				Local	
pycycy	Second partial derivative of aerodynamic rolling moment with respect to yaw attitude				Local	
pyhat_alp	Partial derivative of yhat with respect to angle of attack				Local	
pym	Partial derivative of Y acceleration component with respect to mass			output	Global	calif
pyocpp	Second partial derivative of ty1 with respect to pitch attitude angle				Local	
pyocpy	Second partial derivative of ty1 with respect to pitch and yaw attitude angles				Local	
pyocyy	Second partial derivative of ty1 with respect to yaw attitude angle				Local	
pytcpp	Second partial derivative of ty2 with respect to pitch attitude angle				Local	
pytcpy	Second partial derivative of ty2 with respect to pitch and yaw attitude angles				Local	
pytcyy	Second partial derivative of ty2 with respect to yaw attitude angle				Local	
pyu	Partial derivative of Y acceleration component with respect to Y velocity component			output	Global	calif
pyv	Partial derivative of Y acceleration component with respect to Z velocity component			output	Global	calif
pyw	Partial derivative of Y acceleration component with respect to X velocity component			output	Global	calif
pyx	Partial derivative of Y acceleration component with respect to X position component			output	Global	calif
pyy	Partial derivative of Y acceleration component with respect to Y position component			output	Global	calif

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
pyz	Partial derivative of Y acceleration component with respect to Z position component			output	Global	calif
py_alp	Partial derivative of Y acceleration component with respect to angle of attack				Local	
pz1cp	Second partial derivative of x component of direction cosine of the lateral component of the vehicle (zhat(1)) with respect to the pitch and yaw attitude angles				Local	
pz1cpp	Second partial derivative of x component of direction cosine of the lateral component of the vehicle (zhat(1)) with respect to the pitch attitude angle				Local	
pz1cpy	Second partial derivative of x component of direction cosine of the lateral component of the vehicle (zhat(1)) with respect to the pitch and yaw attitude angles				Local	
pz1cy	Partial derivative of x component of direction cosine of the lateral component of the vehicle (zhat(1)) with respect to the yaw attitude angle				Local	
pz1cyy	Second partial derivative of x component of direction cosine of the lateral component of the vehicle (zhat(1)) with respect to the yaw attitude angle				Local	
pz2cp	Second partial derivative of y component of direction cosine of the lateral component of the vehicle (zhat(2)) with respect to the pitch and yaw attitude angles				Local	
pz2cpp	Second partial derivative of y component of direction cosine of the lateral component of the vehicle (zhat(2)) with respect to the pitch attitude angle				Local	
pz2cpy	Second partial derivative of y component of direction cosine of the lateral component of the vehicle (zhat(2)) with respect to the pitch and yaw attitude angles				Local	
pz2cy	Second partial derivative of y				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
	component of direction cosine of the lateral component of the vehicle (zhat(2)) with respect to the yaw attitude angle					
pz2cyy	Second partial derivative of y component of direction cosine of the lateral component of the vehicle (zhat(2)) with respect to the yaw attitude angle				Local	
pz3cp	Second partial derivative of z component of direction cosine of the lateral component of the vehicle (zhat(3)) with respect to the pitch and yaw attitude angles				Local	
pz3cpp	Second partial derivative of z component of direction cosine of the lateral component of the vehicle (zhat(3)) with respect to the pitch attitude angle				Local	
pz3cpy	Second partial derivative of z component of direction cosine of the lateral component of the vehicle (zhat(3)) with respect to the pitch and yaw attitude angles				Local	
pz3cy	Partial derivative of z component of direction cosine of the lateral component of the vehicle (zhat(3)) with respect to the yaw attitude angle				Local	
pz3cyy	Second partial derivative of z component of direction cosine of the lateral component of the vehicle (zhat(3)) with respect to the yaw attitude angle				Local	
pzcpcp	Second partial derivative of aerodynamic pitching moment with respect to pitch attitude				Local	
pzcycy	Second partial derivative of aerodynamic pitching moment with respect to yaw attitude				Local	
pzm	Partial derivative of Z acceleration component with respect to mass			output	Global	calif
pzu	Partial derivative of Z acceleration component with respect to Y velocity component			output	Global	calif
pzv	Partial derivative of Z acceleration			output	Global	calif

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
	component with respect to Z velocity component					
pzw	Partial derivative of Z acceleration component with respect to X velocity component			output	Global	calif
pzx	Partial derivative of Z acceleration component with respect to X position component			output	Global	calif
pzy	Partial derivative of Z acceleration component with respect to Y position component			output	Global	calif
pzz	Partial derivative of Z acceleration component with respect to Z position component			output	Global	calif
pz_alp	Partial derivative of Z acceleration component with respect to angle of attack				Local	
q	Dynamic pressure	nt/m^2	bdr1i	output	Global	agen2
qalpha	Base force tables		ainit	input	Global	aerodi
qalpha_max	Maximum value of q alpha		ainit	input	Global	qalpha_ max
qref	Reference dynamic pressure	kg/m^2			Local	
r	Radius from earth's center	m	bdr1i	output input	Global	agen2
r2	Radius squared	m^2			Local	
re	Earth's radius	m	ainit	input	Global	const
reyno	Reynold's number				Local	
rho	Atmospheric density				Local	
rthe	Current earth radius			output input	Global	agen2
s(15)	Aerodynamic reference area for each thrust event	m^2	ainit	input	Global	agen1
salp	Sine of angle of attack				Local	
salp_new	Sine of new value of angle of attack				Local	
sazw	Sine of the wind azimuth angle				Local	
schip	Sine of pitch attitude angle		ader1	output input	Global	agen2
schir	Sine of roll attitude angle		ader1	output input	Global	agen2
schiy	Sine of yaw attitude angle		ader1	output	Global	agen2

Variable	Variable Definition	Units	Source	I/O input	Status	Common Block
sdd	Not used				Local	
smach	Saved value of mach number				Local	
solda	Sine of past value of angle of attack (olda)				Local	
sos	Speed of sound	m/sec			Local	
sq	Product of dynamic pressure and aerodynamic reference area				Local	
sreyno	Square root of reynold number				Local	
srho	Sine of angle between aerodynamic reference arm and longitudinal axis of vehicle				Local	
state	State variable array (interpolated)				Local	
stbl	Stored state array				Local	
su	Y component of relative velocity vector	m/sec		output	Global	agen2
sv	Z component of relative velocity vector	m/sec		output input	Global	agen2
sw	X component of relative velocity vector	m/sec		output input	Global	agen2
t	Time from lift-off	sec	dpir	input	Global	forint
tandp1	Tangent of pitch gimbal angle of first equivalent engine				Local	
tandp1_new	Base force tables		ainit	input	Global	aerodi
tandp2	Tangent of pitch gimbal angle of second equivalent engine				Local	
tandp2_new	Base force tables		ainit	input	Global	aerodi
tandy	Tangent of yaw gimbal angle				Local	
tanp	Tangent of fixed value of pitch gimbal angle				Local	
tany	Tangent of fixed value of yaw gimbal angle				Local	
tau	Throttle level				Local	
tau2	Interpolated value of thrust trottle level when using the thrust tailoff option (MPSFLG=6)				Local	
tau_const(10,	Constant value of throttle used for constant throttle after acceleration limit		ainit	input	Global	mps
tbk	Time used in backward integration	sec		input	Global	bakint

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
tbl(201)	Time table for stored state variables	sec		output input	Global	tabs
tem1ca	Temporary variable associated with viscous interaction effects				Local	
tem1cm	Temporary variable associated with viscous interaction effects				Local	
tem1cn	Temporary variable associated with viscous interaction effects				Local	
tem2ca	Temporary variable associated with viscous interaction effects				Local	
tem2cm	Temporary variable associated with viscous interaction effects				Local	
tem2cn	Temporay variable associated with viscous interaction effects				Local	
temp	Temporary variable				Local	
temp1	Temporary variable				Local	
temp2	Temporary variable				Local	
temp3	Temporary variable				Local	
temp4	Temporary variable				Local	
temp5	Temporary variable				Local	
temp6	Temporary variable				Local	
temp7	Temporary variable				Local	
temp8	Temporary variable				Local	
temp9	Temporary variable				Local	
tend	Termination time of forward trajectory (start of backward intgration)	sec	bakrn	input	Global	bakint
tfx	Normal component of fixed engine thrust vector	nt			Local	
tfxx	Normal component of fixed engine thrust vector	nt			Local	
tfy	Axial component of fixed engine thrust vector	nt			Local	
tfyy	Axial component of fixed engine thrust vector	nt			Local	
tfzz	Lateral component of thrust vector	nt			Local	
thr1	Thrust magnitude of engines that are not being throttled during thrust tailoff	nt			Local	
thr2	Thrust magnitude of engines being throttled during thrust tailoff	nt			Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
throt(15)	Throttle table for MPS motor		ainit	input	Global	mps
thrust	Atmospheric thrust magnitude	nt			Local	
thrvac	Vacuum thrust	nt			Local	
tmz	Z component of thrust moment	nt -m	bdr1i	output input	Global	agen5
tne(6,15)	Array which defines the number of engines per thrust event		ainit	input	Global	agen1
tq	Time that the minh phase is initiated	sec	bdr1i	output	Global	agen1
tx	Vertical component of moment balanced thrust	newton	bdr1i	output input	Global	agen5
txcg(15)	Vertical center of gravity table	m	bthrev	input	Global	bodyb
txp	Partial derivative of normal thrust component in moment balance coordinate system with respect to plumbline z velocity component				Local	
txpcp	Second partial derivative of normal component of thrust vector in moment balance coordinate system with respect to pitch and yaw attitude angles				Local	
txpcpp	Second partial derivative of normal component of thrust vector in moment balance coordinate system with respect to pitch attitude angle				Local	
txpcpy	Second partial derivative of normal component of thrust vector in moment balance coordinate system with respect to pitch and yaw attitude angles				Local	
txpcy	Second partial derivative of normal component of thrust vector in moment balance coordinate system with respect to yaw attitude angle				Local	
txpcyy	Second partial derivative of normal component of thrust vector in moment balance coordinate system with respect to yaw attitude angle				Local	
txpu	Partial derivative of normal thrust component in moment balance coordinate system with respect to plumbline y velocity component				Local	
txpv	Partial derivative of normal thrust component in moment balance coordinate system with respect to plumbline z velocity component				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
txpw	Partial derivative of normal thrust component in moment balance coordinate system with respect to plumbline x velocity component				Local	
txpx	Partial derivative of normal component of thrust vector in body coordinate system with respect to plumbline x position component				Local	
txpy	Partial derivative of normal component of thrust vector in body coordinate system with respect to plumbline y position component				Local	
txpz	Partial derivative of normal component of thrust vector in body coordinate system with respect to plumbline z position component				Local	
txp_new	Normal thrust component in alternate system assuming new angle of attack				Local	
tx_new	Normal thrust component assuming new angle of attack				Local	
ty	Longitudinal component of moment balanced thrust	newton	bdr1i	output input	Global	agen5
ty1	Axial thrust vector component of one of the two moment balanced equivalent engines				Local	
ty1_new	Longitudinal thrust component of engine 1 assuming new angle of attack				Local	
ty2	Axial thrust vector component of the second of the two moment balanced equivalent engines				Local	
ty2_new	Longitudinal thrust component of engine 2 assuming new angle of attack				Local	
tycg(15)	Longitudinal center of gravity table	m	bthrev	input	Global	bodyb
typ	Partial derivative of longitudinal thrust component in moment balance coordinate system with respect to plumbline z velocity component				Local	
typcp	Second partial derivative of longitudinal component of thrust vector in moment balance coordinate system with respect to pitch and yaw attitude angles				Local	
typcpp	Second partial derivative of longitudinal component of thrust vector in moment balance coordinate system with respect				Local	

Variable	Variable Definition to pitch attitude angle	Units	Source	I/O	Status	Common Block
typcpy	Second partial derivative of longitudinal component of thrust vector in moment balance coordinate system with respect to pitch and yaw attitude angles				Local	
typcy	Second partial derivative of longitudinal component of thrust vector in moment balance coordinate system with respect to yaw attitude angle				Local	
typcyy	Second partial derivative of longitudinal component of thrust vector in moment balance coordinate system with respect to yaw attitude angle				Local	
typu	Partial derivative of longitudinal thrust component in moment balance coordinate system with respect to plumbline y velocity component				Local	
typv	Partial derivative of longitudinal thrust component in moment balance coordinate system with respect to plumbline z velocity component				Local	
typw	Partial derivative of longitudinal thrust component in moment balance coordinate system with respect to plumbline x velocity component				Local	
typx	Partial derivative of longitudinal component of thrust vector in body coordinate system with respect to plumbline x position component				Local	
typy	Partial derivative of longitudinal component of thrust vector in body coordinate system with respect to plumbline y position component				Local	
typz	Partial derivative of longitudinal component of thrust vector in body coordinate system with respect to plumbline z position component				Local	
typ_new	Total longitudinal thrust component in alternate system assuming new angle of attack				Local	
ty_new	Longitudinal thrust component assuming new angle of attack				Local	
tz	Time of simulation initiation	sec	bdr1i	output	Global	agen1
tz_new	Lateral thrust component assuming new				Local	

Variable	Variable Definition angle of attack	Units	Source	I/O	Status	Common Block
u	Y component of plumbline inertial velocity vector	m/sec	dpir	input	Global	forint
ubl(201)	Array for storage of Y component of inertial velocity vector during forward integration	m/sec	aeosr	input	Global	tabs
udxdy2	1.+ (dx/dy)^2				Local	
umf	One minus the earth flattening coefficient		ainit	input	Global	agen2
umfc	Base force tables		ainit	input	Global	aerodi
v	Z component of plumbline inertial velocity vector	m/sec	dpir	input	Global	forint
va	Spline coefficient for viscous interaction coefficients interpolation				Local	
vaerdd	Array of partial derivatives of viscous interaction coefficients with respect to mach number				Local	
vaero	Array of viscous interaction coefficients				Local	
vb	Spline coefficient for viscous interaction coefficients interpolation				Local	
vbl(201)	Array for storage of Z component of inertial velocity vector during forward integration	m/sec	aeosr	output	Global	tabs
vc	Spline coefficient for viscous interaction coefficients interpolation				Local	
visc	Logical flag indicating the simulation of viscous interaction effects			output	Global	agen5
v r	Magnitude of relative velocity	m/sec	bdr1i	output input	Global	agen2
w	X component of plumbline inertial velocity vector	m/sec	dpir	input	Global	forint
warg	Interpolated wind parameters				Local	
wargd	Derivative of wind parameters with respect to altitude				Local	
wbl(201)	Array for storage of X component of inertial velocity vector during forward integration	m/sec	aeosr	output	Global	tabs
wdotboost	Flowrate of booster	kg/sec			Local	
wdotmps	Flowrate of main propulsion system	kg/sec			Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
wdot_boost(2	Stored booster flowrate tables	kg/sec		output	Global	tabs
wdot_mps(20	Stored main propulsion system flowrate tables	kg/sec			Global	tabs
wgtboost	Booster propellant mass	kg			Local	
wgtmps	MPS propellant mass	kg			Local	
wgt_boost(20	Stored booster mass tables	kg		output input	Global	tabs
wgt_mps(201	Stored MPS mass tables	kg			Global	tabs
wind	Wind vector	m/sec			Local	
wint	Initial vehicle weight	kg	bdr1i	input	Global	olow
wmag	Wind speed magnitude	m/sec	bdr1i	output	Global	agen5
x	X component of plumbline position vector	m	dpir	input	Global	forint
xbl(201)	Array for storage of X component of position vector during forward integration	m	aeosr	input	Global	tabs
xcg	Vertical component of center of gravity	m			Local	
xgpa	Vertical component of gimbal points of equivalent booster engines	m	athrev	input	Global	avggp
xhat	Direction cosine of vehicle normal vector in Plumbline coordinates				Local	
xhat_new	New value of xhat based on angle of attack				Local	
xk1	Coefficient used in stagnation heating equation				Local	
xk2	Coefficient used in stagnation heating equation				Local	
xk3	Coefficient used in stagnation heating equation				Local	
xk4	Coefficient used in stagnation heating equation				Local	
xk5	Coefficient used in stagnation heating equation				Local	
xk6	Coefficient used in stagnation heating equation				Local	
xk7	Coefficient used in stagnation heating equation				Local	
xknud	Knudsen's number				Local	
xlamda	Mean free path				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
xlen	Reference length	m	bthrev	input	Global	bodyb
xm	Current vehicle mass	kg	bdri	input	Global	agen2
xmach	Mach number		bdr1i	output	Global	agen2
xmaug	Auxiliary vehicle mass	kg	athrev	input	Global	agen1
xmbl(201)	Array for storage of vehicle mass (continuous) during forward integration		aeosr	input	Global	tabs
xmi	Integrated vehicle mass less jettisoned inerts.	kg	dpir	input	Global	forint
xmiad	Continuous portion of vehicle mass	kg	ader1	output input	Global	agen2
xmu	Coefficient of viscosity				Local	
xnu1	Temporary variable			output	Global	qalpha_ max
xref	Vertical component of moment reference point		bthrev	input	Global	bodyb
y	Y component of plumbline position vector	m	dpir	input	Global	forint
ybl(201)	Array for storage of Y component of position vector during forward integration	m	aeosr	input	Global	tabs
ycg	Longitudinal component of center of gravity	m			Local	
ycp	Longitudinal component of center of pressure	m			Local	
ygpa	Longitudinal component of gimbal point of booster equivalent engine	m	athrev	input	Global	avggp
yhat	Direction cosine of vehicle axial vector in Plumbline coordinates				Local	
yhat_new	New value of yhat based on new alpha				Local	
yl(7,20)	Integrated values of adjoint equations of motion		dpir	input	Global	bakint
yref	Longitudinal component of moment reference point	m	bthrev	input	Global	bodyb
z	Z component of plumbline position vector	m	dpir	input	Global	forint
zbl(201)	Array for storage of Z position component during forward integration	m	aeosr	input	Global	tabs
zgpa	Lateral component of gimbal point of booster equivalent engines	m	athrev	input	Global	avggp
zhat	Direction cosine of vehicle lateral				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
	vector in Plumbline coordinates					

```
      SUBROUTINE BDR1I                                                  BDR1    1
C*********************************************************************
C       COMPUTES TIME DEPENDENT PORTION OF ADJOINT EQUATIONS
C*********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      parameter (ktbl=201)

      logical mombal,MPFLAG,CALCPAR

      CHARACTER MPNAME*30
      CHARACTER*60 HEAD
      CHARACTER*6 MISION,ATMOSN
      character*12 header

      integer engines

      INTEGER EVENTNUM

      REAL*8 MOBOOS,MOMAIN,BOMDOT,MNMDOT,LONCG,LATCG,NORCG,LONCGP
      REAL*8 LATCGP,NORCGP

      COMMON/AERODI/AEROD(30,17),AEROB(50,6,3),TOMACH(25),DELCA(25),
     *PNMONE(15),BKFABT(50,2),BKFTHR(50,2),DELCN(25),DELCM(25)

      COMMON/AGEN1/TIME(2,16),TAUT(15),TAUW(15),TZERO,TLIFT,TTILT,TMINH,
     *DTZ,TQ,TL,XMAUG,TNE(6,15),ENGDAT(8,15),S(15),WD(15),WJET(15),
     *HEAD,PRINT(15),STEP(15),HNOM(15),HMAX(15),PROP(6),TBEGR,TENDR,
     *CHRDOT,CHIRO,FAZ,BO(3,2),CBAXIL,TCHIR,VRCUT,FPRFAC,CHVEL

      COMMON/AGEN2/AZ,SA,DA,ALF,ALFY,CHIP,SCHIP,CCHIP,CHIY,SCHIY,CCHIY,
     *CHIR,SCHIR,CCHIR,STH,CTH,STHL,CTHL,DPHIZ,RTHE,R,VR,XM,SW,SU,SV,Q,
     *VIV(25),DVARS(13),DELXDW(15,3),DELXDR(7,5),DELXD(15,7),WZERO,ALT,
     *XMACH,QDOT,FAA,FAN,A(3,3),CASE,CT2,UMF,A12W,A22W,A32W,XJEXT,BEU,
     *BYL,BEL,BHMAX(15),BHMN,BSTEP(15),HNMB,HMXB,TPOLY,XMIAD

      COMMON/AGEN3/ITHR,IWD,IXR,JUMP,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISION,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)

      COMMON/AGEN5/CHIBAS,CHPBAS,CHYBAS,CHISAV,TXX,TYY,TZZ,AMX,AMY,AMZ,
     *TMZ,THMFA,SIDE,WMAG,AZW,FANC,FLATC,FOM,VISC

      COMMON/ARDC/PAT(14),ATMOSN,IATMOS,HBIAS,HAERO

      COMMON/ALAT/ALAT,ALONG,ALTLS,NTABLE

      COMMON/AVGGP/XGPA,YGPA,ZGPA

      COMMON/BAKINT/BBANK(2),TBK,TBKD,YL(7,20),YLD(7,20),BSAVE(2700),
     *NBTRG1,TBV1,NBTRG2,TBV2,NBTRG3,TBV3,NBTRG4,TBV4,NBTRG5,TBV5,TEND,
```



```
     *DTB4,DU3,D23,D43,WISSD(20,20)

      COMMON/BGEN3/NOSS,JTB,NENT,NACT,NOM1,NP(7)

      COMMON/BODYB/TBDWT(15),TXCG(15),TYCG(15),XLEN,XREF,YREF,MBS1,MBS2,
     *MBS3,MBS4,MBS5,MBS6,MBS7,MBS8,MBS9,MBS10

      COMMON/CALIF/PXX,PYX,PZX,PXY,PYY,PZY,PXZ,PYZ,PZZ,PXW,PYW,PZW,PXU,
     *PYU,PZU,PXV,PYV,PZV,PXM,PYM,PZM,PDXCP,PDYCP,PDZCP,PDXCY,PDYCY,
     *PDZCY,PDXCPP,PDYCPP,PDZCPP,PDXCPY,PDYCPY,PDZCPY,PDXCYY,PDYCYY,
     *PDZCYY

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM

      common/cp_input/cp_input

      COMMON/DELFAB/DELALT(25),DELFAB(25),M30,M31,ITOL(15)

      COMMON/DELMBS/MBS21,MBS30,MBS31

      common/engines/engines(15,15)

      common/enput/psirst(6,5),header(20),nvrst(5),kcdres(6,5),last

      COMMON/FORINT/HBANK(2),T,TT,W,U,V,X,Y,Z,XMI,ZAP(18),DVAR(25),FSAVE
     *(625),NTRG1,TV1,NTRG2,TV2,NTRG3,TV3,NTRG4,TV4,NTRG5,TV5,NTRG6,TV6,
     *NTRG7,TV7,NTRG8,TV8,NTRG9,TV9,NTRG10,TV10,NTRG11,TV11,NTRG12,TV12,
     *NTRG13,TV13,KIND,KRDER,EU,EL,AYL,HMX,HMN,HNM

      common/gnu/gnu(20)

      COMMON/GRAV/GTO,G11,G22,G23,G33

      COMMON/HEAT/MAXHAT

      COMMON/MLAST/MLAST,MMMM,KKKK

      COMMON/LBM/TIMLBM(30),TLBM(30,2)              ,PLBM(2),MLBM1,MLBM2,
     *ALBM(2),BLBM(2),CLBM(2),VLBM(2)

      COMMON/LBMMBL/MBLB1,MBLB2

      common/mombal/mombal

      COMMON/MPS/TIMMPS(15),FTBL(15),THROT(15),GLIM(15),TAUALT(5),
     *TAUTBL(5),MPS1,MPS2,TGLIM(10,15),GLIMIT,tau_const(10,5),jthrot(15)

      common/mps1/timemps(15),taumps(15),mpsinp(15)

      common/mult/nchiot(6),tbias(6),nstage,nstg(15),wf(5),ibranch

      COMMON/OMSRCS/FOMS(15),FRCS(15),WDOMS(15),WDRCS(15),WDDUMP(15)

      COMMON/OLOW/OLOW,WINT
```

```
      COMMON/PHRS/PHRW,PHRU,PHRV,PHRX,PHRY,PHRZ

      common/qalpha_max/qalpha_max,xnu1

      common/q_partials/pqx,pqy,pqz,pqw,pqu,pqv,alp_new

      common/rest/jstr,ncnrs(5),ntcn,tr(5),lsb,nf8,nopar,wibt(15)

      COMMON/TABS/TBL(ktbl),WBL(ktbl),UBL(ktbl),VBL(ktbl),XBL(ktbl),
     *YBL(ktbl),ZBL(ktbl),XMBL(ktbl),wgt_mps(ktbl),wgt_boost(ktbl),
     *  wdot_mps(ktbl),wdot_boost(ktbl)

      COMMON/VAERO/VAEROD(10,19),M18,M19,MBS18,MBS19

      COMMON/WIND/ALTTBL(100),WTBL(100,2),NWIND

      COMMON/MPROPIN/ MPFLAG
      COMMON/MPROPNM/ MPNAME

      DIMENSION DFAB(1),DDFAB(1),AF(1),BF(1),CF(1),XHAT(3),YHAT(3),
     *ZHAT(3),STBL(ktbl,11),STATE(11),SDD(7),AA(7),BB(7),CC(7),
     *FABT(4),
     *AFAB(4),BFAB(4),CFAB(4),AERO(13),AERODD(17),AAERO(13),BAERO(13),
     *CAERO(13),CG(2),CGBL(15,2),PCG(2),ACG(2),BCG(2),CCG(2),WARG(2),
     *WARGD(2),AW(2),BW(2),CW(2),WIND(3),xhat_new(3),yhat_new(3),
     *pxhat_alp(3),pyhat_alp(3),covl(7)

      COMMON/OUTPUT/OUTTH(6),OUTTV(6),OUTISP(6),OUTWD(6)

      EQUIVALENCE (WMAG,WARG(1)),(AZW,WARG(2)),(PWMDH,WARGD(1)),(PAZWDH,
     *WARGD(2))

      EQUIVALENCE (W,STATE(1)),(U,STATE(2)),(V,STATE(3)),(X,STATE(4)),(Y
     *,STATE(5)),(Z,STATE(6)),(XMI,STATE(7)),(wgtmps,state(8)),
     *   (wgtboost,state(9)),(wdotmps,state(10)),(wdotboost,state(11))

        equivalence (WBL(1),STBL(1,1)),(UBL(1),
     *STBL(1,2)),(VBL(1),STBL(1,3)),(XBL(1),STBL(1,4)),(YBL(1),STBL(1,5)
     *),(ZBL(1),STBL(1,6)),(XMBL(1),STBL(1,7)),(wgt_mps(1),stbl(1,8)),
     *   (wgt_boost(1),stbl(1,9)),(wdot_mps(1),stbl(1,10)),
     *   (wdot_boost(1),stbl(1,11))

      EQUIVALENCE (XCG,CG(1)),(YCG,CG(2)),(TXCG(1),CGBL(1,1)),(TYCG(1),
     *CGBL(1,2)),(PXCGM,PCG(1)),(PYCGM,PCG(2))

      EQUIVALENCE (PCAAM,AERODD(1)),(PCAOM,AERODD(2)),(PCLBM,AERODD(3)),
     *(PCMAM,AERODD(4)),(PCMOM,AERODD(5)),(PCNAM,AERODD(6)),(PCNBM,
     *AERODD(7)),(PCNOM,AERODD(8)),(PCYBM,AERODD(9)),(PCLOM,AERODD(10)),
     *(PCYOM,AERODD(11)),(PCNBOM,AERODD(12)),(pycpm,aerodd(13)),
     *(PFABH,AERODD(14)),(PFNBH,AERODD(15)),(PFMBH,AERODD(16)),
     *(pqrefh,aerodd(17))

      EQUIVALENCE (CAALP,AERO(1)),(CAO,AERO(2)),(CLBETA,AERO(3)),(CMALP,
     *AERO(4)),(CMO,AERO(5)),(CNALP,AERO(6)),(CNBETA,AERO(7)),
     *(CNO,AERO(8)),(CYBETA,AERO(9)),(CLO,AERO(10)),(CYO,AERO(11)),
     *(CNBO,AERO(12)),(ycp,aero(13)),(FAB,FABT(1)),(FNB,FABT(2)),
```



```
     *(FMB,FABT(3)),(qref,fabt(4))

      DIMENSION VAERO(18),VAERDD(18),VA(18),VB(18),VC(18)

      EQUIVALENCE (ACACF,VAERO(1)),(BCACF,VAERO(2)),(CCACF,VAERO(3)),
     *(ACNCF,VAERO(4)),(BCNCF,VAERO(5)),(CCNCF,VAERO(6)),
     *(ACMCF,VAERO(7)),(BCMCF,VAERO(8)),(CCMCF,VAERO(9)),
     *(ACAMF,VAERO(10)),(BCAMF,VAERO(11)),(CCAMF,VAERO(12)),
     *(ACNMF,VAERO(13)),(BCNMF,VAERO(14)),(CCNMF,VAERO(15)),
     *(ACMMF,VAERO(16)),(BCMMF,VAERO(17)),(CCMMF,VAERO(18))

      LOGICAL VISC,PONLY,NOWIND,GLIMIT,cp_input

      DATA XK1,XK2,XK3,XK4,XK5,XK6,XK7/17700.,.0019403,.3048,3048.,
     *110.4,.432,50063./

c**********************************************************************

      PONLY=KWTA(KCYTAB).EQ.2

      T=TEND-TBK                                                        BDR1  43

C**********************************************************************

C     GETS STATE VARIABLES FROM FORWARD RUN                             BDR1  45

c**********************************************************************

      IF(T.lt.TBL(1).and.T.ge.TQ+1.) then

         NENT=ktbl
         MBS5=NENT
         MBS6=0

C     EQUATE TABLES(151) TO TABLES(1)                                   BDR1  49

         TBL(ktbl)=TBL(1)
         WBL(ktbl)=WBL(1)
         UBL(ktbl)=UBL(1)
         VBL(ktbl)=VBL(1)
         XBL(ktbl)=XBL(1)
         YBL(ktbl)=YBL(1)
         ZBL(ktbl)=ZBL(1)
         XMBL(ktbl)=XMBL(1)
         wgt_mps(ktbl)=wgt_mps(1)
         wgt_boost(ktbl)=wgt_boost(1)
         wdot_mps(ktbl)=wdot_mps(1)
         wdot_boost(ktbl)=wdot_boost(1)

         BACKSPACE 3

         READ(3)(TBL(I),WBL(I),UBL(I),VBL(I),XBL(I),YBL(I),ZBL(I),
     *   XMBL(I),wgt_mps(i),wgt_boost(i),wdot_mps(i),wdot_boost(i),
     *   I=1,ktbl-1)

         BACKSPACE 3
```

```
        IF(TBL(ktbl-1).GT.TBL(ktbl)) NENT=ktbl-1

      endif

       call amulg7(mbs5,ktbl,t,tbl,state,stbl)

C*********************************************************************

      XMIAD=XMI
      XM=XMIAD+XMAUG

C*********************************************************************

C     CALCULATE ALTITUDE AND PARTIAL DERIVATIVES W.R.T STATE

C*********************************************************************

      R2=X*X+Y*Y+Z*Z
      R=SQRT(R2)
      CTH=(A(1,2)*X+A(2,2)*Y+A(3,2)*Z)/R
      CT2=CTH*CTH

      CALL AGEO(2)

      TEMP1=G22*X+G23*A(1,2)
      TEMP2=G22*Y+G23*A(2,2)
      TEMP3=G22*Z+G23*A(3,2)
      TEMP4=G33*A(1,2)+G23*X
      TEMP5=G33*A(2,2)+G23*Y
      TEMP6=G33*A(3,2)+G23*Z

      GXX=G11+TEMP1*X+TEMP4*A(1,2)
      GYX=    TEMP1*Y+TEMP4*A(2,2)
      GZX=    TEMP1*Z+TEMP4*A(3,2)

      GYY=G11+TEMP2*Y+TEMP5*A(2,2)
      GZY=    TEMP2*Z+TEMP5*A(3,2)

      GZZ=G11+TEMP3*Z+TEMP6*A(3,2)

      RTHE=UMF*RE/SQRT(UMF*UMF*(1.-CT2)+CT2)

      UMFC=FLAT*(2.-FLAT)/(RE*RE*UMF*UMF)

      ALT=R-RTHE

      DRTHE=RTHE**3*UMFC*CTH/R2

      PHX=X/R-(X*CTH-R*A(1,2))*DRTHE
      PHY=Y/R-(Y*CTH-R*A(2,2))*DRTHE
      PHZ=Z/R-(Z*CTH-R*A(3,2))*DRTHE

C*********************************************************************
```



```
C     CALCULATE RELATIVE VELOCITY AND PARTIAL DERIVATIVES W.R.T STATE

C*********************************************************************

    2 OMX=A12W
      OMY=A22W
      OMZ=A32W

      TEMP=OMY*Z-OMZ*Y
      TEMP1=OMZ*X-OMX*Z
      TEMP2=OMX*Y-OMY*X

c       Include winds in calculation of relative velocity

      WIND(1)=0.
      WIND(2)=0.
      WIND(3)=0.

      NOWIND=.TRUE.

      IF(NWIND.EQ.0) GO TO 200

      HATMOS=ALT + HBIAS

      IF(HATMOS.LT.ALTTBL(1).OR.HATMOS.GT.ALTTBL(NWIND)) GO TO 200

c       compute partials wrt state of components of the wind vector

      NOWIND=.FALSE.

      TEMP3=TEMP*TEMP+TEMP1*TEMP1+TEMP2*TEMP2

      P1=(TEMP1*OMZ-TEMP2*OMY)/TEMP3
      P2=(TEMP2*OMX-TEMP *OMZ)/TEMP3
      P3=(TEMP *OMY-TEMP1*OMX)/TEMP3

      TEMP4=1./SQRT(TEMP3)

      E1=TEMP *TEMP4
      E2=TEMP1*TEMP4
      E3=TEMP2*TEMP4

      PE1X=-E1*P1
      PE1Y=-E1*P2-OMZ*TEMP4
      PE1Z=-E1*P3+OMY*TEMP4

      PE2X=-E2*P1+OMZ*TEMP4
      PE2Y=-E2*P2
      PE2Z=-E2*P3-OMX*TEMP4

      PE3X=-E3*P1-OMY*TEMP4
      PE3Y=-E3*P2+OMX*TEMP4
      PE3Z=-E3*P3

      TEMP5=1./R
      TEMP6=TEMP5*TEMP5*TEMP5
```

```
      PRXX=TEMP5-X*X*TEMP6
      PRXY=      -X*Y*TEMP6
      PRXZ=      -X*Z*TEMP6

      PRYY=TEMP5-Y*Y*TEMP6
      PRYZ=      -Y*Z*TEMP6
      PRZZ=TEMP5-Z*Z*TEMP6

      CALL SPLINE(2,MBS9,100,HATMOS,ALTTBL,WARG,WTBL,WARGD,2,MBS10,AW,
     *            BW,CW)

      SAZW=-SIN(AZW)
      CAZW=-COS(AZW)

      TEMP4=(E3*Y-E2*Z)*TEMP5
      TEMP3=SAZW*E1+CAZW*TEMP4
      WIND(1)=WMAG*TEMP3

      PW1X=PWMDH*PHX*TEMP3+WMAG*(SAZW*(PE1X-PAZWDH*PHX*TEMP4)+CAZW*
     *      ((PE3X*Y-PE2X*Z)*TEMP5+(E3*PRXY-E2*PRXZ)+PAZWDH*PHX*E1))
      PW1Y=PWMDH*PHY*TEMP3+WMAG*(SAZW*(PE1Y-PAZWDH*PHY*TEMP4)+CAZW*
     *      ((PE3Y*Y-PE2Y*Z)*TEMP5+(E3*PRYY-E2*PRYZ)+PAZWDH*PHY*E1))
      PW1Z=PWMDH*PHZ*TEMP3+WMAG*(SAZW*(PE1Z-PAZWDH*PHZ*TEMP4)+CAZW*
     *      ((PE3Z*Y-PE2Z*Z)*TEMP5+(E3*PRYZ-E2*PRZZ)+PAZWDH*PHZ*E1))

      TEMP4=(E1*Z-E3*X)*TEMP5
      TEMP3=SAZW*E2+CAZW*TEMP4

      WIND(2)=WMAG*TEMP3

      PW2X=PWMDH*PHX*TEMP3+WMAG*(SAZW*(PE2X-PAZWDH*PHX*TEMP4)+CAZW*
     *      ((PE1X*Z-PE3X*X)*TEMP5+(E1*PRXZ-E3*PRXX)+PAZWDH*PHX*E2))
      PW2Y=PWMDH*PHY*TEMP3+WMAG*(SAZW*(PE2Y-PAZWDH*PHY*TEMP4)+CAZW*
     *      ((PE1Y*Z-PE3Y*X)*TEMP5+(E1*PRYZ-E3*PRXY)+PAZWDH*PHY*E2))
      PW2Z=PWMDH*PHZ*TEMP3+WMAG*(SAZW*(PE2Z-PAZWDH*PHZ*TEMP4)+CAZW*
     *      ((PE1Z*Z-PE3Z*X)*TEMP5+(E1*PRZZ-E3*PRXZ)+PAZWDH*PHZ*E2))

      TEMP4=(E2*X-E1*Y)*TEMP5
      TEMP3=SAZW*E3+CAZW*TEMP4

      WIND(3)=WMAG*TEMP3

      PW3X=PWMDH*PHX*TEMP3+WMAG*(SAZW*(PE3X-PAZWDH*PHX*TEMP4)+CAZW*
     *      ((PE2X*X-PE1X*Y)*TEMP5+(E2*PRXX-E1*PRXY)+PAZWDH*PHX*E3))
      PW3Y=PWMDH*PHY*TEMP3+WMAG*(SAZW*(PE3Y-PAZWDH*PHY*TEMP4)+CAZW*
     *      ((PE2Y*X-PE1Y*Y)*TEMP5+(E2*PRXY-E1*PRYY)+PAZWDH*PHY*E3))
      PW3Z=PWMDH*PHZ*TEMP3+WMAG*(SAZW*(PE3Z-PAZWDH*PHZ*TEMP4)+CAZW*
     *      ((PE2Z*X-PE1Z*Y)*TEMP5+(E2*PRXZ-E1*PRYZ)+PAZWDH*PHZ*E3))

  200 SW=W-TEMP-WIND(1)
      SU=U-TEMP1-WIND(2)
      SV=V-TEMP2-WIND(3)

      VR=SQRT(SW*SW+SU*SU+SV*SV)
      TEMP=1./VR
```



```
      PVRX=(SV*OMY-SU*OMZ)*TEMP
      PVRY=(SW*OMZ-SV*OMX)*TEMP
      PVRZ=(SU*OMX-SW*OMY)*TEMP

      IF(.not.NOWIND) then

         PVRX=PVRX+(-SW*PW1X-SU*PW2X-SV*PW3X)*TEMP
         PVRY=PVRY+(-SW*PW1Y-SU*PW2Y-SV*PW3Y)*TEMP
         PVRZ=PVRZ+(-SW*PW1Z-SU*PW2Z-SV*PW3Z)*TEMP

      endif

      PVRW=SW*TEMP
      PVRU=SU*TEMP
      PVRV=SV*TEMP

C*********************************************************************

c       Define atmopheric conditions and partial derivatives

c*********************************************************************

      PAT(1)=ALT + HBIAS

      if(iatmos.eq.0) then

         call rraspl(pat)

         pam=pat(2)
         dpdh=pat(3)

      else

         call atmosphere(pat,atmosn,ier,iatmos)

         pam=pat(2)*10000.
         dpdh=pat(3)*10000.

      endif

      pat(1)=alt

      DRDH=PAT(4)
      DSDH=PAT(5)
      RHO=PAT(6)
      SOS=PAT(9)
      XMU=PAT(10)
      DMUH=PAT(11)

      REYNO=RHO*VR*XLEN/XMU

C*********************************************************************

C     partial derivative of dynamic pressure w.r.t state
```

```
C*********************************************************************

      Q=.5*RHO*VR*VR

      TEMP1=RHO*VR
      TEMP2=.5*VR*VR*DRDH

      PQX=TEMP1*PVRX+TEMP2*PHX
      PQY=TEMP1*PVRY+TEMP2*PHY
      PQZ=TEMP1*PVRZ+TEMP2*PHZ

      PQW=TEMP1*PVRW
      PQU=TEMP1*PVRU
      PQV=TEMP1*PVRV

C*********************************************************************

C     MACH NUMBER AND PARTIAL W.R.T STATE

C*********************************************************************

      XMACH=VR/SOS

      TEMP1=1./SOS
      TEMP2=XMACH*DSDH

      PMX=TEMP1*(PVRX-TEMP2*PHX)
      PMY=TEMP1*(PVRY-TEMP2*PHY)
      PMZ=TEMP1*(PVRZ-TEMP2*PHZ)

      PMW=TEMP1*PVRW
      PMU=TEMP1*PVRU
      PMV=TEMP1*PVRV

C*********************************************************************

C     DETERMINE AERO COEFFICIENTS AND PARTIALS W.R.T MACH NUMBER

C*********************************************************************

      smach=xmach
      IF(XMACH.ge.10.) xmach=10.

      CALL SPLINE(13,MBS1,30,XMACH,AEROD(1,14),AERO,AEROD,AERODD,
     *2,mbs2,AAERO,BAERO,CAERO)

      if(xmach.eq.10.) then

         do i=1,13
            aerodd(i)=0.
         enddo

      endif

      xmach=smach
```



```
C*********************************************************************

C     DETERMINE BASE AXIAL FORCE AND PARTIAL W.R.T ALTITUDE

C     CALCULATE ALTITUDE FOR BASE FORCE TABLE LOOKUP

C*********************************************************************

      DALTB=AEROB(1,1,1)-ALTLS
      HGT=ALT+DALTB

      IF(ITHR.lt.NOBASE) then

         L=LSTGE(ITHR)
         IF(L.EQ.2.AND.TNE(1,ITHR).LT.2.9) L=3

         if(l.eq.1) then
            kk=4
         else
            kk=3
         endif

         CALL SPLINE(kk,MBS3,50,HGT,AEROB(1,1,L),FABT,AEROB(1,2,L),
     *   AERODD(14),2,MBS4,AFAB,BFAB,CFAB)

         if(l.eq.1) then
            pfabh=(pfabh*qref*q-fab*pqrefh)/qref**2
            pfabq=fab/qref
            fab=fab*q/qref
         else
            pfabq=0.
         endif

         FYB=0.
         FLB=0.
         FNMB=0.
         PFYBH=0.
         PFLBH=0.
         PFNMBH=0.

      else

         FAB=CBAXIL
         PFABH=0.
         FNB=0.
         FYB=0.
         FMB=0.
         FLB=0.
         FNMB=0.
         PFNBH=0.
         PFYBH=0.
         PFMBH=0.
         PFLBH=0.
         PFNMBH=0.
         pqrefh=0.
         pfabq=0.
```

```
      endif

      IF(ITOL(ITHR).ne.0) then

         CALL AMULG(2,MBS21,25,XMACH,TOMACH,DCAX,DELCA,DCNP,DELCN)
         CALL AMULG(1,MBS21,25,XMACH,TOMACH,DCMZ,DELCM,0,0)

         CAO=CAO+DCAX
         CNO=CNO+DCNP
         CMO=CMO+DCMZ

         CALL SPLINE(1,MBS30,25,HGT,DELALT,DFAB,DELFAB,DDFAB,2,
     *   MBS31,AF,BF,CF)

         FAB=FAB+DFAB(1)
         PFABH=PFABH+DDFAB(1)

      endif
C*********************************************************************
C     DETERMINE ATTITUDE ANGLES
C*********************************************************************
      CALL ACNTRO

      max_qalp=0

      schip=sin(chip)
      cchip=cos(chip)
      schiy=sin(chiy)
      cchiy=cos(chiy)
      schir=sin(chir)
      cchir=cos(chir)

      xhat(1)= cchip*cchir+schip*schiy*schir
      xhat(2)=-schip*cchir+cchip*schiy*schir
      xhat(3)=-cchiy*schir

      yhat(1)= schip*cchiy
      yhat(2)= cchip*cchiy
      yhat(3)= schiy

      zhat(1)= cchip*schir-schip*schiy*cchir
      zhat(2)=-schip*schir-cchip*schiy*cchir
      zhat(3)= cchiy*cchir
C*********************************************************************
C     PARTIAL OF DIRECTION COSINES W.R.T CONTROL
C*********************************************************************
      px1cp= xhat(2)
```



```
      px2cp=-xhat(1)
      px3cp= 0.

      py1cp= yhat(2)
      py2cp=-yhat(1)
      py3cp= 0.

      pz1cp= zhat(2)
      pz2cp=-zhat(1)
      pz3cp= 0.

C       calculate second order partials wrt pitch angle

      px1cpp=-xhat(1)
      px2cpp=-xhat(2)
      px3cpp= 0.

      py1cpp=-yhat(1)
      py2cpp=-yhat(2)
      py3cpp= 0.

      pz1cpp=-zhat(1)
      pz2cpp=-zhat(2)
      pz3cpp= 0.

      IF(.not.PONLY) then

         px1cy=schip*cchiy*schir
         px2cy=cchip*cchiy*schir
         px3cy=schiy*schir

         py1cy=-schip*schiy
         py2cy=-cchip*schiy
         py3cy= cchiy

         pz1cy=-schip*cchiy*cchir
         pz2cy=-cchip*cchiy*cchir
         pz3cy=-schiy*cchir

C       calculate second order partials wrt yaw angle

         px1cyy=-schip*schiy*schir
         px2cyy=-cchip*schiy*schir
         px3cyy= cchiy*schir

         py1cyy=-schip*cchiy
         py2cyy=-cchip*cchiy
         py3cyy=-schiy

         pz1cyy= schip*schiy*cchir
         pz2cyy= cchip*schiy*cchir
         pz3cyy=-cchiy*cchir

C       calculate second order partials wrt pitch and yaw angles

         px1cpy= cchip*cchiy*schir
```

```
         px2cpy=-schip*cchiy*schir
         px3cpy= 0.

         py1cpy=-cchip*schiy
         py2cpy= schip*schiy
         py3cpy= 0.

         pz1cpy=-cchip*cchiy*cchir
         pz2cpy= schip*cchiy*cchir
         pz3cpy= 0.

      endif

C*********************************************************************

C     COMPUTE ANGLE OF ATTACK AND PARTIAL W.R.T STATE

C*********************************************************************

      salp=(xhat(1)*sw+xhat(2)*su+xhat(3)*sv)/vr
      calp=(yhat(1)*sw+yhat(2)*su+yhat(3)*sv)/vr

      alp=atan2(salp,calp)
      qalpha=abs(q*alp)

      xnu1=0.

      if(qalpha.gt.qalpha_max.and.qalpha_max.gt.0.) then

         olda=alp
         alp_new=qalpha_max/q*sign(1.d0,alp)

         colda=cos(olda)
         solda=sin(olda)
         salp_new=sin(alp_new)
         calp_new=cos(alp_new)

         temp1=calp_new*colda+salp_new*solda
         temp2=salp_new*colda-calp_new*solda

         do i=1,3
            xhat_new(i)= xhat(i)*temp1+yhat(i)*temp2
            yhat_new(i)=-xhat(i)*temp2+yhat(i)*temp1
         enddo

         temp1=-salp_new*colda+calp_new*solda
         temp2= calp_new*colda+salp_new*solda

         do i=1,3
            pxhat_alp(i)= xhat(i)*temp1+yhat(i)*temp2
            pyhat_alp(i)=-xhat(i)*temp2+yhat(i)*temp1
         enddo

         max_qalp=1

       endif
```



```
      temp=1./vr

      pax=temp*(calp*(-xhat(2)*omz+xhat(3)*omy)-salp*
     *     (-yhat(2)*omz+yhat(3)*omy))
      pay=temp*(calp*( xhat(1)*omz-xhat(3)*omx)-salp*
     *     ( yhat(1)*omz-yhat(3)*omx))
      paz=temp*(calp*(-xhat(1)*omy+xhat(2)*omx)-salp*
     *     (-yhat(1)*omy+yhat(2)*omx))

      temp5=calp*xhat(1)-salp*yhat(1)
      temp6=calp*xhat(2)-salp*yhat(2)
      temp7=calp*xhat(3)-salp*yhat(3)

      if(.not.nowind) then
         pax=pax-temp*(temp5*pw1x+temp6*pw2x+temp7*pw3x)
         pay=pay-temp*(temp5*pw1y+temp6*pw2y+temp7*pw3y)
         paz=paz-temp*(temp5*pw1z+temp6*pw2z+temp7*pw3z)
       endif

      paw=temp*temp5
      pau=temp*temp6
      pav=temp*temp7

      pacp=temp*(calp*(px1cp*sw+px2cp*su+px3cp*sv)-salp*
     *      (py1cp*sw+py2cp*su+py3cp*sv))

      pacpcp=temp*(-pacp*(salp*(px1cp*sw+px2cp*su+px3cp*sv)+
     *       calp*(py1cp*sw+py2cp*su+py3cp*sv))
     *       +calp*(px1cpp*sw+px2cpp*su+px3cpp*sv)
     *       -salp*(py1cpp*sw+py2cpp*su+py3cpp*sv))

      if(.not.ponly) then

         pacy=temp*(calp*(px1cy*sw+px2cy*su+px3cy*sv)-
     *        salp*(py1cy*sw+py2cy*su+py3cy*sv))

         pacycy=temp*(-pacy*(salp*(px1cy*sw+px2cy*su+
     *          px3cy*sv)+calp*(py1cy*sw+py2cy*su+py3cy*sv))
     *          +calp*(px1cyy*sw+px2cyy*su+px3cyy*sv)
     *          -salp*(py1cyy*sw+py2cyy*su+py3cyy*sv))

         pacpcy=temp*(-pacy*(salp*(px1cp*sw+px2cp*su+
     *          px3cp*sv)+calp*(py1cp*sw+py2cp*su+py3cp*sv))
     *          +calp*(px1cpy*sw+px2cpy*su+px3cpy*sv)
     *          -salp*(py1cpy*sw+py2cpy*su+py3cpy*sv))

       endif

C*********************************************************************

C     CALCULATE SIDESLIP ANGLE AND PARTIALS

C*********************************************************************

      TEMP1=(ZHAT(1)*SW+ZHAT(2)*SU+ZHAT(3)*SV)/VR
```

```
      ALFY=ASIN(TEMP1)

      TEMP2=1./(VR*COS(ALFY))

      PBX=TEMP2*(-ZHAT(2)*OMZ+ZHAT(3)*OMY-TEMP1*PVRX)
      PBY=TEMP2*(-ZHAT(3)*OMX+ZHAT(1)*OMZ-TEMP1*PVRY)
      PBZ=TEMP2*(-ZHAT(1)*OMY+ZHAT(2)*OMX-TEMP1*PVRZ)

      IF(.not.NOWIND) then

         PBX=PBX-TEMP2*(ZHAT(1)*PW1X+ZHAT(2)*PW2X+ZHAT(3)*PW3X)
         PBY=PBY-TEMP2*(ZHAT(1)*PW1Y+ZHAT(2)*PW2Y+ZHAT(3)*PW3Y)
         PBZ=PBZ-TEMP2*(ZHAT(1)*PW1Z+ZHAT(2)*PW2Z+ZHAT(3)*PW3Z)

      endif

      PBW=TEMP2*(ZHAT(1)-TEMP1*PVRW)
      PBU=TEMP2*(ZHAT(2)-TEMP1*PVRU)
      PBV=TEMP2*(ZHAT(3)-TEMP1*PVRV)

      PBCP=TEMP2*(SW*ZHAT(2)-SU*ZHAT(1))

      PBCPCP=TAN(ALFY)*PBCP*PBCP-TEMP2*(ZHAT(1)*SW+ZHAT(2)*SU)

      IF(.not.PONLY) then

         PBCY=TEMP2*(-CCHIR*(CCHIY*(SW*SCHIP+SU*CCHIP)+SV*SCHIY))

         PBCYCY=TAN(ALFY)*PBCY*PBCY-TEMP2*(ZHAT(1)*SW+ZHAT(2)*SU+
     *           ZHAT(3)*SV)

         PBCPCY=TAN(ALFY)*PBCP*PBCY-TEMP2*CCHIY*(XHAT(1)*SW+
     *           XHAT(2)*SU)

      endif

      DELTE=0.
      PDELM=0.

      CNDELE=0.
      CADELE=0.
      CMDELE=0.

      PCADM=0.
      PCNDM=0.
      PCMDM=0.

C*********************************************************************

C     PARTIAL OF AERO COEFFICIENTS W.R.T STATE

C*********************************************************************

      SQ=S(ITHR)*Q
```



```
c     VISC=XMACH.GT.10.
      visc=.false.

      IF(VISC) then

         CAO=0.
         CAALP=0.
         CNO=0.
         CNALP=0.
         CMO=0.
         CMALP=0.
         cyo=0.
         cybeta=0.

         DO I=1,13
            AERODD(I)=0.
         enddo

      endif

      IF (FAB .EQ. 0.0 .OR. SQ .EQ. 0.0 ) THEN

         FABDSQ = 0.0

      ELSE

         FABDSQ = FAB/SQ

      ENDIF

      IF (FNB .EQ. 0.0 .OR. SQ .EQ. 0.0 )THEN

         FNBDQS = 0.0

      ELSE

         FNBDSQ = FNB/SQ

      ENDIF

      CA=CAO+CAALP*ALP+CADELE*DELTE+FABDSQ
      CN=CNO+CNALP*ALP+CNDELE*DELTE+FNBDSQ

      if(max_qalp.eq.1) then
         CA_new=CAO+CAALP*ALP_new+CADELE*DELTE+FABDSQ
         CN_new=CNO+CNALP*ALP_new+CNDELE*DELTE+FNBDSQ
      endif

      TEMP1=PCNOM+ALP*PCNAM

      IF (FAB .EQ. 0.0 ) THEN

         FABDQ = 0.0

      ELSE
```

```
         FABDQ = FAB/Q

       ENDIF

       IF (FNB .EQ. 0.0 ) THEN

          FNBDQ = 0.0

       ELSE

          FNBDQ = FNB/Q

       ENDIF

       IF ( SQ .EQ. 0.0 ) THEN

           P1 = 0.0
           P2 = 0.0
           P3 = 0.0
           P4 = 0.0
           P5  = 0.0
           P6  = 0.0
           P7  = 0.0
           P8  = 0.0
           P9  = 0.0
           P10 = 0.0
           P11 = 0.0
           P12 = 0.0

       ELSE

           P1  = (PFNBH*PHX-FNBDQ*PQX)/SQ
           P2  = (PFNBH*PHY-FNBDQ*PQY)/SQ
           P3  = (PFNBH*PHZ-FNBDQ*PQZ)/SQ
           P4  =           (-FNBDQ*PQW)/SQ
           P5  =           (-FNBDQ*PQU)/SQ
           P6  =           (-FNBDQ*PQV)/SQ
           P7  = (PFABH*PHX-(FABDQ-pfabq)*PQX)/SQ
           P8  = (PFABH*PHY-(FABDQ-pfabq)*PQY)/SQ
           P9  = (PFABH*PHZ-(FABDQ-pfabq)*PQZ)/SQ
           P10 =           (-FABDQ*PQW)/SQ
           P11 =           (-FABDQ*PQU)/SQ
           P12 =           (-FABDQ*PQV)/SQ

       ENDIF

       PCNX=TEMP1*PMX+CNALP*PAX+P1
       PCNY=TEMP1*PMY+CNALP*PAY+P2
       PCNZ=TEMP1*PMZ+CNALP*PAZ+P3

       PCNW=TEMP1*PMW+CNALP*PAW+P4
       PCNU=TEMP1*PMU+CNALP*PAU+P5
       PCNV=TEMP1*PMV+CNALP*PAV+P6

       TEMP1=PCAOM+ALP*PCAAM
```



```
       PCAX=TEMP1*PMX+CAALP*PAX+P7
       PCAY=TEMP1*PMY+CAALP*PAY+P8
       PCAZ=TEMP1*PMZ+CAALP*PAZ+P9

       PCAW=TEMP1*PMW+CAALP*PAW+P10
       PCAU=TEMP1*PMU+CAALP*PAU+P11
       PCAV=TEMP1*PMV+CAALP*PAV+P12

C*********************************************************************

C      VISCOUS DRAG CORRECTIONS WHEN MACH GT 10.0

C*********************************************************************

       SREYNO=SQRT(REYNO)

       IF(.NOT.VISC) GO TO 60

       CALL SPLINE(18,MBS18,10,XMACH,VAEROD(1,19),VAERO,VAEROD,VAERDD,2,
      *MBS19,VA,VB,VC)

C*********************************************************************

C      PARTIAL OF CONTINUUM FLOW COEFFICIENTS W.R.T STATE

C*********************************************************************

       ALPHA=ALP*RAD

       CACF=ACACF+(BCACF+CCACF*ALPHA)*ALPHA

       TEMP1= VAERDD(1)+(VAERDD(2)+VAERDD(3)*ALPHA)*ALPHA
       TEMP2=BCACF+2.*CCACF*ALPHA

       PCACFX=TEMP1*PMX+TEMP2*PAX*RAD
       PCACFY=TEMP1*PMY+TEMP2*PAY*RAD
       PCACFZ=TEMP1*PMZ+TEMP2*PAZ*RAD

       PCACFW=TEMP1*PMW+TEMP2*PAW*RAD
       PCACFU=TEMP1*PMU+TEMP2*PAU*RAD
       PCACFV=TEMP1*PMV+TEMP2*PAV*RAD

       CNCF=ACNCF+(BCNCF+CCNCF*ALPHA)*ALPHA

       TEMP1=VAERDD(4)+(VAERDD(5)+VAERDD(6)*ALPHA)*ALPHA
       TEMP2=BCNCF+2.*CCNCF*ALPHA

       PCNCFX=TEMP1*PMX+TEMP2*PAX*RAD
       PCNCFY=TEMP1*PMY+TEMP2*PAY*RAD
       PCNCFZ=TEMP1*PMZ+TEMP2*PAZ*RAD

       PCNCFW=TEMP1*PMW+TEMP2*PAW*RAD
       PCNCFU=TEMP1*PMU+TEMP2*PAU*RAD
       PCNCFV=TEMP1*PMV+TEMP2*PAV*RAD

       CMCF=ACMCF+(BCMCF+CCMCF*ALPHA)*ALPHA
```

```
      TEMP1=VAERDD(7)+(VAERDD(8)+VAERDD(9)*ALPHA)*ALPHA
      TEMP2=BCMCF+2.*CCMCF*ALPHA

      PCMCFX=TEMP1*PMX+TEMP2*PAX*RAD
      PCMCFY=TEMP1*PMY+TEMP2*PAY*RAD
      PCMCFZ=TEMP1*PMZ+TEMP2*PAZ*RAD

      PCMCFW=TEMP1*PMW+TEMP2*PAW*RAD
      PCMCFU=TEMP1*PMU+TEMP2*PAU*RAD
      PCMCFV=TEMP1*PMV+TEMP2*PAV*RAD

C*********************************************************************

C     PARTIAL OF MOLECULAR FLOW COEFFICIENTS W.R.T STATE

C*********************************************************************

      CAMF=ACAMF+(BCAMF+CCAMF*ALPHA)*ALPHA

      TEMP1=VAERDD(10)+(VAERDD(11)+VAERDD(12)*ALPHA)*ALPHA
      TEMP2=BCAMF+2.*CCAMF*ALPHA

      PCAMFX=TEMP1*PMX+TEMP2*PAX*RAD
      PCAMFY=TEMP1*PMY+TEMP2*PAY*RAD
      PCAMFZ=TEMP1*PMZ+TEMP2*PAZ*RAD

      PCAMFW=TEMP1*PMW+TEMP2*PAW*RAD
      PCAMFU=TEMP1*PMU+TEMP2*PAU*RAD
      PCAMFV=TEMP1*PMV+TEMP2*PAV*RAD

      CNMF=ACNMF+(BCNMF+CCNMF*ALPHA)*ALPHA

      TEMP1=VAERDD(13)+(VAERDD(14)+VAERDD(15)*ALPHA)*ALPHA
      TEMP2=BCNMF+2.*CCNMF*ALPHA

      PCNMFX=TEMP1*PMX+TEMP2*PAX*RAD
      PCNMFY=TEMP1*PMY+TEMP2*PAY*RAD
      PCNMFZ=TEMP1*PMZ+TEMP2*PAZ*RAD

      PCNMFW=TEMP1*PMW+TEMP2*PAW*RAD
      PCNMFU=TEMP1*PMU+TEMP2*PAU*RAD
      PCNMFV=TEMP1*PMV+TEMP2*PAV*RAD

      CMMF=ACMMF+(BCMMF+CCMMF*ALPHA)*ALPHA

      TEMP1=VAERDD(16)+(VAERDD(17)+VAERDD(18)*ALPHA)*ALPHA
      TEMP2=BCMMF+2.*CCMMF*ALPHA

      PCMMFX=TEMP1*PMX+TEMP2*PAX*RAD
      PCMMFY=TEMP1*PMY+TEMP2*PAY*RAD
      PCMMFZ=TEMP1*PMZ+TEMP2*PAZ*RAD

      PCMMFW=TEMP1*PMW+TEMP2*PAW*RAD
      PCMMFU=TEMP1*PMU+TEMP2*PAU*RAD
      PCMMFV=TEMP1*PMV+TEMP2*PAV*RAD
```



```
C*********************************************************************

C     PARTIAL OF KNUDSEN NUMBER W.R.T STATE

C*********************************************************************

      XLAMDA=2.332353E-05*PAT(12)/PAM

      XKNUD=.1407*XLAMDA/VR*(PAM/RHO)**.5

      TEMP=DPDH/PAM-DRDH/RHO+2.*2.3323530E-08*PAT(13)/XLAMDA

      PKNUDX=XKNUD*(.5*TEMP*PHX-PVRX/VR)
      PKNUDY=XKNUD*(.5*TEMP*PHY-PVRY/VR)
      PKNUDZ=XKNUD*(.5*TEMP*PHZ-PVRZ/VR)

      PKNUDW=XKNUD*(-PVRW/VR)
      PKNUDU=XKNUD*(-PVRU/VR)
      PKNUDV=XKNUD*(-PVRV/VR)

      IF(XKNUD.lt..01.or.XKNUD.gt.10.) then

         PKNUDX=0.
         PKNUDY=0.
         PKNUDZ=0.
         PKNUDW=0.
         PKNUDU=0.
         PKNUDV=0.

      endif

      IF(XKNUD.LT..01) XKNUD=.01
      IF(XKNUD.GT.10.) XKNUD=10.

      TEMP1=SIN(9.009*(XKNUD-.01)/RAD)
      TEMP2=COS(9.009*(XKNUD-.01)/RAD)*(CAMF-CACF)*9.009/RAD

      PCAX=(PCAMFX-PCACFX)*TEMP1+TEMP2*PKNUDX+PCACFX+PCAX
      PCAY=(PCAMFY-PCACFY)*TEMP1+TEMP2*PKNUDY+PCACFY+PCAY
      PCAZ=(PCAMFZ-PCACFZ)*TEMP1+TEMP2*PKNUDZ+PCACFZ+PCAZ

      PCAW=(PCAMFW-PCACFW)*TEMP1+TEMP2*PKNUDW+PCACFW+PCAW
      PCAU=(PCAMFU-PCACFU)*TEMP1+TEMP2*PKNUDU+PCACFU+PCAU
      PCAV=(PCAMFV-PCACFV)*TEMP1+TEMP2*PKNUDV+PCACFV+PCAV

      TEMP2=COS(9.009*(XKNUD-.01)/RAD)*(CNMF-CNCF)*9.009/RAD

      PCNX=(PCNMFX-PCNCFX)*TEMP1+TEMP2*PKNUDX+PCNCFX+PCNX
      PCNY=(PCNMFY-PCNCFY)*TEMP1+TEMP2*PKNUDY+PCNCFY+PCNY
      PCNZ=(PCNMFZ-PCNCFZ)*TEMP1+TEMP2*PKNUDZ+PCNCFZ+PCNZ

      PCNW=(PCNMFW-PCNCFW)*TEMP1+TEMP2*PKNUDW+PCNCFW+PCNW
      PCNU=(PCNMFU-PCNCFU)*TEMP1+TEMP2*PKNUDU+PCNCFU+PCNU
      PCNV=(PCNMFV-PCNCFV)*TEMP1+TEMP2*PKNUDV+PCNCFV+PCNV
```

```
      TEMP2=COS(9.009*(XKNUD-.01)/RAD)*(CMMF-CMCF)*9.009/RAD

      PCMX=(PCMMFX-PCMCFX)*TEMP1+TEMP2*PKNUDX+PCMCFX
      PCMY=(PCMMFY-PCMCFY)*TEMP1+TEMP2*PKNUDY+PCMCFY
      PCMZ=(PCMMFZ-PCMCFZ)*TEMP1+TEMP2*PKNUDZ+PCMCFZ

      PCMW=(PCMMFW-PCMCFW)*TEMP1+TEMP2*PKNUDW+PCMCFW
      PCMU=(PCMMFU-PCMCFU)*TEMP1+TEMP2*PKNUDU+PCMCFU
      PCMV=(PCMMFV-PCMCFV)*TEMP1+TEMP2*PKNUDV+PCMCFV

C*********************************************************************

C     CORRECTIONS TO CA AND CN

C*********************************************************************

      CA=(CAMF-CACF)*TEMP1+CACF+CA
      CN=(CNMF-CNCF)*TEMP1+CNCF+CN

   60 CONTINUE

C*********************************************************************

c       Side force

C*********************************************************************

      CS=CYO+CYBETA*ALFY

      TEMP4=PCYOM+PCYBM*ALFY

      PCSX=TEMP4*PMX+CYBETA*PBX
      PCSY=TEMP4*PMY+CYBETA*PBY
      PCSZ=TEMP4*PMZ+CYBETA*PBZ

      PCSW=TEMP4*PMW+CYBETA*PBW
      PCSU=TEMP4*PMU+CYBETA*PBU
      PCSV=TEMP4*PMV+CYBETA*PBV

C*********************************************************************

C     AERO MOMENTS

C*********************************************************************
      if(mombal) then
         IF(.NOT.MPFLAG) THEN
            delwt=wint-xm
            if(ithr.gt.noevnt(1)) DELWT=OLOW-XM

            IF(DELWT.LT..01) DELWT=.01

            CALL SPLINE(2,MBS7,15,DELWT,TBDWT,CG,CGBL,PCG,2,MBS8,ACG,
     *      BCG,CCG)

            PXCGM=-PXCGM
            PYCGM=-PYCGM
```



```
         ELSE
            FTTM=.3048
            EVENTNUM=ITHR
            MOBOOS=PROP(2)-wgtboost*PFKG
            MOMAIN=PROP(1)-wgtmps*PFKG
            BOMDOT=wdotboost*PFKG
            MNMDOT=wdotmps*PFKG
            CALCPAR=.TRUE.
            CALL MASSPRO(EVENTNUM,MOBOOS,MOMAIN,BOMDOT,MNMDOT,LONCG,
     &              LATCG,NORCG,CALCPAR,LONCGP,LATCGP,NORCGP)
            CG(1)=-(NORCG/12.)*FTTM
            CG(2)=-(LONCG/12.)*FTTM
            PCG(1)=-(NORCGP/12.)*FTTM/PFKG
            PCG(2)=-(LONCGP/12.)*FTTM/PFKG
         ENDIF
       else
          pxcgm=0.
          pycgm=0.
       endif

C*********************************************************************

C     FIXED ENGINE FORCES AND MOMENTS

C*********************************************************************

      PTMZX=0.
      PTMZY=0.
      PTMZZ=0.
      PTMZM=0.

      TFXX=0.
      TFYY=0.
      TFZZ=0.

      PFXX=0.
      PFXY=0.
      PFXZ=0.

      PFYX=0.
      PFYY=0.
      PFZY=0.

      PFZX=0.
      PFZY=0.
      PFZZ=0.

      ptxfm=0.
      ptyfm=0.
      ptzfm=0.

      TMZ=0.

      DO 500 N=1,6

      IF(TNE(N,ITHR).LT..001.or.n.eq.2) GO TO 500
```

```
      if(n.eq.1.and.mombal) go to 500

      MMIN=0

      DO 400 I=1,15

      IF(NSYST(I).eq.N.and.engines(i,ithr).eq.1) then

         IF(MMIN.EQ.0) MMIN=I
         MMAX=I

      endif

  400 CONTINUE

      if(n.eq.1.and..not.mombal) then

         THRUST=TNE(1,ITHR)*(THROT(ITHR)*ENGDAT(6,N)*PTN-
     *          ENGDAT(8,N)*PAM)
         PTM=0.
         tau=throt(ithr)

         if(mpsflg(ithr).eq.6) then

            m1=1
            call amulg(1,m1,15,t-time(1,ithr),timemps,tau2,taumps,0,0)

            thr1=throt(ithr)*engdat(6,n)*ptn

            thr2=tau2*engdat(6,n)*ptn

            thrust=0.

            do i=1,15

               if(nsyst(i).eq.1.and.engines(i,ithr).eq.1) then

                  if(mpsinp(i).eq.0) then
                     thrust=thrust+thr1-engdat(8,n)*pam
                  else
                     thrust=thrust+thr2-engdat(8,n)*pam
                  endif
               endif
            enddo

            ptm=0.

         endif

         IF(NBTRG5.eq.2) then

            if(mpsflg(ithr).eq.4) then

               PTM=GLIM(ITHR)*GZERO
               GM=XM*PTM
```



```
               TAU=GM/(TNE(1,ITHR)*ENGDAT(6,N)*PTN)

            endif

            if(mpsflg(ithr).eq.5) then

               tau=tau_const(jthrot(ithr),nstage)

            endif

            THRUST=TNE(1,ITHR)*(TAU*ENGDAT(6,N)*PTN-ENGDAT(8,N)*PAM)

         endif

C*****************************************************************

C     LBM SYSTEM

C*****************************************************************

      else if(n.eq.3) then

         CALL AMULG(1,MBLB1,30,T,TIMLBM,THRVAC,TLBM(1,1),0,0)

         THRUST=THRVAC*PTN-ENGDAT(8,MMIN)*PAM

C*****************************************************************

C     OMS SYSTEM

C*****************************************************************

      else if(n.eq.4) then

         THRUST=FOMS(ITHR)*PTN-ENGDAT(8,MMIN)*PAM

C*****************************************************************

C     RCS SYSTEM

C*****************************************************************

      else if(n.eq.5) then

         THRUST=FRCS(ITHR)*PTN-ENGDAT(8,MMIN)*PAM

C*****************************************************************

C     Fixed mps engines

C*****************************************************************

      else if(n.eq.6) then

         THRUST=TNE(6,ITHR)*(THROT(ITHR)*ENGDAT(6,mmin)*PTN-
     *   ENGDAT(8,mmin)*PAM)
```

```
         PTM=0.

         tau=throt(ithr)

         IF(NBTRG5.eq.2) then

            if(mpsflg(ithr).eq.4) then

               PTM=GLIM(ITHR)*GZERO
               GM=XM*PTM
               TAU=GM/((TNE(1,ITHR)+tne(6,ithr))*ENGDAT(6,mmin)*PTN)

            endif

            if(mpsflg(ithr).eq.5) then

               tau=tau_const(jthrot(ithr),nstage)

            endif

            THRUST-TNE(6,ITHR)*(TAU*ENGDAT(6,mmin)*PTN-
     *      ENGDAT(8,mmin)*PAM)

         endif

      endif

      IF(INT(TNE(N,ITHR)+.1).GT.MMAX-MMIN+1) MMAX=INT(TNE(N,ITHR)+.1)+
     *          MMIN-1

      DO 460 I=1,15

      if(nsyst(i).ne.n.or.engines(i,ithr).eq.0) go to 460

      if(n.ne.6) then

         TANP=TAN(ENGDAT(4,I)/RAD)
         TANY=TAN(ENGDAT(5,I)/RAD)

      else

         tanp=0.
         if(mombal) tanp=-(xcg-engdat(3,i))/(ycg-engdat(1,i))
         tany=0.

c        partials of fixed thrust wrt mass for the current configuration
c        are omitted since they are zero

      endif

      TEMP=1./SQRT(1.+TANP*TANP+TANY*TANY)

      TFX=-THRUST*TEMP*TANP/tne(n,ithr)

      PTXXT=ENGDAT(8,I)*TEMP*TANP*DPDH*PHX
```



```
      PTXYT=ENGDAT(8,I)*TEMP*TANP*DPDH*PHY
      PTXZT=ENGDAT(8,I)*TEMP*TANP*DPDH*PHZ

      TFY=THRUST*TEMP/tne(n,ithr)

      PTYXT=-ENGDAT(8,I)*TEMP*DPDH*PHX
      PTYYT=-ENGDAT(8,I)*TEMP*DPDH*PHY
      PTYZT=-ENGDAT(8,I)*TEMP*DPDH*PHZ

      TFZZ=THRUST*TEMP*TANY/tne(n,ithr)+TFZZ

      ptxfm=-temp*tanp*ptm+ptxfm
      ptyfm=temp*ptm+ptyfm
      ptzfm=temp*tany*ptm+ptzfm

      PFZX=-ENGDAT(8,I)*TEMP*TANY*DPDH*PHX+PFZX
      PFZY=-ENGDAT(8,I)*TEMP*TANY*DPDH*PHY+PFZY
      PFZZ=-ENGDAT(8,I)*TEMP*TANY*DPDH*PHZ+PFZZ

      if(mombal) then

         TMZ=-TFY*(XCG-ENGDAT(3,I))+TFX*(YCG-ENGDAT(1,I))+TMZ

         PTMZX=-PTYXT*(XCG-ENGDAT(3,I))+PTXXT*(YCG-ENGDAT(1,I))+PTMZX
         PTMZY=-PTYYT*(XCG-ENGDAT(3,I))+PTXYT*(YCG-ENGDAT(1,I))+PTMZY
         PTMZZ=-PTYZT*(XCG-ENGDAT(3,I))+PTXZT*(YCG-ENGDAT(1,I))+PTMZZ
      else
         tmz=0.
         ptmzx=0.
         ptmzy=0.
         ptmzz=0.
      endif

      PFXX=PTXXT+PFXX
      PFXY=PTXYT+PFXY
      PFXZ=PTXZT+PFXZ

      PFYX=PTYXT+PFYX
      PFYY=PTYYT+PFYY
      PFYZ=PTYZT+PFYZ

      if(mombal) then
         PTMZM=-TFY*PXCGM+TFX*PYCGM+PTMZM
      else
         ptmzm=0.
      endif

      TFXX=TFX+TFXX
      TFYY=TFY+TFYY

  460 CONTINUE

  500 CONTINUE

c*************************************************************************
```

```
      if(mombal) then
         do i=1,15
            IF(NSYST(I).EQ.1.and.engines(i,ithr).eq.1) GO TO 27
         enddo
      endif

      tandy=0.
      crho=0.
      srho=0.
      txp=0.
      typ=0.
      ty1=0.
      ty2=0.
      temp1=1.
      temp2=1.
      tandp1=0.
      tandp2=0.
      ptxcp=0.
      ptxcpp=0.
      ptxcy=0.
      ptxcyy=0.
      ptxcpy=0.
      ptycp=0.
      ptycpp=0.
      ptycy=0.
      ptycyy=0.
      ptycpy=0.
      ptzcp=0.
      ptzcpp=0.
      ptzcy=0.
      ptzcyy=0.
      ptzcpy=0.
      txp_new=0.
      typ_new=0.
      ptxp_alp=0.
      ptyp_alp=0.

      go to 510

   27 N=I

      if(cp_input) then
         dxpr=0.
         dypr=(ycg-ycp)/xlen
         pdyprm=1./xlen*(ycg-pycpm)
      else
         DXPR=(XCG-XREF)/XLEN
         DYPR=(YCG-YREF)/XLEN
         pdyprm=0.
      endif

      AMX=CNBO+CNBETA*ALFY-CS*DYPR
      AMY=CLO+CLBETA*ALFY+CS*DXPR
      AMZ=CMO+CMALP*ALP+CA*DXPR-CN*DYPR

      if(max_qalp.eq.1) then
```



```
         amz_new=cmo+cmalp*alp_new+ca_new*dxpr-cn_new*dypr
      endif
C*********************************************************************

C     VISCIOUS EFFECTS TO PITCH MOMENTS ARE INCLUDED

C*********************************************************************

      IF(VISC) AMZ=(CMMF-CMCF)*TEMP1+CMCF+CA*DXPR-CN*DYPR

      EXX=PCNBOM+PCNBM*ALFY-cs*pdyprm
      EYX=PCLOM+PCLBM*ALFY
      EZX=PCMOM+PCMAM*ALP-cn*pdyprm

      EZY=CMALP+CAALP*DXPR-CNALP*DYPR

      IF(VISC) then

         TEMP2=(BCAMF+2.*CCAMF*ALPHA)*RAD
         TEMP3=(BCACF+2.*CCACF*ALPHA)*RAD

         TEM1CA=TEMP1*(TEMP2-TEMP3)+TEMP3
         TEM2CA=(2.*(CCAMF-CCACF)*TEMP1+2.*CCACF)*RAD*RAD

         TEMP2=(BCNMF+2.*CCNMF*ALPHA)*RAD
         TEMP3=(BCNCF+2.*CCNCF*ALPHA)*RAD

         TEM1CN=TEMP1*(TEMP2-TEMP3)+TEMP3
         TEM2CN=(TEMP1*2.*(CCAMF-CCACF)+2.*CCACF)*RAD*RAD

         TEMP2=(BCMMF+2.*CCMMF*ALPHA)*RAD
         TEMP3=(BCMCF+2.*CCMCF*ALPHA)*RAD

         TEM1CM=TEMP1*(TEMP2-TEMP3)+TEMP3
         TEM2CM=(TEMP1*2.*(CCMMF-CCMCF)+2.*CCMCF)*RAD*RAD

         EZY=TEM1CM+TEM1CA*DXPR-TEM1CN*DYPR
         EZYY=TEM2CM+TEM2CA*DXPR-TEM2CN*DYPR

      endif

      TEMP2=CNBETA
      TEMP3=CLBETA

      EXZ=TEMP2-CYBETA*DYPR
      EYZ=TEMP2+CYBETA*DXPR

      TEMP1=XLEN*S(ITHR)

      PMAXX=TEMP1*(PQX*AMX+Q*(EXX*PMX+TEMP2*PBX-PCSX*DYPR))+PFNMBH*PHX*
     *      ALFY
      PMAYX=TEMP1*(PQX*AMY+Q*(EYX*PMX+TEMP3*PBX+PCSX*DXPR))+PFLBH*PHX*
     *      ALFY
      PMAZX=TEMP1*(PQX*AMZ+Q*(EZX*PMX+CMALP*PAX+PCAX*DXPR-PCNX*DYPR))+
     *      PFMBH*PHX
```

```
      PMAXY=TEMP1*(PQY*AMX+Q*(EXX*PMY+TEMP2*PBY-PCSY*DYPR))+PFNMBH*PHY*
     *       ALFY
      PMAYY=TEMP1*(PQY*AMY+Q*(EYX*PMY+TEMP3*PBY+PCSY*DXPR))+PFLBH*PHY*
     *       ALFY
      PMAZY=TEMP1*(PQY*AMZ+Q*(EZX*PMY+CMALP*PAY+PCAY*DXPR-PCNY*DYPR))+
     *       PFMBH*PHY

      PMAXZ=TEMP1*(PQZ*AMX+Q*(EXX*PMZ+TEMP2*PBZ-PCSZ*DYPR))+PFNMBH*PHZ*
     *       ALFY
      PMAYZ=TEMP1*(PQZ*AMY+Q*(EYX*PMZ+TEMP3*PBZ+PCSZ*DXPR))+PFLBH*PHZ*
     *       ALFY
      PMAZZ=TEMP1*(PQZ*AMZ+Q*(EZX*PMZ+CMALP*PAZ+PCAZ*DXPR-PCNZ*DYPR))+
     *       PFMBH*PHZ

      PMAXW=TEMP1*(PQW*AMX+Q*(EXX*PMW+TEMP2*PBW-PCSW*DYPR))
      PMAYW=TEMP1*(PQW*AMY+Q*(EYX*PMW+TEMP3*PBW+PCSW*DXPR))
      PMAZW=TEMP1*(PQW*AMZ+Q*(EZX*PMW+CMALP*PAW+PCAW*DXPR-PCNW*DYPR))

      PMAXU=TEMP1*(PQU*AMX+Q*(EXX*PMU+TEMP2*PBU-PCSU*DYPR))
      PMAYU=TEMP1*(PQU*AMY+Q*(EYX*PMU+TEMP3*PBU+PCSU*DXPR))
      PMAZU=TEMP1*(PQU*AMZ+Q*(EZX*PMU+CMALP*PAU+PCAU*DXPR-PCNU*DYPR))

      PMAXV=TEMP1*(PQV*AMX+Q*(EXX*PMV+TEMP2*PBV-PCSV*DYPR))
      PMAYV=TEMP1*(PQV*AMY+Q*(EYX*PMV+TEMP3*PBV+PCSV*DXPR))
      PMAZV=TEMP1*(PQV*AMZ+Q*(EZX*PMV+CMALP*PAV+PCAV*DXPR-PCNV*DYPR))

      IF(VISC) then

         PMAZX=PMAZX+TEMP1*Q*PCMX
         PMAZY=PMAZY+TEMP1*Q*PCMY
         PMAZZ=PMAZZ+TEMP1*Q*PCMZ

         PMAZW=PMAZW+TEMP1*Q*PCMW
         PMAZU=PMAZU+TEMP1*Q*PCMU
         PMAZV=PMAZV+TEMP1*Q*PCMV

      endif

      TEMP2=Q*TEMP1
      TEMP4=TEMP2*CS/XLEN

      PMAXM=-TEMP4*PYCGM
      PMAYM=TEMP4*PXCGM
      PMAZM=TEMP2*(CA*PXCGM-CN*PYCGM)/XLEN

      TEMP3=TEMP2*EXZ

      PMAXCP=TEMP3*PBCP
      PXCPCP=TEMP3*PBCPCP

      IF(.not.PONLY) then

         PMAXCY=TEMP3*PBCY
         PXCYCY=TEMP3*PBCYCY
         PXCPCY=TEMP3*PBCPCY
```



```
      endif

      TEMP3=TEMP2*EYZ

      PMAYCP=TEMP3*PBCP
      PYCPCP=TEMP3*PBCPCP

      IF(.not.PONLY) then

         PMAYCY=TEMP3*PBCY
         PYCYCY=TEMP3*PBCYCY
         PYCPCY=TEMP3*PBCPCY

      endif

      TEMP3=TEMP2*EZY

      PMAZCP=TEMP3*PACP
      PZCPCP=TEMP3*PACPCP

      EZYY=EZYY*TEMP2

      IF(VISC) PZCPCP=PZCPCP+EZYY*PACP*PACP

      IF(.not.PONLY) then

         PMAZCY=TEMP3*PACY
         PZCYCY=TEMP3*PACYCY
         PZCPCY=TEMP3*PACPCY

         IF(VISC) then

            PZCPCY=PZCPCY+EZYY*PACP*PACY
            PZCYCY=PZCYCY+EZYY*PACY*PACY

         endif

      endif

      AMX=AMX*TEMP2
      AMY=AMY*TEMP2
      AMZ=AMZ*TEMP2+FMB

      AMZ=AMZ+TMZ

      if(max_qalp.eq.1) then
         amz_new=amz_new*temp2+fmb+tmz
         pamz_alp=temp2*(cmalp+caalp*dxpr-cnalp*dypr)
      endif

      PMAZX=PMAZX+PTMZX
      PMAZY=PMAZY+PTMZY
      PMAZZ=PMAZZ+PTMZZ

      PMAZM=PMAZM+PTMZM
```

```
      DX=XGPA-XCG
      DY=YGPA-YCG
      DZ=ABS(ZGPA)

      DYI=1./DY
      DZI=1./DZ

      DXDY=DX*DYI
      UDXDY2=1.+DXDY*DXDY

      CRHO=1./SQRT(UDXDY2)
      SRHO=DXDY*CRHO

      CXX=DZI*SRHO
      CXY=DZI*CRHO
      CXZ=DYI*CRHO

      CYX=-CXX
      CYY=-CXY
      CYZ=CXZ

      CZX=DYI/UDXDY2
      CZY=-DXDY*CZX

      TEMP4=-PXCGM+DXDY*PYCGM

      PCRHOM=-SRHO*CZX*TEMP4
      PSRHOM=DXDY*PCRHOM+CRHO*TEMP4*DYI

      PCXXM=DZI*PSRHOM
      PCXYM=DZI*PCRHOM
      PCXZM=DYI*(PCRHOM+CXZ*PYCGM)

      PCYXM=-PCXXM
      PCYYM=-PCXYM
      PCYZM=PCXZM

      PCZXM=CZX*DYI*PYCGM+2.*CRHO*PCRHOM
      PCZYM=DYI*TEMP4*CZX-DXDY*PCZXM

      THRUST=TNE(1,ITHR)*(THROT(ITHR)*ENGDAT(6,N)*PTN-ENGDAT(8,N)*PAM)

      PTM=0.

      tau=throt(ithr)

      if(mpsflg(ithr).eq.6) then

         m1=1
         call amulg(1,m1,15,t_time(1,ithr),timemps,tau2,taumps,0,0)

         thr1=throt(ithr)*engdat(6,n)*ptn

         thr2=tau2*engdat(6,n)*ptn

         thrust=0.
```



```
         do i=1,15

            if(nsyst(i).eq.1.and.engines(i,ithr).eq.1) then

               if(mpsinp(i).eq.0) then

                  thrust=thrust+thr1-engdat(8,n)*pam

               else

                  thrust=thrust+thr2-engdat(8,n)*pam

               endif

            endif

         enddo

         ptm=0.

      endif

      IF(NBTRG5.eq.2) then

         if(mpsflg(ithr).eq.4) then

            PTM=GLIM(ITHR)*GZERO
            GM=XM*PTM
            TAU=GM/(TNE(1,ITHR)*ENGDAT(6,N)*PTN)

         endif

         if(mpsflg(ithr).eq.5) then

            tau=tau_const(jthrot(ithr),nstage)

         endif

         THRUST=TNE(1,ITHR)*(TAU*ENGDAT(6,N)*PTN-ENGDAT(8,N)*PAM)

      endif

      TEMP=-TNE(1,ITHR)*ENGDAT(8,N)*DPDH

      PTX=TEMP*PHX
      PTY=TEMP*PHY
      PTZ=TEMP*PHZ

C*********************************************************************

C     GIMBAL ANGLES AND PARTIALS

C*********************************************************************

      OTHR=1./THRUST
```

```
      TANDP1=-OTHR*(CXX*AMX+CXY*AMY+CXZ*AMZ)
      TANDP2=-OTHR*(CYX*AMX+CYY*AMY+CYZ*AMZ)
      TANDY=-OTHR*(CZX*AMX+CZY*AMY)

      if(max_qalp.eq.1) then
         tandp1_new=-othr*(cxx*amx+cxy*amy+cxz*amz_new)
         tandp2_new=-othr*(cyx*amx+cyy*amy+cyz*amz_new)

         ptandp1_alp=-othr*cxz*pamz_alp
         ptandp2_alp=-othr*cyz*pamz_alp
      endif

      PTP1X=-OTHR*(PTX*TANDP1+CXX*PMAXX+CXY*PMAYX+CXZ*PMAZX)
      PTP1Y=-OTHR*(PTY*TANDP1+CXX*PMAXY+CXY*PMAYY+CXZ*PMAZY)
      PTP1Z=-OTHR*(PTZ*TANDP1+CXX*PMAXZ+CXY*PMAYZ+CXZ*PMAZZ)

      PTP2X=-OTHR*(PTX*TANDP2+CYX*PMAXX+CYY*PMAYX+CYZ*PMAZX)
      PTP2Y=-OTHR*(PTY*TANDP2+CYX*PMAXY+CYY*PMAYY+CYZ*PMAZY)
      PTP2Z=-OTHR*(PTZ*TANDP2+CYX*PMAXZ+CYY*PMAYZ+CYZ*PMAZZ)

      PTYX =-OTHR*(PTX*TANDY +CZX*PMAXX+CZY*PMAYX)
      PTYY =-OTHR*(PTY*TANDY +CZX*PMAXY+CZY*PMAYY)
      PTYZ =-OTHR*(PTZ*TANDY +CZX*PMAXZ+CZY*PMAYZ)

      PTP1W=-OTHR*(CXX*PMAXW+CXY*PMAYW+CXZ*PMAZW)
      PTP1U=-OTHR*(CXX*PMAXU+CXY*PMAYU+CXZ*PMAZU)
      PTP1V=-OTHR*(CXX*PMAXV+CXY*PMAYV+CXZ*PMAZV)

      PTP2W=-OTHR*(CYX*PMAXW+CYY*PMAYW+CYZ*PMAZW)
      PTP2U=-OTHR*(CYX*PMAXU+CYY*PMAYU+CYZ*PMAZU)
      PTP2V=-OTHR*(CYX*PMAXV+CYY*PMAYV+CYZ*PMAZV)

      PTYW =-OTHR*(CZX*PMAXW+CZY*PMAYW)
      PTYU =-OTHR*(CZX*PMAXU+CZY*PMAYU)
      PTYV =-OTHR*(CZX*PMAXV+CZY*PMAYV)

      TEMP4=OTHR*PTM
      TEMP5=TEMP4*AMX-PMAXM
      TEMP6=TEMP4*AMY-PMAYM
      TEMP7=TEMP4*AMZ-PMAZM

      PTP1M=OTHR*(CXX*TEMP5+CXY*TEMP6+CXZ*TEMP7-(PCXXM*AMX+PCXYM*AMY+
     *      PCXZM*AMZ))
      PTP2M=OTHR*(CYX*TEMP5+CYY*TEMP6+CYZ*TEMP7-(PCYXM*AMX+PCYYM*AMY+
     *      PCYZM*AMZ))
      PTYM =OTHR*(CZX*TEMP5+CZY*TEMP6           -(PCZXM*AMX+PCZYM*AMY))

      PTP1CP=-OTHR*(CXX*PMAXCP+CXY*PMAYCP+CXZ*PMAZCP)
      PTP2CP=-OTHR*(CYX*PMAXCP+CYY*PMAYCP+CYZ*PMAZCP)
      PTYCP=-OTHR*(CZX*PMAXCP+CZY*PMAYCP)

      IF(.not.PONLY) then

         PTP1CY=-OTHR*(CXX*PMAXCY+CXY*PMAYCY+CXZ*PMAZCY)
         PTP2CY=-OTHR*(CYX*PMAXCY+CYY*PMAYCY+CYZ*PMAZCY)
```



```
         PTYCY=-OTHR*(CZX*PMAXCY+CZY*PMAYCY)

      endif

      PP1CPP=-OTHR*(CXX*PXCPCP+CXY*PYCPCP+CXZ*PZCPCP)
      PP2CPP=-OTHR*(CYX*PXCPCP+CYY*PYCPCP+CYZ*PZCPCP)
      PYCPCP=-OTHR*(CZX*PXCPCP+CZY*PYCPCP)

      IF(.not.PONLY) then

         PP1CYY=-OTHR*(CXX*PXCYCY+CXY*PYCYCY+CXZ*PZCYCY)
         PP2CYY=-OTHR*(CYX*PXCYCY+CYY*PYCYCY+CYZ*PZCYCY)
         PYCYCY=-OTHR*(CZX*PXCYCY+CZY*PYCYCY)

         PP1CPY=-OTHR*(CXX*PXCPCY+CXY*PYCPCY+CXZ*PZCPCY)
         PP2CPY=-OTHR*(CYX*PXCPCY+CYY*PYCPCY+CYZ*PZCPCY)
         PYCPCY=-OTHR*(CZX*PXCPCY+CZY*PYCPCY)

      endif

      TEMP1=SQRT(1.+TANDP1*TANDP1+TANDY*TANDY)
      TEMP2=SQRT(1.+TANDP2*TANDP2+TANDY*TANDY)

      TY1=THRUST/2./TEMP1
      TY2=THRUST/2./TEMP2

      TXP=-TANDP1*TY1-TANDP2*TY2
      TYP=TY1+TY2

      if(max_qalp.eq.1) then
         TEMP3=SQRT(1.+TANDP1_new**2+TANDY*TANDY)
         TEMP4=SQRT(1.+TANDP2_new**2+TANDY*TANDY)

         TY1_new=THRUST/2./TEMP3
         TY2_new=THRUST/2./TEMP4

         pty1_alp=-thrust/2.*tandp1_new*ptandp1_alp/temp3**3
         pty2_alp=-thrust/2.*tandp2_new*ptandp2_alp/temp4**3

         TXP_new=-TANDP1_new*TY1_new-TANDP2_new*TY2_new
         TYP_new=TY1_new+TY2_new

         ptxp_alp=-ptandp1_alp*ty1_new-tandp1_new*pty1_alp-
     *           ptandp2_alp*ty2_new-tandp2_new*pty2_alp
         ptyp_alp=pty1_alp+pty2_alp
      endif

      TEMP7=TEMP1*TEMP1
      TEMP8=TEMP2*TEMP2

      PTY1X=TY1*(PTX*OTHR-(TANDP1*PTP1X+TANDY*PTYX)/TEMP7)
      PTY1Y=TY1*(PTY*OTHR-(TANDP1*PTP1Y+TANDY*PTYY)/TEMP7)
      PTY1Z=TY1*(PTZ*OTHR-(TANDP1*PTP1Z+TANDY*PTYZ)/TEMP7)

      PTY2X=TY2*(PTX*OTHR-(TANDP2*PTP2X+TANDY*PTYX)/TEMP8)
      PTY2Y=TY2*(PTY*OTHR-(TANDP2*PTP2Y+TANDY*PTYY)/TEMP8)
```

```
      PTY2Z=TY2*(PTZ*OTHR-(TANDP2*PTP2Z+TANDY*PTYZ)/TEMP8)

      PTY1M=TY1*(PTM*OTHR-(TANDP1*PTP1M+TANDY*PTYM)/TEMP7)
      PTY2M=TY2*(PTM*OTHR-(TANDP2*PTP2M+TANDY*PTYM)/TEMP8)

      TXPX=-PTP1X*TY1-PTP2X*TY2-TANDP1*PTY1X-TANDP2*PTY2X
      TXPY=-PTP1Y*TY1-PTP2Y*TY2-TANDP1*PTY1Y-TANDP2*PTY2Y
      TXPZ=-PTP1Z*TY1-PTP2Z*TY2-TANDP1*PTY1Z-TANDP2*PTY2Z

      TYPX=PTY1X+PTY2X
      TYPY=PTY1Y+PTY2Y
      TYPZ=PTY1Z+PTY2Z

      TEMP3=(TANDP1*PTP1CP+TANDY*PTYCP)/TEMP7
      TEMP4=(TANDP2*PTP2CP+TANDY*PTYCP)/TEMP8

      PTY1CP=-TY1*TEMP3
      PTY2CP=-TY2*TEMP4

      TXPCP=-PTP1CP*TY1-PTP2CP*TY2-TANDP1*PTY1CP-TANDP2*PTY2CP
      TYPCP=PTY1CP+PTY2CP

      IF(.not.PONLY) then

         TEMP5=(TANDP1*PTP1CY+TANDY*PTYCY)/TEMP7
         TEMP6=(TANDP2*PTP2CY+TANDY*PTYCY)/TEMP8

         PTY1CY=-TY1*TEMP5
         PTY2CY=-TY2*TEMP6

         TXPCY=-PTP1CY*TY1-PTP2CY*TY2-TANDP1*PTY1CY-TANDP2*PTY2CY
         TYPCY=PTY1CY+PTY2CY

      endif

      PYOCPP=TY1*(3.*TEMP3*TEMP3-(PTP1CP*PTP1CP+TANDP1*PP1CPP+PTYCP*
     *       PTYCP+TANDY*PYCPCP)/TEMP7)
      PYTCPP=TY2*(3.*TEMP4*TEMP4-(PTP2CP*PTP2CP+TANDP2*PP2CPP+PTYCP*
     *       PTYCP+TANDY*PYCPCP)/TEMP8)

      IF(.not.PONLY) then

         PYOCYY=TY1*(3.*TEMP5*TEMP5-(PTP1CY*PTP1CY+TANDP1*PP1CYY+
     *          PTYCY*PTYCY+TANDY*PYCYCY)/TEMP7)
         PYTCYY=TY2*(3.*TEMP6*TEMP6-(PTP2CY*PTP2CY+TANDP2*PP2CYY+
     *          PTYCY*PTYCY+TANDY*PYCYCY)/TEMP8)

         PYOCPY=TY1*(3.*TEMP3*TEMP5-(PTP1CY*PTP1CP+TANDP1*PP1CPY+
     *          PTYCP*PTYCY+TANDY*PYCPCY)/TEMP7)
         PYTCPY=TY2*(3.*TEMP4*TEMP6-(PTP2CY*PTP2CP+TANDP2*PP2CPY+
     *          PTYCP*PTYCY+TANDY*PYCPCY)/TEMP8)
      endif

      TXPCPP=-PP1CPP*TY1-PP2CPP*TY2-2.*(PTP1CP*PTY1CP+PTP2CP*PTY2CP)-
     *       TANDP1*PYOCPP-TANDP2*PYTCPP
      TYPCPP=PYOCPP+PYTCPP
```



```
      PTZCPP=PYCPCP*TYP+2.*PTYCP*TYPCP+TANDY*TYPCPP
      PTXCPP=TXPCPP*CRHO+TYPCPP*SRHO
      PTYCPP=-TXPCPP*SRHO+TYPCPP*CRHO

      IF(.not.PONLY) then

         TXPCYY=-PP1CYY*TY1-PP2CYY*TY2-2.*(PTP1CY*PTY1CY+PTP2CY*
     *          PTY2CY)-TANDP1*PYOCYY-TANDP2*PYTCYY
         TYPCYY=PYOCYY+PYTCYY

         PTZCYY=PYCYCY*TYP+2.*PTYCY*TYPCY+TANDY*TYPCYY

         TXPCPY=-PP1CPY*TY1-PP2CPY*TY2-PTP1CP*PTY1CY-PTP2CP*PTY2CY-
     *          PTP1CY*PTY1CP-PTP2CY*PTY2CP-TANDP1*PYOCPY-TANDP2*PYTCPY
         TYPCPY=PYOCPY+PYTCPY

         PTZCPY=PYCPCY*TYP+PTYCP*TYPCY+PTYCY*TYPCP+TANDY*TYPCPY

         PTXCYY=TXPCYY*CRHO+TYPCYY*SRHO
         PTXCPY=TXPCPY*CRHO+TYPCPY*SRHO

         PTYCYY=-TXPCYY*SRHO+TYPCYY*CRHO
         PTYCPY=-TXPCPY*SRHO+TYPCPY*CRHO

      endif

c*************************************************************************

C     PARTIAL OF THRUST COMPONENTS W.R.T. POSITION

c*************************************************************************

510   PTZX=PTYX*TXP+TANDY*TYPX+PFZX
      PTZY=PTYY*TXP+TANDY*TYPY+PFZY
      PTZZ=PTYZ*TXP+TANDY*TYPZ+PFZZ

      PTXX=TXPX*CRHO+TYPX*SRHO+PFXX
      PTXY=TXPY*CRHO+TYPY*SRHO+PFXY
      PTXZ=TXPZ*CRHO+TYPZ*SRHO+PFXZ

      PTYX=-TXPX*SRHO+TYPX*CRHO+PFYX
      PTYY=-TXPY*SRHO+TYPY*CRHO+PFYY
      PTYZ=-TXPZ*SRHO+TYPZ*CRHO+PFYZ

      PTZCP=PTYCP*TYP+TANDY*TYPCP
      PTXCP=TXPCP*CRHO+TYPCP*SRHO
      PTYCP=-TXPCP*SRHO+TYPCP*CRHO

      IF(.not.PONLY) then

         PTZCY=PTYCY*TYP+TANDY*TYPCY
         PTXCY=TXPCY*CRHO+TYPCY*SRHO
         PTYCY=-TXPCY*SRHO+TYPCY*CRHO

      endif
```

```
c*******************************************************************
C     PARTIAL OF THRUST COMPONENTS W.R.T. VELOCITY
c*******************************************************************

      PTY1W=-TY1*(TANDP1*PTP1W+TANDY*PTYW)/(TEMP1*TEMP1)
      PTY1U=-TY1*(TANDP1*PTP1U+TANDY*PTYU)/(TEMP1*TEMP1)
      PTY1V=-TY1*(TANDP1*PTP1V+TANDY*PTYV)/(TEMP1*TEMP1)

      PTY2W=-TY2*(TANDP2*PTP2W+TANDY*PTYW)/(TEMP2*TEMP2)
      PTY2U=-TY2*(TANDP2*PTP2U+TANDY*PTYU)/(TEMP2*TEMP2)
      PTY2V=-TY2*(TANDP2*PTP2V+TANDY*PTYV)/(TEMP2*TEMP2)

      TXPW=-PTP1W*TY1-PTP2W*TY2-TANDP1*PTY1W-TANDP2*PTY2W
      TXPU=-PTP1U*TY1-PTP2U*TY2-TANDP1*PTY1U-TANDP2*PTY2U
      TXPV=-PTP1V*TY1-PTP2V*TY2-TANDP1*PTY1V-TANDP2*PTY2V

      TYPW=PTY1W+PTY2W
      TYPU=PTY1U+PTY2U
      TYPV=PTY1V+PTY2V

      PTZW=PTYW*TYP+TANDY*TYPW
      PTZU=PTYU*TYP+TANDY*TYPU
      PTZV=PTYV*TYP+TANDY*TYPV

      PTXW=TXPW*CRHO+TYPW*SRHO
      PTXU=TXPU*CRHO+TYPU*SRHO
      PTXV=TXPV*CRHO+TYPV*SRHO

      PTYW=-TXPW*SRHO+TYPW*CRHO
      PTYU=-TXPU*SRHO+TYPU*CRHO
      PTYV=-TXPV*SRHO+TYPV*CRHO

      TX=TXP*CRHO+TYP*SRHO+TFXX
      TY=-TXP*SRHO+TYP*CRHO+TFYY
      TZ=TANDY*TYP+TFZZ

      if(max_qalp.eq.1) then

         TX_new=TXP_new*CRHO+TYP_new*SRHO+TFXX
         TY_new=-TXP_new*SRHO+TYP_new*CRHO+TFYY
         TZ_new=TANDY*TYP_new+TFZZ

         ptx_alp= crho*ptxp_alp+srho*ptyp_alp
         pty_alp=-srho*ptxp_alp+crho*ptyp_alp
         ptz_alp=tandy*ptyp_alp

      endif

      TEMP=Q*S(ITHR)
      TEMP1=TX-TEMP*CN
      TEMP2=TY-TEMP*CA
      TEMP3=TZ+TEMP*CS
```



```
      DX=TEMP1*XHAT(1)+TEMP2*YHAT(1)+TEMP3*ZHAT(1)
      DY=TEMP1*XHAT(2)+TEMP2*YHAT(2)+TEMP3*ZHAT(2)
      DZ=temp1*xhat(3)+TEMP2*YHAT(3)+TEMP3*ZHAT(3)

      TEMP5=TEMP*CNALP

      IF(VISC) TEMP5=TEMP5+TEM1CN*TEMP

      PAACP =PTXCP  TEMP5*PACP
      PAACPP=PTXCPP-TEMP5*PACPCP

      TEM2CN=TEM2CN*TEMP

      IF(VISC) PAACPP=PAACPP+TEM2CN*PACP*PACP

      IF(.not.PONLY) then

         PAACY =PTXCY -TEMP5*PACY
         PAACYY=PTXCYY-TEMP5*PACYCY
         PAACPY=PTXCPY-TEMP5*PACPCY

         IF(VISC) then

            PAACYY=PAACYY+TEM2CN*PACY*PACY
            PAACPY=PAACPY+TEM2CN*PACP*PACY

         endif

      endif

      TEMP5=TEMP*CAALP
      IF(VISC) TEMP5=TEMP5+TEM1CA*TEMP

      PABCP =PTYCP -TEMP5*PACP
      PABCPP=PTYCPP-TEMP5*PACPCP

      TEM2CA=TEM2CA*TEMP

      IF(VISC) PABCPP=PABCPP+TEM2CA*PACP*PACP

      IF(.not.PONLY) then

         PABCY =PTYCY -TEMP5*PACY
         PABCYY=PTYCYY-TEMP5*PACYCY
         PABCPY=PTYCPY-TEMP5*PACPCY

         IF(VISC) then

            PABCYY=PABCYY+TEM2CA*PACY*PACY
            PABCPY=PABCPY+TEM2CA*PACP*PACY

         endif

      endif

      TEMP5=TEMP*CYBETA+FYB
```

```
      PACCP =PTZCP +TEMP5*PBCP
      PACCPP=PTZCPP+TEMP5*PBCPCP

      IF(.not.PONLY) then

         PACCY =PTZCY +TEMP5*PBCY
         PACCYY=PTZCYY+TEMP5*PBCYCY
         PACCPY=PTZCPY+TEMP5*PBCPCY

      endif

      OMASS=1./XM

      PDXCP=(XHAT(1)*PAACP+YHAT(1)*PABCP+ZHAT(1)*PACCP+TEMP1*PX1CP+TEMP2
     *       *PY1CP+TEMP3*PZ1CP)*OMASS
      PDYCP=(XHAT(2)*PAACP+YHAT(2)*PABCP+ZHAT(2)*PACCP+TEMP1*PX2CP+TEMP2
     *       *PY2CP+TEMP3*PZ2CP)*OMASS
      PDZCP=(xhat(3)*paacp+YHAT(3)*PABCP+ZHAT(3)*PACCP+temp1*px3cp+TEMP2
     *       *PY3CP+TEMP3*PZ3CP)*OMASS

      IF(.not.PONLY) then

         PDXCY=(XHAT(1)*PAACY+YHAT(1)*PABCY+ZHAT(1)*PACCY+TEMP1*PX1CY+
     *          TEMP2*PY1CY+TEMP3*PZ1CY)*OMASS
         PDYCY=(XHAT(2)*PAACY+YHAT(2)*PABCY+ZHAT(2)*PACCY+TEMP1*PX2CY+
     *          TEMP2*PY2CY+TEMP3*PZ2CY)*OMASS
         PDZCY=(xhat(3)*paacy+YHAT(3)*PABCY+ZHAT(3)*PACCY+temp1*px3cy+
     *          TEMP2*PY3CY+TEMP3*PZ3CY)*OMASS

      endif

      PDXCPP=(XHAT(1)*PAACPP+YHAT(1)*PABCPP+ZHAT(1)*PACCPP+2.*PX1CP*
     *       PAACP+2.*PY1CP*PABCP+2.*PZ1CP*PACCP+TEMP1*PX1CPP+TEMP2*
     *       PY1CPP+TEMP3*PZ1CPP)*OMASS
      PDYCPP=(XHAT(2)*PAACPP+YHAT(2)*PABCPP+ZHAT(2)*PACCPP+2.*PX2CP*
     *       PAACP+2.*PY2CP *PABCP+2.*PZ2CP*PACCP+TEMP1*PX2CPP+TEMP2*
     *       PY2CPP+TEMP3*PZ2CPP)*OMASS
      PDZCPP=(xhat(3)*paacpp+YHAT(3)*PABCPP+ZHAT(3)*PACCPP+2.*px3cp*
     *       paacp+2.*PY3CP *PABCP+2.*PZ3CP*PACCP+temp1*px3cpp+TEMP2*
     *       PY3CPP+TEMP3*PZ3CPP)*OMASS

      IF(.not.PONLY) then

         PDXCYY=(XHAT(1)*PAACYY+YHAT(1)*PABCYY+ZHAT(1)*PACCYY+2.*PX1CY*
     *          PAACY+2.*PY1CY*PABCY+2.*PZ1CY*PACCY+TEMP1*PX1CYY+TEMP2*
     *          PY1CYY+TEMP3*PZ1CYY)*OMASS
         PDYCYY=(XHAT(2)*PAACYY+YHAT(2)*PABCYY+ZHAT(2)*PACCYY+2.*PX2CY*
     *          PAACY+2.*PY2CY*PABCY+2.*PZ2CY*PACCY+TEMP1*PX2CYY+TEMP2*
     *          PY2CYY+TEMP3*PZ2CYY)*OMASS
         PDZCYY=(xhat(3)*paacyy+YHAT(3)*PABCYY+ZHAT(3)*PACCYY+2.*px3cy*
     *          paacy+2.*PY3CY*PABCY+2.*PZ3CY*PACCY+temp1*px3cyy+TEMP2*
     *          PY3CYY+TEMP3*PZ3CYY)*OMASS

         PDXCPY=(XHAT(1)*PAACPY+YHAT(1)*PABCPY+ZHAT(1)*PACCPY+PX1CY*
     *          PAACP+PX1CP*PAACY+PY1CY*PABCP+PY1CP*PABCY+PZ1CY*PACCP+
```



```
     *          PZ1CP*PACCY+TEMP1*PX1CPY+TEMP2*PY1CPY+TEMP3*PZ1CPY)
     *          *OMASS
         PDYCPY=(XHAT(2)*PAACPY+YHAT(2)*PABCPY+ZHAT(2)*PACCPY+PX2CY*
     *          PAACP+PX2CP*PAACY+PY2CY*PABCP+PY2CP*PABCY+PZ2CY*PACCP+
     *          PZ2CP*PACCY+TEMP1*PX2CPY+TEMP2*PY2CPY+TEMP3*PZ2CPY)
     *          *OMASS
         PDZCPY=(xhat(3)*paacpy+YHAT(3)*PABCPY+ZHAT(3)*PACCPY+px3cy*
     *          paacy+px3cp*paacy+PY3CY*PABCP+PY3CP*PABCY+PZ3CY*PACCP+
     *          PZ3CP*PACCY+temp1*px3cpy+TEMP2*PY3CPY+TEMP3*PZ3CPY)
     *          *OMASS

      endif

      TEMP3=TY1*TANDP1+TY2*TANDP2
      TEMP4=TY1+TY2
      TEMP5=-PTP1M*TY1-PTP2M*TY2-TANDP1*PTY1M-TANDP2*PTY2M
      TEMP6=PTY1M+PTY2M

      PTZM=PTYM*TEMP4+TANDY*TEMP6+ptzfm
      PTXM=-TEMP3*PCRHOM+TEMP4*PSRHOM+CRHO*TEMP5+SRHO*TEMP6+ptxfm
      PTYM=TEMP3*PSRHOM+TEMP4*PCRHOM-SRHO*TEMP5+CRHO*TEMP6+ptyfm

      TEMP4=1./XM
      TEMP5=TEMP4*TEMP4

      PXM=-TEMP5*DX+TEMP4*(PTXM*XHAT(1)+PTYM*YHAT(1)+PTZM*ZHAT(1))
      PYM=-TEMP5*DY+TEMP4*(PTXM*XHAT(2)+PTYM*YHAT(2)+PTZM*ZHAT(2))
      PZM=-TEMP5*DZ+TEMP4*(ptxm*xhat(3)+PTYM*YHAT(3)+PTZM*ZHAT(3))

    4 CONTINUE

      TEMP1=PTXX-S(ITHR)*(PQX*CN+Q*PCNX)
      TEMP2=PTXY-S(ITHR)*(PQY*CN+Q*PCNY)
      TEMP3=PTXZ-S(ITHR)*(PQZ*CN+Q*PCNZ)
      TEMP4=PTYX-S(ITHR)*(PQX*CA+Q*PCAX)
      TEMP5=PTYY-S(ITHR)*(PQY*CA+Q*PCAY)
      TEMP6=PTYZ-S(ITHR)*(PQZ*CA+Q*PCAZ)
      TEMP7=PTZX+S(ITHR)*(PQX*CS+Q*PCSX)
      TEMP8=PTZY+S(ITHR)*(PQY*CS+Q*PCSY)
      TEMP9=PTZZ+S(ITHR)*(PQZ*CS+Q*PCSZ)
C*********************************************************************
C     TOTAL DERIVATIVE W.R.T POSITION
c*********************************************************************

      PXX=OMASS*(XHAT(1)*TEMP1+YHAT(1)*TEMP4+ZHAT(1)*TEMP7)+GXX
      PXY=OMASS*(XHAT(1)*TEMP2+YHAT(1)*TEMP5+ZHAT(1)*TEMP8)+GYX
      PXZ=OMASS*(XHAT(1)*TEMP3+YHAT(1)*TEMP6+ZHAT(1)*TEMP9)+GZX

      PYX=OMASS*(XHAT(2)*TEMP1+YHAT(2)*TEMP4+ZHAT(2)*TEMP7)+GYX
      PYY=OMASS*(XHAT(2)*TEMP2+YHAT(2)*TEMP5+ZHAT(2)*TEMP8)+GYY
      PYZ=OMASS*(XHAT(2)*TEMP3+YHAT(2)*TEMP6+ZHAT(2)*TEMP9)+GZY

      PZX=OMASS*(xhat(3)*temp1+YHAT(3)*TEMP4+ZHAT(3)*TEMP7)+GZX
```

```
      PZY=OMASS*(xhat(3)*temp2+YHAT(3)*TEMP5+ZHAT(3)*TEMP8)+GZY
      PZZ=OMASS*(xhat(3)*temp3+YHAT(3)*TEMP6+ZHAT(3)*TEMP9)+GZZ

      TEMP1=PTXW-S(ITHR)*(PQW*CN+Q*PCNW)
      TEMP2=PTXU-S(ITHR)*(PQU*CN+Q*PCNU)
      TEMP3=PTXV-S(ITHR)*(PQV*CN+Q*PCNV)
      TEMP4=PTYW-S(ITHR)*(PQW*CA+Q*PCAW)
      TEMP5=PTYU-S(ITHR)*(PQU*CA+Q*PCAU)
      TEMP6=PTYV-S(ITHR)*(PQV*CA+Q*PCAV)
      TEMP7=PTZW+S(ITHR)*(PQW*CS+Q*PCSW)
      TEMP8=PTZU+S(ITHR)*(PQU*CS+Q*PCSU)
      TEMP9=PTZV+S(ITHR)*(PQV*CS+Q*PCSV)

C*********************************************************************

C     TOTAL DERIVATIVE W.R.T VELOCITY

C*********************************************************************

      PXW=OMASS*(XHAT(1)*TEMP1+YHAT(1)*TEMP4+ZHAT(1)*TEMP7)
      PXU=OMASS*(XHAT(1)*TEMP2+YHAT(1)*TEMP5+ZHAT(1)*TEMP8)
      PXV=OMASS*(XHAT(1)*TEMP3+YHAT(1)*TEMP6+ZHAT(1)*TEMP9)

      PYW=OMASS*(XHAT(2)*TEMP1+YHAT(2)*TEMP4+ZHAT(2)*TEMP7)
      PYU=OMASS*(XHAT(2)*TEMP2+YHAT(2)*TEMP5+ZHAT(2)*TEMP8)
      PYV=OMASS*(XHAT(2)*TEMP3+YHAT(2)*TEMP6+ZHAT(2)*TEMP9)

      PZW=OMASS*(xhat(3)*temp1+YHAT(3)*TEMP4+ZHAT(3)*TEMP7)
      PZU=OMASS*(xhat(3)*temp2+YHAT(3)*TEMP5+ZHAT(3)*TEMP8)
      PZV=OMASS*(xhat(3)*temp3+YHAT(3)*TEMP6+ZHAT(3)*TEMP9)

C*********************************************************************

c       Define partial derivative of EOM with respect to angle of
c       attack (for calculation during q alpha constraint boundary)

C*********************************************************************

      if(max_qalp.eq.1) then

         TEMP1=TX_new-TEMP*CN_new
         TEMP2=TY_new-TEMP*CA_new
         TEMP3=TZ_new+TEMP*CS

         temp4=ptx_alp-temp*cnalp
         temp5=pty_alp-temp*caalp
         temp6=ptz_alp

         px_alp=omass*(xhat_new(1)*temp4+yhat_new(1)*temp5+
     *          zhat(1)*temp6+pxhat_alp(1)*temp1+pyhat_alp(1)*temp2)
         py_alp=omass*(xhat_new(2)*temp4+yhat_new(2)*temp5+
     *          zhat(2)*temp6+pxhat_alp(2)*temp1+pyhat_alp(2)*temp2)
         pz_alp=omass*(xhat_new(3)*temp4+yhat_new(3)*temp5+
     *          zhat(3)*temp6+pxhat_alp(3)*temp1+pyhat_alp(3)*temp2)

C*********************************************************************
```



```
c       Define coefficient if q alpha limit is currently being
c       enforced

C*********************************************************************

         j=nchiot(nstage)
         k=1
         if(j.ne.0) then
            k=noss+1
            do m=j,1,-1
               if(ithr.le.nvrst(m)) k=k-ncnrs(m)
            enddo
         endif
         NOSSM1=NOSS-1
         km1=k-1
         if(km1.le.0) km1=1

         DO I=1,7
            COVL(I)=YL(I,1)
            DO J=km1,NOSSM1
               COVL(I)=COVL(I)+GNU(J)*YL(I,J+1)
            enddo
         enddo

         f_alp=covl(1)*px_alp+covl(2)*py_alp+covl(3)*pz_alp
c        g1=qalpha_max-abs(q*alp)
         pg1_alp=-q*sign(1.d0,alp)

         xnu1=-f_alp/pg1_alp

         write(31,*) t
         write(31,*) alp*rad,alp_new*rad,xnu1
         write(31,*) covl(1),covl(2),covl(3)
         write(31,*) pdxcpp,pdycpp,pdzcpp
         write(31,*) pdxcyy,pdycyy,pdzcyy
         write(31,*) pdxcpy,pdycpy,pdzcpy
      endif

C*********************************************************************

c       Heating rate partial derivatives

C*********************************************************************

      IF(MAXHAT.EQ.0) RETURN

      HRATE1=XK1*(XK2*RHO)**.5*(VR/XK4)**3.07
      HRATE2=XK5/(XK6*PAT(12)+((VR/XK3)**2)/XK7)

      HRATE=HRATE1*(1.-HRATE2)

      HRATE3=HRATE1*HRATE2*HRATE2/XK5
      HRATE4=HRATE*DRDH/RHO+HRATE3*XK6*PAT(13)
      HRATE5=HRATE*3.07/VR+HRATE3*2./XK7*VR/(XK3*XK3)
```

```
      PHRW=HRATE5*PVRW
      PHRU=HRATE5*PVRU
      PHRV=HRATE5*PVRV

      PHRX=HRATE5*PVRX+HRATE4*PHX
      PHRY=HRATE5*PVRY+HRATE4*PHY
      PHRZ=HRATE5*PVRZ+HRATE4*PHZ

      RETURN
      END
```

3.25 Subroutine BDRI

3.25.1 Purpose

The purpose of this subroutine is to compute derivatives of the adjoint equations.

3.25.2 Variable Listing

3.25.3 Subroutines Called:

Name	Function
DEVISP	Calculates the partial derivative of specific impulse with respect to mass
FUNISP	Calculates specific impulse based on throttle level

3.25.4 Calling Subroutines:

Name	Function
DPIR	Integration routine

3.25.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
alp	Angle of attach based on Q alpha	deg	bdr1i	input	Global	q_partials
disp	Partial derivative of specific impulse with respect to mass				Local	
engdat(8,15)	Engine data matrix		ainit	input	Global	agen1
glim(15)	Acceleration limit	g's	ainit	input	Global	mps
gzero	Gravitational constant (9.80665 m/sec)	m/sec^	ainit	input	Global	const
ithr	Thrust event index number		athrev	input output	Global	agen3
m	Index				Local	
maxhat	Index used to indicate the number of the constraint used for the maximum heating constraint			output input	Global	heat
mpsflg(15)	Flag indicating the type of MPS thrust simulation		ainit	output	Global	agen3
nbtrg5	Acceleration limit trigger for backwards integration		bakrn	input	Global	bakint
nchiot(6)	Optimization flag used with branch trajectory option		ainit	input	Global	mult
ncnrs(5)	Number of constraints at restart		ainit	input	Global	rest
noss	Number of constraints		bakrn	input	Global	bgen3
nstage	Index number of current stage		athrev	input	Global	mult
nvrst(5)	Array specifying the thrust events where intermediate constraints will be imposed		ainit	output	Global	enput
pg1_q	Absolute value of angle of attack	rad			Local	
phrs(6)	Partial derivatives of heat rate with respect to position and velocity state variables			input	Global	phrs
pqu	Partial derivative of dynamic pressure with respect to u velocity component		bdr1i	input	Global	q_partials
pqv	Partial derivative of dynamic pressure with respect to v velocity component		bdr1i	input	Global	q_partials
pqw	Partial derivative of dynamic pressure with respect to w velocity component		bdr1i	input	Global	q_partials
pqx	Partial derivative of dynamic pressure with respect to x position component		bdr1i	input	Global	q_partials
pqy	Partial derivative of dynamic pressure with respect to y position component		bdr1i	input	Global	q_partials
pqz	Partial derivative of dynamic pressure		bdr1i	input	Global	q_partia

Variable	Variable Definition with respect to z position component	Units	Source	I/O	Status	Common Block ls
ptaum	Partial derivative of tau with respect to mass				Local	
ptn	Conversion from pounds to newtons	nt/lbs	ainit	input	Global	const
pxm	Partial derivative of X acceleration component with respect to mass		bdr1i	input	Global	calif
pxu	Partial derivative of X acceleration component with respect to Y velocity component		bdr1i	input	Global	calif
pxv	Partial derivative of X acceleration component with respect to Z velocity component		bdr1i	input	Global	calif
pxw	Partial derivative of X acceleration component with respect to Z velocity component		bdr1i	input	Global	calif
pxx	Partial derivative of X acceleration component with respect to X position component		bdr1i	input	Global	calif
pxy	Partial derivative of X acceleration component with respect to Y position component		bdr1i	input	Global	calif
pxz	Partial derivative of X acceleration component with respect to Z position component		bdr1i	input	Global	calif
pym	Partial derivative of Y acceleration component with respect to mass		bdr1i	input	Global	calif
pyu	Partial derivative of Y acceleration component with respect to Y velocity component		bdr1i	input	Global	calif
pyv	Partial derivative of Y acceleration component with respect to Z velocity component		bdr1i	input	Global	calif
pyw	Partial derivative of Y acceleration component with respect to X velocity component		bdr1i	input	Global	calif
pyx	Partial derivative of Y acceleration component with respect to X position component		bdr1i	input	Global	calif
pyy	Partial derivative of Y acceleration component with respect to Y position component		bdr1i	input	Global	calif
pyz	Partial derivative of Y acceleration component with respect to Z position		bdr1i	input	Global	calif

Variable	Variable Definition component	Units	Source	I/O	Status	Common Block
pzm	Partial derivative of Z acceleration component with respect to mass		bdr1i	input	Global	calif
pzu	Partial derivative of Z acceleration component with respect to Y velocity component		bdr1i	input	Global	calif
pzv	Partial derivative of Z acceleration component with respect to Z velocity component		bdr1i	input	Global	calif
pzw	Partial derivative of Z acceleration component with respect to X velocity component		bdr1i	input	Global	calif
pzx	Partial derivative of Z acceleration component with respect to X position component		bdr1i	input	Global	calif
pzy	Partial derivative of Z acceleration component with respect to Y position component		bdr1i	input	Global	calif
pzz	Partial derivative of Z acceleration component with respect to Z position component		bdr1i	input	Global	calif
tau	Throttle level to maintain acceleration limit				Local	
thr	Vacuum thrust of fixed engines	nt			Local	
tne(6,15)	Array which defines the number of engines per thrust event		ainit	input	Global	agen1
xisp	Specific impulse	sec			Local	
xm	Current vehicle mass	kg	bdri	output	Global	agen2
xmaug	Auxiliary vehicle mass	kg	athrev	input	Global	agen1
xmiad	Continuous portion of vehicle mass	kg	ader1	input	Global	agen2
xnu1	Temporary variable				Global	qalpha_ max
yl(7,20)	Integrated values of adjoint equations of motion		dpir	input	Global	bakint
ylc(7,20)	Adjoint differential equations		bdri	output	Global	bakint

```
      SUBROUTINE BDRI                                                   BDRI   1
C*********************************************************************
C       COMPUTES TOTAL ADJOINT EQUATIONS
C*********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      LOGICAL GLIMIT
C
      CHARACTER*60 HEAD
      CHARACTER*6 MISION
      character*12 header

      COMMON/AGEN1/TIME(2,16),TAUT(15),TAUW(15),TZERO,TLIFT,TTILT,TMINH,
     *DTZ,TQ,TL,XMAUG,TNE(6,15),ENGDAT(8,15),S(15),WD(15),WJET(15),
     *HEAD,PRINT(15),STEP(15),HNOM(15),HMAX(15),PROP(6),TBEGR,TENDR,
     *CHRDOT,CHIRO,FAZ,BO(3,2),CBAXIL,TCHIR,VRCUT,FPRFAC,CHVEL

      COMMON/AGEN2/AA,SA,CA,ALF,ALFY,CHIP,SCHIP,CCHIP,CHIY,SCHIY,CCHIY,
     *CHIR,SCHIR,CCHIR,STH,CTH,STHL,CTHL,DPHIZ,RTHE,R,VR,XM,SW,SU,SV,Q,
     *VIV(25),DVARS(13),DELXDW(15,3),DELXDR(7,5),DELXD(15,7),WZERO,ALT,
     *XMACH,QDOT,FAA,FAN,A(3,3),CASE,CT2,UMF,A12W,A22W,A32W,XJEXT,BEU,
     *BYL,BEL,BHMAX(15),BHMN,BSTEP(15),HNMB,HMXB,TPOLY,XMIAD

      COMMON/AGEN3/ITHR,IWD,IXR,JUMP,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISION,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)

      COMMON/BAKINT/BBANK(2),TBK,TBKD,YL(7,20),YLD(7,20),BSAVE(2700),
     *NBTRG1,TBV1,NBTRG2,TBV2,NBTRG3,TBV3,NBTRG4,TBV4,NBTRG5,TBV5,TEND,
     *DTB4,DU3,D23,D43,WISSD(20,20)

      COMMON/BGEN3/NOSS,JTB,NENT,NACT,NOM1,NP(7)

      COMMON/CALIF/PXX,PYX,PZX,PXY,PYY,PZY,PXZ,PYZ,PZZ,PXW,PYW,PZW,PXU,
     *PYU,PZU,PXV,PYV,PZV,PXM,PYM,PZM,PDXCP,PDYCP,PDZCP,PDXCY,PDYCY,PDZC
     *Y,PDXCPP,PDYCPP,PDZCPP,PDXCPY,PDYCPY,PDZCPY,PDXCYY,PDYCYY,PDZCYY

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM

      common/enput/psirst(6,5),header(20),nvrst(5),kcdres(6,5),last

      COMMON/HEAT/MAXHAT

      COMMON/MPS/TIMMPS(15),FTBL(15),THROT(15),GLIM(15),TAUALT(5),TAUTBL
     *(5),MPS1,MPS2,TGLIM(10,15),GLIMIT,tau_const(10,5),jthrot(15)

      common/mult/nchiot(6),tbias(6),nstage,nstg(15),wf(5),ibranch

      common/qalpha_max/qalpha_max,xnul
```



```
      common/q_partials/pqx,pqy,pqz,pqw,pqu,pqv,alp

      common/rest/jstr,ncnrs(5),ntcn,tr(5),lsb,nf8,nopar,wibt(15)

      COMMON/PHRS/PHRS(6)
c*********************************************************************
c       When engine throttling is used to provide a constant acceleration
c       limit, the backward adjoint equations must be altered to account
c       of the partial of mass with respect to the varying thrust level.
c       When nbtrg5 (the acceleration limit trigger) is on and the mps
c       throttle flag (mpsflg) is equal to 4, the partials and the
c       following equations are required to be calculated.
c*********************************************************************
      IF(NBTRG5.eq.2.and.MPSFLG(ITHR).eq.4) then

         XM=XMIAD+XMAUG
         THR=ENGDAT(6,1)*TNE(1,ITHR)*PTN
         TAU=XM*GZERO*GLIM(ITHR)/THR
         PTAUM=GZERO*GLIM(ITHR)/THR
         XISP=FUNISP(TAU,nstage)
         DISP=DEVISP(TAU,PTAUM,nstage)

      endif
c*********************************************************************
C     LAMBDA DOTS                                                       BDRI  24
c*********************************************************************
      j=nchiot(nstage)

      k=1

      if(j.gt.0) then
         k=noss+1
         do m=j,1,-1
            if(ithr.le.nvrst(m)) k=k-ncnrs(m)
         enddo
      endif

      DO 1 I=1,NOSS

      if(i.lt.k) then

         do l=1,7
            yld(l,i)=0.
         enddo

      else

         pgl_q=-abs(alp)
```

```
      YLD(1,I)=PXW*YL(1,I)+PYW*YL(2,I)+PZW*YL(3,I)+YL(4,I)-
     *           xnu1*pg1_q*pqw
      YLD(2,I)=PXU*YL(1,I)+PYU*YL(2,I)+PZU*YL(3,I)+YL(5,I)-
     *           xnu1*pg1_q*pqu
      YLD(3,I)=PXV*YL(1,I)+PYV*YL(2,I)+PZV*YL(3,I)+YL(6,I)-
     *           xnu1*pg1_q*pqv
      YLD(4,I)=PXX*YL(1,I)+PYX*YL(2,I)+PZX*YL(3,I)-
     *           xnu1*pg1_q*pqx
      YLD(5,I)=PXY*YL(1,I)+PYY*YL(2,I)+PZY*YL(3,I)-
     *           xnu1*pg1_q*pqy
      YLD(6,I)=PXZ*YL(1,I)+PYZ*YL(2,I)+PZZ*YL(3,I)-
     *           xnu1*pg1_q*pqz
      YLD(7,I)=PXM*YL(1,I)+PYM*YL(2,I)+PZM*YL(3,I)

      IF(NBTRG5.eq.2.and.MPSFLG(ITHR).eq.4.and.tau.ge..65) then

         YLD(7,I)=YLD(7,I)-GLIM(ITHR)*(XISP-XM*DISP)/(XISP*XISP)*
     *              YL(7,I)

      endif

    endif

  1 CONTINUE

    IF(MAXHAT.EQ.0) RETURN

    DO 3 I=1,NOSS

    IF(I.eq.MAXHAT) then

       do j=1,6
          YLD(J,MAXHAT)=YLD(J,MAXHAT)+PHRS(J)
       enddo

    endif

  3 CONTINUE

    RETURN
    END
```

3.26 Subroutine BKEND

3.26.1 Purpose

The purpose of this subroutine is to terminate the backward integration and adjust the influence coefficients (if required).

3.26.2 Variable Listing

3.26.3 Subroutines Called:

Name	Function
BSAVEG	Calculates the impulse response functions and integrates the weighting matrix
RTMRK	Terminates the integration routine

3.26.4 Calling Subroutines:

Name	Function
BAKRN	Controls the logic for the backward integration and calls the integration package
BTHREV	Controls the thrust event logic in the backward integration
DPIR	Integration routine

3.26.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
cfr(15)	Critical flowrate	kg/sec	ainit	input	Global	xtra
cisp(15)	MPS specific impulse	sec	ainit	input	Global	xtra
f(15)	MPS engine thrust	newton		input	Global	xtra
fmxx	Temporary variable used for calculation of partial derivatives for tank constraints		athrev	input	Global	fmxx
foms(15)	Thrust table for orbit maneuver system	lbs	ainit	input	Global	omsrcs
frcs(15)	Thrust table for rocket control system	lbs	ainit	input	Global	omsrcs
glim(15)	Acceleration limit	g's	ainit	input	Global	mps
gzero	Gravitational constant (9.80665 m/sec)	m/sec^	ainit	input	Global	const
ibranch	Flag indicating the terminal thrust event of the first trajectory of the branch trajectory			output	Global	mult
ilast	Flag indicating the first thrust event in the last stage			output	Global	ipr
invt	Index				Local	
ipr	Denotes the thrust event from which fuel propellant reserve (FPR) is calculated			input output	Global	ipr
irtflg	Flag indicating FPR calculations			output	Global	agen3
irtls	Return to launch flag			input	Global	agen3
jtb	Index for control table points		bkend	output input	Global	bgen3
kcdphi(20)	Terminal constraint selection flag array		ainit	input	Global	yli
kdb(40)	Array for control parameter flags		ainit	input	Global	auto
kdt(15)	Array used in confunction with the ipr option		ainit	input	Global	ipr
kv	Index				Local	
kv1	Index				Local	
kv2	Index				Local	
mpsflg(15)	Flag indicating the type of MPS thrust simulation		ainit	output	Global	agen3
msw	temporay variable		athrev	input	Global	fmxx
mswch(15)	Flag indicating which thrust events are considered critical			output	Global	agen3
nchiot(6)	Optimization flag used with branch trajectory option		ainit	input	Global	mult
nomor	Index indicating end of integration		bthrev	input	Global	nomor

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
noss	Number of constraints		bakrn	input	Global	bgen3
nowd(15)	Index indicating thrust event where weight drop event will occur		ainit	input	Global	agen3
nstg(15)	Internal index which relates thrust event to stage number		ainit	output	Global	mult
nvnt	Number of thrust events		ainit	input	Global	agen3
nwvnt	Number of weight drop events		ainit	input	Global	agen3
prk4	Temporary variable used in calculation of performance reserve option		ainit	input	Global	ipr
ptgt	Partial derivative of time to acceleration limit with respect to ith thrust event time				Local	
ptgti	Partial derivative of time to acceleration limit with respect to ith thrust event time				Local	
ptjti	Partial derivative of jth thrust event time to ith thrust event time				Local	
ptlti	Partial derivative of lth thrust event time with respect to ith thrust event time				Local	
taut(15)	Time increments for thrust events	sec	anewch	input	Global	agen1
temp	Temporary variable				Local	
temp1	Temporary variable				Local	
temp2	Temporary variable				Local	
tempr	Temporary variable				Local	
tend	Termination time of forward trajectory (start of backward intgration)	sec	bakrn	input	Global	bakint
tglim(10,15)	Time of acceleration limit	sec	atilt	input	Global	mps
time(2,16)	Actual times for thrust events measured from launch	sec	aforun	input	Global	agen1
tne(6,15)	Array which defines the number of engines per thrust event		ainit	input	Global	agen1
vrcut	Magnitude of characteristic velocity cutoff	m/sec	ainit	input	Global	agen1
wddump(15)	Amount of weight dumped during OMS/RCS thrust events	kg	ainit	input	Global	omsrcs
wdoms(15)	Weight flowrate of OMS engines	lbs/sec	ainit	input	Global	omsrcs
wdrcs(15)	Weight flowrate of RCS engines	lbs/sec	ainit	input	Global	omsrcs
wg	Mass at acceleration limit	kg	athrev	input	Global	fmxx

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
xlamb(40,20)	Partial derivatives of payoff and constraints with respect to the optimization parameters		bkend	output	Global	auto
xmd(15)	Propellant flowrate	kg/sec	ainit	input	Global	xtra
xmti	Propellant flowrate for ith thrust event	kg/sec			Local	
xmtj	Propellant flowrate for jth thrust event	kg/sec			Local	
xmtl	Propellant flowrate for lth thrust event	kg/sec			Local	
yl(7,20)	Integrated values of adjoint equations of motion		dpir	input	Global	bakint
ylbt(15,20)	Partial derivatives of the constraints with respect to the thrust event time intervals		bkend	output input	Global	blamb
ylbw(15,20)	Partial derivatives of the constraints with respect to the weight drop event time intervals		bkend	input	Global	blamb
ylf(7,20)	Final value of yl array at the termination of the backward integration		bkend	output	Global	bake

```
      SUBROUTINE BKEND                                                  BKEN   1
C
C      TERMINATES ADJOINT RUN AND ADJUSTS INFLUENCE COEFFICIENTS
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      LOGICAL GLIMIT
C
      CHARACTER*60 HEAD
      CHARACTER*6 MISION

      COMMON/AGEN1/TIME(2,16),TAUT(15),TAUW(15),TZERO,TLIFT,TTILT,TMINH,
     *DTZ,TQ,TL,XMAUG,TNE(6,15),ENGDAT(8,15),S(15),WD(15),WJET(15),
     *HEAD,PRINT(15),STEP(15),HNOM(15),HMAX(15),PROP(6),TBEGR,TENDR,
     *CHRDOT,CHIRO,FAZ,BO(3,2),CBAXIL,TCHIR,VRCUT,FPRFAC,CHVEL

      COMMON/AGEN2/AA,SA,CA,ALF,ALFY,CHIP,SCHIP,CCHIP,CHIY,SCHIY,CCHIY,
     *CHIR,SCHIR,CCHIR,STH,CTH,STHL,CTHL,DPHIZ,RTHE,R,VR,XM,SW,SU,SV,Q,
     *VIV(25),DVARS(13),DELXDW(15,3),DELXDR(7,5),DELXD(15,7),WZERO,ALT,
     *XMACH,QDOT,FAA,FAN,A(3,3),CASE,CT2,UMF,A12W,A22W,A32W,XJEXT,BEU,
     *BYL,BEL,BHMAX(15),BHMN,BSTEP(15),HNMB,HMXB,TPOLY,XMIAD

      COMMON/AGEN3/ITHR,IWD,IXR,JUMP,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISION,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)

      COMMON/AUTO/KDB(40),SAVCP(40),XLAMB(40,20),DP2

      COMMON/BAKE/WISS(20,20),YLF(7,20)

      COMMON/BAKINT/BBANK(2),TBK,TBKD,YL(7,20),YLD(7,20),BSAVE(2700),
     *NBTRG1,TBV1,NBTRG2,TBV2,NBTRG3,TBV3,NBTRG4,TBV4,NBTRG5,TBV5,TEND,
     *DTB4,DU3,D23,D43,WISSD(20,20)

      COMMON/BGEN3/NOSS,JTB,NENT,NACT,NOM1,NP(7)

      COMMON/BLAMB/YLBT(15,20),YLBW(15,20),GAPHI(20),PROD(2,20)

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM

      COMMON/FMXX/FMXX,MSW,WG,SPOA

      COMMON/IPR/IPR,WPMX,ILAST,KDT(15),PRK4,PRTOT,iprop(6),prk5(5)

      COMMON/LBM/TIMLBM(30),THRLBM(30),WDTLBM(30),PLBM(2),MLBM1,MLBM2,
     *ALBM(2),BLBM(2),CLBM(2),VLBM(2)

      COMMON/MPS/TIMMPS(15),FTBL(15),THROT(15),GLIM(15),TAUALT(5),
     *TAUTBL(5),MPS1,MPS2,TGLIM(10,15),GLIMIT,tau_const(10,5),jthrot(15)

      common/mult/nchiot(6),tbias(6),nstage,nstg(15),wf(5),ibranch

      COMMON/NOMOR/NOMOR
```



```
      COMMON/OMSRCS/FOMS(15),FRCS(15),WDOMS(15),WDRCS(15),WDDUMP(15)

      COMMON/XTRA/XMD(15),CFR(15),F(15),CISP(15)

      COMMON/YLI/YLINT(7,20),PHITE(20),PSIREQ(20),KCDPHI(20),PSTR(20)

c**********************************************************************

      IF(NOMOR.EQ.2) GO TO 40

      IF(JTB.NE.1)CALL BSAVEG                                           BKEN  21

C     ADD WD LAMBDA BETA TO THRUST LAMBDA BETA                          BKEN  22

      IF(NWVNT.ne.0) then

         DO I=1,NWVNT
            K=NOWD(I)
            DO J=1,NOSS

C     ADDS WD LAMBDA BETAS TO APPROPRIATE THRUST LAMBDA BETAS           BKEN  26

               YLBT(K,J)=YLBT(K,J)+YLBW(I,J)
            enddo
         enddo

      endif

C  *****                                                              BKEN  28
C  ADD UP ALL LAMBS DOWNSTREAM                                        BKEN  29

      if(ibranch.eq.0) then

         KV=NVNT-1

    3    TEMP=1.

         DO K=1,NOSS
            YLBT(KV,K)=YLBT(KV,K)+YLBT(KV+1,K)*TEMP
         enddo

         KV=KV-1

         IF(KV.GT.0) GO TO 3

      else

         kv=nvnt-1
         kv1=0
         kv2=0

   41    do k=1,noss
            if(nstg(kv).eq.nstg(kv+1)) then

               ylbt(kv,k)=ylbt(kv,k)+ylbt(kv+1,k)
```

```
            if(nchiot(nstg(kv)).eq.0) then
               kv1=kv
            else
               kv2=kv
            endif

         else
            if(kv.eq.ibranch) then
               if(kv2.eq.0) kv2=kv+1
               ylbt(kv,k)=ylbt(kv,k)+ylbt(kv1,k)+ylbt(kv2,k)
            else
               if(nchiot(nstg(kv)).eq.nchiot(nstg(kv+1))) then
                  ylbt(kv,k)=ylbt(kv,k)+ylbt(kv+1,k)
               endif

               if(nchiot(nstg(kv)).eq.0) then
                  kv1=kv
               else
                  kv2=kv
               endif
            endif
         endif
      enddo

c     write(31,*) ' ylbt(',kv,')='
c     write(31,'(1x,9e14.7)') (ylbt(kv,k),k=1,noss)

      kv=kv-1

      if(kv.gt.0) go to 41

   endif

C****  TINKER LAMBDA BETAS IF CONNECTED                                  BKEN  36

   DO 5 I=1,NVNT                                                        BKEN  37

   IF(KDB(I).ne.0) then                                                 BKEN  38

      IF(IPR.eq.0.and.MSWCH(NVNT-1).eq.2) then

         INVT=NVNT-2

         TEMP=-TAUT(NVNT-1)*GLIM(INVT)/CISP(INVT)*TNE(1,I)*XMD(I)/
     *           (TNE(1,INVT)*XMD(INVT))

         TEMP1=WG*EXP(-GLIM(INVT)*(TIME(1,INVT+1)-TGLIM(1,invt))/
     *           CISP(INVT))*EXP(-VRCUT/(GZERO*CISP(INVT+1)))

         DO 29 K=1,NOSS

         IF(KCDPHI(K).eq.1) then

            YLBT(I,K)=TEMP1*TEMP/TAUT(NVNT-1)
```



```
         else

            YLBT(I,K)=YLBT(I,K)+TEMP*YLBT(NVNT-1,K)

         endif

29       CONTINUE

      endif

   endif

   IF(IRTLS.ne.0.and.IPR.ne.0) then

      J=IRTLS
      L=IPR

      XMTI=TNE(1,I)*XMD(I)+WDOMS(I)+WDRCS(I)+WDDUMP(I)
      XMTJ=TNE(1,J)*XMD(J)+WDOMS(J)+WDRCS(J)+WDDUMP(J)
      XMTL=TNE(1,L)*XMD(L)+WDOMS(L)+WDRCS(L)+WDDUMP(L)

      PTJTI=(-(WDOMS(I)+WDDUMP(I))+WDOMS(L)/XMTL*XMTI)/
     *        ((WDOMS(J)+WDDUMP(J))-WDOMS(L)/XMTL*XMTJ)

      IF(IRTFLG.EQ.1) PTJTI=-XMTI/XMTJ

      PTGTI=1./XMTL*(-XMTI-XMTJ*PTJTI)

      TEMP=TNE(1,I)*CFR(I)+TNE(1,J)*CFR(J)*PTJTI+TNE(1,L)*CFR(L)*PTGTI

      PTLTI=PTGTI+FMXX*(TEMP)

      DO 32 K=1,NOSS

      IF(KCDPHI(K).eq.1) then

         YLBT(I,K)=(1.-PRK4)/(CFR(L)/XMD(L)-PRK4)*TEMP

      else

         YLBT(I,K)=YLBT(I,K)+PTJTI*YLBT(J,K)+PTLTI*YLBT(L,K)

      endif

32    CONTINUE

      GO TO 5

   endif

   IF(IRTLS.ne.0.OR.I.le.IRTLS) then

      J=IRTLS
      L=IRTLS+1

      IF(MPSFLG(L).ne.4) then
```

```
            DO 34 K=L,NVNT

            IF(MPSFLG(K).EQ.4) GO TO 35

   34       CONTINUE
   35       L=K

         endif

         TEMP=(F(L)*TNE(1,L)+FOMS(L)+FRCS(L))/(GZERO*GLIM(L))*
     *        EXP(-GLIM(L)*(TEND-TGLIM(1,1)-TAUT(L+1))/CISP(L))*
     *        EXP(-GLIM(L+1)*TAUT(L+1)/CISP(L+1))

         XMTJ=TNE(1,J)*XMD(J)+WDOMS(J)+WDRCS(J)+WDDUMP(J)
         XMTL=TNE(1,L)*XMD(L)+WDOMS(L)+WDRCS(L)+WDDUMP(L)
         XMTI=TNE(1,I)*XMD(I)+WDOMS(I)+WDRCS(I)+WDDUMP(I)

         PTJTI=(-(WDOMS(I)+WDDUMP(I))+WDOMS(L)*XMTI/XMTL)/
     *          (WDOMS(J)+WDDUMP(J)-WDOMS(L)*XMTJ/XMTL)

         IF(IRTFLG.EQ.1) PTJTI=-XMTI/XMTJ

         DO 14 K=1,NOSS

         IF(KCDPHI(K).eq.1) then

            YLBT(I,K)=-TEMP*GLIM(L)/CISP(L)*(XMTI+XMTJ*PTJTI)/XMTL

         else

            YLBT(I,K)=YLBT(I,K)+PTJTI*YLBT(J,K)

         endif

   14    CONTINUE

      endif

      IF(KDT(I).ne.0) then                                          BKEN  39

         J=I+KDT(I)                                                 BKEN  40

         IF(MSW.ne.0) then

            TEMP=-TNE(1,I)*XMD(I)/(TNE(1,J)*XMD(J))+FMXX*
     *           (TNE(1,I)*CFR(I)-CFR(J)/XMD(J)*TNE(1,I)*XMD(I))
            TEMP1=1.
            TEMP2=0.

            IF(MSWCH(J+1).eq.2) then

               TEMP1=EXP(-VRCUT/(GZERO*CISP(J+1)))
               TEMP2=-TAUT(J+1)*GLIM(J)/CISP(J)*(TEMP+TNE(1,I)*
     *          XMD(I)/(TNE(1,J)*XMD(J)))
```



```
            endif

            DO 20 K=1,NOSS

            IF(KCDPHI(K).eq.1) then

               YLBT(I,K)=(1.-PRK4)/(CFR(J)/XMD(J)-PRK4*TEMP1-
     *                    (1.-TEMP1))*(TNE(1,I)*CFR(I)-CFR(J)/
     *                    XMD(J)*TNE(1,I)*XMD(I))*TEMP1

            else

               YLBT(I,K)=YLBT(I,K)+TEMP*YLBT(J,K)+TEMP2*YLBT(J+1,K)

            endif

   20       CONTINUE

         else

            TEMP=CFR(I)/CFR(J)*TNE(1,I)/TNE(1,J)

            IF(I.ge.ILAST.and.IPR.ne.0) then                        BKEN  42

               TEMPR=(1.-CFR(J)*XMD(I)/CFR(I)*PRK4/XMD(J))/(1.-PRK4) BKEN  43
               TEMP=TEMP*TEMPR                                      BKEN  44

            endif

            DO 10 K=1,NOSS                                          BKEN  45

            TEMP1=TEMP*YLBT(J,K)                                    BKEN  46

            IF(KCDPHI(K).eq.1) then                                 BKEN  47

               IF(I.GE.ILAST.AND.IPR.NE.0) TEMP1=TEMP1/TEMPR        BKEN  48

            endif

   10       YLBT(I,K)=YLBT(I,K)-TEMP1

         endif

C     ADJUST UNBOUNDED LAMBDA BETAS FOR PR                          BKEN  51

      else

         J=IPR                                                      BKEN  52

         IF(J.ne.0) then                                            BKEN  53

            IF(MSW.ne.0) then

               TEMP=-TNE(1,I)*XMD(I)/(TNE(1,J)*XMD(J))-FMXX*
     *              (CFR(J)/XMD(J)*TNE(1,I)*XMD(I))
               TEMP1=1.
```

```
            IF(MSWCH(J+1).EQ.2) TEMP1=EXP(-VRCUT/(GZERO*CISP(J+1)))

            DO 22 K=1,NOSS

            IF(KCDPHI(K).eq.1) then

               YLBT(I,K)=(1.-PRK4)/(CFR(J)/XMD(J)-PRK4*TEMP1-(1.-
     *                    TEMP1))*(-CFR(J)/XMD(J)*TNE(1,I)*XMD(I))*
     *                    TEMP1

            else

               YLBT(I,K)=YLBT(I,K)+TEMP*YLBT(J,K)

            endif

  22        CONTINUE

         else

            TEMP=XMD(I)/XMD(J)*TNE(1,I)/TNE(1,J)

            DO 12 K=1,NOSS

            IF(KCDPHI(K).ne.1) then                                    BKEN  56

               YLBT(I,K)=YLBT(I,K)+TEMP*PRK4*YLBT(J,K)/(1.-PRK4)

            endif

  12        CONTINUE

         endif

      endif

   endif

5  continue

C  *****                                                               BKEN  60
C*** PUT ACTIVE YLBTS INTO XLAMB                                       BKEN  61

  40 do i=1,nvnt
      do j=1,noss
        XLAMB(I,J)=YLBT(I,J)
      enddo
   enddo

   do i=1,7
      do j=1,noss
        YLF(I,J)=YL(I,J)
      enddo
   enddo
```



```
      CALL RTMRK                                                       BKEN  69

      RETURN                                                           BKEN  70
      END                                                              BKEN  71
```

3.27 Subroutine BSAVEG

3.27.1 Purpose

This subroutine saves impulse response functions and performs Simpson integration on the weighting matrix.

3.27.2 Variable Listing

3.27.3 Subroutines Called:

None

3.27.4 Calling Subroutines:

Name	Function
BKEND	Terminates the backward integration
BTHREV	Controls the thrust event logic during the backward integration
DPIR	Integration routine

3.27.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a11	Element of Hessian matrix				Local	
a12	Element of Hessian matrix				Local	
a22	Element of Hessian matrix				Local	
ahtb(210)	Previous pitch attitude tables	rad	bsaveg	output	Global	imp
chip	Pitch attitude angle	rad	bdr1i	input	Global	agen2
chiy	Yaw attitude angle	rad	bdr1i	input	Global	agen2
covl(7)	Lagrangian multipliers			output input	Global	covl
d23	Two thirds the value of dtb4			input	Global	bakint
d43	Four thirds the value of dtb4			input	Global	bakint
delts(7)	Time increment used to build pitch and yaw attitude tables during min-H		bsaveg	output	Global	imp
det	Determinate of Hessian matrix				Local	
dtb4	Time increment used with attitude tables during min-H	sec	bthrev	input	Global	bakint
du3	One third the value of dtb4		bthrev	input	Global	bakint
gaphi(20)	Temporary array		bsaveg	output	Global	blamb
gaptb(210,20	Partial derivatives of constraints with respect to pitch attitude		bsaveg	output	Global	imp
gnu(20)	Partial derivative of payoff with respect to constraints		anewch	input	Global	gnu
gsphi	Temporary variable				Local	
gsptb(210,20	Partial derivatives of constraints with respect to yaw attitude		bsaveg	output	Global	imp
hv11	Element of inverse of Hessian matrix				Local	
hv12	Element of inverse of Hessian matrix				Local	
hv22	Element of inverse of Hessian matrix				Local	
ithr	Thrust event index number		athrev	output	Global	agen3
j j	Index				Local	
jtb	Index for control table points		bsaveg	output input	Global	bgen3
kat	Indicator for positive definite values of huu matrix		bsaveg	output	Global	agen3
kcytab	Attitude control table index		bsaveg	output	Global	agen3
kjtb	Index for time and pitch and yaw control table index		bthrev	input	Global	imp
km1	Index				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
kwta(7)	Flag indicating pitch only (1) or pitch and yaw (2) attitude control		ainit	input	Global	agen3
ll	Index used in Simpson's integration				Local	
m	Index				Local	
nbtrg4	Impulse response function trigger for backwards integration			output	Global	bakint
nchiot(6)	Optimization flag used with branch trajectory option		ainit	input	Global	mult
ncnrs(5)	Number of constraints at restart		ainit	input	Global	rest
nom1	Index indicating first iteration pass			input output	Global	bgen3
noss	Number of constraints		mastre	input	Global	bgen3
nossm1	Index (noss-1)				Local	
np(7)	Number of points in attitude tables during min-H phases		ainit	input	Global	bgen3
nstage	Index number of current stage		athrev	input	Global	mult
nvrst(5)	Array specifying the thrust events where intermediate constraints will be imposed		ainit	output	Global	enput
pdxcp	Partial derivative of X acceleration component with respect to pitch attitude angle			input	Global	calif
pdxcpp	Second partial derivative of X acceleration component with respect to pitch attitude angle			input	Global	calif
pdxcpy	Second partial derivative of X acceleration component with respect to pitch and yaw attitude angles			input	Global	calif
pdxcy	Partial derivative of X acceleration component with respect to yaw attitude angle			input	Global	calif
pdxcyy	Second partial derivative of X acceleration component with respect to yaw attitude angle			input	Global	calif
pdycp	Partial derivative of Y acceleration component with respect to pitch attitude angle			input	Global	calif
pdycpp	Second partial derivative of Y acceleration component with respect to pitch attitude angle			input	Global	calif
pdycpy	Second partial derivative of Y			input	Global	calif

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
	acceleration component with respect to pitch and yaw attitude angles					
pdycy	Partial derivative of Y acceleration component with respect to yaw attitude angle			input	Global	calif
pdycyy	Second partial derivative of Y acceleration component with respect to yaw attitude angle			input	Global	calif
pdzcp	Partial derivative of Z acceleration component with respect to pitch attitude angle			input	Global	calif
pdzcpp	Second partial derivative of Z acceleration component with respect to pitch attitude angle			input	Global	calif
pdzcpy	Second partial derivative of Z acceleration component with respect to pitch and yaw attitude angles			input	Global	calif
pdzcy	Partial derivative of Z acceleration component with respect to yaw attitude angle			input	Global	calif
pdzcyy	Partial derivative of Z acceleration component with respect to yaw attitude angle			input	Global	calif
prod(2,20)	Tempoary array			output input	Global	blamb
sitb(210)	Previous yaw attitude table during min-H			output	Global	imp
t	Time from lift-off	sec	dpir	input	Global	forint
tbv4	Time to next determination of impulse response function	sec	bsaveg	output input	Global	bakint
tdet	Temporary variable				Local	
temp	Temporary variable				Local	
ts(7)	Start times to begin min-H phases		bsaveg	output	Global	imp
wah	Temporary variable				Local	
wiss(20,20)	Weighting matrix		bsaveg	output	Global	bake
wissd(20,20)	Weighting matrix derivative		bsaveg	output	Global	bakint
xjext	Flag indicating that payoff will be maximized (=1) or minimized (=-1)		ainit	input	Global	agen2
yl(7,20)	Integrated values of adjoint equations of motion		dpir	input	Global	bakint

```
      SUBROUTINE BSAVEG                                                 BSAV   1
C*********************************************************************
C       SAVES IMPULSE RESPONCE FUNCTIONS AND PERFORMS SIMPSON INTEGRATION
C      ON PRODUCTS OF IMPULSE RESPONCE FUNCTIONS
C*********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      CHARACTER*60 HEAD
      CHARACTER*6 MISION
      character*12 header

      COMMON/AGEN1/TIME(2,16),TAUT(15),TAUW(15),TZERO,TLIFT,TTILT,TMINH,
     *DTZ,TQ,TL,XMAUG,TNE(6,15),ENGDAT(8,15),S(15),WD(15),WJET(15),
     *HEAD,PRINT(15),STEP(15),HNOM(15),HMAX(15),PROP(6),TBEGR,TENDR,
     *CHRDOT,CHIRO,FAZ,BO(3,2),CBAXIL,TCHIR,VRCUT,FPRFAC,CHVEL

      COMMON/AGEN2/AA,SA,CA,ALF,ALFY,CHIP,SCHIP,CCHIP,CHIY,SCHIY,CCHIY,
     *CHIR,SCHIR,CCHIR,STH,CTH,STHL,CTHL,DPHIZ,RTHE,R,VR,XM,SW,SU,SV,Q,
     *VIV(25),DVARS(13),DELXDW(15,3),DELXDR(7,5),DELXD(15,7),WZERO,ALT,
     *XMACH,QDOT,FAA,FAN,A(3,3),CASE,CT2,UMF,A12W,A22W,A32W,XJEXT,BEU,
     *BYL,BEL,BHMAX(15),BHMN,BSTEP(15),HNMB,HMXB,TPOLY,XMIAD

      COMMON/AGEN3/ITHR,IWD,IXR,JUMP,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISION,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)

      COMMON/BAKE/WISS(20,20),YLF(7,20)

      COMMON/BAKINT/BBANK(2),TBK,TBKD,YL(7,20),YLD(7,20),BSAVE(2700),
     *NBTRG1,TBV1,NBTRG2,TBV2,NBTRG3,TBV3,NBTRG4,TBV4,NBTRG5,TBV5,TEND,
     *DTB4,DU3,D23,D43,WISSD(20,20)

      COMMON/BGEN3/NOSS,JTB,NENT,NACT,NOM1,NP(7)

      COMMON/BLAMB/YLBT(15,20),YLBW(15,20),GAPHI(20),PROD(2,20)

      COMMON/CALIF/PXX,PYX,PZX,PXY,PYY,PZY,PXZ,PYZ,PZZ,PXW,PYW,PZW,PXU,
     *PYU,PZU,PXV,PYV,PZV,PXM,PYM,PZM,PDXCP,PDYCP,PDZCP,PDXCY,PDYCY,PDZC
     *Y,PDXCPP,PDYCPP,PDZCPP,PDXCPY,PDYCPY,PDZCPY,PDXCYY,PDYCYY,PDZCYY

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM

      COMMON/COVL/COVL(7)

      common/enput/psirst(6,5),header(20),nvrst(5),kcdres(6,5),last

      COMMON/FORINT/HBANK(2),T,TT,W,U,V,X,Y,Z,XMI,ZAP(18),DVAR(25),FSAVE
     *(625),NTRG1,TV1,NTRG2,TV2,NTRG3,TV3,NTRG4,TV4,NTRG5,TV5,NTRG6,TV6,
     *NTRG7,TV7,NTRG8,TV8,NTRG9,TV9,NTRG10,TV10,NTRG11,TV11,NTRG12,TV12,
     *NTRG13,TV13,KIND,KRDER,EU,EL,AYL,HMX,HMN,HNM
```



```
      COMMON/GNU/GNU(20)

      COMMON/IMP/GAPTB(210,20),GSPTB(210,20),AHTB(210),SITB(210),TS(7)
     *,DELTS(7),KJTB

      common/mult/nchiot(6),tbias(6),nstage,nstg(15),wf(5),ibranch

      common/rest/jstr,ncnrs(5),ntcn,tr(5),lsb,nf8,nopar,wibt(15)

      DIMENSION GSPHI(20)                                               BSAV  34

      if(jtb.le.1) return

      j=nchiot(nstage)

      k=1
      if(j.ne.0) then
         k=noss+1
         do m=j,1,-1
            if(ithr.le.nvrst(m)) k=k-ncnrs(m)
         enddo
      endif

      TBV4=TBV4+DTB4                                                    BSAV  58
      JTB=JTB-1                                                         BSAV  59
      JJ=KJTB+JTB                                                       BSAV  60
c*********************************************************************
C     IMPULSE RESPONSE FUNCTIONS AND I-SY-SY DOTS                       BSAV  61
c*********************************************************************
      IF(NOM1.eq.0) then

         IF(KWTA(KCYTAB).ne.2) then

C     GET HERE ONLY ON NOMINAL WITH KWTA=3                              BSAV  64

            HV11=1./SQRT(PDXCPP*PDXCPP+PDYCPP*PDYCPP+PDZCPP*PDZCPP)
            HV12=0.                                                     BSAV  66
            HV22=1./SQRT(PDXCYY*PDXCYY+PDYCYY*PDYCYY+PDZCYY*PDZCYY)

            GO TO 8                                                     BSAV  68

         endif

         WAH=1./SQRT(PDXCPP*PDXCPP+PDYCPP*PDYCPP+PDZCPP*PDZCPP)

    2    DO I=1,NOSS
            GAPHI(I)=YL(1,I)*PDXCP+YL(2,I)*PDYCP+YL(3,I)*PDZCP
            if(i.gt.1.and.i.lt.k) gaphi(i)=0.
            DO J=1,I
               WISSD(I,J)=WAH*GAPHI(I)*GAPHI(J)
            enddo
```

```
         enddo

         GO TO 11                                                    BSAV  75

      endif

c*********************************************************************

C     GET COV LAMBDAS                                               BSAV  76

c*********************************************************************

      NOSSM1=NOSS-1                                                 BSAV  77

      km1=k-1
      if(km1.le.0) km1=1

      DO I=1,7
         COVL(I)=YL(I,1)
         DO J=km1,NOSSM1
            COVL(I)=COVL(I)+GNU(J)*YL(I,J+1)
         enddo
      enddo

c*********************************************************************

C     GOT COV LAMBDAS                                               BSAV  86

c*********************************************************************

      IF(KWTA(KCYTAB).ne.3) then

         WAH=1./(-XJEXT*(COVL(1)*PDXCPP+COVL(2)*PDYCPP+COVL(3)*PDZCPP))
         TEMP=1./SQRT(COVL(1)**2+COVL(2)**2)                         BSAV  89

         GO TO 2

      endif

c*********************************************************************

C     3-D MIN-H                                                     BSAV  94
C     CHECK HUU FOR POSITIVE DEFINITE                               BSAV  95

c*********************************************************************

      A11=COVL(1)*PDXCPP+COVL(2)*PDYCPP+COVL(3)*PDZCPP
      A22=COVL(1)*PDXCYY+COVL(2)*PDYCYY+COVL(3)*PDZCYY
      A12=COVL(1)*PDXCPY+COVL(2)*PDYCPY+COVL(3)*PDZCPY

      A11=-A11*XJEXT
      A12=-A12*XJEXT
      A22=-A22*XJEXT

      DET=A11*A22-A12*A12                                           BSAV  99
```



```
      TDET=COVL(1)*COVL(1)+COVL(2)*COVL(2)                          BSAV 100

      IF(A11.le.0..or.A22.le.0.) then

         DET=TDET                                                   BSAV 102
         A22=(DET+COVL(3)*COVL(3))**.5

         IF(A22.EQ.0.0 ) THEN
            A11 = 0.0
         ELSE
            A11=DET/A22                                             BSAV 104
         ENDIF

         A12=0.                                                     BSAV 105
         KAT=1

         hv11=0.
         hv12=0.
         hv22=0.

      endif

c*********************************************************************

C       calculate HUU INVERSE

c*********************************************************************

      IF(DET .EQ. 0.0 ) THEN
         HV11 = 0.0
         HV12 = 0.0
         HV22 = 0.0
      ELSE
         HV11= A22/DET                                              BSAV 108
         HV12=-A12/DET                                              BSAV 109
         HV22= A11/DET                                              BSAV 110
      ENDIF
C                                                                   BSAV 111
    8 DO I=1,NOSS
         GAPHI(I)=YL(1,I)*PDXCP+YL(2,I)*PDYCP+YL(3,I)*PDZCP
         GSPHI(I)=YL(1,I)*PDXCY+YL(2,I)*PDYCY+YL(3,I)*PDZCY
         if(i.gt.1.and.i.lt.k) then
            gaphi(i)=0.
            gsphi(i)=0.
         endif
         PROD(1,I)=HV11*GAPHI(I)+HV12*GSPHI(I)
         PROD(2,I)=HV12*GAPHI(I)+HV22*GSPHI(I)

         DO J=1,I
            WISSD(I,J)=GAPHI(J)*PROD(1,I)+GSPHI(J)*PROD(2,I)
         enddo
      enddo

      DO I=1,NOSS
         GAPTB(JJ,I)=PROD(1,I)
         GSPTB(JJ,I)=PROD(2,I)
```

```
      enddo

      GO TO 13                                                          BSAV 123

   11 DO I=1,NOSS
         GAPTB(JJ,I)=WAH*GAPHI(I)
      enddo

   13 AHTB(JJ)=CHIP                                                     BSAV 126
      SITB(JJ)=CHIY

C  ** INTEGRATE ISYSY BY SIMPSONS RULE--                              BSAV 127
C  **                    A=H/3*(Y1+4*Y2+2*Y3+....+4*YN-1 +YN)        BSAV 128

      IF(JTB.ge.NP(KCYTAB)) then                                        BSAV 129

C     GET HERE AT FIRST POINT  (MULT BY 1/3)                           BSAV 130

         LL=2
         DO I=1,NOSS
            DO J=1,I
               WISS(I,J)=WISS(I,J)+ DU3  *WISSD(I,J)
            enddo
         enddo
         RETURN                                                         BSAV 135

      endif

      IF(LL.ne.1) then                                                  BSAV 136

C     GET HERE AT AN EVEN POINT  (MULT BY 4/3)                         BSAV 137

         LL=1
         DO I=1,NOSS
            DO J=1,I
               WISS(I,J)=WISS(I,J)+ D43  *WISSD(I,J)
            enddo
         enddo

         RETURN                                                         BSAV 142

      endif

      IF(JTB.ne.1) then                                                 BSAV 143

C     GET HERE AT AN ODD POINT  (MULT BY 2/3)                          BSAV 144

         LL=2
         DO I=1,NOSS
            DO J=1,I
               WISS(I,J)=WISS(I,J)+D23*WISSD(I,J)
            enddo
         enddo

         RETURN                                                         BSAV 149
```



```
      endif

C     GET HERE AT LAST POINT (JTB=1)  (MULT BY 1/3)                    BSAV 150

      DO I=1,NOSS
         DO J=1,I
            WISS(I,J)=WISS(I,J)+ DU3  *WISSD(I,J)
         enddo
      enddo

      NBTRG4=-1                                                         BSAV 154
      TS(KCYTAB)=T                                                      BSAV 155
      DELTS(KCYTAB)=DTB4                                                BSAV 156

      RETURN                                                            BSAV 157
      END                                                               BSAV 158
```

3.28 Subroutine BTHREV

3.28.1 Purpose

This subroutine controls the thrust event logic in the backward integration. The entry routines BWDEV and BGLIM are included in this subroutine to control weight drop events and acceleration limits, respectively.

3.28.2 Variable Listing

3.28.3 Subroutines Called:

Name	Function
BKEND	Terminates the backward integration
BSAVEG	Calculates the impulse response functions and integrates the weighting matrix
RTMRK	Terminates the integration routine

3.28.4 Calling Subroutines:

Name	Function
BAKRN	Controls the logic for the backward integration and calls the integration package
DPIR	Integration routine

3.28.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
atbdwt(15,2)	Weight loss table for center of gravity tables	kg	ainit	input	Global	body
atxcg(15,2)	Storage array for vertical center of gravity tables	m	ainit	input	Global	body
atycg(15,2)	Storage array for longitudinal center of gravity table	m	ainit	input	Global	body
axlen(2)	Aerodynamic reference length	m		input	Global	body
axref(2)	Vertical aerodynamic moment reference point	m		input	Global	body
ayref(2)	Longitudinal aerodynamic moment reference point	m	ainit	input	Global	body
bstep(15)	Backward integration stepsize	sec		input	Global	agen2
d23	Two thirds the value of dtb4			output	Global	bakint
d43	Four thirds the value of dtb4			output	Global	bakint
delxd(15,7)	State derivative storage at thrust events		athrev	input	Global	agen2
delxdb(7)	State derivative storage for branch trajectory		athrev	input	Global	delxdb
delxdw(15,3)	State derivative discontinuity storage at weight grop events		atilt	input	Global	agen2
drngdt(2)	Time derivative of range from launch site		acstop	input	Global	drngdt
dtb4	Time increment used with attitude tables during min-H	sec	bthrev	output input	Global	bakint
du3	One third the value of dtb4		bthrev	output input	Global	bakint
engdat(8,15)	Engine data matrix		ainit	input	Global	agen1
engines(15,1	This array indicates which engine from the ENGDAT array is being used for a certain thrust event. (0- not being used, 1-being used)		ainit	input	Global	engines
hmxb	Maximum step size for backward integration	sec		output	Global	agen2
hnmb	Nominal step size for backward stepsize	sec		output	Global	agen2
ibranch	Flag indicating the terminal thrust event of the first trajectory of the branch trajectory			output	Global	mult
ithr	Thrust event index number		athrev	output input	Global	agen3
iwd	Index for weight drop events		bthrev	output	Global	agen3

Variable	Variable Definition	Units	Source	I/O input	Status	Common Block
jstart(6)	Index for use with branch trajectory option		ainit	input	Global	istart
jtb	Index for control table points		bthrev	output	Global	bgen3
jthrot(15)	Index indicating number of times the acceleration limit has been reached within a given thrust event		atilt(gli	input output	Global	mps
jump	Jump start flag		ainit	input	Global	agen3
kcdphi(20)	Terminal constraint selection flag array		ainit	input	Global	yli
kcytab	Attitude control table index		bthrev	output input	Global	agen3
kjtb	Index for time and pitch and yaw control table index		bthrev	output	Global	imp
ksave	Index				Local	
lstge(15)	Aerodynamic coefficient flag to distinguish stages		ainit	input	Global	agen3
m	Index				Local	
mblb1	Pointer index for LBM propulsion lookup for backward integration			output	Global	lbmmbl
mblb2	Previous index for LBM propulsion lookup for backward integration			output	Global	lbmmbl
mbs1	Pointer index for aerodynamic lookup in backwards integration			output	Global	bodyb
mbs2	Previous index for aerodynamic interpolation in backwards integration			output	Global	bodyb
mbs21	Pointer index for incremental areodynamic data			output	Global	delmbs
mbs3	Pointer index for base force interpolation in backwards integration			output	Global	bodyb
mbs31	Previous index for incremental base force interpolation			output	Global	delmbs
mbs4	Previous index for base force interpolation in backwards integration			output	Global	bodyb
mbs5	Index for state interpolation			output	Global	bodyb
mbs7	Pointer index for center of gravity interpolation in backwards integration			output	Global	bodyb
mbs8	Previous index for center of gravity interpolation in backwards integration			output	Global	bodyb
mbs9	Pointer index for wind parameter interpolation in backwards integration			output	Global	bodyb

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
minh	Flag to indicate thrust event which begins min-h attitude optimization		ainit	input	Global	agen3
mombal	Flag to specify the use of moment balance equations		ainit	input	Global	mombal
mpsflg(15)	Flag indicating the type of MPS thrust simulation		ainit	output	Global	agen3
n	Index				Local	
nbgct(7)	Index indicating beginning of min-H phases		ainit	input	Global	agen3
nbtrg3	Weight drop event trigger for backward integration			output	Global	bakint
nbtrg4	Impulse response function trigger for backwards integration			output	Global	bakint
nbtrg5	Acceleration limit trigger for backwards integration			output	Global	bakint
nchiot(6)	Optimization flag used with branch trajectory option		ainit	input	Global	mult
ncnrs(5)	Number of constraints at restart		ainit	input	Global	rest
nendct(7)	Index indicating end of min-H phases		ainit	input	Global	agen3
nn	Index corresponding to nbgct				Local	
nomor	Index indicating end of integration			output	Global	nomor
noss	Number of constraints		bakrn	input	Global	bgen3
np(7)	Number of points in attitude tables during min-H phases		ainit	input	Global	bgen3
nstag	Index number of current stage				Local	
nstage	Index number of current stage		athrev	input output	Global	mult
nstg(15)	Internal index which relates thrust event to stage number		ainit	input	Global	mult
nsyst(15)	Index array which specifies the type of engines that will be used for each thrust event		ainit	input	Global	agen3
nvnt	Number of thrust events		ainit	output	Global	agen3
nvrst(5)	Array specifying the thrust events where intermediate constraints will be imposed		ainit	output	Global	enput
sp	Index				Local	
tbdwt(15)	Weight overboard table for center of gravity tables	kg	bthrev	output	Global	bodyb

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
tbk	Time used in backward integration	sec		input	Global	bakint
tbv2	Thrust event trigger time during backward integration		bthrev	output	Global	bakint
tbv3	Weight drop event trigger time during backward integration		bthrev	output	Global	bakint
tbv4	Time to next determination of impulse response function	sec	bthrev	output	Global	bakint
tbv5	Time of acceleration limit event	sec	bthrev	output	Global	bakint
tend	Termination time of forward trajectory (start of backward intgration)	sec	bakrn	input	Global	bakint
tglim(10,15)	Time of acceleration limit	sec	atilt	input	Global	mps
time(2,16)	Actual times for thrust events measured from launch	sec	aforun	input	Global	agen1
tne(6,15)	Array which defines the number of engines per thrust event		ainit	input	Global	agen1
tq	Time that the minh phase is initiated	sec	bdr1i	input	Global	agen1
txcg(15)	Vertical center of gravity table	m	bthrev	output	Global	bodyb
tycg(15)	Longitudinal center of gravity table	m	bthrev	output	Global	bodyb
wd(15)	Weight dropped during weight drop event	kg	ainit	input	Global	agen1
wint	Initial vehicle weight	kg	bdr1i	output	Global	olow
wjet(15)	Jettison weight per thrust event	kg	ainit	input	Global	agen1
wzero	Initial vehicle weight	lbs	anewch	input	Global	agen2
xgpa	Vertical component of gimbal points of equivalent booster engines	m	athrev	output input	Global	avggp
xlen	Reference length	m	bthrev	output	Global	bodyb
xmaug	Auxiliary vehicle mass	kg	athrev	output input	Global	agen1
xmaug_save	Saved value of auxiliary mass used for branch trajectory option	kg	athrev	input	Global	xmaug_save
xref	Vertical component of moment reference point		bthrev	output	Global	bodyb
ygpa	Longitudinal component of gimbal point of booster equivalent engine	m	athrev	output input	Global	avggp
yl(7,20)	Integrated values of adjoint equations of motion		dpir	input	Global	bakint
ylbt(15,20)	Partial derivatives of the constraints with respect to the thrust event time intervals		bthrev	output	Global	blamb

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ylbw(15,20)	Partial derivatives of the constraints with respect to the weight drop event time intervals		bthrev	output	Global	blamb
yref	Longitudinal component of moment reference point	m	bthrev	output	Global	bodyb
zgpa	Lateral component of gimbal point of booster equivalent engines	m	athrev	output input	Global	avggp

```
      SUBROUTINE BTHREV                                                 BTHR   1
C*********************************************************************
C       BACKWARD THRUST EVENT CONTROL ROUTINE
C*********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      LOGICAL VISC,glimit,mombal

      integer engines

      CHARACTER*60 HEAD
      CHARACTER*6 MISION
      character*12 HEADER

      COMMON/AGEN1/TIME(2,16),TAUT(15),TAUW(15),TZERO,TLIFT,TTILT,TMINH,
     *DTZ,TQ,TL,XMAUG,TNE(6,15),ENGDAT(8,15),S(15),WD(15),WJET(15),
     *HEAD,PRINT(15),STEP(15),HNOM(15),HMAX(15),PROP(6),TBEGR,tendr,
     *CHRDOT,CHIRO,FAZ,BO(3,2),CBAXIL,TCHIR,VRCUT,FPRFAC,CHVEL

      COMMON/AGEN2/AA,SA,CA,ALF,ALFY,CHIP,SCHIP,CCHIP,CHIY,SCHIY,CCHIY,
     *CHIR,SCHIR,CCHIR,STH,CTH,STHL,CTHL,DPHIZ,RTHE,R,VR,XM,SW,SU,SV,Q,
     *VIV(25),DVARS(13),DELXDW(15,3),DELXDR(7,5),DELXD(15,7),WZERO,ALT,
     *XMACH,QDOT,FAA,FAN,A(3,3),CASE,CT2,UMF,A12W,A22W,A32W,XJEXT,BEU,
     *BYL,BEL,BHMAX(15),BHMN,BSTEP(15),HNMB,HMXB,TPOLY,XMIAD

      COMMON/AGEN3/ITHR,IWD,IXR,JUMP,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISION,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)

      COMMON/AGEN5/CHIBAS,CHPBAS,CHYBAS,CHISAV,TXX,TYY,TZZ,AMX,AMY,AMZ,
     *TMZ,THMFA,SIDE,WMAG,AZW,FANC,FLATC,FOM,VISC

      COMMON/AVGGP/XGPA,YGPA,ZGPA

      COMMON/BAKINT/BBANK(2),TBK,TBKD,YL(7,20),YLD(7,20),BSAVE(2700),
     *NBTRG1,TBV1,NBTRG2,TBV2,NBTRG3,TBV3,NBTRG4,TBV4,NBTRG5,TBV5,TEND,
     *DTB4,DU3,D23,D43,WISSD(20,20)

      COMMON/BGEN3/NOSS,JTB,NENT,NACT,NOM1,NP(7)

      COMMON/BLAMB/YLBT(15,20),YLBW(15,20),GAPHI(20),PROD(2,20)

      COMMON/BODY/ATBDWT(15,2),ATXCG(15,2),ATYCG(15,2),AXLEN(2),AXREF(2)
     *,AYREF(2),KPI(2),KYI(2),API(10,2),AYI(10,2)

      COMMON/BODYB/TBDWT(15),TXCG(15),TYCG(15),XLEN,XREF,YREF,MBS1,MBS2,
     *MBS3,MBS4,MBS5,MBS6,MBS7,MBS8,MBS9,MBS10

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM
```



```
      COMMON/DELMBS/MBS21,MBS30,MBS31

      common/delxdb/delxdb(7)

      COMMON/DRNGDT/DRNGDT(2)

      common/engines/engines(15,15)

      COMMON/ENPUT/PSIRST(6,5),HEADER(20),NVRST(5),KCDRES(6,5),LAST

      COMMON/FORINT/HBANK(2),T,TT,W,U,V,X,Y,Z,XMI,ZAP(18),DVAR(25),FSAVE
     *(625),NTRG1,TV1,NTRG2,TV2,NTRG3,TV3,NTRG4,TV4,NTRG5,TV5,NTRG6,TV6,
     *NTRG7,TV7,NTRG8,TV8,NTRG9,TV9,NTRG10,TV10,NTRG11,TV11,NTRG12,TV12,
     *NTRG13,TV13,KIND,KRDER,EU,EL,AYL,HMX,HMN,HNM

      COMMON/LBMMBL/MBLB1,MBLB2

      COMMON/IMP/GAPTB(210,20),GSPTB(210,20),AHTB(210),SITB(210),TS(7)
     *,DELTS(7),KJTB

      common/istart/jstart(6),istart(2),startv(40,2),svxinc,svxnod,svazre

      common/mombal/mombal

      common/mps/timmps(15),ftbl(15),throt(15),glim(15),taualt(5),
     *tautbl(5),mps1,mps2,tglim(10,15),glimit,tau_const(10,5),jthrot(15)

      common/mult/nchiot(6),tbias(6),nstage,nstg(15),wf(5),ibranch

      COMMON/NOMOR/NOMOR

      COMMON/OLOW/OLOW,WINT

      common/rest/jstr,ncnrs(5),ntcn,tr(5),lsb,nf8,nopar,wibt(15)

      common/xmaug_save/xmaug_save

      COMMON/YLI/YLINT(7,20),PHITE(20),PSIREQ(20),KCDPHI(20),PSTR(20)
c*********************************************************************

      IF(ITHR.EQ.JUMP.OR.ITHR.EQ.MINH) then

         call bkend

         return

      endif
c*********************************************************************

C***  CHECK FOR END(BEGINNING) OF CHI TABLE BEFORE UPDATING THRUST      BTHR  35
C     CHECK ON CHI TABLES                                              BTHR  36

c*********************************************************************
```

```
      IF(KCYTAB.ne.1) then

         IF(NENDCT(KCYTAB-1).EQ.0) KCYTAB=KCYTAB-1
         IF(NENDCT(KCYTAB-1).eq.ITHR) then

C  *  HAVE ARRIVED AT THE END OF A CHI TABLE                          BTHR  39

            IF(JTB.NE.1) CALL BSAVEG

            KCYTAB=KCYTAB-1                                          BTHR  41
            NBTRG4=1                                                 BTHR  42
            TBV4=TBK                                                 BTHR  43
            JTB=NP(KCYTAB)+1                                         BTHR  44
            KJTB=(KCYTAB-1)*30
            SP=NP(KCYTAB)-1                                          BTHR  46
            NN=NBGCT(KCYTAB)                                         BTHR  47
            N=NENDCT(KCYTAB)                                         BTHR  48

            DTB4=(TIME(1,N)-DMAX1(TQ,TIME(1,NN)))/SP
            DU3=DTB4/3.                                              BTHR  50
            D23=DU3*2.                                               BTHR  51
            D43=DU3*4.

         endif

      endif

      J=NBGCT(KCYTAB)                                                BTHR  53

      IF(J.EQ.ITHR.AND.JTB.NE.1) CALL BSAVEG                         BTHR  54

C     END OF CHI TABLE CHECK                                         BTHR  55

      ITHR=ITHR-1                                                    BTHR  56

      nstag=nstage

      nstage=nstg(ithr)

      HNMB=BSTEP(ITHR)                                               BTHR  57
      HMXB=2.*PRINT(ITHR)                                            BTHR  58

c*********************************************************************

c       turn on acceleration limit trigger if proper

c*********************************************************************

      nbtrg5=-2

      IF((mpsflg(ithr).eq.4.or.mpsflg(ithr).eq.5).and.
     *jthrot(ithr).gt.0) then

         NBTRG5=2
         tbv5=tend-tglim(jthrot(ithr),ithr)
```



```
      endif

c*********************************************************************

c       update total jettison weight

c*********************************************************************

      XMAUG=XMAUG+WJET(ITHR)                                         BTHR  61

      if(nstag.ne.nstage.and.jstart(nstag).ne.0) xmaug=xmaug_save

c*********************************************************************

c       calculate the location and number of mps engines for moment
c       balance calculations.
c       calculate the composite gimbal points of the two engine
c       equivalence

c*********************************************************************

      XGPA=0.
      YGPA=0.
      ZGPA=0.

      if(mombal) then
         do i=1,15

            if(nsyst(i).eq.1.and.engines(i,ithr).eq.1) then
               XGPA=XGPA+ENGDAT(3,I)
               YGPA=YGPA+ENGDAT(1,I)
               ZGPA=ZGPA+ABS(ENGDAT(2,I))
            endif
         enddo

         if(tne(1,ithr).gt.0.) then
            XGPA=XGPA/TNE(1,ITHR)
            YGPA=YGPA/TNE(1,ITHR)
            ZGPA=ZGPA/TNE(1,ITHR)
         endif

      endif

c*********************************************************************

c       initialilize the indices for interpolation and update
c       aerodynamic parameters

c*********************************************************************

      K=LSTGE(ITHR)

      IF(ITHR.eq.NVNT.or.K.ne.LSTGE(ITHR+1).or.nstag.ne.nstage) then

         IF(K.EQ.1) WINT=WZERO
```

```
         MBS1=1
         MBS2=0
         MBS3=1
         MBS4=0
         mbs5=1
         MBS7=1
         MBS8=0
         MBS9=1
         MBS10=0
         MBLB1=1
         MBLB2=0
         MBS21=1
         MBS30=1
         MBS31=0

         XREF=AXREF(K)
         YREF=AYREF(K)
         XLEN=AXLEN(K)

         DO 99 I=1,15

         TBDWT(I)=ATBDWT(I,K)*.45359237
         TXCG(I)=ATXCG(I,K)
         TYCG(I)=ATYCG(I,K)
   99    CONTINUE

      endif
c*************************************************************************
C     GET LAMBDA BETAS                                                  BTHR  71
c*************************************************************************

      j=nchiot(nstage)

      k=1
      if(j.gt.0) then
         k=noss+1
         do m=j,1,-1
            if(ithr.le.nvrst(m)) k=k-ncnrs(m)
         enddo
         ksave=k
      endif
      if(nchiot(nstage).gt.nchiot(nstag)) k=1

      DO 2 I=1,NOSS

      YLBT(ITHR,I)=0.                                                   BTHR  73

      if(i.eq.1.or.i.ge.k) then

         IF(KCDPHI(I).GE.18.AND.KCDPHI(I).LE.20.AND.ITHR.EQ.NVNT)
     *      YLBT(ITHR,I)=DRNGDT(1)

         DO 102 J=1,7
```



```
         if(nchiot(nstage).gt.nchiot(nstag)) then

            ylbt(ithr,1)=delxd(ithr,7)

            go to 102

         endif

         if(ithr.eq.ibranch) then

            if(i.eq.1) then

               ylbt(ithr,i)=ylbt(ithr,i)+delxdb(j)*yl(j,i)+
     *                      delxd(ithr,j)*yl(j,i)

            else if(i.gt.1.and.i.lt.ksave) then

               ylbt(ithr,i)=ylbt(ithr,i)+delxdb(j)*yl(j,i)

            else if(i.ge.ksave) then

               ylbt(ithr,i)=ylbt(ithr,i)+delxd(ithr,j)*yl(j,i)

            endif

         else

            YLBT(ITHR,I)=YLBT(ITHR,I)+DELXD(ITHR,J)*YL(J,I)

         endif
102      continue

      endif

2     continue
c*************************************************************************
C     GOT LAMBDA BETAS                                                  BTHR  77
c*************************************************************************

      TBV2=TEND-TIME(1,ITHR)                                            BTHR  78

      DO K=1,5
         IF(ITHR.eq.NVRST(K)) then

            NOMOR=1

            CALL RTMRK

            return

         endif
```

```
      enddo

      RETURN
C*********************************************************************
      ENTRY BWDEV
C*********************************************************************
C      TRIGGER ROUTINE FOR BACKWARD WEIGHT DROP EVENT
      IWD=IWD-1                                                  BWDE  24
      XMAUG=XMAUG+WD(IWD)                                        BWDE  25
C     GET LAMBDA BETAS                                           BWDE  26
      DO I=1,NOSS
         YLBW(IWD,I)=0.
         DO J=1,3
            YLBW(IWD,I)=YLBW(IWD,I)+DELXDW(IWD,J)*YL(J,I)
         enddo
      enddo
      IF(IWD.ne.1) then
         TBV3=TEND-TIME(2,IWD-1)
      else
         NBTRG3=-2
      endif
      RETURN
C*********************************************************************
      ENTRY BGLIM
C*********************************************************************
C      TRIGGER ROUTINE FOR ACCELERATION LIMIT IN BACKWARD INTEGRATION
      jthrot(ithr)=jthrot(ithr)-1
      if(jthrot(ithr).le.0) then
         NBTRG5=-2
      else
         tbv5=tend-tglim(jthrot(ithr),ithr)
      endif
      RETURN
      END                                                        BTHR 110
```

3.29 Subroutine CAERO

3.29.1 Purpose

The purpose of this subroutine is to calculate the nonlinear aerodynamic coefficients based on input values of mach number, angle of attack, inboard and outboard deflection angles, and sideslip angle.

3.29.2 Variable Listing

3.29.3 Subroutines Called:

Name	Function
INTER	Performs linear interpolation on nonlinear aerodynamic data
SEARCH	Searches a table of values for a unique value.

3.29.4 Calling Subroutines:

Name	Function
ADER	Calculates the time independent portion of the equations of motion during forward integration

3.29.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
aero	Output variable containing current aerodynamic coefficients				Local	
c	Axial force coefficients				Local	
cab	Axial force coefficients				Local	
clb	Yawing moment coefficients				Local	
cmb	Pitch moment coefficients				Local	
cnb	Normal force coefficients				Local	
cxb	Rolling moment coefficents				Local	
cyb	Side force coefficients				Local	
dinbd	Delta inboard elevon angle				Local	
doutbd	Delta outboard elevon angle				Local	
dx	Temporary variable				Local	
im1	Index (i-1)				Local	
im2	Index (i-2)				Local	
ntmach	Number of tabular values in mach number table		ainit	input	Global	aero
tmach(28)	Mach table for aerodynamic tables		ainit	input	Global	aero
vec	Multiplication factors based on inboard and outboard elevon angles				Local	
xmach	Mach number				Local	

```
      SUBROUTINE CAERO(AERO,XMACH,ALPHA,BETA,DINBD,DOUTBD)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION AERO(1),VEC(6)

      DIMENSION CAB(6,2),CNB(6,2),CMB(6,2),CYB(6,2),CXB(6,2),CLB(6,2),
     *C(6,2)

      COMMON/AERO/NTALP(2),NTBETA(2),NTMACH,TALPH(11,2),TBETA(11,2),
     *TMACH(28),CAA(15,11,6),CNA(15,11,6),CMA(15,11,6),CYA(15,11,6),
     *CXA(15,11,6),CLA(15,11,6)

      CALL SEARCH(XMACH,TMACH,NTMACH,IM1,IM2)
      CALL SEARCH(ALPHA,TALPH(1,1),NTALP(1),IA1,IA2)
      CALL SEARCH(BETA,TBETA(1,2),NTBETA(2),IB1,IB2)

      CALL INTER(CAB,CAA,IA1,IA2,ALPHA,TALPH(1,1),IM1,IM2)
      CALL INTER(CNB,CNA,IA1,IA2,ALPHA,TALPH(1,1),IM1,IM2)
      CALL INTER(CMB,CMA,IA1,IA2,ALPHA,TALPH(1,1),IM1,IM2)
      CALL INTER(CYB,CYA,IB1,IB2,BETA,TBETA(1,2),IM1,IM2)
      CALL INTER(CXB,CXA,IB1,IB2,BETA,TBETA(1,2),IM1,IM2)
      CALL INTER(CLB,CLA,IB1,IB2,BETA,TBETA(1,2),IM1,IM2)

      VEC(1)=1.
      VEC(2)=DINBD
      VEC(3)=DINBD*DINBD
      VEC(4)=DOUTBD
      VEC(5)=DOUTBD*DOUTBD
      VEC(6)=DINBD*DOUTBD

      DO 4 I=1,6
      DO 4 J=1,2
    4 C(I,J)=0.

      DO 5 J=1,2
      DO 5 I=1,6
      C(1,J)=C(1,J)+VEC(I)*CAB(I,J)
      C(2,J)=C(2,J)+VEC(I)*CNB(I,J)
      C(3,J)=C(3,J)+VEC(I)*CMB(I,J)
      C(4,J)=C(4,J)+VEC(I)*CYB(I,J)
      C(5,J)=C(5,J)+VEC(I)*CXB(I,J)
    5 C(6,J)=C(6,J)+VEC(I)*CLB(I,J)

      IF(XMACH.LT.TMACH(1)) then

         DO 8 I=1,6
    8    AERO(I)=C(I,1)

         RETURN

      endif

      IF(XMACH.GT.TMACH(NTMACH)) then

         DO 10 I=1,6
```



```
   10    AERO(I)=C(I,2)

         RETURN

      endif

      DX=0.
      IF(IM1.NE.IM2) DX=(XMACH-TMACH(IM1))/(TMACH(IM2)-TMACH(IM1))

      DO 6 I=1,6
    6 AERO(I)=C(I,1)+(C(I,2)-C(I,1))*DX

      RETURN

      END
```

3.30 Subroutine CHIPOL

3.30.1 Purpose

The purpose of this subroutine is to evaluate pitch and yaw attitude polynomials.

3.30.2 Variable Listing

3.30.3 Subroutines Called:

None

3.30.4 Calling Subroutines:

Name	Function
ADER1	Calculates the time dependent portion of the equations of motion during forward integration

3.30.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a1	Coefficients of 1st polynomial				Local	
a2	Coefficients of 2nd polynomial				Local	
m	Index				Local	
n1	Order of polynomial of 1st parameter				Local	
n2	Order of 2nd polynomial				Local	
t	Current time	sec			Local	
y 1	Dependent variable defined by 1st polynomial at current time				Local	
y 2	Dependent variable defined by 2nd polynomial at current time				Local	

```
      SUBROUTINE CHIPOL(N1,N2,T,Y1,A1,Y2,A2)
C
C      EVALUATES PITCH AND YAW ATTITUDE POLYNOMIALS
C           N1     = ORDER OF POLYNOMIAL IN A1
C           N2     = ORDER OF POLYNOMIAL IN A2
C           T      = INDEPENDENT VARIABLE IN POLYNOMIALS
C           Y1     = DEPENDENT VARIABLE EVALUATED BY A1 POLYNOMIAL AT T
C           A1     = COEFFICIENTS OF POLYNOMIAL 1
C           Y2     = DEPENDENT VARIABLE EVALUATED BY A2 POLYNOMIAL AT T
C           A2     = COEFFICIENTS OF POLYNOMIAL 2
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION A1(10),A2(10)

      M=N1+1
      Y1=A1(M)

      DO 1 I=1,N1
      J=M-I
    1 Y1=A1(J)+Y1*T

      M=N2+1
      Y2=A2(M)

      IF(N2.EQ.0) RETURN

      DO 2 I=1,N2
      J=M-I
    2 Y2=A2(J)+Y2*T

      RETURN
      END
```

3.31 Subroutine COPLDF

3.31.1 Purpose

The purpose of this subroutine is write list-directed files.

3.31.2 Variable Listing

3.31.3 Subroutines Called:

Name	Function
GOUT	Displays a character string at the terminal
LDSWI	Reads and creates direct access file
LDWRIT	Generic routine to write a direct access file

3.31.4 Calling Subroutines:

Name	Function
AREAIN	Controls user interface

3.31.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
c6	Name of for009 file				Local	
contrl	Index used to define file number				Local	
fildes	Character indicating name of mission			input	Global	agen3
icop	Index (0=terminate,1=write on a list directed file)				Local	
is	Index				Local	
mision	Character indicating name of mission			input	Global	agen3
na	Name of event from output file				Local	
nb	Name of mission				Local	
next	Index				Local	
nwr	Number of parameters				Local	
sngl	Character indicating name of mission			input	Global	agen3
sword	Single precision equivalence of word array				Local	
word	Parameters from output file				Local	

```
C
C*********************************************************************
C
      SUBROUTINE COPLDF
C
C     COPY FROM LOGICAL UNIT 9 TO LIST-DIRECTED FILE
C     UNIT 9 IS FROM MASTRE PROGRAM OUTPUT
C
C*********************************************************************
C
C234567890123456789012345678901234567890123456789012345678901234567890123456789012
C        1         2         3         4         5         6         7
C
C*********************************************************************
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      REAL*4 SWORD(160)
C
      INTEGER CONTRL
C
      LOGICAL DEMAND,GRAPH
C
      CHARACTER*6 NA(2),NB,C6,MISION
      CHARACTER*12 FILE
      CHARACTER*36 FILDES
C
      COMMON/AGEN3/ITHR,IWD,IXR,JUMP,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISION,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)
      COMMON /CCC/GRAPH,JOUT,DEMAND
C
      DIMENSION WORD(160)
C
      DATA FILE/'            '/
      DATA FILDES/'MASTRE CASE                          '/
C
C
C*********************************************************************
C
 1    CALL PAGIT
C
      CALL GOUT(' ',1)
      CALL GOUT('SELECT ONE OF THE FOLLOWING',30)
      CALL GOUT('0-TERMINATE',12)
      CALL GOUT('1-WRITE ON A LIST-DIRECTED FILE',31)
C
      WRITE(6,2)
 2    FORMAT(1X,'CHOICE> ',$)
      READ(5,3,ERR=1) ICOP
 3    FORMAT(I1)
C
      IF(ICOP .GT.1 .OR. ICOP .LT. 0) GO TO 1
      IF(ICOP .EQ. 0) RETURN
C
```



```
C      REWIND 9
      CONTRL=0
      NEXT=16
      NWR=160
C
      CALL GOUT(' ',1)
      CALL LDSWI(FILE,FILDES,CONTRL,NWR,*1)
       C6=' '
       WRITE(6,*)'INSERT NAME OF YOUR FOR009 FILE CURRENTLY  ',MISION
       READ(5,'(A6)')C6
       IF(C6(1:1).EQ.' ')THEN
         C6=MISION
      ENDIF
      CLOSE(UNIT=9)
      OPEN(UNIT=9,NAME=C6,RECL=163,FORM='UNFORMATTED',STATUS='OLD')
 4    READ(9,ERR=10,END=10) NA,WORD,NB
      DO 5 IS=1,160
 5    SWORD(IS) = SNGL(WORD(IS))
      IF(NA(1).EQ.'ENDPRT') GO TO 10
      CALL LDWRIT(FILE,2,CONTRL,NEXT,SWORD,*27)
      GO TO 4
C
 10   CLOSE(UNIT=CONTRL)
      CALL GOUT('THE FOLLOWING LIST-DIRECTED FILE WAS SUCCESSFULLY COMPL
     *ETED',59)
      WRITE(6,*) FILE
      CALL TWAIT(100)
      GO TO 1
C
 27   CALL GOUT(' ',1)
      CALL GOUT('* ERROR WRITING LDF *',21)
      CALL TWAIT(100)
      GO TO 1
C
      END
```

3.32 Subroutine DBANK

3.32.1 Purpose

The purpose of this subroutine is read a database file and place the data in a storage array.

3.32.2 Variable Listing

3.32.3 Subroutines Called:

None

3.32.4 Calling Subroutines:

Name	Function
AINIT	Input routine
WINDIN	Reads wind data from database

3.32.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a	Output array				Local	
block(1500)	Output storage block			input	Global	sblock
im	Input index (minimum index of storage array)				Local	
in	Input index (maximum index of storage array)				Local	
ip	Absolute value of im				Local	
nr	Number of record on direct access file				Local	

```
      SUBROUTINE DBANK(A,NR,IM,IN)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      LOGICAL DEMAND,GRAPH
C
      REAL*4 BLOCK
C
      COMMON/CCC/GRAPH,JO,DEMAND
      COMMON/SBLOCK/BLOCK(1500)
      DIMENSION A(1)
C
C       DATA BASE IS NOW OPENED IN AINIT
C       SAM POWELL 9/7/89
C
      IF(IM.GT.0) READ(8'NR,ERR=2) BLOCK
      IF(IOSTAT.NE.0) GO TO 4
      IP=IABS(IM)
      J=0
      DO 1 I=IP,IN
      J=J+1
    1 A(J)=DBLE(BLOCK(I))
      RETURN
C
    2 WRITE(JO,3) NR
    3 FORMAT(2X,'ERROR IN READING RECORD NO.',I6,' IN DBANK')
      GO TO 10
    4 WRITE(JO,5)IOSTAT
    5 FORMAT(2X,'ERROR IN OPENING FILE MASTRE.DAT IN DBANK',I4)
   10 STOP
      END
```

3.33 Subroutine DESOLV

3.33.1 Purpose

The purpose of this subroutine is to initialize the inputs for the integration subroutine DPIR.

3.33.2 Variable Listing

3.33.3 Subroutines Called:

Name	Function
DPIR	Integration routine

3.33.4 Calling Subroutines:

Name	Function
AFORUN	Controls logic for forward integration and calls integration routine
BAKRN	Controls logic for backward integration and calls integration routine

3.33.5 Fortran Listing

desolv

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
aco(10)	Constants for integration routine		desolv	output	Global	mod1
bco(10)	Coefficients for integration routine		desolv	output	Global	mod1
big	Big number used for upper bound				Local	
blk	Storage array				Local	
dr1	Entry point of routine that carries out the portion of the derivative evaluation involving the independent variable only				Local	
dr2	Entry point to the routine that carries out the remaining calculations necessary to compute all nact derivatives and stores them in hbk				Local	
dtmn	Minimum step size allowed	sec			Local	
dtmx	Maximum step size allowed	sec			Local	
dusc	Delta u scale factor in integration routine		desolv	output	Global	des1
el	Lower limit for stepsize control	sec	desolv	output	Global	des1
elo	Lower limit for step size control				Local	
ends	Entry point to routine entered at the end of a full integration step				Local	
eost	Logical variable to denote call to termination routine in integration routine			output	Global	des1
error	Logical variable to denote error in integration routine			output	Global	des1
eu(1)	Upper limit for stepsize control	sec	desolv	output	Global	des1
eup	Upper limit for step size control				Local	
hbk	Storage array				Local	
hmn	Minimum step size when using variable step integration routine	sec		output	Global	des1
hmx	Maximum step size	sec		output	Global	des1
hnom	Nominal step size	sec			Local	
hold	Nominal step size	sec		output	Global	des1
id	Index				Local	
iddi	Index				Local	
idl(11)	Integer array that is used as an indicator block for triggers (=1, dependent; =0, independent)			output	Global	des1
irst	Logical variable for integration restart			output	Global	des1

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ityp	Index indicating integration method type				Local	
kerr	Error indicator				Local	
kind	Integration type flag for forward trajectory			output	Global	des1
kon	Control index for control of Runge-Kutta or Adams-Moulton logic in main loop			output	Global	des1
kr(1)	Error indicator in integration routine		desolv	output	Global	des1
ll	Variable dimension used in storage array				Local	
m	Number of derivatives			output input	Global	des1
mmp2	Index (mp1+1)				Local	
mp1	Index in integration routine			output input	Global	des1
mp2	Index in integration routine			output	Global	des1
mxeq	The total number of possible first order differential equations to be solved				Local	
mxntrg	Maximum number of triggers				Local	
nact	Actual number of differential equations to solve				Local	
nbdif	Index used in integration routine		desolv	output	Global	des1
neq	Number of differential equations to integrate			output	Global	des1
nmx	The total number of possible first order differential equations to be solved			output	Global	des1
nn	Index in integration routine			output	Global	des1
nold	Actual number of differential equations to solve in integration routine			output	Global	des1
notrg	Number of triggers				Local	
np	Code index				Local	
nstart	Flag in integration routine			output	Global	des1
ntr1	Trigger flag for 1st event in calling argument				Local	
ntr11	Trigger flag for 11th event in calling argument				Local	
ntr2	Trigger flag for 2nd event in calling argument				Local	
ntr3	Trigger flag for 3rd event in calling argument				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ntr4	Trigger flag for 4th event in calling argument				Local	
ntr5	Trigger flag for 5th event in calling argument				Local	
ntr6	Trigger flag for 6th event in calling argument				Local	
ntr7	Trigger flag for 7th event in calling argument				Local	
ntr8	Trigger flag for 8th event in calling argument				Local	
ntr9	Trigger flag for 9th event in calling argument				Local	
ntrg	Number of triggers			output	Global	des1
rt1	Entry point of the trigger interrupt routine for the 1st event in calling argument				Local	
rt11	Entry point of the trigger interrupt routine for the 11th event in calling argument				Local	
rt2	Entry point of the trigger interrupt routine for the 2nd event in calling argument				Local	
rt3	Entry point of the trigger interrupt routine for the 3rd event in calling argument				Local	
rt4	Entry point of the trigger interrupt routine for the 4th event in calling argument				Local	
rt5	Entry point of the trigger interrupt routine for the 5th event in calling argument				Local	
rt6	Entry point of the trigger interrupt routine for the 6th event in calling argument				Local	
rt7	Entry point of the trigger interrupt routine for the 7th event in calling argument				Local	
rt8	Entry point of the trigger interrupt routine for the 8th event in calling argument				Local	
rt9	Entry point of the trigger interrupt routine for the 9th event in calling argument				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
stepng	Logical variable used for step size control in integration routine			output	Global	des1
temp	Temporary variable				Local	
trip	Stored time used in calculation of step size in integration	sec		output	Global	des2
var1	Name of the variable being tested for the 1st event in calling argument				Local	
var11	Name of the variable being tested for the 11th event in calling argument				Local	
var2	Name of the variable being tested for the 2nd event in calling argument				Local	
var3	Name of the variable being tested for the 3rd event in calling argument				Local	
var4	Name of the variable being tested for the 4th event in calling argument				Local	
var5	Name of the variable being tested for the 5th event in calling argument				Local	
var6	Name of the variable being tested for the 6th event in calling argument				Local	
var7	Name of the variable being tested for the 7th event in calling argument				Local	
var8	Name of the variable being tested for the 8th event in calling argument				Local	
var9	Name of the variable being tested for the 9th event in calling argument				Local	
yl	The smallest value of a dependent variable that will affect the automatic step size logic			output	Global	des1
ylo	The smallest value of a dependent variable that will affect the automatic step size control logic				Local	
zv1	The value of var at which the trigger interrupt routine is to be executed for the 1st event in calling argument				Local	
zv11	The value of var at which the trigger interrupt routine is to be executed for the 11th event in calling argument				Local	
zv2	The value of var at which the trigger interrupt routine is to be executed for the 2nd event in calling argument				Local	
zv3	The value of var at which the trigger interrupt routine is to be executed for				Local	

```
      SUBROUTINE DESOLV (MXEQ,NACT,DR1,DR2,ENDS,NP,KERR,HNOM, HBK,BLK,
C
C       *         *         *         *         *         *         *
C     DESOLV INTEGRATION PACKAGE PROGRAMMED BY JOAN SHEEHAN,          *
C     NOVEMBER 1966, UNDER SUPPORT WORK ORDER WITH MSFC, HUNTSVILLE,  *
C                                                                     *
C     ALABAMA.
C       *         *         *         *         *         *         *
C
     X                ITYP, LL, EUP,ELO,DTMX,DTMN,YLO,
     X                NOTRG,
     X                NTR1, RT1, VAR1, ZV1,  NTR2, RT2, VAR2, ZV2,
     X                NTR3, RT3, VAR3, ZV3,  NTR4, RT4, VAR4, ZV4,
     X                NTR5, RT5, VAR5, ZV5,  NTR6, RT6, VAR6, ZV6,
     X                NTR7, RT7, VAR7, ZV7,  NTR8, RT8, VAR8, ZV8,
     X                NTR9, RT9, VAR9, ZV9,  NTR10,RT10,VAR10,ZV10,
     X                NTR11,RT11,VAR11,ZV11 )
C        WHERE MXEQ =  THE TOTAL NUMBER OF POSSIBLE FIRST ORDER DIFFER-
C                      ENTIAL EQUATIONS TO BE SOLVED
C              NACT =  ACTUAL NUMBER OF DIFFERENTIAL  EQUATIONS TO SOLVE
C              DR1  =  ENTRY POINT TO ROUTINE THAT CARRIES OUT THE
C                      PORTION OF THE DERIVATIVE EVALUATION INVOLVING
C                      THE INDEPENDENT VARIABLE ONLY
C                      DR1 IS OPTIONAL. PUT A ZERO IF NOT DESIRED.
C              DR2  =  ENTRY POINT TO THE ROUTINE THAT CARRIES OUT THE
C                      REMAINING CALCULATIONS NECESSARY TO COMPUTE ALL
C                      NACT DERIVATIVES AND STORES THEM IN HBK
C              ENDS =  ENTRY POINT TO ROUTINE ENTERED AT THE END OF A
C                      FULL INTEGRATION STEP - OPTIONAL.
C                      PUT A ZERO IF NO END-OF-STEP ROUTINE IS DESIRED.
C              NP   =  ONE OF THE FOLLOWING CODE NUMBERS DEPENDING ON
C                      THE OPTIONS DESIRED
C                      ////////////////////////////
C                      VALUES OF NP / DR1 / ENDS /
C                      ////////////////////////////
C                               1  / NO  / YES  /
C                               2  / YES / YES  /
C                               5  / NO  / NO   /
C                               6  / YES / NO   /
C                      ////////////////////////////
C
C              KERR =  ERROR INDICATOR. NORMAL RETURN WHEN KERR=0
C                                       ERROR RETURN WHEN KERR=1
C              HNOM =  NOMINAL STEP SIZE
C              HBK  =  THE FIRST WORD OF A BLOCK OF STORAGE WITH HBK(1)
C                      STARTING IN AN EVEN LOCATION AND -
C                      HBK(1) = CURRENT STEP SIZE
C                      HBK(2) = NOT USED
C                      HBK(3) = INDEPENDENT VARIABLE IN SINGLE
C                               PRECISION (MUST BE PRESET)
C                      HBK(4) = NOT USED
C                      HBK(5) TO HBK(MXEQ+4) = DEPENDENT VARIABLES
C                      HBK(MXEQ+5) TO HBK(2*MXEQ+4) = DERIVATIVES
C              BLK  =  THE FIRST WORD OF A BLOCK OF STORAGE THAT MUST BE
C                      EQUIVALENT TO HBK(2*MXEQ+5) AND MUST HAVE TWO
C                      DIMENSIONS (MXEQ,LL) WHERE LL IS DEFINED BELOW
C              ITYP =  INDICATOR FOR THE TYPE OF INTEGRATION TO BE USED
```



```
C                      ITYP=1, VARIABLE STEP ADAMS-MOULTON
C                      ITYP=2, FIXED STEP RUNGE-KUTTA
C                      ITYP=3, FIXED STEP ADAMS PREDICTOR ONLY
C                      (AN EXTERNAL CHANGE OF HNOM CAUSES A RESTART )
C              LL    =  THE VARIABLE DIMENSION USED IN BLK(MXEQ,LL)
C                      LL=5 WHEN ITYP = 2
C                      LL = 3*(M+2) +4 WHEN ITYP = 1 OR 3
C                      M IS THE ORDER OF THE DIFFERENCES USED
C              EUP   =  UPPER LIMIT FOR STEP SIZE CONTROL
C              ELO   =  LOWER LIMIT FOR STEP SIZE CONTROL
C              DTMX  =  MAXIMUM STEP SIZE ALLOWED
C              DTMN  =  MINIMUM STEP SIZE ALLOWED
C              YLO   =  THE SMALLEST VALUE OF A DEPENDENT VARIABLE THAT
C                       WILL AFFECT THE AUTOMATIC STEP SIZE LOGIC
C              NOTRG=   THE NUMBER OF TRIGGERS TO FOLLOW
C                       THE MAXIMUM NO OF TRIGGERS IS SET AT 11
C              NTR-  =  PLUS OR MINUS AN INTEGER 1 OR 2
C                       PLUS SIGN MEANS TRIGGER IS ON
C                       MINUS SIGN MEANS TRIGGER IS OFF
C                       INTEGER 1 INDICATES NO DISCONTINUITY AFTER
C                                 EXECUTING THE TRIGGER ROUTINE
C                       INTEGER 2 INDICATES A DISCONTINUITY AND FORCES
C                                 A RESTART
C              RT-   =  ENTRY POINT OF THE TRIGGER INTERRUPT ROUTINE
C              VAR-  =  NAME OF THE VARIABLE BEING TESTED
C              ZV-   =  THE VALUE OF VAR AT WHICH THE TRIGGER INTERRUPT
C                       ROUTINE IS TO BE EXECUTED
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      LOGICAL DEMAND,GRAPH
C
      COMMON/CCC/GRAPH,JO,DEMAND
      COMMON/MOD1/ACO(10),BCO(10)
      COMMON /DES1/
     *   KR(1),   NMX,   NEQ, NOLD,     M,   MP1,   MP2,    NN, KIND,  KON,
     *     IFL, NTRG, NIVT, NDVT,   NTR,NSTART,NBDIF,   NNT,   NTT,IDL(11),
     *    IDL2, JJSS,J1000,J2000,J4000,J7000, IRST, EOST,    HD,   SMA,
     *   ERROR,REDOT,STEPNG,
     *   EU(1),   EL,   HMX,   HMN,   YL,     A, ENP1,     D,     B,  SKP,
     *      SK,   HC, DELU, DUSC, HOLD,  VAR,   ZV,   HM,    HP,     R,
     *       Q, DELT,   HMU,  FMU,CK(10),QQ(10),XLJ(11), RJ(11), WJ(11)
     *  COMMON /DES2/     T,   TSV,  TGO, TMIN, TRIP,   TL,    TR
C
      LOGICAL EOST, SMA, HD, IRST, ERROR, ODD, REDOT, STEPNG
C
      DIMENSION HBK(1), BLK(MXEQ,LL)
C
C
      DATA BIG/1.0D30/
C
      GO TO 400
  402 CONTINUE
      MXEQ=MXEQ
      NACT=NACT
      NP=NP
```

```
      KERR=KERR
      HNOM=HNOM
      IDDI=2*MXEQ+4
C
      DO 152 ID=1,IDDI
  152 HBK(ID)=HBK(ID)
      BLK(1,1)=BLK(1,1)
      ITYP=ITYP
      LL=LL
      EUP=EUP
      ELO=ELO
      DTMX=DTMX
      DTMN=DTMN
      YLO=YLO
      NOTRG=NOTRG
      NTR1=NTR1
      NTR2=NTR2
      NTR3=NTR3
      NTR4=NTR4
      NTR5=NTR5
      NTR6=NTR6
      NTR7=NTR7
      NTR8=NTR8
      NTR9=NTR9
      NTR10=NTR10
      NTR11=NTR11
      VAR1=VAR1
      VAR2=VAR2
      VAR3=VAR3
      VAR4=VAR4
      VAR5=VAR5
      VAR6=VAR6
      VAR7=VAR7
      VAR8=VAR8
      VAR9=VAR9
      VAR10=VAR10
      VAR11=VAR11
      ZV1=ZV1
      ZV2=ZV2
      ZV3=ZV3
      ZV4=ZV4
      ZV5=ZV5
      ZV6=ZV6
      ZV7=ZV7
      ZV8=ZV8
      ZV9=ZV9
      ZV10=ZV10
      ZV11=ZV11
  400 CONTINUE
C
      ACO(1)=1.
      ACO(2)=.5
      ACO(3)=5./12.
      ACO(4)=3./8.
      ACO(5)=251./720.
      ACO(6)=95./288.
```



```
      ACO(7)=19087./60480.
      ACO(8)=36799./120960.
      ACO(9)=1070017./3628800.
      ACO(10)=.287075448
      BCO(1)=1.
      BCO(2)=-.5
      BCO(3)=-1./12.
      BCO(4)=-1./24.
      BCO(5)=-19./720.
      BCO(6)=-3./160.
      BCO(7)=-863./60480.
      BCO(8)=-275./24192.
      BCO(9)=-33953./3628800.
      BCO(10)=-.78925542E-2
      NMX = MXEQ
      KIND = ITYP
      NTRG = NOTRG
      EU(1)=EUP
      EL = ELO
      HMX = DTMX
      HMN = DTMN
      YL = YLO
C
   50 MXNTRG = 11
      IF (NTRG .LE. MXNTRG)  GO TO  52
   51 FORMAT (// 52H MAXIMUM NUMBER OF TRIGGERS THIS DECK WILL HANDLE IS
     *,I4/5X,14HRECOMPILE F4IR)
      WRITE (JO,51)  MXNTRG
      KERR = 1
      RETURN
   52 CONTINUE
C INITIALIZATION
C     CALCULATE DELTA U SCALE FACTOR = 0.596E-7
      TEMP=2.**24
      DUSC = 1.0/TEMP
      STEPNG = .FALSE.
      IF (KERR .EQ. 100)  STEPNG = .TRUE.
      KERR = 0
      KR(1)=0
  100 CONTINUE
      DO  55  J=1,MXNTRG
   55 IDL(J) = 0
      HBK(1) = HNOM
      HOLD = HNOM
C
C
C POINT FOR RESTART
  102 FORMAT (///5H NP =I3, 21H - UNACCEPTABLE VALUE)
  103 FORMAT (////7H ITYP =, I3, 11HNOT ALLOWED)
      IF (NP .EQ. 1  .OR.  NP .EQ. 2  .OR.  NP .EQ. 5  .OR.  NP .EQ. 6)
     X     GO TO  105
      WRITE (JO,102) NP
      KERR = 1
      RETURN
  105 IF (KIND .EQ. 1  .OR.  KIND .EQ. 2)  GO TO  106
      IF (KIND .EQ. 3)  GO TO  106
```

```
      WRITE (JO,103)  KIND
      KERR = 1
      RETURN
C
  106 CONTINUE
      NOLD = NACT
      NEQ = NACT
      EOST = .FALSE.
      NN = NP-4
      IF (NP .GT. 2)  GO TO  108
      EOST = .TRUE.
      NN = NP
  108 IRST = .FALSE.
C
C    INITIALIZE FOR RUNGE-KUTTA OR ADAMS-MOULTON START
C           KIND = 1 FOR ADAMS-MOULTON
C           KIND = 2 FOR RUNGE-KUTTA
C           KIND = 3 FOR FIXED STEP ADAMS OPEN FORMULA
C
      GO TO (110, 112, 110), KIND
C     VARIABLE STEP ADAMS-MOULTON
  110 TRIP = HBK(3)
      NSTART = 0
      M=(LL-10)/3
      MP1 = M+1
      MP2 = M+2
C     COUNTER FOR SWITCHING TO A-M AFTER M+2 R-K STEPS
      NBDIF = 0
      GO TO  115
C
C     FIXED STEP SIZE RUNGE-KUTTA THROUGHOUT
  112 TRIP = BIG
      NSTART = 100
      M = 0
      MP1=0
      MP2=0
  115 KON = 2
C           KON IS CONTROL FOR R-K LOGIC OR A-M IN MAIN LOOP
C
      ERROR = .FALSE.
      MMP2 = MP1+1
      CALL DPIR   (HBK(3),HBK(5),HBK(NMX+5),HBK(2*NMX+5),HBK(4*NMX+5),
     *             BLK(1,5),BLK(1,MP2+5),BLK(1,2*MP2+5), MXEQ,MMP2,
     X           HBK, HNOM, NACT,  DR1, DR2, ENDS,
     X                 NTR1, RT1, VAR1, ZV1,  NTR2, RT2, VAR2, ZV2,
     X                 NTR3, RT3, VAR3, ZV3,  NTR4, RT4, VAR4, ZV4,
     X                 NTR5, RT5, VAR5, ZV5,  NTR6, RT6, VAR6, ZV6,
     X                 NTR7, RT7, VAR7, ZV7,  NTR8, RT8, VAR8, ZV8,
     X                 NTR9, RT9, VAR9, ZV9,  NTR10,RT10,VAR10,ZV10,
     X                 NTR11,RT11,VAR11,ZV11 )
C
C                           TO RESTART
      IF (IRST)  GO TO  100
C                           SUBROUTINE INTERNAL ERROR
      IF (ERROR)  KERR = 1
C
```



```
      IF(KR(1).EQ.100) KERR=100
C     (KR IS RESET TO ZERO IN DESOLV INITIALIZATION)
 9999 RETURN
      END
```

3.34 Subroutine DISPPRT

3.34.1 Purpose

The purpose of this subroutine is to output dispersion parameters on a direct access file. The dispersion file is used as input to the Root-Sum-Square (RSS) program.

3.34.2 Variable Listing

3.34.3 Subroutines Called:

Name	Function
ADER	Calculates the time independent portion of the equations of motion during forward integration
ADER1	Calculates the time dependent portion of the equations of motion during forward integration

3.34.4 Calling Subroutines:

Name	Function
AFORUN	Controls logic for forward integration and calls integration routine
DPIR	Integration routine

3.34.5 Fortran Listing

dispprt

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a(3,3)	Tranformation matrix from equatorial to plumbline coordinate systems		aforun	input	Global	agen2
a1	Temporary variable				Local	
alf	Angle of attack	rad	ader	input	Global	agen2
alfy	Sideslip angle	rad	ader	input	Global	agen2
alongo	Launch longitude	rad	ainit	input	Global	alat
alt	Altitude	m	ader	input	Global	agen2
azi	Inertial azimuth angle	rad			Local	
begnow	Eccentric anomaly				Local	
bgenow	Eccentric anomaly				Local	
bgethn	Eccentric anomaly				Local	
c1	Angular momentum				Local	
c3	Twice the energy	m^2/se			Local	
cbgenw	Cosine of eccentric anomaly				Local	
cbgetn	Cosine of eccentric anomaly				Local	
cchip	Cosine of pitch attitude angle		ader1	input	Global	agen2
cchiy	Cosine of yaw attitude angle		ader1	input	Global	agen2
cetanw	Cosine of eccentric anomaly				Local	
cetatn	Cosine of eccentric anomaly				Local	
cmue	Gravitiational constant	m^3/se	ainit	input	Global	const
cphi	Cosine of colatitude				Local	
cth	Cosine of colatitude		dispprt	output input	Global	agen2
d	Direction cosine matrix				Local	
delbge	Delta eccentric anomaly				Local	
deleta	Increment between eccentric anomalies				Local	
dispstep(2)	Increment steps used in output of dispersion trajectories	sec	ainit	input	Global	disp
dlphi	Delta longitude				Local	
dti	Incremental time to predicted impact point	sec			Local	
dtz	Time difference between GRR and launch time	sec	ainit	input	Global	agen1
dvar(25)	Storage array for the state derivatives		ader	input	Global	forint
ecc	Eccentricity				Local	
etanw	Eccentric anomaly based on radius	rad			Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
etatn	Eccentric anomaly based on earth radius	rad			Local	
fom	Total vehicle sensed acceleration	g's		input	Global	agen5
gam	Inertial flight path angle	rad			Local	
gamr	Relative flight path angle	rad			Local	
idistab	Array of dispersion codes				Local	
irec	Numb· of record on direct access storage file		displprt	output input	Global	disp
ivar	Index used to denote the type of dispersion case		ainit	input	Global	disp
k1	Dimension of dispersion choice array				Local	
key	Mode switch to indicate the portion of the routine to execute		aforun	input	Global	disp
nmax	Iteration counter		mastre	output	Global	agen3
omega	Earth's spin rate	rad/sec	ainit	input	Global	const
p	Propellant weight for each engine type	kg		input	Global	agen1
pfkg	Conversion for pounds to kilograms	kg/lbs	ainit	input	Global	const
phi	Longitude	deg			Local	
phimp	Latitude of impact point	deg			Local	
pi	Pi constant (3.14159265)		ainit	input	Global	const
prop(6)	Propellant weight for each engine type	kg	ainit	input	Global	agen1
psffm	Conversion for pounds per square feet from newtons per square meter				Local	
q	Dynamic pressure	nt/m^2	ader	input	Global	agen2
r	Radius from earth's center	m	ader	output input	Global	agen2
range_in	In-plane range	nm			Local	
range_out	Out-of-plane range	nm			Local	
re	Earth's radius	m	ainit	input	Global	const
rsth	The product of radius and the sine of the colatitude				Local	
schip	Sine of pitch attitude angle		ader1	input	Global	agen2
schiy	Sine of yaw attitude angle		ader1	input	Global	agen2
sphi	Sine of colatitude				Local	
sth	Sine of colatitude			output input	Global	agen2
su	Y component of relative velocity vector				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
sv	Z component of relative velocity vector	m/sec		input	Global	agen2
sw	X component of relative velocity vector	m/sec		input	Global	agen2
t	Time from lift-off	sec	dpir	output input	Global	forint
tdisp	Time to begine the print increment from the 2nd array of the dispstep array times	sec	ainit	output	Global	disp
thet	Colatitude angle	rad			Local	
theta	Geocentric latitude angle	deg			Local	
thetg	Geodetic latitude angle	deg			Local	
thimp	Longitude of impact point	deg			Local	
tht1	Colatitude of impact point				Local	
tv15	Time to activate dispersion output file creation	sec	dispprt	output	Global	disp
u	Y component of plumbline inertial velocity vector	m/sec	dpir	input	Global	forint
umf	One minus the earth flattening coefficient		ainit	input	Global	agen2
v	Z component of plumbline inertial velocity vector	m/sec	dpir	input	Global	forint
var	Array of output variables for dispersion output file				Local	
vi	Inertial velocity magnitude	m/sec			Local	
vph	Temporary variable				Local	
vr	Magnitude of relative velocity	m/sec	ader	output input	Global	agen2
vth	Temporary variable				Local	
w	X component of plumbline inertial velocity vector	m/sec	dpir	input	Global	forint
x	X component of plumbline position vector	m	dpir	input	Global	forint
xm	Current vehicle mass	kg	dispprt	output input	Global	agen2
xmaug	Auxiliary vehicle mass	kg	athrev	input	Global	agen1
xmiad	Continuous portion of vehicle mass	kg	ader1	input	Global	agen2
y	Y component of plumbline position vector	m	dpir	input	Global	forint
z	Z component of plumbline position vector	m	dpir	input	Global	forint

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
zap(18)	Additional integrated variables	variabl	desolv	input	Global	forint

```
      Subroutine Dispprt

      parameter (k1=60)

      implicit double precision (a-h,o-z)

      character*60 head
      character*6 mision

      logical visc

      common/agen1/time(2,16),taut(15),tauw(15),tzero,tlift,ttilt,
     *  tminh,dtz,tq,tl,xmaug,tne(6,15),engdat(8,15),s(15),wd(15),
     *  wjet(15),head,print(15),step(15),hnom(15),hmax(15),prop(6),
     *  tbegr,tendr,chrdot,chiro,faz,bo(3,2),cbaxil,tchir,vrcut,
     *  fprfac,chvel

      common/agen2/aa,sa,ca,alf,alfy,chip,schip,cchip,chiy,schiy,
     *  cchiy,chir,schir,cchir,sth,cth,sthl,cthl,dphiz,rthe,r,vr,xm,sw,
     *  su,sv,q,viv(25),dvars(13),delxdw(15,3),delxdr(7,5),delxd(15,7),
     *  wzero,alt,xmach,qdot,faa,fan,a(3,3),case,ct2,umf,a12w,a22w,
     *  a32w,xjext,beu,byl,bel,bhmax(15),bhmn,bstep(15),hnmb,hmxb,
     *  tpoly,xmiad

      common/agen3/ithr,iwd,ixr,jump,kat,kcytab,kpage,ks1,kwta(7),
     *  lines,nbgct(7),nendct(7),nmax,noevnt(5),nowd(15),nvnt,nwvnt,
     *  ihead,minh,mpsflag(15),nsyst(15),iconsw,irtls,ipoly,kindb,
     *  krderb,mision,irtflg,nroll,ncoast,ifact,nobase,lstge(15),
     *  mswch(15)

      common/agen5/chibas,chpbas,chybas,chisav,txx,tyy,tzz,amx,amy,
     *  amz,tmz,thmfa,side,wmag,azw,fanc,flatc,fom,visc

      common/alat/alat,alongo,altls,ntable

      common/const/rad,pi,re,flat,cj,h,dj,cmue,omega,alt1,alt2,ps1,
     *  gzero,ptn,ptkg,psftm,pfn,pfkg,psffm

      common/disp/key,tv15,ivar,tdisp,dispstep(2),irec

      common/forint/hbank(2),t,tt,w,u,v,x,y,z,xm1,zap(18),dvar(25),
     *  fsave(625),ntrg1,tv1,ntrg2,tv2,ntrg3,tv3,ntrg4,tv4,ntrg5,tv5,
     *  ntrg6,tv6,ntrg7,tv7,ntrg8,tv8,ntrg9,tv9,ntrg10,tv10,ntrg11,
     *  tv11,ntrge12,tv12,ntrg13,tv13,kind,krder,eu,el,ayl,hmx,hmn,hnm

      dimension idistab(k1),var(25),d(3,3)

      data idistab/0010,1010,1011,1020,1021,1030,1031,1040,1041,1050,
     *    1051,1060,1061,1070,1071,1080,1081,1090,1091,1100,1101,1110,
     *    1111,1120,1121,1130,1131,
     *    2010,2011,2020,2021,2030,2031,2040,2041,2050,2051,2060,2061,
     *    3010,3011,3020,3021,3030,3031,3040,3041,
     *    4010,4011,4020,4021,
     *    5010,5020,5030,5040,5050,5060,5070,5080,5090/

      if(key.eq.0) then
```



```
         do 10 i=1,k1

         if(ivar.eq.idistab(i)) then
            irec=300*(i-1)
            go to 20
         endif
10       continue
         write(*,*) ' dispersion code ',ivar, ' not found, stop'
         stop
20       continue

         key=1

      endif

c ***************************************************************************
c     The input parameter ivar contains four digits and divided into
c     three parts; itype is the 1st digit, jtype is composed of the
c     2nd and 3rd digits, and ktype is the 4th digit.

c ***************************************************************************
c     itype - dispersion type

c            0___ - nominal
c            1___ - propulsion
c            2___ - aero/environment
c            3___ - mass property
c            4___ - gn&c
c            5___ - composite
c ***************************************************************************
c     jtype - subgrouping                          Record
c                                                  Location
c                                                  pos     neg
c     0010 - nominal                               00001
c ***************************************************************************
c     Propulsion

c     101_ - STME Vacuum thrust                    00301   00601
c     102_ - STME Vacuum Isp                       00901   01201
c     103_ - STME mixture ratio                    01501   01801
c     104_ - STME thrust misalign (pitch)          02101   02401
c     105_ - STME thrust misalign (yaw)            02701   03001
c     106_ - ASRM web action time                  03301   03601
c     107_ - ASRM vacuum Isp                       03901   04201
c     108_ - ASRM propellant loading               04501   04801
c     109_ - ASRM inert weight                     05101   05401
c     110_ - ASRM thrust misalign (pitch)          05701   06001
c     111_ - ASRM thrust misalign (yaw)            06301   06601
c     112_ - ASRM thrust imbalance                 06901   07201
c     113_ - ASRM thrust uncertainty               07501   07801
c ***************************************************************************
c     Aero/Environment

c     201_ - Forebody axial force                  08101   08401
c     202_ - Baseforce                             08701   09001
```

```
c       203_ - Other Aerodynamic coeff          09301   09601
c       204_ - wind profiles (head & tail)      09901   10201
c       205_ - winf profiles (rt & lt cross)    10501   10801
c       205_ - atmospheric density              11101   11401
c ***************************************************************
c       Mass Properties

c       301_ - Core inert weights               11701   12001
c       302_ - Propulsion mod inert wt          12301   12601
c       303_ - Core propellant weight           12901   13201
c       304_ - Center of Gravity                13501   13801
c ***************************************************************
c       GN&C

c       401_ - Booster Pitch steering program   14101   14401
c       402_ - Booster Yaw steering program     14701   15001
c ***************************************************************
c       Composite Trajectory info

c       5010 - Composite (positive)             15301
c       5020 - Composite (negative)             15601
c       5100 - RSS data                         18001
c ***************************************************************
c       ktype - dispersion type

c       ___0 - positive dispersion
c       ___1 - negative dispersion
c ***************************************************************

        call aderl
        call ader

c       Calculate vehicle mass

        xm=xmiad+xmaug

c       Calculate radius (r), inertial velocity (vi), inertial flight path
c       angle (gam), and relative flight path angle (gamr)

        r=sqrt(x*x+y*y+z*z)

        vi=sqrt(w*w+u*u+v*v)

        gam=0.
        if(vi.gt..1e-05) gam=asin((x*w+y*u+z*v)/(r*vi))

        gamr=0.
        if(vr.gt..1e-05) gamr=asin((x*sw+y*su+z*sv)/(r*vr))

        d(1,2)=x/r
        d(2,2)=y/r
        d(3,2)=z/r

        cth=a(1,2)*d(1,2)+a(2,2)*d(2,2)+a(3,2)*d(3,2)
        sth=sqrt(1.-cth*cth)
        rsth=r*sth
```



```
        sphi=a(1,1)*d(1,2)+a(2,1)*d(2,2)+a(3,1)*d(3,2)
        cphi=a(1,3)*d(1,2)+a(2,3)*d(2,2)+a(3,3)*d(3,2)

        d(1,3)=(d(1,2)*cth-a(1,2))/sth
        d(2,3)=(d(2,2)*cth-a(2,2))/sth
        d(3,3)=(d(3,2)*cth-a(3,2))/sth

        d(1,1)=(-y*a(3,2)+z*a(2,2))/rsth
        d(2,1)=(x*a(3,2)-z*a(1,2))/rsth
        d(3,1)=(-x*a(2,2)+y*a(1,2))/rsth

c       Calculate current longitude

        phi=atan2(sphi,cphi)
        if(t.lt..1) phi=dtz*omega
        phi=(phi-omega*(t+dtz))*rad-alongo

c       Calculate inertial azimuth

        vth=d(1,3)*w+d(2,3)*u+d(3,3)*v
        vph=d(1,1)*w+d(2,1)*u+d(3,1)*v

        azi=atan2(vph,-vth)*rad

c       Calculate geocentric and geodetic latitudes

        thet=acos(cth)
        theta=pi/2.-thet

        thetg=atan2(sin(theta),umf*umf*cos(theta))*rad

c       Calculate latitude and longitude of instanteous impact point

        c1=vi*r*cos(gam)
        c3=vi*vi-2.*cmue/r
        a1=abs(cmue/c3)
        p=c1*c1/cmue

        ecc=1.+p*c3/cmue
        phimp=0.
        thimp=0.
        if(ecc.ge.0.) then
           ecc=sqrt(ecc)
           if(re-p/(1.+ecc).ge.re/100.) then
              cetanw=(p-r)/(r*ecc)
              etanw=abs(cetanw)
              if(etanw.gt.1.) then
                 etanw=0.
              else
                 etanw=acos(etanw)
              endif
              if(cetanw.lt.0.) etanw=pi-etanw
              if(gam.lt.0.) etanw=2.*pi-etanw

              cetatn=(p-re)/(re*ecc)
              etatn=abs(cetatn)
```

```
            if(etatn.gt.1.) then
               etatn=0.
            else
               etatn=acos(etatn)
            endif
            if(cetatn.lt.0.) etatn=pi-etatn
            if(gam.lt.0.) etatn=2.*pi-etatn

            cbgenw=(a1-r)/(a1*ecc)
            bgenow=abs(cbgenw)
            if(bgenow.gt.1.) then
               bgenow=0.
            else
               bgenow=acos(begnow)
            endif
            if(cbgenw.lt.0.) bgenow=pi-bgenow
            if(gam.lt.0.) bgenow=2.*pi-bgenow

            cbgetn=(a1-re)/(a1*ecc)
            bgethn=abs(cbgetn)
            if(bgethn.gt.1.) then
               bgethn=0.
            else
               bgethn=acos(bgethn)
            endif
            if(cbgetn.lt.0.) bgethn=pi-bgethn
            if(gam.lt.0.) bgethn=2.*pi-bgethn

            delbge=bgethn-bgenow
            deleta=2.*pi-etatn-etanw

            dti=(delbge-ecc*(sin(bgethn)-sin(bgenow)))/sqrt(cmue/
     *           (a1*a1*a1))
            tht1=acos(cth*cos(deleta)+sth*sin(deleta)*cos(azi))
            dlphi=asin(sin(deleta)*sin(azi)/sin(tht1))

            phimp=phi+(dlphi-omega*dti)*rad
            thimp=90.-tht1*rad

         endif
       endif

c      Calculate in-plane and out-of-plane ranges

       range_in=re*atan2(x,y)
       range_out=re*asin(z/r)

c ***************************************************************************

       var(1)=t
       var(2)=fom
       var(3)=alt/.3048
       var(4)=r/.3048
       var(5)=vr/.3048
       var(6)=vi/.3048
       var(7)=q*psffm
```



```
       var(8)=gamr*rad
       var(9)=gam*rad
       var(10)=zap(7)
       var(11)=dvar(14)
       var(12)=range_in/.3048
       var(13)=range_out/.3048
       var(14)=azi
       var(15)=thetg
       var(16)=phi
       var(17)=thimp
       var(18)=phimp
       var(19)=xm*pfkg
       var(20)=alf*rad
       var(21)=alfy*rad
       var(22)=atan2(schip,cchip)*rad
       var(23)=atan2(schiy,cchiy)*rad
       var(24)=prop(1)-zap(1)*pfkg
       var(25)=prop(2)-zap(2)*pfkg

c ***************************************************************************
c      var(01) - time from liftoff (sec)
c      var(02) - axial acceleration (g's)
c      var(03) - altitude (ft)
c      var(04) - radius (ft)
c      var(05) - relative velocity (ft/sec)
c      var(06) - inertial velocity (ft/sec)
c      var(07) - dynamic pressure (psf)
c      var(08) - relative flight path angle (deg)
c      var(09) - inertial flight path angle (deg)
c      var(10) - stagnation heating (BTU)
c      var(11) - stagnation heating rate (BTU/sec)
c      var(12) - down range (ft)
c      var(13) - cross range (ft)
c      var(14) - inertial azimuth
c      var(15) - geodetic latitude (deg)
c      var(16) - longitude (deg)
c      var(17) - impact latitude (deg)
c      var(18) - impact longitude (deg)
c      var(19) - vehicle weight (lbs)
c      var(20) - angle of attack (deg)
c      var(21) - sideslip angle (deg)
c      var(22) - pitch attitude command (deg)
c      var(23) - yaw attitude command (deg)
c      var(24) - liquid propellant consumed (lbs)
c      var(25) - solid propellant consumed (lbs)
c ***************************************************************************

       irec=irec+1

       if(nmax.eq.0) write(80,rec=irec) var,ivar

       l=1
       if(t.ge.tdisp-.25) l=2

       tv15=t+dispstep(l)
```

```
        return
        end
```

3.35 Subroutine DPIR

3.35.1 Purpose

The purpose of this subroutine is to provide the integration of the equations of motion for both the forward and backward integration. Subroutines called by this subroutine are defined through the calling argument to the subroutine DESOLV. Three integration methods are available which are a variable step Adams-Moulton, a fixed step Runge-Kutta, and a fixed step Adams-Moulton.

3.35.2 Variable Listing

3.35.3 Subroutines Called:

Name	Function
ADER	Calculates the time independent portion of the equations of motion during forward integration
ADER1	Calculates the time dependent portion of the equations of motion during forward integration
AEOSR	Stores state variable during forward integration
ALIFT	Controls lift-off logic based on thrust-to-weight during forward integration
AQMAX	Prints maximum dynamic pressure
AROLL	Calculates parameters and prints roll maneuver events
ATHREV	Controls thrust event logic during forward trajectory
ATHRO	Controls MPS throttle events logic during forward trajectory
ATILT	Controls tilt-over maneuver logic during forward integration
AWDEV	Controls weight drop event logic during forward integration
AXPRT	Prints trajectory output at print intervals during forward integration
BDR1I	Calculates components of the equations of motion for the backward trajectory

BDRI	Calculates equations of motion for the backward trajectory
BGLIM	Controls logic for acceleration limits during backward integration
BKEND	Terminates backward integration
BSAVEG	Calculates the impulse response functions and integrates the weighting matrix
BTHREV	Controls thrust event logic during backward integration
BWDEV	Controls weight drop events during backward integration
DISPPRT	Prints dispersion parameters on output file
GLIMT	Controls acceleration logic during forward integration

3.35.4 Calling Subroutines:

Name	Function
DESOLV	Initializes integration constants and calls integration routine

3.35.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a	Tempoary variable in integration		dpir	output input	Global	des1
aco(10)	Constants for integration routine		desolv	input	Global	mod1
ak	Temporary storage variable for time	sec		output input	Global	des2
b	Temporary variable in integration		dpir	output input	Global	des1
bco(10)	Coefficients for integration routine		desolv	input	Global	mod1
big	Big number used for upper bound				Local	
ck(10)	Coefficients of differences in integration variables			output input	Global	des1
cut	Temporary storage variable for time	sec		output input	Global	des2
d	Temporary variable used in integration			output input	Global	des1
delt	Time interval	sec		output input	Global	des1
delu	Temporary variable used to calculate step size			output input	Global	des1
delx	Temporary storage array				Local	
dely	Temporary storage variable for time	sec	dpir	output input	Global	des2
delz	Temporary storage variable for time	sec	dpir	output input	Global	des2
dt	Step size				Local	
dusc	Delta u scale factor in integration routine		desolv	input	Global	des1
dy	Temporary variable				Local	
el	Lower limit for stepsize control	sec	desolv	input	Global	des1
enp1	Temporary variable used in step size control			output	Global	des1
eost	Logical variable to denote call to termination routine in integration routine			output	Global	des1
error	Logical variable to denote error in integration routine			output	Global	des1
eu(1)	Upper limit for stepsize control	sec	desolv	input	Global	des1
factj	Temporary variable				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ffmp1	Temporary storage variable for time	sec			Local	
fj	Temporary variable				Local	
fk	Temporary storage variable for time				Local	
fm	Temporay variable used to calculate table of backward differences in integration routine				Local	
fmp1	Temporary storage variable for time				Local	
fmu	Temporay variable used to calculate table of backward differences in integration routine			output	Global	des1
fn	Temporary variable				Local	
fnum	Temporary variable				Local	
h	Logical variable used in step size control			output	Global	des1
hc	Integration step size			output input	Global	des1
hd	Logical variable used in step size control			output	Global	des1
hm	Maximum step size			output input	Global	des1
hmn	Minimum step size when using variable step integration routine	sec		output	Global	des1
hmu	Difference in time used in integration routine	sec		output input	Global	des1
hmx	Maximum step size	sec		output	Global	des1
hnom	Nominal step size	sec			Local	
hold	Nominal step size	sec		output	Global	des1
hp	Maximum step size			output input	Global	des1
hswich	Step size to switch to in Adams Moulton				Local	
i1	Index				Local	
i2	Index				Local	
idl(11)	Integer array that is used as an indicator block for triggers (=1, dependent; =0, independent)			output	Global	des1
idl2	Integer to indicate flag triggers			output input	Global	des1
ifl	Step size control indicator (=1, normal return; =2, step size is too big; =3,			output	Global	des1

Variable	Variable Definition step size is too small)	Units	Source	I/O	Status	Common Block
indis	Flag indicating discontinuity				Local	
irst	Logical variable for integration restart			output	Global	des1
isw	Temporary storage variable for time				Local	
j1000	Index			output input	Global	des2
j2000	Index			output input	Global	des2
j4000	Index			output input	Global	des2
j5500	Index				Local	
j7000	Index			output input	Global	des2
jjss	Internal index in integration routine		dpir	output	Global	des1
jnt	Index				Local	
k1	Index				Local	
kind	Integration type flag for forward trajectory			output	Global	des1
kj	Index (k-j)				Local	
kk	Index (k-1)				Local	
kon	Control index for control of Runge-Kutta or Adams-Moulton logic in main loop			output	Global	des1
kr(1)	Error indicator in integration routine		dpir	output	Global	des1
lor	Index				Local	
m	Number of derivatives			input	Global	des1
mm	Index				Local	
mp1	Index in integration routine			input	Global	des1
mp2	Index in integration routine			input output	Global	des1
n	Index				Local	
nbdif	Index used in integration routine		desolv	output input	Global	des1
ndvt	Number of dependent triggers			output	Global	des1
neq	Number of differential equations to integrate			input output	Global	des1
nivt	Number of independent triggers			output	Global	des1

Variable	Variable Definition	Units	Source	I/O input	Status	Common Block
nm	The total number of possible first order differential equations to be solved			input	Global	des1
nmm	Index				Local	
nmx	The total number of possible first order differential equations to be solved			input	Global	des1
nn	Index in integration routine			output	Global	des1
nnn	Number of equations to integrate				Local	
nnt	Trigger index in integration routine			output input	Global	des1
nold	Actual number of differential equations to solve in integration routine			output	Global	des1
nstart	Flag in integration routine			output input	Global	des1
ntr	Number of triggers			output input	Global	des1
ntr1	Trigger flag for 1st event in calling argument				Local	
ntr11	Trigger flag for 11th event in calling argument				Local	
ntr2	Trigger flag for 2nd event in calling argument				Local	
ntr3	Trigger flag for 3rd event in calling argument				Local	
ntr4	Trigger flag for 4th event in calling argument				Local	
ntr5	Trigger flag for 5th event in calling argument				Local	
ntr6	Trigger flag for 6th event in calling argument				Local	
ntr7	Trigger flag for 7th event in calling argument				Local	
ntr8	Trigger flag for 8th event in calling argument				Local	
ntr9	Trigger flag for 9th event in calling argument				Local	
ntrg	Number of triggers			input	Global	des1
ntt	Internal flag in integration routine			input output	Global	des1
odd	Logical flag to define odd events				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
pdt	Half the step size				Local	
prod	Temporary variable				Local	
py	Saved values of integrated equations				Local	
q	Temporary variable			output	Global	des1
qq(10)	Temporary variables used to calculate coefficients of differences			output input	Global	des1
r	Temporary variable used in integration			output input	Global	des1
redot	Logical variable used to advoid recalculation of the differential equation routines			output	Global	des1
rj(11)	Temporary array used for the step size control associated with triggers			output input	Global	des1
sk	Temporary variable used in integration routine			output input	Global	des1
skp	Temporary variable used in integration routine			output input	Global	des1
sma	Logical flag to denote stops in integration to define triggers			output	Global	des1
stepng	Logical variable used for step size control in integration routine			output	Global	des1
sum	Temporary variable				Local	
t	Time	sec		output input	Global	des2
tc	Time variable				Local	
temp	Temporary variable				Local	
tgo	Time to go in integration routine	sec		output input	Global	des2
tl	Temporary variable in integration	sec		output input	Global	des2
tmin	Minimum time when defining step size in integration routine	sec		output input	Global	des2
tp1	Temporary storage variable for time	sec		output input	Global	des2
tp2	Temporary storage variable for time	sec		output input	Global	des2
tr	Temporary variable used in integration			input output	Global	des2
trip	Stored time used in calculation of step	sec		output	Global	des2

Variable	Variable Definition size in integration	Units	Source	I/O input	Status	Common Block
ts	Saved value of time (or independent parameter) in integration routine	sec		output input	Global	des2
tsv	Saved value of time (or independent parameter) in integration routine	sec		output input	Global	des2
var	Name of variables being tested			output input	Global	des1
var1	Name of the variable being tested for the 1st event in calling argument				Local	
var11	Name of the variable being tested for the 11th event in calling argument				Local	
var2	Name of the variable being tested for the 2nd event in calling argument				Local	
var3	Name of the variable being tested for the 3rd event in calling argument				Local	
var4	Name of the variable being tested for the 4th event in calling argument				Local	
var5	Name of the variable being tested for the 5th event in calling argument				Local	
var6	Name of the variable being tested for the 6th event in calling argument				Local	
var7	Name of the variable being tested for the 7th event in calling argument				Local	
var8	Name of the variable being tested for the 8th event in calling argument				Local	
var9	Name of the variable being tested for the 9th event in calling argument				Local	
wj(11)	Temporary variables used inintegration routine			output input	Global	des1
xlj(11)	Temporary variable used in integration routine			output input	Global	des1
y	Limit on dependent variables			output input	Global	des1
yl	The smallest value of a dependent variable that will affect the automatic step size logic			output input	Global	des1
yn	Temporary storage array				Local	
zv	The value of the variable which the trigger interrupt routine is to be executed			output input	Global	des1
zv1	The value of var at which the trigger				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
	interrupt routine is to be executed for the 1st event in calling argument					
zv11	The value of var at which the trigger interrupt routine is to be executed for the 11th event in calling argument				Local	
zv2	The value of var at which the trigger interrupt routine is to be executed for the 2nd event in calling argument				Local	
zv3	The value of var at which the trigger interrupt routine is to be executed for the 3rd event in calling argument				Local	
zv4	The value of var at which the trigger interrupt routine is to be executed for the 4th event in calling argument				Local	
zv5	The value of var at which the trigger interrupt routine is to be executed for the 5th event in calling argument				Local	
zv6	The value of var at which the trigger interrupt routine is to be executed for the 6th event in calling argument				Local	
zv7	The value of var at which the trigger interrupt routine is to be executed for the 7th event in calling argument				Local	
zv8	The value of var at which the trigger interrupt routine is to be executed for the 8th event in calling argument				Local	
zv9	The value of var at which the trigger interrupt routine is to be executed for the 9th event in calling argument				Local	

```
      SUBROUTINE DPIR (TC,Y,DY,PY,YN,DELZ,DELY,DELX, NNN,LOR,
     X                 H, HNOM, N,  DER1, DER2, EOS,
     X                 NTR1, RT1, VAR1, ZV1,  NTR2, RT2, VAR2, ZV2,
     X                 NTR3, RT3, VAR3, ZV3,  NTR4, RT4, VAR4, ZV4,
     X                 NTR5, RT5, VAR5, ZV5,  NTR6, RT6, VAR6, ZV6,
     X                 NTR7, RT7, VAR7, ZV7,  NTR8, RT8, VAR8, ZV8,
     X                 NTR9, RT9, VAR9, ZV9,  NTR10,RT10,VAR10,ZV10,
     X                 NTR11,RT11,VAR11,ZV11 )
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      logical graph,demand
      COMMON/CCC/GRAPH,JO,DEMAND
      DIMENSION TC(2), Y(NNN), DY(NNN), PY(NNN), YN(NNN)
C     DOUBLE PRECISION  PY, YN
      DIMENSION DELX(NNN,LOR),  DELY(NNN,LOR), DELZ(NNN,LOR)
C
      COMMON/MOD1/ACO(10),BCO(10)
C
      COMMON /DES1/
     *   KR(1),   NMX,   NEQ, NOLD,    M,  MP1,  MP2,   NN, KIND,  KON,
     *     IFL, NTRG, NIVT, NDVT,  NTR,NSTART,NBDIF,  NNT,  NTT,IDL(11),
     *    IDL2, JJSS,J1000,J2000,J4000,J7000, IRST, EOST,   HD,  SMA,
     *   ERROR,REDOT,STEPNG,
     *   EU(1),   EL,  HMX,  HMN,   YL,    A, ENP1,    D,    B,  SKP,
     *      SK,   HC, DELU, DUSC, HOLD,  VAR,   ZV,   HM,   HP,    R,
     *       Q, DELT,  HMU,  FMU,CK(10),QQ(10),XLJ(11), RJ(11), WJ(11)
      COMMON /DES2/     T,  TSV,  TGO, TMIN, TRIP,   TL,   TR
C     DOUBLE PRECISION   T,  TSV,  TGO, TMIN, TRIP,   TL,   TR
      LOGICAL EOST, SMA, HD, IRST, ERROR
      LOGICAL  STEPNG
      LOGICAL  ODD, REDOT
      DATA BIG/1.0D30/
C
      GO TO  118
 3525 TC(1)=TC(1)
      Y(1)=Y(1)
      DY(1)=DY(1)
      PY(1)=PY(1)
      YN(1)=YN(1)
      DELZ(1,1)=DELZ(1,1)
      DELY(1,1)=DELY(1,1)
      DELX(1,1)=DELX(1,1)
      NNN=NNN
      LOR=LOR
      H=H
      HNOM=HNOM
      N=N
      NTR1=NTR1
      NTR2=NTR2
      NTR3=NTR3
      NTR4=NTR4
      NTR5=NTR5
      NTR6=NTR6
      NTR7=NTR7
      NTR8=NTR8
```



```
      NTR9=NTR9
      NTR10=NTR10
      NTR11=NTR11
      VAR1=VAR1
      VAR2=VAR2
      VAR3=VAR3
      VAR4=VAR4
      VAR5=VAR5
      VAR6=VAR6
      VAR7=VAR7
      VAR8=VAR8
      VAR9=VAR9
      VAR10=VAR10
      VAR11=VAR11
      ZV1=ZV1
      ZV2=ZV2
      ZV3=ZV3
      ZV4=ZV4
      ZV5=ZV5
      ZV6=ZV6
      ZV7=ZV7
      ZV8=ZV8
      ZV9=ZV9
      ZV10=ZV10
      ZV11=ZV11
  100 IRST = .TRUE.
      RETURN
C
C     SET INITIAL VALUES OF STORAGE AREAS TO ZERO
  118 CONTINUE
      DO  120  I=1,NMX
      PY(I)=Y(I)
  120 YN(I) = 0.
      HD = .TRUE.
      SMA = .TRUE.
      GO TO (121, 126, 121),  KIND
C     FOR A-M MUST ZERO OUT DIFFERENCE TABLES
  121 DO  125  I=1,NMX
      DO  125  J=1,MP2
      DELX(I,J) = 0.
      DELY(I,J) = 0.
  125 DELZ(I,J) = 0.
      A = -ACO(MP1)/BCO(MP2)
  126 HC = H
C
C     TO SET UP INDICATOR BLOCK FOR TRIGGERS THAT ARE DEPENDENT
C          (IDL = 1)  OR INDEPENDENT (IDL = 0),  ALSO CALCULATES THE
C          NUMBER OF EACH
      NIVT = 0
      IF (NTRG)  131, 131, 128
  128 TEMP=TC(1)
      TC(1)=-BIG
      JJSS = 2
      DO  130  J=1,NTRG
      IDL(J) = 1
      JNT=J
```

```
      GO TO  8000
   58 VAR=VAR
      IF(VAR.NE.-BIG) GO TO 60
C     THIS INDICATES A T-STOP, SINCE T(1)
C     WAS SET EQUAL TO -BIG AT THE BEGINNING OF THIS LOOP
      NIVT = NIVT+1
      IDL(J)=0
   60 CONTINUE
      XLJ(J) = 0.
      RJ(J) = 0.
  130 WJ(J) = 0.
      TC(1) = TEMP
  131 T = TC(1)
   62 NDVT = NTRG - NIVT
C
C     CALC DERIVATIVES AT THE CURRENT TIME
      GO TO  (172, 171), NN
  171 CALL  DER1
  172 CALL  DER2
C     GO TO ROUTINE TO GET THE CURRENT TMIN AND TRIGGER ROUTINE
C      AND NUMBER THAT GOES WITH IT.  ROUTINE IS AT 1000
      J1000 = 1
      GO TO  1000
  205 CONTINUE
C
C     GO TO END-OF-STEP CALCULATIONS, IF ANY
      IF (EOST)  CALL EOS
      IF(KR(1).NE.0) GO TO 4910
      DELU = T
      IF (HC .GT. DELU)  DELU=HC
      DELU = DUSC*DELU
C
C*************** MAIN  LOOP  ***************************************
C
  300 CONTINUE
      INDIS = 0
C     TEST FOR CALCULATION ERROR
  310 IF (TMIN + DELU .LT. T) THEN
          WRITE(6,*)'TMIN ',TMIN,' IS < TIME ',T,' - DELU  ',DELU
          WRITE(6,*)'RESETTING TMIN TO T-DELU+.01 AND CONTINUING'
          TMIN=T-DELU+.01
C         GOTO 9000
      ENDIF
      TS = DABS(TMIN-T)
  315 IF(TS.GT.DELU)GO TO 350
C
C   EXECUTE INDEPENDENT VARIABLE TRIGGER ROUTINE
      NNT = NTT
      J4000 = 1
      GO TO  4000
C          TEST FOR A DISCONTINUITY CAUSING A RESTART
  320 CONTINUE
      IF (INDIS)  100, 325, 100
C     AFTER EXECUTING TRIGGER, RE-CALC. NEW TMIN
  325 J1000 = 2
      GO TO  1000
```



```
  330 GO TO  310
C
C     CALCULATE L'S FOR EACH Y-STOP
  350 IF (NDVT .EQ. 0)  GO TO  380
      TL = T
      JJSS = 3
      DO  375  J=1,NTRG
      JNT=J
      IF (IDL(J))  360, 375, 360
  360 GO TO  8000
  365 NTR=NTR
      ZV=ZV
      VAR=VAR
      IF(NTR) 375,370,370
  370 XLJ(J) = VAR - ZV
      IF(XLJ(J)) 375,371,375
  371 IDL(J)=3
      GO TO 3500
  375 CONTINUE
C
C  MOVE ONE STEP FORWARD,  METHOD DEPENDS ON KON
  380 GO TO ( 5000 , 6000 ) , KON
C
C  ADAMS - MOULTON CONTROL LOGIC
C
 5000 CONTINUE
      IF (DABS(T-TGO) .GT. DELU)  GO TO  5400
C     MOVE FORWARD ONE HC STEP
      J5500 = 1
      GO TO  5500
C     ON RETURN FROM ADAMS TEST FOR STEP-SIZE FLAG
 5020 GO TO (5100, 5200, 5300),  IFL
C
C     NORMAL RETURN,  LEAVE STEP SIZE ALONE
 5100 CONTINUE
      IF (STEPNG) GO TO  5150
 5105 TGO = T
      DELU = T
      IF (HC .GT. DELU)  DELU=HC
      DELU = DUSC*DELU
      GO TO  5400
C
C     AT END OF FIRST HC AFTER SWITCHING TO ADAMS MODE, TEST HC TO SEE
C     IF INITIAL STEP WAS TOO BIG.  IF SO RETURN CONTROL TO CALLING PROG
 5110 FORMAT (//22H  INITIAL STEP SIZE OF, 1PE12.5,
     *         17H  HAS BEEN CUT TO, E12.5)
 5150 STEPNG = .FALSE.
      IF (HC .GE. HSWICH)  GO TO  5105
C     STEP HAS BEEN CUT DOWN
      WRITE (JO,5110)  HSWICH, HC
      KR(1)=100
      GO TO  9999
C
C     STEP SIZE IS TOO BIG.  HALVE STEP SIZE AND RESET CONDITIONS AT
C          BEGINNING OF STEP.  DIFFERENCE TABLES MUST BE REGENERATED FOR HC/2.
 5200 HC = HC/2.0
```

```
      H = HC
C
C     USING DIFFERENCES IN DELX GET (M+1) DERIVATIVES BACK FROM THIS
C     TIME AND STORE IN DELZ.   (DELX STORAGE DESTROYED)
      DO 5220  I=1,NEQ
      DO 5210  J=1,MP1
      MM= MP2-J
      DELZ(I,J)=DELX(I,1)
      DO 5210  K=1,MM
 5210 DELX(I,K) = DELX(I,K) - DELX(I,K+1)
 5220 DELZ(I,MP2) = DELX(I,1)
C
C     USING THESE DERIVATIVES IN DELZ AND DERIVATIVES AT STEPS HALF WAY
C     BETWEEN THESE (WHICH ARE CALCULATED USING BACKWARD DIFFERENCE
C     INTERPOLATION), SET UP A TABLE OF (M+1) BACK DERIVATIVES
C     USING HC/2 AND STORE IN DELY
      FM=M
      FMP1=MP1
C          CALC.  (M+1) FACTORIAL
      FFMP1=1.0
      DO 5230  J=2,MP1
      FJ=J
 5230 FFMP1=FFMP1*FJ
C
      ODD= .FALSE.
      DO 5270  NM=1,MP2
      IF  (ODD)  GO TO 5250
C
C      USE Y DOT FROM DELZ       (N=0,2,4,---)
      NMM= NM/2 + 1
      DO 5240 I=1,NEQ
 5240 DELY (I,NM) = DELZ(I,NMM)
      GO TO 5270
C
C       CALC.  Y DOT HALF WAY BETWEEN  (N=1,3,5,  ---)
 5250 FN=NM-1
      FMU=-FN/2.0
      PROD=1.
      DO 5255  J=1,MP1
      FJ=J
 5255 PROD = PROD*(FMU+FJ)
C       CALC.  A(0)
      DO 5265  I=1,NEQ
      AK = PROD/FFMP1
      SUM = AK*DELZ(I,1)
C
      DO 5260 K=2,MP2
      FK=K-2
      AK=-AK*(FMU+FK)/(FMU+FK+1.) * (FMP1-FK)/(FK+1.)
 5260 SUM = SUM+ AK*DELZ(I,K)
 5265 DELY (I,NM) = SUM
C
 5270 ODD = .NOT. ODD
C
C     NOW HAVE (M+1) BACK DERIVATIVES AT HC/2 SAVED IN DELY
C     RE-GENERATE NEW BACKWARD DIFFERENCE TABLES IN DELX
```



```
      K1=MP2
 5275 DO 5280  I=1,NEQ
      TP1 = DELY(I,K1)
      DO 5280  K=1,MP2
      TP2=TP1
      TP1=TP1-DELX(I,K)
 5280 DELX(I,K) = TP2
      K1=K1-1
      IF(K1)   5285, 5285, 5275
C
C     GO TO INTEGRATE ONE ADAMS STEP AGAIN WITH HALF THE STEP SIZE.
C          SETTING J5500=1 FORCES RETURN TO STATEMENT 5020
 5285 J5500=1
      GO TO 5500
C
C     STEP SIZE IS SMALL.  PREPARE TO DOUBLE HC.
 5300 HD = .FALSE.
      FMP1 = MP1
      TRIP = T + FMP1*HC
      NSTART = -1
C
C
C     MOVE DELX TO DELZ AND USING THE DIFFERENCES FROM HERE GET THE
C     M+1 BACK DERIVATIVES AND SAVE THEM IN DELY
C
C          DEL(T-HC)**R = DEL(T)**R - DEL(T)**(R+1)
      DO 5310  I=1, NEQ
      DO  5310  K=1, MP2
 5310 DELZ(I,K) = DELX(I,K)
C
C
      DO 5350  I=1, NEQ
      DO  5330  J=1, MP1
      MM = MP2 - J
      DELY(I,J) = DELZ(I,1)
      DO  5330  K=1,MM
 5330 DELZ(I,K) = DELZ(I,K)-DELZ(I,K+1)
 5350 DELY (I,MP2) = DELZ(I,1)
C
      GO TO 5100
C
C
 5400 IF (TMIN .LE. (TGO-DELU))  GO TO  5430
C     ALL T-STOPS IN THIS STEP HAVE BEEN EXECUTED
C     SET CONDITIONS BACK TO TGO
      T = TGO
      TC(1)=T
      DO 5410  I=1,NEQ
      Y(I) = PY(I)
 5410 DY(I) = DELX(I,1)
C
      IF (REDOT)  GO TO 5425
      GO TO (5416,5415), NN
 5415 CALL DER1
 5416 CALL DER2
      REDOT = .TRUE.
```

```
 5425 IF (EOST)  CALL EOS
      IF(KR(1).NE.0) GO TO 4910
      GO TO  6100
C
C     INTERPOLATE FOR CONDITIONS AT TMIN
 5430 T = TMIN
      TC(1)=T
      J7000 = 1
      GO TO  7000
 5450 GO TO  6100
C
C  USE RUNGE-KUTTA THIS STEP
C      CALCULATE STEP SIZE TO USE
 6000 CONTINUE
      DT=H
      IF(TS.LT.DT) DT=TS
      J2000 = 1
      GO TO  2000
 6010 TGO = T
      DELU = T
      IF (HC .GT. DELU)  DELU=HC
      DELU = DUSC*DELU
C
C     GO TO END-OF-STEP
      IF (EOST)  CALL EOS
      IF(KR(1).NE.0) GO TO 4910
C     TEST FOR ANY Y-STOPS IN THIS STEP
C
C                FLAG  ROUTINE
 6100 CONTINUE
      IDL2 = 0
      IF (NDVT)  6105, 300, 6105
 6105 JJSS = 4
C     FLAG TRIGGERS THAT HAVE HAD A CHANGE OF SIGN IN THIS STEP BY
C          SETTING THE INDICATOR IDL = 2
      DO 6150  J=1,NTRG
      JNT=J
      IF (IDL(J)) 6110, 6150, 6110
 6110 GO TO  8000
 6120 NTR=NTR
      ZV=ZV
      VAR=VAR
      IF(NTR) 6150,6130,6130
 6130 TEMP = VAR - ZV
      IF (SIGN(1.D0,TEMP) .NE. SIGN(1.D0,XLJ(J)) )  GO TO 6140
      IDL(J) = 1
      GO TO  6150
 6140 IDL(J) = 2
      IDL2 = IDL2+1
      RJ(J) = TEMP
      WJ(J) = TEMP
 6150 CONTINUE
C
      IF (IDL2 .EQ. 0)  GO TO  300
C
C
```



```
C      SEARCH ROUTINE
C        SEARCH AND FLAG FIRST  Y-STOP IN THIS TIME STEP
 3000 CONTINUE
C
      HM = BIG
      HP = BIG
      JJSS = 5
      DO 3100  J=1,NTRG
      IF (IDL(J) .LT. 2)  GO TO 3100
      JNT=J
      GO TO  8000
 3110 NTR=NTR
      ZV=ZV
      VAR=VAR
      IF(NTR) 3100,3115,3115
 3115 WJ(J) = VAR - ZV
      IF (SIGN(1.D0,WJ(J)) .NE. SIGN(1.D0,XLJ(J)))  GO TO  3120
      IF (SIGN(1.D0,WJ(J)) .NE. SIGN(1.D0,RJ(J)))  GO TO  3140
C      UNFLAG DEPENDENT VARIABLE J
      IDL(J) = 1
      GO TO  3100
C
 3120 R = WJ(J)*(TL-T)/(WJ(J)-XLJ(J))
      IF (R .LT. HM)  HM = R
      GO TO  3150
C
 3140 R = WJ(J)*(TR-T)/(WJ(J)-RJ(J))
      IF (R .LT. HP)  HP = R
 3150 Q = R
      IDL(J) = 2
      IF (ABS(Q) .LT. DELU)  IDL(J) = 3
 3100 CONTINUE
C
 3180 IF (HM .EQ. BIG)  GO TO 3220
      DO 3200  J=1,NTRG
 3200 RJ(J) = WJ(J)
      TR = T
      DELT = HM
      GO TO  3260
 3220 DO 3240  J=1,NTRG
 3240 XLJ(J) = WJ(J)
      TL = T
      DELT = HP
 3260 IF (ABS(DELT) .LT. DELU)  GO TO  3500
C
C     NOT CONVERGED
      GO TO  (3300, 3400), KON
C     FOR ADAMS-MOULTON, INTERPOLATE FOR Y'S AT T + DELT
C          GET DERIVATIVES AND EOS CONDITIONS
 3300 T = T + DELT
      TC(1)=T
      J7000 = 2
      GO TO  7000
 3350 GO TO  3000
C
C     FOR RUNGE-KUTTA, INTEGRATE TO T + DELT
```

```
 3400 DT = DELT
      J2000 = 2
      GO TO  2000
 3450 IF (EOST)  CALL EOS
      GO TO  3000
C
C     SEARCH HAS CONVERGED WITHIN ACCEPTABLE TOLERANCE ON ONE OF
C          DEPENDENT VARIABLE TRIGGERS
 3500 CONTINUE
      INDIS = 0
      DO 3540  J=1,NTRG
      JJSS = 6
      JNT=J
      GO TO 8000
 3520 NTR=NTR
      IF(NTR) 3540,3530,3530
 3530 IF (IDL(J) .NE. 3)  GO TO  3540
C     GO TO EXECUTE TRIGGER
      NNT = J
      IDL(J) = 1
      J4000 = 2
      GO TO  4000
 3540 CONTINUE
C
C     CALC. NEW TMIN AND NTT
      J1000 = 5
      GO TO  1000
 3600 CONTINUE
C
      GO TO (3680, 3700), KON
C     TEST FOR DISCONTINUITIES, CAUSES A RESTART IN ADAMS
 3680 IF (INDIS .EQ. 0)  GO TO  350
      GO TO  100
C
C     TEST FOR DISCONTINUITIES, CAUSES DERIVATIVES TO BE RE-CALCULATED
C          IN KUTTA
 3700 TGO = T
      IF (INDIS .EQ. 0)  GO TO  300
C
      GO TO (3752, 3751), NN
 3751 CALL DER1
 3752 CALL DER2
      IF (EOST)  CALL EOS
      IF(KR(1).NE.0) GO TO 4910
      GO TO  300
C
C
C*********************************************************************
C  ROUTINE TO CALCULATE CURRENT TMIN FROM LIST OF T-STOPS
C     ALSO CHECKS FOR ANY CHANGE IN ARGUMENTS THAT WILL CAUSE A RESTART
C          STARTS AT 1000 AND RETURNS BY A CALCULATED GO TO USING J1000
C
 1000 CONTINUE
      TMIN = TRIP
      NTT = NSTART
      IF (NIVT)  1060, 1060, 1010
```



```
 1010 JJSS = 1
      DO 1050  J=1,NTRG
      JNT=J
      IF (IDL(J)) 1050, 1020, 1050
 1020 GO TO 8000
 1030 NTR=NTR
      ZV=ZV
      IF(NTR) 1050,1040,1040
 1040 IF(ZV.GT.TMIN)GO TO 1050
      IF(ABS(ZV-TMIN).LE.DELU)GO TO 1050
      TMIN = ZV
      NTT = J
 1050 CONTINUE
 1060 CONTINUE
C     CHECK POSSIBILITY OF A RESTART DUE TO A CHANGE IN ARGUMENTS
      IF (NOLD - N)  100, 1075, 1070
 1070 NOLD = N
      NEQ = N
 1075 IF(ABS(HNOM-HOLD).LE..01D0) GO TO 1080
C     THE NOMINAL STEP SIZE HAS BEEN CHANGED EXTERNALLY
C          MUST RESTART.
      H = HNOM
      HOLD = HNOM
      GO TO  100
C
 1080 GO TO(205,330,5505,5570,3600,350),J1000
C
C*********************************************************************
C  FOURTH ORDER RUNGE-KUTTA ROUTINE
C     TO MOVE ONE STEP FROM T TO T + DT (INITIALIZE DT BEFORE CALLING)
C               USES YN TO SAVE PY(T)
C                    PY AND Y CALC. TOGETHER TO T+DT
C                    DELZ(I,1)  TO SAVE SUM OF K'S
C          RETURN IS BY J2000
C
 2000 TSV = T
      PDT = DT/2.
C     SAVE INITIAL Y'S IN YN
      DO 2002  I=1,NEQ
 2002 YN(I) = PY(I)
C
C          OUTER LOOP
      DO 2060  L=1,4
C          INNER LOOP   FOR ALL N EQUATIONS
      DO 2050  I=1,NEQ
      SK = DT*DY(I)
      GO TO  ( 2010, 2020, 2030, 2040),  L
C     L=1, SK=K1
 2010 DELZ(I,1) = SK
 2015 PY(I) = YN(I) + SK/2.
      Y(I) = PY(I)
      GO TO 2050
C     L=2, SK=K2
 2020 DELZ(I,1) = DELZ(I,1)+2.*SK
      GO TO  2015
C     L=3, SK=K3
```

```
 2030 DELZ(I,1) = DELZ(I,1) + 2.*SK
      PY(I) = YN(I) + SK
      Y(I) = PY(I)
      GO TO  2050
C     L=4, SK=K4
 2040 PY(I) = YN(I) + (DELZ(I,1) + SK)/6.
      Y(I) = PY(I)
 2050 CONTINUE
C
      GO TO  ( 2051, 2057, 2053, 2057),  L
 2051 T = TSV + PDT
      GO TO  2055
 2053 T = TSV + DT
 2055 TC(1)=T
      GO TO  ( 2057, 2056),  NN
 2056 CALL DER1
 2057 CALL DER2
 2060 CONTINUE
C
C          RETURN
      GO TO ( 6010,  3450),  J2000
C*******************************************************************************
C*******************-***********************************************************
C
C     CALLS TO EXECUTE TRIGGER ROUTINES.  NUMBER IS SET IN NNT
C                NNT MAY EQUAL ZERO WHEN GETTING THE FIRST M+2
C                    DIFFERENCES BY RUNGE-KUTTA BEFORE SWITCHING TO ADAMS
C
C                NNT MAY EQUAL -1 WHEN ENOUGH DIFFERENCES ARE SAVED
C                    TO DOUBLE THE STEP SIZE
C          RETURN IS BY J4000
C
 4000 CONTINUE
      IF (NNT)  4500,  4400,  4005
C
 4005 IF (NNT .GT. NTRG)  GO TO  4300
      GO TO  (4010,4020,4030,4040,4050,4060,4070,4080,4090,4100,
     X         4110),  NNT
C
 4010 CALL RT1
      GO TO  4900
 4020 CALL RT2
      GO TO  4900
 4030 CALL RT3
      GO TO  4900
 4040 CALL RT4
      GO TO  4900
 4050 CALL RT5
      GO TO  4900
 4060 CALL RT6
      GO TO  4900
 4070 CALL RT7
      GO TO  4900
 4080 CALL RT8
      GO TO  4900
 4090 CALL RT9
```



```
      GO TO  4900
 4100 CALL RT10
      GO TO  4900
 4110 CALL RT11
      GO TO  4900
C
 4310 FORMAT (///31H TRYING TO CALL TRIGGER ROUTINE, I3,
     X          21H WHICH IS NOT DEFINED/)
 4300 WRITE (JO,4310)  NNT
      GO TO  9900
C
C
C     ROUTINE TO EXECUTE WHEN TAKING THE FIRST M+2 RUNGE-KUTTA STEPS
C          AND GENERATING THE NECESSARY DIFFERENCE TABLES.
C          WHEN NBDIF = M+2 THIS TRIGGER IS SHUT OFF AND CONTROL IS
C          TRANSFERED TO ADAMS CONTROL.
 4400 CONTINUE
      TRIP = T + HC
C          UPDATE DIFFERENCES TABLES
      DO 4410  I=1,NEQ
      TP1 = DY(I)
      DO 4410  K=1,MP2
      TP2 = TP1
      TP1 = TP1 - DELX(I,K)
 4410 DELX(I,K) = TP2
C
      NBDIF = NBDIF+1
      IF (NBDIF .LT. MP2)  GO TO 4995
C
C     WHEN ENOUGH STEPS HAVE BEEN TAKEN, SWITCH TO ADAMS-MOULTON
      KON = 1
      HSWICH = HC
      TRIP = BIG
      NSTART = 100
      GO TO  325
C
C     ROUTINE TO DOUBLE THE STEP SIZE AFTER ENOUGH DIFFERENCES
C          HAVE BEEN SAVED
C     TIME OF THIS TRIGGER IS (M+1)*HC AFTER THE TIME WHEN THE DOUBLING
C     PROCEDURE WAS SET OFF
 4500 CONTINUE
      HC = 2.*HC
      H = HC
      HD = .TRUE.
      TRIP = BIG
      NSTART = 100
C
C     USING DIFFERENCES IN DELX GET (M+1) DERIVATIVES BACK FROM THIS
C     TIME AND STORE THESE IN DELZ
      DO 4530 I=1,NEQ
      DO 4520 J=1,MP1
      MM = MP2-J
      DELZ(I,J) = DELX(I,1)
      DO 4520 K=1,MM
 4520 DELX(I,K) = DELX(I,K) - DELX(I,K+1)
 4530 DELZ(I,MP2) = DELX(I,1)
```

```
C
C     FROM DERIVATIVES IN DELY AND DELZ, SET DELY= BACK DERIVATIVES
C     WITH 2*HC TIME INTERVAL BETWEEN
      ASSIGN 4542 TO ISW
      I1= MP2+1
      I2 = MP2+2
 4535 I1 = I1-1
      IF (I1) 4560, 4560, 4536
 4536 I2 = I2-2
      IF (I2) 4538, 4538, 4540
C     SWITCH TO DELZ TABLES
 4538 ASSIGN 4544 TO ISW
      I2 = I2 + MP1
C
C     IN SAME LOOP, RE-CALC. NEW DIFFERENCE TABLES IN DELX BASED ON NEW
C     TIME STEP
 4540 DO 4550   I=1,NEQ
      GO TO ISW,  (4542,4544)
 4542 DELY(I,I1) = DELY(I,I2)
      GO TO 4545
 4544 DELY (I,I1) = DELZ(I,I2)
 4545 TP1 = DELY(I,I1)
      DO 4550 K=1,MP2
      TP2 = TP1
      TP1 = TP1-DELX(I,K)
 4550 DELX(I,K) = TP2
C
      GO TO 4535
C
 4560 GO TO 4995
C
C
C     AFTER EXECUTING AN EXTERNAL TRIGGER ROUTINE,  COME HERE
 4900 CONTINUE
C          HAS RTMRK BEEN CALLED
      IF(KR(1).EQ.0) GO TO 4920
C          YES, RETURN TO CALLING PROGRAM
 4910 KR(1)=0
      GO TO 9999
C
C     TEST FOR A DISCONTINUITY AFTER TRIGGER EXECUTION
 4920 JNT = NNT
      JJSS = 7
      GO TO 8000
 4930 CONTINUE
      VAR=VAR
C
C     IF THERE IS A DISCONTINUITY, SET INDICATOR
      IF ( IABS(NTR) .EQ. 2)  INDIS = 1
C
 4995 GO TO ( 320 , 3540 ) , J4000
C
C
C*********************************************************************
C
C ADAMS - MOULTON M-TH ORDER PREDICTOR - CORRECTOR ROUTINE WITH
```



```
C     AUTOMATIC STEP SIZE CONTROL
C     MOVES ONE STEP FORWARD USING CURRENT STEP SIZE, HC
C     RETURNS WITH AN INDICATOR,  IFL
C
C
C          IF IFL = 1, NORMAL RETURN
C                      TIME IS UPDATED
C                      Y'S ARE IN Y AND PY
C                      Y DERIVATIVES ARE IN DY
C                      BACK DIFFERENCES UPDATED AND IN DELX
C
C          IF IFL = 2, INDICATES STEP SIZE IS TOO BIG
C                      TIME IS SET BACK TO BEGINNING OF STEP
C                      OLD Y'S ARE PUT BACK INTO Y AND PY
C                      OLD BACK DIFFERENCES ARE STILL IN DELX
C                      OLD DERIVATIVES SET IN DY FROM FIRST COL. OF DELX
C                      END-OF-STEP CONDITIONS ARE RE-CALCULATED
C                      ANY DOUBLING INDICATORS ARE SHUT OFF
C
C          IF IFL = 3, INDICATES STEP SIZE IS TOO SMALL
C                      STATE OF STORAGE IS THE SAME AS FOR NORMAL RETURN
C
C     RETURN IS BY J5500
C
 5500 CONTINUE
      REDOT = .TRUE.
      IF (SMA)  GO TO 5505
C     GET NEW TMIN AND NTT
      SMA = .TRUE.
      J1000 = 3
      GO TO 1000
 5505 TSV = T
C     SAVE CURRENT Y'S IN YN AND CALC PREDICTED Y'S
C
      DO 5520  I=1,NEQ
      YN(I) = PY(I)
      SK = 0.
C          SUM FROM K=0 TO M
      DO 5510  K=1,MP1
 5510 SK = SK + ACO(K)*DELX(I,K)
      PY(I) = YN(I) + HC*SK
 5520 Y(I) = PY(I)
C     CALCULATE DERIVATIVES OF PREDICTED Y'S AT T + HC
      T = TSV + HC
      TC(1)=T
      GO TO (5526, 5525),  NN
 5525 CALL DER1
 5526 CALL DER2
C
      IFL=1
      GO TO (5528, 9900, 5590), KIND
C     LEAVE DIFFERENCES AT BEG. OF STEP IN DELX
C     UPDATE DIFFERENCE TABLES FROM DELX USING PREDICTED
C     DERIVATIVES IN DELZ
 5528 CONTINUE
      DO 5530  I=1,NEQ
```

```
      TP1 = DY(I)
      DO 5530  K=1,MP2
      TP2 = TP1
      TP1 = TP1 - DELX(I,K)
 5530 DELZ(I,K) = TP2
C
C     CALCULATE CORRECTED Y'S AND GET LARGEST ERROR TERM
      ENP1 = 0.
      DO 5550  I=1,NEQ
      SKP=0.
      SK = 0.
      DO 5540  K=1,MP1
      SKP = SKP+ACO(K)*DELX(I,K)
 5540 SK = SK + BCO(K)*DELZ(I,K)
      PY(I) = YN(I) + HC*SK
      Y(I) = PY(I)
      D = ABS(Y(I))
      IF (D .LE. YL)  D = YL
      B = ABS(HC*(SKP-SK))/D/A
 5550 IF (B .GT. ENP1)  ENP1 = B
C     CALC. DERIVATIVES OF CORRECTED Y'S
      CALL DER2
C     TEST ERROR LIMITS FOR AUTOMATIC STEP SIZE CONTROL
      IF (ENP1 .LE. EL)  GO TO 5580
C
      IF(ENP1.LE.EU(1)) GO TO 5590
C     STEP SIZE IS TOO BIG
      IF (HC/2.  .LT. HMN)  GO TO  5590
C     RESET EVERYTHING TO CONDITIONS AT BEGINNING OF STEP
C          INCLUDING END OF STEP CONDITIONS
C          Y(I) AND PY(I) FROM YN(I)
C          DY(I) FROM 1ST COLUMN OF DELX
      T = TSV
      TC(1)=T
      DO 5560  I=1,NEQ
      Y(I) = YN(I)
      PY(I) = YN(I)
 5560 DY(I) = DELX(I,1)
      IF (EOST)  CALL EOS
C
      IFL = 2
      HD = .TRUE.
      TRIP = BIG
      NSTART = 100
      J1000 = 4
C                  TO RECALCULATE TMIN AND NTT
      GO TO  1000
 5570 GO TO  5599
C
C     STEP SIZE IS TOO SMALL
 5580 IF ( .NOT. HD)  GO TO  5590
      IF (2.*HC  .GT.  HMX)  GO TO  5590
      IFL = 3
      SMA = .FALSE.
C
C     UPDATE DIFFERENCE TABLES USING CORRECTED DERIVATIVES
```



```
 5590 DO 5595  I=1,NEQ
      TP1 = DY(I)
      DO 5595  K=1,MP2
      TP2 = TP1
      TP1 = TP1 - DELX(I,K)
 5595 DELX(I,K) = TP2
C
C          RETURN
 5599 GO TO (5020), J5500
C
C*****************************************************************************
C
C  ROUTINE TO INTERPOLATE USING DIFFERENCE FORMULAS FOR THE Y'S AT TIME
C     SET IN T AND USING DIFFERENCE TABLES EVALUATED AT TGO.  BEFORE
C     RETURNING THE DERIVATIVES ARE CALCULATED AND STORED IN DY AND
C     THE END-OF-STEP ROUTINE HAS BEEN EXECUTED.
C            RETURN IS BY  J7000
C
 7000 CONTINUE
      REDOT = .FALSE.
      HMU = TGO - T
      FMU = HMU/HC
      FACTJ = 1.0
      FNUM = 1.0
      DO 7020  J=1,M
      FJ = J
      FACTJ=FACTJ*(FJ+1.)
      FNUM = FNUM*(FMU-FJ)
 7020 QQ(J) =(-1.)**J * FNUM/FACTJ
C     CALCULATE COEFFICIENTS OF DIFFERENCES
      DO 7050  K=1,MP1
      CK(K) = BCO(K)
      IF (K .EQ. 1)  GO TO  7050
      KK = K-1
      DO 7040  J=1,KK
      KJ = K-J
 7040 CK(K) = CK(K) + QQ(J)*BCO(KJ)
 7050 CONTINUE
C
C     CALCULATE NEW Y'S AT T
      DO 7080  I=1,NEQ
      SUM = 0.
      DO 7060  K=1,MP1
 7060 SUM = SUM + CK(K)*DELX(I,K)
 7080 Y(I) = PY(I) - HMU*SUM
C
C     CALCULATE DERIVATIVES
      GO TO  (7092, 7091),  NN
 7091 CALL DER1
 7092 CALL DER2
C          GET EOS CONDITIONS
      IF (EOST)  CALL  EOS
      IF(KR(1).NE.0) GO TO 4910
C
      GO TO (5450, 3350),  J7000
C
```

```
C
C*************************************************************************
C     ROUTINE TO SET UP TRIGGER VARIABLES WITH CURRENT VALUES
 8000 GO TO (1,2,3,4,5,6,7,8,9,10,11),  JNT
C     RETURN TO THESE STATEMENT NOS. DEPENDING ON VALUE OF JJSS
C     JJSS =  1  ,  2  ,  3  ,  4  ,  5  ,  6  ,  7
 8100 GO TO (1030,58,365,6120,3110,3520,4930),JJSS
C
    1 NTR = NTR1
      VAR = VAR1
      ZV = ZV1
      GO TO  8100
    2 NTR = NTR2
      VAR = VAR2
      ZV = ZV2
      GO TO  8100
    3 NTR = NTR3
      VAR = VAR3
      ZV = ZV3
      GO TO  8100
    4 NTR = NTR4
      VAR = VAR4
      ZV = ZV4
      GO TO  8100
    5 NTR = NTR5
      VAR = VAR5
      ZV = ZV5
      GO TO  8100
    6 NTR = NTR6
      VAR = VAR6
      ZV = ZV6
      GO TO  8100
    7 NTR = NTR7
      VAR = VAR7
      ZV = ZV7
      GO TO  8100
    8 NTR = NTR8
      VAR = VAR8
      ZV = ZV8
      GO TO  8100
    9 NTR = NTR9
      VAR = VAR9
      ZV = ZV9
      GO TO  8100
   10 NTR = NTR10
      VAR = VAR10
      ZV = ZV10
      GO TO  8100
   11 NTR = NTR11
      VAR = VAR11
      ZV = ZV11
      GO TO  8100
C
C*************************************************************************
C*************************************************************************
C
```



```
C
 9000 WRITE (JO,9010)  TMIN, T, DELU
 9010 FORMAT (///10H T(MIN) = ,1PE10.3,28H IS LESS THAN CURRENT TIME, ,
     X           E10.3, 17H, MINUS DELTA U, ,E10.3/)
C
 9900 ERROR = .TRUE.
 9999 RETURN
      END
```

3.36 Subroutine FIND

3.36.1 Purpose

The purpose of this subroutine is to determine polynomials based on attitude time history information.

3.36.2 Variable Listing

3.36.3 Subroutines Called:

Name	Function
MATINV	Calculates matrix inverse

3.36.4 Calling Subroutines:

Name	Function
AFORUN	Controls forward integration logic and calls integration routine
ATHREV	Controls thrust event logic during forward integration
ATILT	Controls tilt-over logic during forward integration

3.36.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a	Polynominal coefficients				Local	
b	Temporary matrix				Local	
chi	Array of dependent variables				Local	
dt	Array of independent variables				Local	
norder	Order of required polynomial				Local	

```
      SUBROUTINE FIND(CHI,A,DT,NORDER)
C
C      DETERMINES POLYNOMIALS BASED ON ATTITUDE TIME HISTORY INFORMATION
C          CHI    = ARRAY OF DEPENDENT VARIABLES
C          A      = POLYNOMIAL COEFFICIENTS
C          DT     = ARRAY OF INDEPENDENT VARIABLES
C          NORDER = ORDER OF REQUIRED POLYNOMIAL
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION B(19,19),CHI(9),DT(9),A(9)
C
      IF(NORDER.EQ.0) RETURN

      A(1)=CHI(1)

      DO 1 I=1,NORDER
      DO 1 J=1,NORDER
    1 B(I,J)=DT(I+1)**J

      CALL MATINV(B,NORDER)

      DO 2 I=1,NORDER
      A(I+1)=0.
      DO 2 J=1,NORDER
    2 A(I+1)=A(I+1)+B(I,J)*(CHI(J+1)-A(1))

      K=NORDER+1

      DO 3 I=1,K
    3 A(I)=A(I)/57.29578

      RETURN
      END
```

3.37 Function FUNISP

3.37.1 Purpose

This function calculates the specific impulse based on an input throttle value. The entry routine DEVISP is also contained within this subroutine. The entry routine calculates the partial derivative of the specific impulse with respect to mass.

3.37.2 Variable Listing

3.37.3 Subroutines Called:

None

3.37.4 Calling Subroutines:

Name	Function
ADER	Calculates the time independent portion of the equations of motion during forward integration
AINIT	Input routine
ATHREV	Controls thrust event logic during forward integration
BADLX	Calculates partial derivative of constraints with respect to control parameters
BDRI	Calculates equations of motion for the backward trajectory
GLIMT	Controls acceleration logic during forward integration
MASTRE	Controls total program logic
SUM15STG	Prints weight summary output for liquid vehicle
SUMARY	Prints weight summary output for Space Shuttle
SUMHLLV	Prints weight summary output for Strap-on vehicle

3.37.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
coisp(3,2)	Specific impulse coefficients		ainit	input	Global	cisp

```
        FUNCTION FUNISP(TAU,nstage)
C
C     THIS FUNCTION CALCULATES THE SSME ISP AS A FUNCTION OF POWER
C     LEVEL FROM A QUADRATIC EQUATION WHOSE COEFFICIENTS ARE
C     DETERMINED BY THE ROUTINE SIMUL CALLED IN AINIT.
C
C         ISP = C1*TAU**2 +C2*TAU + C3
C
C     WHERE     C1,C2,C3 ARE COEFFICIENTS
C               TAU IS SSME POWER LEVEL
C
C     THE ENTRY POINT DEVISP IS USED TO CALCULATE THE PARTIAL OF
C     ISP WITH RESPECT TO MASS BY USING THE CHAIN RULE.
C     THE PARTIAL OF ISP WRT POWER LEVEL IS CALCULATED IN THIS
C     ROUTINE AND THE PARTIAL OF POWER LEVEL WRT MASS IS INPUT
C     THRU COMMON.
C
C        @ISP/@TAU = 2.*C1*TAU + C2
C
C     THEN THE PARTIAL OF ISP WRT MASS IS CALCULATED BY
C
C        @ISP/@MASS = @ISP/@TAU * @TAU/@MASS
C
C
C     INPUTS :
C
C         TAU    -  SSME POWER LEVEL
C         COISP  -  COEFFICIENTS OF QUADRATIC EQUATION
C         PTAUM  -  @TAU/@MASS , PARTIAL OF POWER LEVEL WRT MASS
C
C     OUTPUT :
C
C        FUNISP  -  ISP FOR AN INPUT POWER LEVEL
C        DEVISP  -   @ISP/@MASS , PARTIAL OF ISP WRT MASS
C
C
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
        COMMON /CISP  / COISP(3,2)
C
        istg=1
        if(nstage.ge.2) istg=2
        FUNISP=(COISP(1,istg)*TAU+COISP(2,istg))*TAU+COISP(3,istg)
        RETURN
C
        ENTRY DEVISP(TAU,PTAUM,nstage)
C
        istg=1
        if(nstage.ge.2) istg=2
        PISPTA = 2.*COISP(1,istg)*TAU + COISP(2,istg)
        DEVISP = PISPTA*PTAUM
C
        RETURN
        END
```

3.38 Subroutine GOUT

3.38.1 Purpose

This subroutine displays a character string at the terminal and writes it to the print file and run time log file, if they are active.

3.38.2 Variable Listing

3.38.3 Subroutines Called:

None

3.38.4 Calling Subroutines:

Name	Function
AREAIN	Controls user interface
COPLDF	Writes a list directed file based on user requirements
INITIAL	Initializes program input
LDSWI	Creates and writes list directed file
LLPRT	Controls user interface to print various output tables

3.38.5 Fortran Listing

gout

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
jout	Output file index		initial	input	Global	ccc
nc	Number of characters in string				Local	
string	Output string to display				Local	

```
C THIS SUBROUTINE DISPLAYS A CHARACTER STRING AT THE TERMINAL
C AND WRITES IT TO THE PRINT FILE AND RUN TIME LOG FILE IF THEY
C ARE ACTIVE
C
C ENTRIES: GOUT
C
C ARGUMENTS: STRING - STRING TO DISPLAY (CHARACTER)
C            NC - NUMBER OF CHARACTERS IN STRING (INTEGER)
C
      SUBROUTINE GOUT ( STRING,NC )
      LOGICAL DEMAND,GRAPH,TEK19IN
      CHARACTER*1 STRING(NC)

      common/ccc/graph,jout,demand

      IF ( NC .EQ. 0 ) RETURN
C
C --- DISPLAY STRING
      WRITE ( 6,10 ) STRING
C
C --- ECHO OUTPUT ON PRINT FILE OR RUN TIME LOG IF ACTIVATED
      IF ( JOUT .EQ. 31 ) WRITE ( JOUT,10 ) STRING
      RETURN
C
   10 FORMAT ( 1X,130A1 )
      END
```

3.39 Subroutine HOLDIT

3.39.1 Purpose

This subroutine creates a pause within the run stream.

3.39.2 Variable Listing

3.39.3 Subroutines Called:

None

3.39.4 Calling Subroutines:

Name	Function
LLPRT	Controls user interface to print various output tables

3.39.5 Fortran Listing

holdit

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
demand	Logical indicating the operational mode		initial	input	Global	ccc
dummy	Temporary variable				Local	
graph	Unused logical variable				Global	ccc

```
      SUBROUTINE HOLDIT
C
      common/ccc/graph,jout,demand
C
      LOGICAL DEMAND,GRAPH

      CHARACTER*4 DUMMY

      IF (DEMAND .AND. GRAPH) THEN
          WRITE(6,*)' '
          WRITE(6,*)'PRESS RETURN TO CONTINUE'
          READ(5,'(a4)') DUMMY
      ENDIF

      RETURN
      END
```

3.40 Subroutine IINRC

3.40.1 Purpose

This subroutine is called from subroutine AINIT to read integer values from the aerodynamic database.

3.40.2 Variable Listing

3.40.3 Subroutines Called:

None

3.40.4 Calling Subroutines:

Name	Function
AINIT	Input routine

3.40.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
iloc	Index of data base location				Local	
irec	Record number on direct access file				Local	
itmp	Temporary array for storage of data from database				Local	
ival	Output variable				Local	

```
      SUBROUTINE IINRC(IVAL,IREC,ILOC)
C
C         THIS SUBROUTINE IS CALLED FROM AINIT TO READ INTEGER VALUES
C     FROM THE DATA BASE(MASTREDB.DAT).
C
      DIMENSION ITMP(1500)
C
      READ(8'IREC,IOSTAT=IOS)ITMP
      IF(IOS.NE.0) THEN
              WRITE(6,*)' ERROR READING DB. IN IINRC, IREC='
     *                  ,IREC,'   IVAL=',IVAL
              RETURN
      END IF
      IVAL=ITMP(ILOC)
      RETURN
      END
```

3.41 Subroutine INITIAL

3.41.1 Purpose

The purpose of this subroutine is to initialize the program by requesting from the user information concerning whether the session is in a demand or interactive mode. If in a demand mode, additional questions are requested concerning print files and graphic terminal usage.

3.41.2 Variable Listing

3.41.3 Subroutines Called:

Name	Function
GOUT	Displays a character string at the terminal
IS_PROCESS_INTERACTIVE	Inquires whether the current process is in interactive or demand mode

3.41.4 Calling Subroutines:

Name	Function
MASTRE	Controls total program logic

3.41.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ans	Responce from interactive mode				Local	
charay	Print file name				Local	
demand	Logical indicating the operational mode		initial	output	Global	ccc
fname	Output file name				Local	
graph	Unused logical variable			output	Global	ccc
id	Temporary variable				Local	
jout	Output file index		initial	output	Global	ccc
jout	Output file index		initial	input	Global	ccc
pf	Print file flag				Local	
tk19in	Logical flag to indicate graphic terminal			output	Global	ccc

```
      SUBROUTINE INITIAL
C
C    STEVEN SMITH BCSS 2/24/87
C
      INTEGER JOUT
      CHARACTER*4 ANS,ID
      CHARACTER CHARAY*64,FNAME*20
      LOGICAL GRAPH,DEMAND,PF,TK19IN
      COMMON /CCC/GRAPH,JOUT,DEMAND
C
      CALL IS_PROCESS_INTERACTIVE( DEMAND )
C
      CALL SYS$TRNLOG('SYS$ERROR',LEGNTH,CHARAY,,,)
      IF( DEMAND ) THEN
C
C MUST BE DEMAND
C
C CHECK IF PRINT FILE IS REQUIRED
C
        WRITE(6,120)
        READ(5,110)ANS
 120    FORMAT(' DO YOU WANT A PRINT FILE')
        IF(ANS(1:1) .EQ. 'Y') THEN
         PF = .TRUE.
        ELSE
         PF = .FALSE.
        ENDIF     ! END PRINT FILE CHECK
C
C  CHECK IF YOU ARE AT A GRAPHICS TERMINAL
C
        WRITE(6,100)
        READ(5,110)ANS
 100    FORMAT(' ARE YOU AT A GRAPHICS TERMINAL')
        IF ( ANS(1:1) .EQ. 'Y' ) THEN
           GRAPH = .TRUE.
           TK19IN = .TRUE.
        ELSE
           GRAPH = .FALSE.
           TK19IN = .FALSE.
        END IF     !  END GRAPH CHECK
 110    FORMAT(A)
      ELSE      ! MUST BE A BATCH RUN
        PF = .FALSE.
        GRAPH = .FALSE.
      ENDIF     !END DEMAND CHECK
C
C  OPEN PRINT FILE IF NECESSARY
C
      IF (PF) THEN
         JOUT = 31
          ID(1:3) = CHARAY(13:15)
          FNAME = 'MASTREPF.'//ID
          OPEN (31,NAME='FOR031.DAT',TYPE='NEW',ERR=99)
        ENDIF
      RETURN
 99   CALL GOUT(' ERROR OPENING PRINT FILE',25)
```



```
      JOUT=6
      RETURN
      END
```

3.42 Subroutine INTER

3.42.1 Purpose

The purpose of this subroutine is perform linear interpolation of nonlinear aerodynamic data.

3.42.2 Variable Listing

3.42.3 Subroutines Called:

None

3.42.4 Calling Subroutines:

Name	Function
CAERO	Evaluates nonlinear aerodynamic data

3.42.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a	Aerodynamic coefficient storage array				Local	
ang	Independent parameter				Local	
c	Storage of aerodynamic coefficients				Local	
ia1	Index of 2nd element of c array				Local	
ia2	Index of 2nd element of c array				Local	
ic1	Minimum index on 1st element of c array				Local	
ic2	Maximum index on 1st element of c array				Local	
taba	Independent table				Local	
temp	Temporary variable				Local	
temp1	Temporary variable				Local	

```
      SUBROUTINE INTER(A,C,IA1,IA2,ANG,TABA,IC1,IC2)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION A(6,2),C(15,11,6),TABA(1)
      DO 2 I=1,6
      K=0
      DO 2 J=IC1,IC2
      K=K+1
      IF(IC1.EQ.IC2.AND.IC1.NE.1) K=K+1
      TEMP=C(J,IA1,I)
      TEMP1=C(J,IA2,I)
      IF(IA1.EQ.IA2) GO TO 1
      TEMP=TEMP+(TEMP1-TEMP)/(TABA(IA2)-TABA(IA1))*(ANG-TABA(IA1))
    1 A(I,K)=TEMP
    2 CONTINUE
      RETURN
      END
```

3.43 Subroutine IS_PROCESS_INTERACTIVE

3.43.1 Purpose

This subroutine inquires whether the current process is in interactive or demand mode.

3.43.2 Variable Listing

3.43.3 Subroutines Called:

None

3.43.4 Calling Subroutines:

Name	Function
INITIAL	Initializes program execution

3.43.5 Fortran Listing

is_process_interactive

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
answer	Flag denoting interactive status				Local	
buffer	Buffer index				Local	
eq_list	Temporary storage array				Local	
itemlist	Temporary storage array				Local	

```
C*******************************************************************************
C
      SUBROUTINE IS_PROCESS_INTERACTIVE( ANSWER )
C
C*******************************************************************************
C
C -- THIS ROUTINE INQUIRES WHETHER OR NOT THE CURRENT PROCESS IS AN
C    INTERACTIVE ONE, USING THE SYS$GETJPI SYSTEM SERVICE WITH
C    SYNCHRONOUS COMPLETION AND USING DEFAULT EVENT FLAG 0.
C
C -- AUTHOR: BCSS/S.M.SPILLERS    8/29/86
C
       IMPLICIT NONE
       INCLUDE '($JPIDEF)'
       INTEGER*2 ITEMLIST(8)
       INTEGER*4 EQ_LIST(4),STATUS
       EQUIVALENCE (ITEMLIST,EQ_LIST)
       INTEGER*4 BUFFER,SYS$GETJPIW
       EXTERNAL SYS$GETJPIW
       LOGICAL*4 ANSWER

       ITEMLIST(1)= 4                !LENGTH OF RETURN BUFFER IN BYTES
       ITEMLIST(2)= JPI$_MODE        !FUNCTION CODE
       EQ_LIST(2) = %LOC(BUFFER)     !POINTER TO RETURN BUFFER
       EQ_LIST(3) = 0                !CLEAR RETURN LENGTH POINTER
       EQ_LIST(4) = 0                !NULL WORD TO TERMINATE ITEMLIST

       STATUS=SYS$GETJPIW(,,,ITEMLIST,,,)      !SYNCHRONOUS SYSTEM SERVICE REQUES

       IF( BUFFER.EQ.JPI$K_INTERACTIVE ) THEN
           ANSWER= .TRUE.
       ELSE
           ANSWER= .FALSE.
       END IF

       RETURN
       END
```

3.44 Subroutine LDSWI

3.44.1 Purpose

This subroutine creates and writes a direct access file (used in interactive mode).

3.44.2 Variable Listing

3.44.3 Subroutines Called:

Name	Function
GOUT	Prints character strings to the terminal

3.44.4 Calling Subroutines:

Name	Function
COPLDF	Writes a list directed file based on user requirements

3.44.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
contrl	Index number of direct access file				Local	
ctemp	Temporary storage array				Local	
idcttm	Record length divided by 2				Local	
ie	Index to determine the number of times that the file has been read				Local	
ierr	Error index				Local	
irecl	Record length				Local	
itemp	Temporary storage array				Local	
merr	File reading error flag				Local	
nr	Record number				Local	
nwr	Number of records on input file				Local	

```
C234567
      SUBROUTINE LDSWI(FILE,FILDES,CONTRL,NWR,*)
C
C
C-----------------------------------------------------------------
C
C  THIS ROUTINE WAS WRITTEN TO REPLACE THE MIPS LDSWI SUBROUTINE
C
C-----------------------------------------------------------------
C
C
      LOGICAL GRAPH,DEMAND,TK19IN
C
      INTEGER CONTRL
C
      CHARACTER*8 CTEMP(80)
      CHARACTER*12 FILE
      CHARACTER*36 FILDES
C
      COMMON/CCC/GRAPH,JOUT,DEMAND
C
      DIMENSION ITEMP(160)
C
      EQUIVALENCE (ITEMP(1),CTEMP(1))
C
      MERR = 0
      IE = 0
   10 CALL GOUT(' SPECIFY FILE NAME FOR WRITING THE FOLLOWING',44)
      CALL GOUT(FILDES,36)
C
      READ(5,17,ERR=10) FILE
   17 FORMAT(A12)
C
      CONTRL = 25
      OPEN(UNIT=CONTRL,NAME=FILE,TYPE='UNKNOWN',ACCESS='DIRECT',
     *     RECORDSIZE=NWR,ERR=15,IOSTAT=IERR)      !FROM DBDFD
C
      IF(IOSTAT.GT.0) THEN
          WRITE(6,5) FILE
    5     FORMAT(1X,'ERROR OPENING ',A12)
          CLOSE(UNIT=CONTRL)
          STOP
      ENDIF
C
C     WRITE DICTIONARY FROM [MASTRE.DATA]DICTION.DAT TO RECORDS
C     10 AND 11 OF OUTPUT FILE
C
      CLOSE(UNIT=12)
      OPEN(UNIT=12,FILE='[MASTRE.DATA]DICTION.DAT',STATUS='OLD',
     *     ACCESS='DIRECT')
      INQUIRE(UNIT=12,RECL=IRECL)
      WRITE(6,6) FILE,IRECL
    6 FORMAT(1X,A12,'OPENED , IRECL = ',I4)
      NR=10
      READ(12'NR) (ITEMP(I),I=1,IRECL)
      IDCTTM=IRECL/2
```



```
      WRITE(6,4100) (I,CTEMP(I),I=1,IDCTTM)
 4100 FORMAT(10(I4,1X,A8))
      WRITE(25'NR) (ITEMP(I),I=1,IRECL)
      NR=11
      READ(12'NR) (ITEMP(I),I=1,IRECL)
      WRITE(6,4100) (J+IDCTTM,CTEMP(J),J=1,IDCTTM)
      WRITE(25'NR) (ITEMP(I),I=1,IRECL)
      CLOSE(UNIT=12)
      WRITE(6,*) ' DICTIONARY WRITTEN IN RECORDS 10 AND 11 OF ',FILE
C
      RETURN
C
   15 IE = IE + 1
      IF(IE.LT.4) GO TO 10
      IF(MERR.EQ.0) MERR = 45
      RETURN 1
C
      END
```

3.45 Subroutine LDWRIT

3.45.1 Purpose

This generic routine writes to a specified direct access file.

3.45.2 Variable Listing

3.45.3 Subroutines Called:

None

3.45.4 Calling Subroutines:

Name	Function
COPLDF	Writes a list directed file based on user requirements

3.45.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
contrl	Internal file number				Local	
next	Record number				Local	
rec	Internal file number				Local	

```
      SUBROUTINE LDWRIT(FILNAM,IDUM,CONTRL,NEXT,REC,*)
      CHARACTER*12 FILNAM
      INTEGER CONTRL
      DIMENSION REC(160)
      WRITE(CONTRL'NEXT,ERR=1300)REC
      NEXT=NEXT+1
      RETURN
 1300 PRINT *,'ERROR IN LDWRIT'
      RETURN
      END
```

3.46 Subroutine LLPRT

3.46.1 Purpose

This subroutine controls user interface to print constraints and optimized updated values, print control variable update, print parameter summary table, and print load indicator table.

3.46.2 Variable Listing

3.46.3 Subroutines Called:

Name	Function
GOUT	Prints character strings to the terminal
HOLDIT	Pause routine

3.46.4 Calling Subroutines:

Name	Function
MENU	Controls calls to different program modes

3.46.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
betcon	Logical value to indicate parameter convergence				Local	
block	Temporary storage array				Local	
case	Case number				Local	
head	Storage array for headings in printout				Local	
header	Array defining constraint titles				Local	
iback	File control index				Local	
icount	File control index				Local	
ihit10	File control index				Local	
ihit11	File control index				Local	
ihit12	File control index				Local	
ihit13	File control index				Local	
iunit	Output unit names				Local	
iwant	Index				Local	
ja	Index				Local	
jj	Index				Local	
kat	Indicator for positive definite values of huu matrix				Local	
kp	Index				Local	
l3	File control index				Local	
llflag	File control index			output	Global	llflag
name	Character string array				Local	
phite	Status of constraints	variabl			Local	
save	Temporary storage array				Local	
svt	Temporary variable				Local	
units	File control index			output	Global	llflag

```
C*******************************************************************
C
      SUBROUTINE LLPRT
C
C*******************************************************************
C
C234567890123456789012345678901234567890123456789012345678901234567890123456789012
C        1         2         3         4         5         6         7
C
C*******************************************************************
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      CHARACTER*12 NAME(9),IUNIT(9)
      character*12 header(20)
      CHARACTER*60 HEAD
      character*80 out
C
      COMMON/LLFLAG/LLFLAG
C
      DIMENSION BLOCK(22),PHITE(20),SAVE(8,9)
C
C*******************************************************************
C
      IF(LLFLAG.EQ.0)GO TO 11
      LLFLAG=0
C-------------------WRITE END ON 10,11,12,13--------------------------
      END FILE 10
      END FILE 11
      END FILE 12
      END FILE 13

11    IHIT10=0
      IHIT11=0
12    CONTINUE
      IHIT12=0
      IHIT13=0
C-------------------------------------------------------------------
1     IWANT=0

      REWIND 14
      REWIND 2
      REWIND  10
      REWIND 11
      REWIND 12
      REWIND 13
C
      CALL PAGIT
C
      CALL GOUT(' 0 - TERMINATE TO MAIN MENU',27)
      CALL GOUT(' 1 - PRINT CONSTRAINTS AND OPTIMIZED UPDATED VALUES',51
     *)
      CALL GOUT(' 2 - PRINT CONTROL VARIABLE UPDATE',34)
      CALL GOUT(' 3 - PRINT PARAMETER SUMMARY TABLE',34)
      CALL GOUT(' 4 - PRINT LOAD INDICATOR TABLE',31)
C
```



```
      WRITE(6,2)
2     FORMAT(1X,'CHOICE> ',$)
      READ(5,3,ERR=1) IWANT
3     FORMAT(I1)
C
      IF(IWANT.EQ.0) THEN
         LLFLAG=0
         RETURN
      ENDIF

      IF(IWANT.GT.4) GO TO 1
      IF(IWANT.GT.2)GO TO 6000

      ICOUNT=0

      CALL PAGIT

50    CONTINUE

      IF(IHIT10.EQ.0) THEN

         READ(10,ERR=500,END=500) JA,K,CASE,HEADER,PHITE

      ELSE

500      IHIT10=10
         GO TO 2000

      ENDIF

51    CONTINUE

      write(out,100) CASE

100   FORMAT(14H  MASTRE CASE=,F11.4)

      CALL GOUT(OUT,25)

      ICOUNT=ICOUNT+1

      IF(IWANT.NE.1)GO TO 3000

C---------------------TEST  K-------------------------------------

      IF(K.le.5) then

         write(out,101) (HEADER(I),I=1,K)

101      FORMAT(7X,5a12)

         KP=K*12+7

         CALL GOUT(OUT,KP)

         write(out,102) (PHITE(I),I=1,K)
```

```
102       FORMAT(7X,5E12.5)
          CALL GOUT(OUT,KP)
          ICOUNT=ICOUNT+2
       else
          write(out,101) (HEADER(I),I=1,5)
          CALL GOUT(OUT,67)
          write(out,102) (PHITE(I),I=1,5)
          CALL GOUT(OUT,67)
          write(out,101) (HEADER(I),I=6,K)
          KP=(K-5)*12+7
          CALL GOUT(OUT,KP)
          write(out,102) (PHITE(I),I=6,K)
          CALL GOUT(OUT,KP)
          ICOUNT=ICOUNT+4
       endif
2000   continue
C-------------------------------------READ 11-------------------------
198    IF(IHIT11.EQ.0) then
          READ(11,ERR=1988,END=1988) JA,K,BLOCK(3)
          go to 199
1988      IHIT11=11
199       IF(IHIT11.EQ.0) then
             IF(JA.ne.8) then
                IF(JA.EQ.1) write(out,201) K,BLOCK(3)
201             FORMAT(2X,'TAUT(',I2,')=',E15.8)
                IF(JA.EQ.2) write(out,202)BLOCK(3)
202             FORMAT(7X,'WO1=',E15.8)
                IF(JA.EQ.3) write(out,203) BLOCK(3)
203             FORMAT(8X,'AZ=',E15.8)
                IF(JA.EQ.4) write(out,204) K,BLOCK(3)
204             FORMAT(1X,'CPTBL(',I2,')=',E15.8)
```



```
                IF(JA.EQ.5) write(out,205) K,BLOCK(3)
205             FORMAT(1X,'CYTBL(',I2,')=',E15.8)
                IF(JA.EQ.6) write(out,207) K,BLOCK(3)
207             FORMAT(1X,'T PAR(',I2,')=',E15.8)
                IF(JA.EQ.7) write(out,208) K,BLOCK(3)
208             FORMAT(1X,'B PAR(',I2,')=',E15.8)
                CALL GOUT(OUT,26)
                ICOUNT=ICOUNT+1
                GO TO 198
             endif
             IF(ICOUNT.LE.26)GO TO 3000
             CALL HOLDIT
             ICOUNT=0
             CALL PAGIT
          endif
       endif
C-------------------------------------READ 12-------------------------
3000   CONTINUE
       IF(IWANT.NE.2)GO TO 401
       IF(IBACK.EQ.1234)BACKSPACE 12
       IBACK=1234
       CALL GOUT('    TIME     NEW CP    OLD CP   NEW CY   OLD CY   DCP
      *   DCY   ',64)
       ICOUNT=ICOUNT+1
       IF(IHIT12.EQ.12)GO TO 4000
300    IF(IHIT12.EQ.0) READ(12,ERR=30001,END=30001)(BLOCK(I),I=1,7),JJ
       IF(IHIT12.EQ.0)GO TO 3001
30001  IHIT12=12
       GO TO 4000
3001   CONTINUE
       IF((SVT-.1) .GT. BLOCK(1))GO TO 4000
       SVT=BLOCK(1)
```

```
      write(out,301) (BLOCK(I),I=1,7),JJ
301   FORMAT(1X,F9.1,1X,F8.2,1X,F8.2,1X,F8.3,1X,F8.3,1X,F8.3,1X,F8.3,1X,
     *I3)

      CALL GOUT(OUT,67)

      ICOUNT=ICOUNT+1
      IF(ICOUNT.LT.34)GO TO 300

      CALL HOLDIT

      ICOUNT=0

      CALL PAGIT

      CALL GOUT('    TIME     NEW CP    OLD CP   NEW CY   OLD CY   DCP
     *   DCY   ',64)

      GO TO 300
C---------------------READ 13---------------------------------
4000  SVT=BLOCK(1)

      IF(IHIT13.EQ.0)READ(13,ERR=4010,END=4010)(BLOCK(I),I=1,4)

      IF(IHIT13.EQ.0)GO TO 4001

4010  IHIT13=13

      GO TO 5000

4001  CONTINUE

      write(out,400) (BLOCK(I),I=1,4)

400   FORMAT(1X,13HDEL CHIP MAX=,E11.4,14H DEL CHIY MAX=,E11.4,
     *8H BETCON=,L3,6H  KAT=,I3)

      CALL GOUT(OUT,70)

      CALL HOLDIT

      ICOUNT=0

401   IF(IWANT.EQ.1 .AND. (IHIT10+IHIT11).NE.21)GO TO 50

      IF(IWANT.EQ.2)GO TO 50

5000  CONTINUE

      CALL HOLDIT

      CALL PAGIT

      GO TO 11
```



```
C-----------------------READ 14-------------READ 14-----------------------
C-------------------------PARAMETER SUMMARY TABLE----------------------

6000  CONTINUE

      IF(IWANT.GT.3)GO TO 7000

      CALL PAGIT

      CALL GOUT(' PARAMETER SUMMARY TABLE',24)

      CALL GOUT(' ',1)

      BLOCK(1)='      '

      READ(14) HEAD,(BLOCK(I),I=1,21)

      write(out,6001) HEAD

6001  FORMAT(6X,A60)

      CALL GOUT(OUT,66)

      write(out,6002) (BLOCK(I),I=1,3)

 6002 FORMAT(5X,4HTIME,E15.8,7H  RANGE,E15.8,12H RANGE ANGLE,E15.7)

      CALL GOUT(OUT,72)

      write(out,6003) (BLOCK(I),I=4,6)

6003  FORMAT(9H INERTIALE15.8,7H RADIUSE15.8,12H FLIGHT PATHE14.7)

      CALL GOUT(OUT,72)

      write(out,6004)

 6004 FORMAT(1X,8HVELOCITY,41X,5HANGLE)

      CALL GOUT(OUT,55)

      write(out,6005) (BLOCK(I),I=7,9)

6005  FORMAT(9H     INCLE15.8,7H   DES.E15.8,12H       FLIGHTE14.7)

      CALL GOUT(OUT,72)

      write(out,6006)

6006  FORMAT(27X,4HNODE,19X,7HAZIMUTH)

      CALL GOUT(OUT,57)

      write(out,6007) (BLOCK(I),I=10,12)

6007  FORMAT(9H    GDLATE15.8,7H  GCLATE15.8,12H         LONGE14.7)
```

```
      CALL GOUT(OUT,72)
      write(out,6008) (BLOCK(I),I=13,14)
6008  FORMAT(9H INERTIALE15.8,7H     C1E15.8)
      CALL GOUT(OUT,46)
      CALL GOUT('  AZIMUTH',9)
      write(out,6009) (BLOCK(I),I=15,16)
6009  FORMAT(9H   WEIGHTE15.8,7H     C3E15.8)
      CALL GOUT(OUT,46)
      CALL GOUT(' KEPLERIAN ORBITAL PARAMETERS',29)
      write(out,6010) (BLOCK(I),I=17,19)
6010  FORMAT(9H ECNTRICYE15.8,7H APOGEEE15.8,12H     PERIGEEE14.7)
      CALL GOUT(OUT,72)
      write(out,6011)
6011  FORMAT(25X,6HRADIUS,20X,6HRADIUS)
      CALL GOUT(OUT,57)
      write(out,6012) (BLOCK(I),I=20,21)
6012  FORMAT(9H   APOGEEE15.8,7HPERIGEEE15.8)
      CALL GOUT(OUT,46)
      CALL GOUT('   HEIGHT                HEIGHT',31)
      CALL HOLDIT
C-------------------------------------------------------------------------
      GO TO 11
C-----------------------------------READ 2----------READ 2----------
7000  CONTINUE
      CALL PAGIT
      CALL GOUT(' LOAD INDICATOR TABLE',21)
      CALL GOUT(' ',1)
      READ(2) HEAD,NAME,SAVE,IUNIT
      write(out,6001) HEAD
```



```
      CALL GOUT(OUT,66)
      CALL GOUT('                                    FIRST STAGE',44)
      CALL GOUT('                            MINIMUM           MAXIMUM',51)
      CALL GOUT(' LOAD INDICATOR     TIME     VALUE    TIME     VALUE
     *    UNITS',64)
      DO 7001 I=1,9
      write(out,7002) NAME(I),(SAVE(K,I),K=1,4),IUNIT(I)
7001  CALL GOUT(OUT,66)
7002  FORMAT(2X,a12,4X,2(F6.2,2X,F8.2,2X),a12)
      CALL GOUT(' ',1)
      CALL GOUT('                                    SECOND STAGE',45)
      DO 7003 I=1,9
      write(out,7002) NAME(I),(SAVE(K,I),K=5,8),IUNIT(I)
7003  CALL GOUT(OUT,66)
      CALL HOLDIT
C-------------------------------------------------------------------------
      GO TO 11
      END
```

3.47 Subroutine MASSPRO

3.47.1 Purpose

This subroutine is an alternate method of calculating mass properties. The routine, based on predetermined data, calculates the center of gravity components in a dynamic fashion based on the current mass overboard and time changes.

3.47.2 Variable Listing

3.47.3 Subroutines Called:

Name	Function
SPLINE	3rd order spline interpolation model

3.47.4 Calling Subroutines:

Name	Function
ADER1	Calculates the time dependent portion of the equations of motion during forward integration
AINIT	Input routine
BDR1I	Calculates components of the equations of motion for the backward trajectory

3.47.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a	Spline Coefficient for interpolation				Local	
b	Spline coefficient for interpolation				Local	
boblat	Booster lateral component of centerof gravity	ft			Local	
boblatp	Derivative of booster lateral component of CG with respect to mass	ft/lbs			Local	
boblon	Booster longitudinal component of center of gravity	ft			Local	
boblonp	Derivative of booster longitudinal component of CG with respect to mass	ft/lbs			Local	
bobmass	Booster mass overboard	lbs			Local	
bobnor	Booster normal component of centerof gravity	ft			Local	
bobnorp	Derivative of booster normal component of CG with respect to mass	ft/lbs			Local	
bolaspl	Table for booster lateral CG	ft			Local	
bolatcg	Booster lateral CG input table from mass properties program	ft			Local	
boloncg	Booster longitudinal CG input table from mass properties program	ft			Local	
bolospl	Table for booster longitudinal CG	ft			Local	
bomaspl	Independent mass table for booster	lbs			Local	
bomass	Booster mass overboard table	lbs			Local	
bomdot	Booster mass flowrate	lbs/sec			Local	
bomov	Booster mass avialable table	lbs			Local	
bonorcg	Booster normal CG input table from mass properties program	ft			Local	
bonospl	Table for booster normal CG	ft			Local	
c	Spline coefficient for interpolation				Local	
calcpar	Logical flag to indicate calculation of partials				Local	
eventnum	Number of event				Local	
iplatcg	Lateral component of CG for inerts	ft			Local	
iploncg	Longitudinal component of CG for inerts	ft			Local	
ipmass	Inert mass table	lbs			Local	
ipnorcg	Normal component of CG for inerts	ft			Local	
ipstage	Stage inert mass	lbs			Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
last	Previous index for spline interpolation				Local	
latcg	Lateral center of gravity component	m			Local	
latcgp	Partial derivative of lateral CG with respect to mass				Local	
latcgpb	Partial derivative of lateral CG with respect to booster mass				Local	
latcgpm	Partial derivative of lateral CG with respect to main engine mass				Local	
loncg	Total longitudinal center of gravity	m			Local	
loncgp	Partial derivative of total longitudinal CG with respect to mass				Local	
loncgpb	Partial derivative of total longitudinal CG with respect to booster mass				Local	
loncgpm	Partial derivative of total longitudinal CG with respect to main engine mass				Local	
m	Mission designate (equivalent to nb)				Local	
mnblat	Main engine lateral component of center of gravity	in			Local	
mnblatp	Derivative main engine lateral CG with respect to mass	in/lbs			Local	
mnblon	Main engine longitudinal component of center of gravity	in			Local	
mnblonp	Derivative main engine longitudinal CG with respect to mass	in/lbs			Local	
mnbmass	Main engine mass overboard	lbs			Local	
mnbnor	Main engine normal component of center of gravity	in			Local	
mnbnorp	Derivative main engine normal CG with respect to mass	in/lbs			Local	
mnlaspl	Main engine lateral CG table	in			Local	
mnlatcg	Main engine lateral CG input table	in			Local	
mnloncg	Main engine longitudinal CG input table	in			Local	
mnlospl	Main engine longitudinal CG table	in			Local	
mnmaspl	Main engine mass overboard table	in			Local	
mnmass	Main engine mass overboard input table	lbs			Local	
mnmdot	Main engine flowrate	lbs/sec			Local	
mnmov	Main engine mass available	lbs			Local	
mnnorcg	Main engine normal CG input table	in			Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
mnorspl	Main engine normal CG table	in			Local	
moboos	Booster mass	lbs			Local	
momain	Main engine mass				Local	
mpname	Name of alternate mass properties input file			input	Global	mpropnm
norcg	Total normal component of center of gravity	in			Local	
norcgp	Derivative of normal CG with respect to mass	in/lbs			Local	
norcgpb	Derivative of normal CG with respect to booster mass				Local	
norcgpm	Derivative of normal CG with respect to main engine mass				Local	
ntime	Index to indicate first time		ainit	input	Global	

```
      SUBROUTINE MASSPRO(EVENTNUM,MOBOOS,MOMAIN,BOMDOT,MNMDOT,LONCG,
     &              LATCG,NORCG,CALCPAR,LONCGP,LATCGP,NORCGP)

      IMPLICIT NONE

*************************************************************************
*     This subroutine calculates the mass properties of the vehicle     *
*     dynamically as the MASTRE simulation is run.                      *
*                                                                       *
*     Will Borchers, Dynetics, Inc, August, 1992.                       *
*************************************************************************

      INTEGER EVENTNUM,L,M,LAST,I,NTIME

      REAL*8 IPLONCG(15),IPLATCG(15),IPNORCG(15),IPMASS(15),IPSTAGE(15)
      REAL*8 BOLONCG(30),BOLATCG(30),BONORCG(30),BOMOV(30),BOMASS(30)
      REAL*8 MNLONCG(30),MNLATCG(30),MNNORCG(30),MNMOV(30),MNMASS(30)

      REAL*8 MOBOOS,MOMAIN,BOMDOT,MNMDOT
      REAL*8 LONCG,LATCG,NORCG,LONCGP,LATCGP,NORCGP

      REAL*8 BOBLAT,BOBLON,BOBNOR,BOBMASS,BOBLONP,BOBLATP,BOBNORP
      REAL*8 BOBMASSP
      REAL*8 MNBLAT,MNBLON,MNBNOR,MNBMASS,MNBLONP,MNBLATP,MNBNORP
      REAL*8 MNBMASSP
      REAL*8 LONCGPB,LATCGPB,NORCGPB,LONCGPM,LATCGPM,NORCGPM,A,B,C
      REAL*8 BOMASPL(30),BOLOSPL(30,1),BOLASPL(30,1),BONOSPL(30,1)
      REAL*8 MNMASPL(30),MNLOSPL(30,1),MNLASPL(30,1),MNORSPL(30,1)

      LOGICAL MPFLAG,CALCPAR

      CHARACTER MPNAME*30

      COMMON/MPROPIN/ MPFLAG
      COMMON/MPROPNM/ MPNAME

      COMMON/NTIME/NTIME
*
*...Read in data if initial time through.
*
      IF(NTIME.EQ.1) THEN
         OPEN(UNIT=40,FILE=MPNAME,STATUS='OLD',RECL=132)
         READ(40,*)
         READ(40,*)
         DO I = 1 , 15
            READ(40,*) IPLONCG(I),IPLATCG(I),IPNORCG(I),IPMASS(I),
     &                 IPSTAGE(I)
         ENDDO
         READ(40,*)
         READ(40,*)
         READ(40,*)
         DO I = 1 , 30
            READ(40,*,ERR=205)  BOLONCG(I),BOLATCG(I),BONORCG(I),
     &                          BOMOV(I),BOMASS(I)
         ENDDO
         READ(40,*)
```



```
         READ(40,*)
205      READ(40,*)
         DO I = 1 , 30
            READ(40,*) MNLONCG(I),MNLATCG(I),MNNORCG(I),MNMOV(I),
     &                 MNMASS(I)
         ENDDO
*
*...Turn mass, longitude, latitude, and normal arrays around for SPLINE
*
         DO I = 1, 30
            BOMASPL(I)=BOMASS(31-I)
            BOLOSPL(I,1)=BOLONCG(31-I)
            BOLASPL(I,1)=BOLATCG(31-I)
            BONOSPL(I,1)=BONORCG(31-I)
            MNMASPL(I)=MNMASS(31-I)
            MNLOSPL(I,1)=MNLONCG(31-I)
            MNLASPL(I,1)=MNLATCG(31-I)
            MNORSPL(I,1)=MNNORCG(31-I)
         ENDDO

         RETURN

      ENDIF
*
*...Error checking.
*
      IF(IPMASS(EVENTNUM).LT.0.0) THEN
         WRITE(*,*) 'Inert mass exceeded in MASSPRO'
         WRITE(*,*) 'Eventnum = ',EVENTNUM
         STOP
      ELSE IF(MOBOOS.LT.0.0) THEN
         WRITE(*,*) 'Booster mass overboard less than zero '
         WRITE(*,*) 'in Subroutine MASSPRO.'
         STOP
      ELSE IF(MOBOOS.GT.BOMOV(30)) THEN
         WRITE(*,*) 'Booster mass exceeded in MASSPRO, last'
         WRITE(*,*) 'tabular value will be used for cg calculation'
         WRITE(*,*) 'Booster mass used =',MOBOOS
         WRITE(*,*) 'Mass available = ',BOMOV(30)
         MOBOOS=BOMOV(30)
      ELSE IF(MOMAIN.LT.0.0) THEN
         WRITE(*,*) 'Main engine mass overboard less than'
         WRITE(*,*) 'zero in Subroutine MASSPRO.'
         STOP
      ELSE IF(MOMAIN.GT.MNMOV(30)) THEN
          WRITE(*,*) 'Main engine mass exceeded in MASSPRO, last'
          WRITE(*,*) 'tabular value will be used for cg calculation'
          WRITE(*,*) 'Main engine mass used =',MOMAIN
          WRITE(*,*) 'Mass available = ',MNMOV(30)
          MOMAIN=MNMOV(30)
      ENDIF
*
*...Compute booster cgs.
*
      IF(BOMASS(1).EQ.0.0) GOTO 500
      L=1
```

```
      M=1
      LAST=32
      CALL SPLINE(L,M,30,MOBOOS,BOMOV,BOBMASS,BOMASS,BOBMASSP,2,
     &            LAST,A,B,C)
      L=1
      M=1
      LAST=32
      CALL SPLINE(L,M,30,BOBMASS,BOMASPL,BOBLON,BOLOSPL,BOBLONP,2,
     &            LAST,A,B,C)
      L=1
      M=1
      LAST=32
      CALL SPLINE(L,M,30,BOBMASS,BOMASPL,BOBLAT,BOLASPL,BOBLATP,2,
     &            LAST,A,B,C)
      L=1
      M=1
      LAST=32
      CALL SPLINE(L,M,30,BOBMASS,BOMASPL,BOBNOR,BONOSPL,BOBNORP,2,
     &            LAST,A,B,C)
*
*...Compute main engine cgs.
*
500   L=1
      M=1
      LAST=32
      CALL SPLINE(L,M,30,MOMAIN,MNMOV,MNBMASS,MNMASS,MNBMASSP,2,
     &            LAST,A,B,C)
      L=1
      M=1
      LAST=32
      CALL SPLINE(L,M,30,MNBMASS,MNMASPL,MNBLON,MNLOSPL,MNBLONP,2,
     &            LAST,A,B,C)
      L=1
      M=1
      LAST=32
      CALL SPLINE(L,M,30,MNBMASS,MNMASPL,MNBLAT,MNLASPL,MNBLATP,2,
     &            LAST,A,B,C)
      L=1
      M=1
      LAST=32
      CALL SPLINE(L,M,30,MNBMASS,MNMASPL,MNBNOR,MNORSPL,MNBNORP,2,
     &            LAST,A,B,C)
*
*...Combine for total cg location
*
      LONCG=(MNBLON*MNBMASS+BOBLON*BOBMASS+IPLONCG(EVENTNUM)*
     &      IPMASS(EVENTNUM))/(MNBMASS+BOBMASS+IPMASS(EVENTNUM))
      LATCG=(MNBLAT*MNBMASS+BOBLAT*BOBMASS+IPLATCG(EVENTNUM)*
     &      IPMASS(EVENTNUM))/(MNBMASS+BOBMASS+IPMASS(EVENTNUM))
      NORCG=(MNBNOR*MNBMASS+BOBNOR*BOBMASS+IPNORCG(EVENTNUM)*
     &      IPMASS(EVENTNUM))/(MNBMASS+BOBMASS+IPMASS(EVENTNUM))
*
*...If called from ADER1 or 1st call in BDR1I do not calculate partials
*
      IF(.NOT.CALCPAR) THEN
         RETURN
```



```
      ENDIF
*
*...Compute the partial of total longitudinal cg location with respect
*...to the booster mass and main engine mass.
*
      LONCGPB = ((BOBMASS*BOBLONP+BOBLON)*(BOBMASS+MNBMASS+
     &          IPMASS(EVENTNUM))-(BOBMASS*BOBLON+MNBMASS*MNBLON+
     &          IPLONCG(EVENTNUM)*IPMASS(EVENTNUM)))/
     &          (BOBMASS+MNBMASS+IPMASS(EVENTNUM))**2
      LONCGPM = ((MNBMASS*MNBLONP+MNBLON)*(BOBMASS+MNBMASS+
     &          IPMASS(EVENTNUM))-(BOBMASS*BOBLON+MNBMASS*MNBLON+
     &          IPLONCG(EVENTNUM)*IPMASS(EVENTNUM)))/
     &          (BOBMASS+MNBMASS+IPMASS(EVENTNUM))**2
*
*...Compute the partial of total longitudinal cg location with respect
*...to the booster mass and main engine mass.
*
      LATCGPB = ((BOBMASS*BOBLATP+BOBLAT)*(BOBMASS+MNBMASS+
     &          IPMASS(EVENTNUM))-(BOBMASS*BOBLAT+MNBMASS*MNBLAT+
     &          IPLATCG(EVENTNUM)*IPMASS(EVENTNUM)))/
     &          (BOBMASS+MNBMASS+IPMASS(EVENTNUM))**2
      LATCGPM = ((MNBMASS*MNBLATP+MNBLAT)*(BOBMASS+MNBMASS+
     &          IPMASS(EVENTNUM))-(BOBMASS*BOBLAT+MNBMASS*MNBLAT+
     &          IPLATCG(EVENTNUM)*IPMASS(EVENTNUM)))/
     &          (BOBMASS+MNBMASS+IPMASS(EVENTNUM))**2
*
*...Compute the partial of total longitudinal cg location with respect
*...to the booster mass and main engine mass.
*
      NORCGPB = ((BOBMASS*BOBNORP+BOBNOR)*(BOBMASS+MNBMASS+
     &          IPMASS(EVENTNUM))-(BOBMASS*BOBNOR+MNBMASS*MNBNOR+
     &          IPNORCG(EVENTNUM)*IPMASS(EVENTNUM)))/
     &          (BOBMASS+MNBMASS+IPMASS(EVENTNUM))**2
      NORCGPM = ((MNBMASS*MNBNORP+MNBNOR)*(BOBMASS+MNBMASS+
     &          IPMASS(EVENTNUM))-(BOBMASS*BOBNOR+MNBMASS*MNBNOR+
     &          IPNORCG(EVENTNUM)*IPMASS(EVENTNUM)))/
     &          (BOBMASS+MNBMASS+IPMASS(EVENTNUM))**2
*
*...Compute partial of cg with respect to mass
*
      LONCGP=(LONCGPB*BOMDOT+LONCGPM*MNMDOT)/(BOMDOT+MNMDOT)
      LATCGP=(LATCGPB*BOMDOT+LATCGPM*MNMDOT)/(BOMDOT+MNMDOT)
      NORCGP=(NORCGPB*BOMDOT+NORCGPM*MNMDOT)/(BOMDOT+MNMDOT)
*
*...Return
*

      RETURN
      END
```

3.48 Subroutine MATINV

3.48.1 Purpose

This subroutine calculates the inversion of an nxn matrix.

3.48.2 Variable Listing

3.48.3 Subroutines Called:

None

3.48.4 Calling Subroutines:

Name	Function
ANEWCH	Evaluates the optimization and restoration steps for the optimization scheme
FIND	Determines attitude polynomials for 1st stage

3.48.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a	Temporary storage array				Local	
amax	Maximum element of matrix (equivalent to t and swap)				Local	
b	Input/output matrix				Local	
icolum	Column index (equivalent to jcolum)				Local	
ii	Index				Local	
index	Temporary storage index				Local	
ipivot	Pivot array index				Local	
irow	Row index (equivalent to jrow)				Local	
jcolum	Column index (equivalent to icolum)				Local	
jj	Index				Local	
jrow	Row index (equivalent to irow)				Local	
l1	Index				Local	
n	Number of columns or rows				Local	
pivot	Pivot array				Local	
swap	Maximum element of matrix (equivalent to t and amax)				Local	
t	Maximum element of matrix (equivalent to amax and swap)				Local	

```
      SUBROUTINE MATINV(B,N)
C*********************************************************************
C     MATRIX INVERSION
C          B      = INPUT,MATRIX TO BE INVERTED;OUTPUT,INVERTED MATRIX
C          N      = NUMBER OF ROWS
C*********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION A(19,19),B(19,19),PIVOT(19)
      DIMENSION IPIVOT(19),INDEX(19,2)
      EQUIVALENCE (IROW,JROW), (ICOLUM,JCOLUM), (AMAX, T, SWAP)          F4020007
      DO I=1,N
         DO J=1,N
            A(I,J)=B(I,J)
         enddo
      enddo
C*********************************************************************
C     INITIALIZATION
C*********************************************************************
      DO J=1,N
         IPIVOT(J)=0
      enddo
C*********************************************************************
C     SEARCH FOR PIVOT ELEMENT                                         F4020016
C*********************************************************************
      DO 550 I=1,N                                                     F4020014
      AMAX=0.0                                                         F4020018
      DO 105 J=1,N                                                     F4020019
      If(IPIVOT(J).ne.1) then
         DO 100 K=1,N
         if(ipivot(k).gt.1) then
            DO 80 ii=1,N
            DO 80 jj=1,N
   80       B(ii,jj)=A(ii,jj)
```



```
            RETURN
         else if(ipivot(k).lt.1) then
            IF(DABS(AMAX).lt.DABS(A(J,K))) then
               IROW=J
               ICOLUM=K
               AMAX=A(J,K)
            endif
         endif
  100    CONTINUE
      endif
  105 CONTINUE                                                         F4020028
      IPIVOT(ICOLUM)=IPIVOT(ICOLUM)+1                                  F4020029
C*********************************************************************
C     INTERCHANGE ROWS TO PUT PIVOT ELEMENT ON DIAGONAL                F4020031
C*********************************************************************
      if(irow.ne.icolum) then
         DO 200 L=1,N
         SWAP=A(IROW,L)
         A(IROW,L)=A(ICOLUM,L)
  200    A(ICOLUM,L)=SWAP
      endif
      INDEX(I,1)=IROW                                                  F4020044
      INDEX(I,2)=ICOLUM                                                F4020045
      PIVOT(I)=A(ICOLUM,ICOLUM)                                        F4020046
C*********************************************************************
C     DIVIDE PIVOT ROW BY PIVOT ELEMENT                                F4020049
C*********************************************************************
      A(ICOLUM,ICOLUM)=1.D0                                            F4020051
      DO 350 L=1,N                                                     F4020052
  350 A(ICOLUM,L)=A(ICOLUM,L)/PIVOT(I)                                 F4020053
C*********************************************************************
C     REDUCE NON-PIVOT ROWS                                            F4020058
```

```
C*********************************************************************
      DO 550 L1=1,N                                                  F4020060
      IF(L1.ne.ICOLUM) then
         T=A(L1,ICOLUM)
         A(L1,ICOLUM)=0.0
         DO 450 L=1,N
  450    A(L1,L)=A(L1,L)-A(ICOLUM,L)*T
      endif
  550 CONTINUE                                                       F4020069
C*********************************************************************
C     INTERCHANGE COLUMNS                                            F4020071
C*********************************************************************
      DO 710 I=1,N                                                   F4020073
      L=N+1-I                                                        F4020074
      IF (INDEX(L,1).ne.INDEX(L,2)) then
         JROW=INDEX(L,1)
         JCOLUM=INDEX(L,2)
         DO 705 K=1,N
         SWAP=A(K,JROW)
         A(K,JROW)=A(K,JCOLUM)
         A(K,JCOLUM)=SWAP
  705    CONTINUE
      endif
  710 CONTINUE                                                       F4020083
      DO 780 I=1,N                                                   F4020051
      DO 780 J=1,N
  780 B(I,J)=A(I,J)
      RETURN
      END
```

3.49 Subroutine MENU

3.49.1 Purpose

This subroutine provide an interface for the user if the user is in the interactive mode of operation. Based on user input, the subroutine controls the calls to the area mode, execute mode, print mode, and namelist input selection.

3.49.2 Variable Listing

3.49.3 Subroutines Called:

Name	Function
AINIT	Input routine
AREAIN	Controls user interface
LLPRT	Controls user interface printing

3.49.4 Calling Subroutines:

Name	Function
MASTRE	Controls total program logic

3.49.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
demand	Logical indicating the operational mode		initial	input	Global	ccc
itape	Logical flag to indicate tape output			output	Global	tol
iuse	Operational mode index			output	Global	iuse
ntime	Case number identification used in subroutine ainit to reduce input requirements for multiple cases			output input	Global	ntime

```
      SUBROUTINE MENU
C
      LOGICAL ITAPE,DEMAND,GRAPH
C
      COMMON/TOL/ITOL,ITAPE
      COMMON/CCC/GRAPH,JOUT,DEMAND
      COMMON/IUSE/IUSE
      COMMON/NTIME/NTIME
C*************************************************************************
      if(demand) then
C----------------------------AREA MODE-----------------------------------
1        ITAPE=.FALSE.
         CALL AREAIN
C-------------------------------------EXECUTE MODE---------------------
         IF(IUSE.EQ.1) then
C------------------ERROR IF TRY RUNNING WITHOUT INPUT-----------------
            IF(NTIME.NE.1) RETURN
            GO TO 1
         endif
C----------------------------PRINT MODE-----------------------------------
         IF(IUSE.EQ.4) CALL LLPRT
C---------------------------NAMELIST INPUT SELECTION MODE-------------
         IF(IUSE.NE.2) GO TO 1
         CALL AINIT
         GO TO 1
      else
C---------------------------EXECUTE BATCH/DEMAND---------------
         IUSE=1
         CALL AINIT
      endif
      RETURN
```



```
      END
```

3.50 Subroutine RRAINT

3.50.1 Purpose

Initialization routine for the Range Reference atmosphere.

3.50.2 Variable Listing

3.50.3 Subroutines Called:

Name	Function
RRRATM	Range Reference atmosphere model

3.50.4 Calling Subroutines:

Name	Function
AINIT	Input routine

3.50.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
atmos	Name of atmospheric subroutine being used for the simulation				Local	
atmosn	Name of atmospheric subroutine being used for the simulation				Local	
atmtbe	English unit output array of atmopsheric parameters				Local	
atmtbl(110,6	Storage matrix for atmospheric parameters			output input	Global	atmos
demand	Logical indicating the operational mode		initial	input	Global	ccc
dens	Atmospheric density	kg/m^3			Local	
dz1	First altitude increment	ft			Local	
dz2	Second altitude increment	ft			Local	
dz3 .	Third altitude increment	ft			Local	
icnt	Atmospheric parameter table index			output input	Global	icntc
itr	Atmospheric model index				Local	
iz1	Index				Local	
matm1	Index for spline interpolation of atmospheric parameters			output	Global	atmos
matm2	Previous index of atmospheric parameter spline interpolation			output	Global	atmos
maxpt	Maximum number of points in atmospheric table				Local	
mnth	Character variable representing the name of the month used for range reference atmosphere calculations			output	Global	atmos
month	Index for month				Local	
mthrra	Atmospheric month indicator		ainit	output input	Global	atmos
nlines	Index used for output page control				Local	
pres	Atmospheric pressure	slugs/f			Local	
rho	Atmospheric density	slugs/f			Local	
rmonth	Name of month used to define wind table used				Local	
sitemo	Site and month index			output	Global	rrrain
tk	Atmospheric temperature coefficient tables				Local	
tr	Name of launch site array				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
vsos	Speed of sound	m/sec			Local	
xmu	Viscosity coefficient				Local	
z	Altitude	m			Local	
z1	Upper limit of altitude regime for atmopshere calculations				Local	
z2	Upper limit of altitude regime for atmopshere calculations				Local	
z3	Upper limit of altitude regime for atmopshere calculations				Local	

```
      SUBROUTINE RRAINT(ATMOSN)
C
C     INITIALIZATION ROUTINE FOR RANGE REFERENCE ATMOSPHERE
C
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      LOGICAL DEMAND,GRAPH
C
      CHARACTER*1 ANS
      CHARACTER*3 TR(2),RMONTH(12),MNTH
      CHARACTER*6 ATMOS,ATMOSN
C
      COMMON /ATMOS / ATMTBL(110,6),MTHRRA,MATM1,MATM2,MNTH
      COMMON /CCC   / GRAPH,JO,DEMAND
      COMMON /ICNTC / ICNT
      COMMON /RRRAC / RRRAAT(52,7),FINLAT(70,4),ISITE,ITOP,LASTAL
      COMMON /RRRAIN/ INIT,PRNTTB,SITEMO,NCHG,ALTSIG(7,2)
C
      DIMENSION ATMTBE(110,6)
C
      DATA MAXPT,IZ1/110,0/
      DATA TR /'ETR','WTR'/
      DATA RMONTH /'JAN','FEB','MAR','APR','MAY','JUN',
     *             'JUL','AUG','SEP','OCT','NOV','DEC'/
      DATA Z1,Z2,Z3,DZ1,DZ2,DZ3 /1000.,30000.,122050.,30.,1000.,2000./
C
C
C     SPLINE INITIALIZATION INDEX FOR ATMOSPHERE TABLES
C
      MATM1 = 1
      MATM2 = 1
C
C     LAUNCH SITE AND ATMOSPHERE MONTH SELECTION
C
      IF(MTHRRA.LE.12) THEN
          ITR=1
          IF(MTHRRA.LT.0) MTHRRA=1
          MONTH=MTHRRA
          SITEMO=FLOAT(MONTH)+100.
          ATMOS='RRAETR'
      ELSE
          ITR=2
          IF(MTHRRA.GT.24) MTHRRA=24
          MONTH=MTHRRA-12
          SITEMO=FLOAT(MONTH)+200.
          ATMOS='RRAWTR'
      ENDIF
C
      ATMOSN=ATMOS
      MNTH=RMONTH(MONTH)
      ICNT=1
      Z=0.
C
    3 CALL RRRATM(Z,RHO,PRES,TK,VSOS,DRDZ,DPDZ)
C
```



```
      XMU=1.458E-06*(TK**1.5)/(TK+110.4)
      DENS=RHO/1000.
      PRES=PRES*100.
C
      ATMTBL(ICNT,1) = Z           !(M)
      ATMTBL(ICNT,2) = DENS        !(KG/M3)
      ATMTBL(ICNT,3) = PRES        !(N/M2)
      ATMTBL(ICNT,4) = TK          !DEG-K,KINETIC TEMP)
      ATMTBL(ICNT,5) = VSOS        !(M/S)
      ATMTBL(ICNT,6) = XMU         !(N-S/M2)
C
      IF(ICNT.EQ.MAXPT) GO TO 2
      ICNT=ICNT+1
C
C     ALTITUDE INCREMENT SELECTION
C
C     30 METER INCREMENT FOR 0 TO 1 KM
C     1 KM INCREMENT FOR 1 TO 30 KM ALTITUDE
C     2 KM INCREMENT FOR ALTITUDES ABOVE 30 KM
C
      IF(ABS(Z-Z1).LE..01) Z=Z1
      IF(ABS(Z-Z2).LE..01) Z=Z2
      IF((Z-Z1).LT.0. .AND. IZ1.EQ.0) THEN
           Z=Z+DZ1
           IF(Z.GE.Z1) THEN
             Z=Z1
             IZ1=1
           ENDIF
      ELSE IF(Z.GE.Z1 .AND. Z.LT.Z2) THEN
           Z=Z+DZ2
      ELSE IF(Z.GE.Z2) THEN
           Z=Z+DZ3
      ENDIF
C
      IF(Z.LE.Z3) GO TO 3
C
    2 CONTINUE
C
C
C     OPTION TO WRITE DIRECT ACCESS OUTPUT ATMOSPHERIC DATA FILE
C     IF IN DEMAND MODE
C
C     IF(DEMAND) THEN
C
C        WRITE(6,4)
C        READ(5,5) ANS
C        IF(ANS.EQ.'Y') THEN
C
C             OPEN(UNIT=30,FILE='RRAPLOT.DAT',
C     *       STATUS='UNKNOWN',
C     *       ACCESS='DIRECT',
C     *       RECL=12,
C     *       IOSTAT=IOS)
C
C             IF(IOS.GT.0) THEN
C                WRITE(6,*) 'ERROR OPENING OUTPUT FILE (UNIT=30)'
```

```
C                STOP
C              ELSE
C                 WRITE(6,*) 'OUTPUT FILE (RRAPLOT,UNIT=30) OPENED '
C              ENDIF
C
C         NR=16
C         DO 6 J=1,ICNT
C         WRITE(30'NR,ERR=7) (ATMTBL(J,K),K=1,6)
C         NR=NR+1
C6        CONTINUE
C         GO TO 15
C7        WRITE(6,14) NR
C15       CONTINUE
C         ENDIF
C
C
C     ENDIF
C
C     OUTPUT OF ATMOSPHERIC TABLES ON PRINT FILE
C
C
C     METRIC UNITS OUTPUT
C
        NLINES=6
        DO 8 L=1,ICNT
        IF(NLINES.EQ.6) THEN
           WRITE(JO,9) RMONTH(MONTH),TR(ITR)
           WRITE(JO,10)
        ENDIF
C
        WRITE(JO,11) (ATMTBL(L,M),M=1,6)
        NLINES=NLINES+1
        IF(NLINES.EQ.55) NLINES=6
 8      CONTINUE
C
C
C     ENGLISH UNITS OUTPUT
C
        NLINES=6
        DO 16 L=1,ICNT
        IF(NLINES.EQ.6) THEN
           WRITE(JO,9) RMONTH(MONTH),TR(ITR)
           WRITE(JO,17)
        ENDIF
C
        ATMTBE(L,1) = ATMTBL(L,1) / .3048          !(FT)
        ATMTBE(L,2) = ATMTBL(L,2) / 515.397        !(LBM/FT3)
        ATMTBE(L,3) = ATMTBL(L,3) / 47.880258      !(LBF/FT2)
        ATMTBE(L,4) = ATMTBL(L,4) * 9./5.          !(DEG-R,KINETIC TEMP)
        ATMTBE(L,5) = ATMTBL(L,5) / .3048          !(FT/S)
        ATMTBE(L,6) = ATMTBL(L,6) / 47.880258      !(LBF-S/FT2)
C
        WRITE(JO,11) (ATMTBE(L,M),M=1,6)
        NLINES=NLINES+1
        IF(NLINES.EQ.55) NLINES=6
 16      CONTINUE
```



```
C
      IF(DEMAND) WRITE(6,12) RMONTH(MONTH),TR(ITR)
C
         CLOSE(UNIT=40)
C     RETURN
C
 4       FORMAT(1X,' DO YOU WANT TO WRITE DIRECT-ACCESS ATMOSPHERIC OU',
     *   'TPUT FILE ? , ANSWER Y OR N ')
 5       FORMAT(A1)
 9     FORMAT(1H1,20X,A3,'-',A3,'-RANGE REFERENCE ATMOSPHERE'/)
 10    FORMAT(1X,'GEOMETRIC',6X,'DENSITY',8X,'PRESSURE',9X,'KINETIC',
     *        8X,'SPEED OF',6X,'COEFFICIENT'/
     *        1X,'ALTITUDE',36X,'TEMPERATURE',8X,'SOUND',8X,
     *        'OF VISCOSITY'/
     *        4X,'(M)',9X,'(KG/M3)',9X,'(N/M2)',10X,'(DEG-K)',9X,
     *        '(M/S)',10X,'(N-S/M2)'/)
11     FORMAT(1X,F9.2,5(2X,1PE14.7))
12     FORMAT(10X,A3,'-',A3,' RANGE REFERENCE ATMOSPHERE')
 13         FORMAT(1X,'STOP  --- UNABLE TO ASSIGN RRA*DATA TO UNIT 40')
 14       FORMAT(1X,'ERROR WRITING OUTPUT ATMOSPHERIC FILE (UNIT=30)',
     *           '  NR = ',I6)
 17    FORMAT(1X,'GEOMETRIC',6X,'DENSITY',8X,'PRESSURE',9X,'KINETIC',
     *        8X,'SPEED OF',6X,'COEFFICIENT'/
     *        1X,'ALTITUDE',36X,'TEMPERATURE',8X,'SOUND',8X,
     *        'OF VISCOSITY'/
     *        3X,'(FT)',7X,'(SLUG/FT3)',6X,'(LBF/FT2)',9X,'(DEG-R)',8X,
     *        '(FT/S)',8X,'(LBF-S/FT2)'/)
C
      END
```

3.51 Subroutine RRASPL

3.51.1 Purpose

This subroutine provides spline interpolation of the Range Reference atmosphere model.

3.51.2 Variable Listing

3.51.3 Subroutines Called:

Name	Function
SPLINE	3rd order spline interpolation model

3.51.4 Calling Subroutines:

Name	Function
ADER	Calculates the time independent portion of the equations of motion during forward integration
AINIT	Input routine
BDR1I	Calculates components of the equations of motion for the backward trajectory

3.51.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
aatm	Spline coefficients for atmospheric parameter interpolation				Local	
ap	Atmospheric output array	variabl			Local	
apd	Partial derivative output array for atmospheric parameters				Local	
batm	Spline coefficients for atmospheric parameters interpolation				Local	
catm	Spline coefficients for atmospheric parameters interpolation				Local	
matm2	Previous index of atmospheric parameter spline interpolation		rraint	input	Global	atmos
pr	Output array of atmospheric parameters	variabl			Local	

```
      SUBROUTINE RRASPL(PR)
C
C     SPLINE INTERPOLATION OF RANGE REFERENCE ATMOSPHERE TABLE
C     GENERATED BY RRAINT (INITIALIZATION ROUTINE)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      CHARACTER*3 MNTH
C
      COMMON /ATMOS / ATMTBL(110,6),MTHRRA,MATM1,MATM2,MNTH
      COMMON /ICNTC / ICNT
C
      DIMENSION PR(1),AP(5),APD(5),AATM(5),BATM(5),CATM(5)
C
      CALL SPLINE(5,MATM1,ICNT,PR(1),ATMTBL(1,1),AP,ATMTBL(1,2),APD,2,
     *            MATM2,AATM,BATM,CATM)
C
      PR( 2) = AP(2)                    !PRESSURE (N/M2)
      PR( 3) = APD(2)                   !DPDH (N/M2/M)
      PR( 4) = APD(1)                   !DRDH (KG/M3/M)
      PR( 5) = APD(4)                   !DSDH (M/S/M)
      PR( 6) = AP(1)                    !DENSITY (KG/M3)
      PR( 7) = 0.
      PR( 8) = 0.
      PR( 9) = AP(4)                    !SPEED OF SOUND,VSOS (M/S)
      PR(10) = AP(5)                    !XMU,COEFFICIENT OF VISCOSITY (N-S/M2)
      PR(11) = APD(5)                   !DMUDH (N-S/M2/M)
      PR(12) = AP(3)                    !TEMPERATURE,KINETIC (DEG-K)
      PR(13) = APD(3)                   !DTKDH (DEG-K/M)
      PR(14) = 0.
C
      RETURN
      END
```

3.52 Subroutine RRRATM

3.52.1 Purpose

This subroutine is the Range Reference atmosphere model. The file containing this subroutine also includes subroutines RRRAIN, PR63, and VR71. Subroutine RRRAIN interpolates either the monthly range reference table or the final atmosphere table. Subroutine PR63 calculates atmospheric parameters based on the reference Patrick AFB atmosphere model. Subroutine VR71 calculates atmospheric parameters based on the reference Vandenburg AFB atmosphere model.

3.52.2 Variable Listing

3.52.3 Subroutines Called:

Name	Function
PR63	1963 Patrick AFB reference atmosphere model
RRRAIN	Interpolates either the monthly range reference table or the final atmosphere table
VR71	1971 Vandenburg AFB reference atmosphere model

3.52.4 Calling Subroutines:

Name	Function
RRAINT	Initializes the Range Reference atmosphere model

3.52.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a	Altitude	m			Local	
ain	Storage array				Local	
altfac	Conversion from meters to feet	ft/m			Local	
altsig(7,2)	Dispersion input table (altitude and sigma level)	ft		input	Global	rrrain
ans	Temporary stoarage array				Local	
d	Atmospheric density	slugs/f			Local	
degfac	Temperature conversion factor (degrees K to degrees R)				Local	
demand	Logical indicating the operational mode		initial	input	Global	ccc
denfac	Conversion from g/m^3 to slugs/ft^3				Local	
dens	Atmospheric density	slugs/f			Local	
dmb	Molecular weight tables				Local	
dpdz	Partial derivative of atmospheric pressure with respect to altitude				Local	
drdz	Partial derivative of density with respect to altitude				Local	
dtdz	Partial derivative of temperature with respect to altitude				Local	
f	Temporary value				Local	
finlat(70,4)	Final atmosphere table	variabl		output input	Global	rrrac
fpass	Logical flag denoting pass number				Local	
h	Altitude	m			Local	
i1	Index				Local	
i2	Index				Local	
ier	Error flag				Local	
ii	Index				Local	
in	Atmospheric parameters tables (single precision)				Local	
init	Initialization flag			output	Global	rrrain
irrra	Index				Local	
isite	Site index			output input	Global	rrrac
itab	Index				Local	
itop	Table counter index			output input	Global	rrrac

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
itype	Index to indicate use of either the RRRA atmosphere or Final atmosphere tables				Local	
kk	Index				Local	
lastal	Last point counter			output input	Global	rrrac
lastch	Last change flag				Local	
ll	Index				Local	
lmb	Coefficient for calculation of atmospheric pressure derivative				Local	
lprint	Print line index				Local	
mb	Molecular weight table				Local	
mo	Month index number				Local	
mock	Month index				Local	
month	Index for month				Local	
mwt	Molecular weight				Local	
n	Index				Local	
nchg	Number of dispersion changes			input	Global	rrrain
nlev	Counter index				Local	
nnr	Month index index				Local	
nrec	Record number				Local	
p	Atmospheric pressure	slugs/f			Local	
pb	Pressure coefficients				Local	
pbase	Pressure at base				Local	
pk	Atmospheric pressure coefficients				Local	
pr	Output array of atmospheric parameters	variabl			Local	
pravra	Atmospheric data at 90km	variabl			Local	
pres	Atmospheric pressure	slugs/f			Local	
prnttb	Print flag for final atmospheric tables				Global	rrrain
prsfac	Conversion form lbs/ft^2 to kg/m^2				Local	
psl	Sea level pressure				Local	
rho	Atmospheric density	slugs/f			Local	
rhoh	Atmospheric density				Local	
rhok	Atmospheric density coefficient tables				Local	
riat	Print index				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ro	Atmospheric density				Local	
rropr	Correlation coefficient for pressure				Local	
rrotmp	Correlation coefficient for temperature				Local	
rrraat(52,7)	Atmospheric parameter tables			output input	Global	rrrac
sigma	Sigma level				Local	
site	Launch site index				Local	
sitemo	Site and month index		rraint	input	Global	rrrain
spdso	Speed of sound	ft/sec			Local	
t	Atmospheric temperature				Local	
tempk	Kinetic temperature				Local	
tempm	Molecular temperature				Local	
tempv	Virtual temperature				Local	
tk	Atmospheric temperature coefficient tables				Local	
tm	Molecular temperature				Local	
tmb	Molecular temperature coefficient table				Local	
vs	Speed of sound	ft/sec			Local	
vtk	Virtual temperature coefficient tables				Local	
xk3	Temporary variable				Local	
xk4	Temporary variable				Local	
xk5	Temporary variable				Local	
xk6	Temporary variable				Local	
xlmb	Coefficients for calculation of atmospheric pressure				Local	
z	Altitude	m			Local	
z2	Temporary variable				Local	
z3	Temporary variable				Local	
z4	Temporary variable				Local	
z5	Temporary variable				Local	
zb	Altitude table	m			Local	
zft	Altitude in feet	ft			Local	
zi	Altitude table	m			Local	
zkm	Altitude in kilometers	km			Local	

```
      SUBROUTINE RRRATM (Z,RHO,PRES,TM,VS,DRDZ,DPDZ)
C
C
C     THIS IS A MODIFIED VERSION OF SSL'S RRRATM
C     LARRY LOTT/MSFC-NASA,EL24     MAY 5,1986
C     THE RRRATM SUBROUTINE PROVIDES ATMOSPHERIC PROPERTIES AS FUNCTIONS
C OF ALTITUDE FOR THE SIMULATION OF LAUNCH VEHICLE TRAJECTORIES FROM
C ETR AND WTR. DATA FOR THE ATMOSPHERE MODELS ARE TAKEN FROM THE REVISED
C RANGE REFERENCE ATMOSPHERES(RRRA) DEVELOPED BY THE INTER-RANGE
C INSTRUMENTATION GROUP OF THE RANGE COMMANDERS COUNCIL AND SUPPLIED BY
C O. E. SMITH OF NASA-MFSC. ANNUAL AND TWELVE MONTHLY ATMOSPHERES ARE
C AVAILABLE FOR BOTH LAUNCH SITES.
C     THE PROGRAM USER MAY SELECT A MEAN ATMOSPHERE MODEL OR A DISPERSED
C ATMOSPHERE BASED ON THE RRRA STANDARD DEVIATIONS(SIGMA) OF DENSITY,
C PRESSURE, AND TEMPERATURE. THE DISPERSED DENSITY PROFILE IS SPECIFIED
C BY A TABLE OF SIGMA LEVEL (+ OR -) VERSUS ALTITUDE. CORRESPONDING
C PRESSURE AND TEMPERATURE DISPERSIONS ARE COMPUTED AS CONDITIONAL MEAN
C VALUES BY THE METHOD OF BUELL, WHICH ASSURES ADHERENCE TO THE PERFECT
C GAS LAW. A REALISTIC ATMOSPHERE MODEL ALSO OBEYS HYDROSTATIC
C EQUILIBRIUM AND THIS CAN BE ASSURED APPROXIMATELY THROUGH THE DESIGN
C OF THE ALTITUDE-SIGMA TABLE.
C     THE RRRA MEAN AND DISPERSED DATA TABLES COVER 0-70 KM AT THE
C FOLLOWING SPECIFIC LEVELS FOR THE ETR SITE: 0,.003,1,2,---,30,32,34,--
C -70 KM, AND THE SAME FOR THE WTR SITE EXCEPT FOR THE SECOND LEVEL
C WHICH IS CHANGED FROM .003 TO .1. FROM 70 TO 90 KM THE RRRATM SUB-
C ROUTINE FAIRS THE ATMOSPHERIC PROPERTIES INTO THE APPROPRIATE STANDARD
C REFERENCE ATMOSPHERE(PRA OR VRA). THIS IS ACCOMPLISHED BY ADDING A
C LINE OF PRA OR VRA DATA AT 90 KM IN THE ATMOSPHERE TABLE. ABOVE 90 KM
C THE RRRATM ACCESSES THE STANDARD ATMOSPHERE SUBROUTINES, INCLUDED IN
C TAINED IN THE SUBROUTINES.
C THE SUBROUTINE PACKAGE(PRA63 AND VRA71), WITH ALL THE DATA SELF CON-
C    ON THE FIRST PASS, RRRATM SETS UP THE DESIRED ATMOSPHERE TABLE
C COVERING 0 TO 90 KM AND CALLS SUBROUTINE RRRAIN TO INTERPOLATE
C ATMOSPHERIC PROPERTIES AT A GIVEN ALTITUDE. ON SUBSEQUENT PASSES ONLY
C THE INTERPOLATION IS PERFORMED.
C      SUBROUTINE RRRAIN USES LOGARITHMIC INTERPOLATION ON DENSITY AND
C PRESSURE AND THEIR STANDARD DEVIATIONS AND LINEAR INTERPOLATION ON
C TEMPERATURE AND ITS STANDARD DEVIATION. THE ALTITUDE(Z) IS GEOMETRIC
C ALTITUDE ABOVE MEAN SEA LEVEL IN FEET. IF THE CALLING PROGRAM MEASURES
C ALTITUDE FROAM A DIFFERENT REFERENCE(SUCH AS FISHER ELLIPSOID) THEN AN
C ADJUSTMENT SHOULD BE MADE BEFORE CALLING SUBROUTINE RRRATM. THE OUTPUT
C VARIABLES ARE DENSITY(RHO) IN SLUG/FT**3, PRESSURE(PRES) IN PSF,
C TEMPERATURE(TEMP) IN DEGREES RANKINE, SPEED OF SOUND(VS) IN FPS,
C DERIVATIVE OF DENSITY(DRDZ) IN SLUG/FT**4, AND DERIVATIVE OF PRESSURE
C (DPDZ) IN PSF/FT. THE ABOVE VARIABLES ARE PASSED BACK INTO THE CALLING
C PROGRAM THROUGH THE ARGUMENT LIST, WHILE OTHER DATA AND CONTROLS ARE
C PASSED THROUGH THE COMMON ARRAYS AS NOTED BELOW.
C     THE INPUT TAPE WHICH ROCKWELL USES CONTAINS RRRA DATA IN THE
C FOLLOWING ORDER:
C     ETR ANNUAL PLUS 12 MONTHS (RECORDS 1-13)
C     WTR ANNUAL PLUS 12 MONTHS (RECORDS 14-26)
C
C
C  COMMON STATEMENT VALUES
C      INIT   -   1   FIRST PASS TO SET UP TABLE AND INTERPOLATE FIRST
C                     ALTITUDE
```



```
C                 0   USE AFTER FIRST PASS,JUST INTERPOLATE VALUES FOR
C                     ALTITUDE Z
C
C       PRNTTB -  0.  DONT PRINT FINAL TABLE
C                 1.  PRINT FINAL TABLE
C
C       SITEMO    YXX.  Y=1     SITE ETR
C                       Y=2     SITE WTR
C                       XX = 1-12 MONTHLY ATMOSPHERE (JAN-DEC)
C                       XX = 0    ANNUAL ATMOSPHERE
C       LASTAL        LAST POINT COUNTER TO ELIMINATE UNNECESARY OPERATIONS
C
C       NCHG   - (0-7)     NO. OF DISPERSION CHANGES, 0  IS MEAN
C                          ATMOSHPERE
C
C       ITOP   -       INDEX OF TOP OF FINAL TABLE
C
C
C  PROGRAM ARRAYS
C
C     RRRAAT - THE DESIRED MEAN RRRA DATA - ALTITUDE,PRES,TEMP, DENS
C              AND THEIR RESPECTIVE STD. DEVIATIONS
C
C      ALTSIG - DISPERSION INPUT TABLE, UP TO 7 LOCATIONS. THE TABLE
C               IS ALTITUDE(FT) AND SIGMA LEVEL
C
C     FINLAT - FINAL ATMOSPHERE TABLE - ALTITUDE, TEMP,PRES, AND DENS
C
C      ANS     - TRANSFERS INTERPOLATED VALUES FROM SUB. RRRAIN TO
C               SUB. RRRATM
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      REAL*4 SIN
C
      LOGICAL DEMAND,GRAPH,FPASS
C
      COMMON /CCC   / GRAPH,JO,DEMAND
      DIMENSION ANS(6),PRAVRA(4,2)
      DIMENSION IN(366),AIN(366),SIN(366)
      EQUIVALENCE (IN(1),SIN(1))
      COMMON /RRRAC/ RRRAAT(52,7),FINLAT(70,4),ISITE,ITOP,LASTAL
      COMMON /RRRAIN/INIT,PRNTTB,SITEMO,NCHG,ALTSIG(7,2)
      DATA FPASS /.TRUE./
      DATA INIT/1/
C
C PRAVRA IS THE ATMOSPHERE DATA AT 90 KM,THE ALTITUDE TO FAIR FROM
C THE RRRA DATA TO EITHER THE PATRICK OR VANDENBERG DATA
C PATRICK REF.     ALT.     TEMP        PRES     DENS
      DATA PRAVRA/ 90.,   180.65 ,  .001722,  .003322,
C VAFB REF.
     *             90. ,  180.65,    .001804,  .003481/
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  INITIALIZE DATA ON FIRST PASS ONLY, ELSE INTERPOLATE FINAL TABLE
C
      IF(FPASS) THEN
C
      OPEN(UNIT=40,FILE='DATA:RRADATA.DAT',ACCESS='DIRECT',
     *RECL=366,READONLY,STATUS='UNKNOWN',ERR=1340,IOSTAT=IOS,
     *SHARED)
C
 1340 IF(IOS .NE. 0)THEN
           WRITE(JO,1350) IOS
           IF(DEMAND) WRITE(6,1350) IOS
           GO TO 1300
      ELSE
         WRITE(JO,1330)
         IF(DEMAND) WRITE(6,1330)
      ENDIF
C
      FPASS=.FALSE.
      ENDIF
C
 1330 FORMAT(1X,' DATA:RRADATA.DAT FILE ASSIGNED TO UNIT 40')
 1350 FORMAT(1X,' ERROR OPENING RRADATA.DAT ON UNIT 40. IOS=',I4)
C
C
      ALTFAC = 3.28084
      DEGFAC = 1.8
      PRSFAC = 2.08855
      DENFAC = 1.940323E-06
      Z = Z * ALTFAC
      IF (INIT .NE. 1) GOTO 500
C
C DETERMINE LAUNCH SITE AND MONTH
C
      ISITE = SITEMO/100.
      SITE = ISITE
      MO = SITEMO -100.* SITE
      MONTH = MO
  10  CONTINUE
      NNR=2
  50  CONTINUE
      NREC = NREC + 1
      READ(40'NNR,ERR=1300)IN
C
C     CHANGE INPUT ATMOSPHERIC DATA TO DOUBLE PRECISION
C
      DO 72 II=1,366
        AIN(II) = DBLE(SIN(II))
  72  CONTINUE
C
      IER=366
      IF(IER .EQ. 366) GO TO 99
C     WRITE(6,7001) IER,NREC
C
C7001 FORMAT(1H1,'IER= ',I5,' ON RECORD = ',I5)
```



```
C      CALL ERROFF
 1300    PRINT *,' UNABLE TO READ RRA*DATA NNR= ',NNR
         STOP
  99   CONTINUE
       MOCK = AIN(2)
       ITAB = AIN(1)
       IF(MOCK .NE. MONTH)THEN
          NNR=NNR+1
           IF(NNR.LT.29)GOTO50
       PRINT *, 'UNABLE TO MATCH MONTH= ',SITEMO
       ENDIF
C
       IF(ISITE.EQ.2 .AND. NNR.LT.16) THEN
          NNR=NNR+1
          GO TO 50
       ENDIF
C
       NLEV = 0
       DO 60 I=3,366,7
       NLEV = NLEV + 1
       DO 55 J=1,7
       K = (I+J) -1
       RRRAAT(NLEV,J) = AIN(K)
  55   CONTINUE
  60   CONTINUE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C IF NO. OF CHANGES (NCHG)  IS 0 USE RRRA DATA
C
C      PRINT *,'IN RRRATM NCHG= ',NCHG
       IF (NCHG .NE . 0) GOTO 200
       DO 100 I = 1,52
       FINLAT(I,1) = RRRAAT(I,1)
       FINLAT(I,2) = RRRAAT(I,3)
       FINLAT(I,3) = RRRAAT(I,2)
 100   FINLAT(I,4) = RRRAAT(I,4)
C  ADD EITHER PRA63 OR VRA71 AT 90 KM  AT END OF TABLE
       DO 110 I= 1 ,4
 110   FINLAT(53,I) = PRAVRA(I,ISITE)
C  SET TOP OF TABLE COUNTER(IAT) TO 53 AND GOTO CONVERSION SECTION
       ITOP = 53
       GOTO 340
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C DISPERSION ATMOSPHERE SECTION USING TABLE INPUT(ALTSIG) WITH MAXIMUM
C OF SEVEN POINTS.
C
C  PRINT ERROR MESSAGE AND STOP IF TABLE LARGER THAN 7 LOCATIONS
 200   IF (NCHG .LE. 7) GOTO 220
       WRITE (JO,210)
       IF(DEMAND) WRITE (6,210)
  210 FORMAT(' ONLY 7 INPUT LOCATIONS ARE ALLOWED FOR DISPERSION TABLE')
       STOP
C  MAKE SURE THAT ALTITUDES ARE IN ASCENDING ORDER AND WITHIN BOUNDRIES
```

```
C   OF RRRA ATMOSPHERE, IF NOT WRITE ERROR MESSAGE AND STOP
 220  DO 240 I=1,NCHG
      IF ( I .LT. 2) GOTO 230
      IF (ALTSIG(I,1) .GT. ALTSIG(I-1,1)) GOTO 230
      WRITE (JO,225)
      IF(DEMAND) WRITE (6,225)
 225  FORMAT('  ALTITUDES NOT IN ASCENDING ORDER IN DISPERSION'
     *       ,' INPUT TABLE')
      STOP
 230  IF ((ALTSIG(I,1) .LE. 70. ) .AND. (ALTSIG(I,1) .GE. 0.)) GOTO 240
      WRITE (JO,235)
      IF(DEMAND) WRITE (6,235)
 235  FORMAT (' DESIRED CHANGE ALTITUDE NOT WITHIN BOUNDRIES OF ',
     *  'MEAN ATMOSPHERE TABLE')
      STOP
 240  CONTINUE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C CALCULATE THE DISPERSION TABLE USING THE MEAN ATMOSPHERE TABLE AND
C  THE CHANGE TABLE.
C  INITIALIZE THE SIGMA LEVEL TO 0 TO USE MEAN DATA UNTIL THE SIGMA
C IS CHANGED. INITIALIZE FLAGS TO INTERPOLATE MEAN TABLE
C
      SIGMA = 0.
      LASTAL = 1
      ITYPE = 1
C
C INITIALIZE TABLE COUNTERS TO 1 AND LAST CHANGE FLAG TO 0 TO INDICATE
C  THAT IT IS NOT LAST CHANGE
C
      IRRRA = 1
      ITOP = 1
      LASTCH = 0
      DO 300 I = 1,NCHG
      ZKM = ALTSIG(I,1)
C IF DISPERSION ALT. WITHIN A FOOT OF THE MEAN RRRA ALT. CHANGE SIGMA
C  AND SKIP THAT MEAN DATA
 260  IF ((ABS(ZKM - RRRAAT(IRRRA,1))) .LE. .0003048) GOTO 280
C IF DISPERSION ALT. LESS THAN MEAN ALT. CHANGE SIMA LEVEL
C
      IF (ZKM .LT.  RRRAAT(IRRRA,1)) GOTO 290
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C FILL IN FINLAT TABLE WITH MEAN DATA UNTIL MEAN ALT. > CHANGE ALT.
C CALCULATE CORRELATION COEFFICIENT FOR PRESSURE
C C.C. PRES = (COEF. VARIATION OF RHO**2 + COEF. VAR. OF PRES**2 -
C  COEF. VAR. OF TEMP**2) / (2* COEF. VAR. OF RHO * COEF. VAR. OF PRES)
 270  RROPR = ((RRRAAT(IRRRA,7)/RRRAAT(IRRRA,4))**2 +
     *         (RRRAAT(IRRRA,5)/RRRAAT(IRRRA,2))**2 -
     *         (RRRAAT(IRRRA,6)/RRRAAT(IRRRA,3))**2)
     *         /  ((2* RRRAAT(IRRRA,7)/RRRAAT(IRRRA,4))
     *         *  ( RRRAAT(IRRRA,5)/RRRAAT(IRRRA,2)))
C CALCULATE CORRELATION COEFFICIENT FOR TEMPERATURE
C C.C. PRES = (-COEF. VARIATION OF RHO**2 + COEF. VAR. OF PRES**2 -
C  COEF. VAR. OF TEMP**2) / (2* COEF. VAR. OF RHO * COEF. VAR. OF TEMP)
      RROTMP = ((-(RRRAAT(IRRRA,7)/RRRAAT(IRRRA,4))**2) +
```



```
     *         (RRRAAT(IRRRA,5)/RRRAAT(IRRRA,2))**2 -
     *         (RRRAAT(IRRRA,6)/RRRAAT(IRRRA,3))**2)
     *         / ( (2* RRRAAT(IRRRA,7)/RRRAAT(IRRRA,4))
     *             *( RRRAAT(IRRRA,6)/RRRAAT(IRRRA,3)))
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C INTERPOLATE SIGMA LEVEL OF MEAN VALUES IF BETWEEN TWO CHANGE LEVELS
      IF ((LASTCH .EQ. 1) .OR. (I .EQ. 1)) GOTO 275
      F = (RRRAAT(IRRRA,1) - ALTSIG(I-1,1))/(ALTSIG(I,1)-ALTSIG(I-1,1))
      SIGMA = (1-F)*ALTSIG(I-1,2) + F*ALTSIG(I,2)
C FILL IN FINAL ATMOSPHERE TABLE WITH DISPERSED RRRA DATA
 275  FINLAT(ITOP,1) = RRRAAT(IRRRA,1)
C CALCULATE DISPERSION TEMPERATURE (DEG K)
      FINLAT(ITOP,2) = RRRAAT(IRRRA,3) + SIGMA*RROTMP*RRRAAT(IRRRA,6)
C CALCULATE DISPERSION PRESSURE
      FINLAT(ITOP,3) = RRRAAT(IRRRA,2) + SIGMA*RROPR*RRRAAT(IRRRA,5)
C CALCULATE DISPERSION DENSITY
      FINLAT(ITOP,4) = RRRAAT(IRRRA,4) + SIGMA*RRRAAT(IRRRA,7)
      ITOP = ITOP + 1
      IRRRA = IRRRA +1
C  ADD PRA OR VRA DATA AT 90KM IF ABOVE 70K
      IF (IRRRA .GT. 52) GOTO 320
C FILL IN BOTTOM OF FINLAT TABLE WITH RRRA DATA IF LAST CHANGE
      IF (LASTCH .NE. 0) GOTO 270
C  CHECK NEXT MEAN ALTITUDE
      GOTO 260
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C IF ALTITUDES WITHIN A FOOT OF EACH OTHER CHANGE SIGMA MULT. FACTOR
C AND SKIP MEAN DATA POINT
 280  IRRRA = IRRRA + 1
C CHANGE SIGMA LEVEL IF CHANGE ALT. > MEAN ALT.
 290  SIGMA = ALTSIG(I,2)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C INTERPOLATE PRES,TEMP,DENS AND THEIR RESPECTIVE STD. DER. FOR THE
C CHANGE ALTITUDE
      CALL RRRAIN(ZKM,ANS,ITYPE)
C CALCULATE CORRELATION COEFFICIENT FOR PRESSURE
C C.C. PRES = (COEF. VARIATION OF RHO**2 + COEF. VAR. OF PRES**2 -
C  COEF. VAR. OF TEMP**2) / (2* COEF. VAR. OF RHO * COEF. VAR. OF PRES)
      RROPR = ((ANS(6)/ANS(3))**2 + (ANS(4)/ANS(1))**2
     *       - (ANS(5)/ANS(2))**2)
     *         / ( (2* ANS(6)/ANS(3))* (ANS(4)/ANS(1)))
C CALCULATE CORRELATION COEFFICIENT FOR TEMPERATURE
C C.C. PRES = (-COEF. VARIATION OF RHO**2 + COEF. VAR. OF PRES**2 -
C  COEF. VAR. OF TEMP**2) / (2* COEF. VAR. OF RHO * COEF. VAR. OF TEMP)
      RROTMP = ((-(ANS(6)/ANS(3))**2) + (ANS(4)/ANS(1))**2
     *        -(ANS(5)/ANS(2))**2)
     *         / ( (2* ANS(6)/ANS(3)) *(ANS(5)/ANS(2)))
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C FILL IN FINAL ATMOSPHERE TABLE WITH INTERPOLATED DATA
C  ALT KM
      FINLAT(ITOP,1) = ZKM
C CALCULATE DISPERSED TEMPERATURE (DEG K)
      FINLAT(ITOP,2) = ANS(2) + SIGMA*RROTMP*ANS(5)
C CALCULATE DISPERSED PRESSURE
      FINLAT(ITOP,3) = ANS(1) + SIGMA*RROPR*ANS(4)
```

```
C CALCULATE DISPERSED DENSITY
      FINLAT(ITOP,4) = ANS(3) + SIGMA*ANS(6)
C INCREMENT COUNTERS
      ITOP = ITOP+1
C  IF LAST RRRAAT VALUE ADD PRA OR VRA AT 90K
      IF (IRRRA .GT. 52 ) GOTO 320
C FILL IN TOP OF REST OF TABLE  WITH MEAN DATA IF LAST DISP VALUE
C AND NOT LAST RRRA DATA
      IF (NCHG .GT. I) GOTO 300
      LASTCH = 1
      GOTO 270
C ELSE COMPARE NEXT DISPERSION ALTITUDE
 300  CONTINUE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C ADD PATRICK OR VAFB DATA AT 90K
 320  DO 330 I = 1,4
 330  FINLAT (ITOP,I) = PRAVRA(I,ISITE)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C CHANGE FINAL TABLE FINLAT FROM METRIC TO ENGLISH UNITS
 340  DO 350 I= 1,ITOP
C CHANGE ALT FROM KM TO FT
      FINLAT(I,1) = FINLAT(I,1)*3280.84
C CHANGE TEMP FROM DEGREES K TO DEGREES R
      FINLAT(I,2) = FINLAT(I,2)* 1.8
C CHANGE PRES FROM MB TO PSF
      FINLAT(I,3) = FINLAT(I,3)* 2.08855
C CHANGE DENSITY FROM G/M3 TO SLUG/FT3
 350  FINLAT(I,4) = FINLAT(I,4)* 1.940323E-06
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C SKIP PRINT SECTION IF PRNTTB = 0
      IF (PRNTTB .EQ. 0) GOTO 450
C PRINT THE FINAL TABLE PLUS VALUES FROM 72 TO 126 KM WITH INTERVAL OF
C OF 2 KM, 30 ALTITUDE POINTS EVERY PAGE
C  WRITE HEADER TITLE
      WRITE(JO,600)
C DETERMINE NO. OF PRINT PAGES LPRINT
C INITIALIZE COUNTERS AND STARTING ALTITUDES
      RIAT = ITOP + 27
      II =RIAT
      ZFT= 236220.47
      ITYPE = 2
      LPRINT =  RIAT /30.+ .99
C EACH PRINT BLOCK CONTAINS 30 LINES OF DATA
      DO 400 I=1,LPRINT
C PRINT HEADER BLOCK AND GO TO TOP OF NEXT PAGE STATEMENT IF NOT
C FIRST PAGE
      IF (I .EQ. 1) GOTO 360
      WRITE (JO,610)
  360 WRITE (JO,620)
      I1 = (I-1)*30 + 1
      I2 = I1 + 29
      IF (II .LT. I2) I2 = II
      DO 380 J= I1,I2
      IF (J. GE. ITOP) GOTO 370
```



```
C PRINT FINLAT LINE IF ALT. LESS THAN 70 KM
      WRITE (JO,630) (FINLAT(J,K),K=1,4)
      GOTO 380
C PRINT EITHER FAIRED VALUE, PRA OR VRA VALUE IF ABOVE 70 KM
  370 CALL RRRAIN(ZFT,ANS,ITYPE)
      TM = ANS(2)
      PR = ANS(1)
      RHO = ANS(3)
      WRITE (JO,630) ZFT,TM,PR,RHO
      ZFT = ZFT + 6561.68
 380  CONTINUE
      WRITE (JO,640)
 400  CONTINUE
      ZFT = 236220.47
      DO 440 I=1,II
      IF(I.GT. ITOP) GO TO 420
      A = FINLAT(I,1) / ALTFAC
      T = FINLAT(I,2) / DEGFAC
      P = FINLAT(I,3) / PRSFAC
      D = FINLAT(I,4) / DENFAC
      GO TO 430
 420  CONTINUE
      CALL RRRAIN(ZFT,ANS,ITYPE)
      A = ZFT / ALTFAC
      P = ANS(1) / PRSFAC
      T = ANS(2) / DEGFAC
      D = ANS(3) / DENFAC
      ZFT = ZFT + 6561.68
 430  CONTINUE
      IF(MOD(I,30) .EQ. 1) WRITE(JO,650) ISITE,MO
 650  FORMAT(1H1,30X,'SITE = ',I2,'  MONTH = ',I2/8X,
     *'ATMOSPHERE PROPERTY SUMMARY IN METRIC UNITS'/16X,'(KM)',16X,
     *'T(K)',15X,'P(MB)',9X,'RHO(G/M**3)'/)
      WRITE(JO,651) A,T,P,D
 651  FORMAT(1H ,4(7X,1PE13.5))
 440  CONTINUE
      WRITE(JO,610)
C
C RESET LAST POINT USED COUNTER(LASTAL) TO 1 AND INTERPOLATE 1ST ALTITUDE
C
 450  LASTAL = 1
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C INTERPOLATE FINAL FINLAT TABLE AFTER FIRST PASS
C
 500  ITYPE = 2
      CALL RRRAIN(Z,ANS,ITYPE)
C
C RETURN ANSWERS TO CALLING ROUTINE
C
      PRES = ANS(1)
      TM   = ANS(2)
      RHO  = ANS(3)
      DRDZ = ANS(4)
      DPDZ = ANS(5)
```

```
      VS    = ANS(6)
      Z = Z/ALTFAC
      PRES = PRES / PRSFAC
      TM = TM / DEGFAC
      RHO = RHO / DENFAC
      VS = VS * .3048
      DRDZ = DRDZ / DENFAC * 3.28084
      DPDZ = DPDZ / PRSFAC * 3.28084
      INIT = 0
      RETURN
 600  FORMAT (1H1,44X,'ATMOSPHERIC PROPERTY SUMMARY'//)
 610  FORMAT (1H1)
 620  FORMAT (20X,80('*')/20X,'*',8X,'Z',9X,'*',9X,'T',9X,'*',
     *        9X,'P',9X,'*',8X,'RHO',8X,'*'/
     2        20X,'*',5X,'FEET',9X,'*',6X,'(DEG R)',6X,'*',7X,'(PSF)',
     2        7X,'*',4X,'(SLUG/FT**3)',3X,'*'/
     3        20X,80('*'))
 630   FORMAT (20X,'*',2X,1PE13.5,3X,'*',3(3X,1PE13.5,3X,'*'))
 640  FORMAT (20X,80('*'))
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
      SUBROUTINE RRRAIN(Z,ANS,ITYPE)
C
C
C     THIS SUBROUTINE INTERPOLATES EITHER THE MONTHLY RRRA TABLE(RRRAAT)
C  OR THE FINAL ATMOSPHERE TABLE(FINLAT) FOR THE SUBROUTINE  RRRATM.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  INPUT ARGUMENTS
C
C       Z      - ALTITUDE  IN KM FOR RRRA TABLE AND FEET FOR FINAL
C                ATMOSPHERE TABLE
C
C      ITYPE   -  1  INTERPOLATE RRRA ATMOSHPERE TABLE
C                 2  INTERPOLATE FINAL ATMOSPHERE TABLE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  OUTPUT
C
C    ANS       - ARRAY TO SEND BACK ATMOSPHERE DATA TO SUBROUTINE
C                RRRATM
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION ANS(6)
      COMMON /RRRAC/ RRRAAT(52,7),FINLAT(70,4),ISITE,ITOP,LASTAL
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C           ***********************************************
C           * RRRA ATMOSPHERE TABLE INTERPOLATION SECTION*
```



```
C           ***********************************************
C
C  SET LAST POINT COUNTER TO 1 IF IT IS LESS THAN 1
C
      IF (LASTAL .LE. 0 ) LASTAL = 1
C
C  GET DATA FOR CHANGE ALTITUDES
C
      IF (ITYPE .NE. 1) GOTO 50
      DO 10 J = LASTAL,52
      I = J
      IF (Z - RRRAAT(J,1)) 40 ,30,10
  10  CONTINUE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C IF THE ALTITUDES ARE THE SAME THEN USE THE MEAN DATA
C RESET THE LAST POINT COUNTER
  30  DO 35 LL= 1,6
  35  ANS(LL) = RRRAAT(I,LL+1)
      LASTAL = I
      RETURN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  IF NOT THEN INTERPOLATE THE VALUES
C DETERMINE SCALE FACTOR F AND RESET LASTPOINT COUNTER TO MAKE SURE
C  THAT IT IS BELOW NEXT ALTITUDE TO BE INTERPOLATED
C
  40  F = (Z - RRRAAT(I-1,1))/ (RRRAAT(I,1)- RRRAAT(I-1,1))
      LASTAL = I - 1
C INTERPOLATE PRESSURE LOGARITHMICALLY
      ANS(1) = RRRAAT(I-1,2)* ((RRRAAT(I,2)/RRRAAT(I-1,2))**F)
C INTERPOLATE STANDARD DEVIATION PRESSURE LOGARITHMICALLY
      ANS(4) = RRRAAT(I-1,5)* ((RRRAAT(I,5)/RRRAAT(I-1,5))**F)
C INTERPOLATE DENSITY LOGARITHMICALLY
      ANS(3) = RRRAAT(I-1,4)* ((RRRAAT(I,4)/RRRAAT(I-1,4))**F)
C INTERPOLATE STANDARD DEVIATION DENSITY LOGARITHMICALLY
      ANS(6) = RRRAAT(I-1,7)* ((RRRAAT(I,7)/RRRAAT(I-1,7))**F)
C INTERPOLATE TEMPERATURE LINEARLY
      ANS(2) = (1-F) * RRRAAT(I-1,3)+ F* RRRAAT(I,3)
C  INTERPOLATE S.D. TEMPERATURE LINEARLY
      ANS(5) = (1-F) * RRRAAT(I-1,6) + F*RRRAAT(I,6)
      RETURN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C           ***********************************************
C           * FINAL ATMOSPHERE TABLE INTEROLATION SECTION *
C           ***********************************************
C
C
C EXTRAPOLATE VALUE FROM 1ST AND 2ND POINT IF Z BELOW ZERO
C
 50   IF ( Z .GE. 0.) GOTO 100
      I = 2
      GOTO 220
```

```
C
C  INTERPOLATE  FINAL ATMOSPHERE TABLE
C
 100  IF (FINLAT(ITOP,1) - Z) 110,110, 150
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  IF ALT ABOVE FINLAT ALT USE VRA71 OR PRA63 TABLE
C
  110 IF (ISITE .NE. 1) GOTO 120
C  ETR MEAN ATMOSPHERE
      CALL PR63 (Z,RHO,P,TM,VS,DRDZ,DPDZ)
      GOTO 130
C WTR MEAN ATMOSPHERE
 120  CALL VR71(Z,RHO,P,TM,VS,DRDZ,DPDZ)
  130 ANS(1) = P
      ANS(2) = TM
      ANS(3) = RHO
      ANS(4) = DRDZ
      ANS(5) = DPDZ
      ANS(6) = VS
C
C RESET LAST POINT COUNTER TO 1 TO START AT GROUND,IF MULTIPLE CASES RUN
C
      LASTAL = 1
      RETURN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C DETERMINE WHETHER THE DESIRED ALT IS = ,> OR < LAST POINT USED
C
  150 IF (FINLAT(LASTAL,1)- Z) 200,160, 170
C
C IF SAME USE THE TABLE DATA
  160 P = FINLAT(LASTAL,3)
      T=  FINLAT(LASTAL,2)
      RO = FINLAT(LASTAL,4)
      I = LASTAL + 1
      GOTO 245
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C IF DESIRED ALT < TABLE ALT. BACKTRACK UNTIL THATS NOT TRUE
 170  KK = LASTAL - 1
      DO 180 II=1,KK
      J = LASTAL - II
      I = J
      IF (Z - FINLAT(J,1)) 180, 240, 190
 180  CONTINUE
C
C SETUP I AS THE ALT THAT IS GREATER THAN DESIRED ALTITUDE AND INTER-
C POLATE VALUES
C
 190  I = I + 1
      GOTO 220
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C FIND POINT IN TABLE THAT IS < DESIRED ALT IF TABLE ALT > DESIRED ALT
 200  DO 210 J = LASTAL,ITOP
```



```
      I = J
      IF ( FINLAT(J,1) -Z) 210,240,220
 210  CONTINUE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C DETERMINE SCALE FACTOR F
C
 220  F = (Z - FINLAT(I-1,1))/ (FINLAT(I,1)- FINLAT(I-1,1))
      LASTAL=I-1
C INTERPOLATE PRESSURE LOGARITHMICALLY
      P = FINLAT(I-1,3)* ((FINLAT(I,3)/FINLAT(I-1,3))**F)
C INTERPOLATE DENSITY LOGARITHMICALLY
      RO = FINLAT(I-1,4)* ((FINLAT(I,4)/FINLAT(I-1,4))**F)
C INTERPOLATE TEMPERATURE LINEARLY
      T = (1-F) * FINLAT(I-1,2)+ F* FINLAT(I,2)
      GOTO 245
C
C  IF ALT THE SAME USE THE TABLE VALUES
C
  240 P = FINLAT(I,3)
      T = FINLAT(I,2)
      RO = FINLAT(I,4)
      LASTAL = I - 1
      I = I + 1
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C COMPUTE DENSITY AND PRESURE DERIVATIVES
C
 245  DRDZ = (RO * LOG(FINLAT(I,4)/FINLAT(I-1,4)))/
     *       (FINLAT(I,1) - FINLAT(I-1,1))
      DPDZ = (P * LOG(FINLAT(I,3)/FINLAT(I-1,3)))/
     *       (FINLAT(I,1) - FINLAT(I-1,1))
C PUT INTERPOLATED VALUES INTO ANS ARRAY
 250  ANS(1) = P
      ANS(2) = T
      ANS(3) = RO
      ANS(4) = DRDZ
      ANS(5) = DPDZ
C COMPUTE SPEED OF SOUND AND PUT INTO ANS ARRAY
      ANS(6) = 49.02216* SQRT(ANS(2))
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
      SUBROUTINE PR63 (Z,RHOH,P,TM,VS,DRDZ,DPDZ)                        PRA63
C                                                                       PRA63
C
C 1963 PATRICK ENGLISH UNITS, CORRECTS THE SPEED OF SOUND CONSTANT, K,  PRA63
C FROM 65.7703412 TO 65.77050 WHERE K = (GAMMA * RSTAR / MO) **.5       PRA63
C                                                                       PRA63
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION   TMB (14)  , PB (14)  , ZI ( 5)  , RHOK (6,3) ,        PRA63
     * TK(6,5) , XLMB (14)  , ZB (14)  , VTK(6,3) ,   PK (6,5)         PRA63
C                                                                       PRA63
```

```
C                                                                       PRA63
      DATA  PBASE /  6.23101759E-5 /                                    PRA63
      DATA   ZI /10832.1E0,  17853.3E0,  .28E5,  .49E5,  .83004E5  /    PRA63
C                                                                       PRA63
      DATA   PK                    / 1.6871582 E-2  , -1.1425176 E-4  , PRA63
     *              -1.3612327 E-9 , 7.3624145 E-14 , -1.0800315 E-17 , PRA63
     *               3.3046432 E-22,-7.9910777 E-2  , -8.1046438 E-5  , PRA63
     *              -5.5522383 E-9 , 3.1116969 E-13 , -1.6687827 E-17 , PRA63
     *               3.8319351 E-22, 9.8414277 E-1  , -2.6976917 E-4  , PRA63
     *               8.5227541 E-9 ,-3.9620263 E-13 ,  1.0146471 E-17 , PRA63
     *              -1.0264318 E-22, 1.14118495E+1  , -4.11497477E-4  , PRA63
     *               1.33664855E-8 ,-3.59518975E-13 ,  5.10097254E-18 , PRA63
     *              -2.89055894E-23, 9.99324461E0   , -2.58298177E-4  , PRA63
     *               3.76139346E-9 ,-4.20887236E-14 ,  1.60182148E-19 , PRA63
     *              -1.92508927E-25                                   / PRA63
C                                                                       PRA63
      DATA     RHOK                      /                            PRA63
     *              1.3302117 E-2   , -8.8502064 E-5   , -4.2143056 E-9  PRA63
     *            , 5.9517557 E-13  , -3.9744789 E-17  ,  7.8771273 E-22 PRA63
     *            , 1.2667122 E-1   , -1.3373147 E-4   ,  2.0667371 E-9  PRA63
     *            , 2.3396109 E-13  , -3.2562503 E-17  ,  7.9035209 E-22 PRA63
     *            , 9.2751266 E-1   , -1.4349679 E-4   , -2.8271736 E-9  PRA63
     *            , 4.7480092 E-14  ,  1.8863246 E-18  ,-4.2702411 E-23/ PRA63
C                                                                       PRA63
      DATA     TK                       /                             PRA63
     *              2.9667877 E+2   , -6.7731001 E-3 ,  8.4619805 E-7  , PRA63
     *             -1.7004049 E-10 ,  1.1451454 E-14, -2.4898788 E-19 , PRA63
     *              2.6892151 E+2   ,  4.3075352 E-3 , -8.9159672 E-7  , PRA63
     *             -2.8929791 E-11 ,  5.0724856 E-15, -1.1490372 E-19 , PRA63
     *              3.7064557 E+2   , -3.2858965 E-2 ,  2.0645636 E-6  , PRA63
     *             -4.3283944 E-11 , -5.7507242 E-17,  8.2924583 E-21 , PRA63
     *              2.0447980 E+1   ,  2.07698384E-2 , -8.63038789E-7  , PRA63
     *              1.66392417E-11 , -9.30076185E-17, -4.09005108E-22 , PRA63
     *             -4.98865953E+2   ,  3.92137281E-2 , -4.95180601E-7  , PRA63
     *             -3.26219854E-12 ,  9.66650364E-17, -4.78844279E-22 / PRA63
C                                                                       PRA63
      DATA    VTK                       /                             PRA63
     *              2.9937265 E+2   , -7.717628  E-3 ,  9.4867202 E-7  , PRA63
     *             -1.7136592 E-10 ,  1.1074297 E-14, -2.3294094 E-19 , PRA63
     *              2.6892151 E+2   ,  4.3075352 E-3 , -8.9159672 E-7  , PRA63
     *             -2.8929791 E-11 ,  5.0724856 E-15, -1.1490372 E-19 , PRA63
     *              3.7064557 E+2   , -3.2858965 E-2 ,  2.0645636 E-6  , PRA63
     *             -4.3283944 E-11 , -5.7507242 E-17,  8.2924583 E-21 / PRA63
C                                                                       PRA63
      DATA   ZB              /   9.0 E4 , 1.0 E5 , 1.1 E5 , 1.2 E5 ,   PRA63
     *                 1.5 E5 , 1.6 E5 , 1.7 E5 , 1.9 E5 , 2.3 E5 ,   PRA63
     *                 3.0 E5 , 4.0 E5 , 5.0 E5 , 6.0 E5 , 7.0 E5 /   PRA63
C                                                                       PRA63
      DATA  TMB    /  .18065E3,.21065E3,.26065E3,.36065E3,             PRA63
     *         .96065E3,.111065E4,.121065E4,.135065E4,.155065E4,       PRA63
     *       .183065E4,.216065E4,.242065E4,.259065E4,.270065E4  /      PRA63
C                                                                       PRA63
      DATA XLMB              /   .003E0 , .005E0 , .010E0 , .020E0 ,   PRA63
     *                .015E0 ,  .010E0 , .007E0 , .005E0 , .004E0 ,   PRA63
     *                .0033E0,  .0026E0, .0017E0, .0011E0, .0011E0/   PRA63
C                                                                       PRA63
```



```
      DATA   PB              /       .172244361 E-4  , .315971712 E-5  ,  PRA63
     *             .774389807 E-6  , .265977111 E-6  , .535849383 E-7  ,  PRA63
     *             .391284945 E-7  , .295911117 E-7  , .178715656 E-7  ,  PRA63
     *             .739258171 E-8  , .200573116 E-8  , .430456606 E-9  ,  PRA63
     *             .117315480 E-9  , .370198961 E-10 , .128115330 E-10 /  PRA63
C                                                                        PRA63
C                                                                        PRA63
      H = Z * .3048E0                                                    PRA63
C       IS ALTITUDE GREATER THAN 700000. METERS      NO   NO   YES       PRA63
C                                                                        PRA63
      IF(     H     - 700000.E0)                     15 , 15 , 490       PRA63
C                                                                        PRA63
   15 N = 1                                                              PRA63
      IF (H) 100,60,20                                                   PRA63
  100 H = 0. E0                                                          PRA63
      GO TO 60                                                           PRA63
C                                                                        PRA63
C       IS ALTITUDE LESS THAN 83004. METERS         YES   NO   NO       PRA63
   20 IF( H - 83004.E0)                             40 , 30 , 30        PRA63
C                                                                        PRA63
C       LOCATE H IN ZI TABLE AND SET N.             YES   NO   NO       PRA63
   40 IF( H - ZI(N) )                               60 , 60 , 50        PRA63
   50 N = N + 1                                                          PRA63
      IF (N .LT. 14)  GO TO 40                                           PRA63
      N  =  14                                                           PRA63
      GO TO 60                                                           PRA63
C                                                                        PRA63
C      TABLE LOCATION ESTABLISHED, NOW CALCULATE POWERS OF H.            PRA63
   60 TEMPK = ( TK (1,N) + H* ( TK (2,N) + H*  ( TK (3,N) + H* ( TK (4,N PRA63
     *) + H*  ( TK (5,N) + H* ( TK (6,N) ))))))                          PRA63
      DTDZ = TK(2,N)+2.E0*H*TK(3,N)+3.E0*H*H*TK(4,N)+4.E0*H*H*H*TK(5,N)  PRA63
     1+5.E0*H*H*H*H*TK(6,N)                                              PRA63
C                                                                        PRA63
C       IS ALTITUDE LESS THAN 28000. METERS     YES    NO    NO         PRA63
      IF( H - 28000.E0)                         120 , 140 , 140          PRA63
C                                                                        PRA63
  120 PRES =10.E0* EXP( PK (1,N) + H* ( PK (2,N) + H*  ( PK (3,N) + H* ( PRA63
     * PK (4,N) + H*  ( PK (5,N) + H* ( PK (6,N) ))))))                 PRA63
      DPDZ=PRES*(PK(2,N)+2.E0*H*PK(3,N)+3.E0*H*H*PK(4,N)                 PRA63
     *+ 4.E0*H*H*H*PK(5,N)+5.E0*H*H*H*H*PK(6,N))                         PRA63
C                                                                        PRA63
      DENS =1.16790729E0* EXP(RHOK(1,N) + H* (RHOK(2,N) + H*  (RHOK(3,N) PRA63
     * + H* (RHOK(4,N) + H*  (RHOK(5,N) + H* (RHOK(6,N) ))))))          PRA63
      DRDZ = DENS*(RHOK(2,N)+2.E0*H*RHOK(3,N)+3.E0*H*H*RHOK(4,N)         PRA63
     *+4.E0*H*H*H*RHOK(5,N)+5.E0*H*H*H*H*RHOK(6,N))*.59141E-03           PRA63
C                                                                        PRA63
C       IS ALTITUDE GREATER THAN 12000. METERS    NO    NO   YES       PRA63
      IF ( H - 12000.E0)                          130 , 130 , 160       PRA63
C                                                                        PRA63
  130 TEMPV = ( VTK (1,N) + H* ( VTK (2,N) + H*( VTK (3,N) + H* ( VTK (4 PRA63
     *,N) + H*( VTK (5,N) + H* ( VTK (6,N) ))))))                       PRA63
C     DTDZ=VTK(2,N)+2.D0*H*VTK(3,N)+3.D0*H*H*VTK(4,N)                    PRA63
C    *+4.D0*H*H*H*VTK(5,N)+5.D0*H*H*H*H*VTK(6,N)                        PRA63
      GO TO 170                                                          PRA63
C                                                                        PRA63
  140 PRES =.000980665E0* EXP( PK (1,N) + H* ( PK (2,N) + H*  ( PK (3,N) PRA63
```

```
     * + H* ( PK (4,N) + H*  ( PK (5,N) + H* ( PK (6,N) ))))))          PRA63
      DPDZ = PRES*(PK(2,N)+2.E0*H*PK(3,N)+3.E0*H*H*PK(4,N)              PRA63
     *+4.E0*H*H*H*PK(5,N)+5.E0*H*H*H*H*PK(6,N) )                        PRA63
  150 DENS   =  34.83676E0* PRES / TEMPK                                PRA63
      DRDZ=34.83676E0*((TEMPK*DPDZ-PRES*DTDZ)/(TEMPK*TEMPK))*.59141E-03 PRA63
  160 TEMPV  = TEMPK                                                    PRA63
  170 SPDSO  = SQRT (TEMPV)                                             PRA63
      TEMPM  = TEMPK                                                    PRA63
C                                                                       PRA63
      GO TO  500                                                        PRA63
C                                                                       PRA63
  300 TEMPK  =  180.65E0                                                PRA63
      DTDZ = 0.E0                                                       PRA63
      PRES   = PBASE *  EXP(    -1.18266367 E3 * ( H - 83004.E0)/       PRA63
     *                          ( H + 6344860.E0)              )       PRA63
      DPDZ = -PRES*1.18266367E3*6427864.E0/(H+6344860.E0)**2            PRA63
      GO TO 150                                                         PRA63
C                                                                       PRA63
C       IS ALTITUDE LESS THAN 90000. METERS      YES   NO  NO           PRA63
   30 IF ( H - 90000.E0)                         300 , 70, 70           PRA63
C                                                                       PRA63
C       FIND ALTITUDE IN TABLE AND SET N                                PRA63
   70 IF ( H - ZB(N) )   85 , 400 , 80                                  PRA63
   80 N = N + 1                                                         PRA63
      IF (N .LT. 14)  GO TO 70                                          PRA63
      N  =  14                                                          PRA63
      GO TO 400                                                         PRA63
C                                                                       PRA63
   85 N = N - 1                                                         PRA63
      IF (N.LT. 1) N=1                                                  PRA63
  400 TEMPM  = TMB(N) +XLMB(N) * (H - ZB(N) )                           PRA63
C     DTDZ = XLMB(N)                                                    PRA63
      PRES  =  EXP( LOG(PB(N)) +                                        PRA63
     *   1.373301523 E12 * LOG( TMB(N) /(TMB(N) +XLMB(N)*(H-ZB(N)))) /  PRA63
     *      (XLMB(N) * (6344860.E0+H) * (6344860.E0+ZB(N)) )     )     PRA63
      XK3 = (1.373301523E12/(XLMB(N)*(6344860.E0+H)*(6344860.E0+ZB(N)))) PRA63
      XK4 = TMB(N)/TEMPM                                                PR
      XK5 = XLMB(N)/TEMPM                                               PRA63
      XK6 = XK5+(LOG(XK4)/(6344860.E0+H))
      DPDZ = -XK3*PRES*XK6                                              PRA63
C                                                                       PRA63
      DENS   =  34.83676E0*PRES / TEMPM                                 PRA63
      DRDZ = .206028E-01*((TEMPM*DPDZ-PRES*XLMB(N))/(TEMPM*TEMPM))      PRA63
      SPDSO  = SQRT (TEMPM)                                             PRA63
      GO TO 500                                                         PRA63
C                                                                       PRA63
  490 PRES   =  0.E0                                                    PRA63
      TEMPM = 2700.65E0                                                 PRA63
      DENS   =  0.E0                                                    PRA63
      SPDSO  = 51.9677804E0                                             PRA63
C     DTDZ = 0.D0                                                       PRA63
      DPDZ = 0.E0                                                       PRA63
      DRDZ = 0.E0                                                       PRA63
C                                                                       PRA63
  500 RHOH   = DENS  * .0019403203359E0                                 PRA63
      P=PRES*.09832697E0  *2124.0776E0                                  PRA63
```



```
      TM     = TEMPM*1.8E0                                              PRA63
      VS     = SPDSO * 65.7705000E0                                     PRA63
C     DADZ=(.5D0*VS*DTDZ/TM)*.3048D0*1.8D0                              PRA63
      DPDZ = DPDZ*63.6587E0                                             PRA63
C                                                                       PRA63
      RETURN                                                            PRA63
C                                                                       PRA63
      END                                                               PRA63
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
      SUBROUTINE VR71 (H,RHO,P,TM,VS,DRDZ,DPDZ)                         VRA71
C
C
C     VANDENBERG REFERENCE ATMOSPHERE 1971                              VRA71
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION         PB(14), ZI(5), PK(6,5), RHOK(6,3), TK(6,5), VTK( VRA71
     16,3), ZB(14), TMB(14), LMB(14), DMB(14), MB(14)                   VRA71
      REAL*8 LMB,MB,MWT                                                   VRA71
      DATA PSL/10.1899040E0/                                            VRA71
      DATA PBASE/8.2382921E-5/                                          VRA71
      DATA ZI    /.995E4,.1804E5,.2825E5,.49065E5,.8175E5/              VRA71
      DATA   PK                      /1.8812367E-2,-1.1845111E-4,-3.5545155 VRA71
     1E-10,-6.5016675E-14,-5.2355006E-18,3.4885003E-22,7.5926341E-2,-1.3 VRA71
     2107406E-4,1.4637277E-9,-3.1016812E-13,1.3256585E-17,-1.7447999E-22 VRA71
     3,-1.5119643E0,6.4840176E-5,-2.6797416E-9,-6.7408841E-13,3.2442591E VRA71
     4-17,-4.3323941E-22,-5.0774336E-2,-1.3229173E-4,-3.3047358E-10,-2.8 VRA71
     5081769E-14,1.1235247E-18,-9.9039449E-24,4.9682465E-1,-2.0600983E-4 VRA71
     6,1.1167046E-9,1.3301416E-14,-3.5535259E-19,1.5827238E-24/         VRA71
      DATA   RHOK                    /1.5470847E-2,-1.2568957E-4,8.886513 VRA71
     16E-9,-1.5579407E-12,1.2916496E-16,-4.3994795E-21,-1.8048718E0,3.19 VRA71
     217064E-4,-2.3142097E-8,-1.1163903E-12,1.2861068E-16,-3.0063455E-21 VRA71
     3,-2.6647834E0,2.6811921E-4,-8.3362456E-9,-9.9651556E-13,5.1729970E VRA71
     4-17,-7.0315170E-22/                                               VRA71
      DATA   TK                      /2.8588177E2,3.1763281E-3,-3.1118251E- VRA71
     16,5.2518793E-10,-4.6448685E-14,1.6145641E-18,6.6695824E2,-8.606758 VRA71
     20E-2,3.0132231E-6,3.8218058E-10,-3.3634352E-14,7.5058913E-19,6.644 VRA71
     37200E2,-5.5557022E-2,5.6912588E-7,1.7131166E-10,-7.9541590E-15,1.0 VRA71
     4455683E-19,-3.1441590E2,4.4022912E-2,-8.4909942E-7,-1.8977607E-11, VRA71
     58.5503680E-16,-7.8943270E-21,-2.0223418E3,1.0809620E-1,-1.1246815E VRA71
     6-6,-1.5919380E-11,3.5857350E-16,-1.7624631E-21/                   VRA71
      DATA   VTK                     /2.8715277E2,2.1517327E-3,-2.7196694E VRA71
     1-6,4.5126345E-10,-3.9876511E-14,1.3941650E-18,6.6695824E2,-8.60675 VRA71
     280E-2,3.0132231E-6,3.8218058E-10,-3.3634352E-14,7.5058913E-19,6.64 VRA71
     347200E2,-5.5557022E-2,5.6912588E-7,1.7131166E-10,-7.9541590E-15,1. VRA71
     40455683E-19/                                                      VRA71
      DATA ZB    /9.E4,1.E5,1.1E5,1.2E5,1.5E5,1.6E5,1.7E5,1.9E5,2.3E5,  VRA71
     *            3.E5,4.E5,5.E5,6.E5,7.E5 /                            VRA71
      DATA TMB /.18065E3,.21065E3,.26065E3,.36065E3,.96065E3,.111065E4,  VRA71
     * .121065E4,.135065E4,.155065E4,.183065E4,.216065E4,.242065E4,      VRA71
     * .259065E4,.270065E4  /                                           VRA71
      DATA  LMB             /3.E-3,5.E-3,10.E-3,20.E-3,15.E-3,10.E-3,7.E-3 VRA71
     1,5.E-3,4.E-3,3.3E-3,2.6E-3,1.7E-3,1.1E-3,1.1E-3/                  VRA71
      DATA MB /.289644E2,.2888E2,.2856E2,.2807E2,.2692E2,.2666E2,       VRA71
     * .264E2,.2585E2,.247E2,.2266E2,.1994E2,.1794E2,.1684E2,.1617E2/   VRA71
```

```
      DATA  DMB           /-0.844E-5,-3.20E-5,-4.9E-5,-3.833E-5,2*-2.60E VRA71
     1-5,-2.75E-5,-2.875E-5,-2.914E-5,-2.72E-5,-2.0E-5,-1.1E-5,-0.67E-5, VRA71
     2-0.67E-5/                                                          VRA71
      DATA  PB           /.180492E-4,.330408E-5,.808440E-6,.277325E-6,.5 VRA71
     157663E-7,.407055E-7,.307736E-7,.185742E-7,.767520E-8,.207905E-8,.4 VRA71
     245370E-9,.121190E-9,.381855E-10,.131952E-10/                       VRA71
    5 Z  =  H*.3048E0                                                    VRA71
      IF (Z-.7E6)  15,15,10                                              VRA71
   10 PRES  =  0.E0                                                      VRA71
      TEMPM =  .270065E4                                                 VRA71
      SPDSO =  .519677804E2                                              VRA71
      DENS  =  0.E0                                                      VRA71
C     DTDZ  =  0.D0                                                      VRA71
      DPDZ  =  0.E0                                                      VRA71
      DRDZ  =  0.E0                                                      VRA71
      GO TO 110                                                          VRA71
   15 N  =  1                                                            VRA71
      IF (Z)  200,20,20                                                  VRA71
   20 IF (Z-8175.E1) 25,40,40                                            VRA71
  200 Z = 0.E0                                                           VRA71
      GO TO 35                                                           VRA71
   25 IF (Z-ZI(N))   35,30,30                                            VRA71
   30 N=N+1                                                              VRA71
      IF (N .GE. 5)  GO TO 35                                            VRA71
      GO TO 25                                                           VRA71
   35 Z2=Z*Z                                                             VRA71
      Z3=Z2*Z                                                            VRA71
      Z4=Z2*Z2                                                           VRA71
      Z5=Z2*Z3                                                           VRA71
      GO TO 60                                                           VRA71
   40 IF (Z-.9E5)    100,45,45                                           VRA71
   45 IF (Z-ZB(N)) 55,105,50                                             VRA71
   50 N=N+1                                                              VRA71
      IF (N .GE. 14) GO TO 105                                           VRA71
      GO TO 45                                                           VRA71
   55 N=N-1                                                              VRA71
      GO TO 105                                                          VRA71
C     KINETIC TEMPERATURE FOR (0 TO 81750)                               VRA71
   60 TEMPK=TK(1,N)+TK(2,N)*Z+TK(3,N)*Z2+TK(4,N)*Z3+TK(5,N)*Z4+TK(6,N)*Z VRA71
     15                                                                  VRA71
      DTDZ = TK(2,N)+2.E0*Z*TK(3,N)+3.E0*Z*Z*TK(4,N)+4.E0*Z*Z*Z*TK(5,N)  VRA71
     1+5.E0*Z*Z*Z*Z*TK(6,N)                                              VRA71
      IF (Z-2825.E1)   65,75,75                                          VRA71
C     PRESSURE FOR (0 TO 28250)                                          VRA71
   65 PRES =.1E2 * EXP(PK(1,N)+PK(2,N)*Z+PK(3,N)*Z2+PK(4,N)*Z3+    PK(5,N VRA71
     1)*Z4+PK(6,N)*Z5)                                                   VRA71
      DPDZ=PRES*(PK(2,N)+2.E0*Z*PK(3,N)+3.E0*Z*Z*PK(4,N)                 VRA71
     *+ 4.E0*Z*Z*Z*PK(5,N)+5.E0*Z*Z*Z*Z*PK(6,N))                         VRA71
C     DENSITY FOR (0 TO 28250)                                           VRA71
      DENS=1.2172E0* EXP(RHOK(1,N)+RHOK(2,N)*Z+RHOK(3,N)*Z2+RHOK(4,N)*Z3 VRA71
     * + RHOK(5,N)*Z4+RHOK(6,N)*Z5)                                       VRA71
      DRDZ = DENS*(RHOK(2,N)+2.E0*Z*RHOK(3,N)+3.E0*Z*Z*RHOK(4,N)         VRA71
     *+4.E0*Z*Z*Z*RHOK(5,N)+5.E0*Z*Z*Z*Z*RHOK(6,N))*.59141E-03           VRA71
      IF (Z-.12E5)     70,70,85                                          VRA71
C      VIRTUAL TEMPERATURE FOR (0 TO 12000)                              VRA71
   70 TEMPV=(VTK(1,N)+VTK(2,N)*Z+VTK(3,N)*Z2+VTK(4,N)*Z3+VTK(5,N)*Z4+VTK VRA71
```



```
     1(6,N)*Z5)                                                          VRA71
C     DTDZ=VTK(2,N)+2.D0*Z*VTK(3,N)+3.D0*Z*Z*VTK(4,N)                    VRA71
C    *+4.D0*Z*Z*Z*VTK(5,N)+5.D0*Z*Z*Z*Z*VTK(6,N)                         VRA71
      GO TO 90                                                           VRA71
C     PRESSURE FOR (28250 TO 81750)                                      VRA71
   75 PRES = .1E2* EXP(PK(1,N)+PK(2,N)*Z +PK(3,N)*Z2 +PK(4,N)*Z3 +PK(5,N VRA71
     1)*Z4+PK(6,N)*Z5)                                                   VRA71
      DPDZ = PRES*(PK(2,N)+2.E0*Z*PK(3,N)+3.E0*Z*Z*PK(4,N)               VRA71
     *+4.E0*Z*Z*Z*PK(5,N)+5.E0*Z*Z*Z*Z*PK(6,N) )                         VRA71
C     DENSITY FOR (28250 TO 90000)                                       VRA71
   80 DENS = 34.83676 * (PRES/TEMPK)                                     VRA71
      DRDZ=34.83676E0*((TEMPK*DPDZ-PRES*DTDZ)/(TEMPK*TEMPK))*.59141E-03  VRA71
   85 TEMPV=TEMPK                                                        VRA71
C      SPEED OF SOUND (0 TO 90000)                                       VRA71
   90 SPDSO  =  20.0468E0 *  SQRT(TEMPV)                                 VRA71
      MWT   =  28.9644E0                                                 VRA71
      TEMPM=TEMPK                                                        VRA71
C     CALCULATIONS COMPLETE , RETURN TO MAIN PROGRAM                     VRA71
      GO TO 110                                                          VRA71
  100 TEMPK =  180.65E0                                                  VRA71
C     PRESSURE FOR (81750 TO 90000)                                      VRA71
      PRES =PBASE* EXP((-1.376598941E12*(Z-.8175E5))/(.18065E3*(.6348794 VRA71
     * E7+Z) * (.6348794E7 + .8175E5)))                                  VRA71
      DPDZ = -PRES*1.18266367E3*6427864.E0/(Z+6344860.E0)**2             VRA71
      DTDZ = 0.E0                                                        VRA71
      GO TO 80                                                           VRA71
C MOLECULAR WEIGHT FOR (90000 TO 700000)
 105  MWT=MB(N)+DMB(N)*(Z-ZB(N))
C      MOLECULAR TEMPERATURE FOR (90000 TO 700000)                       VRA71
      TEMPM=TMB(N)+LMB(N)*(Z-ZB(N))                                      VRA71
C      KINETIC TEMPERATURE FOR (90000 TO 700000)                         VRA71
      TEMPK =  (MWT/28.9644E0) * TEMPM                                   VRA71
      PRES = EXP(LOG(PB(N))+(1.376598941E12/(LMB(N)*(.6348794E7 + Z) *   VRA71
     * (.6348794E7 +ZB(N))))*LOG(TMB(N)/(TMB(N)+(LMB(N)*(Z-ZB(N))))))   VRA71
      DENS  = .3483676E2 * PRES/TEMPM                                    VRA71
      SPDSO = 20.0468E0 *  SQRT(TEMPM)                                   VRA71
      TEMPV =  0.E0                                                      VRA71
C     DTDZ =  LMB(N)                                                     VRA71
      XK3 = (1.373301523E12/( LMB(N)*(6344860.E0+Z)*(6344860.E0+ZB(N)))) VRA71
      XK4 = TMB(N)/TEMPM                                                   VR
      XK5 =  LMB(N)/TEMPM                                                VRA71
      XK6 = XK5+(LOG(XK4)/(6344860.E0+Z))
      DPDZ = -XK3*PRES*XK6                                               VRA71
      DRDZ = .206028E-01*((TEMPM*DPDZ-PRES* LMB(N))/(TEMPM*TEMPM))       VRA71
  110 RHO  =  DENS * .0019403203359E0                                    VRA71
      P  =  (PRES/PSL) * 2128.2056E0                                     VRA71
      TM  =  TEMPM * 1.8E0                                               VRA71
      VS  =  SPDSO / .3048E0                                             VRA71
C     DADZ=(.5D0*VS*DTDZ/TM)*.3048D0*1.8D0                               VRA71
      DPDZ = DPDZ*63.6587E0                                              VRA71
      RETURN                                                             VRA71
      END                                                                VRA71
```

3.53 Subroutine RTMRK

3.53.1 Purpose

This subroutine, when called, designates termination of the integration routine.

3.53.2 Variable Listing

3.53.3 Subroutines Called:

None

3.53.4 Calling Subroutines:

Name	Function
ATHREV	Controls thrust event logic during forward integration
BKEND	Terminates backward integration
BTHREV	Controls thrust event during backward integration

3.53.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
kr(1)	Error indicator in integration routine		rtmrk	output	Global	des1

```
      SUBROUTINE RTMRK
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      LOGICAL EOST,SMA,HD,IRST,ERROR,STEPNG,REDOT,ODD
C
      COMMON /DES1/
     *   KR(1),   NMX,   NEQ, NOLD,     M,   MP1,   MP2,    NN, KIND,   KON,
     *     IFL, NTRG, NIVT, NDVT,   NTR,NSTART,NBDIF,   NNT,   NTT,IDL(11),
     *    IDL2, JJSS,J1000,J2000,J4000,J7000,  IRST, EOST,    HD,   SMA,
     *   ERROR,REDOT,STEPNG,
     *   EU(1),    EL,   HMX,   HMN,    YL,     A, ENP1,     D,     B,  SKP,
     *      SK,    HC, DELU, DUSC, HOLD,  VAR,    ZV,    HM,    HP,     R,
     *       Q, DELT,  HMU,  FMU,CK(10),QQ(10),XLJ(11), RJ(11), WJ(11)
      KR(1)=2
C
      RETURN
      END
```

3.54 Subroutine SDATA

3.54.1 Purpose

This is a block data subroutine which stores data for the weight summary table for the Space Shuttle.

3.54.2 Variable Listing

3.54.3 Subroutines Called:

N/A

3.54.4 Calling Subroutines:

N/A

3.54.5 Fortran Listing

sdata

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a(4,67)	Weight storage array for output	lbs	ainit	output	Global	astor

```
      BLOCK DATA
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      COMMON/ASTOR/A(4,67)
C
C     DATA FOR SUMMARY TABLE
C         BASED ON MISSION 1 WINTER WEIGHT SUMMARY DATA FROM RI
C         TRAJECTORY UPDATE(12/15/80) TO ASCENT PERFORMANCE DATA BOOK
C
      DATA(A(I, 1),I=1,3)/0.,0.,0./
      DATA(A(I, 2),I=1,3)/0.,0.,0./
      DATA(A(I, 3),I=1,3)/1608.,1608.,1608./
      DATA(A(I, 4),I=1,3)/2799.,2799.,2799./
      DATA(A(I, 5),I=1,3)/140821.,140821.,140821./
      DATA(A(I, 6),I=1,3)/4259.,4259.,4259./
      DATA(A(I, 7),I=1,3)/0.,0.,0./
      DATA(A(I, 8),I=1,3)/903.,338.,338./
      DATA(A(I, 9),I=1,3)/579.,579.,579./
      DATA(A(I,10),I=1,3)/0.,0.,0./
      DATA(A(I,11),I=1,3)/0.,0.,0./
      DATA(A(I,12),I=1,3)/809.,240.,240./
      DATA(A(I,13),I=1,3)/799.,799.,799./
      DATA(A(I,14),I=1,3)/20484.,20484.,20484./
      DATA(A(I,15),I=1,3)/0.,0.,0./
      DATA(A(I,16),I=1,3)/0.,0.,0./
      DATA(A(I,17),I=1,3)/0.,0.,0./
      DATA(A(I,18),I=1,3)/1337.,2268.,2268./
      DATA(A(I,19),I=1,3)/1383.,1383.,1383./
      DATA(A(I,20),I=1,3)/0.,0.,0.1/
      DATA(A(I,21),I=1,3)/0.,0.,0./
      DATA(A(I,22),I=1,3)/0.,0.,0./
      DATA(A(I,23),I=1,3)/252.,252.,252./
      DATA(A(I,24),I=1,3)/70990.,70990.,70990./
      DATA(A(I,25),I=1,3)/4378.,4480.,4480./
      DATA(A(I,26),I=1,3)/0.,0.,0./
      DATA(A(I,27),I=1,3)/1047.,1100.,1100./
      DATA(A(I,28),I=1,3)/0.,0.,0./
      DATA(A(I,29),I=1,3)/230.,154.,154./
      DATA(A(I,30),I=1,3)/0.,0.,0./
      DATA(A(I,31),I=1,3)/0.,0.,0./
      DATA(A(I,32),I=1,3)/0.,0.,0./
      DATA(A(I,33),I=1,3)/0.,0.,0./
      DATA(A(I,34),I=1,3)/0.,0.,0./
      DATA(A(I,35),I=1,3)/0.,0.,0./
      DATA(A(I,36),I=1,3)/0.,76.,76./
      DATA A(1,37)/0./
      DATA A(1,38)/0./
      DATA A(1,39)/0./
      DATA A(1,40)/0./
      DATA A(1,41)/0./
      DATA A(1,42)/0./
      DATA A(1,43)/0./
      DATA A(1,44)/0./
      DATA A(1,45)/0./
      DATA A(1,46)/0./
```



```
      DATA A(1,47)/351168./
      DATA A(1,48)/0./
      DATA A(1,49)/0./
      DATA A(1,50)/0./
      DATA A(1,51)/0./
      DATA A(1,52)/0./
      DATA A(1,53)/0./
      DATA A(1,54)/0./
      DATA A(1,55)/0./
      DATA A(1,56)/0./
      DATA A(1,57)/0./
      DATA A(1,58)/0./
      DATA A(1,59)/0./
      DATA A(1,60)/0./
      DATA A(1,61)/4./
      DATA A(1,62)/0./
      DATA A(1,63)/0./
      DATA A(1,64)/12426./
      DATA A(1,65)/8691./
      DATA A(1,66)/51./
      DATA A(1,67)/0./
C
      END
```

3.55 Subroutine SEARCH

3.55.1 Purpose

The purpose of this subroutine is to search a table of values for a unique value. Integer values are output which provide the table indicies which surround the value. Used with the interpolation of three dimensional nonlinear aerodynamic data.

3.55.2 Variable Listing

3.55.3 Subroutines Called:

None

3.55.4 Calling Subroutines:

Name	Function
CAERO	Evaluates nonlinear aerodynamics

3.55.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
ang	Search parameter				Local	
ia	Table index before value				Local	
ib	Table index after value				Local	
n	Number of points in table				Local	
tab	Table of values				Local	
temp	Temporary variable				Local	
temp1	Temporary variable				Local	

```
      SUBROUTINE SEARCH(ANG,TAB,N,IA,IB)

c       The purpose of this subroutine is to search a table of
c       values for a unique value.  ia and ib are output and provide
c       the table indicies which surround the value

c       input variables:
c       ang     -       search parameter or value
c       tab     -       table of values
c       n       -       number of points in table

c       output variables:
c       ia      -       table index before value
c       ib      -       table index after value
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION TAB(1)

      TEMP=ANG-TAB(1)

      DO 1 I=2,N
      TEMP1=ANG-TAB(I)
      IF(TEMP*TEMP1.LE.0.) then
         ia=i-1
         ib=i
         return
      endif
      TEMP=TEMP1
    1 CONTINUE

      IF(ANG.LE.TAB(1)) IA=1
      IF(ANG.GE.TAB(N)) IA=N
      IB=IA

      RETURN
C
      END
```

3.56 Subroutine SIMUL

3.56.1 Purpose

This subroutine solves three quadratic equations simulataneously and returns the coefficients. Used to define the specific impulse versus throttle level polynomials based on tabular values.

3.56.2 Variable Listing

3.56.3 Subroutines Called:

None

3.56.4 Calling Subroutines:

Name	Function
AINIT	Input routine

3.56.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a	Temporary storage matrix				Local	
b	Temporary storage matrix				Local	
biginx	Largest index				Local	
bigval	Largest value				Local	
coef	Output coefficient array				Local	
isp	Specific impulse table	sec			Local	
istg	Stage index				Local	
k	Number of columns minus 1				Local	
ncols	Number of columns				Local	
pl	Power level tables				Local	
temp	Temporary variable				Local	
val	Temporary variable				Local	

```
      SUBROUTINE SIMUL(PL,ISP,COEF)
C
C     THIS SUBROUTINE SOLVES THREE QUADRATIC EQUATIONS SIMULTANEOUSLY
C     AND RETURNS THE COEFFICIENTS
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      REAL*8 ISP
C
      INTEGER BIGINX
C
      DIMENSION A(3,4),B(3,3),PL(3,2),ISP(3,2),COEF(3,2)
C
      do istg=1,2
         DO I=1,3
            A(I,1)=PL(I,istg)**2
            A(I,2)=PL(I,istg)
            A(I,3)=1.0
            A(I,4)=ISP(I,istg)
         enddo
C
C     SEARCH FOR LARGEST PIVOT ELEMENT
C
          NCOLS=4
 6        K=NCOLS-1
          IF(K.EQ.1) GO TO 11
          BIGINX=1
          BIGVAL=ABS(A(1,1))
          DO I=2,K
             VAL=ABS(A(I,1))
             IF(BIGVAL.lt.VAL) then
                BIGVAL=VAL
                BIGINX=I
             endif
          enddo
C
C     MAKE DECISION ON ROW INTERCHANGE
C
C
          IF(BIGINX.ne.1) then
C
C     MAKE ROW INTERCHANGE
C
             DO J=1,NCOLS
                TEMP=A(BIGINX,J)
                A(BIGINX,J)=A(1,J)
                A(1,J)=TEMP
             enddo
          endif
C
C     CALCULATE ELEMENTS OF THE B MATRIX
C
 11       DO J=2,NCOLS
             DO I=2,3
                B(I-1,J-1)=A(I,J)-A(1,J)*A(I,1)/A(1,1)
             enddo
```



```
          enddo
          DO J=2,NCOLS
             B(3,J-1)=A(1,J)/A(1,1)
          enddo
C
C     REDUCE COLUMN COUNTER BY ONE
C
          NCOLS=NCOLS-1
C
C     ASSIGN B MATRIX TO THE NAME A
C
          DO J=1,NCOLS
             DO I=1,3
                A(I,J)=B(I,J)
             enddo
          enddo
C
C     CHECK TO SEE IF THE A MATRIX CONSISTS OF JUST ONE COLUMN
C
          IF(NCOLS.ne.1) go to 6
C
C     STORE THE SINGLE MATRIX OF THE A MATRIX IN COEF ARRAY
C
          DO I=1,3
             COEF(I,istg)=A(I,1)
          enddo
        enddo
C
        RETURN
        END
```

3.57 Subroutine SPLINE

3.57.1 Purpose

This subroutine provides a third order spline interpolation method.

3.57.2 Variable Listing

3.57.3 Subroutines Called:

None

3.57.4 Calling Subroutines:

Name	Function
ADER	Calculates the time independent portion of the equations of motion during forward integration
ADER1	Calculates the time dependent portion of the equations of motion during forward integration
BDR1I	Calculates components of the equations of motion for the backward trajectory
MASSPRO	Calculates dynamic mass properties
RRASPL	Interpolates Range Reference atmosphere parameters using the spline method

3.57.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a	Storage array for spline coefficients				Local	
b	Storage array for spline coefficients				Local	
c	Storage array for spline coefficients				Local	
dlx	Temporary variable				Local	
dlxi	Temporary variable				Local	
dlxmi	Temporary variable				Local	
dlxpi	Temporary variable				Local	
dx	Temporary variable				Local	
last	Previous index in table				Local	
m	Array index				Local	
mm1	m-1				Local	
mp1	m+1				Local	
mp2	m+2				Local	
n	Dependent variable tables				Local	
r	Temporary variable				Local	
r2	Temporary variable				Local	
s	Temporary array				Local	
sm1	Temporary array				Local	
sp1	Temporary array				Local	
x	Independent variable table				Local	
xptxpt	Temporary variable				Local	
xt	Independent variable table				Local	
xtxt	Temporary variable				Local	
y	Dependent variable tables				Local	
yd	Partial derivative of dependent variable with respect to independent variable				Local	
yt	Dependent variable tables				Local	

```
      SUBROUTINE SPLINE(L,M,N,X,XT,Y,YT,YD,K,LAST,A,B,C)
C
CC     THIRD ORDER SPLINE INTERPOLATION ROUTINE
CC         L      = NUMBER OF DEPENDENT VARIABLES
CC         M      = PLACE IN THE ARRAY(SET=1 INITIALLY;UPDATED BY ROUTINE)
CC         N      = NUMBER OF POINTS IN INDEPENDENT TABLE
CC         X      = VALUE OF INDEPENDENT VARIABLE
CC         XT     = INDEPENDENT VARIABLE TABLE
CC         Y      = VALUE(S) OF DEPENDENT VARIABLE(S)
CC         YT     = DEPENDENT VARIABLE TABLE(S)
CC         YD     = PARTIAL DERIVATIVES OF DEPENDENT WRT INDEPENDENT
CC         K      = 1,DO NOT CALCULATES PARTIALS;2,CALCULATE PARTIALS
CC         LAST   = PREVIOUS PLACE IN ARRAY(SET=0 INITIALLY;UPDATED BY ROUTINE)
CC         A      = STORAGE ARRAY FOR SPLINE COEFFICIENTS
CC         B      = STORAGE ARRAY FOR SPLINE COEFFICIENTS
CC         C      = STORAGE ARRAY FOR SPLINE COEFFICIENTS
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION XT(1),YT(N,L),S(18),SM1(18),SP1(18),YD(1),Y(1),A(1),
     *B(1),C(1)                                                          SPLI   4
      IF(M.LT.1.OR.M.GT.N)M=1                                            SPLI   5
      IF(X-XT(M).GT.0.)GO TO 2                                           SPLI   6
    1 IF(M.EQ.1)GO TO 4                                                  SPLI   7
      M=M-1                                                              SPLI   8
      IF(X-XT(M))1,4,4                                                   SPLI   9
    2 IF(M.EQ.N)GO TO 5                                                  SPLI  10
      M=M+1                                                              SPLI  11
      IF(X- XT(M).GT.0.)GO TO 2                                          SPLI  12
    5 M=M-1                                                              SPLI  13
    4 MP1=M+1                                                            SPLI  14
      DX=X-XT(M)                                                         SPLI  15
      DLX=XT(MP1)-XT(M)
      IF (DLX .EQ. 0.0) THEN      ! ADDED BY E. BLACK 10-17-1986
          DLXI = 0.0              ! ADDED
      ELSE                        ! ADDED
          DLXI = 1./DLX           ! ADDED
      ENDIF                       ! ADDED                                SPLI  17
      R=DX*DLXI                                                          SPLI  18
      R2=R*R                                                             SPLI  19
      IF(M.EQ.LAST)GO TO 34                                              SPLI  20
      MM1=M-1                                                            SPLI  21
      MP2=MP1+1                                                          SPLI  22
      DO 6 I=1,L                                                         SPLI  23
    6 S(I)=(YT(MP1,I)-YT(M, I))*DLXI                                     SPLI  24
      IF(M.LT.N-1)GO TO 31
      XTXT = XT(M) - XT(MM1)      ! ADDED BY E. BLACK 10-17-1986
      IF (XTXT .EQ. 0.0) THEN     ! ADDED BY E. BLACK 10-17-1986
          DLXMI = 0.0             ! ADDED
      ELSE                        ! ADDED
          DLXMI = 1. / XTXT       ! ADDED
      ENDIF                       ! ADDED                                SPLI  26
      DO 7 I=1,L                                                         SPLI  27
      SM1(I)=(YT(M,I)-Y T(MM1,I))*DLXMI                                  SPLI  28
    7 SP1(I)=2.*S(I)-SM1(I)                                              SPLI  29
      GO TO 33
```



```
   31 CONTINUE                                                           SPLI  30
      XPTXPT = XT(MP2) - XT(MP1)     ! ADDED BY L. LOTT 02-27-87
      IF (XPTXPT .EQ. 0.0) THEN      ! ADDED
          DLXPI = 0.0                ! ADDED
      ELSE                           ! ADDED
          DLXPI = 1. / XPTXPT        ! ADDED
      ENDIF                          ! ADDED
      DO 8 I=1,L                                                         SPLI  32
    8 SP1(I)=(YT(MP2,I)-YT(MP1,I))*DLXPI                                 SPLI  33
      IF(M.GT.1)GO TO 32                                                 SPLI  34
      DO 9 I=1,L                                                         SPLI  35
    9 SM1(I)=2.*S(I)-SP1(I)                                              SPLI  36
      GO TO 33                                                           SPLI  37
   32 CONTINUE                                                           SPLI  38
      DLXMI=1./(XT(M)-XT(MM1))                                           SPLI  39
      DO 10 I=1,L                                                        SPLI  40
   10 SM1(I)=(YT(M,I)-Y T(MM1,I))*DLXMI                                  SPLI  41
   33 CONTINUE                                                           SPLI  42
      DO 11 I=1,L                                                        SPLI  43
      A(I)=.5*(SP1(I)+SM1(I))-S(I)                                       SPLI  44
      B(I)=.5*(S(I)-SP1(I))+S(I)-SM1(I)                                  SPLI  45
   11 C(I)=.5*(S(I)+SM1(I))                                              SPLI  46
   34 IF(K.EQ.2)GO TO 13                                                 SPLI  47
      DO 12 I=1,L                                                        SPLI  48
   12 Y(I)=YT(M,I)+(A(I)*R2+B(I)*R+C(I))*DX                              SPLI  49
      LAST=M                                                             SPLI  50
      RETURN                                                             SPLI  51
   13 DO 14 I=1,L                                                        SPLI  52
      Y(I)=YT(M,I)+(A(I)*R2+B(I)*R+C(I))*DX                              SPLI  53
   14 YD(I)=3.*A(I)*R2+2.*B(I)*R+C(I)                                    SPLI  54
      LAST=M                                                             SPLI  55
      RETURN                                                             SPLI  56
      END                                                                SPLI  57
```

3.58 Subroutine SUM15STG

3.58.1 Purpose

This subroutine reads trajectory output file, calculates weights based on events, and prints weight summary table for liquid vehicle.

3.58.2 Variable Listing

3.58.3 Subroutines Called:

Name	Function
FUNISP	Calculates specific impulse based on throttle level

3.58.4 Calling Subroutines:

Name	Function
AFORND	Defines parameters at the terminal and at intermediate points on the forward trajectory

3.58.5 Fortran Listing

sum15stg

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a(4,67)	Weight storage array for output	lbs	ainit	output input	Global	astor
b	Input array from output file produced by aprtn				Local	
c	Trajectory events names array				Local	
chvel	Characteristic velocity	m/sec	mastre	input	Global	agen1
deltav	Delta velocity required	ft/sec			Local	
dvisp	Specific impulse of upper stage	sec			Local	
dwdump	Amount of RCS dumped propellant	lbs			Local	
dwfix	Amount of propellant burned by fixed MPS engines	lbs			Local	
dwlbm	Amount of LBM propellant burned	lbs			Local	
dwmps	Amount of MPS propellant burned	lbs			Local	
dwoms	Amount of OMS propellant burned	lbs			Local	
dwrcs	Amount of RCS propellant burned	lbs			Local	
dwsrm	Amount of SRM propellant burned	lbs			Local	
etprwt	Total ET propellant weight	lbs			Local	
fpr	Flight performance reserve factor				Local	
fprfac	Flight performance reserve factor		ainit	input	Global	agen1
gzero	Gravitational constant (9.80665 m/sec)	m/sec^	ainit	output input	Global	const
head	Storage array for headings in printout		ainit	input	Global	agen1
jum	Jump start flag		ainit	input	Global	agen3
jump	Jump start flag				Local	
kflag6	Flag index to indicate lift-off mode				Local	
l1	Index				Local	
l2	Index				Local	
lbm	Denotes the simulation of liquid booster module			output	Global	lonoff
lbmtnk	Total LBM propellant weight	lbs			Local	
m	Mission designate (equivalent to nb)				Local	
mision	Mission designate (equivalent to nb)				Local	
misson	Not used				Local	
n	Index				Local	
na	Format used in variable format				Local	
nak	Format used in variable format				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
nak1	Format used in variable format				Local	
nak2	Format used in variable format				Local	
name	Name of events in weight output table				Local	
nb	Mission designate name from output file				Local	
nbk	Formats for variable output format				Local	
nbk1	Formats for variable output format				Local	
nbk2	Formats for variable output format				Local	
nbk3	Formats for variable output format				Local	
nbk4	Formats for variable output format				Local	
nbk5	Formats for variable output format				Local	
nfor	Variable format parameter				Local	
nord	Index for variable formatting the output				Local	
nstop	Terminal portion of variable format statement				Local	
nstr	Formats for variable output format				Local	
ntable	Flag indicating that output tables will (>0) or will not (=0) be generated			output	Global	alat
numb	Number of parameters to be printed for each line of output				Local	
oms	Total OMS propellant weight	lbs			Local	
omsdum	Not used	lbs			Local	
omstnk	Total OMS propellant weight	lbs			Local	
prctet	Not used				Local	
prctor	Not used				Local	
prop(6)	Propellant weight for each engine type	kg	ainit	input	Global	agen1
ptkg	Conversion from pounds to kilograms	kg/lbs	ainit	output	Global	const
rcs	Total RCS propellant weight	lbs			Local	
rcstnk	Total RCS propellant weight	lbs			Local	
stime	Saved value of time	sec			Local	
svlbm	Saved value of burned LBM propellant	lbs			Local	
svmps	Saved value of burned MPS propellant	lbs			Local	
svoms	Saved value of burned OMS propellant	lbs			Local	
svrcs	Saved value of burned RCS propellant	lbs			Local	
svsrm	Saved value of burned SRM propellant	lbs			Local	
t	Time from liftoff	sec			Local	

sum15stg

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
tglim	Not used	sec			Local	
xm	Total vehicle mass	lbs			Local	
z	Names of mission designate				Local	

```
      SUBROUTINE Sum15stg
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      COMMON/CCC/GRAPH,JO,DEMAND

      LOGICAL DEMAND,GRAPH,LBM,JUMP

      CHARACTER*42 NAME(42)
      CHARACTER*6 NSTR(2),NBK(4),NBK1(4),NBK2(4),NBK3(4),NBK4(4),NBK5(4)
     *,NFOR(18),Z(4),NB,NSTOP,NAK1,NAK2,NAK,MISION,MISSON
      character*20 na,c(10)
      CHARACTER*60 HEAD

      REAL*8 LBMTNK

      DIMENSION NORD(42),NUMB(42),B(160),PRCTET(3),PRCTOR(3)

      COMMON/AGEN1/TIME(2,16),TAUT(15),TAUW(15),TZERO,TLIFT,TTILT,TMINH,
     *DTZ,TQ,TL,XMAUG,TNE(6,15),ENGDAT(8,15),S(15),WD(15),WJET(15),
     *HEAD,PRINT(15),STEP(15),HNOM(15),HMAX(15),PROP(6),TBEGR,TENDR,
     *CHRDOT,CHIRO,FAZ,BO(3,2),CBAXIL,TCHIR,VRCUT,FPRFAC,CHVEL

      COMMON/AGEN3/ITHR,IWD,IXR,JUM ,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISSON,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)

      COMMON/ALAT/ALAT,ALONG,ALTLS,NTABLE

      COMMON/ASTOR/A(4,67)

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM

      COMMON/LONOFF/LBM

      COMMON/OMSRCS/FOMS(15),FRCS(15),WDOMS(15),WDRCS(15),WDDUMP(15)

      EQUIVALENCE(T,B(1)),(XM,B(6)),(DWMPS,B(67)),(DWSRM,B(68)),
     *(DWLBM,B(69)),(DWOMS,B(70)),(DWRCS,B(71)),(DWFIX,B(72)),
     *(DWDUMP,B(129)),(MISION,NB)

      DATA NUMB/2,2,2,2,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,2,2,
     *2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1/

      DATA NORD/5,1,1,1,2,6,1,1,5,1,5,1,1,1,1,1,1,5,1,1,1,1,1,4,1,5,
     *1,1,5,1,1,1,1,1,1,1,5,1,5,1,1,5/

      DATA C/
     *       'IGNITION            ','LIFT-OFF           ',
     *       'LBM IGNITION        ','SEPARATION         ',
     *       'LBM SEPARATION      ','CORE SEPARATION    ',
     *       'INJECTION           ','ENDPRT             ',
     *       'INTERMEDIATE        ','                   '/
      DATA Z/'NOM   ','ENGOUT','POST N','POST A'/
```



```
      DATA PRCTET/.487,.450,.799/
      DATA PRCTOR/.513,.550,.201/

      DATA NAME( 1)/'PAYLOAD                                          '/
      DATA NAME( 2)/'UPPER STAGE - JETTISON WEIGHT                    '/
      DATA NAME( 3)/'            - INERT STRUCTURE                    '/
      DATA NAME( 4)/'            - FLIGHT PERFORMANCE RESERVE         '/
      DATA NAME( 5)/'            - DELTA VELOCITY PROPELLANT          '/
      DATA NAME( 6)/'            - EXCESS PROPELLANT(PROP LD)         '/
      DATA NAME( 7)/'            - FUEL BIAS                          '/
      DATA NAME( 8)/'            - PROP RESIDUALS/UNUSABLE            '/
      DATA NAME( 9)/'WEIGHT AT ORBIT INSERTION                        '/
      DATA NAME(10)/'UPPER STAGE - PROPELLANT BURNED                  '/
      DATA NAME(11)/'WEIGHT AFTER 2ND STAGE SEPARATION                '/
      DATA NAME(12)/'PERFORMANCE MARGIN - PAYLOAD                     '/
      DATA NAME(13)/'                   - OTHER                       '/
      DATA NAME(14)/'2ND STAGE - JETTISON WEIGHT                      '/
      DATA NAME(15)/'          - STAGE INERT WEIGHT (LESS ENG)        '/
      DATA NAME(16)/'          - NONPROPULSIVE CONSUMABLES            '/
      DATA NAME(17)/'          - DRY WEIGHT OF ENGINE(S)              '/
      DATA NAME(18)/'2ND STAGE - TOTAL WEIGHT AT 2ND STAGE SEP        '/
      DATA NAME(19)/'          - PROP RESIDUALS/UNUSABLES             '/
      DATA NAME(20)/'          - FLIGHT PERFORMANCE RESERVE           '/
      DATA NAME(21)/'          - FUEL BIAS                            '/
      DATA NAME(22)/'          - MPS THRUST BUILDUP                   '/
      DATA NAME(23)/'          - BOILOFF,OVERFILL,DRAINBACK           '/
      DATA NAME(24)/'EXCESS MPS PROPELLANT (MPS PROP LOADED)          '/
      DATA NAME(25)/'SHROUD WEIGHT SEPARATED AT MECO                  '/
      DATA NAME(26)/'INJECTED WEIGHT AT MECO                          '/
      DATA NAME(27)/'SHROUD JETTISON WEIGHT                           '/
      DATA NAME(28)/'PROPELLANT BURNED - SEPARATION TO MECO           '/
      DATA NAME(29)/'WEIGHT AT SEPARATION                             '/
      DATA NAME(30)/'1ST STAGE - JETTISON WEIGHT                      '/
      DATA NAME(31)/'          - STAGE INERT WEIGHT (LESS ENG)        '/
      DATA NAME(32)/'          - DRY WEIGHT OF ENGINE(S)              '/
      DATA NAME(33)/'          - NONPROPULSIVE CONSUMABLES            '/
      DATA NAME(34)/'          - FUEL BIAS                            '/
      DATA NAME(35)/'          - PROP RESIDUALS/UNUSABLES             '/
      DATA NAME(36)/'PROPELLANT BURNED - LIFTOFF TO SEPARATION        '/
      DATA NAME(37)/'LIFTOFF (T/W=1.0)                                '/
      DATA NAME(38)/'MPS PROPELLANT                                   '/
      DATA NAME(39)/'IGNITION COMMAND (T-0)                           '/
      DATA NAME(40)/'1ST STAGE - MPS THRUST BUILDUP                   '/
      DATA NAME(41)/'          - BOILOFF,OVERFILL,DRAINBACK           '/
      DATA NAME(42)/'PRELAUNCH WEIGHT (T-5 MIN)                       '/

      DATA NSTR/'(1X,A4','2     '/,NSTOP/')     '/
      DATA NAK1/',F8.0 '/,NAK2/',8X   '/
      DATA NBK1/',14X  ',','      ','      ','      '/
      DATA NBK2/',2H (',','F6.0,5','H F/S)',',1X   '/
      DATA NBK3/',3X,6H','MARGIN',',5X   ',','      '/
      DATA NBK4/',1X,2H','( ,F8.','0,1H),',',2X    '/
      DATA NBK5/',6X,F8',',0    ',','      ','      '/

      GZERO=9.80665
      PTKG=2.2046223
```

```
      ETPRWT=PROP(1)
      LBMTNK=PROP(3)
      OMSTNK=PROP(4)
      RCSTNK=PROP(5)

      N=1

   28 L1=1
      L2=29

      TGLIM=10000.

      M=0
      OMS=0.
      RCS=0.

      K=1
      L=1

      kflag6=0
      JUMP=.FALSE.

      IF(N.EQ.1) REWIND 9

   31 READ(9,END=1,ERR=210) NA,B,NB

      IF(NA.EQ.C(8)) GO TO 31

      DWMPS=DWMPS+DWFIX

      IF(M.eq.0) then
         do i=1,4
            IF(MISION.EQ.Z(I)) GO TO 30
         enddo
   30    M=I
      endif

      IF(M.GT.2) GO TO 31
      IF(NA.EQ.C(10)) GO TO 31
      do k=1,7
         IF(NA.EQ.C(K)) GO TO 32
      enddo
      GO TO 31
   32 if(k.eq.1) then
         STIME=T
         IF(ABS(T).gt..1E-03) then
            JUMP=.TRUE.
            L=JUM
            SVMPS=DWMPS
            SVOMS=DWOMS
            SVRCS=DWRCS
         else
            A(1,42)=XM
         endif
         GO TO 31
       else if(k.eq.2) then
```



```
         A(1,37)=XM
         A(1,38)=DWMPS
         A(1,39)=A(1,37)+A(1,38)
         A(1,42)=A(1,39)+A(1,40)+A(1,41)
         SVMPS=DWMPS
         SVSRM=DWSRM
         GO TO 31
       else if(k.eq.3) then
         A(1,55)=XM
         A(1,57)=DWMPS-SVMPS
         A(1,58)=DWSRM-SVSRM
         SVMPS=DWMPS
         SVSRM=DWSRM
         LBM=.TRUE.
         GO TO 31
       else if(k.eq.4) then
         A(1,29)=XM
         A(1,36)=DWMPS-SVMPS
         SVMPS=DWMPS
         GO TO 31
       else if(k.eq.5) then
         A(1,40)=XM
         A(1,45)=DWMPS-SVMPS
         A(1,46)=DWLBM-SVLBM
         SVMPS=DWMPS
         SVLBM=DWLBM
         GO TO 31
       else if(k.eq.6) then
         A(m,26)=XM
         A(m,28)=DWMPS-SVMPS
         SVOMS=DWOMS
         kflag6=1
         GO TO 31
       else if(k.eq.7) then
         if(kflag6.eq.0) then
            A(M,26)=XM
            A(M,28)=DWMPS-SVMPS
         endif
         kflag6=0
         a(1,30)=a(1,31)+a(1,32)+a(1,33)+a(1,34)+a(1,35)

         FPR=XM*(1.-EXP(-CHVEL*FPRFAC/(GZERO*FUNISP(1.,2))))

C        A(M,26)=PRCTET(M)*FPR
C        A(M,20)=PRCTOR(M)*FPR-A(M,21)
         a(m,24)=etprwt-(a(m,19)+a(m,20)+a(m,21)+a(m,22)+
     *            a(m,23)+a(m,28)+a(1,34)+a(1,35)+
     *            a(1,36)+a(1,38)+a(1,40)+a(1,41))
         a(m,26)=a(1,29)-(a(m,27)+a(m,28))
         a(m,18)=a(m,26)-a(m,25)
         a(m,14)=a(m,15)+a(m,16)+a(m,17)+a(m,19)+a(m,20)+a(m,21)+
     *            a(m,24)
       endif

   46 a(m,11)=a(m,18)-(a(m,12)+a(m,13)+a(m,14))
      deltav=a(3,5)
```

```
      dvisp=a(4,5)
      if(dvisp.gt.0.) then
         a(m,5)=xm*(1.-exp(-deltav*.3048/(gzero*dvisp)))
      else
         a(m,5)=0.
      endif

      a(m,10)=dwoms-svoms
      a(m,9)=a(m,11)-a(m,10)
      a(m,6)=omstnk-(a(m,4)+a(m,5)+a(m,7)+a(m,8)+a(m,10))
      a(m,2)=a(m,3)+a(m,4)+a(m,5)+a(m,6)+a(m,7)+a(m,8)
      a(m,1)=a(m,9)-(a(m,2)+a(m,5)+a(m,6))

      M=0
      GO TO 31

    1 WRITE(JO,101) HEAD
      WRITE(21,101)HEAD

  101 FORMAT(1H1,////36X,A60///49X,'NOMINAL',14X,'ENGINE OUT'//)

      NFOR(1)=NSTR(1)
      NFOR(2)=NSTR(2)
      NFOR(18)=NSTOP

      DO 5 K=L1,L2

      NAK=NAK1
      IF(NORD(K).EQ.0.OR.NORD(K).EQ.5) NAK=NAK2

      NFOR(3)=NAK
      NFOR(8)=NAK

      NFOR(13)=NAK

      IF(K.GT.28) then
         NFOR(8)=NAK2
         nfor(13)=nak2
      endif

      do i=1,4
         NBK(I)=NBK1(I)
         IF(NORD(K).EQ.2) NBK(I)=NBK2(I)
         IF(NORD(K).EQ.3) NBK(I)=NBK3(I)
         IF(NORD(K).EQ.4) NBK(I)=NBK4(I)
         IF(NORD(K).EQ.5) NBK(I)=NBK5(I)
         IF(NORD(K).EQ.6) NBK(I)=NBK4(I)
      enddo

      do i=1,4

         J=I+3

         NFOR(J)=NBK(I)
         IF(NORD(K).EQ.3) NFOR(J)=NBK1(I)
```



```
         J=I+8

         NFOR(J)=NBK(I)

         IF(K.GT.28) NFOR(J)=NBK1(I)

         IF(NORD(K).EQ.6) NFOR(J)=NBK1(I)

         J=I+13

         NFOR(J)=NBK(I)
         IF(NORD(K).NE.5.OR.K.GT.24) NFOR(J)=NBK1(I)

      enddo

      N=NUMB(K)

      IF(NORD(K).EQ.5.AND.K.NE.L1) WRITE(JO,102)
      IF(NORD(K).EQ.5.AND.K.NE.L1) WRITE(21,102)

      if(n.ne.0) then
         if(nord(k).eq.2) then
            write(jo,nfor) name(k),a(1,k),deltav,a(2,k),deltav
            write(21,nfor) name(k),a(1,k),deltav,a(2,k),deltav
         else if(nord(k).eq.4) then
            WRITE(JO,NFOR) NAME(K),A(1,K),ETPRWT,A(2,K),a(3,k)
            WRITE(21,NFOR) NAME(K),A(1,K),ETPRWT,A(2,K),a(3,k)
         else if(nord(k).eq.6) then
            write(jo,nfor) name(k),a(1,k),omstnk,a(2,k)
            write(21,nfor) name(k),a(1,k),omstnk,a(2,k)
         else
            WRITE(JO,NFOR) NAME(K),(A(J,K),J=1,N)
            WRITE(21,NFOR) NAME(K),(A(J,K),J=1,N)
         endif
      else
         WRITE(JO,NFOR) NAME(K)
         WRITE(21,NFOR) NAME(K)
      endif

      IF(NORD(K).EQ.5) WRITE(JO,102)
      IF(NORD(K).EQ.5) WRITE(21,102)

  102 FORMAT( )

    5 CONTINUE

      IF(L1.eq.1) then
         L1=29
         L2=42
         GO TO 1
      endif

      IF(NTABLE.LE.1.OR.N.EQ.NTABLE) RETURN

      N=N+1
```

```
      GO TO 28

  210 WRITE(*,*) ' ERROR READING TAPE 9'

      RETURN
      END
```

3.59 Subroutine SUMARY

3.59.1 Purpose

This subroutine reads trajectory output file, calculates weights based on events, and prints weight summary table for Space Shuttle.

3.59.2 Variable Listing

3.59.3 Subroutines Called:

Name	Function
FUNISP	Calculates specific impulse based on throttle level

3.59.4 Calling Subroutines:

Name	Function
AFORND	Defines parameters at the terminal and at intermediate points on the forward trajectory

3.59.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a(4,67)	Weight storage array for output	lbs	ainit	output input	Global	astor
b	Input array from output file produced by aprtn				Local	
c	Trajectory events names array				Local	
chvel	Characteristic velocity	m/sec	mastre	input	Global	agen1
dwdump	Amount of RCS dumped propellant	lbs			Local	
dwfix	Amount of propellant burned by fixed MPS engines	lbs			Local	
dwlbm	Amount of LBM propellant burned	lbs			Local	
dwmps	Amount of MPS propellant burned	lbs			Local	
dwoms	Amount of OMS propellant burned	lbs			Local	
dwrcs	Amount of RCS propellant burned	lbs			Local	
dwsrm	Amount of SRM propellant burned	lbs			Local	
etprwt	Total ET propellant weight	lbs			Local	
fpr	Flight performance reserve factor				Local	
fprfac	Flight performance reserve factor		ainit	input	Global	agen1
gzero	Gravitational constant (9.80665 m/sec)	m/sec^	ainit	output input	Global	const
head	Storage array for headings in printout		ainit	input	Global	agen1
jum	Jump start flag		ainit	input	Global	agen3
jump	Jump start flag				Local	
l1	Index				Local	
l2	Index				Local	
lbm	Denotes the simulation of liquid booster module			output	Global	lonoff
lbmtnk	Total LBM propellant weight	lbs			Local	
m	Mission designate (equivalent to nb)				Local	
mision	Mission designate (equivalent to nb)				Local	
misson	Not used				Local	
n	Index				Local	
na	Format used in variable format				Local	
nak	Format used in variable format				Local	
nak1	Format used in variable format				Local	
nak2	Format used in variable format				Local	
name	Name of events in weight output table				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
nb	Mission designate name from output file				Local	
nbk	Formats for variable output format				Local	
nbk1	Formats for variable output format				Local	
nbk2	Formats for variable output format				Local	
nbk3	Formats for variable output format				Local	
nbk4	Formats for variable output format				Local	
nbk5	Formats for variable output format				Local	
nfor	Variable format parameter				Local	
nord	Index for variable formatting the output				Local	
nstop	Terminal portion of variable format statement				Local	
nstr	Formats for variable output format				Local	
ntable	Flag indicating that output tables will (>0) or will not (=0) be generated			output	Global	alat
ntra	Additional name array used if LBM is not simulated				Local	
numb	Number of parameters to be printed for each line of output				Local	
oms	Total OMS propellant weight	lbs			Local	
omsdum	Not used	lbs			Local	
omstnk	Total OMS propellant weight	lbs			Local	
prctet	Percentage of external tank to total vehicle proprellant mass				Local	
prctor	Percentage of orbiter to total propellant weight				Local	
prop(6)	Propellant weight for each engine type	kg	ainit	input	Global	agen1
ptkg	Conversion from pounds to kilograms	kg/lbs	ainit	output	Global	const
rcs	Total RCS propellant weight	lbs			Local	
rcstnk	Total RCS propellant weight	lbs			Local	
stime	Saved value of time	sec			Local	
svlbm	Saved value of burned LBM propellant	lbs			Local	
svmps	Saved value of burned MPS propellant	lbs			Local	
svoms	Saved value of burned OMS propellant	lbs			Local	
svrcs	Saved value of burned RCS propellant	lbs			Local	
svsrm	Saved value of burned SRM propellant	lbs			Local	
t	Time from liftoff	sec			Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
tglim	Not used	sec			Local	
xm	Total vehicle mass	lbs			Local	
z	Names of mission designate				Local	

```
      SUBROUTINE SUMARY
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      COMMON/CCC/GRAPH,JO,DEMAND

      LOGICAL DEMAND,GRAPH,LBM,JUMP

      CHARACTER*42 NAME(67),NTRA(2)

      CHARACTER*6 NSTR(2),NBK(4),NBK1(4),NBK2(4),NBK3(4),NBK4(4),NBK5(4)
     *,NFOR(18),Z(5),NB,NSTOP,NAK1,NAK2,NAK,MISION,MISSON

      character*20 na,c(10)

      CHARACTER*60 HEAD

      REAL*8 LBMTNK

      DIMENSION NORD(67),NUMB(67),B(160),PRCTET(3),PRCTOR(3)

      COMMON/AGEN1/TIME(2,16),TAUT(15),TAUW(15),TZERO,TLIFT,TTILT,TMINH,
     *DTZ,TQ,TL,XMAUG,TNE(6,15),ENGDAT(8,15),S(15),WD(15),WJET(15),
     *HEAD,PRINT(15),STEP(15),HNOM(15),HMAX(15),PROP(6),TBEGR,TENDR,
     *CHRDOT,CHIRO,FAZ,BO(3,2),CBAXIL,TCHIR,VRCUT,FPRFAC,CHVEL

      COMMON/AGEN3/ITHR,IWD,IXR,JUM ,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISSON,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)

      COMMON/ALAT/ALAT,ALONG,ALTLS,NTABLE

      COMMON/ASTOR/A(4,67)

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM

      COMMON/LONOFF/LBM

      COMMON/OMSRCS/FOMS(15),FRCS(15),WDOMS(15),WDRCS(15),WDDUMP(15)

      EQUIVALENCE(T,B(1)),(XM,B(6)),(DWMPS,B(67)),(DWSRM,B(68)),
     *(DWLBM,B(69)),(DWOMS,B(70)),(DWRCS,B(71)),(DWFIX,B(72)),
     *(DWDUMP,B(129)),(MISION,NB)

      DATA NUMB/3,3,3,3,3,3,4,3,3,4,4,3,3,3,2,2,2,3,3,3,3,3,3,3,3,3,4,
     *3,3,0,3,3,3,3,3,1,0,1,1,1,0,0,1,1,1,1,1,1,0,0,1,1,1,1,0,1,1,1,1,1,
     *1,1,1,1,1,1/

      DATA NORD/1,1,1,1,1,1,6,1,1,6,2,1,1,1,5,1,1,1,1,1,1,5,1,1,1,1,3,4,
     *1,5,0,1,1,1,1,1,5,0,1,5,1,0,0,1,1,5,1,1,1,0,0,1,1,1,5,0,1,1,5,1,1,
     *1,5,1,1,1,5/

      DATA C/
     *     'IGNITION          ','LIFT-OFF          ',
```



```
     *     'LBM IGITION       ','SEPARATION        ',
     *     'LBM SEPARATION    ','RLTS/AOA          ',
     *     'INJECTION         ','ENDPRT            ',
     *     'INTERMEDIATE      ','                  '/
      DATA Z/'NOM   ','AOA   ','RTLS  ','POST N','POST A'/
      DATA PRCTET/.487,.450,.799/
      DATA PRCTOR/.513,.550,.201/

      DATA NAME( 1)/'PAYLOAD                                        '/
      DATA NAME( 2)/'OMS KIT                                        '/
      DATA NAME( 3)/'MISSION KIT                                    '/
      DATA NAME( 4)/'PERSONNEL                                      '/
      DATA NAME( 5)/'ORBITER - EMPTY                                '/
      DATA NAME( 6)/'         - NONPROPULSIVE CONSUMABLES           '/
      DATA NAME( 7)/'         - USABLE RCS (RCS LOADED)             '/
      DATA NAME( 8)/'         - RCS RESERVES                        '/
      DATA NAME( 9)/'         - RCS RESIDUALS                       '/
      DATA NAME(10)/'         - USABLE OMS (OMS LOADED)             '/
      DATA NAME(11)/'         - OMS PROPELLANT (DEORBIT)            '/
      DATA NAME(12)/'         - OMS RESERVES                        '/
      DATA NAME(13)/'         - OMS RESIDUALS                       '/
      DATA NAME(14)/'SSME EMPTY                                     '/
      DATA NAME(15)/'ORBITER AT ORBIT INSERTION                     '/
      DATA NAME(16)/'MECO OMS MANEUVER - OMS PROPELLANT             '/
      DATA NAME(17)/'                  - RCS PROPELLANT             '/
      DATA NAME(18)/'TRAPPED MPS PROPELLANT - ORBITER               '/
      DATA NAME(19)/'                       - SSME                  '/
      DATA NAME(20)/'FLIGHT PERFORMANCE RESERVE - ORBITER           '/
      DATA NAME(21)/'                           - SSME              '/
      DATA NAME(22)/'ORBITER AT OMS IGNITION                        '/
      DATA NAME(23)/'RCS PROPELLANT-ET SEPARATION                   '/
      DATA NAME(24)/'EXTERNAL TANK - EMPTY                          '/
      DATA NAME(25)/'              - RESIDUALS                      '/
      DATA NAME(26)/'FLIGHT PERFORMANCE RESERVE - ET                '/
      DATA NAME(27)/'FUEL BIAS                                      '/
      DATA NAME(28)/'EXCESS MPS PROPELLANT (MPS PROP LOADED)        '/
      DATA NAME(29)/'EXPELLED MPS PROPELLANT                        '/
      DATA NAME(30)/'INJECTED WEIGHT AT MECO                        '/
      DATA NAME(31)/'PROPELLANT BURNED-AOA/RTLS TO MECO             '/
      DATA NAME(32)/'  MPS                                          '/
      DATA NAME(33)/'  OMS                                          '/
      DATA NAME(34)/'  RCS                                          '/
      DATA NAME(35)/'OMS PROPELLANT DUMPED (RCS BURN)               '/
      DATA NAME(36)/'EXPELLED MPS PROPELLANT                        '/
      DATA NAME(37)/'WEIGHT AT AOA/RTLS                             '/
      DATA NAME(38)/'MPS PROPELLANT BURNED-LBM SEPARATION           '/
      DATA NAME(39)/'  TO RTLS/AOA                                  '/
      DATA NAME(40)/'WEIGHT AT LBM SEPARATION                       '/
      DATA NAME(41)/'LBM INERT WEIGHT                               '/
      DATA NAME(42)/'PROPELLANT BURNED-SRM SEPARATION TO            '/
      DATA NAME(43)/'  LBM SEPARATION                               '/
      DATA NAME(44)/'  MPS                                          '/
      DATA NAME(45)/'  LBM                                          '/
      DATA NAME(46)/'ET+ORBITER AFTER SRM SEPARATION                '/
      DATA NAME(47)/'SRM INERT WEIGHT                               '/
      DATA NAME(48)/'SRM RESIDUAL PROPELLANT                        '/
```

```
      DATA NAME(49)/'INERTS CONSUMED                                '/
      DATA NAME(50)/'PROPELLANT BURNED-LBM IGNITION                 '/
      DATA NAME(51)/'  TO SRM SEPARATION                            '/
      DATA NAME(52)/'  MPS                                          '/
      DATA NAME(53)/'  LBM                                          '/
      DATA NAME(54)/'  SRM                                          '/
      DATA NAME(55)/'LBM IGNITION (T+5 SEC)                         '/
      DATA NAME(56)/'PROPELLANT BURNED-LIFTOFF TO LBM IGNITION      '/
      DATA NAME(57)/'  MPS                                          '/
      DATA NAME(58)/'  SRM                                          '/
      DATA NAME(59)/'LIFTOFF (T/W=1.0)                              '/
      DATA NAME(60)/'SRB PROPELLANTS                                '/
      DATA NAME(61)/'SRB INERTS                                     '/
      DATA NAME(62)/'MPS PROPELLANT                                 '/
      DATA NAME(63)/'SRB IGNITION COMMAND (T-0)                     '/
      DATA NAME(64)/'MPS THRUST BUILDUP + SRM IGNITION DELAY        '/
      DATA NAME(65)/'OVERFILL,DRAINBACK,AND BOILOFF                 '/
      DATA NAME(66)/'NONPROPULSIVE CONSUMABLES                      '/
      DATA NAME(67)/'PRELAUNCH WEIGHT (T-5 MIN)                     '/

      DATA NTRA( 1)/'MPS PROPELLANT BURNED-SRM SEPARATION           '/
      DATA NTRA( 2)/'PROPELLANT BURNED - LIFTOFF                    '/
      DATA NSTR/'(1X,A4','2     '/,NSTOP/')     '/
      DATA NAK1/',F8.0 '/,NAK2/',8X    '/
      DATA NBK1/',14X  ','      ','      ','      '/
      DATA NBK2/',2H (','F5.1,5','H F/S)','2X   '/
      DATA NBK3/',3X,6H','MARGIN','5X   ','      '/
      DATA NBK4/',1X,2H','( ,F8.','0,1H),','2X    '/
      DATA NBK5/',6X,F8','.0    ','      ','      '/

      GZERO=9.80665
      PTKG=2.2046223
      ETPRWT=PROP(1)
      LBMTNK=PROP(3)
      OMSTNK=PROP(4)
      RCSTNK=PROP(5)

      N=1

   28 L1=1
      L2=37

      TGLIM=10000.

      M=0
      OMS=0.
      RCS=0.

      K=1
      L=1

      OMSDUM=0.
      JUMP=.FALSE.

      IF(N.EQ.1) REWIND 9
```



```
   31 READ(9,END=21,ERR=210) NA,B,NB

      IF(NA.EQ.C(8)) GO TO 31

      DWMPS=DWMPS+DWFIX

      IF(M.eq.0) then

         DO 29 I=1,5
         IF(MISION.EQ.Z(I)) GO TO 30
   29    CONTINUE
   30    M=I

      endif

      IF(M.GT.3.AND.NA.EQ.C(7)) then

         M=M-3
         A(M,16)=DWOMS-SVOMS
         A(M,17)=DWRCS-SVRCS

         go to 46

      endif

      IF(M.GT.3) GO TO 31

      IF(NA.EQ.C(10)) GO TO 31

      DO 25 K=1,7
      IF(NA.EQ.C(K)) GO TO 32
   25 CONTINUE

      GO TO 31

   32 if(k.eq.1) then

         STIME=T

         IF(ABS(T).gt..1E-03) then

            JUMP=.TRUE.
            L=JUM
            SVMPS=DWMPS
            SVOMS=DWOMS
            SVRCS=DWRCS

         else

            A(1,63)=XM

         endif

         GO TO 31

      else if(k.eq.2) then
```

```
      A(1,59)=XM
      A(1,62)=DWMPS-A(1,64)-A(1,65)
      A(1,60)=DWSRM-A(1,61)
      A(1,67)=A(1,63)+A(1,64)+A(1,65)+A(1,66)
      SVMPS=DWMPS
      SVSRM=DWSRM

      GO TO 31

    else if(k.eq.3) then

      A(1,55)=XM
      A(1,57)=DWMPS-SVMPS
      A(1,58)=DWSRM-SVSRM
      SVMPS=DWMPS
      SVSRM=DWSRM
      LBM=.TRUE.

      GO TO 31

    else if(k.eq.4) then

      A(1,46)=XM
      A(1,52)=DWMPS-SVMPS
      A(1,53)=DWLBM
      A(1,54)=DWSRM-SVSRM-A(1,49)
      SVMPS=DWMPS
      SVLBM=DWLBM
      SVSRM=DWSRM

      GO TO 31

    else if(k.eq.5) then

      A(1,40)=XM
      A(1,44)=DWMPS-SVMPS
      A(1,45)=DWLBM-SVLBM
      SVMPS=DWMPS
      SVLBM=DWLBM

      GO TO 31

    else if(k.eq.6) then

      A(1,37)=XM
      A(1,39)=DWMPS-SVMPS
      SVMPS=DWMPS

      GO TO 31

    else if(k.eq.7) then

      A(M,30)=XM
      A(M,32)=DWMPS-SVMPS
      A(M,33)=DWOMS-DWDUMP
```



```
      A(M,34)=DWRCS
      A(M,35)=DWDUMP
      SVOMS=DWOMS
      SVRCS=DWRCS
      FPR=XM*(1.-EXP(-CHVEL*FPRFAC/(GZERO*FUNISP(1.,2))))

      A(M,26)=PRCTET(M)*FPR
      A(M,20)=PRCTOR(M)*FPR-A(M,21)
      A(M,28)=ETPRWT-(A(M,18)+A(M,19)+A(M,20)+A(M,21)+A(M,25)+
     *          A(M,26)+A(M,27)+A(M,29)+A(M,32)+A(M,36)+A(1,39)+
     *          A(1,44)+ A(1,52)+A(1,57)+A(1,62)+A(1,64)+A(1,65))
      A(M,22)=A(M,30)-(A(M,23)+A(M,24)+A(M,25)+A(M,26)+A(M,27)+
     *          A(M,28)+A(M,29))

    endif

 46 A(M,15)=A(M,22)-(A(M,16)+A(M,17)+A(M,18)+A(M,19)+A(M,20)+A(M,21))
    A(M,10)=OMSTNK-(A(M,11)+A(M,12)+A(M,13)+A(M,16)+A(M,33)+A(M,35))
    A(M,7)=RCSTNK-(A(M,8)+A(M,9)+A(M,17)+A(M,23)+A(M,34))
    A(M,1)=A(M,15)

    DO 47 I=2,14
 47 A(M,1)=A(M,1)-A(M,I)

    A(4,7)=A(3,7)
    A(3,7)=A(2,7)
    A(2,7)=RCSTNK
    A(4,10)=A(3,10)
    A(3,10)=A(2,10)
    A(2,10)=OMSTNK
    A(4,11)=A(3,11)
    A(3,11)=A(2,11)
    A(2,11)=0.
    A(4,28)=A(3,28)
    A(3,28)=A(2,28)*1.06

  1 WRITE(JO,101) HEAD

    WRITE(21,101)HEAD

101 FORMAT(1H1,36X,A60///49X,7HNOMINAL,17X,3HAOA,19X,4HRTLS//)

    NFOR(1)=NSTR(1)
    NFOR(2)=NSTR(2)
    NFOR(18)=NSTOP

    DO 5 K=L1,L2

    IF(.not.LBM) then

      IF(K.eq.38) NAME(38)=NTRA(1)

      IF(K.GE.40.AND.K.LE.45) GO TO 5

      IF(K.eq.50) NAME(50)=NTRA(2)
```

```
      IF(K.EQ.53.OR.(K.GE.55.AND.K.LE.58)) GO TO 5

     endif

     NAK=NAK1
     IF(NORD(K).EQ.0.OR.NORD(K).EQ.5) NAK=NAK2

     NFOR(3)=NAK
     NFOR(8)=NAK

     NFOR(13)=NAK
     IF(K.EQ.17.OR.K.GT.36) NFOR(13)=NAK2

     IF(K.GT.36) NFOR(8)=NAK2

     DO 2 I=1,4

     NBK(I)=NBK1(I)
     IF(NORD(K).EQ.2) NBK(I)=NBK2(I)
     IF(NORD(K).EQ.3) NBK(I)=NBK3(I)
     IF(NORD(K).EQ.4) NBK(I)=NBK4(I)
     IF(NORD(K).EQ.5) NBK(I)=NBK5(I)
     IF(NORD(K).EQ.6) NBK(I)=NBK4(I)

   2 CONTINUE

     DO 3 I=1,4

     J=I+3

     NFOR(J)=NBK(I)
     IF(NORD(K).EQ.3) NFOR(J)=NBK1(I)

     J=I+8

     NFOR(J)=NBK(I)

     IF(K.EQ.9.OR.K.EQ.10.OR.K.GT.36) NFOR(J)=NBK1(I)

     IF(NORD(K).EQ.6) NFOR(J)=NBK1(I)

     J=I+13

     NFOR(J)=NBK(I)
     IF(NORD(K).NE.5.OR.K.GT.36) NFOR(J)=NBK1(I)

   3 CONTINUE

     N=NUMB(K)

     IF(NORD(K).EQ.5.AND.K.NE.L1) WRITE(JO,102)
     IF(NORD(K).EQ.5.AND.K.NE.L1) WRITE(21,102)

     IF(N.ne.0) then

        IF(NORD(K).eq.4) then
```



```
           WRITE(JO,NFOR) NAME(K),A(1,K),ETPRWT,A(2,K),
     *     A(3,K),A(4,K)
           WRITE(21,NFOR) NAME(K),A(1,K),ETPRWT,A(2,K),
     *     A(3,K),A(4,K)

        else

           WRITE(JO,NFOR) NAME(K),(A(J,K),J=1,N)
           WRITE(21,NFOR) NAME(K),(A(J,K),J=1,N)

        endif

     else

        WRITE(JO,NFOR) NAME(K)
        WRITE(21,NFOR) NAME(K)

     endif

     IF(NORD(K).EQ.5) WRITE(JO,102)
     IF(NORD(K).EQ.5) WRITE(21,102)

 102 FORMAT( )

   5 CONTINUE

     IF(L1.eq.1) then

        L1=37
        L2=67

        GO TO 1

     endif

     A(2,7)=A(3,7)
     A(3,7)=A(4,7)
     A(2,10)=A(3,10)
     A(3,10)=A(4,10)
     A(2,11)=A(3,11)
     A(3,11)=A(4,11)
     A(3,28)=A(4,28)

     IF(NTABLE.LE.1.OR.N.EQ.NTABLE) RETURN

     N=N+1

     GO TO 28

  21 BACKSPACE 9

     RETURN

 210 WRITE(*,*) ' ERROR READING TAPE 9'
```

```
      RETURN
      END
```

3.60 Subroutine SUMHLLV

3.60.1 Purpose

This subroutine reads trajectory output file, calculates weights based on events, and prints weight summary table for Strap-on vehicle.

3.60.2 Variable Listing

3.60.3 Subroutines Called:

Name	Function
FUNISP	Calculates specific impulse based on throttle level

3.60.4 Calling Subroutines:

Name	Function
AFORND	Defines parameters at the terminal and at intermediate points on the forward trajectory

3.60.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
a(4,67)	Weight storage array for output	lbs	ainit	output input	Global	astor
b	Input array from output file produced by aprtn				Local	
c	Trajectory events names array				Local	
chvel	Characteristic velocity	m/sec	mastre	input	Global	agen1
dwdump	Amount of RCS dumped propellant	lbs			Local	
dwfix	Amount of propellant burned by fixed MPS engines	lbs			Local	
dwlbm	Amount of LBM propellant burned	lbs			Local	
dwmps	Amount of MPS propellant burned	lbs			Local	
dwoms	Amount of OMS propellant burned	lbs			Local	
dwrcs	Amount of RCS propellant burned	lbs			Local	
dwsrm	Amount of SRM propellant burned	lbs			Local	
etprwt	Total ET propellant weight	lbs			Local	
fpr	Flight performance reserve factor				Local	
fprfac	Flight performance reserve factor		ainit	input	Global	agen1
gzero	Gravitational constant (9.80665 m/sec)	m/sec^	ainit	output input	Global	const
head	Storage array for headings in printout		ainit	input	Global	agen1
jum	Jump start flag		ainit	input	Global	agen3
jump	Jump start flag				Local	
l1	Index				Local	
l2	Index				Local	
lbm	Denotes the simulation of liquid booster module			output	Global	lonoff
lbmtnk	Total LBM propellant weight	lbs			Local	
m	Mission designate (equivalent to nb)				Local	
mision	Mission designate (equivalent to nb)				Local	
misson	Not used				Local	
n	Index				Local	
na	Format used in variable format				Local	
nak	Format used in variable format				Local	
nak1	Format used in variable format				Local	
nak2	Format used in variable format				Local	
name	Name of events in weight output table				Local	

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
nb	Mission designate name from output file				Local	
nbk	Formats for variable output format				Local	
nbk1	Formats for variable output format				Local	
nbk2	Formats for variable output format				Local	
nbk3	Formats for variable output format				Local	
nbk4	Formats for variable output format				Local	
nbk5	Formats for variable output format				Local	
nfor	Variable format parameter				Local	
nord	Index for variable formatting the output				Local	
nstop	Terminal portion of variable format statement				Local	
nstr	Formats for variable output format				Local	
ntable	Flag indicating that output tables will (>0) or will not (=0) be generated			output	Global	alat
numb	Number of parameters to be printed for each line of output				Local	
omstnk	Total OMS propellant weight	lbs			Local	
prctet	Not used				Local	
prctor	Not used				Local	
prop(6)	Propellant weight for each engine type	kg	ainit	input	Global	agen1
ptkg	Conversion from pounds to kilograms	kg/lbs	ainit	output	Global	const
rcstnk	Total RCS propellant weight	lbs			Local	
stime	Saved value of time	sec			Local	
svlbm	Save value of burned LBM propellant	lbs			Local	
svmps	Saved value of burned MPS propellant	lbs			Local	
svsrm	Saved value of burned SRM propellant	lbs			Local	
t	Time from liftoff	sec			Local	
tglim	Not used	sec			Local	
xm	Total vehicle mass	lbs			Local	
z	Names of mission designate				Local	

```
      SUBROUTINE SUMHLLV

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      COMMON/CCC/GRAPH,JO,DEMAND

      LOGICAL DEMAND,GRAPH,LBM,JUMP

      CHARACTER*42 NAME(52)
      CHARACTER*6 NSTR(2),NBK(4),NBK1(4),NBK2(4),NBK3(4),NBK4(4),NBK5(4)
     *,NFOR(18),Z(2),NB,NSTOP,NAK1,NAK2,NAK,MISION,MISSON
      character*20 na,c(10)
      CHARACTER*60 HEAD

      REAL*8 LBMTNK

      DIMENSION NORD(52),NUMB(52),B(160),PRCTET(3),PRCTOR(3)

      COMMON/AGEN1/TIME(2,16),TAUT(15),TAUW(15),TZERO,TLIFT,TTILT,TMINH,
     *DTZ,TQ,TL,XMAUG,TNE(6,15),ENGDAT(8,15),S(15),WD(15),WJET(15),
     *HEAD,PRINT(15),STEP(15),HNOM(15),HMAX(15),PROP(6),TBEGR,TENDR,
     *CHRDOT,CHIRO,FAZ,BO(3,2),CBAXIL,TCHIR,VRCUT,FPRFAC,CHVEL

      COMMON/AGEN3/ITHR,IWD,IXR,JUM ,KAT,KCYTAB,KPAGE,KS1,KWTA(7),LINES,
     *NBGCT(7),NENDCT(7),NMAX,NOEVNT(5),NOWD(15),NVNT,NWVNT,IHEAD,
     *MINH,MPSFLG(15),NSYST(15),ICONSW,IRTLS,IPOLY,KINDB,KRDERB,MISSON,
     *IRTFLG,NROLL,NCOAST,IFACT,NOBASE,LSTGE(15),MSWCH(15)

      COMMON/ALAT/ALAT,ALONG,ALTLS,NTABLE

      COMMON/ASTOR/A(4,67)

      COMMON/CONST/RAD,PI,RE,FLAT,CJ,H,DJ,CMUE,OMEGA,ALT1,ALT2,PSL,
     *GZERO,PTN,PTKG,PSFTM,PFN,PFKG,PSFFM

      COMMON/LONOFF/LBM

      COMMON/OMSRCS/FOMS(15),FRCS(15),WDOMS(15),WDRCS(15),WDDUMP(15)

      EQUIVALENCE(T,B(1)),(XM,B(6)),(DWMPS,B(67)),(DWSRM,B(68)),
     *(DWLBM,B(69)),(DWOMS,B(70)),(DWRCS,B(71)),(dwfix,b(72)),
     *(DWDUMP,B(129)),(MISION,NB)

      DATA NUMB/2,2,2,2,2,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,2,2,2,2,
     *1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1/

      DATA NORD/5,1,1,1,2,6,1,1,5,1,5,1,1,1,1,1,1,5,1,1,1,1,1,4,1,5,1,1,
     *5,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,5,1,1,5,1,1,5/

      DATA C/
     *     'IGNITION          ','LIFT-OFF          ',
     *     'LBM IGNITION      ','SEPARATION        ',
     *     'LBM SEPARATION    ','CORE SEPARATION   ',
     *     'INJECTION         ','ENDPRT            ',
     *     'INTERMEDIATE      ','                  '/
```



```
      DATA Z/'NOM   ','ENGOUT'/

      DATA PRCTET/.487,.450,.799/
      DATA PRCTOR/.513,.550,.201/

      DATA NAME( 1)/'PAYLOAD                                    '/
      DATA NAME( 2)/'UPPER STAGE - JETTISON WEIGHT              '/
      DATA NAME( 3)/'              - INERT STRUCTURE            '/
      DATA NAME( 4)/'              - FLIGHT PERFORMANCE RESERVE '/
      DATA NAME( 5)/'              - DELTA VELOCITY PROPELLANT  '/
      DATA NAME( 6)/'              - EXCESS PROPELLANT(PROP LD) '/
      DATA NAME( 7)/'              - FUEL BIAS                  '/
      DATA NAME( 8)/'              - PROP RESIDUALS/UNUSABLE    '/
      DATA NAME( 9)/'WEIGHT AT ORBIT INSERTION                  '/
      DATA NAME(10)/'UPPER STAGE - PROPELLANT BURNED            '/
      DATA NAME(11)/'WEIGHT AFTER 2ND STAGE SEPARATION          '/
      DATA NAME(12)/'PERFORMANCE MARGIN - PAYLOAD               '/
      DATA NAME(13)/'                   - OTHER                 '/
      DATA NAME(14)/'2ND STAGE - JETTISON WEIGHT                '/
      DATA NAME(15)/'            - STAGE INERT WEIGHT (LESS ENG)'/
      DATA NAME(16)/'            - NONPROPULSIVE CONSUMABLES    '/
      DATA NAME(17)/'            - DRY WEIGHT OF ENGINE(S)      '/
      DATA NAME(18)/'2ND STAGE - TOTAL WEIGHT AT 2ND STAGE SEP  '/
      DATA NAME(19)/'            - PROP RESIDUALS/UNUSABLES     '/
      DATA NAME(20)/'            - FLIGHT PERFORMANCE RESERVE   '/
      DATA NAME(21)/'            - FUEL BIAS                    '/
      DATA NAME(22)/'            - MPS THRUST BUILDUP           '/
      DATA NAME(23)/'            - BOILOFF,OVERFILL,DRAINBACK   '/
      DATA NAME(24)/'EXCESS MPS PROPELLANT (MPS PROP LOADED)    '/
      DATA NAME(25)/'SHROUD WEIGHT SEPARATED AT MECO            '/
      DATA NAME(26)/'INJECTED WEIGHT AT MECO                    '/
      DATA NAME(27)/'SHROUD JETTISON WEIGHT                     '/
      DATA NAME(28)/'PROPELLANT BURNED - SEPARATION TO MECO     '/
      DATA NAME(29)/'WEIGHT AT SEPARATION                       '/
      DATA NAME(30)/'1ST STAGE - JETTISON WEIGHT                '/
      DATA NAME(31)/'            - STAGE INERT WEIGHT (LESS ENG)'/
      DATA NAME(32)/'            - DRY WEIGHT OF ENGINE(S)      '/
      DATA NAME(33)/'            - NONPROPULSIVE CONSUMABLES    '/
      DATA NAME(34)/'            - FUEL BIAS                    '/
      DATA NAME(35)/'            - PROP RESIDUALS/UNUSABLES     '/
      DATA NAME(36)/'STRAP-ON  - JETTISON WEIGHT                '/
      DATA NAME(37)/'            - INERT MOTOR WEIGHT (LESS ENG)'/
      DATA NAME(38)/'            - DRY WEIGHT OF ENGINE(S)      '/
      DATA NAME(39)/'            - NONPROPULSIVE CONSUMABLES    '/
      DATA NAME(40)/'            - FUEL BIAS                    '/
      DATA NAME(41)/'            - PROP RESIDUALS/UNUSABLES     '/
      DATA NAME(42)/'            - INERTS CONSUMED (SRM ONLY)   '/
      DATA NAME(43)/'PROPELLANT BURNED - LIFTOFF TO SEPARATION  '/
      DATA NAME(44)/'              MPS                          '/
      DATA NAME(45)/'              STRAP-ON                     '/
      DATA NAME(46)/'LIFTOFF (T/W=1.0)                          '/
      DATA NAME(47)/'MPS PROPELLANT                             '/
      DATA NAME(48)/'STRAP-ON PROPELLANT                        '/
      DATA NAME(49)/'IGNITION COMMAND (T-0) WEIGHT              '/
      DATA NAME(50)/'MPS       - THRUST BUILDUP                 '/
      DATA NAME(51)/'            - BOILOFF,OVERFILL,DRAINBACK   '/
```

```
      DATA NAME(52)/'PRELAUNCH WEIGHT (T-5 MIN)                '/

      DATA NSTR/'(1X,A4','2     '/,NSTOP/')     '/
      DATA NAK1/',F8.0 '/,NAK2/',8X   '/
      DATA NBK1/',14X  ','      ','      ','      '/
      DATA NBK2/',2H (','F5.1,5','H F/S)','2X   '/
      DATA NBK3/',3X,6H','MARGIN',',5X   ','      '/
      DATA NBK4/',1X,2H','( ,F8.','0,1H),','3X    '/
      DATA NBK5/',6X,F8','.0    ','      ','      '/

      GZERO=9.80665
      PTKG=2.2046223
      ETPRWT=PROP(1)
      LBMTNK=PROP(3)
      OMSTNK=PROP(4)
      RCSTNK=PROP(5)

      N=1

   28 L1=1
      L2=29

      TGLIM=10000.

      M=0
      K=1
      L=1

      kflag6=0
      JUMP=.FALSE.

      IF(N.EQ.1) REWIND 9

   31 READ(9,END=1,ERR=210) NA,B,NB

      IF(NA.EQ.C(8)) GO TO 31

      dwmps=dwmps+dwfix

      IF(M.eq.0) then

         DO I=1,2
            IF(MISION.EQ.Z(I)) GO TO 30
         ENDDO
   30    M=I

      endif

      IF(NA.EQ.C(10)) GO TO 31

      DO  K=1,7
         IF(NA.EQ.C(K)) GO TO 32
      ENDDO

      GO TO 31
```



```
   32 if(k.eq.1) then
         STIME=T
         IF(ABS(T).gt..1E-03) then
            JUMP= .TRUE.
            L=JUM
            SVMPS=DWMPS
            SVOMS=DWOMS
            SVRCS=DWRCS
         else
            A(1,49)=XM
         endif
         go to 31
      else if(k.eq.2) then
         A(1,46)=XM
         A(1,47)=DWMPS
         A(1,48)=DWSRM-A(1,48)
         A(1,52)=A(1,49)+A(1,50)+A(1,51)
         SVMPS=DWMPS
         SVSRM=DWSRM
         GO TO 31
      else if(k.eq.3) then
         A(1,55)=XM
         A(1,57)=DWMPS-SVMPS
         A(1,58)=DWSRM-SVSRM
         SVMPS=DWMPS
         SVSRM=DWSRM
         LBM=.TRUE.
         GO TO 31
      else if(k.eq.4) then
         A(1,29)=XM
         A(1,44)=DWMPS-SVMPS
         A(1,45)=DWSRM-SVSRM-A(1,42)
         a(1,30)=a(1,31)+a(1,32)+a(1,33)+a(1,34)+a(1,35)
         a(1,36)=a(1,37)+a(1,38)+a(1,39)+a(1,40)+a(1,41)
         SVMPS=DWMPS
         SVSRM=DWSRM
         GO TO 31
      else if(k.eq.5) then
         A(1,40)=XM
         A(1,44)=DWMPS-SVMPS
         A(1,45)=DWLBM-SVLBM
         SVMPS=DWMPS
         SVLBM=DWLBM
         GO TO 31
      else if(k.eq.6) then
         A(m,26)=XM
         A(m,28)=DWMPS-SVMPS
         SVMPS=DWMPS
         SVOMS=DWOMS
         kflag6=1
         GO TO 31
      else if(k.eq.7) then
         if(kflag6.eq.0) then
            A(M,26)=XM
            A(M,28)=DWMPS-SVMPS
         endif
```

```
         kflag6=0
         FPR=XM*(1.-EXP(-CHVEL*FPRFAC/(GZERO*FUNISP(1.,2))))
         a(m,24)=etprwt-(a(m,19)+a(m,20)+a(m,21)+a(m,22)+
     *             a(m,23)+a(m,28)+a(1,34)+a(1,35)+a(1,44)+
     *             a(1,47)+a(1,50)+a(1,51))
         a(m,18)=a(m,26)-a(m,25)
         a(m,14)=a(m,15)+a(m,16)+a(m,17)+a(m,19)+a(m,20)+a(m,24)
         a(m,11)=a(m,18)-a(m,12)-a(m,13)-a(m,14)
       endif
       deltav=a(3,5)
       dvisp=a(4,5)
       IF(dvisp.gt.0.) then
          a(m,5)=xm*(1.-exp(-deltav*.3048/(gzero*dvisp)))
       ELSE
          a(m,5)=0.
       ENDIF

       a(m,10)=dwoms-svoms
       a(m,9)=a(m,11)-a(m,10)
       a(m,6)=omstnk-(a(m,4)+a(m,5)+a(m,7)+a(m,8))
       a(m,2)=a(m,3)+a(m,4)+a(m,5)+a(m,6)+a(m,7)+a(m,8)
       a(m,1)=a(m,9)-a(m,2)
       m=0
       go to 31

     1 WRITE(JO,101) HEAD
       WRITE(21,101)HEAD

   101 FORMAT(1H1,////36X,A60///49X,'NOMINAL',14X,'ENGINE OUT'//)

       NFOR(1)=NSTR(1)
       NFOR(2)=NSTR(2)
       NFOR(18)=NSTOP

       DO 5 K=L1,L2

       NAK=NAK1
       IF(NORD(K).EQ.0.OR.NORD(K).EQ.5) NAK=NAK2

       NFOR(3)=NAK
       NFOR(8)=NAK
       NFOR(13)=NAK

       DO I=1,4
          NBK(I)=NBK1(I)
          IF(NORD(K).EQ.2) NBK(I)=NBK2(I)
          IF(NORD(K).EQ.3) NBK(I)=NBK3(I)
          IF(NORD(K).EQ.4) NBK(I)=NBK4(I)
          IF(NORD(K).EQ.5) NBK(I)=NBK5(I)
          IF(NORD(K).EQ.6) NBK(I)=NBK4(I)
       enddo

       DO I=1,4
          J=I+3
          NFOR(J)=NBK(I)
          IF(NORD(K).EQ.3) NFOR(J)=NBK1(I)
```



```
          J=I+8
          NFOR(J)=NBK(I)
          IF(K.GT.29) NFOR(J)=NBK1(I)
          IF(NORD(K).EQ.6) NFOR(J)=NBK1(I)
          J=I+13
          NFOR(J)=NBK(I)
          IF(NORD(K).NE.5.OR.K.GT.24) NFOR(J)=NBK1(I)
       enddo

       N=NUMB(K)

       IF(NORD(K).EQ.5.AND.K.NE.L1) WRITE(JO,102)
       IF(NORD(K).EQ.5.AND.K.NE.L1) WRITE(21,102)

       if(n.ne.0) then
          if(nord(k).eq.2) then
             write(jo,nfor) name(k),a(1,k),deltav,a(2,k),deltav
             write(21,nfor) name(k),a(1,k),deltav,a(2,k),deltav
          else if(nord(k).eq.4) then
             WRITE(JO,NFOR) NAME(K),A(1,K),ETPRWT,A(2,K),a(3,k)
             WRITE(21,NFOR) NAME(K),A(1,K),ETPRWT,A(2,K),a(3,k)
          else if(nord(k).eq.6) then
             write(jo,nfor) name(k),a(1,k),omstnk,a(2,k)
             write(21,nfor) name(k),a(1,k),omstnk,a(2,k)
          else
             WRITE(JO,NFOR) NAME(K),(A(J,K),J=1,N)
             WRITE(21,NFOR) NAME(K),(A(J,K),J=1,N)
          endif
       else
          WRITE(JO,NFOR) NAME(K)
          WRITE(21,NFOR) NAME(K)
       endif

       IF(NORD(K).EQ.5) WRITE(JO,102)
       IF(NORD(K).EQ.5) WRITE(21,102)
     5 CONTINUE

       IF(L1.eq.1) then
          L1=29
          L2=52
          GO TO 1
       endif

   102 FORMAT( )

       IF(NTABLE.LE.1.OR.N.EQ.NTABLE) RETURN

       N=N+1

       GO TO 28

   210 write(*,*) ' error reading tape 9'

       RETURN
       END
```

3.61 Subroutine TERMINATE

3.61.1 Purpose

This subroutine disposes of the print file, if required, and terminates the program.

3.61.2 Variable Listing

3.61.3 Subroutines Called:

None

3.61.4 Calling Subroutines:

Name	Function
AREAIN	Controls user interface

3.61.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
cdisp	Name of disposition of output file				Local	
cmsg	Output message (variable name)				Local	
demand	Logical indicating the operational mode		initial	input	Global	ccc
iopt	Input flag parameter				Local	
jout	Output file index		initial	input	Global	ccc

```
      SUBROUTINE TERMINATE
C  THIS ROUTINE WILL DISPOSE OF THE ALTERNATE PRINT FILE
C  CREATED BY MASTRE
C
C    STEVEN SMITH BCSS 2/25/87
      COMMON / CCC / GRAPH,JOUT,DEMAND
C
      CHARACTER*30 CDISP,CMSG
      LOGICAL GRAPH,DEMAND
      INTEGER JOUT
C      THE PRINT FILE IS UNIT 31
C      IF THERE IS A PRINT FILE THEN ASK THE USER IF THEY
C      WANT TO PRINT THE FILE, KEEP THE FILE, PRINT AND KEEP THE
C      FILE OR PRINT AND DELETE THE FILE
      IF ( DEMAND ) THEN
        IF ( JOUT .EQ. 31 ) THEN
 10       PRINT*,'SELECT DISPOSITION FOR PRINT FILE '
          PRINT*,'    '
          PRINT*,' 0 - DO NOT PRINT AND DELETE DISC COPY'
          PRINT*,' 1 - DO NOT PRINT AND SAVE DISC COPY'
          PRINT*,' 2 - PRINT AND DELETE DISC COPY'
          PRINT*,' 3 - PRINT AND SAVE DISC COPY'
          WRITE(6,100)
 100      FORMAT(' MASTRE > ',$)
          IOPT=4
          READ(5,20) IOPT
          IF(IOPT.EQ.0) THEN
                CDISP='DELETE'
                CMSG='DELETED'
          ELSE IF(IOPT.EQ.1) THEN
                CDISP='KEEP'
                CMSG='SAVED'
          ELSE IF(IOPT.EQ.2) THEN
                CDISP='PRINT/DELETE'
                CMSG='PRINTED AND DISC COPY DELETED'
         ELSE IF(IOPT.EQ.3) THEN
         CDISP='PRINT'
                CMSG='PRINTED AND DISC COPY SAVED'
         ELSE
         GO TO 10
         END IF    ! END OPTION CHECK
         CLOSE(UNIT=31,DISPOSE=CDISP)
         PRINT*,' OUTPUT FILE HAS BEEN ',CMSG
       END IF ! END OF PRINT FILE DISPOSITION IN DEMAND MODE
      ELSE
C
C JOB IS BATCH
C
       CLOSE(UNIT=31,DISPOSE='PRINT/DELETE')
      END IF
C
 20   FORMAT(I1)
      STOP 'LEAVING MASTRE'
      END
```

3.62 Subroutine TRANFM

3.62.1 Purpose

This subroutine multiplies a 3x3 matrix times a 3x1 vector to produce a 3x1 vector.

3.62.2 Variable Listing

3.62.3 Subroutines Called:

None

3.62.4 Calling Subroutines:

Name	Function
APRTN	Prints trajectory block output and output files

3.62.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
b	Input vector				Local	
c	Output vector from matrix and vector multiplicaton				Local	
e	Input matrix				Local	

```
      SUBROUTINE TRNSFM(C,E,B)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION C(3),E(3,3),B(3)
      DO 10 I=1,3
      C(I)=0.
      DO 10 J=1,3
   10 C(I)=C(I)+E(I,J)*B(J)
      RETURN
      END
```

3.63 Subroutine WINDIN

3.63.1 Purpose

This subroutine allows the user to choose, in an interactive mode, the option of selecting a mean monthly wind for either an eastern or western test range data model.

3.63.2 Variable Listing

3.63.3 Subroutines Called:

Name	Function
DBANK	Reads wind data from database

3.63.4 Calling Subroutines:

Name	Function
AINIT	Input routine

3.63.5 Fortran Listing

Variable	Variable Definition	Units	Source	I/O	Status	Common Block
altalt	Altitude variable from input wind tables	km			Local	
alttbl(100)	Altitude table for wind tables	m	ainit	output	Global	wind
azwazw	Wind azimuth from wind tables	deg			Local	
azwtbl(100)	Wind azimuth table	rad	ainit	output	Global	initp
demand	Logical indicating the operational mode		initial	input	Global	ccc
iii	Index				Local	
itr	Index for test range				Local	
iwnum	Wind table type index		ainit	input	Global	iwnc
linein	Temporary string array to read unused data from input wind files				Local	
month	Index for month				Local	
nlines	Page count index				Local	
nwind	Number of tabular values in wind parameter tables		ainit	output	Global	wind
rmonth	Name of month used to define wind table used				Local	
tr	Name of launch test range (ETR or WTR)				Local	
uuuu	Wind component from wind tables (wind measured from the north)				Local	
vvvv	Wind component from wind tables (wind measured from the east)				Local	
wswsws	Wind magnitude from wind tables	ft/sec			Local	
wtbl(100)	Wind speed tables	m/sec	ainit	output	Global	wind
wv	Wind speed table	ft/sec			Local	
wvalt	Independent altitude table for wind interpolation	ft			Local	
wvaz	Wind azimuth table	deg			Local	

```
      SUBROUTINE WINDIN
C
C     THIS SUBROUTINE GIVES THE USER THE OPTION OF SELECTING A MEAN
C     MONTHLY WIND FOR EASTERN OR WESTERN TEST RANGE. THE WIND DATA
C     IS STORED IN RECORDS 81-85 OF THE DATA BASE.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      LOGICAL DEMAND,GRAPH
C
      COMMON /CCC   / GRAPH,JO,DEMAND
      COMMON /INITP / MISSON(6),WTJET(15),WDLBS(15),AZWTBL(100),BCD(22)
      COMMON /IWNC  / IWNUM
      COMMON /WIND  / ALTTBL(100),WTBL(100),AZTBL(100),NWIND
C
      DIMENSION WVALT(100),WV(100),WVAZ(100)
C
       REAL*4 ALTALT,UUUU,VVVV,WSWSWS,AZWAZW
      CHARACTER*3 TR(2),RMONTH(12)
      CHARACTER*6 MISSON
      character*12 BCD
      CHARACTER*80 LINEIN
C
      DATA TR /'ETR','WTR'/
C
      DATA RMONTH /'JAN','FEB','MAR','APR','MAY','JUN',
     *             'JUL','AUG','SEP','OCT','NOV','DEC'/
C
      IF(IWNUM.LE.12) THEN
         ITR=1
         MONTH=IWNUM
         NWIND=71
      ELSE
         ITR=2
         MONTH=IWNUM-12
         NWIND=88
         IF(MONTH.EQ.6 .OR. MONTH.EQ.9 .OR. MONTH.EQ.12) NWIND=87
      ENDIF
C
        IF(IWNUM.EQ.0)THEN
                CALL DBANK(WVALT,83,1201,1300)
                CALL DBANK(WVAZ,83,-1301,1400)
                CALL DBANK(WV,83,-1401,1500)
                NWIND=51
                GOTO25
        ENDIF
C
C     SP****NEW CODE FOR MONTHLY ENVELOPING WINDS
C     IWNUM=50
C
      IF(IWNUM.EQ.50)THEN
            OPEN(UNIT=69,FILE='AWOUT.PRT',STATUS='OLD',
     1           SHARED,READONLY)
                DO 6969 III=1,8
                READ(69,'(A80)')LINEIN
 6969           CONTINUE
```



```
                DO 6970 III=1,100
                READ(69,*,ERR=6913,END=6913)ALTALT,UUUU,VVVV,
     1          WSWSWS,AZWAZW
                ALTTBL(III)=DBLE(ALTALT*1000.)
                AZWTBL(III)=DBLE(AZWAZW)
                WTBL(III)=DBLE(WSWSWS*.3048)
 6970           CONTINUE
 6913           NWIND=III-1
       GOTO269
      ENDIF
      GO TO ( 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,
     *        13,14,15,16,17,18,19,20,21,22,23,24),IWNUM
C
  1   CONTINUE                                      !JAN-ETR MEAN WIND
      CALL DBANK(WVALT,80, 301,400)
      CALL DBANK(WVAZ ,82,   1,100)
      CALL DBANK(WV   ,82,-101,200)
      GO TO 25
C
  2   CONTINUE                                      !FEB-ETR MEAN WIND
      CALL DBANK(WVALT,80, 401,500)
      CALL DBANK(WVAZ ,82, 201,300)
      CALL DBANK(WV   ,82,-301,400)
      GO TO 25
C
  3   CONTINUE                                      !MAR-ETR MEAN WIND
      CALL DBANK(WVALT,80, 501,600)
      CALL DBANK(WVAZ ,82, 401,500)
      CALL DBANK(WV   ,82,-501,600)
      GO TO 25
C
  4   CONTINUE                                      !APR-ETR MEAN WIND
      CALL DBANK(WVALT,80, 601,700)
      CALL DBANK(WVAZ ,82, 601,700)
      CALL DBANK(WV   ,82,-701,800)
      GO TO 25
C
  5   CONTINUE                                      !MAY-ETR MEAN WIND
      CALL DBANK(WVALT,80, 701,800)
      CALL DBANK(WVAZ ,82, 801,900)
      CALL DBANK(WV   ,82,-901,1000)
      GO TO 25
C
  6   CONTINUE                                      !JUN-ETR MEAN WIND
      CALL DBANK(WVALT,80, 801,900)
      CALL DBANK(WVAZ ,82, 1001,1100)
      CALL DBANK(WV   ,82,-1101,1200)
      GO TO 25
C
  7   CONTINUE                                      !JUL-ETR MEAN WIND
      CALL DBANK(WVALT,80, 901,1000)
      CALL DBANK(WVAZ ,83,   1,100)
      CALL DBANK(WV   ,83,-101,200)
      GO TO 25
C
  8   CONTINUE                                      !AUG-ETR MEAN WIND
```

```
      CALL DBANK(WVALT,80, 1001,1100)
      CALL DBANK(WVAZ ,83, 201,300)
      CALL DBANK(WV   ,83,-301,400)
      GO TO 25
C
 9    CONTINUE                                  !SEP-ETR MEAN WIND
      CALL DBANK(WVALT,80, 1101,1200)
      CALL DBANK(WVAZ ,83,  401, 500)
      CALL DBANK(WV   ,83, -501, 600)
      GO TO 25
C
 10   CONTINUE                                  !OCT-ETR MEAN WIND
      CALL DBANK(WVALT,80, 1201,1300)
      CALL DBANK(WVAZ ,83,  601, 700)
      CALL DBANK(WV   ,83, -701, 800)
      GO TO 25
C
 11   CONTINUE                                  !NOV-ETR MEAN WIND
      CALL DBANK(WVALT,80, 1301,1400)
      CALL DBANK(WVAZ ,83,  801, 900)
      CALL DBANK(WV   ,83, -901,1000)
      GO TO 25
C
 12   CONTINUE                                  !DEC-ETR MEAN WIND
      CALL DBANK(WVALT,80, 1401,1500)
      CALL DBANK(WVAZ ,83, 1001,1100)
      CALL DBANK(WV   ,83,-1101,1200)
      GO TO 25
C
 13   CONTINUE                                  !JAN-WTR MEAN WIND
      CALL DBANK(WVALT,81,  301, 400)
      CALL DBANK(WVAZ ,84,    1, 100)
      CALL DBANK(WV   ,84, -101, 200)
      GO TO 25
C
 14   CONTINUE                                  !FEB-WTR MEAN WIND
      CALL DBANK(WVALT,81,  401, 500)
      CALL DBANK(WVAZ ,84,  201, 300)
      CALL DBANK(WV   ,84, -301, 400)
      GO TO 25
C
 15   CONTINUE                                  !MAR-WTR MEAN WIND
      CALL DBANK(WVALT,81,  501, 600)
      CALL DBANK(WVAZ ,84,  401, 500)
      CALL DBANK(WV   ,84, -501, 600)
      GO TO 25
C
 16   CONTINUE                                  !APR-WTR MEAN WIND
      CALL DBANK(WVALT,81,  601, 700)
      CALL DBANK(WVAZ ,84,  601, 700)
      CALL DBANK(WV   ,84, -701, 800)
      GO TO 25
C
 17   CONTINUE                                 !MAY-WTR MEAN WIND
      CALL DBANK(WVALT,81,  701, 800)
      CALL DBANK(WVAZ ,84,  801, 900)
```



```
      CALL DBANK(WV   ,84, -901,1000)
      GO TO 25
C
 18   CONTINUE                                  !JUN-WTR MEAN WIND
      CALL DBANK(WVALT,81,  801, 900)
      CALL DBANK(WVAZ ,84, 1001,1100)
      CALL DBANK(WV   ,84,-1101,1200)
      GO TO 25
C
 19   CONTINUE                                  !JUL-WTR MEAN WIND
      CALL DBANK(WVALT,81,  901,1000)
      CALL DBANK(WVAZ ,85,    1, 100)
      CALL DBANK(WV   ,85, -101, 200)
      GO TO 25
C
 20   CONTINUE                                  !AUG-WTR MEAN WIND
      CALL DBANK(WVALT,81,1001,1100)
      CALL DBANK(WVAZ ,85, 201, 300)
      CALL DBANK(WV   ,85,-301, 400)
      GO TO 25
C
 21   CONTINUE                                  !SEP-WTR MEAN WIND
      CALL DBANK(WVALT,81,1101,1200)
      CALL DBANK(WVAZ ,85, 401, 500)
      CALL DBANK(WV   ,85,-501, 600)
      GO TO 25
C
 22   CONTINUE                                  !OCT-WTR MEAN WIND
      CALL DBANK(WVALT,81,1201,1300)
      CALL DBANK(WVAZ ,85, 601, 700)
      CALL DBANK(WV   ,85,-701, 800)
      GO TO 25
C
 23   CONTINUE                                 !NOV-WTR MEAN WIND
      CALL DBANK(WVALT,81,1301,1400)
      CALL DBANK(WVAZ ,85, 801, 900)
      CALL DBANK(WV   ,85,-901,1000)
      GO TO 25
C
 24   CONTINUE                                 !DEC-WTR MEAN WIND
      CALL DBANK(WVALT,81,1401,1500)
      CALL DBANK(WVAZ ,85,1001,1100)
      CALL DBANK(WV   ,85,-1101,1200)
C
 25   CONTINUE
C
C     CONVERT WIND DATA TO METRIC UNITS
C
      DO 27 I=1,100
      ALTTBL(I) = WVALT(I)*.3048D0
      WTBL(I) = WV(I)*.3048D0
        IF(WVAZ(I).LT.0.)WVAZ(I)=-WVAZ(I)
 27   AZWTBL(I) = WVAZ(I)
269   CONTINUE
C
      NLINES=4
```

```
      DO 30 I=1,100
      IF(NLINES.EQ.4) THEN
        IF(IWNUM.LT.0)THEN
                WRITE(JO,*)'NO WINDS'
                WRITE(JO,28)
        ELSEIF(IWNUM.EQ.50)THEN
         WRITE(JO,*)'ANNUAL ENVELOPING WINDS'
        ELSEIF(IWNUM.LT.1)THEN
        WRITE(JO,*)'MEAN ANNUAL WINDS'
        WRITE(JO,28)
        ELSE
         WRITE(JO,26) IWNUM,TR(ITR),RMONTH(MONTH)
         WRITE(JO,28)
                ENDIF
      ENDIF
      WRITE(JO,29) ALTTBL(I),WVALT(I),AZWTBL(I),WTBL(I),WV(I)
      NLINES=NLINES+1
      IF(NLINES.EQ.55) NLINES=4
 30   CONTINUE
C
      IF(DEMAND)THEN
        IF(IWNUM.LT.1)THEN
                WRITE(6,*)'MEAN ANNUAL WINDS'
        ELSEIF(IWNUM.EQ.50)THEN
             WRITE(6,*)'ANNUAL ENVELOPING WINDS'
        ELSE
         WRITE(6,31) IWNUM,TR(ITR),RMONTH(MONTH)
        ENDIF
        ENDIF
C
 26   FORMAT(1H1,8X,' WIND NUMBER IS ',I2,4X,A3,'-',A3,' MEAN WIND ')
 28   FORMAT(2X,'ALTITUDE',4X,'ALTITUDE',5X,'DIRECTION',5X,'SPEED',
     *       4X,'SPEED'/
     *       5X,'(M)',8X,'(FT)',9X,'(DEG)',7X,'(M/S)',4X,'(FT/S)'/)
 29   FORMAT(1X,F10.2,2X,F10.2,5X,F7.2,5X,F7.2,2X,F7.2)
 31   FORMAT(10X,' WIND NUMBER IS ',I2,4X,A3,'-',A3,' MEAN WIND ')
C
      RETURN
C
      END
```

Appendix A
Root-Sum-Square (RSS) Prgram listing

```
      Program RSS_Main

      implicit double precision (a-h,o-z)

      parameter (nd=28,kvar=25,kp=kvar-1,krec=4*(kvar+1),kd=300)

      character*6 name
      character*30 file_name
      character*25 var_name
      character*4 type
      character*1 cont,tab

      common/store/store(kd,nd,kp,2)
      common/ia_nom/ia_nom
      common/rssp/rssp(kd,kp),rssn(kd,kp),rss1p(kd,kp),rss1n(kd,kp),
     * rss2p(kd,kp),rss2n(kd,kp)
      common/time/time(kd)
      common/il/il(7),iu(7)
      common/iftype/iftype(kp)
      common/var_name/var_name(kp)
      common/name/name(kp)
      common/ict/ict(nd,2)

      dimension var(kvar),mvar(kp)

      data il/1,2,7,15,21,25,27/
      data iu/1,6,14,20,24,26,28/

      data iftype/4,1,1,2,2,3,4,4,3,4,1,1,3,4,3,4,3,1,3,3,3,3,1,1,1/

c     il and iu define the lower and upper limits of the storage
c     allocation for each dispersion classification

c     Each trajectory data is stored within kd records on this file

      data name/'  acc ','  alt ','radius',' velr ',' veli ','   Q  ',
     *  ' gamr ',' gami ','  AHI ',' hrate','dwnrng','crsrng',' azii ',
     *  ' latd ',' long ',' lati ',' longi','weight',' alpha',' beta ',
     *  ' chip ',' chiy ','liqpro','solpro'/

      data var_name/
     * 'axial acceleration (g''s)'     ,'altitude (ft)',
     * 'radius (ft)'                   ,'relative velocity (ft/s)',
     * 'inertial velocity (ft/s)'      ,'dynamic pressure (psf)',
     * 'rel flght path angle (deg)'    ,'int flght path angle (deg)',
     * 'stagnation heating (BTU)'      ,'stag heating rate (BTU/s)',
     * 'down range (ft)'               ,'cross range (ft)',
     * 'inertial azimuth (deg)'        ,'geodetic latitude (deg)',
     * 'longitude (deg)'               ,'impact latitude (deg)',
     * 'impact longitude (deg)'        ,'vehicle weight (lbs)',
     * 'angle of attack (deg)'         ,'sideslip angle (deg)',
     * 'pitch att command (deg)'       ,'yaw att command (deg)',
     * 'liq prop consumed (lbs)'       ,'solid prop consumed (lbs)'/

      ifirst=0
```



```
10    write(*,*) '************************************************'
      write(*,*) ' Select from the following menu:'
      write(*,*) '     1 - Generate rss output tables'
      write(*,*) '     2 - Generate plot file of rss & composites'
      write(*,*) '     3 - Generate plot file of trajectory'
      write(*,*) '         parameters from the dispersion file'
      write(*,*) '     4 - Stop'
      write(*,*) '************************************************'

      read(*,*) nopt
      if(nopt.lt.1.or.nopt.gt.4) then
         write(*,*) ' input out of range, try again'
         go to 10
      endif

      if(nopt.eq.4) stop

      if(ifirst.eq.0) then

         write(*,*) ' Input name of dispersion run file'
         read(*,'(a30)') file_name

         open(unit=8,access='direct',file=file_name,recl=krec,
     *   status='old')

      endif

      if(nopt.eq.1.or.nopt.eq.2) then

         call rss(ifirst)

         if(nopt.eq.1) call table
         if(nopt.eq.2) call plot_rss

      endif

      if(nopt.eq.3) call plot_parm(ifirst)

      ifirst=1

      go to 10

      end


c ***************************************************************************

      Subroutine rss(ifirst)

      implicit double precision (a-h,o-z)

      parameter (nd=28,kvar=25,kp=kvar-1,kd=300)

      character*30 file_name
      character*4 type
      character*1 cont,tab
```

```
      common/store/store(kd,nd,kp,2)
      common/ia_nom/ia_nom
      common/rssp/rssp(kd,kp),rssn(kd,kp),rss1p(kd,kp),rss1n(kd,kp),
     * rss2p(kd,kp),rss2n(kd,kp)
      common/time/time(kd)
      common/il/il(7),iu(7)
      common/ict/ict(nd,2)

      dimension var(kvar)

      data nrmax/15900/

c     nrmax is the maximum number of records on the storage file

c     File 8 contains all dispersion runs and the first read is coded
c     to define the type of dispersion

      ipl=0

      if(ifirst.eq.1) go to 40

      nr=1
      istop=0
      ia_nom=0

10    read(8,rec=nr,err=30) var,ivar

      if(ivar.eq.0) then

         nr=nr+kd
         if(nr.ge.nrmax) go to 40
         go to 10

      endif

c ***************************************************************************
c     The input parameter ivar contains four digits and divided into
c     three parts; itype is the 1st digit, jtype is composed of the
c     2nd and 3rd digits, and ktype is the 4th digit.

c ***************************************************************************
c     itype - dispersion type

c             0___ - nominal
c             1___ - propulsion
c             2___ - aero/environment
c             3___ - mass property
c             4___ - gn&c
c             5___ - composite
c ***************************************************************************
c     jtype - subgrouping                        Record
c                                               Location
c                                               pos     neg
c ***************************************************************************
c     Nominal
```



```
c     0010 - nominal                            00001
c ***************************************************************************
c     Propulsion

c     101_ - STME Vacuum thrust                 00301   00601
c     102_ - STME Vacuum Isp                    00901   01201
c     103_ - STME mixture ratio                 01501   01801
c     104_ - STME thrust misalign (pitch)       02101   02401
c     105_ - STME thrust misalign (yaw)         02701   03001
c     106_ - ASRM web action time               03301   03601
c     107_ - ASRM vacuum Isp                    03901   04201
c     108_ - ASRM propellant loading            04501   04801
c     109_ - ASRM inert weight                  05101   05401
c     110_ - ASRM thrust misalign (pitch)       05701   06001
c     111_ - ASRM thrust misalign (yaw)         06301   06601
c     112_ - ASRM thrust imbalance              06901   07201
c     113_ - ASRM thrust uncertainty            07501   07801
c ***************************************************************************
c     Aero/Environment

c     201_ - Forebody axial force               08101   08401
c     202_ - Baseforce                          08701   09001
c     203_ - Other Aerodynamic coeff            09301   09601
c     204_ - wind profiles (head & tail)        09901   10201
c     205_ - winf profiles (rt & lt cross)      10501   10801
c     206_ - atmospheric density                11101   11401
c ***************************************************************************
c     Mass Properties

c     301_ - Core inert weights                 11701   12001
c     302_ - Propulsion mod inert wt            12301   12601
c     303_ - Core propellant weight             12901   13201
c     304_ - Center of Gravity                  13501   13801
c ***************************************************************************
c     GN&C

c     401_ - Booster Pitch steering program     14101   14401
c     402_ - Booster Yaw steering program       14701   15001
c ***************************************************************************
c     Composite Trajectory info

c     5010 - Composite (positive)               15301
c     5020 - Composite (negative)               15601
c     5100 - RSS data                           18001

c ***************************************************************************
c     ktype - dispersion type

c      ___0 - positive dispersion
c      ___1 - negative dispersion
c ***************************************************************************

      write(type,'(i4)') ivar
      read(type,'(i1,i2,i1)') itype,jtype,ktype

      if(itype.gt.1) itype=itype+1
```

```
        jtype=il(itype+1)+jtype-1
        k=ktype+1

        ict(jtype,k)=1

        ia=0

        go to 25

20      nr=nr+1

        read(8,rec=nr,err=30) var,ivar

        if(ivar.eq.9999) then

           istop=1
           nr=kd*(nr/kd+1)+1
           if(jtype.eq.1) ia_nom=ia
           if(ia.lt.ia_nom) then
              do j=ia+1,ia_nom
                 do i=1,kp
                    store(j,jtype,i,k)=store(j,1,i,1)
                 enddo
              enddo
           endif
           if(nr.ge.nrmax) go to 40
           go to 10

        endif

25      ia=ia+1

        if(ia.gt.kd) then

           nr=kd*(nr/kd+1)+1
           ia=ia-1
           if(nr.ge.nrmax) go to 40
           go to 10

        endif

        if(jtype.eq.1) time(ia)=var(1)

        if(abs(time(ia)-var(1)).lt..5) then

           do i=1,kp
              store(ia,jtype,i,k)=var(i+1)
           enddo

c ***************************************************************************
c       var(01) - time from liftoff (sec)
c       var(02) - axial acceleration (g's)
c       var(03) - altitude (ft)
c       var(04) - radius (ft)
c       var(05) - relative velocity (ft/sec)
```



```
c       var(06) - inertial velocity (ft/sec)
c       var(07) - dynamic pressure (psf)
c       var(08) - relative flight path angle (deg)
c       var(09) - inertial flight path angle (deg)
c       var(10) - stagnation heating (BTU)
c       var(11) - stagnation heating rate (BTU/sec)
c       var(12) - down range (ft)
c       var(13) - cross range (ft)
c       var(14) - inertial azimuth
c       var(15) - geodetic latitude (deg)
c       var(16) - longitude (deg)
c       var(17) - impact latitude (deg)
c       var(18) - impact longitude (deg)
c       var(19) - vehicle weight (lbs)
c       var(20) - angle of attack (deg)
c       var(21) - sideslip angle (deg)
c       var(22) - pitch attitude command (deg)
c       var(23) - yaw attitude command (deg)
c       var(24) - liquid propellant consumed (lbs)
c       var(25) - solid propellant consumed (lbs)
c ***************************************************************************

           go to 20

        endif

        do i=1,kp
           store(ia,jtype,i,k)=store(ia,1,i,1)
        enddo

        go to 25

c ***************************************************************************

30      write(*,*) ' error in reading dispersion file, record ',irec
        stop

c ***************************************************************************

40      ipl=ipl+kd

        ndm1=26

        do 50 i=1,ia_nom
        do 50 k=1,kp

        rssp(i,k)=0.
        rssn(i,k)=0.
        rss1p(i,k)=0.
        rss1n(i,k)=0.
        rss2p(i,k)=0.
        rss2n(i,k)=0.

        do 50 j=2,ndm1

        ds1=0.
```

```
        ds2=0.

        if(ict(j,1).eq.1) then

c ***************************************************************************
c       current dispersion is a positive increment
c ***************************************************************************

          ds1=store(i,j,k,1)-store(i,1,k,1)
          ds12=ds1*ds1

          if(ds1.ge.0.) then
             if(j.lt.18.or.j.gt.20) then
                rss1p(i,k)=rss1p(i,k)+ds12
             else
                rss2p(i,k)=rss2p(i,k)+ds12
             endif
          else
             if(j.lt.18.or.j.gt.20) then
                rss1n(i,k)=rss1n(i,k)+ds12
             else
                rss2n(i,k)=rss2n(i,k)+ds12
             endif
          endif

        endif

        if(ict(j,2).eq.1) then

c ***************************************************************************
c       current dispersion is a negative increment
c ***************************************************************************

          ds2=store(i,j,k,2)-store(i,1,k,1)
          ds22=ds2*ds2

          if(ds1*ds2.le.0.) then

c ***************************************************************************
c       if the dispersions are different signs, then the rss is
c       computed normally
c ***************************************************************************

             if(ds2.ge.0.) then
                if(j.lt.18.or.j.gt.20) then
                   rss1p(i,k)=rss1p(i,k)+ds22
                else
                   rss2p(i,k)=rss2p(i,k)+ds22
                endif
             else
                if(j.lt.18.or.j.gt.20) then
                   rss1n(i,k)=rss1n(i,k)+ds22
                else
                   rss2n(i,k)=rss2n(i,k)+ds22
                endif
             endif
```



```
          else

c ***************************************************************************
c       if the dispersions are of the same sign, the largest
c       value is chosen for the rss
c ***************************************************************************

             if(ds1.ge.0.) then
                if(j.lt.18.or.j.gt.20) then
                   rss1p(i,k)=rss1p(i,k)-ds12
                else
                   rss2p(i,k)=rss2p(i,k)-ds12
                endif
             else
                if(j.lt.18.or.j.gt.20) then
                   rss1n(i,k)=rss1n(i,k)-ds12
                else
                   rss2n(i,k)=rss2n(i,k)-ds12
                endif
             endif

             ds3=max(ds12,ds22)

             if(ds1.ge.0.) then
                if(j.lt.18.or.j.gt.20) then
                   rss1p(i,k)=rss1p(i,k)+ds3
                else
                   rss2p(i,k)=rss2p(i,k)+ds3
                endif
             else
                if(j.lt.18.or.j.gt.20) then
                   rss1n(i,k)=rss1n(i,k)+ds3
                else
                   rss2n(i,k)=rss2n(i,k)+ds3
                endif
             endif

          endif

        endif

50      continue

        do i=1,ia_nom
           do k=1,kp
              rss1p(i,k)=sqrt(rss1p(i,k))
              rss1n(i,k)=sqrt(rss1n(i,k))
              rss2p(i,k)=sqrt(rss2p(i,k))
              rss2n(i,k)=sqrt(rss2n(i,k))

              rssp(i,k)=rss1p(i,k)+rss2p(i,k)
              rssn(i,k)=rss1n(i,k)+rss2n(i,k)
           enddo
        enddo
```

```
c ***************************************************************************
c       current rss values are written to portion of storage file
c       for reporting and establishing composite trajectory
c ***************************************************************************

        lvar=5100

        lr=18001
        lr1=lr+kd

        do i=1,ia_nom
           lr=lr+1
           lr1=lr1+1

           write(8,rec=lr) time(i),(rssp(i,j),j=1,kp),lvar
           write(8,rec=lr1) time(i),(rssn(i,j),j=1,kp),lvar
        enddo

        if(ipl.lt.kd.and.istop.eq.0) go to 10

        lr=lr+1
        lr1=lr1+1
        lvar=9999

        write(8,rec=lr) var,lvar
        write(8,rec=lr1) var,lvar

        return
        end

c ***************************************************************************

        Subroutine table

        implicit double precision (a-h,o-z)

        parameter (nd=28,kvar=25,kp=kvar-1,kd=300)

        character*6 it(2),name
        character*16 jt(6)
        character*30 file_name
        character*25 var_name
        character*4 type
        character*1 cont,tab
        character*36 namer1,namer2,namer3
        character*8 ftype1(4),ftype2(4),form(17),name1(8,6)
     *  ,name2(8,6),name3(8,6)

        common/store/store(kd,nd,kp,2)
        common/ia_nom/ia_nom
        common/rssp/rssp(kd,kp),rssn(kd,kp),rss1p(kd,kp),rss1n(kd,kp),
     *  rss2p(kd,kp),rss2n(kd,kp)
        common/time/time(kd)
        common/il/il(7),iu(7)
        common/iftype/iftype(kp)
        common/var_name/var_name(kp)
```



```
        common/name/name(kp)
        common/ict/ict(nd,2)

        dimension var(kvar),disp(10),mvar(kp)

        data ftype1/',f10.0',',f10.1',',f10.2',',f10.3'/
        data ftype2/',f8.0',',f8.1',',f8.2',',f8.3'/

        data it/' Pos  ',' Neg  '/

        data jt/'Propulsion','Propulsion','Aero/Environment',
     *  'Mass Properties','G, N, & C','Composite'/

        data (name1(i,1),i=1,5) /'   STME ','   STME ','   STME ',
     *     ' STME Pt',' STME Yw '/
        data (name2(i,1),i=1,5)/'  Vacuum','  Vacuum',' Mixture',
     *     '  thrust','  thrust'/
        data (name3(i,1),i=1,5)/'  thrust','    Isp ','  ratio ',
     *     ' misalgn',' misalgn'/

        data (name1(i,2),i=1,8)/ '   ASRM ','   ASRM ','   ASRM ',
     *     '   ASRM ',' ASRM Pt',' ASRM Yw','   ASRM ','   ASRM '/
        data (name2(i,2),i=1,8)/ ' web act','  Vacuum','    prop',
     *     '    inert','  thrust','  thrust','  thrust','  thrust'/
        data (name3(i,2),i=1,8)/ '    time ','     Isp ',' loading',
     *     '  weight',' misalgn',' misalgn','   imbal','  uncert'/

        data (name1(i,3),i=1,6)/ 'Forebody','  Base  ','  Other ',
     *     '  wind  ','  wind  ','    atmos'/
        data (name2(i,3),i=1,6)/ '  axial ',' Force  ','   Aero ',
     *     '  Prof  ','  Prof  ',' density'/
        data (name3(i,3),i=1,6)/ '  Force ','        ','  coeff ',
     *     ' hd & tl',' rt&lt c','        '/

        data (name1(i,4),i=1,4)/ '  Core  ','  Prop  ','  Core  ',
     *     ' Center '/
        data (name2(i,4),i=1,4)/ ' inert  ',' inert  ','  prop  ',
     *     '    of   '/
        data (name3(i,4),i=1,4)/ '   wgt   ','   wgt   ','   wgt   ',
     *     ' gravity'/

        data (name1(i,5),i=1,2)/ ' Booster',' Booster'/
        data (name2(i,5),i=1,2)/ '  Pitch ','   Yaw  '/
        data (name3(i,5),i=1,2)/ ' program',' program'/

        data (name1(i,6),i=1,6)/ '    pos  ','    neg  ',' system ',
     *     ' system ','  envir ',' envir '/
        data (name2(i,6),i=1,6)/ '   comp ','   comp ','    pos  ',
     *     '    neg  ','    pos  ','   neg  '/
        data (name3(i,6),i=1,6)/ '        ','        ','    rss  ',
     *     '    rss  ','    rss  ','   rss  '/

        data namer1/'   RSS      RSS      Nominal    Nominal'/
        data namer2/'   Pos      Neg      + Pos      - Neg  '/
        data namer3/'                      RSS        RSS   '/
```

```
c ***********************************************************************
c       write optional table output
c ***********************************************************************

        write(*,*) ' Input name of output table file'
        read(*,'(a30)') file_name

        open(unit=20,file=file_name,status='new')

        do i=1,12
           i1=i+12

           write(*,'(i3,3h - ,a,t40,i3,3h - ,a)') i,var_name(i),
     *     i1,var_name(i1)
        enddo

        write(*,*) ' Do you wish to output all variable tables ?'
        write(*,*) ' (y/n) [n]'
        read(*,'(a1)') cont

        if(cont.eq.'Y'.or.cont.eq.'y') then

           nvar=24
           do i=1,24
              mvar(i)=i
           enddo
        else

           write(*,*) ' How many variables will be selected ?'
           read(*,*) nvar

           write(*,*)
     *     ' Input the number of the variables from above table'
           read(*,*) (mvar(i),i=1,nvar)

        endif
        form(1)='(1x,f5.1'
        do i=2,16
           form(i)='        ,'
        enddo
        form(17)=')'

        do 80 l=2,7

        ii=il(l)
        jj=iu(l)
        lm1=l-1

        do 80 jk=1,nvar

        j=mvar(jk)

        form(2)=ftype1(iftype(j))
```



```
        kl=2
        if(l.eq.7) kl=1
        do 80 k=1,kl

        lines=0

        do 80 m=1,ia_nom

        nn=0
        kk=2

        do 70 n=ii,jj

        nn=nn+1

        disp(nn)=0.
        if(ict(n,k).ne.0) disp(nn)=store(m,n,j,k)-store(m,1,j,1)

        kk=kk+1
        form(kk)=ftype2(iftype(j))

70      continue

        if(l.eq.7) then
           do ll=1,4
              kk=kk+1
              form(kk)=ftype2(iftype(j))
           enddo
        endif

        form(kk+1)=ftype2(iftype(j))
        form(kk+2)=form(kk+1)
        form(kk+3)=ftype1(iftype(j))
        form(kk+4)=form(kk+3)

        lines=lines+1

        if(lines.eq.1.or.lines.gt.50) then

           write(20,300) jt(lm1),name(j),it(k)
           write(20,315)
           if(l.lt.7) then
              write(20,305) (name1(i,l-1),i=1,nn),namer1
              write(20,310) (name2(i,l-1),i=1,nn),namer2
              write(20,305) (name3(i,l-1),i=1,nn),namer3
           else
              write(20,305) (name1(i,l-1),i=1,6),namer1
              write(20,310) (name2(i,l-1),i=1,6),namer2
              write(20,305) (name3(i,l-1),i=1,6),namer3
           endif
           write(20,315)
           lines=1

        endif

        tp=store(m,1,j,1)+rssp(m,j)
```

```
        tn=store(m,1,j,1)-rssn(m,j)

        if(l.lt.7) then

           write(20,form) time(m),store(m,1,j,1),(disp(i),i=1,nn),
     *     rssp(m,j),rssn(m,j),tp,tn

        else

           write(20,form) time(m),store(m,1,j,1),(disp(i),i=1,nn),
     *     rss1p(m,j),rss1n(m,j),rss2p(m,j),rss2n(m,j),rssp(m,j),
     *     rssn(m,j),tp,tn

        endif


80      continue

        close(20)

        return

300     format(1h1/60x,a/5x,2a6)
305     format(16x,10(a))
310     format(2x,'time',3x,'nom val',10(a))
315     format(1x,65(2h *))

        end

c ***************************************************************************

        Subroutine plot_rss

        implicit double precision (a-h,o-z)

        parameter (nd=28,kvar=25,kp=kvar-1,kd=300)

        character*6 it(2),name,plot_name(5,kp)
        character*16 jt(6)
        character*30 file_name
        character*25 var_name
        character*4 type
        character*1 cont,tab
        character*12 ftype3(4),form2(26)

        common/store/store(kd,nd,kp,2)
        common/ia_nom/ia_nom
        common/rssp/rssp(kd,kp),rssn(kd,kp),rss1p(kd,kp),rss1n(kd,kp),
     *  rss2p(kd,kp),rss2n(kd,kp)
        common/time/time(kd)
        common/il/il(7),iu(7)
        common/iftype/iftype(kp)
        common/var_name/var_name(kp)
        common/name/name(kp)

        dimension mvar(kp),plot(5,kp)
```



```
        data ftype3/',5(f10.0,a1)',',5(f10.1,a1)',',5(f10.2,a1)',
     *  ',5(f10.3,a1)'/

c ***************************************************************************
c       write optional rss plot output (for Cricket Graph)
c ***************************************************************************

        tab=char(9)

        write(*,*) ' Input name of plot file'
        read(*,'(a30)') file_name

        open(unit=30,file=file_name,status='new',recl=1000)

        do i=1,12
           i1=i+12

           write(*,'(i3,3h - ,a,t40,i3,3h - ,a)') i,var_name(i),
     *     i1,var_name(i1)

        enddo

        write(*,*) ' Do you wish to output all variables ?'
        write(*,*) ' (y/n) [n]'
        read(*,'(a1)') cont

        if(cont.eq.'Y'.or.cont.eq.'y') then

           nvar=kp
           do i=1,kp
              mvar(i)=i
           enddo

        else

           write(*,*) ' How many variables will be selected ?'
           read(*,*) nvar

           write(*,*)
     *     ' Input the number of the variables from above table'
           read(*,*) (mvar(i),i=1,nvar)

        endif

        form2(1)='(1x,f5.1,a1'
        do i=2,25
           form2(i)='             ,
        enddo
        form2(26)=')'

        do m=1,nvar
           j=mvar(m)

           plot_name(1,m)=name(j)
           plot_name(2,m)='+ RSS'
```

```
         plot_name(3,m)='- RSS'
         plot_name(4,m)='+ COMP'
         plot_name(5,m)='- COMP'
      enddo

      write(30,'(a1,a,a1,24(a,a1,a,a1,a,a1,a,a1,a,a1))') '*',
     *  'time',tab,((plot_name(i,j),tab,i=1,5),j=1,nvar)

      do i=1,ia_nom
         kk=1

         do m=1,nvar
            j=mvar(m)

            plot(1,m)=store(i,1,j,1)
            plot(2,m)=store(i,1,j,1)+rssp(i,j)
            plot(3,m)=store(i,1,j,1)-rssn(i,j)
            plot(4,m)=store(i,27,j,1)
            plot(5,m)=store(i,28,j,1)

            kk=kk+1
            form2(kk)=ftype3(iftype(j))
         enddo

         write(30,form2) time(i),tab,((plot(j,k),tab,j=1,5),
     *     k=1,nvar)

      enddo

      close(30)

      return
      end

c ***************************************************************************

      Subroutine plot_parm(ifirst)

      implicit double precision (a-h,o-z)

      parameter (nd=28,kvar=25,kp=kvar-1,kd=300)

      character*6 name
      character*30 file_name,disp_names(26)
      character*4 type
      character*1 cont,tab
      character*25 var_name
      character*12 form3(30),ftype4(4),ftype5(4)
      character*15 plot_name(2,kvar),dname(27)

      common/store/store(kd,nd,kp,2)
      common/ia_nom/ia_nom
      common/rssp/rssp(kd,kp),rssn(kd,kp),rss1p(kd,kp),rss1n(kd,kp),
     *  rss2p(kd,kp),rss2n(kd,kp)
      common/time/time(kd)
      common/name/name(kp)
```



```
      common/il/il(7),iu(7)
      common/var_name/var_name(kp)
      common/iftype/iftype(kp)

      dimension var(kvar),mvar(kvar),plot(2,kvar)

      data ftype4/',f10.0,a1',',f10.1,a1',',f10.2,a1',',f10.3,a1'/
      data ftype5/',2(f10.0,a1)',',2(f10.1,a1)',',2(f10.2,a1)',
     *  ',2(f10.3,a1)'/

      data nrmax/15900/

      data disp_names/
     *  'STME Vacuum thrust',          'STME Vacuum Isp',
     *  'STME mixture ratio',          'STME thrust misalign (pitch)',
     *  'STME thrust misalign (yaw)',  'ASRM web action time',
     *  'ASRM vacuum Isp',             'ASRM propellant loading',
     *  'ASRM inert weight',           'ASRM thrust misalign (pitch)',
     *  'ASRM thrust misalign (yaw)',  'ASRM thrust imbalance',
     *  'ASRM thrust uncertainty',     'Forebody axial force',
     *  'Base force',                  'Other Aerodynamic coef',
     *  'Wind profiles (head & tail)', 'Wind profiles (rt & lt)',
     *  'Atmospheric density',         'Core inert weights',
     *  'Propulsion mod inert wt',     'Core propellant weight',
     *  'Center of Gravity',           'Booster Pitch steering',
     *  'Booster Yaw steering',        ' '/

      data dname/'STME Vac thr ','STME Vac Isp ','STME Mix R   ',
     *           'STME Pit mis ','STME Yaw mis ','ASRM web act ',
     *           'ASRM Vac Isp ','ASRM prop ld ','ASRM iner wgt',
     *           'ASRM Pit mis ','ASRM Yaw mis ','ASRM thr imb ',
     *           'ASRM thr unc ','axial force  ','base force   ',
     *           'Other aero   ','Wind (H & T) ','Wind (R & L) ',
     *           'Atmos density','Core inert wt','Prop inert wt',
     *           'Core prop wt ','Cent of Grav ','Boost pitch  ',
     *           'Boost yaw    ','comp (pos)   ','comp (neg)   '/

      if(ifirst.eq.1) go to 40

      nr=1
      istop=0
      ia_nom=0

10    read(8,rec=nr,err=30) var,ivar

      if(ivar.eq.0) then

         nr=nr+kd
         if(nr.ge.nrmax) go to 40
         go to 10

      endif

      write(type,'(i4)') ivar
      read(type,'(i1,i2,i1)') itype,jtype,ktype
```

```
      if(itype.gt.1) itype=itype+1

      jtype=i1(itype+1)+jtype-1
      k=ktype+1

      ia=0

      go to 25

20    nr=nr+1

      read(8,rec=nr,err=30) var,ivar

      if(ivar.eq.9999) then

         istop=1
         nr=kd*(nr/kd+1)+1
         if(jtype.eq.1) ia_nom=ia
         if(ia.lt.ia_nom) then
            do j=ia+1,ia_nom
               do i=1,kp
                  store(j,jtype,i,k)=store(j,1,i,1)
               enddo
            enddo
         endif
         if(nr.ge.nrmax) go to 40
         go to 10

      endif

25    ia=ia+1

      if(ia.gt.kd) then

         nr=kd*(nr/kd+1)+1
         ia=ia-1
         if(nr.ge.nrmax) go to 40
         go to 10

      endif

      if(jtype.eq.1) time(ia)=var(1)

      if(abs(time(ia)-var(1)).lt..5) then

         do i=1,kp
            store(ia,jtype,1,k)=var(i+1)
         enddo

         go to 20

      endif

      do i=1,kp
         store(ia,jtype,i,k)=store(ia,1,i,1)
      enddo
```



```
      go to 25

c *********************************************************************

30    write(*,*) ' error in reading dispersion file, record ',irec
      stop

c *********************************************************************

40    continue

      tab=char(9)

      write(*,*) ' Input name of plot file'
      read(*,'(a30)') file_name

      open(unit=40,file=file_name,status='new',recl=1000)

      do i=1,12
         i1=i+12

         write(*,'(i3,3h - ,a,t40,i3,3h - ,a)') i,var_name(i),
     *   i1,var_name(i1)

      enddo

      write(*,*)
     * ' Select the number of one parameter from above list'
      read(*,*) npar

      do i=1,13
         i1=i+13

         write(*,'(i3,3h - ,a,t40,i3,3h - ,a)') i,disp_names(i),
     *   i1,disp_names(i1)

      enddo

      write(*,*) ' Do you wish to output all dispersion cases ?'
      write(*,*) ' (y/n) [n]'
      read(*,'(a1)') cont

      if(cont.eq.'Y'.or.cont.eq.'y') then

         nvar=25

         do i=1,25
            mvar(i)=i
         enddo

      else

         write(*,*) ' How many dispersion cases will be selected ?'
         read(*,*) nvar
```

```
      write(*,*)
     *  ' Input the number of the dispersion cases from above table'
        read(*,*) (mvar(i),i=1,nvar)

      endif

      form3(1)='(1x,f5.1,a1'
      do i=2,29
         form3(i)='                ,
      enddo
      form3(30)=')'

      do m=1,nvar
         j=mvar(m)

         plot_name(1,m)='+ '//dname(j)
         plot_name(2,m)='- '//dname(j)
      enddo

      write(40,'(a1,4(a,a1),25(a,a1,a,a1))') '*','time',tab,
     *  name(npar),tab,dname(26),tab,dname(27),tab,
     *  ((plot_name(i,j),tab,i=1,2),j=1,nvar)

      do i=1,ia_nom
         kk=1

         do k=1,3
            kk=kk+1
            form3(kk)=ftype4(iftype(npar))
         enddo

         do m=1,nvar
            j=mvar(m)+1

            plot(1,m)=store(i,j,npar,1)
            plot(2,m)=store(i,j,npar,2)

            kk=kk+1
            form3(kk)=ftype5(iftype(npar))
         enddo

         write(40,form3) time(i),tab,store(i,1,npar,1),tab,
     *     store(i,27,npar,1),tab,store(i,28,npar,1),tab,
     *     ((plot(j,k),tab,j=1,2),k=1,nvar)

      enddo

      close(40)

      return
      end
```

Appendix B
Description of MASTRE User Friendly Interface (UFI) Program

The following appendix presents a listing of the names and definitions of the subroutines used in the MASTRE User Friendly Interface (UFI) Program . These subroutines use the VAX system Screen Management Guidelines (SMG) routines to provide a screen interface with the user.

B.1 Generic Program Flow

Most subroutines in the MASTRE User Friendly Interface (UFI) follow either of two logical flows: one associated with displaying a menu and allowing the user to request a given item and two, a display of menus and parameters where the user views and is allowed to modify the data. The first flow displays the menu and the user determines, by use of the cursor keys, his selection and presses the return key. By pressing the return key, the routine associated with the selection is called by the residing routine. The user interface is the keyboard up and down cursor arrows keys and the return key.

The second flow is more complicated since it utilizes many of the keyboard features and allows the user to modify the input data items. The data parameters associated with the particular screen are first converted to character strings so that the parameters can be displayed on the screen. These parameters are displayed as either data tables or shown next to the parameter's definition. The user can then use the up, down, left, and right cursor keys; the find key; the next and previous screen keys; the insert and remove keys; and the select and return keys to review and/or modify the data. Whenever the data is modified, the user presses the select key and types the new value. The new value is read as a character string, checked for errors in numerical format, and displayed on the screen. When the user finishes with a screen, he presses the return key and the data on the screen is converted from character strings to numerical data.

The following subsections provide the subroutine names and a short description of the non-system and SMG system routines.

B.2 Non-system Subroutine Descriptions

The following alphabetic list describes the non-system subroutines used in the MASTRE User Friendly Interface (UFI) Program. The subroutine "Ufi" is the main program and the program is executed by the command "Run Ufi".

Subroutine	Description
Additional_aero	Additional aerodynamic parameters routine. Displays the parameters for the additional aerodynamic data and lets the user update the data.
Aero_data	Displays the submenu for the "review/modify data" from the general aerodynamic data menu and calls related subroutines. (linear_aero_data, base_force_ data, elevon_ data, and viscous_data).
Aerodynamic_data	General aerodynamic data routine. Displays the main menu for the general aerodynamic data and calls major aerodynamic subroutines (aero_ data, additional_aero, and dispersion_aero).
Att_tables	Calls the min-H data table data conversion and display routines (minh_head, minh_data, and minh_to_num) and allows the user to modify the data.
Attitude_control	General attitude control routine. Displays the menu for the attitude control options and calls the user selected routines (premin and min_att).
Backtonum	Converts character strings to numerical data for engine number to thrust event correlation routine.

Base_data	Converts base force numerical data to character strings or display on the screen.
Base_force_data	Base force data routine. Displays the base force data submenu, calls the conversion and display routines (base_force_head, base_data, and base_force_to_num), and allows the user to modify the data.
Base_force_head	Outputs the headings for the base force data displays.
Base_force_to_num	Coverts the character strings to base force numerical data.
Branch	Branch trajectory option input routine. Displays the "Branch Traj Variables" menu, converts the numerical data to character string, allows the user to modify the data, and converts the character string to numerical data.
Buildcom	Build command file routine. Displays the screen for the build new command file option and calls the "pregunta" subroutine to display questions for the user to answer as the command file is built.
Cls	Clear screen routine.
Coast	Coast parameter input routine. Displays the "Coast Phase" menu, converts the numerical data to character string, allows the user to modify the data, and converts the character string to numerical data.
Com_files	Command file generation and modification routine. Displays the command file submenu, calls the

	"buildcom" routine to build a new command file, displays the screens to modify the command files, and allows the user to modify the command file.
Comfiles	Calls system routines to define files with a given extension in a desired directory and displays these file names on the command file screens.
Constants	General constants routine. Displays the "Constants" menu and calls the user requested routines (earth_constants, integration_constants, and optimization_constants).
Digits	Calculates the number of digits to the right and to the left of the decimal point of a number.
Disp_aero_data	Converts the dispersion aerodynamic numerical data to character strings for display on the screen.
Disp_aero_head	Displays the headings for the dispersion aerodynamic data on the screen.
Disp_aero_to_num	Converts the character strings to dispersion aerodynamic numerical data.
Disp_file	Displays the "File Options" menu and allows the user to create a new dispersion output file and/or reinitialize an existing dispersion file.
Disp_parameters	Additional dispersion parameters input routine. Converts the numerical data to character strings, displays the "Additional Dispersion Inputs" menu, allows the user to modify the data, and converts the character strings to numerical data.

Dispersion_data	Aerodynamic dispersion data routine. Calls the aerodynamic dispersion related data routines (disp_aero_head, disp_aero_data, and disp_aero_to_num) and allows the user to modify the data.
Dispersions	Dispersion input routine. Displays the dispersion menus and allows the user to select the dispersion parameter and the direction of the dispersion. Calls the "disp_file" routine to create or reinitialize a dispersion output file and the "disp_parameters" to define additional dispersion parameters.
Dof	Directory of files routine. Calls system routine to provide a directory of files based on the extension and displays these filename on the screen.
Driver	Plotting interface routine. Displays the plot general menu and calls the user specified routines (plot_data, set_up, mdata, and pplot).
Earth_constants	Earth constants input routine. Displays the "Constants" menu and the parameters. Since this is a "Read-only" menu, the user cannot modify the data.
Elev_data	Converts the elevon numerical data to character strings.
Elevon_data	Elevon data routine. Calls the elevon data routines (elevon_head, elev_data, and elevon_to_num) and allows the user to modify the data.
Elevon_head	Outputs the headings for the elevon data on the screen.

Elevon_to_num	Converts character strings to elevon numerical data.
Environmental_data	General environmental data routine. Displays the "Environmental Data" menu, calls the "wind_data" routine if wind data parameters are required, displays the "Atmosphere Models" menu and allows the user to select one of the menu items.
Files	General file retrieval routine. Calls the routine to define the files based on the directory and extension name, displays these file names as a matrix on the screen, and allows the user to select the desired file name. The calling routine then uses the file name to read the file with the appropriate format.
Fnroll	Rolls a number between an upper and lower bounds by a given direction. Used for window scrolling in multiple directions.
Getfiles	Calls system routines to define file names with certain directory and extension.
Getvars	Displays the choice of parameter names that can be used to define the variables to be plotted.
Gpd	General propulsion data routine. Displays the general propulsion data submenu and calls user selected routines (ind_engine_data, propellant, thne, and theng).
Gpddata	Converts the numerical data to character strings for the individual engine data.

Gpd_heading	Outputs the headings on the screen for the individual engine data tables and calls the "gdp_data" subroutine.
Igint	Finds the greatest integer for an input real number.
Ind_engine_data	Individual engine data routine. Calls the conversion and display routines (gpdheading and r_to_num) and allows the user to modify the data.
Inputsmenu	Main menu display routine. Displays the main menu and calls the user requested routines (launch_and_initial_conditions, thrust_event_ related_data, stage_related_data, propulsion_data, aerodynamic_data, attitude_ control, mass_ properties, environmental_data, constants, operational_options, optimization_data, output_ options, dispersions, com_files, and plott).
Integration_constants	Integration constants input routine. Converts the numerical data to character strings, displays the "Integration Constants" menu and associated parameters, allows the user to modify the data, and converts the character strings to numerical data.
Intermed	Intermediate constraints input routine. Displays the intermediate constraint menu and allows the user to select from the constraint menu codes, define the values of the constraints, and indicate the number of the thrust event where the constraints will be enforced.
Inttostr	Generic conversion from integer to character string routine.

Isp_data_store	Converts numerical data to character strings for specific impulse input data.
Isp_head	Outputs heading to screen for specific impulse input data.
Isp_to_num	Converts character string to numerical data for the specific impulse input data.
Ivarval	Dispersion option routine. Used when building a command file to display the various dispersion options and set the proper option flags.
Jump_start	General jump start option routine. Displays the jump start general menu and calls user specified routines (jump_start_booster and jump_start_ upper_stage).
Jump_start_booster	Booster jump start option routine. Displays the booster jump start parameters, converts the numerical data to character strings for display, allows the user to modify the data, and converts the character strings to numerical data.
Jump_start_upper_stage	Upper stage jump start option routine. Displays the jump start parameters for the upper stage, converts the numerical data to character data, allows the user to modify the data, and converts the character data to numerical data.
Launch_and_initial_ conditions	Launch and initial conditions output routine. Displays the menu and parameters associated with the launch and initial conditions. Converts numerical data to character strings, allows the user to modify the data, and converts the character strings to numerical data.

Lbm	Liquid booster module input routine. Calls the conversion and display routines (lbm_head, lbm_data, and lbm_to_num) and allows the user to modify the data.
Lbm_data	Coverts numerical data to character strings for LBM input data.
Lbm_head	Outputs heading on screen for LBM input data.
Lbm_to_num	Converts character strings to numerical data for LBM input data.
Lin_aero_data	Coverts linear aero numerical data to character strings for display on screen.
Lin_aero_head	Outputs the headers for the linear aerodynamic data.
Lin_aero_to_num	Converts character string data to linear aerodynamic numerical data.
Linear_aero_data	Linear aerodynamic input data routine. Calls the linear aero related routines (lin_aero_head, lin_aero_data, and lin_aero_to_num) and allows the user to modify the data.
Listcodes	Constraint codes output routine. Displays the constraints codes for both the intermediate and terminal constraint menus.
Mass_properties	General mass properties routine. Displays mass properties menu, and, if the tables option is required, calls the conversion and display routines

	(mpheading, mpdata, and mptonum) and allows the user to modify the data.
Minh_att	Min-H attitude input routine. Displays the min-H attitude submenu and calls user selected routines (phase_control and att_tables).
Minh_data	Converts the min-H attitude table numerical data to character strings.
Minh_head	Outputs the headings for the min-H attitude table displays on the screen.
Minh_to_num	Converts the min-H table data character strings to numerical data.
Moment_balance	Moment balance option routine. Displays the moment balance option menu and allows the user to select between moment balance and no moment balance.
Mpdata	Converts numerical data to character strings for the mass properties tabular input option.
Mpheading	Outputs heading on the screen for the mass properties tabular input option.
Mpl_isp_data	MPS specific impulse input routine. Calls the conversion and display routines (isp_head, isp_ data_ store, and isp_to_num) and allows the user to modify the data.
Mps	Main Propulsion System input routine. Displays main propulsion menu and calls user specified routines (tddata, throttable, tailoff, mpl_isp_data, and mps_constants).

Mps_constants	MPS constants input routine. Displays MPS constants menu, converts numerical data to character strings, allows the user to modify the data, and converts the character strings to numerical data.
Mptonum	Converts character strings to numerical data for the mass properties tabular input option.
Nonlinear_aero_data	Not Used.
Omsrcs	OMS/RCS input routine. Calls the conversion and display routines (omsrcs_head, omsrcs_data, and omsrcs_to_num) and allows the user to modify the data.
Omsrcs_data	Converts numerical data to character data for the OMS/RCS input data.
Omsrcs_head	Outputs heading to the screen for the OMS/RCS input data.
Omsrcs_to_num	Converts character strings to numerical data for the OMC/RCS input data.
Operational_options	Operational options input routine. Displays the operational options menu and calls the user selected routines (termtable, termdat, intermed, operating_variables, time_def, jump_start, and special_options)
Operating_variables	Operating variable input routine. Displays the operating variables menu, converts the numerical data to character strings, allows the user to modify

	the data, and converts the character strings to numerical data.
Optimization_constants	Optimization constants input routine. Converts the numerical data to character strings, displays the "Optimization Constants" menu and associated parameters, allows the user to modify the data, and converts the character strings to numerical data.
Optimization_data	Optimization data input routine. Converts the numerical data to character strings, displays the "optimization data" menu and associated parameters, allows the user to modify the data, and converts the character strings to numerical data.
Output_options	General output options input routine. Displays the "Output Summary Table Options" menu and calls the user specified routines (weight_summary, parameter_summary, and prop_summary).
Parameter_summary	Parameter summary output table routine. Displays the message "Will this option be used (y/n)?" and, if the user answers yes, displays the "Program Options" menu. The user selects from the menu and the table parameters and units are displayed.
Pbordr	Finds the intersection of the line drawn between two points with the border of a graph.
Phase_control	Min-H phase input routine. Displays the min-H phase control data, converts the numerical data to character strings, allows the user to modify the data, and converts the character data to numerical data.

Pickvar	Non-scrolling menu selection routine. Allows the user to select an item from a non-scrolling menu.
Plot_data	Data plotting routine. Calls the "getvars" and "plot_var" routines.
Plot_var	Allows the user to select the X and Y variables to be plotted.
Plott	General plotting input routine. Displays the types of plotters that are available to the user and calls the "driver" routine.
Poly_att	Polynomial attitude input routine. Displays the attitude polynomial data, converts the numerical data to character strings, allows the user to modify the data, and converts the character strings to numerical data.
Pplot	Routine which plots data.
Pregunta	Used when building a new command file to prompt the user with appropriate questions, waits for a response, and stores the answer in an answer array.
Premin	Pre-min-H data input routine. Displays the submenu for the pre-min-H attitude options and calls user selected routines (premin_att_control and qalpha_att).
Premin_att_control	Pre-min-H attitude control input routine. Calls the conversion and display routines (premin_head, premin_data, and premin_to_num) and allows the user to modify the data.

Premin_data	Converts the attitude control numerical data to character strings for display.
Premin_head	Outputs the headings for the pre-min H attitude control data on the screen.
Premin_to_num	Converts character strings to attitude control numerical data.
Prop_summary	Propulsion summary output routine. Displays the message "Will this option be used (y/n)?" and, if the user answers yes, converts the numerical data to character strings, displays the "Program Options" menu, allows the user to modify the data and converts the character strings to numerical data.
Propellant	Propellant weights routine. Displays the propellant weight data, converts the numerical data to character strings, allows the user to modify the data, and converts the character strings to numerical data.
Propulsion_data	General propulsion input routine. Displays the general propulsion menu and calls user specified routines (gpd, mps, srm, lbm, and omsrcs).
Qalpha_att	Qalpha attitude input routine. Displays the qalpha data, converts the numerical data to a character string, allows the user to modify the data, and converts the character string to numerical data.
Qmax_constraint	Maximum dynamic pressure constraint parameters input routine. Displays the Qmax options menu and calls the user specified routines (throttable and qmax_variables).

Qmax_variables	Maximum dynamic pressure input variable routine. Displays the Qmax variables menu, converts the numerical data to character strings, allows the user to modify the data, and converts the character strings to numerical data.
R_to_num	Converts character strings to numerical data for the individual engine inputs.
Realtostr	Generic real number to character string conversion routine.
Roundoff	Rounds the input number to a specified digit.
Savecomfile	Save command file routine. Displays "save file" prompt and saves the new or modified command file to the file specified by the user.
Savenl	Save file routine. Displays question to the user "Do you wish to save changes to (extension) file (yes/no)" and if the answer is yes, creates a new version number or new file name based on the user's request.
Scale	Finds an appropriate scale to use for an axis given the maximum and minimum values of data along that axis.
Scroll_pickvar	Scrolling menu selection routine. Allows the user to select an item from a scrolling menu.
Selectfile	Select file routine. User interface routine to allow the user to select from the files menu generated in the "getfiles" subroutine.

Set_up	Scaling routine for plot routine. Allows the user to request either automatic or manual scaling. If manual scaling is requested, a series of menus are generated which request scaling information from the user.
Special_options	General special options input routine. Displays the special options general menu and calls the user specified routines (branch, moment_balance, qmax_constraint, and coast).
Srm	Solid rocket motor input routine. Displays "Solid Rocket Data" menu; calls, if requested by the user, the "files" subroutine to obtain the input SRM data file and reads the SRM input file; and, if requested by the user, calls the "srm_data" subroutine to display and allow the user to modify the data.
Srm_data	Solid rocket motor data input routine. Calls the display and conversion routines (srm_head, srm_place_data, and srm_to_num) and allows the user to modify the data.
Srm_head	Outputs heading to screen for SRM input table data.
Srm_place_data	Converts numerical data to character strings for SRM input table data.
Srm_to_num	Converts character strings to numerical data for SRM input table data.
Stage_data	Converts numerical data to character strings for stage related data.

Stage_related_data	Stage related input data input routine. Displays the stage related data menu, calls the conversion routines (stage_data and stage_to_num), and allows the user to modify the data.
Stage_to_num	Converts character strings to numeric data for stage related data.
Strtoint	Generic character string to integer number conversion routine.
Strtonum	Converts character strings to numerical data for thrust event related data input routine.
Strtoreal	Generic character string to real number conversion routine.
T_to_num	Converts character data to numerical data for the MPS throttle input data.
Tailoff	MPS Thrust tailoff input routine. Calls the coversion and display routines (tailoff_head, tailoff_data, and tailoff_to_num) and allows the user to modify the data.
Tailoff_data	Converts numerical data to character strings for the tailoff data.
Tailoff_head	Outputs headings to the screen for the tailoff data.
Tailoff_to_num	Converts character strings to numerical data for the tailoff data.
Tddata	Thrust event related MPS data input routine. Calls the conversion and display routines (terheading and ter_to_num) and allows the user to modify the data.

Termdat	Terminal constraint input routine. Converts the numerical data to character strings, allows the user to select the constraint mode and the constraint values, and converts the character strings to numerical data.
Termtable	Terminal constraint display routine. Displays portions of the terminal constraint menu.
Teststrg	Character string test routine. Tests to determine if an asterisk, a comma, or a slash has been placed in a numerical field. Characters which are not recognized by the "error" portion of the read statement.
Terdata	Converts numerical data to character strings for the thrust event related MPS data.
Terheading	Outputs heading to the screen for the thrust event related MPS data and calls the conversion routine "terdata".
Ter_to_num	Converts character strings to numerical data for the thrust event related MPS data.
Theng	Thrust event to engine number correlation routine. Calls the coversion and display routines (theng_ head, theng_data, and theng_to_num) and allows the user to modify the data.
Theng_data	Converts the numerical data to character strings for the thrust event to engine number correlation tables.

Theng_head	Outputs the headings for the thrust event to engine number correlation tables to the screen
Theng_to_num	Converts the character strings to numerical data for the thrust event to engine number correlation routine.
Therddata	Converts numerical data to character strings for thrust event related data routine.
Therdheading	Outputs heading to screen for thrust event related data input table.
Thne	Engine number to thrust event correlation routine. Calls the conversion and display routines (thneheading, thnedata, and backtonum) and allows the user to modify the data.
Thnedata	Converts numerical data to character strings for engine number to thrust event correlation routine.
Thneheading	Outputs heading to screen for engine number to thrust event engine correlation routine.
Throttable	MPS throttle table input routine. Calls the conversion and display routines (tt_heading, tt_data, and t_to_num) and allows the user to modify the data.
Thrust_event_related_ data	Thrust event related data input routine. Calls the conversion and display routines (therdheading, therddata, and strtonum) and allows the user to modify the data.
Time_def	Time definition input routine. Displays the "Operating Variables" menu, converts the

	numerical data to character data, allows the user to modify the data, and converts the character data to numerical data.
Tt_data	Converts numerical data to character strings for the MPS throttle input data.
Tt_heading	Outputs headings to the screen for the MPS throttle input data.
Ufi	User friendly interface main program. Displays the initial message "MASTRE USER INTERFACE", calls the "files" routine to select the directory and the general input namelist file, reads the general input namelist file, displays menus to allow the user to select the simulation options, calls the "inputsmenu" routine which displays the main menu, calls the "savenl" routine to save the general input namelist file, and calls the "cls" routine to clear the screen.
Visc_aero_data	Converts the viscous interaction numerical data to character strings for display on the screen.
Visc_aero_head	Outputs the header for the viscous interaction data screens.
Visc_aero_to_num	Converts the characters strings to the viscous interaction numerical data.
Viscous_data	Viscous interaction input data routine. Calls the viscous interaction aerodynamic data related routines (visc_aero_head, visc_aero_data, and visc_aero_to_num) and allows the user to modify the data.

Weight_assign	Converts numerical data to character strings for the weights summary output tables data.
Weight_summary	Weight summary control routine. Displays the "Weight Summary Options" menu; calls, if requested by the user, the "files" routine to select the weights namelist input file, and calls the "weight_tables" routine to display and let the user modify the data.
Weight_tables	Weight tables input routine. Calls the "weight_assign" conversion routine, displays the menu associated with the type of vehicle system being simulated, allows the user to modify the data, and calls the "weights_to_num" conversion routine.
Weights_to_num	Converts character strings to numerical data for the weight summary tables.
Wind_data	Wind data input routine. Displays the "Wind Options" menu and allows the user to select from one of the menu items. Calls the "wind_tables" routine if the user selects the "User Wind Tables" selection.
Wind_tables	User wind table input routine. Calls the conversion and displays routines (winds_head, winds_data, and winds_to_num) and allows the user to modify the data.
Winds_data	Converts numerical data to character strings for wind table input data.
Winds_head	Outputs heading to the screen for the wind table input data.

Winds_to_num	Converts character strings to numerical data for wind table input data.

B.3 Screen Management Guidelines (SMG) Routines

The VAX System Screen Management Guidelines (SMG) routines are used in the MASTRE User Friendly Interface (UFI) Program to provide the screen interface between the computer and the user. The following provides a listing and short definition of the SMG routines that are used in the UFI program. For complete instructions on how to use the SMG routines, the user should consult the VAX SMG user manuals.

Subroutine	Description
SMG$create_pasteboard	Creates storage paste board
SMG$create_menu	Creates menu on screen
SMG$create_virtual_display	Creates window display
SMG$create_virtual_keyboard	Creates interface between keyboard and screen
SMG$delete_menu	Deletes menu from screen
SMG$erase_display	Erases display on window
SMG$label_border	Labels user message to the top of the window
SMG$paste_virtual_display	Paste created window on screen
SMG$put_line	Puts character line at cursor designated by call to SMG$set_cursor_abs
SMG$put_chars	Places characters on screen
SMG$put_chars_highwide	Places large characters on screen

SMG$read_keystroke	Read keystrokes from the keyboard
SMG$read_string	Write and read character string from the keyboard
SMG$scroll_display_area	Defines size of the scrolling area
SMG$select_from_menu	Menu selection interface
SMG$set_cursor_abs	Places cursor to allow reading data
SMG$set_cursor_mode	Turns cursor mode off and on
SMG$unpaste_virtual_display	Remove display from window

Appendix C
Subroutine Definitions for Mass Properties Routines

The following list provides the subroutines names and the definitions of the functions of these subroutines used in the MASTRE Mass Properties Program.

Subroutine	Description
Calc	Determines which file is to be calculated when editing of an input file is complete. Accessed by Cg.
Cg	Main program for mass properties routines. Allows user to access inert payloads, solid motors, throttable engines, non-throttable engines, hybrid motors, and merge module options.
Cgip	Controls which files to edit for the Inert Payloads Module. Called by Cg.
Cls	Clears the screen. Called by Cg.
Cpcg	Calculates the center of gravity location of a circularly perforated grain. It is used in conjunction with the Solid Motor option, and is called from Srmdop.
Daytah	Reads in inert component data from input files. Accessed by Cg, Cgip, and Throttable.
Dof	Reads specified directory to find files with specified extensions. Accessed by Cg, Mergeufi, Mrg, Solids, and Throttable.
Ellipse	Computes the volume and cg location of and ellipsoidal body given the two axial length limits. Accessed by Tecgcal, Necgcal, and Hycgcal.
Enewton	Calculates the liquid height in an ellipsoidal end using a Newton-Raphson iteration method. Accessed by Necgcal and Tecgcal.
Engtyp	Allows user to select which type of engine or motor module to use in the booster or main engine section of the Merge Module. Accessed by Mergeufi.

Fnroll	Dictates next cursor position when extreme sides of edit windows are reached. Called by Cg, Mergeufi, Mrg, Nmlst7, Nmlsts, Pickvar, Sc2d, Stagedrop, and Throttable.
Getinp	Reads in data from input file for Solid Motor Module. Called by Solids.
Hybridread	Reads grain and time-flowrate-mixture ratio data from input file for Hybrid Motor Module. Called by Throttable.
Hybridsave	Writes grain and time-flowrate-mixture ratio data to output file for Hybrid Motor Module. Called by Throttable.
Hycgcal	Calculates the liquid propellant and solid grain cg location versus mass overboard of a hybrid motor using data passed from Newcghy. The data is then passed back to Newcghy.
Inout	Controls insertions and deletions from data tables. Accessed by Sc2d.
Insdel	Adds and deletes lines in the throttable engine, non-throttable engine, and hybrid motor propellant entry screens. Called by Throttable.
Insertdelete	Adds and deletes lines when editing inert component data. Called by Scroll2d.
Merge	Compiles all of the given data file types (inert components, boosters, and main engines) together to form composite type cg's versus mass overboard. Writes outputs to a ".MMM" file. Called by Mergeufi.
Mergeufi	Controls access to the inert payload, booster, and main engine sections of the Merge Module. Stores the returned data from these sections and calls the inert drop routine. Accessed by Cg.
Moreareas	Reads in tabular input for Solid Motor Module. Called by Getinp.
Mrg	Allows specifications of different modules and their relative positions which are used to compose a vehicle in the Merge Module. Module names

	specified in this subroutine are written out at the end of the routine to a ".MMM" file.. Called by Mergeufi.
Necgcal	Calculates the propellant cg location versus mass overboard of a non-throttable engine using data passed from Newcgne. The data is then passed back to Newcgne.
Newcghy	Reads in hybrid engine data from a user specified ".HMI" file, passes the data to Hycgcal for calculation, computes the inert component cg location, and writes the calculated data out to a ".HMM" file.
Newcgip	Reads in inert component data from a user specified ".IPI" file, calculates the cg information from the data, and then writes the calculated data out to a ".IPM" file.
Newcgne	Reads in non-throttable engine data from a user specified ".NEI" file, passes the data to Necgcal for calculation, computes the inert component cg location, and writes the calculated data out to a ".NEM" file.
Newcgsol	Calls Srmdop, transforms the returned data, computes the inert component cg location, and writes the calculated data to a ".SMM" file. Called by Cg.
Newcgte	Reads in throttable engine data from a user specified ".TEI" file, passes the data to Tecgcal for calculation, calculates the inert component cg location, and writes the calculated data out to a ".TEM" file.
Newfile	Controls input setup if a new file is selected in the Inert Payloads Module. Called by Cgip.
Nmlst7	Controls the inputs and outputs for the tabular inputs in the Solid Motor Module. Accessed by Smenu.
Nmlsts	Controls the inputs and outputs, except for the tabular inputs, for the Solid Motor Module. Accessed by Smenu.
Ogive	Computes the volume and cg location of an ogival

	body given the two axial length limits. Accessed by Tecgcal, Necgcal, and Hycgcal.
Onewton	Calculates the liquid height in an ogival end using a Newton-Raphson iteration method. Accessed by Necgcal and Tecgcal.
Pickvar	General routine for selecting an item from a vertical list of available choices. Called by Cg, Cgip, Mergeufi, Saveinp, Smenu, Solids, and Throttable.
Saveinp	Saves data for Solid Motor Module. Called by Smenu.
Saveit	Cycles through file saving options, and saves data for Inert Payloads Module. Called by Daytah, Newfile, and Throttable.
Savethrot	Saves tank and inert component inputs for throttable engines and non-throttable engines. Also saves tank input for hybrid motors. Called by Throttable.
Sc2d	Controls table scrolling for all throttable engine, non-throttable engine, and hybrid motor input. Called by Throttable.
Scroll2d	Controls the input of inert components for all modules. Called by Sipp, Newfile, and Daytah.
Sipp	Establishes link to input solid motor inert components. Called by Smenu.
Smenu	Controls access to the different input screens of the Solid Motor Module. Called by Solids.
Solids	Allows user to select the directory and file to edit for the Solid Motor Module. Called by Cg.
Spline	General cubic spline calculation routine. Accessed by Hycgcal, Merge, Necgcal, Newcgsol, Srmdop, Tabcg, and Tecgcal.
Srt	Sorting routine to make sure that tabular input data is in ascending mass overboard format. Accessed by Cg.
Srmdop	Calculates the solid grain cg versus mass overboard

	for a solid motor. The input data is read from a specified ".SMI" file. The calculated data is passed to Newcgsol. Options are available to output detailed mass properties, an input thrust versus time trace, and an output thrust versus time trace. An output file, DOP.OUT, is always generated. Called by Newcgsol.
Stagedrop	Allows selection of the inerts that are to be dropped at thrust events. Writes selections to a ".MRG" file. Called by Mergeufi.
Starcg	Calculates the cg location of a star grain segment. It is used in conjunction with the Solid Motor option, and is called from Srmdop.
Tabcg	Computes the cg locations versus mass overboard of a tank that has been input in tabular form. Accessed by Necgcal and Tecgcal.
Tecgcal	Calculates the propellant cg location versus mass overboard and the inert component cg location of a throttable engine using data passed from Newcgte. The data is then passed back to Newcgte.
Termpor	Calculates the cg effects of termination ports on a solid grain. Used when a star grain or circularly perforated grain has termination ports specified. Accessed by Cpcg and Starcg.
Throttable	Reads and allows editing of liquid tank data for non-throttable, throttable, and hybrid engines. Also allows editing of hybrid grain data. Accessed by Cg.
Typethree	Prints out the tabular tank data when a tabular tank is selected in a liquid tank input sequence. Accessed by Throttable.