CADAC PROGRAM DOCUMENTATION

VOLUME III

Version 3.1
June 2000

Prepared For:
Air Force Research Laboratory (AFRL/MNGG)
Eglin Air Force Base, Florida 32542

Prepared By:
TYBRIN Corporation
1030 Titan Court
Ft. Walton Beach, FL 32547

DISTRIBUTION UNLIMITED

REVISIONS

June 2000 - Release 3.1 document created. Documents revised to include references and revisions to PC programs KPLOT, PITA, GLOBE, DrawOLE and CADX were updated to reflect modifications and clarifications of their functions. References to the preprocessing utilities DFHEAD, CADIN and INPUT are also updated to reflect their modifications and clarifications of their functions.

Sept 1998 - Release 3.0 document created. Documents revised to include references and revisions to new PC programs KPLOT, PITA, CHKINT and CONVRT. References to programs SWEEP, CONTOUR, ELLIPSE, QPRINT, CADIN, INPUT, MCAP, WinDRAW, DrawOLE and CADX were updated to reflect modifications and clarifications of their functions. The real-time CADAC capability is fully implemented and documented.
KPLOT now includes 2DIM, PITA, GLOBE, WinDRAW and DrawOLE. SWEEP includes CONTOUR, ELLIPSE, QPRINT and MCAP.
Release of CADAC CD-ROM.

July 1997 - Release 2.1 document created. Contour program modified to fix reported problems with the Color box. Program Sweep modified to allow user entered sigma values. Final version of SWEEP 5 methodology fully implemented. CADIN modified to include additional standard (executive level) variables and their definitions.

May 1997 - Release 2.0 document created. Documents revised to include references and revisions to PC programs SWEEP, ELLIPSE, CONTOUR and WinDRAW. References to programs CADAC, CADIN, QPRINT, and to MCAP were updated to reflect modifications and clarifications to their functions. Various editorial changes were implemented to correct reported errors. Programs SASP and SWEEP were combined into one program (SWEEP) to simplify the program series execution. Other programs QPRINT, MCAP, CONTOUR, ELLIPSE and WinDRAW are executed from within the main SWEEP program but also can be executed separately. New SWEEP5 methodology was added to the CADAC executive.

October 1996 - Release 1.1 document created. Documents revised to include references to new PC programs SASP, SWEEP, TCON, and INPUT. References to programs CADAC, CADIN, QPRINT and to file HEAD.ASC were updated to reflect modifications and clarifications to their functions. Various editorial changes were implemented to correct reported errors. Disk file packing list and install procedures have been modified for better use of disk space. The use of ZIP compressed files has been initiated

March 1995 - Release 1.0 document created.

Table of Contents

Table of Contents (Continued)

Section Page

Table of Contents (Continued)

Table of Contents (Continued)

Section Page

Table of Contents (Continued)

Table of Contents (Continued)

Table of Contents (Concluded)

List of Figures

List of Tables

List of Tables (Concluded)

1. INTRODUCTION

Program CADAC, Computer Aided Design of Aerospace Concepts, provides an environment for the development of general purpose, digital computer simulations of time phased dynamic systems. It manages input and output, generates stochastic noise sources, controls all state variable integration and provides post-processing data analysis and display. CADAC has proven its adaptability to many simulation tasks: air-to-ground weapons, air-to-air missiles, ground-to-space and space-to-ground vehicles, and airplanes. The CADAC environment is suitable for 3DOF, 5DOF, and 6DOF simulations. It supports deterministic and Monte Carlo runs. Output can be listed or plotted.

This version (3.1) is a revision of the third release of CADAC for Personal Computers (June 2000). The distribution of CADAC is unrestricted, however, the US Government does not warrant the product nor does it accept any liability for its use.

The documentation consists of four parts, ***Quick Start***, ***User Documentation***, ***Program Documentation*** and ***Real-time CADAC Documentation***. They are written in Microsoft Word 97 and can be found on the ftp site under the filenames QUICK31.DOC, USER31.DOC, CADAC31.DOC and RTCAD30.DOC. For the novice who wants to get a quick overview of CADAC and run the supplied test case ***Quick Start*** is the only document needed to read**.** ***Quick Start*** provides the experienced user with file lists and schematics that serve as quick reference for simulation modifications.

The ***User Documentation*** addresses all capabilities of the CADAC development environment. It should give answers to most questions that come up during the design of a new trajectory simulation. Summary tables examples and matrix utilities provide useful references. The serious CADAC user should read this entire document

The ***Program Documentation*** (this report) provides greater detail for many subjects from building the input and the header files, the integration routine, the generation of stochastic variables, the execution of multi-runs, sweep runs, or Monte Carlo runs to the utilities that aid in building, documenting and analyzing CADAC simulations. It should be used as a reference as more specific questions arise.

The ***Real-time CADAC Documentation*** addresses all the functionality and capabilities involved in the real-time CADAC methodology. It provides detailed procedures in executing CADAC to generate the necessary data files needed to produce a real-time version of the modules. Step-by-step instructions are also detailed for executing the CONVRT program, which create the real-time CADAC code used in a simulator or a test bed.

1.1 CADAC History

The structure of CADAC originated with Litton Industries in the mid 1960's. Since this time, several organizations have adopted this simulation environment: DIMODS (Rockwell), MAVERICK (Hughes), MODIGSI (GBU-15 SPO), ENDOSIM (Army Materiel Command) and several others. CADAC was first employed at USAF/AFMC/ASC during its third generation of development in 1978. Current users include ASC/XR; AFRL/MNG; SWC; NAWC, China Lake, CA; DRA, Farnborough, UK; IABG, Munich, GE and ONERA, France. A large library of subsystem Modules exists from previous applications for adaptation to future simulations.

The early versions of CADAC were executed on the CYBER 6600 with mechanical card readers as input devices. Listings of block data provided the major output and, occasionally, CALCOM plotter graphs were used. Assigned computer memory resources restricted the size of the trajectory programs and forced the programmer to write efficient code. During the early seventies CADAC consisted of the basic executive routine and modules that simulated the MGGB (Modular Guided Glide Bomb). As the GBU-15 program matured modules were written for the planar and cruciform configurations, DME guidance, INS navigation and radiometric seekers. Portable typewriter terminals, acoustically coupled through phone lines to the CYBER, increased input flexibility. The capabilities were extended to simulate AMRAAM, cruise missiles and dispenser munitions.

With the introduction of the VAX in the mid eighties, memory restrictions were removed and CRT terminals increased the turn-around time drastically. The CADAC executive code was significantly improved with multi-run and Monte Carlo capabilities. The program developer was given several tools for debugging and documenting the code. The dominant output form now was graphics, custom made for the Tektronix color terminals. CADAC simulations were extended to trans-atmospheric vehicles and anti-satellite missiles. The conceptual phase of JDAM was supported with several end-to-end guided vehicle simulations. Advanced air-to-air concepts were simulated and the sweep methodology developed to automate the generation of launch envelopes and footprints.

With the advent of powerful IBM compatible PCs, the CADAC Trajectory Simulation Development Tool was converted to the DOS and Windows 3.1 environment and now the Windows 95/98/NT environment. The capability of the CYBER 6600 is now packaged in a small box with software and graphics that are much more flexible and easier to use. A typical trajectory runs four times faster on the 486DX2-66 processor than on the VAX 4000. Additional utilities are provided to ease the input data file generation, the cross-check of equivalenced variables and the averaging of stochastic runs. Output graphics are now displayed with a commercially available plot program. The major emphasis is on full 6 DOF short range air-to-air missile simulations.

The first release for the PC occurred in March 1995. It was a complete trajectory simulation development tool for personal computers under the DOS/Windows 3.1 operating system. The second release (ver 1.1) occurred in May 1997 and addressed reported program problems. The version 2.0 release added several new features for the PC system as well as moving from the DOS/Win3.1 environment to Windows 95/NT systems. The sweep functions were greatly enhanced under release 2.0. Release 2.1 is a revision to 2.0 with minor functional and editorial modifications. Release 3.0 saw the inclusion of the real-time CADAC application as well as some functional and editorial modifications. Release 3.1 included updates and functional modifications to the graphical CADAC Studio.

2. CADAC ENVIRONMENT

The development of a simulation in the CADAC environment requires a person acting as both simulation programmer and environment user. This person must provide program code to tailor the CADAC environment for the specific missile system, vehicle, etc., being simulated. This person must also become familiar with the execution commands and options provided by the CADAC environment for executing the simulation. For this reason, the terms "user" and "programmer" may be used interchangeably to refer to the person utilizing the CADAC environment to program and then execute a simulation.

The CADAC environment is composed of an assortment of files. These files may contain one or more subroutines/functions, complete programs or data in specific formats. These files can be grouped into three categories: CADAC program files, CADAC input files and CADAC utilities. The CADAC program files consist of the FORTRAN 77 program code providing the missile/vehicle simulation. The CADAC input files provide information to the simulation pertaining to the flight scenario, the desired input and desired output. The CADAC utilities assist the user in the development of the simulation and input files and provide analysis tools for the data generated by the simulation.

The CADAC development environment has been designed toward satisfying two apparently conflicting goals with one simulation model. These goals are:

1) The simulation model should at all times reflect the current real world system
2) The model must have the capability of rapid modification in order to correspond in a timely manner to requested hypothetical situations

The first goal is met by maintaining strict configuration control over the baseline model (the executive routines), while the second goal is met by making the source code for the modules available to each user. The user may temporarily modify the baseline to model a non-baseline system. Modification by the user is further facilitated by the program modularity, which is analogous to the system it simulates.

3. CADAC INPUT

During execution, the program obtains data from two input files, INPUT.ASC and HEAD.ASC, to set program options, select output variables, define variable values and set up conditions to halt the simulation.

3.1 INPUT.ASC

The INPUT.ASC file contains statements used to define the simulation. This file consists of user-friendly statements which allow the user to define the simulation. Each statement, which is not case sensitive, can contain up to 132 characters. All the variables referred to in the INPUT.ASC statements must be contained in the current HEAD.ASC file. Prior to executing CADAC, the free-formatted INPUT.ASC file must be converted to the fixed-formatted file CADIN.ASC by executing program CADIN. Program CADIN is discussed in Section 5.4.1. Table 1 summarizes the statements found in a INPUT.ASC file.

Table 1. INPUT.ASC Statement Summary.

STATEMENT	DEFINITION
ASSIGNMENTS:	assigns values to a variable once during a pause in the trajectory integration; the value assigned depends on the format used:
VAR = Real Value	assigns a real value to VAR
VAR = VAR2	assigns the value of VAR2 to VAR
VAR = GAUSS(Mean,StdDev)	assigns a value selected from the Gaussian distribution with the given mean and standard deviation to VAR
VAR = UNIF(Mean,Width)	assigns a value selected from the uniform distribution with the given mean and uniform width to VAR
VAR = EXPO(λ)	assigns a value selected from the exponential distribution with the given λ to VAR
VAR = RAYLE(α)	assigns a value selected from the Rayleigh distribution with the given α to VAR
VAR = SIGN(Value)	assigns the absolute value of Value if a random number generated is less than .5 or the negative of the absolute value of Value if a random number generated is greater than or equal to .5 to VAR
VAR = INT(Value)	assigns an integer value to VAR
! (comment)	any statement that contains an exclamation mark in the first column
CLEAR	deactivates the currently defined functions
FUNCTIONS:	generates a value from a time-dependent function, which is evaluated twice during each integration step, to a variable; the value of the function, which can be assigned to (=), added to (+), subtracted from (-) or multiplied by (*) the VAR, depends on the format used (in these formats, ^ signifies one of the combination characters):
FUNC VAR ^ COS(Amp,Period)	generates a value from a cosine wave function with the given amplitude and period which can be combined with VAR
FUNC VAR ^ DECAY(Amp,Rate)	generates a value from a decay function with the given amplitude and decay rate which can be combined with VAR
FUNC VAR ^ DIFF(VAR2,VAR3)	generates the difference between the variable VAR2 and the variable VAR3 which can be combined with VAR
FUNC VAR ^ EQUALS(VAR2)	the value of VAR2 is combined with VAR
FUNC VAR ^ GAUSS(StDev,TCor)	generates a value from a time correlated Gaussian function with the given standard deviation and time correlation which can be combined with VAR

Table 1. INPUT.ASC Statement Summary. (Continued)

STATEMENT	DEFINITION
FUNC VAR ^ MARKOV(StDev,TCor)	generates a value from a time correlated Gaussian function with the given standard deviation and time correlation which can be combined with VAR
FUNC VAR ^ PARAB(Rnge,Val)	generates a value from a parabolic curve function with the given value (where value is equivalent to 4 times the distance between the focus and the vertex) and range which can be combined with VAR
FUNC VAR ^ PROD(VAR2,VAR3)	generates the product of the variable VAR2 and the variable VAR3 which can be combined with VAR
FUNC VAR ^ RAMP(Rnge,Val)	generates a value from a ramp function with the given ramp value and range which can be combined with VAR
FUNC VAR ^ RAYLE(Mean,Md)	generates a value from a Rayleigh function with the given mean and Rayleigh mode which can be combined with VAR
FUNC VAR ^ SIN(Amp,Period)	generates a value from a sine wave function with the given amplitude and period which can be combined with VAR
FUNC VAR ^ SQR(Amp,Period)	generates a value from a square wave function with the given amplitude and period which can be combined with VAR
FUNC VAR ^ STEP(StepVal)	generates a value from a step function with the given step value which can be combined with VAR
FUNC VAR ^ SUM(VAR2,VAR3)	generates the sum of the variable VAR2 and the variable VAR3 which can be combined with VAR
FUNC VAR ^ TRI(Rnge,Per)	generates a value from triangular wave function with the given amplitude range and period which can be combined with VAR
FUNC VAR ^ UNIF(Mn,Wdth)	generates a value from a uniform function with the given mean and uniform width which can be combined with VAR
IF StageCriteria	defines the stage criteria that must be satisfied for a stage completion
LOAD	indicates the end of a set of group statements, used for Multi-Run, see Section 3.1.1.4 for example
MONTE NumOfRuns	indicates the number of trajectories to be made, used for Multi-Runs, see Section 3.1.1.4 for example
RANDOM(Seed,RandomOptions)	Assigns the initial seed value to the variable RANSEED and sets the method to use for setting the random seed, see Section 3.1.2.3.6
RUN	informs the CADAC Executive to complete the simulation based on the information that has been obtained from the input file up to this time
SAVE	causes the CADAC Executive to save the state of the program
STOP	indicates that the end of the lead statements has been reached and program execution can be stopped
TITLE execution title	indicates the title of the CADAC Execution; this statement must be included in INPUT.ASC
VECTOR VectName VectVal	assigns the value of all the elements of the vector VectName
VECTORV VectName Val_1, Val_2, ...	assigns Val_1, Val_2,... to the elements of the vector VectName
HEADER TextLines END	the HEADER block allows the user to enter up to 5 lines of text
MODULE Module Names END	the MODULE block allows the user to indicate the modules to be executed during the CADAC simulation and the calling sequence of the modules

Table 1. INPUT.ASC Statement Summary. (Concluded)

STATEMENT	DEFINITION
SWEEP MODE ModeType LIMIT CritVal< CritVar<[MaxVal] NUM NumBins/NumTrajs RANGE MnRng<RngVar<MxRng ANGLE MnAng<AngVar<MxRng END	the SWEEP block allows the user to set the variables used in the SWEEP option of CADAC. See Section 3.1.2.4.3 for complete descriptions
WEATHER	the WEATHER block allows the user to input weather data for the atmospheric modules, see Section 3.1.2.4.5 for complete details

3.1.1 INPUT.ASC Formats

INPUT.ASC has several general formats. Of these formats, seven are most common and are discussed in this section. This discussion is presented at this point in the report to give the user an overview of the input formats and capabilities and to provide the user with a basic understanding for Section 3.1.2, the discussion of the statement formats.

The first non-comment statement of the INPUT.ASC file is always a TITLE statement; the user may enter a title of up to 72 characters. The ordering of the statements in the input file following the TITLE statement is the responsibility of the user. CADAC does not have a required ordering except in rare cases. However the ordering of the statements dictates the actions of the CADAC simulation and is therefore important. A standard format is practiced by most CADAC users.

3.1.1.1 Single Trajectory INPUT.ASC File

The first non-comment statement of a INPUT.ASC file is the TITLE statement. The user can include the MONTE statement following the TITLE statement with a value of 1 to indicate that a single trajectory is to be simulated or the MONTE statement can be omitted. Following the TITLE statement and the MONTE statement is the block of module definitions dictating the user supplied modules to be used in the simulation. In the module block, the names of the modules are indicated by two characters. (For more information on the module block, see Section 3.1.2.4.2.) Following the module block is the block of stage 0 statements; these statements initialize the variables at the start of the simulation. Common statements/blocks in this section are the assignment statements, comment statements, WEATHER blocks, VECTOR statements, HEADER blocks and FUNC statements. An INPUT.ASC input file demonstrating the standard file format is shown in Figure 1; this file generates a simple, single trajectory. In this particular input file, there is only one stage, stage number 0. Following this stage is the RUN statement that tells CADAC to execute the simulation to completion (i.e., when the trajectory vehicle impacts with the ground or other termination conditions are met). The STOP statement indicates that the end of the input file has been reached and signals the program to stop execution.

The set of statements from the first statement in the input file to the RUN statement is sometimes called the primary trajectory because the data from these statements is loaded into a set of storage arrays within the executive. The stored data is copied into a set of working arrays for the simulation. Under certain conditions, this primary trajectory is used as the basis for a new trajectory and modified by INPUT.ASC statements. The primary trajectory has a maximum limit of 500 statements.

```
        TITLE CADAC Single Trajectory Input File
        !Comment
        MODULES
        D3  SWEEP
        G1  TARGET
        G2  AIR DATA
        S1  SEEKER
        S2  RADAR
        S4  INS
        C1  GUIDANCE
        C2  CONTROL
        A1  AERO COEF
        A2  PROPULSION
        A3  FORCES
        D1  DYNAMICS
        D2  ROTATION
        END MODULES
        !
        ! *** Assignment statements ***
        !
        OPTMET = 1.
        DVT1E  = 284.2
             •
             •
             •
        PPP = .5
        CPP = .5
        DER = .01
        !
        RUN
        STOP
```

Figure 1. Simple, Single Trajectory INPUT.ASC File.

3.1.1.2 INPUT.ASC File With Stages

More than one stage may be entered on the file; Figure 2 shows an INPUT.ASC file with three stages. A stage criteria is placed at the end of stage 0 with the IF statement. Additional stages are then added following stage 0. The final stage again ends with the RUN statement, then the STOP statement signals the end of the file. These multiple stages are considered a part of the primary trajectory. A maximum of 20 stages may be entered on the input file and each stage can contain up to 60 statements. Trajectory restarts may also be executed by issuing a SAVE statement immediately after an IF statement. For this type of simulation to be meaningful, assignment statements should be included after the SAVE statement. Succeeding trajectories are executed starting at the point at which the simulation was saved using the SAVE statement.

```
        TITLE CADAC Single Trajectory Input File with Stages
        MODULES
```

```
D3  SWEEP
G1  TARGET
G2  AIR DATA
S1  SEEKER
S2  RADAR
S4  INS
C1  GUIDANCE
C2  CONTROL
A1  AERO COEF
A2  PROPULSION
A3  FORCES
D1  DYNAMICS
D2  ROTATION
END MODULES
!
! *** Assignment statements ***
OPTMET = 1.
DVT1E = 284.2
     •
     •
     •
PPP = .5
CPP = .5
DER = .01
!
IF TIME  > .5        ! *** STAGE 1:
MGUID = INT(13.)
MAUT  = INT(144.)
!
IF DBT1 <  11111.2   ! *** STAGE2:
MGUID = INT(15.)
!
RUN
STOP
```

Figure 2. Single Trajectory INPUT.ASC File With Three Stages.

3.1.1.3 Monte Carlo INPUT.ASC File

If a Monte Carlo set of trajectories is to be generated, then the value of the NumOfRuns variable in the MONTE statement must be greater than 1. Figure 3 shows an example of an input file creating Monte Carlo simulations. To generate the set of Monte Carlo trajectories, the primary trajectory is loaded into the working arrays and simulated. After the first simulation, the primary trajectory is again loaded into the working arrays and again simulated. This looping procedure occurs the number of times indicated on the MONTE statement. Of course, for the Monte Carlo runs to be meaningful, stochastic variables must be present in the assignment statements.

```
TITLE Typical Monte Carlo CADAC Input File
```

```
MONTE CARLO  3
MODULES
D3  SWEEP
G1  TARGET
G2  AIR DATA
S1  SEEKER
S2  RADAR
S4  INS
C1  GUIDANCE
C2  CONTROL
A1  AERO COEF
A2  PROPULSION
A3  FORCES
D1  DYNAMICS
D2  ROTATION
END MODULES
!
! *** Assignment statements ***
OPTMET = 1.
DVT1E  = 284.2
     .
     .
     .
CPP = .5
DER  = .01
!
RUN
STOP
```

Figure 3. Sample INPUT.ASC File for Monte Carlo Simulations.

3.1.1.4 INPUT.ASC File With Group Statement Sets

Another method of generating a set of trajectories is to include modifications to the primary trajectory at the end of the file. The statements indicating the modifications to the primary trajectory are referred to as group statements. Figure 4 shows an INPUT.ASC file with two sets of group statements. The first set of group statements starts with the first statement after the RUN statement and includes all of the following statements, ending with the LOAD statement. When CADAC processes this INPUT.ASC file, (in the form of a CADIN.ASC file), the primary trajectory will be read, stored in the storage arrays, copied to the working arrays and then simulated. The program will then load a copy of the primary trajectory into the working arrays from the storage arrays. The first statement after the primary trajectory (the first group statement) is then read. The group statements include the keyword STAGE with a StageNo that is used to indicate the stage of the primary trajectory that the statement is to modify. If this integer is 0 or the STAGE keyword is not included, then stage 0 of the primary trajectory is searched for a statement with matching statement type and defining variable. If a match is found, then the new statement is used in place of the original statement. If a match is not found then a message is displayed and the new statement is ignored. If the group statement StageNo is non-zero, then the statements from the primary trajectory for the indicated stage is searched for a match. If a match is found, then the data from the new statement is used in place of the original data. If no match is found, then the statement is added to the indicated stage as long as the maximum number of statements allowed within the stage (60) has not been reached. There is not a limit to the number of groups that may be input on the file since this data is read directly from the input file and is not saved to storage arrays. As the group cards tend to modify existing cards, there is not a defined maximum to the number of statements for each group. However, there is a maximum number of statements that are allowed within each stage. If the group statements are added to the existing stage statements and the number of the statements within the stage exceeds the maximum limit, an error message is generated and any future statements to be appended to that stage are ignored. NOTE: The SWEEP block can only be used in the primary trajectory card, not a group.

```
TITLE INLOAL1.ASC: LOAL Multi-Run,  GNAV=2,3,4
…..
DER = .00123         ! E Integration step - s
IF TIME > .3         ! ### BEGIN OF STAGE 1 ###
MSEEK = INT(2)       ! D/G =2:Enable, =3:Acquisition, =4:Lock, =5:Blind
MGUID = INT(3)       ! D/G =0:None, =2:LAG, =3:Pro-Nav, =6:Aug.Pro-Nav
GNAV = 3             ! D Navigation gain - m/s
MAUT = INT(3) ! D =1:Rate; =2:Look angle; =3:Integ acc; =4:Prop acc
IF DBT1 < 4000       ! ### BEGIN OF STAGE 2 ###
MSEEK = INT(3)       ! D/G =2:Enable, =3:Acquisition, =4:Lock, =5:Blind
IF TIME > 15         ! ### STOP RUN ON TIME OUT ###
RUN
GNAV = 2 STAGE 1     ! D Navigation gain - m/s
LOAD
GNAV = 4 STAGE 1     ! D Navigation gain - m/s
LOAD
STOP
```

Figure 4. INPUT.ASC File With Groups.

3.1.1.5 INPUT.ASC File Utilizing Sweep Methodology

Another method of generating a set of related trajectory simulations is accomplished by using the sweep methodology. The sweep methodology is activated by including the special SWEEP statement block into the primary trajectory card set and setting the SWEEP keyword in the HEAD.ASC header record. The sweep methodology is similar to executing CADAC in a double-nested DO loop. The DO control variables are two of the simulation variables, one controlling an inner loop and one controlling an outer loop. The method of incrementing the looping variables is initialized by the sweep statements and controlled by a set of sweep specific modules. The sweep modules are included with the CADAC Executive code. For more specific information on the sweep methodology, refer to Section 5.2. Figure 5 displays a typical input file using the sweep methodology.

```
TITLE CADAC Input File Utilizing the Sweep Methodology
```

```
MODULES
D3 SWEEP
G1 TARGET
G2 AIR DATA
S1 SEEKER
S2 RADAR
S4 INS
C1 GUIDANCE
C2 CONTROL
A1 AERO COEF
A2 PROPULSION
A3 FORCES
D2 ROTATION
END MODULES
!
OPTMET = 1.
DVT1E = 284.2
HBE = 1000.0
     •
     •
     •
SWEEP
  MODE  CONST
  RANGE/INNER  10000 < RANGE < 20000  DELT = 1000
  ANGLE/OUTER  20.0 < AZBTL < 40.0 DEG  DELT = 20
  LIMIT 0 < DBT1 < 9999.0
END SWEEP
PPP = 0.5
DER = .01
!
IF TIME > .5         ! *** STAGE 1:
MGUID = INT(13.)
MAUT = INT(144)
!
IF DBT1  <  11111.2  ! *** STAGE 2:
MGUID = INT(15)
RUN
!
STOP
```

Figure 5. INPUT.ASC File Utilizing the Sweep Methodology.

3.1.1.6 INPUT.ASC File Utilizing Sweep Methodology Group Statement Sets

Multiple trajectory simulations with restarting can also be accomplished by adding a SAVE statement immediately after an IF statement on the INPUT.ASC along with a SWEEP block. Combining the INPUT.ASC examples of Section 3.1.1.4 and Section 3.1.1.5 can provide an example of this type of simulation. Refer to these sections for more detail on these methodologies.

3.1.1.7 Monte Carlo INPUT.ASC File Utilizing Sweep Methodology

Monte Carlo trajectories can be used within a sweep run by using the MONTE statement and setting NumOfRuns greater than 1. By using this method, errors are introduced into a weapon module and multiple trajectories generated for a particular range and angle. For more information regarding the sweep methodology refer to Section 5.2. The looping procedure for this type of CADAC execution is discussed in Section 3.1.1.3. Figure 6 shows an example of a Monte Carlo input file used with the sweep methodology. This scenario of sweep executions is meaningless for a MODE of OUT or ALL.

```
        TITLE CADAC Input File Utilizing the Monte Carlo Sweep Methodology
        MONTE CARLO 20
        MODULES
        D3 SWEEP
        G1 TARGET
        G2 AIR DATA
        S1 SEEKER
        S2 RADAR
        S4 INS
        C1 GUIDANCE
        C2 CONTROL
        A1 AERO COEF
        A2 PROPULSION
        A3 FORCES
        D2 ROTATION
        END MODULES
        !
        OPTMET = 1.
        DVT1E = 284.2
        HBE = 1000.0
        SWRWL1=GAUSS(0,.5.0)
        SWRWL2=GAUSS(0,.5.0)
        SWRWL3=GAUSS(0,.5.0)
             •
             •
             •
        MINIT  = INT( 111.0 )
        SWEEP
        MODE  CONSTANT DELTA
        Num  4               ! *** Number of Trajectories
        RANGE/INNER  10000 < RANGE < 20000        DELTA = 1000
        ANGLE/OUTER  20.0 < AZBTL < 40.0  DEG     DELTA = 20
        END SWEEP
        PPP = 0.5
        DER = .01
        !
        IF TIME > .5         ! *** STAGE 1:
        MGUID = INT(13.)
        MAUT = INT(144)
        !
        IF DBT1  <  11111.2  ! *** STAGE 2:
        MGUID = INT(15)
        RUN
        !
        STOP
```

Figure 6. INPUT.ASC File Utilizing the Monte Carlo Sweep Methodology.

3.1.2 INPUT.ASC Statements

There are a variety of valid statement types which can be included in INPUT.ASC. These statement types include comment statements, command statements, variable assignment statements and block statements. Each statement type is discussed in the following sections in detail and the general format for the statement type is given. For all the general formats, the following is true: the order of the information in the statement is relevant; any data/commands included within brackets ([ ]) are optional; if an exclamation mark (!) is included in the statement, all information after the exclamation mark is considered a comment.

3.1.2.1 Comment Statements

Any statement that has an exclamation mark (!) in the first column is treated as a comment. Comment statements can be placed anywhere within the INPUT.ASC file. Comment lines with “!” in column 1 will be written as type 04 cards in CADIN.ASC file.

3.1.2.2 Variable Assignment Statements

Assignment statements allow values to be assigned to variables. Only one variable may be assigned a value by each assignment statement; multiple assignment statements may be entered in the INPUT.ASC. If the variable being assigned is defined as a vector in the HEAD.ASC, then each element in the vector is assigned the value specified. The assignment statement has three formats. A variable may be set to a numeric value, to another variable or to a function value. Each format is discussed in detail in the following sections.

3.1.2.2.1 Initializing Variables to Real Numeric Values

The general format for initializing a variable to a numeric value is:

VarName = Value [*Factor1][STAGE StageNo][! comment]

where VarName is the name of a single variable, an element of a vector or array, a vector or an array included in the HEAD.ASC file. VarName is the name of the variable/vector/array being initialized and Value is the required numeric value to which the variable is being set. The user can modify the original Value by including a multiplication factor, Factor1, preceded by an asterisk (*). If the user wishes the assignment to be made in a stage other than stage 0, the STAGE keyword must be included followed by a stage number. Valid stage numbers are 1 through 20. Comments can be included on an assignment statement by preceding the comment with an exclamation mark (!). NOTE: to set a variable to an integer value, the INT function must be used. In the following example, the variable HLE is being assigned the value 50.0 in STAGE 0:

HLE = 10.0*5

3.1.2.2.2 Initializing Variables to Other Variables

This assignment statement equates a variable to the current value of another variable, allowing the user to save the value of a variable, at a given time, for future use. The general format for initializing a variable to a another variable is:

$$\text{VarName} = \text{VarName2}[\text{STAGE StageNo}][!\ \text{comment}]$$

where VarName is the name of a single variable, an element of a vector or array, a vector or an array in the HEAD.ASC file and VarName2 is the name of single variable or an element of a vector or array also included in the HEAD.ASC file. VarName is the name of the variable/vector/array being initialized and VarName2 is the variable to which VarName is being set. If the user wishes the assignment to be made in a stage other than stage 0, the STAGE keyword must be included followed by a stage number. Valid stage numbers are 1 through 20. Comments can be included on an assignment statement by preceding the comment with an exclamation mark (!). The following example, the value of HLE is being saved to the variable SHLE in STAGE 0:

$$\text{SHLE} = \text{HLE}$$

3.1.2.2.3 Initializing Variables to Function Values

The general format for initializing a variable to a function value is:

$$\text{VarName} = \text{FunctionName}(\ )[\text{STAGE StageNo}][!\ \text{comment}]$$

where VarName is the name of a single variable, an element of a vector or array, a vector or an array included in the HEAD.ASC file. VarName is the name of the variable/vector/array being initialized and FunctionName is a function that generates a value being stored in VarName. The formats for the valid functions and a brief description of the function are given in the following sections. If the user wishes the assignment to be made in a stage other than stage 0, the STAGE keyword must be included followed by a stage number. Valid stage numbers are 1 through 20. Comments can be included on an assignment statement by preceding the comment with an exclamation mark (!).

3.1.2.2.3.1 Variable Assignment Statement - GAUSS Function

The GAUSS function provides a value selected from the Gaussian distribution having a user selected mean and standard deviation (sigma). The format for the GAUSS function is:

$$\text{GAUSS}(\text{Mean}[*\text{Factor1}], \text{StandDev}[*\text{Factor2}])[*\text{Factor3}]$$

where Mean and StandDev are required numerical values while Factor1, Factor2 and Factor3 are optional numerical multiplication factors. The multiplication factors allow the user to modify the parameters of the GAUSS function while retaining a set mean and standard deviation. Any combination of the factors can be included in the GAUSS function. NOTE: both Mean and StandDev are multiplied by Factor3; therefore, if both Factor1 and Factor3 are included, the mean becomes (Mean*Factor1*Factor3) and the standard deviation becomes (StandDev*Factor3).

The Gaussian distributed random value is generated by a call to the function GAUSS. The GAUSS function generates random variables distributed in the following manner:

$$f(x) = (X * \text{StandDev}) + \text{Mean}$$

where X is a Gaussian random variable distributed over (0,1), Mean is the user input mean value for the variable and StandDev is the user input standard deviation.

In the following example, the variable HLE will be given a Gaussian distributed random value having a mean of 5.0 and a standard deviation of 10.0:

$$\text{HLE} = \text{GAUSS}(1.25 * 2,\ 2.5 * 2) * 2$$

3.1.2.2.3.2 Variable Assignment Statement - UNIF Function

The UNIF function provides a value selected from the uniform distribution having a user selected mean and width. The format for the UNIF function is:

$$\text{UNIF}(\text{Mean}[*\text{Factor1}],\ \text{UnifWidth}[*\text{Factor2}])[*\text{Factor3}]$$

where Mean and UnifWidth are required numerical values while Factor1, Factor2 and Factor3 are optional numerical multiplication factors. The multiplication factors allow the user to modify the parameters of the UNIF function while retaining a set mean and width. Any combination of the factors can be included in the UNIF function. NOTE: both Mean and UnifWidth are multiplied by Factor3; therefore, if Factor1 and Factor3 are both included, the mean becomes (Mean*Factor1*Factor3) and width becomes (UnifWidth*Factor3).

The uniform distributed random value is generated by a call to the function UNIF. The UNIF function generates random variables distributed in the following manner:

$$f(x) = \text{Mean} + (2.0 * \text{UnifWidth}) * (\text{RANF}(\ ) - 0.5)$$

where Mean is the mean value for the variable, UnifWidth is half the total width of the variable and RANF is a uniform random number between 0 and 1 (the program utilizes the system's random number generator to obtain values for RANF).

In the following example, the variable HLE will be given a uniform distributed random value having a mean of 2.0 and a UnifWidth of 5.0:

$$\text{HLE} = \text{UNIF}(1.0,\ 2.5) * 2.0$$

3.1.2.2.3.3 Variable Assignment Statement - EXPO Function

The EXPO function provides a value selected from the exponential distribution having the user selected distribution parameter, λ. The format for the EXPO function is:

$$\mathrm{EXPO}(\lambda[\text{*Factor1}])[\text{*Factor2}]$$

where λ is a required numerical value while Factor1 and Factor2 are optional numerical multiplication factors. The multiplication factors allow the user to modify the distribution parameter, λ, of the EXPO function while retaining a set λ. Any combination of the factors can be included in the EXPO function. If Factor1 and Factor2 are both included, λ becomes (λ *Factor1*Factor2).

The exponentially distributed random value is generated by a call to the function EXPO. The EXPO function generates random variables distributed in the following manner:

$$f_x(x) = e^{-\lambda x}$$

The mean, μ_X, and the variance, σ^2_x, of the function are:

$$\mu_x = \lambda$$
$$\sigma_x^2 = \lambda^2$$

More information on the derivation of the formulas and statistics implemented in EXPO may be found in Appendix B.1.1. In the following example, the variable HLE will be given a random value from the exponential distribution having a mean of 5.0:

$$\mathrm{HLE} = \mathrm{EXPO}(2.5*2)$$

3.1.2.2.3.4 Variable Assignment Statement - RAYLE Function

The RAYLEI function provides a value selected from the Rayleigh distribution having the user input distribution parameter, α. The format for the RAYLEI function is:

$$\mathrm{RAYLEI}(\alpha[\text{*Factor1}])[\text{*Factor2}]$$

where α is the required numerical value while Factor1 and Factor2 are optional numerical multiplication factors. The multiplication factors allow the user to modify the parameter α of the RAYLEI function while retaining a set α. Any combination of the factors can be included in the RAYLEI function. If both Factor1 and Factor2 are included, α becomes the value (α *Factor1*Factor2).

The Rayleigh distribution generates a call to the program function RAYLEIGH. The RAYLEIGH function generates random variables distributed in the following manner:

$$f(x) = \frac{x e^{\frac{-x^2}{2\alpha^2}}}{\alpha^2}$$

where the mean, μ_X, standard deviation, σ_X, and mode, M_o, of the function are, respectively:

$$\mu_x = \alpha\sqrt{\frac{\pi}{2}}$$

$$\sigma_x = \alpha^2\left(2 - \frac{\pi}{2}\right)$$

$$M_0 = \sigma$$

More information on the derivation of the formulas and the statistics implemented in the RAYLE function may be found in Appendix B.1.2. In the following example, the variable HLE will be given the value determined by the Rayleigh function with a user input mode of 5.0:

$$\text{HLE} = \text{RAYLEI}(2.5) * 2$$

3.1.2.2.3.5 Variable Assignment Statement - SIGN Function

The format for the SIGN function is:

$$\text{SIGN}(\text{SignValue } [\text{*Factor1}])[\text{*Factor2}]$$

where SignValue is a required numerical value while Factor1 and Factor2 are optional numerical multiplication factors. The multiplication factors allow the user to modify SignValue of the SIGN function while retaining a set SignValue. Any combination of the factors can be included in the SIGN function. If Factor1 and Factor2 are both included, SignValue becomes (SignValue*Factor1*Factor2).

When this function is selected, module SIGN is called by the executive. This module returns a value designated by the function:

$$f(x) = \begin{cases} \|SignValue\|; RNUM < 0.5 \\ -\|SignValue\|; RNUM \geq 0.5 \end{cases}$$

where RNUM is a uniformly distributed random number over the interval (0,1) and SignValue is a user input value included with the statement.

In the following example, the variable HLE will be given a value of either 5.0 or -5.0 depending on the random number value returned by the system's random number generator:

$$\text{HLE} = \text{SIGN}(5.0)$$

3.1.2.2.3.6 Variable Assignment Statement - INT Function

The INT function allows the user to set a variable to an integer value. The format for the INT function is:

INT(Value [*Factor1]) [*Factor2]

where Value is a required numerical value while Factor1 and Factor2 are optional numerical multiplication factors. The multiplication factors allow the user to modify the Value of the INT function while retaining a set Value. Any combination of the factors can be included in the INT function. If Factor1 and Factor2 are both included, Value becomes (Value*Factor1*Factor2).

An example utilizing this function is found in the D1 module. The variable MINIT is a flag indicating the initialization mode for defining the target and launch positions. In the following example, the flag MINIT is set to the integer 111:

MINIT = INT(111)

If the Value is not a whole number, Value is rounded to the nearest whole number. If the INT function was not included, then 111.0 would be stored in MINIT, which would generate an error.

3.1.2.3 Command Statements

Command statements are INPUT.ASC statements that contain a designated keyword at the beginning of the statement line. The following sections describe the statement formats and descriptions.

3.1.2.3.1 CLEAR Statement

The CLEAR statement provides a method for deactivating the currently defined functions. When this statement is issued, all of the currently defined functions are deactivated. If other control functions need to remain defined, the function may be redefined after the CLEAR statement, using the FUNC statement.

The format of the CLEAR statement is:

CLEAR [FUNCTIONS]

The word "FUNCTIONS" may be included to clarify the statement, but it is not required.

3.1.2.3.2 FUNC Statement

A variable may be assigned values from a time-dependent function by using the FUNC statement. This differs from the assignment statement (which assigns a value to a variable once during a pause in the trajectory integration) in that the function value is evaluated and assigned to the variable twice during each integration step (once during the predictor cycle and again during the corrector cycle). The value of the function depends on:

1) the trajectory time and system state when the function was defined

2) the selected function and associated user-input parameters
3) the current trajectory time and system state

These function definitions are useful for defining forcing functions for input into systems, superimposing noise on selected variables and combining selected variables as required by the calculations within a simulation. The user can have up to 25 active functions defined at any time during the simulation. The control functions are activated and deactivated through the use of the FUNC and CLEAR statements. Each control function performs the following computation:

$$C(k,t) = F\left(i,\, S(t_o),\, L(i),\, S(t) \right)$$

where

C(k,t)	=	the value of the kth common location at time t
F(i)	=	the ith control function whose value is a function of the functions defined by the ith control functions

The function, F(i), is a function of

$S(t_o)$	=	the state of the system when the function was activated
L(i)	=	the parameters from the ith type card activated
S(t)	=	the current state of the system at time t

Therefore, the value of F(i) can be determined from the following information:

1) Time of activation
2) Type of combination
3) Type of function
4) Variable to be controlled
5) A constant or a common location or the value of the variable to be controlled at activation time
6) The second constant or common location

After the statements have been read, checked and stored, no further action is taken toward function evaluation until the simulation begins. Once simulation has begun, then the functions are evaluated. The order of evaluation is dependent on the variable number being controlled. The module CNTROL controls the evaluation of the defined functions. Module CNTROL is called by module AUXSUB; AUXSUB coordinates the execution of the user supplied modules. Module AUXSUB is called twice during the integration calculation; once after the predictor stage, then again after the corrector stage. This causes CNTROL to evaluate all functions twice during the integration.

In AUXSUB, prior to evaluation of any of the user supplied modules, module CNTROL is called and any functions affecting C elements 1 through 99 or 1800 through 3510 are evaluated. Then the execution of the user supplied modules is initiated. After a user supplied module is executed, module CNTROL is again called and any functions affecting the C elements assigned to that module are evaluated. Table 2 shows the assignment of C element locations to the user supplied modules.

Table 2. Valid Module Names.

MODULE NAME	MODULE TYPE	Reserved Common Locations	COMMENTS
G1	Target	100-199	2 Points
G2	Air Data	200-299	Standard/Table
G3	Kinematic	300-399	
G4	Terminal	1750-1799	Not Module
S1	Seeker	400-499	TV, IIR, Radar
S2	Sensor	500-599	GPS
S3	NAV Filter	600-699	Kalman
S4	INS Model	700-799	Strap Down
C1	Guidance	800-899	Pro.-NAV
C2	Autopilot	900-999	Acceleration, Angles
C3	TVC	1000-1099	
C4	Acuator	1100-1199	
A1	Aero Coefficients	1200-1299	Cruciform, Planar
A2	Propulsion	1300-1399	Rocket, Turbojet, Ramjet
A3	Forces & Moments	1400-1499	
A4	Growth	1500-1599	
D1	Translational	1600-1699	
D2	Rotational	1700-1749	
D3	Sweep	1800-1899	
D4	Growth	1900-1999	

3.1.2.3.2.1 FUNC Statement Format

The FUNC statement has 4 formats that determine how the calculated function is to be utilized:

```
VarName   =  Function Name ( )    [ STAGE StageNo ]   [ ! comment ]
VarName   +  Function Name ( )    [ STAGE StageNo ]   [ ! comment ]
VarName   -  Function Name ( )    [ STAGE StageNo ]   [ ! comment ]
VarName   *  Function Name ( )    [ STAGE StageNo ]   [ ! comment ]
```

where VarName is the name of a single variable, an element of a vector or array, a vector or an array included in the HEAD.ASC file. VarName is the variable being defined by the function statement and FunctionName is the function that is to be used to provide the value for VarName. The combination characters (i.e., "=", "+", "-" and "*") determine how the resulting values from FunctionName are combined or stored with the current contents of VarName. If the combination character is "=", the function value is stored in VarName. If the combination character is the "+", then the function value is added to the current value of VarName. If the combination character is the "-", then the function value is subtracted from the current value VarName. If the combination character is the "*", then VarName is set to the function value multiplied by the current value of VarName. If the user wishes the function to be defined in a stage other than stage 0, the STAGE keyword must be included followed by the stage number. Valid stage numbers are 1 through 20.

Comments can be included on a FUNC statement by preceding the comment with an exclamation mark (!).

3.1.2.3.2.2 FUNC Statement - COS Function

The format for the COS function is:

$$\mathrm{COS(Amplitude[*Factor1], Period[*Factor2])[*Factor3]}$$

where Amplitude and Period are required numerical values while Factor1, Factor2 and Factor3 are optional numerical multiplication factors. The multiplication factors allow the user to modify the parameters of the COS function while retaining a set amplitude and period. Any combination of the factors can be included in the COS function. NOTE: both Amplitude and Period are multiplied by Factor3; therefore, if Factor1 and Factor3 are both included, the amplitude becomes (Amplitude*Factor1*Factor3) and the period becomes (Period*Factor3).

This function generates a cosine wave by using the following equation:

$$\mathrm{X = Amplitude * COS(6.2838 * (t - t_0) / Period)}$$

where $t_0$is the time at which the function was defined, Amplitude is the amplitude of the cosine curve and Period is the period of the curve. If the amplitude is 0.0, CADAC will load the value of the controlling variable at the time the function is defined for the amplitude.

In the following example, the variable AZBTL is assigned a value generated from the cosine function with an amplitude of 1 and a period of 2π:

$$\mathrm{FUNC\ AZBTL = COS(1, 2 * 3.1419)}$$

3.1.2.3.2.3 FUNC Statement - DECAY Function

The format for the DECAY function is:

$$\mathrm{DECAY(Amplitude[*Factor1], DecayRate[*Factor2])[*Factor3]}$$

where Amplitude and DecayRate are required numerical values while Factor1, Factor2 and Factor3 are optional numerical multiplication factors. The multiplication factors allow the user to modify the parameters of the DECAY function while retaining a set amplitude and decay rate. Any combination of the factors can be included in the DECAY function. NOTE: both Amplitude and DecayRate are multiplied by Factor3; therefore, if Factor1 and Factor3 are both included, the amplitude of the function becomes (Amplitude*Factor1*Factor3) and the decay rate becomes (DecayRate*Factor3). This function generates a decay function following the equation:

$$X = Amplitude * EXPO(-ABS(DecayRate * (t - t_0)))$$

where t_o is the time when the function was defined, Amplitude is the amplitude of the function and DecayRate is the rate of decay of the function. If the amplitude is 0.0, CADAC will load the value of the controlling variable at the time the function is defined for the amplitude.

In the following example, the variable HLE is assigned the current value of HLE multiplied by a value generated by the DECAY function with an amplitude equal to the controlling variable and a decay rate of 1.0:

$$\text{FUNC HLE} * \text{DECAY}(5.0 * 0.0, 1.0)$$

3.1.2.3.2.4 FUNC Statement - DIFF Function

The format for the DIFF function is :

$$\text{DIFF}(\text{VarName2}, \text{VarName3})$$

where VarName2 and VarName3 are the names of single variables or an element of a vector or array included in the HEAD.ASC file.

The value of the function follows the equation:

$$X = \text{VarName2} - \text{VarName3}$$

In the following example, the variable HLE is being assigned the current value of HLE plus the difference between SBEL(2) and SBEL(1):

$$\text{FUNC HLE} + \text{DIFF}(\text{SBEL(2)}, \text{SBEL(1)})$$

3.1.2.3.2.5 FUNC Statement - EQUALS Function

The format for the EQUALS function is:

$$\text{EQUALS}(\text{VarName2})$$

where VarName2 is the name of a single variable or an element of a vector or array included in the HEADER.ASC file. This function allows the data in one variable to be transferred to the variable being defined by the function, thereby making the variables equal. The function is defined by the equation:

$$X = \text{VarName2}$$

The modifications to the variables being defined are then performed according to the combination character entered on the FUNC statement.

In the following example, the variable HLE is set to the current value of HLE multiplied by the current value of SBEL(1):

$$\text{FUNC HLE} * \text{EQUALS}(\text{SBEL(1)})$$

3.1.2.3.2.6 FUNC Statement - MARKOV/GAUSS Function

The valid formats for the MARKOV/GAUSS function are:

MARKOV(StandDev[*Factor1], TimeCorr[*Factor2])[*Factor3]
GAUSS(StandDev[*Factor1], TimeCorr[*Factor2])[*Factor3]

The keywords GAUSS and MARKOV are both accepted for this function. In this function, StandDev and TimeCorr are required numerical values while Factor1, Factor2 and Factor3 are optional numerical multiplication factors. The multiplication factors allow the user to modify the parameters of the MARKOV/GAUSS function while retaining a set standard deviation and time correlation. Any combination of the factors can be included with the function. NOTE: both the StandDev and the TimeCorr are multiplied by Factor3; therefore, when both Factor1 and Factor3 are included, the time correlation becomes (TimeCorr*Factor3) and the standard deviation becomes (StandDev*Factor2*Factor3).

This function computes a time correlated, Gaussian random variable with standard deviation, σ, and a time correlation coefficient, ß. The computation of the function follows four steps:

1. Compute G, a Gaussian random variable with a mean of 0 and a standard deviation of σ
2. Compute A = Exp(-β*Dt)
3. X = G * SQRT(0.0 - A*A) + LastX * A
4. Save: LastX = X

where Dt is the integration step size of the simulation, StandDev is the sigma value for the Gaussian distribution and β (TimeCorr) is the beta for the exponential function. If the standard deviation is 0.0, CADAC will load the value of the controlling variable at the time the function is defined for the standard deviation.

In the following example, the variable HLE is being set to the current value of HLE less the value generated by the MARKOV function having a standard deviation of 20 and a time correlation of 5:

FUNC HLE – MARKOV(10*2,5)

3.1.2.3.2.7 FUNC Statement - PARAB Function

The format for the PARAB function is:

PARAB(Val[*Factor1], Range[*Factor2])[*Factor3]

where Val and Range are required numerical values while Factor1, Factor2 and Factor3 are optional numerical multiplication factors. The multiplication factors allow the user to modify the parameters of the PARAB function while retaining a set Val and Range. Both Val and Range are multiplied by Factor3; therefore, if Factor1 and Factor3 are included, Val becomes (Val*Factor1*Factor3) and Range becomes (Range*Factor3).

PARAB generates a parabolic curve that is restricted by {-Range ≤ X ≤ Range } and calculated by the equation:

$$X = Val * (t - t_0)^2$$

where t_0 is the time at which the function was defined. Using the standard definition of a parabola, Val is equivalent to 4 times the distance between the focus and the vertex. If X exceeds Range (or -Range), then Range (or -Range) is returned. If Range is 0.0, CADAC loads the value of the controlling variable at the time the function is defined for the range.

In the following example, HLE is assigned a value generated by the PARAB function using 10.0 and is restricted to the interval [0,100.0]:

$$\text{FUNC HLE} = \text{PARAB}(10.0,100)$$

3.1.2.3.2.8 FUNC Statement - PROD Function

The format for the PROD function is:

$$\text{PROD}(\text{VarName2}, \text{VarName3})$$

where VarName2 and VarName3 are the names of single variables or an element of a vector or array included in the HEAD.ASC file.

The value of the function follows the equation:

$$X = \text{VarName2} * \text{VarName3}$$

In the following example, the variable HLE is being set to the current value of HLE multiplied by the product of SBEL(1) and SBEL(2):

$$\text{FUNC HLE} * \text{PROD}(\text{SBEL(1)},\text{SBEL(2)})$$

3.1.2.3.2.9 FUNC Statement - RAMP Function

The format for the RAMP function is:

$$\text{RAMP}(\text{RampVal}\,[\text{*Factor1}], \text{Range}\,[\text{*Factor2}])\,[\text{*Factor3}]$$

where RampVal and Range are required numerical values while Factor1, Factor2 and Factor3 are optional numerical multiplication factors. The multiplication factors allow the user to modify the parameters of the RAMP function while retaining a set RampVal and Range. Any combination of the factors can be included in the RAMP function. NOTE: both RampVal and Range are multiplied by Factor3; therefore, if Factor1 and Factor3 are both included, the RampVal variable becomes (RampVal*Factor1*Factor3) and the range becomes (Range*Factor3).

This statement generates a RAMP function (constant derivative), restricted to {-Range ≤ X ≤ Range } and computed by the equation:

$$X = \mathrm{RampVal} * (t - t_0)$$

where t_0 is the time at which the function is defined. If X exceeds Range (or -Range), then the value Range (or -Range) is returned by the function. If the value of the range is 0.0, CADAC will load the value of the controlling variable at the time the function is defined for the range.

In the following example, the variable HLE is being assigned the current value of HLE less the value generated by the RAMP function having a slope of 20.0 and defined over the interval [0,1000.0]

$$\mathrm{FUNC\ HLE - RAMP(10 * 2.0{,}1000)}$$

3.1.2.3.2.10 FUNC Statement - RAYLE Function

The format of the RAYLE function is:

$$\mathrm{RAYLE\,(\,Mean[*Factor1], RayleMode[*Factor2], Multiplier[*Factor3]\,)\,[*Factor4]}$$

where Mean, RayleMode and Multiplier are required numerical values while Factor1, Factor2, Factor3 and Factor4 are optional numerical multiplication factors. The multiplication factors allow the user to modify the parameters of the RAYLE function while retaining a set Mean, RayleMode and Multiplier. Any combination of the factors can be included in the RAYLE function. NOTE: Mean, RayleMode and Multiplier are all multiplied by Factor4; therefore, if Factor1, Factor3 and Factor4 are included, Mean becomes (Mean*Factor1*Factor4), RayleMode becomes (RayleMode*Factor4) and Multiplier becomes (Multiplier*Factor3 *Factor4).

The function provides a pulse of designated width, whose height follows the Rayleigh distribution and the time between pulses follows the exponential distribution. The function returns the value as follows:

$$f(x) = \begin{cases} 0.0; & C_{t_o} \le t \le t_o + (N * DER) \\ H_{obs}; & t < C_{t_o} \end{cases}$$

where

t	=	The current time
N	=	The Multiplier variable. NOTE: if N≤0, N is set to 5
DER	=	The integration step size
C_{t_0}	=	The time an obstacle is encountered; this value is set after each obstacle has been passed to the value: $C_{t_0} = t + \Delta_{t_{obs}}$ where: $\Delta_{t_{obs}}$ = Delta time to next obstacle; distributed exponentially with a user mean
H_{obs}	=	The height of the obstacle; Distributed Rayleigh with the user-input RayleMode
t_0	=	The time when the obstacle is encountered

The Multiplier variable is the integer multiplier on the integration time used to determine the pulse width. The Mean variable is the mean of the exponential function used to generate the time occurrence of the next pulse. The RayleMode variable is the Rayleigh mode used to calculate the height of the pulse. If the mean value of the exponential function is 0.0, CADAC will load the value of the controlling variable at the time the function is defined for Mean. The user should be aware that the FUNC statements are evaluated twice per integration interval; once after the predictor cycle and once after the corrector cycle. However, this has no effect on this function. Also, if the user modifies the integration step size while an obstacle is being encountered, the new value of DER will not affect the current obstacle width but will affect the width of the next obstacle.

In the following example, the variable HLE is being assigned the value generated by the RAYLE function with a mean of 20.0, a mode of 5 and a multiplier of 2.0:

$$\text{FUNC HLE} = \text{RAYLE}(10*2.0,5,2)$$

3.1.2.3.2.11 FUNC Statement - SIN Function

The format of the SIN function is:

$$\text{SIN}(\text{Amplitude[*Factor1]}, \text{Period[*Factor2]})\text{[*Factor3]}$$

where Amplitude and Period are required numerical values while Factor1, Factor2 and Factor3 are optional numerical multiplication factors. The multiplication factors allow the user to modify the parameters of the SIN function while retaining a set amplitude and period. Any combination of the factors can be included in the SIN function. NOTE: both Amplitude and Period are multiplied by Factor3; therefore, if Factor1 and Factor3 are included, the amplitude becomes (Amplitude*Factor1*Factor3) and the period becomes (Period*Factor3).

This function generates a sine wave by using the following equation:

$$X = \text{Amplitude}*\text{SIN}(6.2838*(t - t_0)/\text{Period})$$

where t_0 is the time at which the function was defined, Amplitude is the amplitude of the sine curve and Period is the period of the curve. If the amplitude is 0.0, CADAC will load the value of the controlling variable at the time the function is defined for the amplitude.

In the following example, the variable AZBTL is being assigned the current value of AZBTL less the value generated by the SIN function with an amplitude of 5 and a period of 4π:

$$\text{FUNC AZBTL} - \text{SIN}(5,4*3.1419)$$

3.1.2.3.2.12 FUNC Statement - SQR Function

The format for the SQR function is:

$$\text{SQR}(\text{Amplitude[*Factor1]}, \text{Period[*Factor2]})\text{[*Factor3]}$$

where Amplitude and Period are required numerical values while Factor1, Factor2 and Factor3 are optional numerical multiplication factors. The multiplication factors allow the user to modify the parameters of the SQR function while retaining a set amplitude and period. Any combination of the factors can be included in the SQR function. NOTE: both Amplitude and Period are multiplied by Factor3; therefore, if Factor1 and Factor3 are included, the amplitude becomes (Amplitude*Factor1*Factor3) and the period becomes (Period*Factor3).

This function generates a square wave using the SQR function:

$$X = \mathrm{SQR}(\mathrm{Amplitude}, \mathrm{Period}, (t - t_0))$$

This function follows the logic:

$$\begin{aligned} X &= \quad \mathrm{Amplitude} \quad \text{when } \mathrm{TIP} \le 0.5 * \mathrm{Period} \\ X &= -\mathrm{Amplitude} \quad \text{when } \mathrm{TIP} \le 0.5 * \mathrm{Period} \end{aligned}$$

where TIP is the elapsed time within the given Period. TIP is calculated as:

$$\mathrm{TIP} = \mathrm{AMOD}(t - t_0, \mathrm{Period})$$

If the value of the amplitude is 0.0, CADAC will load the value of the controlling variable at the time the function is defined for the amplitude.

In the following example, the variable HLE is being assigned the value generated by the SQR function with an amplitude of 5 and a period of 10:

$$\text{FUNC HLE} = \mathrm{SQR}(5,10)$$

3.1.2.3.2.13 FUNC Statement - STEP Function

The formats for the STEP function are:

$$\mathrm{STEP}(\mathrm{StepVal}[*\mathrm{Factor1}])[*\mathrm{Factor2}]$$

where StepVal is a required numberical value while Factor1 and Factor 2 re optional numberical multiplication factors. The multiplicaiton factors allow the user to modify the parameter of the STEP function while retaining a set step value. Any combination of the factors can be included in the STEP function. NOTE: therefore, if both Factor1 and Factor2 are included, the step value becomes (StepVal*Factor1*Factor2).

This function generates a step function defined as:

$$X = \mathrm{StepVal}$$

If the value of the step is 0.0, CADAC will load the value of the controlling variable at the time the function is defined for the step value.

In the following example HLE is being assigned the value generated by the STEP function with a step value of 10.0:

$$\mathrm{FUNC\ HLE = STEP(10.0)}$$

3.1.2.3.2.14 FUNC Statement - SUM Function

The format for the SUM function is:

$$\mathrm{SUM(\,VarName2, VarName3)}$$

where VarName2 and VarName3 are the names of single variables or an element of a vector or array included in the HEAD.ASC file.

The value of the function follows the equation:

$$\mathrm{X = VarName2 + VarName3}$$

In the following example, the variable HLE is being assigned the current value of HLE plus the sum of the current values of HBE and HTE:

$$\mathrm{FUNC\ HLE + SUM(\,HBE, HTE\,)}$$

3.1.2.3.2.15 FUNC Statement - TRI Function

The format for the TRI function is:

$$\mathrm{TRI(\,AmplitudeRange[*Factor1], Period[*Factor2])[*Factor3]}$$

where AmplitudeRange and Period are required numerical values while Factor1, Factor2 and Factor3 are optional numerical multiplication factors. The multiplication factors allow the user to modify the parameters of the TRI function while retaining a set amplitude range and period. Any combination of the factors can be included in the TRI function. If Factor1 and Factor3 are included, AmplitudeRange becomes (AmplitudeRange*Factor1*Factor3) and Period becomes (Period*Factor3).

The TRI function generates a triangular wave from the equation:

$$\mathrm{X = TRI(\,AmplitudeRange, Period, t - t_0\,)}$$

where t_0 is the time at which the function was defined. AmplitudeRange is the amplitude of the function and Period is the period of the function. The following information is provided on the function:

$$\mathrm{-AmplitudeRange \leq Amplitude \leq AmplitudeRange}$$
$$\mathrm{0 \leq Time \leq (N/2) * Period}$$

where N is the number of periods needed to evaluate the TRI function at the simulation time. If the value of the amplitude range is 0.0, the CADAC program will load the value of the controlling variable at the time the function is defined for the amplitude.

In the following example, the variable HLE is being assigned the value generated by the TRI function with the parameter values 20.0 and 3.1419:

$$\text{FUNC HLE} = \text{TRI}(10*2.0,3.1419)$$

3.1.2.3.2.16 FUNC Statement - UNIF Function

The format for the UNIF function is:

$$\text{UNIF}(\text{Mean}[*\text{Factor1}],\text{HalfWidth}[*\text{Factor2}])[*\text{Factor3}]$$

where Mean and HalfWidth are required numerical values while Factor1, Factor2 and Factor3 are optional numerical multiplication factors. The multiplication factors allow the user to modify the parameters of the UNIF function while retaining a set mean and half width. Any combination of the factors can be included in the UNIF function. NOTE: Both Mean and HalfWidth are multiplied by Factor3; therefore, if Factor1 and Factor3 are both included, the mean becomes (Mean*Factor1*Factor3) and the half width becomes (HalfWidth*Factor3).

This function provides a uniform random variable following the computation:

$$X = \text{UNIF}(\text{HalfWidth},\text{Mean})$$

in which the function UNIF is stated as:

$$X = \text{Mean} + 2.0*\text{HalfWidth}*\text{RANF}(-0.5)$$

where RANF is a random variable between 0 and 1.0. The UNIF function returns a uniformly distributed variable on the interval:

$$(\text{Mean} - \text{HalfWidth}) \le X \le (\text{Mean} + \text{HalfWidth})$$

If the value of the half width is 0.0, CADAC will load the value of the controlling variable at the time the function is defined for the amplitude.

In the following example, the variable HLE is being assigned the current value of HLE less the value generated by the UNIF function with a mean of 20 and a half width of 1.5:

$$\text{FUNC HLE} - \text{UNIF}(10.*2.0,1.5)$$

3.1.2.3.3 IF Statement

Each trajectory is divided into segments called stages. The stages are processed one at a time; a stage must be completed prior to initiation of the next stage. The IF statement defines the stage criteria that must be satisfied for a stage completion. The format for this statement is:

IF StageCriteria 1 [OR StageCriteria 2] [! comment]

where StageCriteria1 and StageCriteria2 can take on any of the following formats:

```
VarName1 > Value1              [! comments]
VarName2 < Value1              [! comments]
VarName1 > VarName2            [! comments]
VarName1 < VarName2            [! comments]
VarName1 > VarName1 + Value2     [! comments]
VarName1 < VarName1 + Value2     [! comments]
VarName1 < VarName1 - Value2     [! comments]
VarName1 > VarName1 - Value2     [! comments]
```

VarName1 and VarName2 are the names of single variables or an element of a vector or array included in the HEAD.ASC file. VarName1 is the stage criteria variable, VarName2 is an optional test criteria variable, Value1 is a test criteria value and Value2 is a value that can be added or subtracted from the stage criteria variable. The information on the right of the comparison sign (< or >) is considered the test value. When any single criteria is satisfied, the stage completes and the next stage is initiated. If the variable equals the test value, then a stage criteria is satisfied regardless of the comparison sign.

If the format with VarName2 is used, the test value is set to the value of VarName2 at the time of the test. If the test value is the sum of the criteria variable and a Value2, the test value is calculated as the sum of the criteria value at the time the stage is defined plus Value2. It is important to understand that the variable values used to define the test value are the sampled values that exist when the test is defined, not when the test is performed.

In the following example, a staging occurs when HLE is less than or equal to HBE or when HLE is greater than or equal to 1000.0:

```
IF HLE < HBE OR HLE > 1000.0
```

3.1.2.3.4 LOAD Statement

The LOAD statement indicates the end of a set of group statements. The format is:

```
LOAD
```

This statement causes the primary trajectory to be reloaded from storage and then modified by the group statements read from INPUT.ASC. The LOAD statement must always follow a set of group statements.

3.1.2.3.5 MONTE Statement

CADAC has the ability to process the primary trajectory statements and produce the single simulation corresponding to the input data. Also within the program is a looping methodology; given a primary trajectory, the program has the ability to re-run the data. This ability is accessed through the MONTE statement. This looping ability, when used with the functions available with the variable assignment statements, provides the basis for producing a set of Monte Carlo generated data for analysis purposes. The MONTE statement initializes the multi-run variable in the CADAC executive. This variable indicates to the executive code that multiple simulations are to be calculated for the five set of primary trajectory cards.

The format of the statement is:

```
MONTE  NumofRuns
```

where NumOfRuns indicates the number of simulations that are to be produced for the given primary trajectory statement set. If NumOfRuns is 0 or 1 or if the MONTE statement is not included in the INPUT.ASC, the simulation produces a single trajectory.

The MONTE statement can also be used in conjunction with the SWEEP block. It is only valid with SWEEP block for sweep modes 0 (CONST), 1 (INCR), 2 (DECR), and 3 (INCR/DECR). It is not valid for sweep modes 4 (OUT) and 5 (ALL).

3.1.2.3.6 RANDOM Statement

The RANDOM statement allows the user to set the random seed and the method used to initialize the seed. The format of the statement is:

```
RANDOM( RandomSeed [, RandomMethod] )
```

where RandomSeed is the value used to set the random seed and RandomMethod is the method used to set the seed. The methods available for initializing the random seed are:

ONCE	-The random seed is set once when the RANDOM statement is encountered
EVERY	-The random seed is set when the RANDOM statement is processed, regardless of the number of times the statement occurs in the input and regardless of where the statement occurs in the input
RUNGROUP	-The random seed is set to F(RandomSeed,Jrun,IGROUP) when the RANDOM statement is encountered. This provides automatic initialization of the seed to different seed values based on the user entered value RandomSeed. The seed value is printed to the tabular output every time the seed is initialized
RUN	-The random seed is set to F(RandomSeed,Jrun,1) when the RANDOM statement is encountered. This provides automatic initialization of the seed to different seed values based about the user entered value RandomSeed with the same set of initial seeds being generated for each group of trajectories. The seed value is printed to the tabular output every time the seed is initialized

If RandomMethod is not included in the statement, the default method, ONCE, is used.

3.1.2.3.7 RUN Statement

The RUN statement tells the input routine in the CADAC Executive to complete the simulation based on the information that has been obtained from the input file up to this time. The format for the RUN statement is:

```
RUN
```

The reaction of the program after this trajectory has been simulated depends on the input file. If the simulation is a multi-run, then the next trajectory will be started. If the sweep methodology is being utilized, then the sweep methodology will increment the appropriate variables and test for the end of the trajectory set.

If neither the sweep methodology nor a multi-run case is being exercised, then the program reads more statements from INPUT.ASC. The program determines if a set of group statements exist on the input file, and if so, processes them accordingly until a LOAD statement or the end-of-file is encountered on the input file.

3.1.2.3.8 SAVE Statement

The SAVE statement causes the CADAC Executive to save the state of the program. The format for this statement is:

```
SAVE
```

The SAVE statement must always follow an IF statement. Only one SAVE statement is allowed in the input file INPUT.ASC. If more than one is included, CADIN issues an error message and program execution is halted.

When the SAVE statement is detected in the input data, the current program state, at the time the card is processed, is saved to a CSAVE.ASC data file. The trajectory simulation then continues as directed by the primary trajectory cards. When the primary trajectory is completed and if a multi-run or Monte Carlo case is being executed, then the appropriate set of group input statements is processed, the state data (previously saved to the CSAVE.ASC file) is restored and execution of this new trajectory continues as directed by the trajectory statements.

3.1.2.3.9 STOP Statement

The STOP statement indicates that the end of the lead statements has been reached and program execution can be stopped. The format for the statement is:

```
STOP
```

If the STOP statement is not the last statement in the INPUT.ASC input file, then it will be automatically written to the output file.

3.1.2.3.10 TITLE Statement

The TITLE statement must be the first non-comment statement. The format for this statement is:

TITLE [CADAC execution title]

where "CADAC execution title" represents a 72 character string.

Since the run title must be the first non-comment statement, the word TITLE is optional.

3.1.2.3.11 VECTOR Statement

The VECTOR statement allows the user to assign all the elements of a vector/array to a single numerical value. The format for this statement is:

VECTOR VectorName VectorValue [STAGE StageNo] [! comment]

where VectorValue is the value to which every element in VectorName is to be initialized. If the user wishes the assignment to be made in a stage other than 0, the STAGE keyword must be included followed by a stage number. Valid stage numbers are 1 through 20. Comments can be included on an assignment statement by preceeding the comment with an exclamation point (!). The sum of the number of VECTOR and VECTORV statements entered in the INPUT.ASC for stage 0 of the trajectory simulation must not exceed 10. Likewise, the sum of the number of VECTOR and VECTORV statements in the remaining stages must not exceed 5. In the following example, all the elements of the vector SBEL are being set to 0.0:

VECTOR SBEL 0.0 ! Initalize SBEL

3.1.2.3.12 VECTORV Statement

The VECTORV statement allows the user to assign individual numerical values to vectors/arrays. The format for the VECTORV statement is:

VECTORV VectorName [STAGE StageNo] [! comment]

$\text{Value}_1, \text{Value}_2, \ldots, \text{Value}_{\text{DimensionVectorName}}$

where Value_1, Value_2, etc., are the values to be assigned to the elements of VectorName. The values must be separated by at least one space or by a comma and there must be a value for each element of the vector/array. The values can be placed on a single line or on multiple lines. In addition, comment statements can be placed among the data lines. For example, if the 2x3 dimensioned array, AN_ARRAY, is being initialized, the following statements are valid:

```
VECTORV AN_ARRAY
        10.0, 15.0, 20.0
        14.0, 16.0, 18.0

VECTORV AN_ARRAY
!   (1,1)  (1,2)  (1,3)   (2,1)  (2,2)  (2,3)
    10.0   15.0   20.0    14.0   16.0   18.0

VECTORV AN_ARRAY
10.0    15.0
20.0    14.0
16.0    18.0
```

Each of the these statements initialize AN_ARRAY(1,1) to 10.0, AN_ARRAY(1,2) to 15.0, AN_ARRAY(1,3) to 20.0, AN_ARRAY(2,1) to 14.0, AN_ARRAY(2,2) to 16.0 and AN_ARRAY(2,3) to 18.0.

If the user wishes the assignment to be made in a stage other than 0, the STAGE keyword must be included followed by a stage number. Valid stage numbers are 1 through 20. Comments can be included on an assignment statement by preceeding the comment with an exclamation point (!). The total number of VECTOR and VECTORV statements entered in the INPUT.ASC for stage 0 of the trajectory simulation can not exceed 10. Likewise, the sum of the number of VECTOR and VECTORV statements in the remaining stages can not exceed 5.

3.1.2.4 Block Statements

A block statement is a group of related statements with the beginning of the block signaled by a keyword statement and the end is signaled by the keyword "END." The user can included comment statements anywhere within the block. While the spaces at the beginning of the statements within a block are not required, including them makes the INPUT.ASC easier to read. The general format for a block statement is:

```
BlockKeyword        [ ! comment ]
  Block Statement   [ ! comment ]
  Block Statement   [ ! comment ]
        .
        .
        .
  Block Statement   [ ! comment ]
END [ BlockKeyword ]
```

The following sections describe the command blocks that can be included in a INPUT.ASC file.

3.1.2.4.1 HEADER Block

The HEADER block allows the user to enter up to 5 lines of text containing up to 80 characters. If more than 5 lines are included, CADIN issues an error message and the program execution is halted. The format is:

```
HEADER[S]  [! comment ]
  Text line1
  Text line2
  Text line3
  Text line4
  Text line5
END [HEADERS]
```

3.1.2.4.2 MODULE Block

The Module block is used to select the modules that are to be executed during the specific CADAC trajectory simulation. The format for the MODULE block is:

```
MODULE[S]
  ModuleName  [module definition]
  ModuleName  [module definition]
          .
          .
          .
  ModuleName  [module definition]
END [MODULES]
```

where ModuleName is the name of the module being selected. The words “module definition” may be included to clarify the statement, but if is not required. Table 2 lists all the modules available for the MODULE block and provides information on the module names and a typical description of the module. It is the user's responsibility to include a module block following the TITLE statement. Only one module is selected per statement, with multiple module assignment statements defining the list of modules to be executed. If an invalid module name is entered, an error message is issued in the ERROR.ASC. Examples of MODULE blocks can be seen in Figures 1 through 6.

The modules are executed in the same order as entered in the INPUT.ASC; therefore it is very important to sequence the statements properly. For instance, if a module outputs variables required as input by another module, then the module assignment statement for the module producing the output must be placed in the MODULE block before the module assignment statement for the module requiring the data as input.

The modules of program CADAC perform the aerodynamic calculations required for aircraft/missile flight trajectory simulation. These modules are available for the user to modify in order to meet the specific simulation requirements. Each module usually consists of at least 2 subroutines:

1) Variable initialization subroutine which is executed at the beginning of a trajectory simulation
2) The computational subroutine which is executed at each integration step

The computational subroutine may be comprised of one or more smaller modules to provide code structure and efficiency within the module.

3.1.2.4.3 SWEEP Block

The SWEEP block allows the user to generate launch envelopes and footprints. Only one of these blocks are allowed in the INPUT.ASC and the block is only valid in CADAC when the user includes the SWEEP keyword in the HEAD.ASC option list. The general format for the SWEEP block is as follows:

```
SWEEP
  MODE      ModeType
  LIMIT     CritVal < CritValVarName [ < CritMax ]
  NUM       NumTraj/NumBinS
  RANGE     RanMin < RanVar < RanMax             DELT=RanDel
  ANGLE     AngMin < AngVar < AngMax [DEG]       DELT=AngDel
END [SWEEP]
```

The statements can be placed in the block in any order, but only one of each statement type can be included. If more than one statement type is entered, CADIN issues an error message and the program execution is halted. Likewise, if a necessary statement is not included, then an error message is given and the program execution is halted. An example of a SWEEP block is illustrated in Figure 5. The information shown in [ ]'s is optional for all statements.

The user enters the SWEEP methodology mode by using the **MODE** statement. This statement must be included in the SWEEP block. ModeType is the SWEEP mode and has one the following values:

CONST[ANT DELTA]	- Mode 0 – Coarse Fixed Grid, Monte Carlo
INCR[EASING DELTA]	- Mode 1 – Fine Outer Grid, Monte Carlo
DECR[EASING DELTA]	- Mode 2 – Fine Outer Grid, M.C.
INCR/DECR[EASING DELTA]	- Mode 3 – Fine Inner and Outer Grid, M.C
OUT[ER BOUNDARY]	- Mode 4 – Search for Single Boundary
ALL [BOUNDARIES]	- Mode 5 – Search for Multiple Boundaries

The **LIMIT** statements can be used for all mode types, however, all parameters are not necessary for each mode. The CritVal parameter is not needed for sweep modes 0 through 3, therefore, the value has no meaning. However, for these modes, a zero must be entered as a placeolder:

```
LIMIT 0 < CritValVarName < [CritMax]
```

For modes 4 and 5, CritVal is the testing criteria used to search for a boundary within the footprint or launch envelope. The parameter CritValVarName is the name of the single variable or an element of a vector or array included in the HEAD.ASC file. In this statement, CritValVarName is the variable used in the sweep methodology testing. The user may also enter the CritMax value for the testing criteria. If the trajectory ends abnormally (LCONV>2), the value of CritMax, LCONV, and TRCOND will be inserted into the IMPACT.ASC file if in HEAD.ASC the user inserts the lines:

```
 *  1772    TRCOND
 *  1805    CRITMAX
 *I 2020    LCONV
```

Note: TRCOND must be present in the HEAD.ASC file, but currently does not affect the SWEEP methodology.

If the user does not enter the CritMax value it defaults to 9.0×10^5. Note: The Graphics SWEEP Analysis application uses LCONV and CritMax to display failed trajectories in Monte Carlo CADAC sweep cases. TRCOND is used to display failed trajectories in deterministic CADAC sweep cases. TRCOND identifies the user defined termination condition (defined in the A1I module and activated by setting the variable MSTOP to 1) which can be written to the TABOUT.ASC and IMPACT.ASC files. The termination code for a particular aimpoint is viewable through the SWEEP application if:

1) the CADAC sweep case is deterministic (i.e. MONTE statement not used)
2) TRCOND is present in the IMPACT.ASC file
3) The value of CritMax is set equal to 0 through the use of the LIMIT statement
4) The variable being contoured is miss distance
5) The trajectory terminated with LCONV = 4 (if LCONV = 3,5,6 or 7 the terminiatin code “T10” is displayed

The **NUM** statement has a different meaning for each mode type (see Table 3); it can either indicate the number of trajectories computed along a ray or represent the number of binary searches. The **NUM** statement is not required for mode 0 and is ignored if included. For mode 0 the NumTraj (number of trajectories) is determined internally by the following equation:

NumTraj = ((RanMax – RanMin) / RanDel) + 1

where these parameters are defined by the RANGE command.
The NumTraj and NumBinS parameter is a numerical parameter as described in Table 3.

Table 3. NUM Definitions.

MODE	DEFINITION
0: CONSTANT	NumTraj*
1: INCREASING RANGE in decreasing step size	NumTraj*
2: DECREASING RANGE in decreasing step size	NumTraj*
3: INCREASING/DECREASING step size	NumTraj*
4: OUTER boundary test	NumBinS†
5: ALL boundaries test	NumBinS†

*: Number of trajectories

†: Number of binary searches

The **RANGE** statement allows the user to define the inner variable of the sweeping grid. This statement must be included in the SWEEP block. The RanMin parameter is the minimum value that the inner variable is to assume, RanMax is the maximum value that the inner variable is to assume and RanVar is the name of the variable that is to be used as the inner variable and must be present in HEAD.ASC. If the sweep mode is CONST, OUT or ALL, then the user must include the DELT keyword with a value for the step size (RanDel) for the inner variable. DELT is not required for other modes and therefore ignored.

The **ANGLE** statement allows the user to define the outer variable of the sweeping grid. This statement must be included in the SWEEP block. The AngMin parameter is the minimum value that the outer variable is to assume, AngMax is the maximum value that the outer variable is to assume and AngVar is the name of the variable that is to be used and must be present in HEAD.ASC. AngDel is the step size for the outer variable. The default unit for the angle is radians. If the user wishes to enter the angle data in degrees, the optional keyword DEG must be included after the angle limits and before the DELT keyword.

The MONTE statement can also be used in conjunction with the SWEEP block. It is only valid with SWEEP block for sweep modes 0 (CONST), 1 (INCR), 2 (DECR), and 3 (INCR/DECR). It is not valid for sweep modes 4 (OUT) and 5 (ALL). By using the MONTE statement with the SWEEP block, Monte Carlo trajectories will be generated for each aimpoint.

3.1.2.4.3.1 MODE: <u>CONST</u>ANT DELTA (Mode 0)

For this mode, CADAC uses a constant delta to increment the range variable and, like always, rays of constant angular separation to map out the aimpoints of the sweeping grid. Applications are for 'first cut' deterministic (single trajectory against each aimpoint) launch envelopes or for Monte Carlo (multiple trajectories against each aimpoint) runs with post-processing by the SWEEP graphics analysis package. The constant range delta is based on the number of trajectories along the inner variable. The number of trajectories (NUMR) is calculated internally by the following equation:

NUMR = ((RanMax – RanMin) / RanDel) + 1

In this option, the inner and outer incrementing is stopped if the current value exceeds the respective maximums entered by the user.

The user enters sweep data using the following keywords for sweep mode CONST:

```
SWEEP
  MODE        CONST
  LIMIT       0 < CritValVarName < [CritMax]
  RANGE       RanMin < RanVar < RanMax              DELT=RanDel
  ANGLE       AngMin < AngVar < AngMax [DEG]        DELT=AngDel
END[SWEEP]
```

The MODE statement CONST indicates 'constant range delta'. The critical variable name (like miss distance), CritValVarName, is entered using the LIMIT statement. This parameter is needed to indicate which variable obtains the CritMax value when a sweep trajectory ends abnormally (i.e. LCONV > 2). The value of CritMax, LCONV, and TRCOND will be inserted into the IMPACT.ASC file if in HEAD.ASC the user inserts the lines:

```
 *  1772    TRCOND
 *  1805    CRITMAX
 *I 2020    LCONV
```

Note: TRCOND must be present in the HEAD.ASC file, but currently does not affect the SWEEP methodology.

If the user does not enter the CritMax value it defaults to 9.0x10^5. CritVal has been assigned the dummy value zero. The RANGE statement is used to indicate the inner variable (RanVar) minimum (RanMin) and maximum (RanMax) values along with the inner variable step size (RanDel). The ANGLE statement is used to indicate the outer variable (AngVar) minimum (AngMin) and maximum (AngMax) values along with the outer variable step size (AngDel). The default unit for the outer variable is radians; if the user enters the outer variable in degrees, the DEG parameter must be entered as shown.

3.1.2.4.3.2 MODE: INCREASING RANGE in Decreasing Step Size (Mode 1)

For this option, CADAC uses a varying delta to increment the range variable and, like always, rays of constant angular separation to map out the aimpoints of the sweeping grid. The range is initialized to the midpoint between the minimum and maximum range values used on the RANGE statement. The range is then incremented so that the delta between range values decreases as the range variable approaches it's maximum. Applications refer to investigations of outer boundaries with a refining grid structure towards the outer areas. The delta for each trajectory is calculated internally based on the user input NUM= NumTraj:

$$SR_i = R_{mid} + \sqrt{R_{mid}{}^2\left(1+\frac{2}{N}\right) + SR_{i-1}{}^2 - 2 * R_{mid} * SR_{i-1}}$$

where

SR_i	=	the range delta for the trajectory
R_{mid}	=	the midpoint between the user entered minimum (RanMin) and maximum (RanMax) value for the range
N	=	NumTraj + 2
SR_{i-1}	=	the range delta that was used in the previous trajectory simulation

In this option, the range incrementing is stopped when the number of trajectories NUM has been executed along the current ray. The angle incrementing stops if the current value exceeds the maximum entered by the user.

The user enters sweep data using the following keywords for input for sweep mode INCRE:

```
SWEEP
  MODE       INCR
  LIMIT      0 < CritValVarName < [CritMax]
  RANGE      RanMin < RanVar < RanMax
  ANGLE      AngMin < AngVar < AngMax [DEG]      DELT=AngDelt
  NUM        NumTraj
END[SWEEP]
```

The MODE statement INCR indicates a sweep mode of 1 with decreasing step size in range towards the outer boundary. The critical variable name, CritValVarName, is entered using the LIMIT statement. This parameter is needed to indicate which variable obtains the CritMax value when a sweep trajectory reaches an incorrect end (i.e. LCONV > 2). The value of CritMax, LCONV, and TRCOND will be inserted into the IMPACT.ASC file if in HEAD.ASC the user inserts the lines:

```
 *  1772    TRCOND
 *  1805    CRITMAX
 *I 2020    LCONV
```

Note: TRCOND must be present in the HEAD.ASC file, but currently does not affect the SWEEP methodology.

CritVal has been assigned the dummy value zero. If the user omits the CritMax parameter, CritMax is set to 9.0×10^5. The RANGE statement is used to indicate the inner variable (RanVar) minimum (RanMin) and maximum (RanMax). The RANGE DELT parameter is not used for this sweep mode and therefore ignored if included. The ANGLE statement is used to indicate the outer variable (AngVar) minimum (AngMin) and maximum (AngMax) values along with the outer variable step size (AngDel). The default units for the outer variable are radians; if the user enters the outer variable in degrees, the DEG parameter must be entered as shown. The number of trajectories (NumTraj) produced along a ray is entered using the NUM statement.

3.1.2.4.3.3 MODE: DECREASING RANGE in Increasing Step Size (Mode 2)

For this option, CADAC uses a varying delta to increment the range variable and, like always, rays of constant angular separation to map out the aimpoints of the sweeping grid. The range variable is initialized to the maximum value entered by the user using the RANGE statement. It then decrements towards the midpoint with increasing range intervals and stops there. This mode generates the same grid as mode 1. It only differs in the direction the rays are traversed. The delta for each trajectory is calculated internally based on the user input NUM= NumTraj:

$$SR_i = R_{mid} - \sqrt{R_{mid}{}^2\left(1-\frac{2}{N}\right) + SR_{i-1}{}^2 - 2 * R_{mid} * SR_{i-1}}$$

where

SR_i	=	the range delta for the trajectory
R_{mid}	=	the midpoint between the user entered minimum (RanMin) and maximum (RanMax) value for the range
N	=	NumTraj + 2
SR_{i-1}	=	the range delta that was used in the previous trajectory simulation

In this option, the range incrementing is stopped when the number of trajectories NUM has been executed along the current ray. The angle incrementing stops if the current value exceeds the maximum entered by the user.

The user enters sweep data using the following keywords for input for sweep mode DECR:

```
SWEEP
  MODE       DECR
  LIMIT      0 < CritValVarName < [CritMax]
  RANGE      RanMin < RanVar < RanMax
  ANGLE      AngMin < AngVar < AngMax [DEG]       DELT=AngDel
  NUM        NumTraj
END[SWEEP]
```

The MODE statement DECR parameter indicates a sweep mode of 2, decreasing range in increasing step size until midpoint. The critical variable name, CritValVarName, is entered using the LIMIT statement. This parameter is needed to indicate which variable obtains the CritMax value when a sweep trajectory reaches an incorrect end (i.e. LCONV > 2). The value of CritMax, LCONV, and TRCOND will be inserted into the IMPACT.ASC file if in HEAD.ASC the user inserts the lines:

```
 *  1772    TRCOND
 *  1805    CRITMAX
 *I 2020    LCONV
```

Note: TRCOND must be present in the HEAD.ASC file, but currently does not affect the SWEEP methodology.

If the user does not enter the CritMax value it defaults to $9.0x10^5$. CritVal has been assigned the dummy value zero. The RANGE statement is used to indicate the inner variable (RanVar) minimum (RanMin) and maximum (RanMax). The RANGE DELT parameter is not used for the sweep mode and therefore ignored if included. The ANGLE statement is used to indicate the outer variable (AngVar) minimum (AngMin) and maximum (AngMax) values along with the outer variable step size (AngDel). The default units for the outer variable are radians; if the user enters the outer variable in degrees, the DEG parameter must be entered as shown. The number of trajectories (NumTraj) produced along a ray is entered using the NUM statement.

3.1.2.4.3.4 MODE: INCREASING/DECREASING Step Size (Mode 3)

For this option, CADAC uses increasing and decreasing range deltas and, like always, rays of constant angular separation to map out the aimpoints of the sweeping grid. The range is initialized to the minimum value as entered by the RANGE statement. The range is then incremented in increasing range deltas until the midpoint of the ray is reached and thereafter the range delta is decreased until the maximum range is reached. Applications refer to the investigation of inner and outer boundaries simultaneously. The delta for each trajectory is calculated internally based on the user input NUM= NumTraj:

$$SR_i = R_{mid} - \sqrt{R_{mid}{}^2\left(1-\frac{2}{N}\right) + SR_{i-1}{}^2 - 2*R_{mid}*SR_{i-1}}$$

where

SR_i	=	the range delta for the trajectory
R_{mid}	=	the midpoint between the user entered minimum (RanMin) and maximum (RanMax) value for the range
N	=	NumTraj + 2
SR_{i-1}	=	the range delta that was used in the previous trajectory simulation

The equation used to compute SR_i is utilized until the number of trajectories reaches:

(NumTraj +1)/2.

When the above criteria is satisfied, the midpoint is assumed to be near or achieved. At which time the following equation is utilized for the range delta from the range midpoint to the range maximum:

$$SR_i = R_{mid} + \sqrt{R_{mid}{}^2\left(1+\frac{2}{N}\right) + SR_{i-1}{}^2 - 2*R_{mid}*SR_{i-1}}$$

In this option, the range incrementing is stopped when the number of trajectories NUM has been executed along the current ray. The angle incrementing stops if the current value exceeds the maximum entered by the user.

The user enters sweep data using the following keywords for input for sweep mode INCR/DECR:

```
SWEEP
  MODE        INCR/DECR
  LIMIT       0 < CritValVarName < [CritMax]
  RANGE       RanMin < RanVar < RanMax
  ANGLE       AngMin < AngVar < AngMax [DEG]        DELT=AngDelt
  NUM         NumTraj
END[SWEEP]
```

The MODE statement INCR/DECR parameter indicates a sweep mode of 3, increasing/decreasing step size. The critical variable name, CritValVarName, is entered using the LIMIT statement. This parameter is needed to indicate which variable obtains the CritMax value when a sweep trajectory reaches an incorrect end (i.e. LCONV > 2). The value of CritMax, LCONV, and TRCOND will be inserted into the IMPACT.ASC file if in HEAD.ASC the user inserts the lines:

```
*  1772    TRCOND
*  1805    CRITMAX
*I 2020    LCONV
```

Note: TRCOND must be present in the HEAD.ASC file, but currently does not affect the SWEEP methodology.

If the user does not enter the CritMax value it defaults to 9.0×10^5. CritVal has been assigned the dummy value zero. The RANGE statement is used to indicate the inner variable (RanVar) minimum (RanMin) and maximum (RanMax). The RANGE DELT parameter is not used for the sweep mode and therefore ignored if included. The ANGLE statement is used to indicate the outer variable (AngVar) minimum (AngMin) and maximum (AngMax) values along with the outer variable step size (AngDel). The default units for the outer variable are radians; if the user enters the outer variable in degrees, the DEG parameter must be entered as shown. The number of trajectories (NumTraj) produced along a ray is entered using the NUM statement.

3.1.2.4.3.5 MODE: <u>OUT</u>ER BOUNDARY TEST (Mode 4)

This mode lays down a sweeping grid of aimpoints like mode 0, and thereafter refines the grid near the outer boundary through a binary search on the so-called critical variable, CritValVarName, as provided by the LIMIT statement. The number of binary searches, NumBinS, is input by NUM. Applications refer to determining precisely those launch conditions that produce certain miss distance boundaries. The CritVal variable as input by the LIMIT statement in the SWEEP block contains the value that the output variable is to achieve to trigger a binary search.

To find the neighborhood of the desired boundary, the constant search grid is laid down. The number of trajectories, NUMR, to be performed along a ray is calculated internally by using the range maximum, minimum and delta values as input by the RANGE statement:

NUMR = ((RanMax – RanMin)/ RanDel) + 1

The range value is initialized to the RanMin and incremented using RanDel. NUMR trajectories are thus calculated with the CritValVarName values written to a scratch file. These values are searched backwards until the first value is encountered that is less than CritVal. This aimpoint is then used as a starting point for a refined binary search. The user must specify the number of binary search trajectories, NumBinS, using the NUM statement. After the boundary point has been identified (or even if no binary search was triggered) the angle variable is incremented by AngDel and the next ray is analyzed. The angle incrementing stops if the current value exceeds the maximum entered by the user.

The user enters data using the following keywords for mode OUT:

```
SWEEP
  MODE      OUT
  LIMIT     CritVal < CritValVarName < [CritMax]
  RANGE     RanMin < RanVar < RanMax              DELT=RanDel
  ANGLE     AngMin < AngVar < AngMax [DEG]        DELT=AngDel
  NUM       NumBinS
END[SWEEP]
```

The MODE statement OUT indicates a refined outer boundary search. The critical variable name, CritValVarName, and its CritVal value are entered using the LIMIT statement. If the trajectory ends abnormally (LCONV>2), the value of CritMax, LCONV, and TRCOND will be inserted into the IMPACT.ASC file if in HEAD.ASC the user inserts the lines:

```
 *  1772    TRCOND
 *  1805    CRITMAX
 *I 2020    LCONV
```

Note: TRCOND must be present in the HEAD.ASC file, but currently does not affect the SWEEP methodology.

If the user does not enter the CritMax value it defaults to $9.0\text{x}10^5$. The RANGE statement is used to indicate the inner variable (RanVar) minimum (RanMin) and maximum (RanMax). The RANGE DELT parameter is used to indicate the inner variable step size. The ANGLE statement is used to indicate the outer variable (AngVar) minimum (AngMin) and maximum (AngMax) values along with the outer variable step size (AngDel). The default units for the outer variable are radians; if the user enters the outer variable in degrees, the DEG keyword must be entered as shown. The number of binary searches (NumBinS) performed to locate the outer boundary is entered using the NUM statement

3.1.2.4.3.6 MODE: ALL BOUNDARIES TEST (Mode 5)

This option is similar to sweep mode OUT, except that it can execute more than one binary search to find multiple boundaries. A sweeping grid of aimpoints is laid down as in mode 0, and refined as boundaries are encountered along the ray through binary searches on the so-called critical variable, CritValVarName, as provided by the LIMIT statement. The number of binary searches, NumBinS, is given by NUM. Examples refer to applications where multiple boundaries are expected. The CritVal variable as input by the LIMIT statement in the SWEEP block contains the value that the output variable is to achieve to trigger a binary search.

To find the neighborhood of the desired boundary, the constant search grid is laid down. The number of trajectories, NUMR, to be performed along a ray is calculated internally by using the range maximum, minimum and delta values as input by the RANGE statement:

NUMR = ((RanMax – RanMin)/ RanDel) + 1

The range value is initialized to the RanMin and incremented using RanDel. NUMR trajectories are thus calculated with the CritValVarName values written to a scratch file. These values are checked as each trajectory is generated. The first boundary is encountered when the critical variable is less than the critical value. This aimpoint is then used as a starting point for a refined binary search. Once the binary search is completed, new trajectories are generated checking the critical variable after each. If the value should again become greater than CritVal a second binary search is conducted. The number of binary search trajectories, NumBinS, is the same in each case and is specified by the NUM statement. This process continues along the ray until RanMax is reached. Then the angle variable is incremented by AngDel and the next ray is analyzed. The angle incrementing stops if the current value exceeds the maximum entered by the user.

This procedure will in effect determine the locations of an inner boundary (determined by the first binary search process), a hole/bayou location (determined by the second and third binary search process) and an outer boundary (determined by the last binary search process) as illustrated in Figure 7.

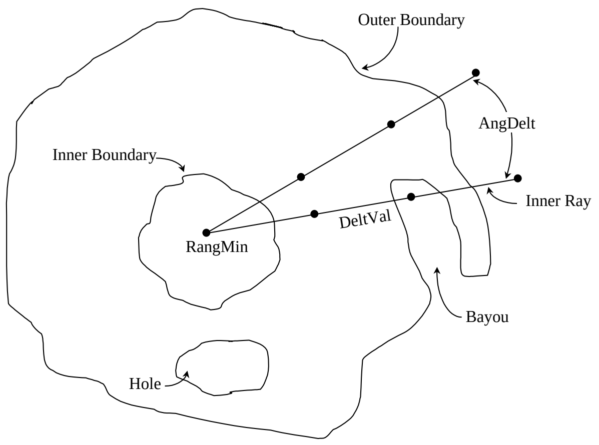

Figure 7. Sweep Boundaries.

The user enters data using the following keywords for mode ALL:

```
SWEEP
  MODE ALL
  LIMIT      CritVal < CritValVarName < [CritMax]
  RANGE      RanMin < RanVar < RanMax              DELT=RanDel
  ANGLE      AngMin < AngVar < AngMax [DEG]        DELT=AngDel
  NUM        NumBinS
END[SWEEP]
```

The MODE statement ALL indicates a refined all boundary search. The critical variable name, CritValVarName, and its CritVal value are entered using the LIMIT statement. If the trajectory ends abnormally (LCONV>2), the value of CritMax, LCONV, and TRCOND will be inserted into the IMPACT.ASC file if in HEAD.ASC the user inserts the lines:

```
 *  1772    TRCOND
 *  1805    CRITMAX
 *I 2020    LCONV
```

Note: TRCOND must be present in the HEAD.ASC file, but currently does not affect the SWEEP methodology.

If the user does not enter the CritMax value it defaults to $9.0x10^5$. The RANGE statement is used to indicate the inner variable (RanVar) minimum (RanMin) and maximum (RanMax). The RANGE DELT parameter is used to indicate the inner variable step size. The ANGLE statement is used to indicate the outer variable (AngVar) its minimum (AngMin) and maximum (AngMax) values along with the outer variable step size (AngDel). The default units for the outer variable are radians; if the user enters the outer variable in degrees, the DEG keyword must be entered as shown. The number of binary searches (NumBinS) performed to locate a boundary is entered using the NUM statement.

3.1.2.4.4 VECTOR Block

The format for the VECTOR block is:

```
VECTOR  VectorName             [STAGE StageNo] [! comment]
   Value or Function or VarName   [STAGE StageNo] [! comment]
   Value or Function or VarName   [STAGE StageNo] [! comment]
              .
              .
              .
   Value or Function or VarName   [STAGE StageNo] [! comment]
END [VECTOR]
```

The VECTOR block allows the user to assign consecutive elements of a vector/array to values, functions, other variables or a single element of a vector or array. The following example illustrates the VECTOR block. If AN_ARRAY is a 2x3 array and the following VECTOR block is included in the INPUT.ASC:

```
VECTOR AN_ARRAY   STAGE 2
 12.3
 RAYLE( 5.0 ) *2.0              STAGE 3
 15.0                           stage 4
 SBEL(1)
END VECTOR
```

then AN_ARRAY(1,1) will be assigned the value 12.3 in stage 2; AN_ARRAY(1,2) will be assigned a value generated by the Rayleigh function with the parameter 10 in stage 3; AN_ARRAY(1,3) will be assigned the value 15.0 in stage 4; and AN_ARRAY(2,1) will be set to SBEL(1) in stage 2. NOTE: An assignment statement does not have to be included for each element of the array. The user can specify the stage either on the block statement line or on the assignment statement. If a stage is set on both statements, the element is set in the stage from the assignment statement.

3.1.2.4.5 WEATHER Block

The WEATHER block allows the user to input weather data for the atmospheric modules. The format for this block is:

```
WEATHER
     Alt    WindDir     WindVel     AirDens     Temp   Pressure
     Alt    WindDir     WindVel     AirDens     Temp   Pressure
            .
            .
            .
     Alt    WindDir     WindVel     AirDens     Temp   Pressure
END [WEATHER]
```

where Alt, WindDir, WindVel, AirDens, Temp and Pressure are numerical values for altitude, wind direction, wind velocity, air density, temperature and pressure respectively. If the density, temperature and pressure are not going to be used, then zeros must be entered in their place as place holders. The numbers must be separated by at least one space or a comma. Up to 50 records of weather data may be entered. If more than 50 records are entered, an error message is given in the ERROR.ASC file. The data must be arranged such that altitude increases from one record to the next.

The units of the weather data are dependent on the flag OPTMET. This flag indicates the model units: English (OPTMET = 0) and Metric (OPTMET = 1). The units of each data are shown in Table 4 for both unit systems.

Table 4. Weather Parameter Units.

Parameter	English Units (OPTMET=0)	Metric Units (OPTMET=1)
Altitude	Feet (ASL)	Meter (ASL)
Wind Direction	Degrees	Degrees
Wind Velocity	Feet/Second	Meter/Second
Air Density	Slugs/Feet^3	Kilograms/Meter^3
Temperature	Degrees F	Degrees C
Pressure	Pounds/Feet^2	Pascals

An example of a WEATHER block is:

```
WEATHER
!       Alt      WindDir    WindVel      AirDens      Temp        Pressure
        0.0      1206.5       0.0          212.0       0.0          667.8
     1640.0      1137.6       0.0          220.0       0.0          668.2
                 .
                 .
                 .
    91840.0        25.6       0.0          255.0       0.0          586.5
END WEATHER
```

3.2 HEAD.ASC

HEAD.ASC is referred to as the variable definition file. This file contains the scroll variable list and the plot variable list. The scroll list is used to indicate which variable data is to be displayed on the screen or written to the tabular output file, TABOUT.ASC. The plot variable list is used to indicate the data to be placed on the plot files, TRAJ.BIN/ASC, INIT.ASC/BIN, TRACK.ASC/BIN, STAT.BIN/ASC and to provide the user with a list of variable definitions. By defining the variables in the file via DFHEAD, the user can document the variables and include them in a report.

3.2.1 HEAD.ASC Development

The HEAD.ASC file can be developed by using the following procedure:

1) Create the MODULE.FOR. This is a single file containing all of the modules to be used in the CADAC simulation
2) Run program MKHEAD (see Section 3.3.1 for further details). This program produces a partial HEAD as output
3) Edit the resulting HEAD file:
   (a) The user should modify either the modules or the HEAD.ASC file and rerun MKHEAD until no error messages are produced and no flags indicating misuse appear

(b) Flag the variables to be output on the TRAJ plot file by placing an * in column 2 and indicate integer variables by placing an I in column 3 Flag the variables for output to the INIT and/or TRACK file by placing an “I” in column 1 for INIT, and “T” for TRACK, or a “B” for both
(c) Define each variable in columns 20-80 of each line using DFHEAD. An * in column 1 serves as a comment line - a definition can be continued or started on such a comment line and will be ignored by CADAC. This definition is optional
(d) Add a scroll variable list to the file; up to 16 variables can be scrolled. The scroll list precedes the definition list
(e) Edit the option list. The user may select from SCROLL/ NOSCROLL, STGMSG/ NOSTGMSG, TRAJBIN/ NOTRAJBIN, TRAJASC/NOTRAJASC, STATBIN/ NOSTATBIN, TABOUT/ NOTABOUT, INTMSG/ NOINTMSG, ECHOIN/ NOECHOIN, INITASC/ NOINITASC, INITBIN/ NOINITBIN, TRACKASC/ NOTRACKASC, TRACKBIN/ NOTRACKBIN, RANVAR/ NORANVAR or SWEEP/ NOSWEEP. The keyword SCROLL / NOSCROLL must start in column 1. These keywords are explained in Section 3.2.2.1

3.2.2 HEAD.ASC Format

The HEAD.ASC input file is divided into 4 sections: the option list, the scroll variable list, the comment list and the plot variable list. Each of these sections have a distinctive format and must occur on the file in the order listed. A discussion on the format of each section follows. Figure 8 shows a sample HEAD.ASC input file.

```
SCROLL  ECHOIN  INTMSG TRAJBIN TRAJASC STATBIN NOSTATASC RANVAR TABOUT STGMSG
  2 2000        TIME
  I 0800        MGUID
  1 1330        AMASS
  0 1602        SBEL1
  0 1615        HBE
  3 0913        ALPHAP
  3 1651        THTVL
  0 0130        DBT1
  2 0835        TGOAVG
  2 0221        VMACH
*
*  A sample HEAD file for the report only
*
 *  0001        ERRVAL
 *I 0002        IERR
 *  0003        AERR
    0051        REARTH
    0052        CRAD
B   0115        ST1EL1  Position of the T1 wrt E
B   0117        ST1EL3  Position of the T1 wrt E
  I 0200        MAIR    Atmospheric selection Flag
  I 0400        MSEEK   Seeker Option Flag
       •
       •
       •
    0511        ST1CEL(3)  Target position measured by AI radar - m
T   0511        ST1CEL1  Target position measured by AI radar - m
T   0512        ST1CEL2  Target position measured by AI radar - m
T   0513        ST1CEL3  Target position measured by AI radar - m
       •
       •
       •
  I 0800        MGUID   Guidance Flag
    0802        ALCOM   Lateral acceleration command
    0803        ANCOM   Normal acceleration command
  I 0900        MAUT    Autopilot Flag
    0901        ALPHAC  Angle of attack command
    0908        ALPHA   Angle-of-attack
    0909        BETA    Sideslip angle
  I 1600        MINIT   Initialization Flag
 *  1602        SBEL1   Position of Missile rel to E in l.
 *  1603        SBEL2   Position of Missile rel to E in l.
 *  1604        SBEL3   Position of Missile rel to E in l.
    1624        RANGEL  Launch range in horizontal plane
       •
       •
       •
  I 1649        SBELS1   State variable of vehicle position – m
  I 1650        SBELS2   State variable of vehicle position – m
  I 1651        SBELS3   State variable of vehicle position – m
```

Figure 8. Sample HEAD.ASC File.

3.2.2.1 HEAD.ASC Option List

The option list in the HEAD.ASC file is a single line at the top of the file. This line consists of up to 14 keywords indicating output options available in program CADAC. These keywords are: SCROLL/ NOSCROLL, TABOUT/ NOTABOUT, ECHOIN/ NOECHOIN, INTMSG/ NOINTMSG, STGMSG/ NOSTGMSG, TRAJBIN/ NOTRAJBIN, TRAJASC/

NOTRAJASC, STATBIN/ NOSTATBIN, INITASC/ NOINITASC, INITBIN/ NOINITBIN, TRACKASC/ NOTRACKASC, TRACKBIN/ NOTRACKBIN, RANVAR/ NORANVAR or SWEEP/ NOSWEEP. The order of these keywords is not important.

The SCROLL/ NOSCROLL keyword controls the output of the scroll variables to the TABOUT file. If SCROLL is included, the variable data of the variables listed in the scroll list is displayed on the TABOUT file. If NOSCROLL is included or the keyword is omitted, the listing of data on the TABOUT file is suppressed.

The TABOUT/ NOTABOUT keyword controls the direction of the tabular output (referred to as TABOUT). If the keyword TABOUT is included in the option list, the output is directed to the file TABOUT.ASC; if NOTABOUT is included or the keyword is omitted, the output is directed to the screen.

The ECHOIN/ NOECHOIN keyword controls the display of the input card data on the TABOUT file. If ECHOIN is included, the input card deck is echoed to TABOUT. If NOECHOIN is included or the keyword is omitted, the input card deck is not echoed to TABOUT.

The INTMSG/ NOINTMSG keyword controls the display of integration messages to the TABOUT file. If INTMSG is included, the CADAC integration messages generated by the AMRK module are written to the TABOUT file. If NOINTMSG is included or the keyword is omitted, the integration messages are not written to the file.

The STGMSG/ NOSTGMSG keyword controls the display of staging messages to the TABOUT file. If STGMSG is included, the CADAC staging messages will be written to the TABOUT file. If NOSTGMSG is included or the keyword is omitted, the staging messages are not written to the TABOUT file.

The TRAJBIN/ NOTRAJBIN keyword controls the creation of a binary trajectory plot data file. If TRAJBIN is included, a binary TRAJ data file is created. If NOTRAJBIN is included or the keyword is omitted, the binary TRAJ file is not created.

The TRAJASC/ NOTRAJASC keyword controls the creation of an ASCII trajectory plot data file. If TRAJASC is included, an ASCII TRAJ data file is created. If NOTRAJASC is included or the keyword is omitted, the ASCII TRAJ file is not created.

The STATBIN/ NOSTATBIN keyword controls the creation of the binary statistics plot data file. If STATBIN is included, a binary STAT data file is created. If NOSTATBIN is included or the keyword is omitted, the binary STAT file is not created.

The STATASC/ NOSTATASC keyword controls the creation of an ASCII statistics plot data file. If STATASC is included, an ASCII STAT data file is created. If NOSTATASC is included or the keyword is omitted, the ASCII STAT file is not created.

The INITASC/ NOINITASC keyword controls the creation of an ASCII real-time CADAC initialization file INIT.ASC file. If INITASC is included, an ASCII INIT file is created. If NOINITASC is included or the keyword is omitted, the ASCII INIT file is not created.

The INITBIN/ NOINITBIN keyword controls the creation of a binary real-time CADAC initialization file INIT.BIN file. If INITBIN is included, the binary INIT file is created. If NOINITBIN is included or the keyword is omitted, the binary INIT file is not created.

The TRACKASC/ NOTRACKASC keyword controls the creation of an ASCII real-time CADAC target track data file, TRACK.ASC. If TRACKASC is included, an ASCII TRACK file is created. If NOTRACKASC is included or the keyword is omitted, the ASCII TRACK file is not created.

The TRACKBIN/ NOTRACKBIN keyword controls the creation of a binary real-time CADAC target track data file, TRACK.BIN. If TRACKBIN is included, the binary TRACK file is created. If NOTRACKBIN is included or the keyword is omitted, the binary TRACK file is not created.

The RANVAR/ NORANVAR keyword controls the creation of the RANVAR output file which contains the values selected by the random distribution generators for stochastic variables defined in INPUT. If RANVAR is included, the RANVAR output file is created. If NORANVAR is included or the keyword is omitted, the RANVAR file is not created.

The SWEEP/ NOSWEEP keyword controls the SWEEP methodology execution. If SWEEP is included, a SWEEP run is to be executed requiring the presence of the SWEEP records, type 19, 20 and 21 cards. If NOSWEEP is included or the keyword is omitted, then a single run execution will be made and the SWEEP records should not be included in the input files. If a SWEEP run is to be executed and the sweep cards are not included an error message is displayed and the run terminated. Likewise, if a single run is to be executed and the sweep cards are present, an error message is displayed and the run terminated.

3.2.2.2 HEAD.ASC Scroll Variable List

The HEAD.ASC scroll variable list is the list of variables that are to be printed to TABOUT. If SCROLL is selected on the option list, the scroll variable list must follow the option list. The scroll variable list contains up to 16 records. Each record represents one variable whose data is to be displayed to TABOUT. The format of the scroll list records is given in Table 5.

Table 5. Scroll Variable List Format.

COLUMN NUMBER	VALID VALUES	DEFINITION
1	*	Indicates end of the scroll list
	' '	Card contains a scroll variable
3		Output format indicator:
	I	Integer format
	' '	List Directed Format '*'
	1 - 5	No. of decimal places in an F format
5-8	1 - 3510	The C location of the variable to be displayed
13-20	8 Characters	The name associated with the variable

The scroll list must end with a record with an * in the first column. The records containing variable entries must contain a blank in the first column so that the end of the list is not signaled.

Column 3 contains an indicator for the format used to display the data. The user may enter the character "I", to indicate an integer format. If the user does not enter an “I” for the integer format and expects integer data, the data displayed on the TABOUT/screen may be inaccurate. When the “I” flag is entered, the format used is F8.0. If a blank is entered, the format, F8.0, is used to display the data. If an integer, N, between 1 and 5 is entered, then the integer indicates the number of digits right of the decimal to use in the display format; the format F8.N is used to display the data. If an integer greater than 5 is entered, the maximum value of 5 is used in the format. A format with greater than 5 digits to the right of the decimal with a maximum width of 8 generates an error.

The integer entered in columns 5 through 8 is the C location of the variable whose data is to be displayed. The integer must be between 1 and 3510. In columns 13 through 20, the user may enter the name of the data to be displayed as a header for the data on TABOUT. The user may enter up to 8 characters.

3.2.2.3 HEAD.ASC Comment List

The comment list follows the scroll variable list on the HEAD.ASC file. The comment list is for the user's benefit only; the data entered on the records are ignored by the CADAC program. The comment records have an asterisk (*) in column 1 and the user's comment in columns 2 through 80. The user may enter an unlimited number of comments in the HEAD.ASC file or none at all. However, the user should not confuse the last record of the scroll variable list with a comment record. The last record of the scroll variable list is a blank record with an asterisk in column 1; this record must occur as a part of the scroll variable list even if no comment records are entered.

3.2.2.4 HEAD Plot Variable List

Following the comments list is the plot variable list. This list is used to indicate the variable data to be written to the plot output files; TRAJ, STAT, INIT, and TRACK. The MKHEAD program output provides a basis for creating the plot variable list. The output provided by MKHEAD is then modified depending on the user's intentions. The user may enter as many records in the plot variable list as desired; however, only 70 variables may be selected for display on the plot output files. The end of the file signals the end of the plot variable list.

The first column is used to indicate whether the record contains a user-entered comment a plot variable definition, or a real-time CADAC output variable. If the first column contains an asterisk, the record must contain a comment. The comments are for the user's benefit only; program CADAC ignores the data entered on these lines. The format of the comments follows the same format as those in the comments list. If the first character is a blank, then the record contains a plot variable definition. If the first column contains an "I", the variable will be output to the target track initialization data file. If the first column contains a "T", the variable will be output to the target track time history data file. If the first column contains a "B", the variable will be output to both the target track initialization, and traget track time history data files. These records follow the format shown in Table 6.

If the record contains a plot variable, then the second column of the record is used to indicate whether the data for that variable is to be written to the TRAJ and STAT plot files. An asterisk in this column selects the data to be written to the plot files. A space indicates the data is not written to the plot files.

The third column of a record containing a plot variable is used to indicate the format of the variable. This column is used by program CADAC. If an "I" is placed in the column, the variable data is treated as integer. If the user does not enter an “I” for the integer format and expects integer data, the data written to the plot files may be inaccurate. In program CADAC, any other value causes the variable data to be treated a real data.

Table 6. Plot Variable List Format.

COLUMN NUMBER	VALID VALUES	DEFINITION
1	*	Indicates a comment card
	' '	Card contains plot variable data
	I	Variable output to target track initialization data file
	T	Variable output to target track time history data file
	B	Variable output to both target track initialization and time history data files
2	*	Variable output to trajectory and/or statistical plot data file
	' '	Do not write data to the plot file
3		Output format indicator:
	I	Integer format
	' '	Listed directed format '*'
	0 - 6	No. of decimal places in an F format
5 - 8	1 - 3510	The C location of the variable
13 - 20	Character	The name associated with the variable
21 - 80	Character	A comment defining the variable

Columns 5 through 8 are used to indicate the C location of the plot variable and columns 13 through 20 are used to indicate the name associated with the C location. Columns 21 through 80 may then be used to enter the definition of the C location. This definition is not used by program CADAC; however, this definition does provide documentation for the simulation.

3.3 CADAC Utilities for Input Creation and Documentation

3.3.1 MKHEAD

The MKHEAD utility reads the CADAC modules and searches for variable names that have been equivalenced to C array locations. Once a variable name has been determined, the name and the associated C array location are written to an output file. When a variable is an array, MKHEAD will record the dimensions to the output file. NOTE: The dimensioning of the array must be done in a DIMENSION statement, not a REAL or INTEGER statement. The utility also checks for conflicts between variables and indicates trouble spots. Conflicts are defined by the following cases: (1) when the same variable name is equivalenced to multiple C array locations, (2) when the same C array location is equivalenced to multiple variable names, (3) when the number of dimensions in the EQUIVALENCE and DIMENSION statements do not match and (4) when an array variable overlaps its C array locations with another variable. These trouble spots are indicated by messages to the screen and conflicts of type 1 and 2 are flagged in the output file. Execution of MKHEAD and the output file are discussed in the following sections.

3.3.1.1 MKHEAD Execution

Program MKHEAD is a DOS application that can be added to the DIGITAL Visual FORTRAN TOOL option or can be executed in a DOS window by entering the following statement at the DOS prompt (version number must be included, ex: MKHEAD2):

```
C:>\CAD\TEST\MKHEAD
```

When program MKHEAD is executed, the user is informed of the current directory and is prompted for the name of the file containing the CADAC modules:

```
Current Directory is :
C:\CAD\TEST

Enter name of modules file (input)
Default = MODULE.FOR :
```

The user may enter a filename and press <RETURN> or select the default filename by pressing <RETURN>. If an invalid filename is entered, the user receives the message:

```
* * * Invalid Filename * * *
  Do you wish to continue? (Y or N)
  Default = Y :
```

If the user selects to continue execution (by entering a "y" and pressing <RETURN> or by simply pressing <RETURN>), the user is prompted again for the file containing the CADAC modules, otherwise program execution is terminated.

Once a valid CADAC modules file is entered, MKHEAD will proceed to locate variables and information associated with each variable, storing the data internally until all the variables have been determined. While MKHEAD is executing, the following messages are displayed on the screen, one at a time, informing the user of the status of the program:

```
Initializing index files

Extracting DIMENSION variables and EQUIVALENCE variables

Eliminating duplicate EQUIVALENCE variables

Sorting the EQUIVALENCE variables in ascending order

Checking for different C locations using same variable name

Including dimensions for variables
```

Once all the variables have been located the following prompt appears:

Do you wish to save the variables to a file? (Y or N)
Default = Y :

If the user selects to create a variable file, the following prompt appears on the screen:

Current Directory is :
C:\CAD\TEST

Enter name of file to contain variables (output)
Default = HEAD.ASC :

The user may enter a name for the HEAD file followed by <RETURN> or the user may use the default by simply pressing <RETURN>. Regardless of whether the variable file is created, the user is then given the message:

POTENTIAL PROBLEMS IN HEAD.ASC ARE LISTED BELOW

Followed by potential problems if any exist. The user is then prompted:

Press RETURN to continue

This prompt allows the user to view the potential problems before the DOS window is exited.

3.3.1.2 MKHEAD Output File

The output file produced by MKHEAD is the basis of the HEAD.ASC file used by CADAC as described in Section 3.2 of this report. The format of the file resulting from the MKHEAD execution is as follows: record 1 contains the option list, record 2 contains an asterisk * in column 1; the remaining records contain a C array location in columns 5-8, a flag (A) for multiple C array locations with same variable name in column 10, a flag (*) for multiple variable names for a C array location in column 11 and the variable and dimensions in columns 13-25. A sample HEAD.ASC generated by MKHEAD is shown in Figure 9.

```
SCROLL NOECHOIN NOINTMSG NOSTGMSG NORANVAR TRAJBIN NOTRAJASC NOSTATBIN NOSTATASC NOTABOUT
*
    51    REARTH
    52    CRAD
    54    AGRAV
   201    TEMPK
   202    PRESS
   203    RHO
   204    VSOUND
   205    GRAV
   206    VMACH
   207    PDYNMC
   301    CK
   310    Q0D
   311    Q0
          •
          •
          •
   347    ETBL
   348    TLB(3,3)
   451 A  EVT1EL(3)
   802    ANCOM
   803    ALCOM
   825 A  EVT1EL(3)
   900    MAUT
   901    MFREEZE
   902    ACTBW
   903    PACL
   904    WACL
   905    ZACL
   906    ANLIM
   907    ALLIM
   908    DQLIMX
   909    DRLIMX
   910    DPLIMX
   911    PHICOMX
   912    WRCL
   913    ZRCL
   914    YYD
   915    YY
   916    ZZD
   917    ZZ
   919    DPCX
   920    DQCX
   921    DRCX
   922    DETP
   923    DETDUM
   924    GAINFB(3)
          •
          •
          •
  1710    PX
  1711    QX
  1712    RRX
  1713    WBEB(3)
  2000  * TIME
  2561    NIP
  2562    IPL(100)
  2866    ICOOR
  2867    IPLV(100)
```

Figure 9. Sample HEAD.ASC Generated by MKHEAD.

3.3.2 DFHEAD

The DFHEAD utility documents a HEAD.ASC type file by taking the definitions of variables found in the CADAC modules and writing them to the HEAD.ASC file. The HEAD.ASC file may be in the form used by CADAC or the form produced by MKHEAD. The program reads the CADAC modules and searches comments for the following string: " = ". When the designated string is found, the variable name to the left of the string is matched with the same variable name in the HEAD.ASC. The definition that appears to the right of the string is then stored internally by DFHEAD in conjunction with the variable name. When all the comments containing the string have been read and matched, the stored information is written to the HEAD.ASC file. Variable name definitions that may have previously existed in the HEAD.ASC are automatically replaced. If the variables found in the HEAD.ASC are CADAC executive variables, an appropriate definition is supplied by DFHEAD. The execution of DFHEAD and the HEAD.ASC file produced are discussed in the following sections.

3.3.2.1 DEFHEAD Execution

Program DFHEAD is a DOS application that can be added to the DIGITAL Visual FORTRAN TOOL option or can be executed in a DOS window by entering the following statement at the DOS prompt (version number must be included, ex: DFHEAD2):

```
C:>\CAD\TEST\DFHEAD
```

When program DFHEAD is executed, the user is informed of the current directory and is prompted for the name of the file containing the CADAC modules:

```
Current Directory is :
C:\CAD\TEST

Enter name of file containing variables to define (input)
Default = HEAD.ASC :
```

The user may enter a filename and press <RETURN> or select the default filename by pressing <RETURN>. If an invalid filename is entered the user receives the message:

```
* * * Invalid Filename * * *
  Do you wish to continue? (Y or N)
  Default = Y :
```

If the user selects to continue execution (by entering a "y" and pressing <RETURN> or by simply pressing <RETURN>), the user is prompted again for the file containing the variables to define, otherwise program execution is terminated.

Once a valid variable file is entered, DFHEAD prompts for the CADAC modules file:

```
Current Directory is :
C:\CAD\TEST
```

Enter name of modules file (input)
Default = MODULE.FOR :

The user may enter a filename and press <RETURN> or select the default filename by pressing <RETURN>. If an invalid filename is entered, the user receives the message:

*** * * Invalid Filename * * ***
Do you wish to continue? (Y or N)
Default = Y :

If the user selects to continue execution (by entering a "y" and pressing <RETURN> or by simply pressing <RETURN>), the user is prompted again for the file containing the CADAC modules, otherwise program execution is terminated.

Once a valid CADAC modules file is entered, the user is prompted to enter the name of the file to contain the defined variables:

Current Directory is :
C:\CAD\TEST

Enter name of file to contain defined variables (output)
Default = HEAD.ASC :

The user may enter a filename and press <RETURN> or select the default filename by pressing <RETURN>. NOTE: the output variable file can have the same name as the input variable file.

While DFHEAD is executing, the following messages are displayed on the screen, one at a time, informing the user of the status of the program. NOTE: If no variable definitions are found in the modules file the user is informed and program execution terminates after the first message without the new variable file being created:

Extracting variable definitions from module file

Reading variables from input variable file

Matching variables with definitions

Creating file with defined variables

3.3.2.2 DFHEAD Output File

The HEAD.ASC type file generated by DFHEAD will be the HEAD.ASC read by DFHEAD during execution with the addition of variable definitions appearing in columns 26-80. Figure 10 shows the HEAD.ASC of Figure 9 documented by DFHEAD.

```
SCROLL NOECHOIN NOINTMSG NOSTGMSG NORANVAR TRAJBIN NOTRAJASC NOSTATBIN NOSTATASC NOTABOUT
*
    51    REARTH     E Radius of Earth - m
```

```
  52    CRAD       E Conversion from radians to degrees
  54    AGRAV      D Gravitational constant - m/s^2
 201    TEMPK      G Atmospheric temperature - degK
 202    PRESS      O Atmospheric pressure - Pa
 203    RHO        O Atmospheric density - kg/m^3
 204    VSOUND     G Sonic speed - m/sec
 205    GRAV       O Gravity acceleration - m/s^2
 206    VMACH      O Mach number of rocket
 207    PDYNMC     O Dynamic pressure - Pa
 301    CK         D Quaternion orthonoramilty factor
 310    Q0D
 311    Q0         S quaternions, 0-th component
        •
        •
        •
 347    ETBL       G Error of direction cosine matrix nonorthogonality
 348    TLB(3,3)   G T.M. of local level wrt body axes
 451 A  EVT1EL(3)  D Target velocity error - m/s
 802    ANCOM      O Normal acceleration - g's
 803    ALCOM      O Lateral acceleration - g's
 825 A  EVT1EL(3)  D Target velocity error - m/s
 900    MAUT       D Autopilot modes =2:Prop accel; =3:Integral accel
 901    MFREEZE    D Freeze trajectory for autopilot responses
 902    ACTBW      D Actuator band width - rad/s
 903    PACL       D Location of accel close loop real pole - rad/s
 904    WACL       D Nat freq of accel close loop complex pole - rad/s
 905    ZACL       D Damping of accel close loop complex pole
 906    ANLIM      D Normal acceleration limiter - g's
 907    ALLIM      D Lateral acceleration limiter - g's
 908    DQLIMX     D Pitch flap control limiter - deg
 909    DRLIMX     D Yaw flap control limiter - deg
 910    DPLIMX     D Roll command limiter - deg
 911    PHICOMX    D Commanded roll angle - deg
 912    WRCL       D Natural freq of roll closed loop complex pole - rad/s
 913    ZRCL       D Damping of roll closed loop pole
 914    YYD
 915    YY         S Yaw feed-forward integration variable- m/s
 916    ZZD
 917    ZZ         S Pitch feed-forward integration variable- m/s
 919    DPCX       O Roll flap command deflection - deg
 920    DQCX       O Pitch flap command deflection - deg
 921    DRCX       O Yaw flap command deflection - deg
 923    DETDUM     G Determinate of intermediate matrix at inversion
 924    GAINFB(3)   G Feedback gain of rate, accel and control
        •
        •
        •
1710    PX         S Body roll angular velocity in body axes - deg/s
1711    QX         S Body pitch angular velocity in body axes - deg/s
1712    RRX        S Body yaw angular velocity in body axes - deg/s
1713    WBEB(3)    G Angular vel of veh wrt earth in body axes - rad/s
2000  * TIME       E Simulation time - s
2561    NIP        E Integration counter
2562    IPL(100)   E Start of derivative c-array locations
2866    ICOOR      E Predictor-corrector flag
2867    IPLV(100)  E Start of state c-array locations
```

Figure 10. Sample HEAD.ASC Documented by DFHEAD.

3.3.3 AHEAD

The AHEAD utility reads a HEAD.ASC type file and produces an alphabetical list of the variable names. The HEAD.ASC file may be in the form used by CADAC or the form produced by MKHEAD or of DFHEAD. The list of the variables, associated C array locations and variable definitions are written to the output file AHEAD.ASC. The execution of AHEAD and the format of the AHEAD.ASC file are discussed in the following sections.

3.3.3.1 AHEAD Execution

Program AHEAD is a DOS application that can be added to the DIGITA Visual FORTRAN TOOL option or executed in a DOS window by entering the following at the DOS prompt (version number must be included, ex: AHEAD2):

```
C:>\CAD\TEST\AHEAD
```

When program AHEAD is executed, the user is informed of the current directory and is prompted for the name of the file containing the variables to alphabetize:

```
Current Directory is :
C:\CAD\TEST

Enter name of file containing variables (input)
Default = HEAD.ASC :
```

The user may enter a filename and press <RETURN> or select the default filename by pressing <RETURN>. If an invalid filename is entered, the user receives the message:

```
* * * Invalid Filename * * *
  Do you wish to continue? (Y or N)
  Default = Y :
```

If the user selects to continue execution (by entering a "y" and pressing <RETURN> or by simply pressing <RETURN>), the user is prompted again for the file containing the variables to alphabetize, otherwise program execution is terminated.

Once a valid variable file is entered, AHEAD prompts for the name of the file to contain the alphabetized variables:

```
Current Directory is :
C:\CAD\TEST

Enter name of file containing alphabetized variables (output)
Default = AHEAD.ASC :
```

The user may enter a filename and press <RETURN> or select the default filename by pressing <RETURN>. After the filenames have been entered, the following messages are displayed on the screen, one at a time, informing the user of the status of the program:

Reading HEAD.ASC file

Alphabetizing variables

Creating AHEAD.ASC

3.3.3.2 AHEAD Output File

The output file, AHEAD.ASC, generated by the AHEAD utility contains an alphabetical list of the variables and associated information as read from a HEAD.ASC type file. The records of the AHEAD.ASC file will have the following format: variable name and dimension in columns 1-13, C array location in columns 15-18 and variable definition in columns 20-80. Figure 11 shows the HEAD.ASC of Figure 10 alphabetized by program AHEAD.

3.3.4 FRLOC

The FRLOC utility reads a HEAD.ASC type file and determines the available or unused C array locations. The HEAD.ASC file may be in the form used by CADAC or the form produced by MKHEAD or DFHEAD. FRLOC uses the C array location and the variable array dimensions in the HEAD.ASC file to determine unassigned C array locations. In determining vacant C array locations, FRLOC checks for incomplete array dimensioning and sends an error message to the output file. The execution and output file of FRLOC are discussed in the following sections.

3.3.4.1 FRLOC Execution

Program FRLOC is a DOS application that can be added to the DIGITAL Visual FORTRAN TOOL option or can be executed in a DOS window by entering the following statement at the DOS prompt (version number must be included, ex: FRLOC2):

```
C:>\CAD\TEST\FRLOC
```

When program FRLOC is executed, the user is informed of the current directory and is prompted for the name of the variable file:

```
Current Directory is :
C:\CAD\TEST

Enter name of file containing variables (input)
Default = HEAD.ASC :
```

The user may enter a filename and press <RETURN> or select the default filename by pressing <RETURN>. If an invalid filename is entered the user receives the message:

```
* * * Invalid Filename * * *
  Do you wish to continue? (Y or N)
  Default = Y :
```

If the user selects to continue execution (by entering a "y" and pressing <RETURN> or by simply pressing <RETURN>), the user is prompted again for the file containing the variables, otherwise program execution is terminated.

Once a valid variable filename is entered, the user is prompted to enter the name of the file to contain the available C locations:

```
Current Directory is :
C:\CAD\TEST

Enter name of file to contain C locations (output)
Default = FRLOC.ASC :
```

The user may enter a filename or select the default. Once the output filename is entered, FRLOC executes without any further interaction with the user.

3.3.4.2 FRLOC Output File

The FRLOC output file, FRLOC.ASC, consists of a header record followed by the available C locations for the selected HEAD.ASC type file. The locations are listed individually or as an inclusive interval. FRLOC also writes an error message to the file if an incomplete array dimension has been encountered in the HEAD.ASC. Figure 12 shows a sample FRLOC.ASC generated from the HEAD.ASC file of Figure 8.

```
ACTBW        902   D Actuator band width - rad/s
 AE         1303   D Exit area of rocket nozzle - m^2
 AGRAV        54
 AI11       1701   D Roll moment of inertia - kg*m^2
 AI33       1702   D Pitch or yaw moment of inertia - kg*m^2
 ALCOM       803
 ALLIM       907   D Lateral acceleration limiter - g's
 ALPHA0X    1637   D Initial angle-of-attack - deg
 ALPHAX      344   G Angle of attack - deg
 ALPP        342   O Total angle of attack - rad
 ALPPX       340   O Total angle of attack - deg
 ANCOM       802
 ANLIM       906   D Normal acceleration limiter - g's
 BETA0X     1638   D Initial side slip angle - deg
 BETAX       345   G Side slip angle - deg
 CA         1220   O Axial force coefficient
 CA0        1236   G Axial force coeff at zero incidence -
 CAA        1237   G Axial force coeff alpha derivative - 1/deg
 CAD        1238   G Axial force coeff of control surface drag - 1/deg**2
 CK          301   D Quaternion orthonoramilty factor
 CLL        1223   O Rolling moment coefficient
 CLLAP      1242   G Rolling moment coeff of incidence angle - 1/deg**2
 CLLDP      1243   G Rolling moment coeff of roll control deflec - 1/deg
              •
              •
              •
 THRUST     1313   O Rocket thrust parallel to vehicle centerline - N
 THTBL       335   G Pitching angle of vehicle - rad
 THTBLX      338   I/G Pitching angle of vehicle - deg
 THTVL0X    1643   D Initial vertical flight path angle - deg
 THTVLX     1641   G Vertical flight path angle - deg
 TLB(3,3)    348   G T.M. of local level wrt body axes
 VBEB(3)    1613   S Vehicle velocity in body axes - m/s
 VBEBD(3)   1610
 VBEL(3)    1633   O Vehicle velocity in local level axes - m/s
 VMACH       206   O Mach number of rocket
 VMASS      1309   O Vehicle mass - kg
 VMASS0     1307   D Vehicle mass at launch - kg
 VSOUND      204   G Sonic speed - m/sec
 WACL        904   D Nat freq of accel close loop complex pole - rad/s
 WBEB(3)    1713   G Angular vel of veh wrt earth in body axes - rad/s
 WNQ        1262   G Natural frequency of airframe pitch dynamics - rad/s
 WRCL        912   D Natural freq of roll closed loop complex pole - rad/s
 YY          915   S Yaw feed-forward integration variable- m/s
 YYD         914
 ZACL        905   D Damping of accel close loop complex pole -
 ZETQ       1263   G Damping of airframe pitch dynamics -
 ZRCL        913   D Damping of roll closed loop pole -
 ZZ          917   S Pitch feed-forward integration variable- m/s
 ZZD         916
```

Figure 11. Sample AHEAD.ASC File.

```
Unused locations in array C(3510)
   1 -   50
  53
  55 -  200
 208 -  300
 302 -  309
 318 -  319
 329 -  333
 357 -  801
 804 -  899
 918
 928 -  930
 933 - 1099
1101
1104 - 1109
1118
1122 - 1202
1204 - 1219
1226 - 1229
1252 - 1259
1265 - 1299
1301 - 1302
1311 - 1312
1314 - 1400
1402 - 1409
1416 - 1609
1622 - 1629
1644 - 1700
1703
1716 - 1999
2001 - 2560
2662 - 2865
2967 - 3510
```

Figure 12. Sample FRLOC.ASC File.

3.3.5 DFMOD

The DFMOD utility documents the modules by taking the variable definitions from a HEAD.ASC and inserting them into the CADAC module code. The CADAC module code has a specific ordering for the placement of EQUIVALENCE statements involving the C array. The EQUIVALENCE statements are separated into categories with the category name in a comment statement above each block of EQUIVALENCE statements. The EQUIVALENCE statements are then listed consecutively after the comment statement. DFMOD only transfers definitions associated with the "INPUT FROM OTHER MODULES" category.

The program performs the transfer in the following manner. The program reads the module and searches for the comment statement containing the text "INPUT FROM". When the match is found, the EQUIVALENCE statements following the comment until the next occurring comment statement are searched. The variable names are extracted from the EQUIVALENCE statements and matched with variables in the HEAD.ASC file. When the end of the equivalence block is reached all the variables and definitions are written as a block of comments to the CADAC module. If comments already exist for the equivalence block, they will be replaced. The HEAD.ASC file used as input may be in the form used by CADAC or in the form produced by MKHEAD or DFHEAD. The execution of DFMOD and its output file are discussed in the following sections.

3.3.5.1 DFMOD Execution

Program DFMOD is a DOS application that can be added to the DIGITA Visual FORTRAN TOOL option or can be executed in a DOS window by entering the following statement at the DOS prompt (version number must be included, ex: DFMOD2):

```
C:>\CAD\TEST\DFMOD
```

When program DFMOD is executed, the user is informed of the current directory and is prompted for the name of the variable file:

```
Current Directory is :
C:\CAD\TEST

Enter name of file containing variables (input)
Default = HEAD.ASC :
```

The user may enter a filename and press <RETURN> or select the default filename, by pressing <RETURN>. If an invalid filename is entered the user receives the message:

```
* * * Invalid Filename * * *
  Do you wish to continue? (Y or N)
  Default = Y :
```

If the user selects to continue execution (by entering a "y" and pressing <RETURN> or by simply pressing <RETURN>), the user is prompted again for the file containing the variables, otherwise program execution is terminated.

Once a valid variable file is entered, DFMOD prompts for the CADAC modules file:

```
Current Directory is :
C:\CAD\TEST

Enter name of  modules file (input)
Default = MODULE.FOR  :
```

The user may enter a filename and press <RETURN> or select the default filename, by pressing <RETURN>. If an invalid filename is entered, the user receives the message:

```
* * * Invalid Filename * * *
  Do you wish to continue? (Y or N)
  Default = Y :
```

If the user selects to continue execution (by entering a "y" and pressing <RETURN> or by simply pressing <RETURN>), the user is prompted again for the file containing the CADAC modules, otherwise program execution is terminated.

The user is then prompted to enter the name of the modules file to contain the definitions from the variable file:

```
Current Directory is :
C:\CAD\TEST

Enter name of the output modules file (output)
Default = MODULE.FOR  :
```

The user may enter a filename and press <RETURN> or select the default filename by pressing <RETURN>. Once the output modules file is entered, the program executes to completion.

3.3.5.2 DFMOD Output File

The output file produced by DFMOD is a copy of the MODULE.FOR file read by the program with inserted comment blocks. The comment blocks are the variable definitions pulled from the HEAD.ASC file. The first line of the inserted comment block is a blank line with a "C" in the first column. The remaining lines in the block will have a "C" in column 1, the variable starting in column 7, followed by "= " and then the variable definition. The use of "= " as a separator prevents the DFHEAD utility, (Section 3.3.2), which searches for " = ", from picking up the "input from other modules" comments from the CADAC modules. Figure 13 contains an excerpt of a CADAC module showing equivalence and comment blocks.

```
      •
      •
      •
C*** INPUT FROM OTHER MODULES
C
      EQUIVALENCE (C(1613),VBEB(1))
      EQUIVALENCE (C(1705),PP)
      EQUIVALENCE (C(1707),QQ)
      EQUIVALENCE (C(1709),RR)
      EQUIVALENCE (C(2000),T)
C
C VBEB= S Vehicle velocity in body axes - m/s
C PP= S Body roll angular velocity - rad/s
C QQ= S Body pitch angular velocity - rad/s
C RR= S Body yaw angular velocity - rad/s
C
C*** STATE VARIABLES
C
      EQUIVALENCE (C(0310),Q0D)
      EQUIVALENCE (C(0311),Q0)
      EQUIVALENCE (C(0312),Q1D)
      EQUIVALENCE (C(0313),Q1)
      EQUIVALENCE (C(0314),Q2D)
      EQUIVALENCE (C(0315),Q2)
      EQUIVALENCE (C(0316),Q3D)
      EQUIVALENCE (C(0317),Q3)
C
C Q0 = S Quaternion, 0-th component
C Q1 = S Quaternion, 1-th component
C Q2 = S Quaternion, 2-th component
C Q3 = S Quaternion, 3-th component
C
C***  OUTPUT TO OTHER MODULES
C
      EQUIVALENCE (C(0320),TBL(1,1))
      EQUIVALENCE (C(0340),ALPPX)
      EQUIVALENCE (C(0341),PHIPX)
      EQUIVALENCE (C(0342),ALPP)
      EQUIVALENCE (C(0343),PHIP)
C
C TBL(3,3) = O Transf matrix of body wrt local level axes
C ALPPX = O Total angle of attack - deg
C PHIPX = O Aerodynamic roll angle - deg
C ALPP = O Total angle of attack - rad
C PHIP = O Aerodynamic roll angle - rad
      •
      •
      •
```

Figure 13. Excerpt from CADAC Module Documented by DFMOD.

3.3.6 CHKINT

The CHKINT utility documents the uses of the C-locations with respect to state variable integration. This utility is to help the user ensure no variables that are to be integrated are being used else where for other purposes. CHKINT does this by first making a record of the C-locations used by varaibles from the HEAD.ASC file then searches the modules file(s) for C-locations used for state variable integration which are located in the initialization modules. As the record of used C-locations is being prepared from the HEAD.ASC file and the modules file(s), CHKINT searches the record to compare the present C-location to the C-locations already in the record for a possible match. If a match is found an error is given.

3.3.6.1 CHKINT Execution

CHKINT is a Windows application that can be added to the DIGITAL Visual FORTRAN TOOL option. In addition, CHKINT execution can be initiated by using the RUN option in Windows or by double-clicking on the CHKINT executable file in Windows EXPLORER. The user should follow a few guidelines in order for CHKINT to execute properly: (1) the user must follow the <<CODE CONVENTIONS>> in the modules, (2) the user must execute the program MKHEAD.EXE creating a new header file (this header file DOES NOT have to be named HEAD.ASC) and (3) the user then executes the program CHKINT.EXE. For more information on CHKINT execution refer to the on-line help in the CHKINT menu.

3.3.6.2 CHKINT Input and Output Files

The input files for the CHKINT program are the HEAD.ASC file and the modules file(s) containing the code for the simulation. The user should be aware that certain restrictions apply to the coding structure used in reference to the state variables in the intialization modules. These coding conventions are illustrated in Figure 14.

The output file of CHKINT is CHKINT.ASC. This file contains all the occurences of state variable C-location use in the modules as well as errors resulting from multiple use of state variable C-locations in the modules. An example of the contents of CHKINT.ASC is:

```
------------    C-Location: 915 Variable: YY found in Head.asc is used in
                E:\XR98\ChkintTest\Module.for at Line#: 2303 in SUBROUTINE C2I
```

If C-location "915" was also used elsewhere the contents would indicate an error:

```
--- ERROR ---   C-Location: 915 was used in E:\XR98\ChkintTest\Module.for at Line#: 2303
and is used in E:\XR98\ChkintTest\Module.for at Line#: 2308   in SUBROUTINE C2I
```

```
Example 1, Single State Integration:

       IPL(NIP) = 1610  <= C-location
       IPLV(NIP) = 1611 <= C-location

Example 2, Single State Integration:

       ILOC = 1610  <= C-location
       DO J = 1, 6
       IPL(NIP) = ILOC
       IPLV(NIP) = ILOC +1
       ILOC=ILOC+2
       NIP = NIP + 1
       ENDDO

       NOTE:  The "1" and "6" of the "DO" line can be any integer
                     The "1" in the IPLV line can be any integer

Example 3, Three State Vector Integration:

       ILOC = 1610  <= C-location
       DO J = 0, 2
       IPL(NIP) = ILOC + J
       IPLV(NIP) = ILOC + J + 3
       NIP = NIP + 1
       ENDDO

       NOTE:  The "0" and "2" of the "DO" line can be any integer
                     The IPLV line could be omitted
                     The "3" in the IPLV line can be any integer

Example 4, Three State Vector Integration:

       DO J = 1610, 1612  <= C-locations
                        IPL(NIP) = J
                        IPLV(NIP) = J + 3
                        NIP = NIP + 1
       ENDDO

       NOTE:  The IPLV line could be omitted
                     The "3" in the IPLV line can be any integer

Example 5, Legacy Single State Integration:

       J = 1610  <= C-location
       IPL(NIP) = J
       IPLV(NIP) = J + 3

       NOTE:  The IPLV line could be omitted
                     The "3" in the IPLV line can be any integer

Example 6, Legacy Three State Vector Integration:

       DO J = 1610, 1618, 4  <= C-locations
              IPL(NIP) = J
              IPLV(NIP) = J + 3
              NIP = NIP + 1
       ENDDO
       NOTE:  The IPLV line could be omitted
                     The "3" in the IPLV line can be any integer
```

Figure 14. Coding Structure.

4. CADAC EXECUTIVE

The CADAC executive, CADX.OBJ, is a binary program file that contains the object code (binary instructions) that controls the simulation execution. This file is created by the DIGITAL Visual FORTRAN compiler. The user should link the file to their airframe/sensor modules prior to execution. This module is required for execution of the CADAC program.

4.1 CADAC Executive Information

This section contains information for the CADAC executive. Topics discussed in this section include the integration method used in CADAC, a discussion on C variables utilized by the executive, suggested output variables, vector definitions and integration error messages.

4.1.1 CADAC Integration

CADAC uses the modified Euler numerical integration method with a predictor corrector. The modified Euler method is classified as a Runge-Kutta method of order two, the order of its local truncation error. The modified Euler method is as follows:

Given $Y' = f(t,y)$, $Y = f(t,y)$ is estimated by:

$$w_{i+1} = w_i + \frac{h}{2}\left[f(t_i, w_i) + f\left(t_{i+1}, w_i + h\, f(t_i, w_i)\right)\right]$$

given w_0 is the initial value of y (*Numerical Analysis*, Burden, Faires & Reynolds, 1981).

An error notification methodology is included in the integration module. This methodology is controlled by the flag INTMSG/ NOINTMSG, entered in the HEAD.ASC file. If NOINTMSG is entered in the HEAD.ASC option list, no integration messages are displayed to TABOUT. If INTMSG is entered in the HEAD.ASC, a message is displayed in TABOUT when a single state variable has produced the largest error for 100 consecutive integration intervals. Once the message is displayed for a specific variable, consecutive messages for that variable are suppressed until a different variable assumes the largest error for 100 integration steps. A sample of the integration message is:

```
*** INTEGRATION WARNING ***
C LOCATION:  1680  generated an error of 0.005
```

4.1.2 ICOOR - Predictor/Corrector Flag

The ICOOR flag is an executive routine variable used to ensure that any user written code is executed when desired by the user. The code may be executed during the initialization subroutines, immediately following predictor loops, immediately following corrector loops or immediately following non-predictor-corrector loops. ICOOR indicates the phase of execution. ICOOR, located in common location 2866, assumes the following values:

ICOOR	PROGRAM FLOW
-2	At the beginning of program execution prior to the user written initialization subroutines. Flow has not passed through the integration subroutine, AMRK.
-1	After going through all user written initialization subroutines. This occurs at the beginning of program execution only. Flow has not passed through the integration subroutines, AMRK. or After going through all user written subroutines immediately after staging with new staging values inserted in common. Flow does not pass through the integration subroutine, AMRK.
0	After going through all user written subroutines following a predictor loop in AMRK.
1	After going through all user written subroutines following a corrector loop in AMRK.

4.1.3 CADAC Executive Output Flags

Several variables are used by the CADAC executive to control the CADAC output. These variables, which should be initialized in each INPUT.ASC file by the user, are elements of the C array but are not used by the user supplied modules. Instead, the executive routines depend on them. The variables are PPP (C(2005)), CPP (C(0205)), DER (C(2664)) and OPTMET (C(0053)).

The PPP variable controls the frequency in which data is written to the TRAJ plot file. The variable is initialized to a time interval. When data is written to TRAJ, the time of the written data is stored. At the next integration step, the interval between the simulation time and the previous written time is computed and compared to the interval in PPP. When the interval is larger than the value in PPP, then new data is written to TRAJ and the written time updated. Otherwise, writing to TRAJ is omitted until the time interval equals or exceeds PPP. Initializing PPP to a very large value will prohibit data from being written to the TRAJ plot file.

The CPP variable controls the frequency in which data is written to TABOUT as well as the target track data file, TRACK.ASC. The CPP contains the print interval for the output data. From this print interval, the next time that a print should occur is computed. The simulation time is then compared to this print time. When the simulation time is equal or exceeds the print time, then data is scrolled to the screen or written to the TABOUT file and a new print time is computed. If the CPP variable is to be used to indicate the print time for the target track data file, the value should be set tot the approximate MIL-SIM integration time and unchanged throughtout the CADAC run.

The DER variable is the integration time step size. This variable indicates the time of the next integration step. Computational stability is improved if the numerical value carries several digits, i.e. DER = 0.0123.

The OPTMET variable is a flag that indicates the units for all CADAC variables. The user may select from the English system (OPTMET = 0) or metric system (OPTMET = 1).

4.1.4 Defining Vectors in CADAC

State variables can be declared using two methods. The first consists of a very strict format. If the variable resided in location C(LOC), then the derivative of the variable must reside in location C(LOC - 3). An array, IPL(1:100), (which corresponds to C(2562:2661)), is utilized to indicate which C locations contained state variables by loading the array with the C location of the state variable derivative.

An additional method for declaring state variables supports this previous definition of the state variables as well as provides flexibility to support the definitions of vectors. A second array IPLV(1:100), (which corresponds to C(2867:2966)), points to the state variable, much like the array IPL points to the derivative of the state variable. If the IPLV array contains a 0 value, then the location of the state variable is defined as (IPL+3), as per the original methodology. Otherwise this array contains the state variable location.

If the user wishes to define a state vector of consecutive C locations, the state vector definition is made in the desired module. A vector containing the derivative of the state vector is also defined in the module. Then, in the initialization subroutine for that module, the user loads the IPL array with the C locations of the vector containing the derivatives and the IPLV array with the C locations of the state vector. Figure 15 shows an example of defining a vector in module D1. The state variable vector, VECT, is defined in D1 along with the vector for the derivative, DVECT. In the initialization module, the C locations of the state vector are loaded into the IPLV array and the C locations of the derivative vector are loaded into the IPL array. The variable VAR1 is an example of the first methodology.

```
Sample code for module D1I:

       DIMENSION  IPL(100), IPLV(100)
       COMMON     C(3510)

       EQUIVALENCE ( C(2562), IPL(1) )
       EQUIVALENCE ( C(2967), IPLV(1) )
       EQUIVALENCE ( C(2561), N       )

       LOCATION = 1680
       DO  I = 0, 2
          IPL(N) = LOCATION + I
          IPLV(N) = LOCATION + I + 3
          N = N + 1
       ENDDO

       IPL(N) = 1690
       N = N + 1

Sample code for module D1:

       DIMENSION  VECT(3), DVECT(3)
       COMMON     C(3510)

       EQUIVALENCE ( C(1680), DVECT )
       EQUIVALENCE ( C(1683), VECT  )
       EQUIVALENCE ( C(1690), DVAR1 )
       EQUIVALENCE ( C(1693), VAR1  )
```

Figure 15. Vector Definition Example.

5. CADAC EXECUTION

5.1 Sample CADAC Executions

This section discusses two sample INPUT.ASC files in detail. These samples, along with the examples discussed in Section 3.1.1, demonstrate the flexibility of the CADAC program in modeling different scenarios. The samples covered in this section are a submunition dispensing simulation and a multiple run simulation.

5.1.1 Submunition Dispersion

CADAC is capable of simulating submunition dispersion. Missile trajectories, which have a common path up to a certain stage then diverge, can also be simulated using the same technique as the submunition dispersion methodology.

To simulate the submunition dispersion, the user includes a SAVE statement in the INPUT.ASC file, simulating the dispersion point. There may be only one SAVE statement per INPUT.ASC file and the SAVE statement should occur directly after a stage. The carrier missile trajectory is simulated up to the dispersion point, then the SAVE statement is placed in the INPUT.ASC file. Any stages following the SAVE statement should simulate a stage within the submunition trajectory.

When the input file is executed, the trajectory is simulated up to the SAVE statement. When the SAVE statement is processed, the values in all of the common locations (C(1:3510)) are written to a scratch file, CSAVE.ASC. The submunition trajectory is then processed as directed by the input until the RUN statement is reached. After the RUN statement is reached, the C array is initialized from the data saved to the CSAVE.ASC file. The first group card set following the RUN statement are then processed. These group cards should contain modifications for the submunition trajectory simulation cards following the dispersion point (SAVE statement). Any statements occurring prior to the dispersion point should not be repeated. Once all of the group statements are processed, the trajectory simulation is performed STARTING AT THE DISPERSION POINT instead of the initial stage. By including the MONTE CARLO statement, the user can simulate multiple submunition trajectories for each group statement set.

Figure 16 shows an example of a submunition dispersion INPUT.ASC file. The file is an actual input file with some assignment statements omitted. The first trajectory starts at time zero and has three stages plus impact. All other trajectories start at a time greater than 25 seconds. The SAVE statement is placed at the beginning of the second stage (indicated in Figure 16 by "!**Stage 1"). This causes all global variables to be written to CSAVE.ASC at the start of the second stage and, in the later trajectories, read at the beginning of the second stage. The ASSIGNMENT statements placed between the SAVE statement and RUN statements should always contain at least one distributive variable to insure proper Monte Carlo functioning.

```
L3SC6, SCALP, TARG 2, NFT, POP-UP 5000 M, MONTE CARLO
MONTE CARLO 30
!
MODULES
   G1 TARGET
   G2 AIR DATA
   S1 SENSOR
   S4 INS
   C1 GUIDANCE
   C2 CONTROL
   A1 AERO COEF
   A2 PROPULSION
   A3 FORCES
   D1 DYNAMICS
   D2 ROTATION
END MODULES
!
OPTMET = 1.0                ! *** Stage 0
MAIR = INT ( 0.0 )
PSIWLX = 90.0
DVAEL = 10.0
       •
       •      {insert ASSIGNMENT statements as needed}
       •
 MSEEK = INT( 1.0 )
 MTURN = INT(1.0)
 PPP = 1.0
 CPP = 1.0
 DER = 0.05
!
IF TIME > 25.0                     ! *** Stage 1
!
SAVE
!
ST1EL3  = -10
MSEEK = INT( 1.0 )
NMAP = INT( 2.0 )
NFIXM = INT( 10.0 )
RACQ = GAUSS( 1200., 300.0 )
SWEL1 = 0.
SWEL2 = 0.
SWEL3 = -10.
SWRWL1 =  GAUSS( 0., 5.0 )
SWRWL2 =  GAUSS( 0., 5.0 )
SWRWL3 =  GAUSS( 0., 5.0 )
RANDPB =  GAUSS( 0., 0.001 )
RANDTB =  GAUSS( 0., 0.001 )
RANDPC = GAUSS( 0.,  0.0025 )
RANDTC = GAUSS( 0.,  0.0025 )
RANDDC = GAUSS( 0.,  1.75 )
```

Figure 16. Sample INPUT.ASC for Submunition Dispersion.

```
       DTIMMP = 0.5
       DTIMCR = 0.5
       FOVYAW = 0.07
       FOVPIT = 0.05
       RMIN = 250.0
       MGUID= INT( 30.0 )
       PSIFLC = 0.0
       THTFLC  = -0.1745
       MAUT= INT( 13.0 )
       PPP = 1.0
       !
       IF  DBE   <   5000.0                  ! *** Stage 2
       !
       MGUID = INT( 33.0  )
       MAUT = INT( 14.0 )
       PPP = 1.0
       !
       RUN
       !
       ST1EL3 =-50.  STAGE 1
       LOAD
       !
       ST1EL3 = -100.        STAGE 1
       LOAD
       !
       STOP
```

Figure 16. Sample INPUT.ASC for Submunition Dispersion. (Concluded)

The sample INPUT.ASC file generates 30 trajectories in the first group having a target altitude of -10. Following the RUN statement, the target altitude is changed to -50 and 30 more trajectories are run as group 2. Following the first LOAD statement, target altitude is changed to -100 and again, 30 more trajectories are run as group 3. Following the LOAD statement, the STOP statement indicates termination of CADAC execution.

The STAT file generated for a submunition dispersion simulation contains selected data for the trajectories. The data is written to the file at each stage and at the end of each trajectory. Therefore, the first trajectory will have more records than the following trajectories since all trajectory stages are computed for the first trajectory and remaining trajectories start at the dispersion point. Typically, the user is interested in only the impact (or end) points. Hence, when the bivariate option of KPLOT is executed, the option for selected stages must not be chosen.

Graphics can be obtained for the multiple submunition trajectories by performing the following steps:

1) Edit the INPUT.ASC file and omit the Monte Carlo statement
2) Using the TRAJ file as input, run program KPLOT

One trajectory of the carrier missile will be drawn, terminating at the point at which the SAVE statement was placed. The trajectory of the first submunition continues from there. Next, the trajectories of each of the other submunitions are drawn, starting at the dispersion point and terminating at impact.

Graphics generated from the STAT data by the sample INPUT.ASC file, show three statistical ellipses, one for each group of impacts. Variables included on the STAT data are the time and missile coordinates (XYZ) with respect to the target. Statistics and graphics are then performed based on these values. The position, size and shape of each ellipse show the effect of increasing the altitude of the target.

5.1.2 Sample INPUT.ASC with the Multiple Run Option

Figure 17 is a sample INPUT.ASC file with some ASSIGNMENT statements omitted. Each trajectory generated by this file starts at time zero and has three stages prior to impact (stages 0,1,2). The SAVE statement was not used for this case because the distributive variables on ASSIGNMENT statements in stage 0 generate different conditions for each trajectory. The SAVE statement would bypass this stage for trajectories after the first trajectory. The sample INPUT.ASC file generates 30 trajectories per group, for three groups. The first group is executed with a target altitude of -10. Following the RUN statement, the target altitude is modified by the first set of group statements to -50.0. The second set of group statements modify the target altitude to -100.0. Following the second LOAD statement, the STOP statement terminates program execution.

The graphics generated by the STAT file from this sample INPUT.ASC, displays three statistical ellipses, one for each group of impacts. Variables recorded on the STAT file included time and missile coordinates (XYZ) at impact with respect to the target. Statistics and graphics are performed based on these values. The position, size and shape of each ellipse show the effect of increasing the altitude of the target.

```
L3SC6, SCALP, TARG 2, NFT, POP-UP 5000 M, MONTE CARLO
MONTE CARLO 30
!
MODULES
   G1 TARGET
   G2 AIR DATA
   S1 SENSOR
   S4 INS
   C1 GUIDANCE
   C2 CONTROL
   A1 AERO COEF
   A2 PROPULSION
   A3 FORCES
   D1 DYNAMICS
   D2 ROTATION
END MODULES
!
! *** Assignment Statements
OPTMET = 1.
MAIR = INT( 0.0 )
PSIWLX  = 90.
DVAEL = 0.
MSEEK=INT( 1.0        )
NMAP = INT( 1.0       )
NFIXM = INT( 1.0 )
RACQ = GAUSS( 1200., 300. )
SWEL1 = -7000.0
SWEL2 = 12000.0
SWEL3 = -100.0
SWRWL1 = GAUSS( 0.0, 30.
SWRWL2 = GAUSS( 0.0, 30. )
MTURN = INT( 1.0 )
       •
       •      {insert type ASSIGNMENT statements as needed}
       •
PPP = 1.0
CPP = 1.0
DER = 0.05
!
IF WAY > 0.1
!
STEL3 = -10
MSEEK  = INT( 1.0 )  NMAP = 1.2
NFIXM = INT( 10. )
RACQ = GAUSS( 1200., 300. )
SWEL1 = 0.
SWEL2 = 0.
SWEL3 = -10.
SWRWL1 = GAUSSS( 0., 5.0 )
SWRWL2 = GAUSSS( 0., 5.0 )
```

Figure 17. Sample INPUT.ASC With Multiple Run Option.

```
       SWRWL3 = GAUSS( 0., 5.0 )
       RANDPB = GAUSSS( 0., 0.001 )
       RANDTB = GAUSSS( 0., 0.001 )
       RANDPC = GAUSSS( 0., 0.0025 )
       RANDTC = GAUSSS( 0., 0.0025 )
       RANDDC = GAUSSS( 0., 1.75 )
       DTIMMP = 0.5
       DTIMCR = 0.5
       FOVYAW = 0.07
       FOVPIT  = 0.05
       RMIN = 250.0
       MGUID = INT( 30.0 )
       PSIFLC = 0.0
       THTFLC = -0.1745
       MAUT =INT( 13.0 )
       PPP = 1.0
       !
       IF DBE < 5000.0
       MGUID = INT( 33.0 )
       MAUT = INT( 14.0 )
       PPP = 1.0
       RUN
       !
       ST1EL3 = -50. STAGE  1
       LOAD
       !
       ST1EL3 = -100.       STAGE 1
       LOAD
       !
       STOP
```

Figure 17. Sample INPUT.ASC With Multiple Run Option. (Concluded)

5.2 CADAC Sweep Methodology

The sweep methodology executes multiple CADAC trajectories by varying the launch point or aimpoint for successive trajectories within one input file. The sweep methodology executes the CADAC trajectories within two nested loops. The sweep methodology controls the initialization and incrementing of the loop variables, detects when all trajectory simulations have been generated and stops the looping process. Six options provide the user with a selection of incrementing algorithms for the range variable. (These options are discussed in Section 3.1.2.4.3.1). Although any two CADAC variables in Cartesian or polar coordinates can be the control variables for the loops, the outer and inner variables are usually expressed in polar coordinates with range being the inner variable and an azimuth or elevation angle as the outer variable.

The looping variables and incrementing algorithm inputs are initialized by the SWEEP block in the INPUT.ASC file. After executing the CADIN program, the resulting Type 19, 20 and 21 cards in the CADIN.ASC file are used by the CADAC Executive code to call the appropriate Sweep Logic Module. Execution is controlled by the desired Sweep Logic Module which is included in the CADAC Executive code and are not intended to be modified by the user. These logic modules perform the sweep incrementing calculations and provide logic control for both the CADAC executive code and the user supplied modules.

5.2.1 Sweep Input Modules

The CADAC executive code develops the input required for the sweep methodology. These subroutines read the data input from the SWEEP block of the INPUT.ASC file and load the appropriate C locations with the data. Table 7 shows the C locations and definitions utilized by the sweep methodology. Section 3.1.2.4.3 contains information on the SWEEP input block and format.

5.2.2 Sweep Logic Modules

The sweep methodology utilizes two of the trajectory simulation variables as the loop controllers. Typical looping variables are the range to the target (the inner variable) and angle off the target (the outer variable). The trajectory simulation output resulting from the variation in these variables is grouped into rays emanating outward from the sweep center point. Output data occurs at the range values along each ray.

Table 7. Sweep Variable C Definitions.

C Location	C Name	Definition	Input.ASC Name†
1800	ISWEEP	Sweep option flag	Mode Type
1801	CRITNO	Critical variable number	From HEAD.ASC
1802	CRITVAL	Critical test value	CritVal
1803	SEARNO	Number of binary searches*	NumBins
1804	NUMR	Number of aimpoints per ray**	NumTraj
1805	CRITMAX	Value to be placed in CRITVAR when termination reaches incorrect end (LCONV > 2)	MaxVal
1811	ANGLNO	Outer variable number	From HEAD.ASC
1812	ANGMIN	Outer variable minimum	MinAngl
1813	ANGMAX	Outer variable maximum	MaxAngl
1814	ANGDEL	Outer variable delta	AngDelt
1815	ANGUNT	Outer variable units flag	Program Determined
1821	RANGNO	Inner variable number	From HEAD.ASC
1822	RANMIN	Inner variable minimum	MinRang
1823	RANMAX	Inner variable maximum	MaxRang
1824	RANDEL	Inner variable delta	DeltVal
1831-1833	SCEL	Location of the center of the sweep	

* Used in sweep modes OUT and ALL only

** Used in all sweep modes except OUT and ALL

† See Section 3.1.2.4.3 SWEEP Block

The user selects the simulation variables to be used as the looping variables by assigning the appropriate C locations; (C(ANGLNO)) for the outer loop and (C(RANGNO)) for the inner loop. The maximum and minimum values for the variables and the increment are also entered by the user through the RANGE and ANGLE statements. The sweep option is selected by setting the MODE statement to the desired option. The remaining statements in the SWEEP block specify the data needed for the option selected.

For each option, the inner and outer loop variables are initialized to the respective initial values and the described trajectory simulated. The inner loop is then incremented by the delta determined by the selected option and the new trajectory simulated. This continues until the stopping criteria for the inner loop is satisfied. Once the inner loop has been completed, the outer loop variable is incremented accordingly by the user-input delta and the inner loop methodology is restarted. When the current outer loop stopping criteria is satisfied, the sweep methodology is terminated and the program stopped.

5.2.3 SWEEP Files

There are three files that are created when CADAC is executed using the SWEEP option. The impact output file, IMPACT.ASC, contains the CADAC variables (noted by an asterisk in the HEAD.ASC file) for each individual trajectory at the trajectory terminatino point only. Termination point will result from: 1) a Point of Closest Approach (PCA) to the target or 2) a large miss of the target. The G4 module is called by the CADAC executive every integration interval to determine if the trajectory has entered the PCA sphere around the target. When it does, the G4 module interpolates the time of closest approach and the associated parameters. Large misses of the target are usually the result of a stopping criteria being met, impact with the ground (also checked for in module G4) or termination on staging. In these cases, the IMPACT.ASC file contains the values of the CADAC variables noted in the HEAD.ASC file at the time the simulation was terminated with the critical variable, CritValVarName, containing the CRITMAX value. This substitution allows the SWEEP program to contour around "holes" in footprints and launch envelopes. In addition, the LCONV (C(2020)) variable contains an integer flag indicating the reason each trajectory was terminated. LCONV values are displayed in Table 8. For accurate SWEEP results (as generated by the SWEEP application), LCONV must be present in the IMPACT.ASC file; outputting LCONV to the IMPACT.ASC file is accomplished by placing an '*' in column 2 and an 'I' in column 3 of the HEAD.ASC file. The CRITMAX variable must also be present in the IMPACT.ASC file for an accurate SWEEP analysis. The CRITMAX variable is entered by the user using a text editor. The CRITMAX variable location is 1805 and column 2 and 3 must be marked accordingly. If the user does not enter the CRITMAX variable to the HEAD.ASC list, the user is then required to enter the CRITMAX value during the execution of the SWEEP application.

Table 8. LCONV Values.

LCONV Value	LCONV Set In	Meaning
0	Executive Code	Set at start of trajectory
1		Undefined
2	G4 Module	Trajectory was terminated within the PCA sphere
3	G4 Module	Trajectory was terminated by impact with the ground
4	G4 Module	Trajectory was terminated by a Termination Code
5	Executive Code	Trajectory was terminated by staging

The sweep methodology relies on two scratch files generated by the executive modules: IMPACT10.ASC contains the criteria value, inner loop value and outer loop values for one or more trajectories; and IMPACT7.ASC contains the plot variables generated at the end of a trajectory, for one or more trajectories. For more information on these files, refer to Sections 6.1.6, 6.1.7 and 6.1.8, respectively.

5.2.4 SWEEP Implementation Methodology(Single Trajectory per Aimpoint)

In order to implement the sweep methodology 1) the variable MINIT (C(1600)) must be assigned an integer value greater than 99 and 2) variables must be set in the ASSIGNMENTS block of INPUT.ASC file. When running the sweep options, single trajectory runs are also possible, although the initialization is different from the single trajectory mode. Helpful notes for the implementation and usage of the SWEEP mode are also included in this section.

5.2.4.1 Sweep Implementation Notes

Some helpful hints when running the sweep option with CADAC:

1) Before running numerous trajectories with the OUT or ALL mode, run the option with the number of binary searches set to 0. This allows the user to observe trajectories for selected rays, insuring that the sweep center point, maximum and minimum values for the inner variable are properly chosen.
2) Designating a target directly under the launch point produces uncertain results with the sweep methodology.
3) The maximum value of the angle should be greater than the minimum value. For example, to execute a sweep from 270 degrees to 90 degrees, the user should enter a minimum angle of 270 degrees and a maximum angle of 450 degrees.
4) Once the subroutines are set up for sweep runs, single runs can be easily made by setting the minimum values of the range and angle equal to the desired trajectory conditions. The maximum values of the range and angle should be less than the minimum values plus the increment. In the cases of the OUT and ALL modes, the number of searches should be set to 0 to obtain single runs of a trajectory.
5) The files produced by the SWEEP option (TRAJ.ASC, TRAJ.BIN, STAT.ASC. STAT.BIN and IMPACT.ASC) can be input to the QPRINT, KPLOT or SWEEP programs. The TRAJ.ASC and TRAJ.BIN files are usually NOT selected in the HEAD.ASC file when using the SWEEP option since the resulting files can be extremely large. A sweep run of various sectors, i.e. a 0 to 90 degree sector and a 90 to 270 degree sector, can be merged in a text editor (*.ASC files only) to form a combined data file from 0 to 270 degrees. However, the header before the second sector must be deleted in the combined data file.
6) If the binary search of the OUT or ALL mode uses distance as the critical value, then the critical variable name and common locations number should reflect the correct variable. The appropriate interpolated miss distance is printed to the output file immediately after the line containing the message, "PARAMETERS AT IMPACT TIME" for air-to-ground simulations and “INTERCEPT TIME” for air-to-air simulations. The time printed on this line is the interpolated time of closest approach

to the target. The time of the last integration step and the value of the critical variable as defined in the LIMIT statement is printed after the message "FLT TIME =".

5.2.5 SWEEP Implementation Methodology (Multiple Trajectories per Aimpoint)

If it is desired to run Monte Carlo runs at each aimpoint when implementing the sweep methodology, simply enter the number of runs in the MONTE statement of the INPUT.ASC file for a single trajectory per aimpoint sweep run. At least one VAR statement must be assigned from a distribution to make the run meaningful.

5.3 Real-Time CADAC Methodlogy

The real-time CADAC methodology converts a batch CADAC model into a single missile subroutine that can be used in a man-in-the-loop simulator (MIL frame) or as a check-out Test Bed. The expression ‘real-time’ in this document is to be understood as just an indicator that the missile simulation is suitable for real-time execution. Whether it will actually run real-time in the MIL frame depends on the processing power of the facility. However, the coded file has stripped down the CADAC executive routine to the bare minimum. The efficiency and the speed of execution of the modules that define the missile are the responsibility of the user/developer.

The real-time coded file is based upon a user-generated set of MODULES with an associated CADIN.ASC and HEAD.ASC file. The real-time CADAC file will have the target information removed although this information is included in the MODULES, CADIN.ASC and HEAD.ASC files. The MODULES should be executed in a batch format with the associated CADIN.ASC and HEAD.ASC file, which should have the appropriate option list keywords to generated real-time CADAC target track data files. The CONVRT program is then executed with input provided by 4 files and CADRTxx.FOR being the output file. The following sections provide an overview on creating a real-time CADAC application; further details can be found in the ***Real-time CADAC Documentation***.

5.3.1 Real-time CADAC Files

The real-time CADAC converter program, CONVRT, requires 4 input data files associated with a given missile simulation to create an accurate FORTRAN coded real-time file. These files are MODULES.FOR, CADIN.ASC, HEAD.ASC, and INIT.ASC.

5.3.1.1 MODULES.FOR File

The MODULES.FOR file is the user-generated modules that simulate the given missile. This file is appended to the resulting real-time CADAC coded file, CADRTxx.FOR. Subroutines that model the target (subroutine G1) and the radar track (subroutine S2) and their associated initialization routines are removed. The data output from these subroutines are provided in the INIT and TRACK data files. Since CADAC may be used to create 5 DOF or 6

DOF simulations, this difference extends to the variable names and locations for target and radar/data link information. The user must ensure that the appropriate variables for 5 or 6 DOF models are indicated in the HEAD.ASC file associated with the module file. For more information on the difference in variables refer to the ***Real-time CADAC Documentation.***

5.3.1.2 CADIN.ASC File

The CADIN.ASC file is the trajectory-controlling file. This file directs the flight of the missile as well as control output print intervals. This file is incorporated into the CADRTxx.FOR file with all target data initializations removed. The user should initialize the CPP variable (C(2015)) set to the approximate MIL frame integration step size and have it remain unchanged throughout the trajectory. The CPP value will be the time interval for data written to the TRACK data link update file. The user should not include save state cards (type 90) or groups cards (type 12) within the CADIN.ASC file.

5.3.1.3 HEAD.ASC File

The HEAD.ASC file serves two purposes in the real-time CADAC methodology: (1) indicates the target track data files, INIT and TRACK, need to be output and their format and (2) indicates the variable data to be output to the INIT and TRACK target data files.

Including the appropriate keywords in the HEAD.ASC option list creates the output track data files. ASCII INIT and TRACK files are created by including the keywords INITASC and TRACKASC, respectively. If the user wishes to create a binary form of these files, the keywords INITBIN and TRACKBIN should be included.

Placing the appropriate letter in column 1 of the HEAD.ASC plot variable list indicates the variables output to each file. There are two types of variables used in the real-time CADAC application, Initialization and Tracked. Placing an ‘I’ in column 1 indicates an initialization variable. Placing a ‘T’ in column 1 indicates a tracked variable. Placing a ‘B’ in column 1 indicates that the variable is output as both an initialization and tracked variable. If any variable is an array, the array must first be expanded before indicating the elements as initialization or tracked.

5.3.1.4 INIT File

This file is the real-time CADAC target initialization file. The format of the file is the same as any CADAC plot file. This file contains the initialization data of the target state variables and data link, output generated by subroutines G1I and S2I. The data is written to this file after initialization modules have been processed and before the integration begins. The data contained on this file is for one instance in time.

5.3.1.5 TRACK File

This file is the real-time CADAC target track data link update file. The format of the file is the same as any CADAC plot file. This file contains a time history of data link information, output generated from subroutines G1 and S2. The data rate of the TRACK file is determined by the TABOUT write interval CPP(C(2015)); it should equal the integration step size of the MIL frame. Only the real-time CADAC Test Bed needs this file.

5.3.2 CONVRT

The CONVRT program changes the missile simulation from a CADAC batch program into a single missile subroutine CADRTxx.FOR callable from the MIL frame. The CONVRT program also creates a stand alone Test Bed file such that the code may be tested prior to its inclusion in the MIL frame. Figure 18. illustrates the flow of creating a real-time CADAC file for use in the MIL frame or as a Test Bed.

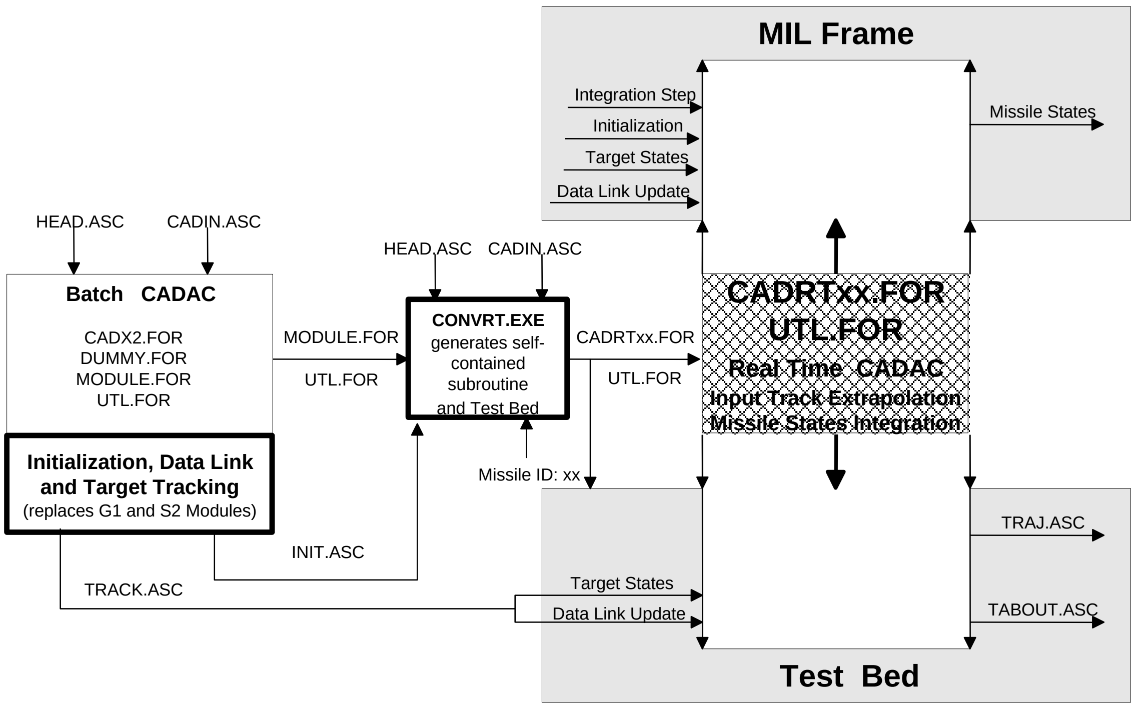

Figure 18. Process Flow of Real-Time CADAC File for use in MIL Frame or Test Bed.

The xx designation on the real-time CADAC filename identifies a particular missile. This designation is appended to all subroutine names and labeled COMMONS within the module file. This designation is not added to the matrix utilities in the UTL.FOR file. NOTE: the real-time CADAC file must be compiled and linked with the CADAC matrix utilities file: UTL3.FOR.

The CONVRT program reduces the executive routine drastically for the real-time CADAC version. To take advantage of the fewer array locations, the C(3510) array is converted into a B(2620) array. The user is cautioned against using a B array in any module subroutines. The locations assigned in the modules remain unchanged, but the executive variables have been compacted into new locations.

CADRTxx.FOR requires from the MIL frame the initialization of its state variables and the input target states and data link. It also requires the time step of the MIL frame in location. CADRTxx.FOR outputs the missile states and status flags. All communication with the MIL frame occurs through the B array.

To check the CADRTxx.FOR subroutine, a Test Bed may be used, optionally created by CONVRT. A particular test case is first run using the batch CADAC that results in the creation of the INIT and TRACK data files. CADRTxx.FOR in the Test Bed version is compiled together with UTL3.FOR is a self-contained program. It requires input from the TRACK data file. It outputs trajectory information on TRAJ.ASC/BIN at the data rate selected by the PPP (C(2005)) and, optionally, an output file TABOUTxx.ASC that records the selected trajectory variables, warning messages and intercept variables.

5.3.2.1 Creating CADRTxx.FOR for Test Bed Check-Out

The Batch CADAC program must first be prepared to generate the two input files INIT.ASC and TRACK.ASC (or in binary form INIT.BIN and TRACK.BIN). They are created from Batch CADAC by appending INITASC (or INITBIN) and TRACKASC (or TRACKBIN) on the option list of the HEAD.ASC file.

Input to the CONVRT program are the MODULE.FOR, HEAD.ASC, and CADIN.ASC files used during the Batch CADAC execution. Also required is target initialization file INIT.ASC.

With these files in the local directory, the CONVRT program can be executed; the program is executed by double-clicking CONVRT in the Windows Explorer or using the Execute CONVRT option from the DIGITAL Visual FORTRAN compiler after the CONVRT source has been compiled and linked.

When the program is executed, the user is informed of the current directory and is prompted for the name of the file containing the user defined modules:

```
Current Directory is :
D:\CAD\Convrt

Input the MODULES.FOR filename:
Default = MODULE.FOR :
```

The user may enter a filename and press <RETURN>, or select the default filename by pressing <RETURN>. If an invalid filename is entered, the user receives the message:

```
* * * Invalid Filename * * *
   Do you wish to continue? (Y or N)
   Default = Y:
```

If the user selects to continue execution (by entering a “y” and pressing <RETURN>, or by simply pressing <RETURN>), the user is prompted again for the file containing the user defined modules, otherwise program execution is terminated.

Once a valid module file is entered, CONVRT prompts the user for the name of the CADIN file associated with the modules:

```
Current Directory is :
D:\CAD\Convrt

Input the CADIN.ASC filename:
Default = CADIN.ASC :
```

The user may enter a filename and press <RETURN>, or select the default filename by pressing <RETURN>. If an invalid filename is entered, the user receives the message:

```
* * * Invalid Filename * * *
   Do you wish to continue? (Y or N)
   Default = Y:
```

If the user selects to continue execution (by entering a "y" and pressing <RETURN>, or by simply pressing <RETURN>), the user is prompted again for the CADIN file, otherwise program execution is terminated.

Once a valid CADIN file is entered, CONVRT prompts the user for the name of the HEAD file associated with the modules:

```
Current Directory is :
D:\CAD\Convrt

Input the HEAD.ASC filename:
Default = HEAD.ASC :
```

The user may enter a filename and press <RETURN>, or select the default filename by pressing <RETURN>. If an invalid filename is entered, the user receives the message:

```
* * * Invalid Filename * * *
   Do you wish to continue? (Y or N)
   Default = Y:
```

If the user selects to continue execution (by entering a "y" and pressing <RETURN>, or by simply pressing <RETURN>), the user is prompted again for the HEAD file, otherwise program execution is terminated.

Once a valid HEAD file is entered, CONVRT prompts the user to select the type of CADRTxx.FOR file to create, MIL Frame or Test Bed:

```
Create Real-Time CADAC for
MIL Frame (1) or Test Bed (2):
Default = MIL Frame (1):   2
```

The user may enter 2 if the Test Bed CADRTxx.FOR file is to be created; otherwise the user may enter 1 and press <RETURN> or simply press <RETURN> to have CONVRT create the MIL frame file. Once the user enters a selection, CONVRT prompts the user for the Missile ID:

Input MISSILE ID -
Return for none:

The user may enter the 2 digit missile ID and press <RETURN> to continue execution or press <RETURN> for no missile ID and continue execution. The missile ID is not a required input, however, for creating multiple real-time CADAC files, the missile ID helps in the identifying each real-time model.

Program execution continues with CONVRT prompting the user for the target initialization file:

Current Directory is :
D:\CAD\Convrt

Input the INITIALIZATION filename:
Default = INIT.ASC :

The user may enter a filename and press <RETURN>, or select the default filename by pressing <RETURN>. If an invalid filename is entered, the user receives the message:

*** * * Invalid Filename * * ***
Do you wish to continue? (Y or N)
Default = Y:

If the user selects to continue execution (by entering a "y" and pressing <RETURN>, or by simply pressing <RETURN>), the user is prompted again for the initialization file, otherwise program execution is terminated. CONVRT then executes to completion without any other interaction with the user.

To execute the CADRTxx.FOR file for Test Bed check-out, the user must compile, link and run the CADRTxx.FOR together with UTL3.FOR. When executing the compiled real-time program, the TRACK file must be in the local directory to provide data link updates to the real-time code. It will write selected parameters and diagnostics to the screen or to the TABOUT.ASC file according to the options selected in the HEAD.ASC file. To compare the trajectories created in the batch mode with the real-time trajectories, the TRAJ.BIN or TRAJ.ASC option in the HEAD.ASC file should be exercised. The smoothing and extrapolation of the track data that arrive at a lower data rate than in the batch mode (every integration step) could cause differences in the trajectories together with the random number selections for the noise models.

5.3.2.2 Creating CADRTxx.FOR for the MIL Frame

To generate the real-time missile fly out subroutine CADRTxx.FOR, callable from the MIL frame, the CONVRT program is executed with the 'MIL Frame' option. The files that must be present in the local directory are MODULE.FOR, CADIN.ASC, HEAD.ASC and INIT.ASC. The version of CADRTxx.FOR created using this method will not execute by itself; this file is included in the MIL frame along with the CADAC matrix utility routines, UTL3.FOR.

When the program is executed, the user is informed of the current directory and is prompted for the name of the file containing the user defined modules:

```
Current Directory is :
D:\CAD\Convrt

Input the MODULES.FOR filename:
Default = MODULE.FOR :
```

The user may enter a filename and press <RETURN>, or select the default filename by pressing <RETURN>. If an invalid filename is entered, the user receives the message:

```
* * * Invalid Filename * * *
   Do you wish to continue? (Y or N)
   Default = Y:
```

If the user selects to continue execution (by entering a "y" and pressing <RETURN>, or by simply pressing <RETURN>), the user is prompted again for the file containing the user defined modules, otherwise program execution is terminated.

Once a valid module file is entered, CONVRT prompts the user for the name of the CADIN file associated with the modules:

```
Current Directory is :
D:\CAD\Convrt

Input the CADIN.ASC filename:
Default = CADIN.ASC :
```

The user may enter a filename and press <RETURN>, or select the default filename by pressing <RETURN>. If an invalid filename is entered, the user receives the message:

```
* * * Invalid Filename * * *
   Do you wish to continue? (Y or N)
   Default = Y:
```

If the user selects to continue execution (by entering a "y" and pressing <RETURN>, or by simply pressing <RETURN>), the user is prompted again for the CADIN file, otherwise program execution is terminated.

Once a valid CADIN file is entered, CONVRT prompts the user for the name of the HEAD file associated with the modules:

```
Current Directory is :
```

```
D:\CAD\Convrt

Input the HEAD.ASC filename:
Default = HEAD.ASC :
```

The user may enter a filename and press <RETURN>, or select the default filename by pressing <RETURN>. If an invalid filename is entered, the user receives the message:

```
* * * Invalid Filename * * *
   Do you wish to continue? (Y or N)
   Default = Y:
```

If the user selects to continue execution (by entering a “y” and pressing <RETURN>, or by simply pressing <RETURN>), the user is prompted again for HEAD file, otherwise program execution is terminated.

Once a valid HEAD file is entered, CONVRT prompts the user to select the type of CADRTxx.FOR file to create, MIL Frame or Test Bed:

```
Create Real-Time CADAC for
MIL Frame (1) or Test Bed (2):
Default = MIL Frame (1):
```

The user may enter 2 if the Test Bed CADRTxx.FOR file is to be created; otherwise the user may enter 1 and press <RETURN> or simply press <RETURN> to have CONVRT create the MIL frame file. Once the user enters a selection, CONVRT prompts the user for the Missile ID:

```
Input MISSILE ID -
Return for none:
```

The user may enter the 2 digit missile ID and press <RETURN> to continue execution or press <RETURN> for no missile ID and continue execution. The missile ID is not a required input, however, for creating multiple real-time CADAC files, the missile ID helps in the identifying each real-time model.

Program execution continues with CONVRT prompting the user for the target initialization file:

```
Current Directory is :
D:\CAD\Convrt

Input the INITIALIZATION filename:
Default = INIT.ASC :
```

The user may enter a filename and press <RETURN>, or select the default filename by pressing <RETURN>. If an invalid filename is entered, the user receives the message:

```
* * * Invalid Filename * * *
```

```
Do you wish to continue? (Y or N)
Default = Y:
```

If the user selects to continue execution (by entering a "y" and pressing <RETURN>, or by simply pressing <RETURN>), the user is prompted again for the initialization file, otherwise program execution is terminated. CONVRT then executes to completion without any other interaction with the user.

5.4 CADAC Execution Utilities

5.4.1 CADIN

CADIN is an intermediate pre-processing utility program that allows the user to generate CADAC input files. CADIN translates the statements in the user generated input file, INPUT.ASC, and outputs a CADIN.ASC file compatible with program CADAC. CADIN verifies each statement; when an invalid statement is encountered, an error message is written to the output file ERROR.ASC along with the line containing the invalid format. The variables contained in the statements found in the INPUT.ASC input file must be contained in a HEAD.ASC file which must be located in the default directory. In addition, the user must ensure that the first reference to a vector/array in the HEAD.ASC contains the dimension of the vector/array.

5.4.1.1 CADIN Execution

Program CADIN is a DOS application that can be added to the DIGITAL Visual FORTRAN TOOL option or can be executed in a DOS window by entering the following statement at the DOS prompt (version number must be included, ex: CADIN2):

```
C:>\CAD\TEST\CADIN
```

When program CADIN is executed, the user is informed of the current directory and is prompted for the name of the variable file:

```
Current Directory is :
C:\CAD\TEST

Enter name of the Free-Formatted input file:
Default = INPUT.ASC :
```

The user may enter a filename and press <RETURN> or select the default filename by pressing <RETURN>. If an invalid filename is entered the user receives the message:

```
* * * Invalid Filename * * *
  Do you wish to continue? (Y or N)
```

Default = Y :

If the user selects to continue execution (by entering a "y" and pressing <RETURN> or by simply pressing <RETURN>), the user is prompted again for the file containing the variables, otherwise program execution is terminated. If the user enters a valid free-formatted file, the following message is displayed on the screen, informing the user of the status of the program:

*** * *Expanding HEAD.ASC File * * ***

The HEAD.ASC file is expanded to include all the elements of an array or a vector. After the HEAD.ASC file is expanded, the "Now processing Commands in INPUT file." is displayed to indicate the program status.

If errors occurred during execution, the user receives a message such as:

```
* * *     5 ERRORS WERE FOUND IN INPUT FILE     * * *
REFER TO ERROR.ASC FILE IN CURRENT DIRECTORY
              PRESS <RETURN> TO CONTINUE
```

5.4.1.2 CADIN Files

The 6 files associated with the execution of program CADIN are INPUT.ASC, HEAD.ASC, INPUT.BAK, CADIN.ASC, INPUT.TP5 and ERROR.ASC.

5.4.1.2.1 INPUT.ASC

INPUT.ASC is both an input and output file of CADIN. This file is a user generated file that contains the statements that provide information on the flight scenario for the simulation. This free-formatted file provides maximum flexibility in the simulation applications. Each record in this file is called a statement. Each statement, which is not case sensitive, can contain up to 132 characters. All statements included in the original INPUT.ASC file are copied to a backup file, INPUT.BAK for archive purposes. A new INPUT.ASC file is created with the statements of the original INPUT.ASC file with the exception of assignment statements, vector statements and function statements. In the case of these statements, if a variable definition is included in the HEAD.ASC for the variable being assigned, this definition is included in the output INPUT.ASC file as a comment. If a comment was already included on the assignment statement, it is replaced with the HEAD.ASC definition. If a definition is not included in the HEAD.ASC file, the original comment for the assignment statement is maintained.

5.4.1.2.2 HEAD.ASC

HEAD.ASC is the file containing the variables used in the INPUT.ASC file. This file must be located in the user's default directory. For a description of this file, refer to Section 3.2.

5.4.1.2.3 INPUT.BAK

INPUT.BAK is created during execution of CADIN. This file is a backup of the input file INPUT.ASC before the comments from the HEAD.ASC file are included due to the execution of CADIN. If an input file with a name other than INPUT.ASC is converted by CADIN, this file will have that filename with a .BAK extension. For example, if the user enters TEST1.ASC for the input file, this file will be named TEST1.BAK.

5.4.1.2.4 CADIN.ASC

The output file CADIN.ASC is a fixed formatted file that is used for input into CADAC. See Appendix A for a description of the contents of this file.

5.4.1.2.5 INPUT.TP5

The output file INPUT.TP5 is identical to CADIN.ASC. This file is created for archive purposes. If an input file with a name other than INPUT.ASC is converted by CADIN, this file will have that filename with a .TP5 extension. For example, if the user enters TEST1.ASC for the output file, this file will be named TEST1.TP5.

5.4.1.2.6 ERROR.ASC

This file contains the errors that occurred during execution of CADIN along with the line that contained the error. NOTE: If an error occurs during execution of CADIN, the default output files are created without the error included.

5.4.2 EQMAP

EQMAP creates a file (EQMAP.ASC) that shows how the variables in a HEAD.ASC (or AHEAD.ASC) file are used in a MODULE.FOR file. EQMAP searches each of the modules for each variable in the HEAD.ASC file that the user specified. EQMAP determines:

- If the variable is found in the current module
- If the variable is an argument for the module
- If the variable is being assigned a value
- If the variable is being used in the assignment of another variable
- The C location to which the variable is equivalenced
- The variable type
- The variable dimension
- If the variable is in a DATA statement
- If the variable is in the criteria of an IF statement
- If the variable is used in a subroutine call

- If the variable name was changed when it was the argument of a subroutine

5.4.2.1 EQMAP Execution

Program EQMAP is a DOS application that can be added to the DIGITAL Visual FORTRAN TOOL option or can be executed in a DOS window by entering the following statement at the DOS prompt (version number must be included, ex: EQMAP2):

```
C:>\CAD\TEST\EQMAP
```

When program EQMAP is executed, the user is informed of the current directory and is prompted for the name of the modules file:

```
Current Directory is :
C:\CAD\TEST

Enter name of modules file (input)
Default = MODULE.FOR :
```

The user may enter a filename and press <RETURN> or select the default filename by pressing <RETURN>. If an invalid filename is entered the user receives the message:

```
* * * Invalid Filename * * *
  Do you wish to continue? (Y or N)
  Default = Y :
```

If the user selects to continue execution (by entering a "y" and pressing <RETURN> or by simply pressing <RETURN>), the user is prompted again for the file containing the modules, otherwise program execution is terminated.

Once a valid modules file is entered, EQMAP prompts for the CADAC variable file:

```
Current Directory is :
C:\CAD\TEST

Enter name of  file containing variables (input)
Default = HEAD.ASC :
```

The user may enter a filename and press <RETURN> or select the default filename by pressing <RETURN>. If an invalid filename is entered, the user receives the message:

```
* * * Invalid Filename * * *
  Do you wish to continue? (Y or N)
  Default = Y :
```

If the user selects to continue execution (by entering a "y" and pressing <RETURN> or by simply pressing <RETURN>), the user is prompted again for the file containing the CADAC modules, otherwise program execution is terminated.

The user is then prompted to enter the name of the file to contain the cross referenced variables:

```
Current Directory is :
C:\CAD\TEST

Enter name of file to contain cross referenced variables (output)
Default = EQMAP.ASC :
```

The user may enter a filename and press <RETURN> or select the default filename by pressing <RETURN>. Once the name of the cross reference file is entered, the program prompts for the variables to cross reference:

```
Enter range of C locations for variables to cross reference

Valid formats (spaces are ignored):
    100      (single variable                               )
    - 100    (all variables up to 100                       )
    100 -    (all variables greater than or equal to 100    )
    99-200   (all variables between 99 and 200, inclusive   )
    ALL      (all variables                                 )

Default = ALL :
```

The user can enter the desired variables to cross reference or can cross reference all the variables. If an invalid entry is made, the user receives the message:

```
* * * ERROR READING VALUE * * *
```

and is prompted again to enter the variables to cross reference. Once the variables are selected, the user is informed of the status of the program by the message:

```
* * * Splitting MODULE.FOR * * *
```

Once the MODULE.FOR is split, the variables in the variable file are displayed on the screen one at a time, informing the user of the program status.

5.4.2.2 EQMAP Output

The output from EQMAP is a list of the variables found in the user selected HEAD.ASC file. If the variable had a definition in the HEAD.ASC this definition is located on the same line as the variable to the right. If the variable was in the range selected by the user to cross reference, the cross reference for the variable is included on the next line, starting in column 23 if the variable file was a HEAD.ASC format or in column 28 if the variable file was a AHEAD.ASC type file. Figure 19 shows an excerpt from an EQMAP.ASC file created from a HEAD.ASC file. As an example, the variable VBEB is located in the subroutines D1I, D1 and G3. In D1I, the variable, which is a real array with a dimension of 3 and a starting C location of 1613, is used in the assignment of another variable (= on right of subroutine name). In D1, the variable, which is a real array with a dimension of 3 and a starting C location of 1613, is found in the calls to the matrix utilities MATMUL and MATABS. In G3, the variable, which is a real array with a dimension of 3 and a starting C location of 1613, is assigned a value (= on left of subroutine name), is found in an IF statement (>) and in the call to the matrix utility MATABS.

Table 9 lists the symbols found in the output file generated by EQMAP and their meaning.

Table 9. EQMAP Symbols.

SYMBOL	INDICATES:
[ ]	the variable is an argument for the current module; the name of the module is located within the brackets
=	the variable is being assigned a value if the "=" is on the left side of the subroutine name; the variable is being used in the assignment of another variable if the "=" is on the right side of the subroutine name
C(Integer Val)	the C location to which the variable is equivalenced
R	the variable is a real
I	the variable is an integer
DP	the variable is a double precision real
(Integer Val)	the dimension of the variable
D	the variable is in a data statement
>	the variable is used in the criteria of an IF statement
:	the variable name was changed when it was the argument of a subroutine; the left side of the ":" is the name of the subroutine and the right side of the ":" is the new name of the variable
;	the end of the information for the current module

```
       1613   VBEB(3)        S      Vehicle velocity in body axis - m/s
                             D1I=R(3) C(1613); D1 R(3) C(1613) MATMUL, MATABS;
                             =>G3 R(3) C(1613) MATABS;
       1616   SBELD(3)
                             D1R(3) C(1616) MATEQL=;
       1619   SBEL(3)        S      Vehicle position wrt point E in body axes - m
                             =D1I R(3) C(1619); =D1 R(3) C(1619);
       1630   FSPB(3)        O      Specific force in body axes - m/s
                             =C2PROP R(3) C(1630); =C2INT R(3) C(1630); D1
                             R(3) C(1630) MATCON=,MATABS;
       1633   VBEL(3)        O      Vehicle velocity in local level axes - m/s
                             D1I R(3) C(1633);=>D1 R(3) C(1633) MATMUL=,MATEQL;
       1636   DVBE           O      Vehicle speed - m/s
                             =A1 R C(1636);=A1DER R C(1636);=C2PROP R C(1636);
                             =C2INT R C(1636); =D1I R C(1636); =D1 = R C(1636)
                             MATABS=; =G2 R C(1636); =G3 R MATABS=;
       1637   ALPHA0X        D      Initial angle-of-attack - deg
                             =D1I R C(1637);
       1638   BETA0X         D      Initial slide slip angle - deg
                             =D1I R C(1638);
       1639   HBE            O      Height above ground - m
                             D1I= R C(1639); D1= R C(1639); =>G2 R C(1639);
       1640   PSIVLX         G      Heading angle - deg
                             D1= R C(1640);
       1641   THTVLX         G      Vertical flight path angle - deg
                             D1= R C(1641);
       1642   PSIVL0X        D      Initial horizontal flight path angl - deg
                             =D1I R C(1642);
       1643   THTVL0X        D      Initial vertical flgith angle - deg
                             =D1I R C(1643);
       1701   AI11           D      Roll moment if inertia - kg*m^2
                             =A1DER R C(1701); =D2 R C(1701);
       1702   AI33           D      Pitch or yaw moment of inertia - kg*m^2
                             =A1DER R C(1702); =D2 R C(1702);
       1704   PPD
                             D2= R C(1704);
       1705   PP             S      Body roll angular velocity - rad/s
                             =C2ROLL R C(1705); =D2 R C(1705) VECVEC; =G3 R C(1705);
       1706   QQD
                             D2= R C(1706);
       1707   QQ             S      Body pitch angular velocity - rad/s
                             =C2PROP R C(1707); =C2INT R C(1707); =D2 R C(1707) VECVEC;
                             =G3 R C(1707);
       1708   RRD
                             D2= R C(1708);
       1709   RR             S      Body yaw angular velocity - rad/s
                             =C2PROP R C(1709); =C2INT R C(1709); =D2 R C(1709) VECVEC;
                             =G3 R C(1709);
       1710   PPX            S      Body roll angular velocity in body axes - deg/s
                             =A1 R C(1710); D2= R C(1710);
       1711   QQX            S      Body pitch angular velocity in body axes - deg/s
                             =A1 R C(1711); D2= R C(1711);
```

Figure 19. Excerpt from an EQMAP.ASC File.

5.4.3 INPUT

INPUT is an intermediate pre-processing utility program that allows the user to generate a free-formatted CADAC file. INPUT translates the statements in the fixed-formatted input file, CADIN.ASC, and outputs a free-formatted INPUT.ASC file compatible with program CADIN.

INPUT verifies each statement; when an invalid statement is encountered, an error message is written to the output file ERROR.ASC along with the line containing the invalid format. The variables contained in the statements found in the CADIN.ASC input file must be contained in a HEAD.ASC file which must be located in the default directory. In addition, the user must ensure that the first reference to a vector/array in the HEAD.ASC contains the dimension of the vector/array.

5.4.3.1 INPUT Execution

Program INPUT is a DOS application that can be added to the DIGITAL Visual FORTRAN TOOL option or can be executed in a DOS window by entering the following statement at the DOS prompt (version number must be included, ex: INPUT2):

```
C:>\CAD\TEST\INPUT
```

When program INPUT is executed, the user is informed of the current directory and is prompted for the name of the fixed-formatted file to transform:

```
This Program Inputs a Fixed-Formatted File and Outputs a Free-Formatted File.
Current Directory is :
C:\CAD\TEST

Enter name of Fixed-Formatted file to transform
Default = CADIN.ASC :
```

The user may enter a filename and press <RETURN> or select the default filename by pressing <RETURN>. If an invalid filename is entered the user receives the message:

```
* * * Invalid Filename * * *
  Do you wish to continue? (Y or N)
  Default = Y :
```

If the user selects to continue execution (by entering a "y" and pressing <RETURN> or by simply pressing <RETURN>), the user is prompted again for the fixed-formatted file, otherwise program execution is terminated. If the user enters a vaid fixed-formatted filename, the user now has the option of entering the free-formatted output filename:

```
Enter name of Free-Formatted file for output:
Default = INPUT.ASC
```

The user continues execution by entering the name of the output file and pressing <RETURN> or by simply pressing <RETURN> (thus creating an output file INPUT.ASC).

After the output filename has been entered, the following message is displayed on the screen, informing the user of the program status:

*** * *Expanding HEAD.ASC File * * ***

The HEAD.ASC file is expanded to include all the elements of an array or a vector found in the HEAD.ASC file. After the HEAD.ASC file is expanded, the message "Now Processing Cards of Input file!" to indicate the program status.

If errors occurred during execution, the user receives a message such as:

```
* * *     5 ERRORS WERE FOUND IN INPUT FILE     * * *
REFER TO ERROR.ASC FILE IN CURRENT DIRECTORY
              PRESS <RETURN> TO CONTINUE
```

5.4.3.2 INPUT Files

The 5 files associated with the execution of program INPUT are INPUT.ASC, HEAD.ASC, CADIN.TP5, CADIN.ASC and ERROR.ASC.

5.4.3.2.1 INPUT.ASC

INPUT.ASC is an output file of INPUT. This file contains the statements that provide information on the flight scenario for the simulation. This free-formatted file provides maximum flexibility in the simulation applications. Each record in this file is called a statement. Each statement, which is not case sensitive, can contain up to 132 characters. All the variables referred to in the statements found in the INPUT.ASC file must be contained in the current HEAD.ASC file. All the statements included in the input CADIN.ASC file are converted to the output INPUT.ASC file with the variable definition from the HEAD.ASC file with the exception of the vector statement. If a comment was already included on the assignment statement, it is replaced with the HEAD.ASC definition. If a definition is not included in the HEAD.ASC file, no comment is included in the INPUT.ASC file for that statement.

5.4.3.2.2 HEAD.ASC

HEAD.ASC is the file containing the variables used in the INPUT.ASC file. This file must be located in the user's default directory. For a description of this file, refer to Section 3.2.

5.4.3.2.3 CADIN.TP5

CADIN.TP5 is created during the execution of INPUT. This file is a backup of the input file CADIN.ASC for archiving purposes. If an input file with a name other than CADIN.ASC is converted by INPUT, this file will have that filename with a .TP5 extension. For example, if the user enters TEST1.ASC for the input file, this file will be named TEST1.TP5.

5.4.3.2.4 CADIN.ASC

CADIN.ASC is a fixed-formatted file that is the output of CADIN. See Appendix A for a description of the contents of this file.

5.4.3.2.5 ERROR.ASC

This file contains the errors that occurred during execution of INPUT along with the line that contained the error. NOTE: If an error occurs during execution of INPUT, the default output files will be created without the error line included.

6. CADAC OUTPUT

During the simulation, a tabular output file and up to 8 of the plot files can be generated at user-selected sample rates. For the sweep methodology option, an additional plot output file is generated. A checkpoint file can be written or read at any time through an input option. These files can be used as input into a variety of CADAC post execution utilities.

6.1 OUTPUT

6.1.1 TRAJ File

The TRAJ plot file is a binary (.BIN extension) or ASCII (.ASC extension) file that contains data in a format that is compatible with KPLOT, PITA, MCAP and QPRINT. Data for up to 70 variables may be stored on the file. The variable data printed on this file is selected through the HEAD.ASC input file. The print rate of the data to this file is taken from the global variable PPP (C location 2005) with the size limit on this file being dependent on the disk space available. The first line of the file is a title record. This record is a 100 character string. The format for an ASCII file requires that the first character in the title be a "1". The binary file may have any other character as the first character, but a " " is preferred. Following the title record is a record with three integers. The integers written to the file are C locations 1982, 1983 and the executive variable NPLOTVAR. Variable NPLOTVAR indicates the number of plot variables whose data is written to the file.

Following the integer data are the character acronyms for the variables whose data is written to the file. These acronyms are 8 characters in length and are separated by a string of 8 blanks. The variable data follows the acronym data. At a given time, the value of each selected variable is written to the file. Time is always the first variable printed to TRAJ by CADAC. The data is written to this file during the trajectory simulation at user-designated intervals.

When a trajectory simulation is completed by program CADAC, a data set with time set to -1.0 is written to the TRAJ plot file. This record acts as a separator between the sets of trajectory data. In the KPLOT graphics program, for example, the -1.0 record indicates that the following data should be a separate segment from the previous. At the end of the file, another -1.0 time record is written to indicate that the end of the run has been reached. The data displayed with the final -1.0 record is the maximum value for each variable generated during the most recent trajectory simulation.

6.1.2 STAT File

The STAT plot file is a binary (.BIN extension) or ASCII (.ASC extension) file that contains data in a format that is compatible with the HISTO and BIVAR options of the KPLOT graphics program as well as the ELLIPSE program. Data for up to 70 variables may be stored on the file. Data values are output only at each stage and at impact along the trajectories. The first line in this file is a title record. This record is a 100 character string. The ASCII formatted files require that the first character in the title be a "1". The binary files in this format may have any other character as the first character but a " " is preferred. Following the title record is a record with three integers. The integers written to the file are C locations 1982, 1983 and the executive variable NPLOTVAR. Variable NPLOTVAR indicates the number of plot variables whose data is written to the file.

Following the integer data are the character acronyms for the variables whose data is written to the file. These acronyms are 8 characters in length and are separated by a string of 8 blanks. The variable data follows the acronym data. At a given time, the value of each selected variable is written to the file including the current run number followed by the current group number. Time is always the first variable printed to STAT by CADAC. The data is written to this file only at each stage and at the end of the trajectory simulation.

6.1.3 CSAVE.ASC

During the CADAC execution, the state of the trajectory simulation can be saved. When this option is selected, the state data is saved to a file named CSAVE.ASC. The presence of a SAVE statement in the input file INPUT.ASC causes all global C locations, C(1-3510), to be written to CSAVE.ASC at the trajectory time when the SAVE statement was encountered. The proper use of this option can increase the efficiency of running certain classes of sensitivity studies, Monte Carlo runs, submunitions dispensing, etc.

6.1.4 TABOUT.ASC

The tabular output file contains a variety of data. The contents of the file are controlled by the program user. Data displayed on this file can include: a list of the input card deck as read by the executive code; a maximum of 16 variable values; and integration messages. In addition, trajectory messages generated by the module code can be written to the TABOUT.ASC by using a WRITE statement with a unit number of 6. Program options allow the user to display the data on the terminal screens as the program executes or to place the data in an output file. The rate the data is printed to the file is taken from the global variable CPP (C location 2015).

6.1.5 RANVAR.ASC

The RANVAR.ASC file contains the values assigned to random, stochastic variables using a type 3 or type 11 card. These calculated values are written to the file at each time step for all such values.

6.1.6 IMPACT.ASC

The IMPACT.ASC file contains the data generated by the sweep methodology. This file has the same format as the TRAJ file generated by the CADAC executive and contains the plot variable data that is generated at the end of the simulated trajectory. The generation of this file is controlled by the sweep modules. For all sweep modes, the file will contain at least NUM sets of data for each outer trajectory.

6.1.7 IMPACT10.ASC Scratch File

The IMPACT10.ASC scratch file is produced by the CADAC executive when the sweep methodology is activated. This file, which contains the criteria value, inner loop value and outer loop value for one or more trajectories, provides communication between the CADAC executive modules and the sweep modules called by the user supplied modules. This file is written with a list-directed format and contains the data in the following order:

C(RANGNO)   C(ANGLNO)   C(CRITNO)

Data is written to this file by the executive module STGE3 only when the mode OUT or ALL is selected and LCONV is greater than 2. In this module, the LCONV variable is set by the G4 module which is also called from STGE3 or in the executive routine. A value of 2 signals that the trajectory simulation was stopped, usually due to achieving the point of closest approach or due to impact. This file is rewound at the start of each ray's calculations and deleted at the end of the CADAC execution by the sweep modules.

6.1.8 IMPACT7.ASC Scratch File

The IMPACT7.ASC scratch file is produced by the CADAC executive when the sweep methodology is activated. The contents of this file are used to produce the IMPACT.ASC output file and provides communication between the CADAC executive modules and the sweep modules. This file is written with a list-directed format and contains the same data that is written to the plot files. Data is written to this file by the executive module STGE3 during all of the sweep options when LCONV is greater than 2. In this module, the LCONV variable is set by the G4 module which is also called from STGE3 or in the executive routine. A value of 2 or greater signals that the trajectory simulation was stopped, usually due to achieving the point of closest approach or due to impact. This file is rewound at the start of each trajectory simulation and deleted at the end of the CADAC execution by the sweep modules.

This file is used as communication between the sweep modules and the CADAC executive. The C array contains the initialization values for the next trajectory when the sweep logic is performed. Without this file, the sweep logic would not have access to the data generated by the trajectories. The contents of this file are used by the sweep logic to create the IMPACT.ASC file.

6.1.9 INIT File

The INIT data file is a binary (.BIN extension) or ASCII (.ASC extension) file that contains data in a format similar to other CADAC plot files TRAJ and STAT. Data for up to 70 variables may be stored on the file. The variable data printed on this file is selected through the HEAD.ASC input file. The first line of the file is a title record. This record is a 100 character string. The format of an ASCII file requires that the first character in the title be a “1”. The binary file may have any other character as the first character, but a “ “ is preferred. Following the title record is a record of three integers. The integers written to the file are C locations 1982, 1983 and the executive variable NPLOTVAR. Variable NPLOTVAR indicates the number of variables whose data is written to the file.

Following the integer data are the character acronyms for the variables whose data is written to the file. These acronyms are 8 characters in length and are separated by a string of 8 blanks. The variable data follows the acronym data. The data is printed in this file only once: after the initialization modules have been processed and before the integration begins. Time is always the first variable printed to INIT by CADAC.

6.1.10 TRACK File

The INIT data file is a binary (.BIN extension) or ASCII (.ASC extension) file that contains data in a format similar to other CADAC plot files TRAJ and STAT. Data for up to 70 variables may be stored on the file. The variable data printed on this file is selected through the HEAD.ASC input file. The print rate of the data to the file is taken from the global variable CPP (C location 2015) with the size limit on this file being dependent on the disk space available. For this file, the CPP variable should be set to the approximate MIL frame integration step size and remain unchanged throughout the simulation. The first line of the file is a title record. This record is a 100 character string. The format of an ASCII file requires that the first character in the title be a “1”. The binary file may have any other character as the first character, but a “ “ is preferred. Following the title record is a record of three integers. The integers written to the file are C locations 1982, 1983 and the executive variable NPLOTVAR. Variable NPLOTVAR indicates the number of variables whose data is written to the file.

Following the integer data are the character acronyms for the variables whose data is written to the file. These acronyms are 8 characters in length and are separated by a string of 8 blanks. The variable data follows the acronym data. At a given time, the value of each variable is written to the file. Time is always the first variable print to TRACK by CADAC. The data is written to the file during trajectory simulation at user-designated intervals. Unlike the TRAJ file, the TRACK file contains data for only one trajectory, since the real-time CADAC application only simulates one missile in real-time.

6.2 CADAC Post Processing Utilities

Several programs are available to process the results of a CADAC execution. Each of these programs are addressed separately.

6.2.1 QPRINT

The QPRINT utility provides the user with a method to view portions of the data produced by CADAC. This data is in the TRAJ or STAT file format as discussed in Sections 6.1.1 and 6.1.2, respectively. An interactive input screen allows the user to manipulate the data file contents for viewing and/or output purposes. The contents of the data files can be displayed on the screen and/or written to an output file. The data can be manipulated such that only portions of the data are output. A time increment can be used to determine how much data is pulled from the file for output. Up to 36 variables can be selected for data display. The formats for each of these variables can be modified by the user.

QPRINT is a Windows application that can be added to the DIGITAL Visual FORTRAN TOOL option. In addition, QPRINT execution can be initiated by using the Icon in Windows or by double clicking on the QPRINT executable file in Windows EXPLORER. QPRINT is one of several programs called by the SWEEP application. For more information on QPRINT execution, refer to the on-line help in the QPRINT menu.

6.2.2 KPLOT

The KPLOT utility provides the user with a method to graphically view the data produced by CADAC. KPLOT has 4 graphical display options: two dimensional (2DIM) plots, strip charts, histograms and bivariate plots. In addition to the 4 graphical display options, the user may also start the PITA, GLOBE and WinDRAW applications. KPLOT is a Windows 95/98/NT application that is used in conjunction with the DrawOLE graphics engine. KPLOT execution can be initiated by clicking on the KPLOT icon in the Windwos Start Menu, by using the RUN option on the Windows Start Menu or by double clicking on the KPLOT executable file in the Windows Explorer. For information on KPLOT execution, refer to the on-line help in the KPLOT menus.

6.2.2.1 2DIM Plots

The 2DIM option allows the user to create 1 or 2 plots on a single screen with each plot having 1 to 3 dependent variables. From within the DrawOLE screen, the user can make modifications such as: changing the characteristics of the plotted trajectory (including changing the line styles, colors and thicknesses) and changing the label and legend fonts. Input to 2DIM is the TRAJ data file created by CADAC.

6.2.2.2 CHARTS

The CHARTS option in KPLOT allows the user to plot up to 12 strip chart plots (all having the same independent variable) on one screen. The user has the option to include a zero grid line on the plots. Input to CHARTS is the TRAJ data file created by CADAC.

6.2.2.3 HISTO

The HISTO option provides a 1-dimensional histogram plot for the data from the STAT file generated from a CADAC multi-run engagement. The option superimposes a normal, exponential or Rayleigh curve over the histogram. The user can make modifications to the plot from within the DrawOLE screen.

6.2.2.4 BIVAR

The BIVAR option generates and displays error ellipses around user specified variables. This option generates target centered CEPs and/or centroid centered CEPs, both with user selected containment probabilities for up to 6 pairs of data. Like the previous KPLOT options, the user can make modifications from within the DrawOLE screen. Input to BIVAR is the STAT data file created by CADAC.

6.2.3 PITA

The PITA application provides the user the ability to graphically display and animate the data produced by CADAC. PITA is a Windows 95/98/NT application that is used in conjunction with the DrawOLE graphics engine. PITA execution can be initiated by using the PITA option in the KPLOT menu, or by double clicking on the PITA executable file in the Windows Explorer. The PITA application allows the user to create 3 dimensional plots using an Euler axis system (with z being positive down) as well as the normal axis system (z is positive up). The user can also apply stringers from the trace to the floor of the axis system and rotate the system around the x-axis and the y-axis. The PITA application also allows the user to place up to 12 strip charts along the right side of the screen. From within the DrawOLE window, the user may animate the displayed trajectories and strip charts at three different speeds. From within the DrawOLE screen the user can make modifications such as: removing/adding stringers, changing the rotation angles, changing the characteristics of the plotted trajectory and adding elements to the plot. Input to PITA is the TRAJ data file created by CADAC. For information on PITA execution refer to the on-line help available in PITA.

6.2.4 GLOBE

The GLOBE application provides the user the ability to graphically display trajectories over a globe background. GLOBE is a Windows 95/98/NT application that is used in conjunction with the DrawOLE graphics engine. GLOBE execution can be initiated by using the GLOBE option in the KPLOT menu, or by double clicking the GLOBE executable file in the Windows Explorer. The GLOBE application allows the user to display up to 4 different trajectories along with stringers, time hacks and a color trace over a globe background. From within the DrawOLE window, the user may rotate the globe to another position, add latitude and longitude arcs to the globe and add terrain to the globe. For information on GLOBE execution refer to the on-line help available in GLOBE.

6.2.5 MCAP

The MCAP program performs analysis of TRAJ data generated by CADAC multi-run/Monte Carlo simulations. The analysis provides an "averaging of the trajectories" and determines the effects of trajectory variations at various times along the trajectories. MCAP is also used to “average the trajectories” for a given aimpoint associated with a Monte Carlo Sweep

CADAC execution. MCAP is one of several programs called from within the SWEEP application.

6.2.5.1 MCAP Program Execution

MCAP is a Windows application that can be added to the DIGITAL Visual FORTRAN TOOL option. In addition, MCAP execution can be initiated by using the RUN option in Windows or by double clicking on the MCAP.EXE file in Windows EXPLORER. For information on executing MCAP, refer to the on-line help in the MCAP menu.

6.2.5.2 MCAP Input and Output Files

Three input files and three output files are associated with program MCAP. The input file (default name TRAJ.BIN) contains the data to be analyzed. An optional input file, TLE.ASC, containing the Target Location Errors (TLE) may be utilized if selected by the user. The MCAP.LOG file contains the current parameter values; this file is accessed by the OPEN command under the FILE menu option and generated by the SAVE command under the FILE menu option. For more information on the log file, refer to the MCAP on-line help. The RUN menu option initiates the analysis and produces a file with the default name MCAP.ASC. The MCAP_ERR.LIS is produced by MCAP and contains any error messages that occur during the statistical sampling and analysis.

6.2.5.2.1 MCAP Input Data File

The input data file has the same format as the CADAC-produced TRAJ file. Program MCAP accepts either a binary or ASCII input file. The trajectory data on the file must conform to the following restrictions:

1) Time must be positive and increasing within a single trajectory set
2) Trajectory data sets must be terminated either by a record having "-1.0" as time or by the end of file

The file must contain a variable that indicates the current trajectory time. Typically, the first variable on the TRAJ file is TIME, but this placement is not a requirement. Files produced by the CADAC program meet these restrictions.

6.2.5.2.2 TLE.ASC Format and Contents

The TLE.ASC input file is an ASCII file containing a title (maximum 80 characters) in the first record. This title record is primarily for identification to the user and is only displayed in the menu screen. The remaining records on the file contain the following target location error data values: σ_x, ρ_{xy}, ρ_{xz}, ρ_{yx}, σ_y, ρ_{yz}, ρ_{zx}, ρ_{zy}, σ_z. The data must be in this order and each value separated by at least one space or by a comma. The data may exist on a single record or multiple records. This file is an option input file; the incorporation of the TLE data is selected by setting the TLE option to YES in the interactive screen. Filenames other than the default (TLE.ASC) may be entered . Figure 20 displays the contents of a sample TLE.ASC file.

```
      Sample TLE data  (20 NOV 93)
      1.4    0.5   0.3
      0.5    0.8  -0.3
      0.3   -0.3   1.0
```

Figure 20. Sample TLE.ASC File.

6.2.5.2.3 MCAP.ASC Format

The MCAP.ASC file has the same format as the CADAC-produced TRAJ file with the exception that it is always an ASCII file. The MCAP.ASC title is composed from the TRAJ title with the character "1" placed in the first column to indicate the ASCII format. The record following the title contains the number of data variables stored on the MCAP.ASC; a maximum of 70 variables is allowed per file. The next group of records contains the MCAP.ASC variable acronyms that define the data and indicate the order on the file. Time is the first variable in the set and has the acronym, "TIME". The number of points is the second variable and has the acronym, "NO. PNTS". The remaining variables are dependent on the 1-, 2- or 3-dimensional variables selected by the user during the interactive session. The remaining file records contain the actual data values computed by the analysis for these variables. A data value for each variable is generated at every sample time. The data is grouped by sample time and written to the file in ascending time.

6.2.5.2.3.1 MCAP.ASC Acronyms

The MCAP.ASC acronyms provided for the statistical parameters are dependent on the original TRAJ acronyms, the dimension of the statistic and the parameter type. An acronym basis is formed by removing any numerics located at the end of the original TRAJ acronym and, if necessary, truncating the acronym to either 6 or 7 characters. For multi-dimensionally related variables, the acronym basis is formed from the original TRAJ acronym for the first variable of the data set. A character code corresponding to the statistical parameter is then appended to the acronym basis.

For 1-dimensional variables, the mean, standard deviation, mean plus standard deviation and mean minus standard deviation parameters are generated by the analysis. The character codes for these parameters are "M", "S", "SU" and "SL", respectively. For 2-dimensional variables, the error ellipse major axis, the minor axis and the angle of orientation are generated by the analysis. The character codes for these parameters are "A", "B" and "PX", respectively. For the 2-dimensional variables, the user may select to have the mean-centered CEP or the zero-centered CEP calculated; the character codes for these parameters are "CM" and "CZ", respectively. For the 3-dimensional variables, the variable means and specific elements from the correlation coefficient matrix are generated by the analysis. The character codes indicating the mean of the variables are "1M", "2M" and "3M"; the numbers signify the variable position as entered during the interactive session. The following elements of the covariance matrix are generated by the analysis: (1,1), (2,2), (3,3), (1,2), (1,3), (2,3); the character codes for these parameters are "11", "22", "33", "12", "13" and "23", respectively. Table 10 contains examples of the variable acronyms used on the MCAP.ASC.

Table 10. MCAP.ASC Acronym Examples.

TRAJ Acronym	MCAP.ASC Basis	MCAP.ASC Acronym	Statistic Definition
THTVLX	THTVLX	THTVLXM THTVLXS THTVLXSU THTVLXSL	THTVLX Mean THTVLX Standard Deviation THTVLX Mean + (R * Sigma) THTVLX Mean - (R * Sigma) (R is a user selected multiplication factor)
ALPHA BETA	ALPHA	ALPHAA ALPHAB ALPHAPX ALPHACM or ALPHACZ	Error Ellipse Major Axis Error Ellipse Minor Axis Error Ellipse Orientation Mean Centered CEP or Zero Centered CEP
SBEL1 SBEL2 SBEL3	SBEL	SBEL1M SBEL2M SBEL3M SBEL11 SBEL22 SBEL33 SBEL12 SBEL13 SBEL23	SBEL1 Mean SBEL2 Mean SBEL3 Mean SBEL1 Standard Deviation SBEL2 Standard Deviation SBEL3 Standard Deviation Correlation Coefficients: SBEL1 wrt SBEL2 SBEL1 wrt SBEL2 SBEL2 wrt SBEL3

6.2.5.3 MCAP Analysis Methodology

This section discusses the data analysis performed by the MCAP program during execution. The analysis consists of accumulating data from all of the trajectories on the input file ("sampling" each trajectory), then calculating statistical parameters for this data set.

When the RUN command is entered, the user selections are checked for validity. If errors are found, the program displays error messages and prompts for commands. If the input data is valid, the list of times at which the trajectory data is to be sampled (sample times) are calculated from the user-input first time, last time and delta time interval. The remainder of the analysis is performed in two steps: (1) the trajectory data is sampled and split into data sets corresponding with each sample time and (2) the statistics are calculated for each data set. The results of the statistical calculations are written to the MCAP.ASC output file.

6.2.5.3.1 MCAP Data Sampling Process

The data sampling process is applied to all of the trajectories on the input file. In the process, the program reads the TRAJ input and searches for the sample times within each trajectory. When data for a sample time is located, the data is written to a temporary file corresponding with the sample time. The sampling process divides each trajectory into three sections: the start, the middle and the end of the trajectory. These sections exhibit special cases when locating data for the sample times.

6.2.5.3.1.1 Trajectory Start

Several special cases must be considered at the start of the trajectory. Figure 21 illustrates these cases with a time line showing sample times (labeled with S) and times of the trajectory data from the TRAJ file (labeled with T). The first case is determining the sample times (S1 and S2 for example) that occur prior to the first trajectory time (T1); trajectory data does not exist at these sample times and therefore data is not written to the corresponding temporary files. Another special case occurs at sample time S3, the time just prior to T1. If the time difference between S3 and T1 is less than half the user-input time interval, then the trajectory data at T1 is considered to correspond with the S3 sample time and is written to the temporary file. This test also detects the case where T1 is equal to S3. If the time difference is greater than or equal to half the time interval, then no trajectory data is considered to exist at the S3 sample time and the next sample time is evaluated. By definition, once T1 is less than the current sample time under evaluation, the middle of the trajectory is reached. For example, the S4 sample time belongs to the middle of the trajectory.

```
S1     S2     S3                   S4
O ____ O ____ O ____ X ____ X ____ O ____ X _________ . . .

                     T1     T2            T3
```

Figure 21. Trajectory Start Illustration.

6.2.5.3.1.2 Trajectory Middle

Two cases are considered for the middle of the trajectory: (1) multiple trajectory times occurring between sample times and (2) multiple sample times occurring between trajectory times. This is illustrated by Figure 22. The sample times in the middle of the trajectory are processed by locating a trajectory time on each side of the current sample time. In the illustration, when processing the S12 sample time, the trajectory data is read until T17 is reached; at this instance, T16 is less than S12 and T17 is greater than S12. The time differences between S12 and T16 and between S12 and T17 are calculated and the nearest point (in time) is selected and written to the temporary file. Then the next sample time in the list (S13) is tested to determine if it is located between T16 and T17. If T17 is less than S13, then the trajectory data is read from the file until a trajectory time greater than S13 is reached. If two sample times fall between the same pair of trajectory times (for example, S13 and S14 in the illustration), then the same trajectory times are used for both sample times; the trajectory time nearest to the sample time is selected and written to the temporary file.

```
                               S12 S13   S14
. . .  X ____ X ____ X ____ X _O _X ____ O ____ O ____ X _________ . . .

       T13    T14    T15    T16 T17             T18
```

Figure 22. Trajectory Middle Illustration.

6.2.5.3.1.3 Trajectory End

The end of the trajectory is determined by reaching either the end of the sample times, the end of the trajectory by detecting a -1.0 trajectory time or the end of the trajectory by reaching the end of the TRAJ file. If the last sample time is processed prior to reaching the end of the trajectory, then trajectory data is read from the TRAJ until the -1.0 time record or end of file is reached.

If the end of the trajectory is reached prior to the end of the sample times, then the last time data in the trajectory is written to the temporary files for all remaining sample times. Because of this feature, if a last sample time is much greater than the last file time, the temporary data file generated at this sample time will contain the last data record from each trajectory on the input file; for most CADAC scenarios, this file will be the same as the IMPACT file produced by CADAC.

For all cases, the program prepares to process the next trajectory on the file until the end of file occurs. When end of file is detected, the sampling processes is completed.

6.2.5.3.2 MCAP Statistical Calculations

In the statistical calculations, data is read from a temporary file generated by the sampling process and the statistical parameters are calculated from this data. This process is repeated for all temporary files generated by the sampling process. The dimension of the user-selected variables dictate the statistical parameters that are calculated. For every 1-dimensional variable selected, the program computes the mean, standard deviation, mean plus the standard deviation and the mean minus the standard deviation. For each pair of 2-dimensional variables, the program computes the lengths of the containment ellipse major and minor axis, the containment ellipse orientation and the CEP (if selected by the user). For each set of 3-dimensional variables, the program calculates the means and standard deviations of all three variables and the covariance coefficients.

6.2.5.3.2.1 1-Dimensional Mean (M)

The mean of the 1-dimensional variables is computed by summing the data values for the variable (from the temporary file) and dividing the sum of the number of values in the data set:

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

6.2.5.3.2.2 1-Dimensional Standard Deviation (S)

The standard deviation for 1-dimensional variables is computed with the following equation (*CRC Standard Mathematical Tables*, 28th Edition, 1987):

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i)^2 - n(\overline{x})^2}{n-1}}$$

These equations were implemented with a minor adjustment. Due to the computer representation of numbers, a very small negative number sometimes occurs under the radical. MCAP traps this error and sets the value to 0.0.

6.2.5.3.2.3 1-Dimensional Mean Plus Standard Deviation (SU)

The mean plus standard deviation (SU) parameter is computed from the user-selected multiplication factor set with the SF command, the mean calculated from the methodology of Section 6.2.5.3.2.1 and the standard deviation calculated from the methodology of Section 6.2.5.3.2.2. The SU parameter is calculated from these values by using the following equation:

$$SU = \overline{x} + (SFactor * \sigma)$$

6.2.5.3.2.4 1-Dimensional Mean Minus Standard Deviation (SL)

The mean minus standard deviation (SL) parameter is computed from the user-selected multiplication factor set with the SF command, the mean calculated from the methodology of Section 6.2.5.3.2.1 and the standard deviation calculated from the methodology of Section 6.2.5.3.2.2. The SL parameter is calculated from these values by using the following equation:

$$SL = \overline{x} - (SFactor * \sigma)$$

6.2.5.3.2.5 2-Dimensional Major Ellipse Axis (A)

The major ellipse axis calculated for 2-dimensional variable sets is the length of the major axis of a mean-centered ellipse enclosing a user-input percentage of the statistically inferred distribution of the sample data. The derivation of ellipse radius (R) in *CADAC Graphic Program Modification, Phase II, Volume 1: Statistical Analysis Programs* (Engineering Report ER-TC-SEU 87-02,1987) provides the equation for R in polar coordinates:

$$R = \sqrt{\frac{C^2(1-\rho^2)}{1-\rho\sin(2\theta)}} \qquad (1)$$

where

$$C^2 = -2\ln\left(1 - \frac{P}{100}\right)$$

$$\rho = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

$$\sigma_{xy} = \frac{\sum_{i=1}^{n}\left(x_i y_i\right) - n\bar{x}\bar{y}}{n-1}$$

and P is the user-input percentage. The maximum R value is obtained by evaluating equation (1) with θ = 0. Transforming back into Cartesian coordinates and solving for the major axis length produces:

$$A = 2R\sigma_x$$

These equations were implemented with a minor adjustment. By definition, the correlation coefficient is in the interval [-1, 1]. Due to the computer representation of numbers, the difference $(1-\rho^2)$ sometimes results in a very small, negative number. MCAP corrects this value to 0.0 and writes a message to the MCAP_ERR.LIS file.

If the option to incorporate Target Location Errors (TLE) into the statistical calculations has been selected with the TL command, then the standard deviations and correlation coefficient are loaded into a 3x3 matrix and added to the TLE matrix:

$$\begin{bmatrix} \sigma_x^2 & \rho & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}_{\mathrm{MCAP}} + \begin{bmatrix} \sigma_x^2 & \rho_{xy} & \rho_{xz} \\ \rho_{yx} & \sigma_y^2 & \rho_{yz} \\ \rho_{zx} & \rho_{zy} & \sigma_z^2 \end{bmatrix}_{\mathrm{TLE}} \tag{2}$$

The resulting ρ and σ_x are then used in the calculation of the major ellipse axis length as previously discussed.

6.2.5.3.2.6 2-Dimensional Minor Ellipse Axis (B)

The minor ellipse axis calculated for 2-dimensional variable sets is the length of the minor axis of a mean-centered ellipse enclosing a user-input percentage of the statistically inferred distribution of the sample data. This length, B, is generated using the same method discussed in Section 6.2.5.3.2.5 for generaing A. However, after the transformation back to Cartesian coordinates is performed, the solution for the length of the minor axis is produced by the equation:

$$B = 2R\sigma_y$$

If the incorporation of TLE data has been selected, then the TLE data is added to the standard deviations and correlation coefficient as shown in equation (2) of Section 6.2.5.3.2.5. The resulting ρ and σ_y are then used in the calculation of the minor ellipse axis length as previously discussed.

6.2.5.3.2.7 2-Dimensional Ellipse Orientation (P)

This statistic is the mean-centered containment ellipse angular orientation (in degrees) referenced to the first variable axis. Equation 8 from "Integration of the General Bivariate Gaussian Distribution Over and Offset Circle" (DiDonato and Jarnagin, 1961) is used to solve for the angle:

$$\theta = \frac{1}{2} TAN^{-1}\left[\frac{2\rho\sigma_x\sigma_y}{\sigma_x^2 - \sigma_y^2}\right]$$

where ρ is the correlation coefficient defined in Section 6.2.5.3.2.5. This value of theta is then converted to degrees. NOTE: If $(\sigma_x^2 - \sigma_y^2) = 0.0$, then θ = 45 degrees.

If the incorporation of TLE data has been selected, then the TLE data is added to the standard deviations and correlation coefficient as shown in equation (2) of Section 6.2.5.3.2.5. The resulting ρ, σ_x and σ_y are used in the calculation of the ellipse orientation as previously discussed.

6.2.5.3.2.8 2-Dimensional CEP (C)

Current CEP methodologies were researched to locate a methodology suitable for program MCAP. For this statistical evaluation, the CEP is defined as the radius of a circle (either mean-centered or arbitrarily placed) whose volume contains 50% of an elliptical Gaussian distribution. In program MCAP, the user selects either the distribution mean or the origin, (0,0), as the CEP center location. Many of the methodologies located by the research defined the CEP as the radius of a circle containing 50% of the sample data; as this is not the desired CEP definition, these methodologies were rejected. The majority of the remaining CEP methodologies restricted the statistical distribution to the circular Gaussian distribution where $\sigma_x = \sigma_y = \sigma$; these were also rejected as the statistical distribution encountered by MCAP is elliptical. Of the remaining documentation, only "Integration of the General Bivariate Gaussian Distribution over an Offset Circle" by DiDonato and Jarnagin (1961) provided for an offset CEP circle, therefore this method was selected for implementation into program MCAP.

DiDonato and Jarnagin provided two formulas (Equations 5 and 13) for calculating the integral of the uncorrelated elliptical Gaussian distribution over the area of an arbitrarily centered circle in the zero X-Y plane. The article discussed the author's implementation of Equation 13 and provided a table of test cases comparing the author's implementation with another integration methodology (Table 1 from DiDonato and Jarnagin,1961). Equation 13 was implemented in a stand alone test program but failed to produce the same results published in the article. Further study of the reference revealed questions concerning the correctness of Equations 9 and 10. Therefore Equation 5 from DiDonato and Jarnagin:

$$P = \frac{\frac{R}{\sigma_x}}{\sqrt{2\pi}} \int F(t)[\, Erf\,(d_{11}) - Erf\,(d_{01})\,]\, t\, dt$$

where

$$F(t) = e^{-\frac{1}{2}\left[\frac{h - R(1-t^2)}{\sigma_x}\right]^2} + e^{-\frac{1}{2}\left[\frac{h + R(1-t^2)}{\sigma_x}\right]^2}$$

$$d_{11} = \frac{k + Rt\sqrt{2-t^2}}{\sqrt{2}\sigma_y}$$

$$d_{01} = \frac{k - Rt\sqrt{2-t^2}}{\sqrt{2}\sigma_y}$$

was then implemented in a test program. This implementation produced results similar to those published in Table 1 of DiDonato and Jarnagin. While the implementation of Equation 5 responded extremely well for the given test cases, the implementation has difficulties when the function attains the shape of a spike with virtually all of the area within 1/100 of the range of integration [h + R, h - R] (See note (a) on page 375 of DiDonato and Jarnagin). When the input parameters generate this behavior, the integration returns a probability of 0.0 for the given radius. Limits were placed on the input variables to detect the cases that cause this behavior. These limits are:

$$\frac{1}{6} \le \frac{\sigma_x}{\sigma_y} \le 6 \qquad 0 \le \frac{|h|}{\sigma_x} \le 20 \qquad 0 \le \frac{|k|}{\sigma_y} \le 20$$

With these limits, the implementation was deemed adequate for calculating the CEP in MCAP.

6.2.5.3.2.8.1 CEP Calculation Implementation

The CEP4_CALC module controls the calculation of the CEP statistic given the X and Y means, sigmas and correlation of the distribution and the center location of the CEP circle. The methodology from DiDonato and Jarnagin requires an uncorrelated Gaussian distribution located at the origin. Therefore the first step performed by this controlling module is to rotate the system to uncorrelate the data. Then the system is translated so that the distribution is centered at the origin. Next, the input data is tested to insure that it is within the limits defined for the calculations. Since the DiDonato and Jarnagin methodology produces the probability for a given radius and not the radius for a given probability, an iteration technique is initiated to locate the CEP radius. An upper and lower estimate of the CEP is calculated and the corresponding probabilities computed. If the desired probability, 50%, falls above the upper bound, then the current upper bound is loaded as the lower bound and a new upper bound calculated. If the desired probability falls below the lower bound, then the current lower bound is loaded as the upper bound and a new lower bound computed. This continues until bounds for the 50% probability are found. Once the radii bound the 50% probability, then interpolation is used to locate a radius that produces the 50% probability within a specified tolerance. This interpolation increases until the tolerance level is satisfied.

The module PROB_CALC, the driver for the implementation of Equation 5 from DiDonato and Jarnagin, calculates the integral of uncorrelated, elliptical Gaussian distribution over a circle, given the distribution X and Y sigmas, the circle location and circle radius. This module assumes that the distribution is centered at the origin. The integral was solved using the method of Gaussian quadratures; an explanation of this method can be found in *Numerical Analysis* (Burden, Faires and Reynolds, 1991). The roots and corresponding coefficients for the Gaussian quadrature methodology can be found in *CRC Standard Math Tables* (28th, 1987). The integral was divided and modules were written to evaluate each part. The error function, Erf, was evaluated from the complementary error function, Erfc; the module calculating Erfc was provided by *Numerical Recipes in FORTRAN: the Art of Scientific Computing* (Press, 1992), page 214.

6.2.5.3.2.9 3-Dimensional Means (1M, 2M and 3M)

For each 3-dimensional variable set, the means of each variable are calculated by the analysis. The means are calculated using the same methodology as discussed Section 6.2.5.3.2.1.

6.2.5.3.2.10 3-Dimensional Standard Deviations (11, 22 and 33)

For each 3-dimensional variable set, the standard deviation of each variable is calculated by the analysis. The standard deviation calculation is the same as discussed in Section 6.2.5.3.2.2.

If the option to incorporate Target Location Errors (TLE) into the statistical calculations has been selected with the TL command, then the standard deviations and correlation coefficients are calculated and loaded into a 3x3 matrix. The TLE matrix is then added to this matrix using the following equation:

$$\begin{bmatrix} \sigma_x^2 & \rho_{xy} & \rho_{xz} \\ \rho_{yx} & \sigma_y^2 & \rho_{yz} \\ \rho_{zx} & \rho_{zy} & \sigma_z^2 \end{bmatrix}_{\text{MCAP}} + \begin{bmatrix} \sigma_x^2 & \rho_{xy} & \rho_{xz} \\ \rho_{yx} & \sigma_y^2 & \rho_{yz} \\ \rho_{zx} & \rho_{zy} & \sigma_z^2 \end{bmatrix}_{\text{TLE}}$$

The resulting standard deviations and correlation coefficients are written to the output data file.

6.2.5.3.2.11 3-Dimensional Correlation Coefficients (12, 13 and 23)

For each 3-dimensional variable set, a set of correlation coefficients is calculated by the analysis. The variables within a set are numbered in their order of selection (from left to right in the screen display) and the correlation coefficients between variables 1 and 2, variables 1 and 3 and variables 2 and 3 are calculated. The correlation coefficient is calculated for a variable pair (for example: X and Y) from the following equations:

$$\rho = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

where

$$\sigma_{xy} = \frac{\sum_{i=1}^{n} \left(x_i \, y_i \right) - n\overline{x}\overline{y}}{n-1}$$

and σ_x and σ_y are the standard deviations of the individual variables. If the incorporation of TLE data has been selected, then the TLE data is added to these statistical parameters.

6.2.6 SWEEP

The SWEEP utility provides the user with a method of reducing the data from a CADAC multi-run/Monte Carlo Sweep simulation to a single set of statistics for the selected range and angle variables. The user may generate a statistical file compatible with the CONTOUR application based on the following options: Mean, Variance, Standard Deviation, Mean Deviation, Root Sum Squared of Mean and Standard Deviation, Root Mean Squared and Bivariate Statistical CEP (including the CEP Percent). The user may also generate specific statistical files compatible with the ELLIPSE application or the CONTOUR application. The user may also perform normalization (Appendix C) of impact points by removing gross errors and direct hits before generating other statistical output.

6.2.6.1 SWEEP Execution

SWEEP is a Windows application that can be added to the DIGITAL Visual FORTRAN TOOL option. In addition, SWEEP execution can be initiated by using the RUN option in Windows or by double-clicking on the SWEEP.EXE file in Windows EXPLORER. For more information on SWEEP execution refer to the on-line help in the SWEEP menu.

6.2.6.2 SWEEP Input and Output Files

The input file (default name IMPACT.ASC) contains multiple data sets associated with each aimpoint. If the Target Location Errors (TLE) option is selected, an optional input file, TLE.ASC, containing the Target Location Errors may be utilized. The SWEEP output file is dependent on whether the BOUNDARIES AND CONTOURS option or the ERROR ELLIPSES option is selected. If the BOUNDARIES AND CONTOURS option is selected, the generated statistics are placed in a file compatible with the CONTOUR program; the output filename is dependent on the selected statistic methodology prefixed with the word SWEEP. If the ERROR ELLIPSES option is selected, the generated statistics are place in a file compatible with the ELLIPSE program; the output filename is SASP.ASC. Several working files are generated by the SWEEP program which the user will never access. However, these files are vital to the proper execution of SWEEP and must not be deleted unless deleted or over written by the SWEEP program itself. A list of these working files is as follows:

CONTOUR.LOG	ELLIPSE.LOG
MCAP.LOG	QPRINT.LOG
SWEEPF1.LOG	SWEEPF2.LOG
DA.ASC	DAIMP.ASC
DH.ASC	GE.ASC
NM.ASC	
SWEEP.MEA	SWEEP.VAR
SWEEP.STD	SWEEP.MDV
SWEEP.RMS	SWEEP.RSS
SWEEP.BIV	
SWEEP.WDR	CONTOUR.WDR
ELLIPSE.WDR	
MCAP.ASC	QPRNTIMP.ASC
QPRNTTT.ASC	QPRNTTTM.ASC

6.2.6.2.1 SWEEP Input Data File

The input data file has the same format as the CADAC-produced IMPACT output file, IMPACT.ASC. This SWEEP file must be generated by a CADAC multi-run or Monte Carlo simulation or deterministic simulation. The SWEEP file data is aimpoint dependent, not time dependent; each aimpoint is usually defined by polar coordinates. When produced under a multi-run/Monte Carlo simulation condition, the file contains a set of data points associated with each aimpoint. The IMPACT.ASC file must contain the aimpoint position (range and angle in either degrees or radians). The SWEEP data on the file must conform to the following restrictions:

1) The data must be arranged on the file in ascending order by angle and range location.
2) The delta between the angular coordinates of the aimpoints must be constant.

6.2.6.2.2 TLE.ASC Format and Contents

The TLE.ASC input file is an ASCII file containing a title (maximum 80 characters) in the first record and target location errors in the remaining records. For more information on the TLE.ASC input file refer to Section 6.2.5.2.2.

6.2.6.2.3 SWEEP Output Files

The SWEEP output files have the same format as the SWEEP files regardless of the selected program options; a SWEEP output file is generated each time the continue command is selected. The contents of the SWEEP output file are dependent on the selected program options.

6.2.6.2.3.1 Boundaries and Contours Statistical Calculation Option Output File Components

When the Boundaries and Contours option is selected, the SWEEP output file contains the statistic, as selected from the interactive screen, associated with each aimpoint from the input file. The base name of the output file is "SWEEP". The filename extension is dependent on the statistic selected for calculation. Table 11 displays the statistic and the associated filename extension. For example, if the mean statistic is selected, then the SWEEP output filename containing the results of the analysis is "SWEEP.MEA". This file will be used by the CONTOUR program.

Table 11. Statistic Output Filename Extension.

Statistic Name	Output File Extension
Mean	.MEA
Variance	.VAR
Standard Deviation	.STD
Mean Deviation	.MDV
Root Sum Squared of Mean and Standard Deviation	.RSS
Root Sum Squared	.RMS
Bivariate Circular Error Probability	.BIV

6.2.6.2.3.2 Error Ellipses Option Output File Contents

When the Error Ellipses option is selected, the output file contains eight values for each aimpoint: the aimpoint range coordinate, the aimpoint angle coordinate, the X variable mean, the X variable standard deviation, the Y variable mean, the Y variable standard deviation, the correlation coefficient between the X and Y data and the number of points.

This data is placed in the file SASP.ASC in the default directory. The acronyms on the SASP.ASC file for the aimpoint range and angle coordinate are the same as the acronyms from the input file. The acronyms for the generated statistics are based on the original SWEEP input file acronyms and the type of statistic performed. A character code corresponding to the statistical parameter is then appended to the acronym basis: "M" for the mean statistic and "S" for the standard deviation statistic. The acronyms for the correlation coefficient and number of points are "RHOXY" and “NPTS”, respectively. This file will be used by the ELLIPSE program.

6.2.6.3 SWEEP Methodology

This section discusses the methodology implemented in the SWEEP program. When the user executes SWEEP, the menu parameters are checked for validity. If invalid parameter data is found, error messages are displayed and the user is returned to the menu screen. If the data is valid, then the selected program option is performed. Once SWEEP has performed the necessary statistical analysis, the user is automatically taken to the next phase of the process whether that be CONTOUR or ELLIPSE or other utility program required to execute based on the options the user has selected in the SWEEP menus. When the process is complete, the user is automatically exited from all programs.

6.2.6.3.1 Statistical Calculations

As the statistical calculations must be performed for each of the up-to-70 variables per data record, the data record is treated as a vector with the calculations being performed for each element. Each statistic is calculated using the same basic method. This method reads the input data file and accumulates all of the data vectors until a new aimpoint is detected. Then the following tasks are performed:

- Calculate the sum of the vectors
- Calculate the sum of the squared vector
- Calculate the number of vectors associated with the aimpoint
- Create a temporary file containing the vector data for the current aimpoint

The selected statistical calculations are performed for the current aimpoint data. The variance, standard deviation, root mean squared, mean deviate, root sum squared and bivariate circular error probability statistic calculations modify the range and angle variables. Since the aimpoint coordinate is required by the SWEEP Windows application, the range and angle variables are reset to the current aimpoint coordinate for these statistics. The resulting vector is then written to the output file. The statistic is calculated for each aimpoint until the end of the input file is detected. The calculations specific to each statistic type are discussed in the remainder of this section.

6.2.6.3.1.1 Mean Calculation

The mean statistic module calculates the sum of the vectors and the number of vectors associated with the aimpoint. The mean is calculated by dividing the sum vector by the number of vectors in the data set. The resulting mean vector is then written to the output file. This calculation and method is performed for all aimpoints on the input file.

6.2.6.3.1.2 Variance Calculation

The variance statistic module calculates the sum of the vectors, the sum of the squared vectors and the number of vectors associated with the aimpoint. The mean vector is first calculated by dividing the sum vector by the number of vectors in the data set. The resulting mean vector, number of data points and the sum of the squared vector is passed to a module which calculates the variance. The variance is calculated from the equation:

$$\sigma^2 = \frac{\Sigma_{i=1}^{n}(x_i^2) - n\bar{x}}{n - 1}$$

The aimpoint coordinate variables are reset and the resulting variance vector is then written to the output file. The calculations are repeated for all aimpoints on the input file.

6.2.6.3.1.3 Standard Deviation Calculation

The standard deviation statistic module calculates the sum of the vectors, the sum of the squared vectors and the number of vectors associated with the aimpoint. The mean vector is first calculated by dividing the sum vector by the number of vectors in the data set. The resulting mean vector, number of data points and the sum of the squared vector is used to calculate the variance as discussed in Section 6.2.6.3.1.1.2. The square root of each variance vector is then taken producing the standard deviation. Sometimes a negative number occurs under the radical in these calculations; typically this occurs due to the computer representation of the data values. If the element value under the radical sign is negative, the module sets the element value to -9999.99 to signify the square of a negative value has been attempted. The module then checks for elements having the value -9999.99 and resets them to the value 0.0. The aimpoint coordinates are reset and the resulting standard deviation vector is then written to the output file. These calculations are repeated for all aimpoints on the input file.

6.2.6.3.1.4 Mean Deviation Calculation

The mean deviation statistic is calculated using the following equation:

$$\text{MEAN DEVIATE} = \frac{\Sigma_{i=1}^{n}|x_i - \bar{x}|}{n}$$

The mean deviate calculation module calculates the sum of the vectors, the number of vectors and creates a temporary file of the vectors associated with the aimpoint. The mean data vector is calculated by dividing the sum vector by the number of vectors in the data set. The temporary file is rewound and the mean vector is subtracted from each data vector on the file; the absolute value of the resultant vector is computed and added to a summation vector. Once all the data points associated with the aimpoint have been processed, the resulting summation vector is divided by the number of vectors in the data set. The aimpoint coordinates are reset and the resulting mean deviate vector is then written to the output file. These calculations are repeated for all aimpoints on the input file.

6.2.6.3.1.5 Root-Sum-Squared (RSS) of Mean and Standard Deviation Calculation

The Root-Sum-Squared (RSS) of the mean and standard deviation statistic is calculated using the following equation:

$$\text{RSS} = \sqrt{\bar{x}^2 + \sigma_x^2}$$

The RSS calculation module calculates the sum of the vectors, sum of the squared vectors and the number of vectors associated with the aimpoint. The mean vector is calculated by dividing the sum vector by the number of vectors in the data set. The variance vector is calculated from the mean vector, sum of the squared vector and the number of vectors as discussed in Section 6.2.6.3.1.1.2. Then the mean vector is squared and added to the variance vector and the square root of the result is taken. The aimpoint coordinates are reset and the resulting RSS vector is then written to the output file. These calculations are repeated for all aimpoints on the input file.

6.2.6.3.1.6 Root Mean Squared (RMS) Calculation

The Root-Mean-Squared (RMS) statistic is calculated using the following equation:

$$\text{RMS} = \sqrt{\frac{\Sigma_{i=1}^{n}(x_i^2)}{n}}$$

The RMS calculation module calculates the sum of the squared vectors and the number of vectors associated with the aimpoint. The sum squared vector is divided by the number of vectors in the data set. The square root of this vector is then calculated. The aimpoint coordinates are reset and the resulting RMS vector is then written to the output file. These calculations are repeated for all aimpoints on the input file.

6.2.6.3.1.7 Bivariate Circular Error Probability (CEP) Calculation

The Bivariate Circular Error Probability (CEP) is calculated for each variable using the same process as the MCAP CEP calculation. A discription of this calculation is found starting in Section 6.2.5.3.2.8.

6.2.6.3.2 ELLIPSE Program Input File Creation Methodology

When the ERROR ELLIPSES option is selected in the SWEEP program and the continue command is issued with valid parameter data, the statistical data required by program ELLIPSE is calculated for the selected X and Y variable at each aimpoint and written to the output file along with the aimpoint range and angle coordinate.

The X and Y data are handled in these calculations as a two element vector and many of the modules performing vector mathematics from the statistical analysis option are utilized. If the Target Location Errors (TLE) are to be added to the ELLIPSE input data, then the TLE matrix is read from the designated TLE input file. The statistical calculations are performed in the following manner. First the following calculations are performed for the data vector associated with the current aimpoint coordinate:

- The sum of the vectors
- The sum of the squared vector
- The sum of the product of X and Y
- The counting of the number of vectors associated with the aimpoint

The current aimpoint range and angle coordinate is loaded into the output vector. The mean vector is then calculated by dividing the sum vector by the number of vectors in the data set.

The mean vector, the sum of the squared vector and the number of vectors in the data set are used to calculate the variance as discussed in Section 6.2.6.3.1.1.2. The square root of each element of the variance vector is taken, producing the standard deviation. Sometimes a negative number occurs under the radical in these calculations; typically this occurs due to the computer representation of the data values. If the element under the radical sign is negative, the module sets the element value to -9999.99 to signify the square of a negative value has been attempted. The module then checks for elements having the value -9999.99 and resets them to the value 0.0.

If the number of vectors in the data set is greater than 1, then the covariance between the X and Y variables is computed from the following equation:

$$\sigma_{xy} = \frac{S_{i=1}^{n}(X_i \ Y_i) - n\overline{X}\overline{Y}}{n - 1}$$

If the standard deviations of X and Y are equal, then the correlation coefficient is set to 0.0; otherwise the correlation coefficient is calculated from the following equation:

$$\rho_{xy} = \frac{\sigma_{xy}}{\sigma_x \ \sigma_y}$$

If the Target Location Errors (TLE) are to be added to the ELLIPSE statistics, then the standard deviations and correlation coefficient are loaded into a 3x3 matrix and added to the TLE matrix:

$$\begin{bmatrix} \sigma_x^2 & \rho_{xy} & 0 \\ \rho_{yx} & \sigma_y^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}_{SASP} + \begin{bmatrix} \sigma_x^2 & \rho_{xy} & \rho_{xz} \\ \rho_{yx} & \sigma_y^2 & \rho_{yz} \\ \rho_{zx} & \rho_{zy} & \sigma_z^2 \end{bmatrix}_{TLE}$$

The X variable mean and sigma, Y variable mean and sigma and the correlation coefficient are then loaded into the output vector and the vector is written to the output file named SASP.ASC in the default directory. These calculations are repeated for all aimpoints on the input file.

6.2.6.4 ELLIPSE

The ELLIPSE application provides a graphical file of a special type input created from a CADAC multi-run/Monte Carlo Sweep simulation. This input file must contain the aimpoint defined by the Sweep methodology and the standard deviation and correlation coefficient statistics. The ELLIPSE application file when viewed by WinDRAW displays icons at each aimpoint and can show the relationship between the target size and the containment ellipse.

6.2.6.4.1 ELLIPSE Program Execution

ELLIPSE is a Windows application that can be added to the DIGITAL Visual FORTRAN TOOL option. In addition, ELLIPSE execution can be initiated using the RUN command in Windows or by double-clicking on the ELLIPSE executable file in Windows EXPLORER. The preferred method for executing the ELLIPSE program is to allow the SWEEP program to run the application. For more information on the execution of the ELLIPSE application, refer to the on-line help in the ELLIPSE menus.

6.2.6.4.2 ELLIPSE Input and Output Files

Three input files and three output files are associated with the ELLIPSE application. The input file (default name SASP.ASC) contains the statistical data which can influence the icon angle and size. The SASP.LOG file contains the current parameter values; this file is accessed by the OPEN .LOG file command under the file menu option or loaded automatically when called from SWEEP. (For more information on the log file, refer to the ELLIPSE on-line help.) The ICON.ASC file contains information about icons that can be displayed with the ellipse plot. The SAVE or SAVE AS command under the FILE menu option produces a file which is compatible with the WinDRAW application. The ELLIPSE_ERR.LIS is produced by ELLIPSE and contains any error messages which occur during the menu and graphics interactive sessions.

6.2.6.4.2.1 SASP.ASC Input File

The input data file has the same format as the CADAC-produced STAT.ASC or IMPACT.ASC files. The ELLIPSE application requires data based on a CADAC multi-run/Monte Carlo SWEEP output file. The SASP file data is aimpoint dependent, not time dependent; each aimpoint is defined by polar coordinates. When produced under a multi-run/Monte Carlo simulation condition, the SASP file contains data associated with each aimpoint. The SASP.ASC file must contain the aimpoint position (range and angle) and for each aimpoint, the sigmas and correlation coefficient of an X,Y data pair as well as the number of points. The ELLIPSE application reads the SASP.ASC file for data in the following order: range, angle, X mean, X standard deviation, Y mean, Y standard deviation, the correlation coefficient between X and Y and the number of points. If the SASP.ASC file does not have these data in the correct order, the user may select the appropriate acronym for the necessary data.

6.2.6.4.2.2 ICON.ASC Input File

The ICON.ASC input file contains information for the icon to be drawn in WinDRAW. The data on an ICON.ASC file includes the icon name, size, number of points that make up the icon and the points themselves. Figure 23 illustrates a sample ICON.ASC file.

```
1 TOWER          18  ← ICON NUMBER, ICON NAME, NUMBER OF POINTS THAT MAKE UP ICON
 50.00 feet          ← ICON SIZE, SIZE UNITS
  .2     .0      0   ← ICON POINTS WITH PEN DROP FLAG: 0 = Don't Drop, 1 = Drop Point
  .5     .8      1
  .8     .0      1
  .3125  .3      1
  .6125  .5      1
  .3875  .5      1
  .6875  .3      1
  .2     .0      1
  .4     .7      0
  .3     .7      1
  .4     .8      0
  .3     .9      1
  .5     .9      0
  .5    1.0      1
  .6     .8      0
  .7     .9      1
  .6     .7      0
  .7     .7      1
 2 BOAT          9   ← ICON NUMBER, ICON NAME, NUMBER OF POINTS THAT MAKE UP ICON
 50.00 feet          ← ICON SIZE, SIZE UNITS
  .4     .4      0   ← ICON POINTS WITH PEN DROP FLAG: 0 = Don't Drop, 1 = Drop Point
  .8     .4      1
  .4    1.0      1
  .4     .3      1
  .1     .3      1
  .3     .0      1
  .7     .0      1
  .9     .3      1
  .4     .3      1
 3 ARROW         6   ← ICON NUMBER, ICON NAME, NUMBER OF POINTS THAT MAKE UP ICON
 50.00 feet          ← ICON SIZE, SIZE UNITS
  .5     .0      0   ← ICON POINTS WITH PEN DROP FLAG: 0 = Don't Drop, 1 = Drop Point
  .5     .6      1
  .2     .6      1
  .5    1.0      1
  .8     .6      1
  .5     .6      1
```

Figure 23. Sample ICON.ASC Input File.

6.2.6.4.2.3 WinDRAW.WDR Output File Format

The SWEEP.WDR output file has the same format as the WinDRAW WINDRAW.WDR files. The WDR file generated by ELLIPSE contains several segment types. The WDR file is automatically saved when the user selects the continue menu command.

6.2.6.4.3 ELLIPSE Display Generation Methodology

This section discusses the data display generated by ELLIPSE when the user enters the continue command. The display is generated from the icons defined in the menu screen and the data from the SASP.ASC input file.

6.2.6.4.3.1 Graphics Display Generation

When the continue command is selected in the ELLIPSE program, the screen selections are checked for validity. If errors are found, the error messages are displayed and the user is returned to the menu screen so that valid data may be entered. If the input data is valid, ELLIPSE preprocesses the SASP.ASC input file data and produces a working file for generating the graphics display. The working file contains the coordinates of the aimpoint converted to the Cartesian coordinate system (X_a, Y_a), the X sigma (σ_x), Y sigma (σ_y), the correlation coefficient between X and Y (ρ), the X mean (μ_x) and the Y mean (μ_y). The maximum and minimum X and Y aimpoint coordinates are also determined and used as a basis in calculating the graphics display field of view (FOV). These maximum and minimum values are multiplied by four times the maximum icon magnification factor. These results are then passed into a module that adjusts the X and Y delta ratio to match the screen ratio, providing a square display. First the axis and labels are placed on the data file, icons are then added to the file if selected by the user and the data set statistics corresponding to an aimpoint are read from the working file.

If a data angle is to be applied to the icon, then the data set statistics are used to calculate the angle of the containment ellipse using the following equation:

$$\theta = \frac{1}{2}\mathrm{TAN}^{-1}\left[\frac{2\rho\sigma_x\sigma_y}{\sigma_x^2 - \sigma_y^2}\right]$$

This is the angle of rotation referenced to the X axis of the icon with counter-clockwise the positive rotation direction. If the data angle is not to be applied to the icon, then the icon rotation angle is set to 0.0.

If the data magnifications are to be applied to the icon, then the lengths of the containment ellipse major and minor axes are calculated in the following manner. The derivation of ellipse radius (R) provides the equation for R in polar coordinates:

$$R = \sqrt{\frac{C^2\ (1 - \rho^2)}{1 - \rho\sin(2\theta)}}\ 1 \qquad (3)$$

where

$$C^2 = -2\ln\left(1 - \frac{P}{100}\right)$$

and P is the user-input ellipse containment percentage. The maximum R value is obtained by evaluating Equation (3) with θ = 0.0. Transforming back into Cartesian coordinates and solving for the major axis length gives the result:

$$A = 2R\sigma_x$$

The solution for the length of the minor axis produces the result:

$$B = 2R\sigma_y$$

The lengths of the axes are the data magnification factors applied to the icon as selected by the user. Since the icons are based on a square with the limits [0,1], a circular icon can be transformed into an ellipse by multiplying (magnifying) in the X direction by the ellipse X axis length and in the Y direction by the ellipse Y axis length. Magnifying the unit square icon by the user-entered magnification does not affect this transformation of the circle to an ellipse. If the data magnifications are not to be applied to the icon, then these factors are set to 1.0. These data X and Y magnification factors (FX and FY, respectively) are currently applied to the icon.

The icon center (X_i, Y_i) is set to the aimpoint location (X_a, Y_a). This process is repeated for each aimpoint on the file and with each defined icon. After the data are placed on the file the aimpoint and legend information are added. The user may view the resulting graphics file with the program WinDRAW.

6.2.6.5 CONTOUR

The CONTOUR utility provides the user with a method to graphically contour the SWEEP variables contained within a CADAC generated IMPACT.ASC file. This contour may be performed using the SPLINE feature of the Graphics screen or automatically done by using the AUTO menu option in the Graphics screen. CONTOUR is a Windows application that can be added to the DIGITAL Visual FORTRAN TOOL option. In addition, CONTOUR execution can be initiated by using the RUN option in Windows or by double-clicking on the CONTOUR exectutable file in Windows EXPLORER. The preferred method to execute the CONTOUR program is to allow SWEEP to control its execution.

The CONTOUR graphics program allows the user to interactively build contours on SWEEP data. These contours may be saved and recalled by CONTOUR or viewed and enhanced by WinDRAW. For more information on CONTOUR execution, refer to the on-line help in the SWEEP menu.

6.2.7 WinDRAW

The WinDRAW application was created to support graphical annotation of data (saved to the default filename WINDRAW.WDR) and to provide interactive drawing capabilities. The WinDRAW application is a Windows 95/98/NT application that can be executed from another program as an OLE server, run from the WinDRAW icon on the Windows start menu or if associated with the WDR file extension it can be started from within Explorer by double clicking on a file of the WDR type. In addition, WinDRAW execution is performed by using the Continue option button from the SWEEP program.

The WDR file type is an ASCII type text file containing the data necessary to create or reproduce a graphics image. The file consists of a title record, a header record containing xmax/xmin, ymax/ymin data for the screen size and four header flags (Table 12). This data is followed by two text header records and a series of data records (Table 13). Each data record or segment contains information required to reproduce the draw object, this data includes location, color (Table 14), fill patterns (Table 15) , marker type (Table 16), line widths (Table 17) , line type (Table 18) and rotation angle.

The image created with WinDRAW can be copied to the clipboard, as a bitmap file or exported to several file types. For more information on the WinDRAW application, refer to the WinDRAW on-line help.

Table 12. WinDRAW Header Record.

WinDRAW Header			
Map	Grid	Units	Density
0 - grid 1 - map	0 - border only 1 - tic tac toe 2 - x,y axis @ center 3 - x,y axis @ lower left 4 - x,y axis @ lower center 5 - user grids	1 - feet 2 - nm 3 - m 4 - km	user defined

Table 13. WDR File Contents.

segment	type	color	# of points	fill	xref	yref	p1	p2	p3	p4	next records
	1	2	3	4	5	6	7	8	9	10	next rec.
Text	7	IC	--	--	Xpt	Ypt	Size	Font	# of char	Rot	Text
Line	8	C	2	--	Xpt	Ypt	Length	LW	LT	Rot	Points
Fan	9	C	3	IF	Xpt	Ypt	Length	LW	LT	Rot	Points
Circle	10	C	130	IF	Xpt	Ypt	Radius	LW	LT	Rot	Points
Least Square	15	C	VAR	IF	Xpt	Ypt	---	LW	LT	Rot	Points
Spline	16	C	VAR	IF	Xpt	Ypt	--	LW	LT	Rot	Points
Point	17	C	VAR	IF	Xpt	Ypt	M		--	Rot	Points
Box	18	C	5	IF	Xpt	Ypt	Length	LW	LT	Rot	Points
Ellipse	19	C	130	IF	Xpt	Ypt	Major	LW	LT	Rot	Points
Create	20	C	VAR	IF	Xpt	Ypt	--	LW	LT	Rot	Points
Comp	21		VAR	IF	Xpt	Ypt				Rot	Segments
Icon	25	C	Icon #	---	Xpt	Ypt	Xmag	Ymag	---	Rot	File Name
Table	30	Row	Col								“title”, “headers”
Scale	31	C	#tics	---	Xpt	Ypt	Delta	--	--	--	“title”
Aim Drop	32	C	VAR	---	Xpt	Ypt	Maj	Min	AP#	Rot	“Ic name”, Ic num, Ic rot, data
Aimpoint	35	Row	Col		Xpt	Ypt	Impact	Traj	SF		Stats
Globe Arc	33	C	2	IF	Xpt	Ypt	---	LW	LT	Arc Type	Points
Grid Scale	40	C	---	---	Xpt	Ypt	X-axis	Y-axis	Delta	Rot	---
	1	2	3	4	5	6	7	8	9	10	next rec.

Table 14. WinDRAW Colors(C).

Index	Color	Index	Color
0	Black	13	Pink
1	White	14	Dark Gray
2	Red	15	Lite Gray
3	Green	16	Lite Cyan
4	Blue	17	Olive
5	Cyan	18	Sky Blue
6	Magenta	19	Brown
7	Yellow	20	Navy Blue
8	Orange	21	Royal Blue
9	Dark Green	22	Rust
10	Lite Green	23	Dark Red
11	Lite Blue	24	Lime Green
12	Purple		

Table 15. WinDRAW Fill Patterns(IF).

Index	Pattern	Index	Pattern
0	none	4	cross horizontal and vertical
1	solid	5	diagonal up
2	horizontal	6	diagonal down
3	vertical	7	cross diagonal

Table 16. WinDRAW Marker Types(M).

Index	Marker	Index	Marker
0	point	5	boxed x
1	plus	6	box
2	asterisk	7	diamond
3	o	8	bow tie
4	x	9	triangle

Table 17. WinDRAW Line Width(LW).

Index	Style	Index	Style
0	thinnest	6	
1		7	
2		8	
3		9	
4		10	thickest
5			

Table 18. WinDRAW Line Style(LT).

Index	Style
0	Solid
1	Dash
2	Dot
3	dash dot
4	dash dot dot
5	none

APPENDIX A

CADIN.ASC CARD DEFINITIONS

A.1 CADIN.ASC CARD DEFINITIONS

The statements the user places in the INPUT.ASC file which are processed by the CADIN program are assigned a card type. Each card type performs a specific function. Table A-1 shows the card types that are valid in program CADAC and the corresponding definition.

Table A-1. Lead Card Type Definitions.

Card Type	Card Definition
1	Subroutine Selection
2	Module Selection
3	Variable Initialization
4	User Comments
5	Multi-run Selection
6	Finish Simulation
8	Atmospheric Input
9	Header Input
10	Stage Definition
11	Function Definition
12	End of Group
13	End of CADIN.ASC
16	End of Stage Criteria
19	Sweep Outer Variable Definition
20	Sweep Inner Variable Definition
21	Sweep Criteria Definition
90	Save Simulation State

All of the cards follow the same basic format with the exception of the type 8, 9 and 10 cards. The basic card format is shown in Table A-2. The definition of each variable is dependent on the card type with the exception of the MOD variable. The following sections discuss the card type and the variable definitions particular to that card. Deviations from this basic format are noted in the descriptions of the cards. The variable names associated with the card data in Table A-2 are used in the following sections to refer to the data entered at that particular card location.

Table A-2. Basic Card Format.

Card Column	Format	Column Name	Card Used By
1-2	I2	ICARD	All Cards
3-8	A6	CHAR1	1,2,3,4,8,11,19,20,21
9-14	A6	CHAR2	4,8,11
15-20	A6	CHAR3	3,4,8
21-25	I5	LOC	1,2,3,5,9,10,11,19,20,21
26-30	I5	MODE	3,5,11,19,20,21
31-45	E5.9	V1	3,11,19,20,21
46-60	E5.9	V2	3,11,19,20,21
61-62	I2	MOD	3,11

The MOD variable is used when a card is included in a group card set. This variable then indicates the stage in the primary trajectory where the card is to be placed. Valid values for this variable are a blank or zero (indicating the initial stage) or 1 through 20.

A.1.1 Type 1 Cards - Subroutine Selection

The type 1 cards select the subroutines to be called after each integration step. Typically, these cards are located at the top of the input deck after the title and the optional multi-run card. Currently, only two subroutines are available, OUPT3 and STGE3. These subroutines perform output and stage testing. These subroutines must be selected for CADAC to execute properly. The subroutines and their associated locations are given in Table A-3.

Table A-3. Subroutine Location and Function.

Subroutine Number	Location	Function
OUPT3	2	Output the print and plot data.
STGE3	3	Test stage criteria for stop/pause in simulation.

Thus two type 1 cards are required in each CADIN.ASC file. They must appear in the lead cards for the stage 0 of the primary trajectory. The ordering of the cards dictates the order the modules are executed after the integration step. The definitions of the variables in the type 1 card are shown in Table A-4.

Table A-4. Type 1 Card Definitions.

Column Number	Variable Name	Definition
1-2	ICARD	The card number. Must be " 1".
3-8	CHAR1	Subroutine name. Should be either "OUPT3" or "STGE3".
21-25	LOC	Subroutine number. Must be either 2 or 3.

Program CADIN automatically includes type 1 cards in the CADIN.ASC file.

A.1.2 Type 2 Cards - Module Selection

The type 2 cards are used to select the modules that are to be executed during the specific CADAC trajectory simulation. The definitions of the valid entries for type 2 cards are shown in Table A-5. Only one module is selected per card, with multiple type 2 cards defining the list of modules to be executed. If a module outputs variables required as input by another module, then the order of execution becomes important. The modules are executed in the same order as selected by type 2 cards; therefore it is very important to sequence the cards properly. The type 2 card for the module producing the output must come before the type 2 card for the module requiring the data as input. Table A-6 provides information on the module names, associated numbers, reserved common locations and a typical description of the module.

Program CADIN places a type 2 card in the CADIN.ASC file for each module in the module block.

Table A-5. Type 2 Card Definitions.

Column Number	Variable Name	Definition
1-2	ICARD	The card number. Must be " 2".
3-8	CHAR1	Module Name.
21-25	LOC	Module Number.

Table A-6. Module Names and Associated Common Locations.

MODULE NAME	ASSOCIATED NUMBER	MODULE TYPE	Reserved Common Locations	COMMENTS
G1	22	Target	100-199	2 Points
G2	23	Air Data	200-299	Standard/Table
G3	24	Kinematic	300-399	
G4	25	Terminal	1750-1799	Not Module
G5	26			
S1	28	Seeker	400-499	TV, IIR, Radar
S2	29	Sensor	500-599	GPS
S3	30	NAV Filter	600-699	Kalman
S4	31	INS	700-799	Strap Down
S5	32			
C1	7	Guidance	800-899	Pro.-NAV
C2	8	Autopilot	900-999	Acceleration, Angles
C3	9	TVC	1000-1099	
C4	10	Acuator	1100-1199	
A1	2	Aero Coefficients	1200-1299	Cruciform, Planar
A2	3	Propulsion	1300-1399	Rocket, Turbojet, Ramjet
A3	4	Forces & Moments	1400-1499	
A4	5	Growth	1500-1599	
D1	17	Translational	1600-1699	
D2	18	Rotational	1700-1749	
D3	19	Growth	1825-1899	
D4	20	Debug	1900-1999	

A.1.3 Type 3 Cards - Variable Initialization

The type 3 cards allow values to be assigned to variables in the C array. Only one variable may be assigned a value by each card; multiple type 3 cards may be entered on the input deck. The type 3 cards have three different definitions. The card may be used to define real variables, integer variables or to initialize a real variable to a function value. Each definition is discussed in detail in the following sections.

The card format for the type 3 cards is the basic format discussed in Section 3. However, definitions of the variables is dependent on the usage of the type 3 card. It is important to note that the variable being defined is identified by the numeric value in columns 21-25; the description in columns 3-8 is available for the user's benefit only.

CADIN converts the assignment statements in a INPUT.ASC file to type 3 cards.

A.1.3.1 Initializing Real Variables

The type 3 card can be used to assign values to real variables equivalenced to C array elements. Table A-7 shows the definition of the card entries for performing this task.

Table A-7. Type 3 Definitions for Initializing Real Variables.

Column Number	Variable Name	Definition
1-2	ICARD	The card number. Must be " 3".
3-8	CHAR1	The variable name.
21-25	LOC	The variable location.
31-45	V1	The value to be assigned to the variable.
61-62	MOD	The stage of the card.

An example of a type 3 card initializing a real variable in stage 0 of the primary trajectory is:

```
                          Card Column
0        1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901
03SBEL2              1060     10000.0
```

This example sets the variable C(1060), SBEL2, to the value 10,000.0.

A.1.3.2 Initializing Integer Variables

The type 3 card may also be used to assign values to integer variables equivalenced to a C array element. The value is integerized then assigned to an IC array at the designated element. The IC array is equivalenced to the C array. This array is necessary to provide compatibility between the method of storing real and integer numbers within the computer memory. Table A-8 shows the definitions of the data locations when the type 3 card is used to define integers.

Table A-8. Type 3 Definitions for Initializing Real Variables.

Column Number	Variable Name	Definition
1-2	ICARD	The card number. Must be " 3".
3-8	CHAR1	The variable name.
21-25	LOC	The variable location.
26-30	MODE	Flag indicating the integer assignment.
31-45	V1	The value to be assigned to the variable.
61-62	MOD	The stage assignment.

The following example shows a card of type 3 used to set the MINIT flag to 111:

```
                          Card Column
0        1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901
03MINIT              1600    1111.0
```

A.1.3.3 Initializing To A Function Value

The type 3 card can also be used to initialize a variable to the value of a predefined function. Up to six functions are available for use with the type 3 card. These functions are the Gauss, Uniform, Exponential, Rayleigh, Sign and Equals functions. The functions are selected by entering a special keyword associated with each function, left justified, in card columns 5 through 20 of the type 3 card (variable CHAR3). The format for each type 3 function card is discussed in the following sections. It should be noted that the variables are only given the function value at the time the card is processed: this card does NOT provide a new value at each integration interval.

A.1.3.3.1 Type 3 - GAUSS Function

The selection of the GAUSS function provides a value selected from the Gaussian (Normal) distribution having a user selected mean and standard deviation (sigma). The Gaussian distribution is selected by entering the keyword "GAUSS" as CHAR3 in the card. The definitions of the valid card variables for this function are shown in Table A-9.

Table A-9. Type 3 Definitions for the Gaussian Function.

Column Number	Variable Name	Definition
1-2	ICARD	The card number. Must be " 3".
3-8	CHAR1	The variable name.
15-20	CHAR3	The function name: "GAUSS".
21-25	LOC	The variable location.
31-45	V1	The mean of the Gaussian distribution.
46-60	V2	The sigma of the Gaussian distribution.
61-62	MOD	The stage assignment.

An example of using the Gaussian function in the type 3 card is:

```
                              Card Column
0         1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901
03HLE          GAUSS  1616      5.0            10.0
```

In this example, the variable HLE, element 1616 in the C array, will be given a Gaussian distributed random value having a mean of 5.0 and a standard deviation of 10.0.

A.1.3.3.2 Type 3 - UNIF Function

The selection of the uniform function provides a value selected from the uniform distribution having a user selected mean and width. The uniform distribution is selected by entering the keyword "UNIF" as CHAR3 in the card. The definitions of the valid card variables for this function are shown in Table A-10.

Table A-10. Type 3 Definitions for the Uniform Function.

Column Number	Variable Name	Definition
1-2	ICARD	The card number. Must be " 3".
3-8	CHAR1	The variable name.
15-20	CHAR3	The function name: "UNIF".
21-25	LOC	The variable location.
31-45	V1	The mean of the uniform distribution.
46-60	V2	The (width/2) of the uniform distribution.
61-62	MOD	The stage assignment.

An example of using the uniform function in the type 3 card is:

```
                                   Card Column
0        1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901
03HLE         UNIF   1616     2.0            5.0
```

In this example, the variable HLE, element 1616 in the C array, will be given a uniform distributed random value having a mean of 2.0 and a width of 10.0.

A.1.3.3.3 Type 3 - EXPO Function

The selection of the exponential function provides a value selected from the exponential distribution having the user selected distribution parameter. The exponential distribution is selected by entering the keyword "EXPO" as CHAR3. The definitions of the valid card variables for this function are shown in Table A-11.

Table A-11. Type 3 Definitions for the Exponential Function.

Column Number	Variable Name	Definition
1-2	ICARD	The card number. Must be " 3".
3-8	CHAR1	The variable name.
15-20	CHAR3	The function name: "EXPO".
21-25	LOC	The variable location.
31-45	V1	The mean of the exponential function.
61-62	MOD	The stage assignment.

An example of a type 3 input card utilizing the exponential function is:

```
                                   Card Column
0        1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901
03HLE         EXPO   1616     5.0
```

The variable HLE, location 1616 in the C array, will be given an random value from the exponential distribution having a mean of 5.0.

A.1.3.3.4 Type 3 - RAYLEI Function

The selection of the Rayleigh distribution provides a value selected from the Rayleigh distribution having the user input distribution parameter. The Rayleigh distribution is selected by entering the keyword "RAYLEI" as CHAR3. The definition of the valid card values for this function are given in Table A-12.

Table A-12. Type 3 Definitions for the Rayleigh Function.

Column Number	Variable Name	Definition
1-2	ICARD	The card number. Must be " 3".
3-8	CHAR1	The variable name.
15-20	CHAR3	The function name: "RAYLEI".
21-25	LOC	The variable location.
31-45	V1	The function parameter value.
61-62	MOD	The stage assignment.

An example of a type 3 input card utilizing the Rayleigh function is:

```
                         Card Column
0        1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901
03HLE         RAYLEI 1616     5.0
```

The variable HLE, location 1616 in the C array, will be given the value determined by the RAYLEIGH function with a user input mode of 5.0.

A.1.3.3.5 Type 3 - SIGN Function

The SIGN function is selected by entering the keyword "SIGN" as CHAR3. When this function is selected, module SIGN is called by the executive. The definition of valid card values for this function are given in Table A-13.

Table A-13. Type 3 Definitions for the SIGN Function.

Column Number	Variable Name	Definition
1-2	ICARD	The card number. Must be " 3".
3-8	CHAR1	The variable name.
15-20	CHAR3	The function name: "SIGN".
21-25	LOC	The variable location.
31-45	V1	The absolute value of the base.
61-62	MOD	The stage assignment.

An example of a type 3 input card utilizing the SIGN function is:

```
                                Card Column
0        1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901
03HLE         SIGN   1616     5.0
```

The variable HLE, element 1616 in the C array, will be given a value of either 5.0 or -5.0 depending on the random number value returned by the system's random number generator.

A.1.3.3.6 Type 3 - EQUALS Function

The EQUALS function is selected by entering the keyword "EQUALS" as CHAR3. Table A-14 shows the definition of the variables for this card type. This function equates the variable to the current value of another variable. This allows the value of a variable at a given time to be saved for future use. The V1 variable is the C element location of the variable to be saved.

Table A-14. Type 3 Definitions for the EQUALS Function.

Column Number	Variable Name	Definition
1-2	ICARD	The card number. Must be " 3".
3-8	CHAR1	The variable name.
15-20	CHAR3	The function name: "EQUALS".
21-25	LOC	The variable location.
31-45	V1	Variable location of the value.
61-62	MOD	The stage assignment.

An example of a type 3 input card utilizing the EQUALS function is:

```
                                Card Column
0        1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901
03HLE         EQUALS 1616     296.0
```

The variable HLE, element 1616 in the C array, will be given the current value located in the C array at element 296.

A.1.4 Type 4 Cards - Comment Cards

Type 4 cards allow the user to input short, descriptive comments within the lead card deck. The definition for the card variables is given in Table A-15. Since the card follows the base card format, it is important that the comment does not exceed column 20 otherwise an error on the input will occur. The data entered with the type 4 card is ignored and the next card in the deck is processed.

Table A-15. Type 4 Card Definitions.

Column Number	Variable Name	Definition
1-2	ICARD	The card number. Must be " 4".
3-8	CHAR1	A comment field.
9-14	CHAR2	A comment field.
15-20	CHAR3	A comment field.

CADIN converts COMMENT statements in an INPUT.ASC to type 4 cards.

A.1.5 Type 5 Cards - Multi-run Card

The type 5 card initializes the multi-run variable in the CADAC executive. This card indicates to the executive code that multiple simulations are to be calculated for the given set of primary trajectory cards. The format of the type 5 card is given in Table A-16. Typically, this card is placed at the top of the CADIN.ASC file, following the title and prior to the type 1 cards defining the modules to be executed after integration steps. The single integer input with the card type indicates the number of simulations that are to be produced for the given primary trajectory card set.

Table A-16. Type 5 Card Definitions.

Column Number	Variable Name	Definition
1-2	ICARD	The card number. Must be " 5".
26-30	MODE	The number of runs to be executed.

CADIN converts the MONTE statement to a type 5 card.

A.1.6 Type 6 Cards - Finish Trajectory Simulation

The type 6 card tells the input routine to complete the simulation based on the information that has been obtained from the input file up to this time. The reaction of the program after this trajectory has been simulated depends on the input file. If the multi-run card (type 5) has been included in the input deck, then the next trajectory will be started. If the sweep methodology is being utilized, then the sweep methodology will increment the appropriate variables and test for the end of the trajectory set.

If neither the sweep methodology nor a multi-run case is being exercised, then the program will read more cards from the CADIN.ASC file. The program will determine if a set of group cards exist on the input file, and if so, process them accordingly until a type 13 card or the end-of-file is encountered on the input file.

The format of the type 6 card follows the basic format with the only valid variable being the card type (columns 1-2). Any other input data on the card is ignored by program CADAC or causes an input error if the data type does not match the expected data type in the card format.

CADIN converts the RUN statement to a type 6 card.

A.1.7 Type 7 Card - Vector Initialization

The type 7 card allows the user to initialize minor vectors/arrays defined within the C array. The type 7 card is similar to entering multiple type 3 cards that initialize consecutive C locations. This card type differs slightly from the basic card format in that multiple data records may be entered following a special header card. The special header card format follows the base card format and the variable definitions are given in Table A-17. The data records are formatted in a list-directed format with the requirement that columns 1 and 2 remain blank. A maximum of 10 type 7 cards may be entered in stage 0 of the trajectory simulation input card deck and a total of 5 type 7 cards may be entered in the remaining stages.

Table A-17. Type 7 Header Card Definitions.

Column Number	Variable Name	Definition
1-2	ICARD	The card number. Must be " 7".
3-8	CHAR	The vector name.
21-25	LOC	The first vector C location.
26-30	MODE	Number of elements in the vector with a " ", "+" or "-" in front.
31-45	V1	The value to be assigned to all of the vector elements.
61-62	NSTAGE	The stage of the card.

The special header card indicates which vector is being initialized and where the first element of the vector resides within the C array. The absolute value of the MODE variable contains the number of elements in the vector. The MODE variable also contains a flag indicating the type of vector initialization to be performed. This flag is in the form of the sign of the data. If the sign of the MODE variable is "-", then all of the vector elements are initialized to the value input in V1. The vector elements assigned this value are the C locations starting at LOC and continuing through and including location ($\mathrm{LOC} + \|\mathrm{MODE}\|$).

If the MODE is positive (a " " or "+" in front of the value), then the vector elements are assigned individual values input by the user on the data records. The user must enter $\|\mathrm{MODE}\|$ data values on records following the special type 7 header card. These data values must start beyond column 2 and not exceed column 72. The data values must be separated by at least one space or by a comma. As many data records may be input as required to enter the correct number of values.

CADIN converts the VECTOR and VECTORV statements to type 7 cards.

A.1.8 Type 8 Cards - Input Weather Data

The type 8 card allows the user to input weather data for the atmospheric modules. This card format differs slightly from the base format in that it allows multiple cards to be entered. The first card in the sequence must be a special header card. The format follows the base format and the variable definitions are given in Table A-18. Card columns 3 through 20 are commonly left blank but may contain some information such as the words "WEATHER -DATE." This data is not utilized by program CADAC but may be helpful to the user for documenting the input data.

Table A-18. Type 8 Header Card Definitions.

Column Number	Variable Name	Definition
1-2	ICARD	The card number. Must be " 8".
3-8	CHAR1	User-entered comments.
9-14	CHAR2	User-entered comments.
15-20	CHAR3	User-entered comments.

Immediately following the type 8 header card is the weather data cards. The weather data is input in a list-directed format on each record. The data is entered in a specific order on each record. The ordering of the data is: altitude, wind direction, wind velocity, density, temperature and pressure. If the density, temperature and pressure are not going to be used, then zeros must be entered in their place; numbers must exist as place holders. Up to 50 sets of weather data may be entered. The data must be arranged such that altitude increases from one set to the next. The end of the data is signaled by a specially formatted card; the card must have a negative number in the altitude column, followed by 5 zeros. This special end-of-deck card forces the program out of the weather data input mode.

CADIN converts the WEATHER block to a type 8 card with weather data cards.

A.1.9 Type 9 Cards - Header Card

This card allows the user to input up to 5 cards containing text data. This data can be used for documentation purposes. The card type may be used in the primary trajectory and in the group card sets. The card format for the type 9 card differs slightly from the base card format in that it allows multiple cards to be entered. The first card in the sequence must be a special header card. The format of the special header card follows the base format and the variable definitions are given in Table A-19.

The header data cards then follow the type 9 card. The data cards for the type 9 input have a character format. Up to 80 characters may be entered per record with a maximum of 5 records.

Table A-19. Type 9 Header Card Definitions.

Column Number	Variable Name	Definition
1-2	ICARD	The card number. Must be " 9".
21-25	LOC	The number of header cards. Must be an integer in the range [1:5].

CADIN converts the HEADER block to a type 9 card.

A.1.10 Type 10 Cards - Staging Criteria

The type 10 card defines the stage criteria that must be satisfied for a stage test. The stage criteria are tested at the end of each integration step to determine if the stage has been completed.

Once a stage is completed, the input data for the next stage is processed and the next stage commences.

The type 10 card has two valid formats. The first card is a special header card and follows the basic card format. The definitions for this card is given in Table A-20. Its function is to indicate to the program the number of criteria tests that define the stage criteria. A blank is acceptable as a default value for the number of stage criteria; when the blank is detected, the assumption is made that only a single criteria test and card exists for the stage. Each criteria test is entered on a single card following the special header card. The format of the criteria test card does NOT follow the basic card format; the format is shown in Table A-21. A maximum of two criteria tests (and therefore criteria test records) may be entered. When multiple criteria tests are entered, the tests are considered with the "OR" operator. When any single test is satisfied, the stage completes and the next stage is initiated.

Table A-20. Type 10 Header Card Definitions.

Column Number	Variable Name	Definition
1-2	ICARD	The card number. Must be "10".
21-25	LOC	The number of criteria test cards. Must be either a blank or an integer in the range [:2] , left justified.

Table A-21. Type 10 Criteria Test Card Definitions.

Column Number	Variable Name	Format	Definition
1-6	NAME	A6	Staging variable name.
7-11	LOC	I5	Staging variable C location.
12-16	INCRS	I5	Test selection flag.
18-23	NAME2	A6	Comparative variable name.
24-38	VAL	E5.9	Comparative C location.
40-45	UNITS	I6	Unused.
46-50	KODE	I5	Stage method flag. Must be -1, 0 or 1.

In the criteria testing, a C location value is tested against a test value. The user may select among options on the type of test performed and on the form of the test value; these options provide the user with flexibility in defining the stages. The LOC variable is the element number in the C array that is to be tested as the stage criteria variable. NAME is the name associated with this location but is present for documentation purposes only; it is not used by the program code to determine the criteria variable.

There are three options available for the test that is to be performed. This is controlled by the INCRS variable. If the INCRS variable is 0, then the staging occurs when the C element is less than the criteria test value. If INCRS variable is 1, then the staging occurs when the C element is greater than the test value. If the C element equals the test value, then a stage will occur regardless of the INCRS flag.

The KODE variable in the criteria data card provides a flag for defining the criteria test value. Three options are available; KODE may have a value of 0, 1 or -1. When KODE is 0, the test value is defined as the value entered on the card for variable VAL:

(0) TestValue = VAL

The NAME2 variable is not used when KODE is set to 0.

When KODE is 1, the VAL variable becomes a location in the C array. The test value is then set to the value of this C location at the time of the test:

(1) TestValue = C(VAL)

The NAME2 variable is the name of the C element being used as the test criteria value. The NAME2 variable is for documentation purposes only and is not used by the code to determine the C location.

When KODE is set to -1, the test value is calculated as the sum of the criteria value, at the time the stage is defined, and the value in VAL:

(-1) TestValue = C(LOC) + VAR

It is important to understand that the value of the test criteria used to define the test value, is sampled when the test is defined, not when the test is performed. The NAME2 variable is not used for this case.

CADIN converts the IF statement to a type 10 card.

A.1.11 Type 11 Cards - Define Forcing Functions

Type 11 cards are useful for defining forcing functions, superimposing noise on selected variables and combining selected variables. The user can have up to 25 active functions defined at any given time during the simulation. The control functions are activated, deactivated and defined through the use of the type 11 card.

CADIN converts the FUNC statements to type 11 cards. In addition, the CLEAR FUNCTIONS statement is converted to a type 11 card.

A.1.11.1 Card 11 Definitions

Six quantities are stored for the control functions when the type 11 card is read. The format of the type 11 card is shown in Table A-22. This information is stored along with the current simulation time in the executive common named /CCOM/. The variable number, LOC, determines the variable being defined by the function. The variable name entered on the card, CHAR1, is for documentation purposes only; the name is not used to determine the variable being defined with the function.

Table A-22. Type 11 Format Definitions.

Column	Variable Name	Common Name	Format	Description
1-2	ICARD	--	I2	Card type. Must be "11".
3-8	CHAR1	--	A6	Variable name.
9		ICC	A1	Combination code " ", "=", "+", "-" or "*"
10-15	CHAR2	ICF	A5	Function name.
21-25	LOC	ICL	I5	Variable location.
26-30	MODE	ICL2	I5	Integer data for function.*
31-45	V1	CA	E5.9	Primary real data for function.*
46-60	V2	CL	E5.9	Secondary real data for function.*
61-62	MOD	-	I2	Stage assignment.

* Definition of the value depends on the function type selected.

The combination mode, read as a single string with the function name in CHAR2, determines how the calculated function value is to be utilized. Valid characters for this flag are the blank, "=", "+", "-" and "*". After the function has been evaluated, this character determines how the resulting value is combined or stored with the current contents of the variable being defined by the function. If the combination mode is a blank or "=", then the function value is stored in the variable, C(LOC). If the mode is the "+", then the function value is added to the current contents of the variable. If the mode is the "-", then the function value is subtracted from the variable. If the mode is the "*", then the function value is multiplied with the current value of the variable, C(LOC), and the results stored in the variable.

The function name, read in CHAR2, is the character string that indicates which function is to be used to provide the value for the variable. Table A-23 shows the list of valid function names and their definitions. The remaining card values, MODE, V1 and V2, are used to supply data to the function. The definitions of these values is dependent on the function selected and discussed in the following sections. However, if the V1 variable is not entered or a 0.0 is entered, the program will load the value of the controlling variable, C(LOC), at the time the function is defined, as the value of this variable.

A.1.11.2 Card 11 - STEP Function

The STEP function is defined as:

$X = V1$

The value of the step is entered in the V1 variable of the type 11 card. If the V1 variable is not entered or a 0.0 is entered, the program will load the value of the controlling variable, C(LOC), at the time the function is defined, as the value of the step. The variables MODE and V2 are not utilized by this function.

Table A-23. Type 11 Card Function Definitions.

Function Number	Function Name	Definition
1	STEP	Step function (Constant)
2	RAMP	Ramp (Constant derivative)
3	PARAB	Parabola (Constant second derivative)
4	SIN	Sine wave
5	COS	Cosine wave
6	TRI	Triangular wave
7	GAUSS	Gaussian noise
8	UNIF	Uniform random noise
9	DECAY	Exponential decay
10	SQR	Square wave
11	SUM	Sum of two variables
12	PROD	Produce of two variables
13	DIFF	Difference of two variables
14	END	Deactivates all function variables
15	RAYLE	Rayleigh noise at exponential rate
16	EQUALS	Equates first variable to a second variable

A.1.11.3 Card 11 - RAMP Function

The RAMP function (constant derivative) is computed by the equation:

$$X = V2 * (t - t_o)$$

and is defined over the range:

$$- V1 \leq X \leq V1$$

The value t_o is the time at which the function is defined. The MODE variable is not utilized in this function definition.

A.1.11.4 Card 11 - PARAB Function

The PARAB function generates a parabolic curve calculated by the following equation:

$$X = V2 * (t - t_o)^2$$

for the range:

$$- V1 \leq X \leq V1$$

The t_o value is the time at which the function was defined by the type 11 card. The MODE variable is not utilized for this function.

A.1.11.5 Card 11 - SIN Function

The SIN function generates a sine wave by using the following equation:

$$X = V1 * SIN(6.2838 * (t - t_o) / V2)$$

where t_o is the time at which the function was defined. In this definition, V1 is the amplitude of the sine curve and V2 is the period of the curve. The MODE variable is not utilized by this function.

A.1.11.6 Card 11 - COS Function

The COS function generates a cosine wave by using the following equation:

$$X = V1 * COS(6.2838 * (t - t_o) / V2)$$

where t_o is the time at which the function was defined. In this definition, V1 is the amplitude of the cosine curve and V2 is the period of the curve. The MODE variable is not utilized by this function.

A.1.11.7 Card 11 - TRI Function

The TRI function generates a triangular wave from the equation:

$$X = TRI(V1, V2, t - t_o)$$

where TRI is a function and t_o is the time at which the function was defined by the type 11 card. The V1 variable is the amplitude of the function and V2 variable is the period of the function. The following information is provided on the function:

$$-V1 \leq \text{Amplitude} \leq V1$$
$$0 \leq \text{Time} \leq (N/2) * V2$$

where N is the number of periods needed to evaluate this function at the simulation time. The MODE variable is not utilized by this function.

A.1.11.8 Card 11 - GAUSS Function

The GAUSS function computes a time correlated, Gaussian random variable with standard deviation, σ, and a time correlation coefficient, ß. The computation of the function follows these four steps:

1) Compute G, a Gaussian random variable with mean of 0 and a standard deviation of σ
2) Compute A = EXP(-ß * Dt)

3) X = G * SQRT(.0 - A*A) + LastX * A
4) Save: LastX = X

The Dt value is the integration step size of the simulation. The V1 variable is the σ value for the Gaussian distribution. The V2 variable is the β for the exponential function.

A.1.11.9 Card 11 - UNIF Function

The UNIF function provides a uniform random variable following the computation:

X = UNIF(V1, V2)

in which the function UNIF is stated as:

UNIF = V2 + 2.0 * V1 * RANF(-0.5)

where RANF is a random variable between 0 and 1.0. The UNIF function returns a uniformly distributed variable on the interval $(V2 - V1) \leq X \leq (V2 + V1)$. The MODE variable is not utilized under this function.

A.1.11.10 Card 11 - DECAY Function

The DECAY function is computed by the following equation:

X = V1 * EXP(-ABS(V2 * (t-t_o)))

where t_o is the time when the function was defined. The V1 variable is the amplitude of the function and the V2 variable is the decay rate of the function.

A.1.11.11 Card 11 - SQR Function

The SQR function generates a square wave using:

X = SQR(V1, V2, (t - t_o))

This function applies the following logic:

X = V1 when TIP $\leq$ 0.5 * V2 and X = -V1 when TIP $\geq$ 0.5 * V2

where TIP is the elapsed time within a given period, V2. TIP is calculated as:

TIP = AMOD(t - t_o, V2)

The MODE variable is not used for this function.

A.1.11.12 Card 11 - SUM Function

The SUM function assigns the value of the defined C location to be the sum of two other C location values. The V1 and V2 variables are not values but the element locations in the C array of the variables to be summed. The decimal needs to be included with the element numbers on the card. The value of the function follows the equation:

$$X = C(V1) + C(V2)$$

The MODE variable is not defined for this function.

A.1.11.13 Card 11 - PROD Function

The PROD function assigns the value of the defined C location to be the product of two other C location values. The V1 and V2 variables are not values but the element locations in the C array of the variables to be multiplied. The decimal needs to be included with the element numbers on the card. The value of the function follows the equation:

$$X = C(V1) * C(V2)$$

The MODE variable is not defined for this function.

A.1.11.14 Card 11 - DIFF Function

The DIFF function assigns the value of the defined C location to be the difference of two other C location values. The V1 and V2 variables are not values but the element locations in the C array of the variables to be differenced. The decimal needs to be included with the element numbers on the card. The value of the function follows the equation:

$$X = C(V1) - C(V2)$$

The MODE variable is not defined for this function.

A.1.11.15 Card 11 - END Function

The END function does not provide a value for assignment to a C location. The function provides a method for deactivating the currently defined functions. The only valid variables on the card for this function are the card type, ICARD, and the function name in CHAR2. When this function is selected, all of the currently defined functions are deactivated. If other control functions need to remain defined, the function may be re-defined after the END card.

A.1.11.16 Card 11 - RAYLE Function

The RAYLE function provides a special noise function. The function provides a pulse of designated width, whose height follows the Rayleigh distribution and the time between pulses follows the exponential distribution.

The MODE variable is the integer multiplier on the integration time used to determine the pulse width. The V1 variable is the mean of the exponential function used to generate the time occurrence of the next pulse. The V2 variable is the Rayleigh mode used to calculate the height of the pulse. The user should be aware that the type 11 functions are evaluated twice per each integration interval; once after the predictor cycle and once after the corrector cycle. However, this has no effect on this function. Also, if the user modifies the integration step size while an obstacle is being encountered, the new value of DER will not affect the current obstacle width but will affect the width of the next obstacle.

A.1.11.17 Card 11 - EQUALS Function

The EQUALS function allows the data in one C location to be transferred to the C location being defined by the function, thereby making the locations equal. The function is defined by the equation:

$$X = C(V1)$$

The modifications to the C location being defined are then performed according to the combination code entered on the card. The V2 and MODE variables are not defined for this function.

A.1.12 Type 12 Cards - End Of Group Cards

This card is used to indicate the end of a set of group cards. This card differs from the type 6 card in that:

1) The type 6 card is used to indicate termination of the primary trajectory cards. At this point no more input is read from CADIN.ASC until the trajectory is finished. The primary trajectory is stored in a special set of saving arrays
2) Type 12 card causes the primary trajectory to be reloaded from storage and then modified by the group card set read from CADIN.ASC

The type 12 card must always follow a set of group cards. This card has only one valid variable, the ICARD variable, which should contain the text: "12".

CADIN converts the LOAD statement to a type 12 card.

A.1.13 Type 13 Cards - End Of Lead Card Deck

The type 13 card indicates that the end of the lead cards has been reached and the program can be stopped. The type 13 card must always follow the last type 12 card. This card has only one valid variable, the ICARD variable, which should contain the text: "13".

CADIN converts the STOP statement to a type 13 card.

A.1.14 Type 16 Cards - Continue Simulation

The type 16 card is used in conjunction with the stage or type 10 cards. The type 16 card indicates that the program must stop processing cards from the input and compute the trajectory until the current stage criteria have been met. A type 16 card always follows the last criteria card of the type 10 card. When the computation of the stage is complete, the program resumes processing the input cards and the trajectory simulation is continued. This card has only one valid variable, ICARD, which should contain the text: "16".

CADIN automatically places a type 16 card in CADIN.ASC when an IF statement (type 10 card) is encountered.

A.1.15 Type 19 Cards - Sweep Outer Variable Definition

This card allows the outer variable of the sweep methodology to be defined. The initialization variable indicator MINIT (C(1600)) is usually used to indicate the type of initialization desired:

MINIT < 100	No SWEEP runs
100 ≤ MINIT < 200	Horizontal SWEEP desired
200 ≤ MINIT	Vertical SWEEP desired

Only one card of type 19 may be entered in the CADIN.ASC file.

The definitions of the variables on the type 19 card are shown in Table A-24. The LOC variable is the C location of the variable that is to be used as the outer variable in the sweep methodology. The CHAR variable is the character name of the C location but is used for documentation purposes only. The MODE variable is used to enter the step size for the outer variable. The V1 and V2 variables are the minimum and maximum values, respectively, that the outer variable is to assume. The MOD variable is a flag indicating the units of the outer variable. A value of 0 indicates that the units are in degrees and a conversion to radians is necessary; a value of 1 indicates that the units are in radians and no conversion of the units is necessary. Since the MOD variable is used to enter data pertaining to the type 19 card, the card cannot be used in a set of group cards where the MOD variable is used to indicate the stage number.

Table A-24. Type 19 Card Definitions.

Column Number	Variable Name	Definition
1-2	ICARD	The card number. Must be "19"
3-8	CHAR1	Outer variable name
21-25	LOC	Outer variable location
26-30	MODE	Delta for loop steps
31-45	V1	Minimum value for the outer variable
46-60	V2	Maximum value for the outer variable
61-62	MOD	Units indicator

When the type 19 card is encountered, the data entered on the card is loaded in C locations assigned to the sweep methodology. Table A-25 shows the C locations and associated definitions.

Table A-25. Type 19 Card C Locations.

C Location	C Name	Definition
1811	ANGLNO	Outer variable number.
1812	ANGMIN	Outer variable minimum.
1813	ANGMAX	Outer variable maximum.
1814	ANGDEL	Outer variable delta.
1815	ANGUNT	Outer variable units flag.

CADIN converts the ANGLE statement in a SWEEP block to a type 19 card.

A.1.16 Type 20 Card - Sweep Inner Variable Definition

This card allows the inner variable of the sweep methodology to be defined. The initialization variable indicator MINIT (C(1600)) is usually used to indicate the type of initialization desired:

MINIT < 100	No SWEEP runs
100 ≤ MINIT < 200	Horizontal SWEEP desired
200 ≤ MINIT	Vertical SWEEP desired

Only one card of type 20 may be entered in the CADIN.ASC file.

The definitions of the variables on the type 20 card are shown in Table A-26. The LOC variable is the C location of the variable that is to be used as the inner variable in the sweep methodology. The CHAR variable is the character name of the C location but is used for documentation purposes only. The MODE variable is used to enter the step size for the inner variable. The V1 and V2 variables are the minimum and maximum values, respectively, that the inner variable is to assume. The MOD variable is not defined for this card.

Table A-26. Type 20 Card Definitions.

Column Number	Variable Name	Definition
1-2	ICARD	The card number. Must be "20".
3-8	CHAR1	Inner variable name.
21-25	LOC	Inner variable location.
26-30	MODE	Delta for loop steps.
31-45	V1	Minimum value for the inner variable.
46-60	V2	Maximum value for the inner variable.

When the type 20 card is encountered, the data entered on the card is loaded in C locations assigned to the sweep methodology. Table A-27 shows the C locations and associated definitions.

Table A-27. Type 20 Card C Locations.

C Location	C Name	Definition
1821	RANGNO	Inner variable location.
1822	RANMIN	Inner variable minimum.
1823	RANMAX	Inner variable maximum.
1824	RANDEL	Inner variable delta.

CADIN converts the RANGE statement in the SWEEP block to a type 20 card.

A.1.17 Type 21 Cards - Sweep Criteria Definition

This card allows the sweep criteria data to be defined. The initialization variable indicator MINIT (C(1600)) is usually used to indicate the type of initialization desired:

MINIT < 100	No SWEEP runs
100 ≤ MINIT < 200	Horizontal SWEEP desired
200 ≤ MINIT	Vertical SWEEP desired

Only one card of type 21 may be entered in the CADIN.ASC file.

The definitions of the variables on the type 21 card are shown in Table A-28. Some of these definitions are dependent on the sweep option selected. The MODE variable is an integer indicating the sweep option selected by the user. There are six options numbered 0 through 5. Option 0 corresponds to the CONSTANT DELTA mode discussed in Section 3.1.2.4.3.1.1. Option 1 corresponds to the INCREASING RANGE in decreasing step size mode discussed in Section 3.1.2.4.3.1.2. Option 2 corresponds to the DECREASING RANGE in increasing step size mode discussed in Section 3.1.2.4.3.1.3. Option 3 corresponds to the INCREASING/DECREASING STEP SIZE mode discussed in Section 3.1.2.4.3.1.4. Option 4 corresponds to the OUTER BOUNDARY TEST mode discussed in Section 3.1.2.4.3.1.5 and Option 5 corresponds to the ALL BOUNDARY TEST mode of Section 3.1.2.4.3.1.6.

If for the type 21 card, option 0 is selected, the only variables used are LOC, MODE and V1; these are optional and therefore not necessary. If option 1 through 3 is selected, the only required variable is V2. This variable is the number of trajectories to be executed for each ray (single outer variable value). Option 4 or 5, the LOC variable is the C location of the variable to be used as the testing criteria in the sweep methodology. The CHAR variable is the character name of the criteria variable but is used for documentation purposes only. The V1 variable is the critical value the criteria value is to be tested against. The MOD variable indicates the number of binary searches to be performed under option 4 or 5. Since the MOD variable is used to enter data pertaining to the type 21 card, the card cannot be used in a set of group cards where the MOD variable is used to indicate the stage number.

Table A-28. Type 21 Card Definitions.

Column Number	Variable Name	Definition
1-2	ICARD	The card number. Must be "21".
3-8	CHAR1	Criteria variable name.
21-25	LOC	Criteria variable location.
26-30	MODE	Sweep mode.
31-45	V1	Critical value.
46-60	V2	Number of trajectories.
61-62	MOD	Number of binary searches.

When the type 21 card is encountered, the data entered on the card is loaded in C locations assigned to the sweep methodology. Table A-29 shows the C locations and associated definitions.

Table A-29. Type 21 Card C Locations.

C Location	C Name	Definition
1801	CRITNO	Critical variable number
1802	CRITVAL	Critical test value
1803	SEARNO	Number of binary searches*
1804	NUMR	Number of trajectories.**

* Used in sweep option 4 only

** Used in sweep options 0 through 3 only but not required for option 1

CADIN converts the MODE, NUM and LIMIT statements in a SWEEP block to a type 21 card.

A.1.18 Type 90 Cards - Save Simulation State

The type 90 card causes the CADAC executive to save the state of the program. The type 90 card must always follow a type 6 card. Only one card variable, ICARD, is valid on the type 90 card; this card must contain the text: "90". Only one tape 90 card is allowed in a CADIN.ASC file.

When the type 90 card is detected in the input data, the current program state, at the time the card is processed, is saved to a CSAVE.ASC data file. The trajectory simulation then continues as directed by the primary trajectory cards. When the primary trajectory is complete and a multi-run case is to be executed, the appropriate set of group input cards is processed, the state data saved to the CSAVE.ASC file is restored and execution of this new trajectory continues as directed by the trajectory cards.

CADIN converts the SAVE statement to a type 90 card.

APPENDIX B

Statistics and Derivations for Random Number Distributions

B.1 Statistics and Derivations for Random Number Distributions

To generate random deviates with a uniform probability, p(x)dx (the probability of generating a number between x and x + dx) is defined as:

$$p(x)dx = \begin{cases} dx & 0 < x < 1 \\ 0 & \text{otherwise} \end{cases}$$

therefore:

$$|p(y)dy| = |p(x)dx|$$

becomes:

$$|p(y)dy| = |p(x)dx|$$

$$p(y) = \frac{dx}{dy}$$

Since:

$$p(y) = f(y)$$

the equation becomes:

$$f(y) = \frac{dx}{dy}$$

The solution to this equation is simply:

$$x = F(y)$$

where F(y) is the indefinite integral of f(y). The transformation needed to transform a uniform deviate to a deviate distributed as f(y) is then found by taking the inverse of F(y):

$$y(x) = F^{-1}(x)$$

This technique was used to derive the formulas for generating random numbers following the exponential and Rayleigh distributions. Appendix Section B.1.1 discusses the derivations pertaining to the exponential distribution and Appendix Section B.1.2 discusses the Rayleigh distribution.

B.1.1 Exponential Distribution

The probability density function of the exponential distribution is:

$$f(y) = \lambda e^{-\lambda y}$$

where λ is a constant. Using the method described in Section B.1, the exponential deviates were computed as follows. Starting with the definition of the exponential probability distribution function:

$$f(y) = \lambda e^{-\lambda y}$$

letting λ = 1, then:

$$f(y) = e^{-y}$$

$$F(y) = \int e^{-y}\ dy$$

$$F(y) = -e^{-y}$$

letting $x = e^{-y}$ and solving for y:

$$F^{-1}(x) = -\ln x$$

From this we get the function EXPDEV:

$$\text{EXPDEV} = -\text{ALOG}(\text{RANF}(\ ))$$

where RANF is a uniform random number generator with a range of (0,1). Function EXPDEV is exponentially distributed with a mean of 1 and variance of 1. Exponential random numbers with a different mean and variance can be generated by multiplying the value generated by EXPDEV by the desired mean, as seen in the module EXPON. For example, to generate a random number with a desired mean variance of ($1/\lambda$, $1/\lambda^2$):

$$E(x) = E\left(\frac{1}{\lambda}y\right) \qquad \text{where } y \approx \exp(1,1)$$

$$= \frac{1}{1}E(y) \qquad \text{since } E(ax) = aE(x)$$

since y ≈ exp(1,1), E(y) = 1 and therefore E(x) = 1/λ. And for the variance:

$$
\begin{aligned}
Var(x) &= Var\left(\frac{1}{\lambda} y\right) && \text{where } y \approx \exp(1,1) \\
&= \frac{1}{\lambda^2} Var(y) && \text{since } Var(ax) = a^2 Var(x)
\end{aligned}
$$

since y ≈ exp(1,1), Var(y) = 1 and so Var(x) = $1/\lambda^2$. Therefore (EXPDEV*(1/λ)) is exponentially distributed with a mean of 1/λ and a variance of $1/\lambda^2$. Module EXPON performs these calculations as a driver for function EXPDEV. Figure B-1 shows these functions implemented in FORTRAN. Figures B-2 and B-3 show the derivation of the mean and variance of the exponential distribution, respectively.

```
      FUNCTION EXPDEV()
C-----------------------------------------------------------------------------------------
C
C     EXPDEV -    function to generate a random variable with an
C                 exponential distribution with a mean and variance
C                 of (1,1) using a uniform random function with a mean
C                  of (0,1).
C
C                 ( returns an exponentially distributed, positive,
C                 random deviate of unit mean using RANF() as the
C                 source of uniform deviates )
C
C-----------------------------------------------------------------------------------------
C
      EXPDEV = - ALOG ( RANF() )
C
      RETURN
      END



      FUNCTION EXPON( RMEAN )
C-----------------------------------------------------------------------------------------
C
C     This function generates a random variable that is exponentially
C     distributed random with a mean of RMEAN.
C
C-----------------------------------------------------------------------------------------
C
C     RMEAN -     the mean of the exponential distribution; input by the
C                 user.
C
C     EXPDEV -    function to generate a random variable with an
C                 exponential distribution with a mean and variance
C                 of (1,1) using a uniform random function with a mean
C                 of (0,1).
C
C      EXPON  -   a random variable having a exponential distribution
C                  with a mean and variance of (RMEAN,RMEAN**2).
C
C-----------------------------------------------------------------------------------------
C
      EXPON = EXPDEV() * RMEAN
C
      RETURN
      END
```

Figure B-1. Function EXPDEV and Function EXPON.

$$
\begin{aligned}
E(y) &= \int_0^\infty y\, f(y)\, dy \\
&= \int_0^\infty \; y\,(\lambda\, e^{-\lambda y})\, dy \\
&= \int_0^\infty \; \lambda\, y\, e^{-\lambda y}\, dy
\end{aligned}
$$

using parts:

λy	$e^{-\lambda y}\, dy$
λ	${}^{+}\left(-\frac{1}{\lambda} e^{-1\lambda y}\right)$
0	${}^{-}\left(-\frac{1}{\lambda^2} e^{-\lambda y}\right)$

$$
\begin{aligned}
E(y) &= \left[-y\, e^{-\lambda y} - \frac{1}{\lambda}\; e^{-\lambda y}\right]_0^\infty \\
&= \left(-\infty\, e^{-\infty} - \frac{1}{\lambda}\; e^{-\infty}\right) + \left(0\, e^{0} + \frac{1}{\lambda}\; e^{0}\right) \\
E(y) &= \frac{1}{\lambda}
\end{aligned}
$$

Figure B-2. Expected Value of the Exponential Distribution.

$$\mathrm{Var}\ (\mathrm{y}) = \mathrm{E}\left(\mathrm{y}^2\right)\ -\ \left[\mathrm{E}(\mathrm{y})\right]^2$$

$$\begin{aligned}
E\left(\mathrm{y}^2\right) &= \int_0^\infty \mathrm{y}^2\ f(y)\,dy \\
&= \int_0^\infty \mathrm{y}^2\,\lambda\, e^{-\lambda y}\,dy \\
&= \int_0^\infty \lambda\,\mathrm{y}^2\, e^{-\lambda y}\,dy
\end{aligned}$$

using parts:

$$\begin{array}{ll}
\lambda y^2 & e^{-\lambda y}\,dy \\
\hline
2\lambda y & {}^{+}\left(-\frac{1}{\lambda}e^{-1\lambda y}\right) \\
2\lambda & {}^{-}\left(-\frac{1}{\lambda^2}e^{-\lambda y}\right) \\
0 & {}^{+}\left(-\frac{1}{\lambda^3}e^{-\lambda y}\right)
\end{array}$$

$$\mathrm{E}\left(\mathrm{y}^2\right) = \left[-y^2e^{-\lambda y} - 2\left(\frac{1}{\lambda}\right)ye^{-\lambda y} - 2\left(\frac{1}{\lambda^2}\right)e^{-\lambda y}\right]_0^\infty$$

$$= -\left[\left(\infty e^{-\lambda y} + 2\left(\frac{1}{\lambda}\right)\infty e^{-\infty} + 2\left(\frac{1}{\lambda^2}\right)e^{-\infty}\right) + \left(0 + 0 + 2\left(\frac{1}{\lambda^2}\right)e^{0}\right)\right]$$

$$\mathrm{E}\left(\mathrm{y}^2\right) = \frac{2}{\lambda^2}$$

$$\begin{aligned}
Var(y) &= \mathrm{E}\left(\mathrm{y}^2\right) - \left[\mathrm{E}(\mathrm{y})\right]^2 \\
&= \frac{2}{\lambda^2} - \left(\frac{1}{\lambda}\right)^2 \\
&= \left(\frac{1}{\lambda}\right)^2
\end{aligned}$$

Figure B-3. Variance of the Exponential Distribution.

B.1.2 Rayleigh Distribution

The probability density function of the Rayleigh distribution is:

$$f(y) = \frac{y}{\alpha^2} e^{\frac{-y^2}{2\alpha^2}}$$

where α is a constant. Using the method outlined in Section B.1, the Rayleigh deviates were computed as follows:

$$f(y) = \frac{y}{\alpha^2} e^{\frac{-y^2}{2\alpha^2}}$$

letting $\alpha = 1$:

$$f(y) = y\,e^{\frac{-y^2}{2}}$$

$$F(y) = \int y\,e^{\frac{-y^2}{2}}\,dy$$

Let u = ($y^2/2$), then du = y dy. Substituting:

$$\Rightarrow \int e^{-u}\,du$$

$$= -e^{-u}$$

Re-substituting u = ($y^2/2$) gives:

$$F(y) = -e^{\frac{-y^2}{2}}$$

Letting $x = -e^{-\frac{y^2}{2}}$ and solving for *y* gives:

$$F^{-1}(x) = \sqrt{-2\ln x}$$

From this we get the function RAYDEV:

```
RAYDEV = SQRT( -2.0 * ALOG ( RANF (  ) ) )
```

where RANF is a uniform random number generator with a range of (0,1). Function RAYDEV is Rayleigh distributed with an α value of 1, which gives a mean of $\sqrt{\pi/2}$ and a variance of (2-π /2). Rayleigh random numbers with a different mean and variance can be generated by multiplying the value generated by RAYDEV by the constant α. This is validated by the following:

$$\begin{aligned} E(x) &= E(\alpha y) && \text{where } y \approx \text{Rayleigh}\left(\sqrt{\frac{\pi}{2}}, \left(2-\sqrt{\frac{\pi}{2}}\right)\right) \\ &= \alpha E(y) && \text{since } E(ax) = aE(x) \end{aligned}$$

since y ≈ Rayleigh ($\sqrt{\pi/2}$, (2-π/2)), E(y) = $\sqrt{\pi/2}$, then E(x) = $\alpha\sqrt{\pi/2}$. And for the variance:

$$\begin{aligned} Var(x) &= Var(\alpha y) && \text{where } y \approx \text{Rayleigh}\left(\sqrt{\frac{\pi}{2}}, \left(2-\sqrt{\frac{\pi}{2}}\right)\right) \\ &= \alpha^2 Var(y) && \text{since } \mathrm{Var}(ax) = a^2 Var(x) \end{aligned}$$

since, y ≈ Rayleigh ($\sqrt{\pi/2}$, (2-π/2)), Var(y) = (2 - π/2), then Var(x) = α^2 (2- π/2). Therefore, (RAYDEV * α) is Rayleigh distributed with a mean of $\alpha\sqrt{\pi/2}$ and a variance of α^2 (2-π/2). This algorithm is implemented in the RAYLEIGH module. Figure B-4 shows these algorithms implemented in FORTRAN. Figures B-5 and B-6 show the derivation of the mean and variance for the Rayleigh distribution.

```
      FUNCTION RAYDEV()
C--------------------------------------------------------------------------------------
C     RAYDEV -    function to generate a random variable with a
C                 RAYLEIGH distribution with a mean and variance
C                 of ( SQRT(PI/2), (2-PI/2) ) using a uniform random
C                 function with a mean of (0,1).
C     ( returns a RAYLEIGH distributed, positive, random deviate of PI/2
C     mean and (2-PI/2) using RANF() as the source of uniform deviates )
C
C     The deviate is found by finding the inverse of the indefinite integral
C     of the pdf. ie. If the pdf is given by f(y), the indefinite  integral of f(y)
C      is F(y).  The deviate is then the inverse of F(y).
C
C     The Rayleigh is                ( X / alpha**2 ) * exp( - X**2 / ( 2 * alpha**2) )
C
C     With an alpha of 1, we get     X * exp( - X**2 / 2 )
C
C     The indefinite integral is:    exp( -X**2 / 2 )
C
C     The inverse is then:           SQRT( -2 * LN(Y) )
C
C     Where y is a uniformly distributed random variable.
C--------------------------------------------------------------------------------------
C
      RAYDEV = SQRT( 2.0 * ( - 1.0 * ALOG( RANF() )  ) )
      RETURN
      END



      FUNCTION RAYLEIGH( SIG )
C--------------------------------------------------------------------------------------
C     RAYLEIGH = X E**-( X**2 / SIG**2 *2 ) / SIG ** 2
C     MEAN = SIG * SQRT( PI / 2 )
C     VARIANCE = SIG**2 * ( 2 - PI / 2 )
C     RAYDEV ~ RAYLEIGH( SQRT(PI/2), (2-PI/2) )
C
C     Multiplying by sigma gives a mean of SIG*SQRT(PI/2) and a
C     variance of SIG**2 ( 2-PI/2 )
C---------------------------------------------------------------------
C
      RAYLEIGH = RAYDEV() * SIG
      RETURN
      END
```

Figure B-4. Function RAYDEV and Function RAYLEIGH.

$$E(y) = \int_0^\infty y \; f(y) \; dy \qquad \text{(Step 1)}$$

$$= \int_0^\infty y \left(\frac{y}{\alpha^2} \; e^{\frac{-y^2}{2\alpha^2}} \right) dy \qquad \text{(Step 2)}$$

Let β = 1 / α^2 and substitute it into the equation:

$$\Rightarrow \int_0^\infty \beta \; y^2 \; e^{\frac{-\beta y^2}{2}} \; dy \qquad \text{(Step 3)}$$

$$= \int_0^\infty y \; e^{\frac{-\beta y^2}{2}} \; \beta \; y \; dy \qquad \text{(Step 4)}$$

Let u = (βy^2)/2, then du = βy dy. Solving u = (βy^2)/2 for y gives: y = $u^{\frac{1}{2}} \sqrt{2/\beta}$. Substituting these equations into Step 4:

$$\Rightarrow \int_0^\infty \sqrt{\frac{2}{\beta}} u^{\frac{1}{2}} \; e^{-u} \; du \qquad \text{(Step 5)}$$

$$= \sqrt{\frac{2}{\beta}} \int_0^\infty u^{\frac{1}{2}} \; e^{-u} \; du \qquad \text{(Step 6)}$$

$$\Rightarrow \sqrt{\frac{2}{\beta}} \int_0^\infty u^{\left(\frac{3}{2}-1\right)} \; e^{-u} \; du \qquad \text{(Step 7)}$$

Recall the gamma function, i.e.,

$$\Gamma(a) = \int_0^\infty x^{(a-1)} \, e^{-x} \, dx$$

with the characteristics:

$$\Gamma(a+1) = a\,\Gamma(a)$$
$$\Gamma(1) = 1$$
$$\Gamma\left(\frac{1}{2}\right) = \sqrt{\pi}$$

Substituting the gamma function, with **a** = 3/2, into Step 7 gives:

$$E(y) = \sqrt{\frac{2}{\beta}} \; \Gamma\left(\frac{3}{2}\right) \qquad \text{(Step 8)}$$

Figure B-5. Expected Value of the Rayleight Distribution.

$$= \sqrt{\frac{2}{\beta}}\ \Gamma\left(\frac{1}{2}+1\right) \qquad \text{(Step 9)}$$

$$E(y) = \sqrt{\frac{2}{\beta}}\ \left(\frac{1}{2}\right)\left(\Gamma\left(\frac{1}{2}\right)\right) \qquad \text{(Step 10)}$$

$$E(y) = \sqrt{\frac{2}{\beta}}\ \left(\frac{1}{2}\right)\sqrt{\pi} \qquad \text{(Step 11)}$$

$$= \sqrt{\frac{2}{\beta}\left(\frac{1}{4}\right)(\pi)} \qquad \text{(Step 12)}$$

$$= \sqrt{\frac{\pi}{2\beta}} \qquad \text{(Step 13)}$$

Recall, $\beta = 1 / \alpha^2$ so Step 13 then becomes

$$\mathrm{E}(\mathrm{y}) = \sqrt{\frac{\pi\alpha^2}{2}} \qquad \text{(Step 14)}$$

$$\mathrm{E}(\mathrm{y}) = \alpha\sqrt{\frac{\pi}{2}}$$

Figure B-5. Expected Value of the Rayleight Distribution. (Concluded)

$$E(y) = E(y^2) - [E(y)]^2 \qquad \text{(Step 1)}$$

$$E(y^2) = \int_0^\infty y^2 \; f(y)\, dy \qquad \text{(Step 2)}$$

$$= \int_0^\infty y^2 \left(\frac{y}{\alpha^2} \; e^{\frac{-y^2}{2\alpha^2}} \right) dy \qquad \text{(Step 3)}$$

Let β = 1 / α^2 :

$$E(y^2) = \int_0^\infty \beta y^3 e^{\frac{-\beta y^2}{2}} \; dy \qquad \text{(Step 4)}$$

$$= \int_0^\infty y^2 \; e^{\frac{-\beta y^2}{2}} \; \beta \; y \; dy \qquad \text{(Step 5)}$$

Let u = (β y^2)/2, then du = β y dy. Solving u = (β y^2)/2 for y^2 gives: y^2 = (2u)/β. Substituting into Step 5:

$$\Rightarrow \int_0^\infty \frac{2}{\beta} \, u \, e^{-u} \, du \qquad \text{(Step 6)}$$

$$= \frac{2}{\beta} \int_0^\infty \; u^{2-1} \; e^{-u} \, du \qquad \text{(Step 7)}$$

Using the gamma function (as described in solving for the expected value of the Rayleigh) gives:

$$\mathrm{E}(\mathrm{y}^2) = \; \frac{2}{\beta} \, \Gamma(2) \qquad \text{(Step 8)}$$

$$= \; \frac{2}{\beta} \, \Gamma(1+1) \qquad \text{(Step 9)}$$

$$= \; \frac{2}{\beta} (1) \Gamma(1) \qquad \text{(Step 10)}$$

$$= \; \frac{2}{\beta} \qquad \text{(Step 11)}$$

Recall β = (1 / α^2), so Step 11 then becomes

$$\mathrm{E}(\mathrm{y}^2) = 2\alpha^2 \qquad \text{(Step 12)}$$

$$\mathrm{Var}(\mathrm{y}) = \; \mathrm{E}(\mathrm{y}^2) \; - \; [\mathrm{E}(\mathrm{y})]^2$$

$$\mathrm{Var}(\mathrm{y}) \; = \; 2\alpha^2 \; - \; \frac{\alpha^2 \pi}{2} \qquad \text{(Step 13)}$$

$$\mathrm{Var}(\mathrm{y}) \; = \; \alpha^2 \left(2 - \frac{\pi}{2} \right) \qquad \text{(Step 14)}$$

Figure B-6. Variance of the Rayleigh Distribution.

APPENDIX C

Delivery Accuracy Normalization Methodology

C.1 NORMALIZE

The objective of this methodology is to normalize the distribution of impact points by removing gross errors and direct hits from the data set at each aimpoint. The following methodology was developed, documented and implemented by Steve McGrath of NAWCWD at China Lake, CA

C.1.1 Input

The following quantities are input to the subroutine:

- R_T - Target Radius.
- R_{GRAN} - Granularity in the estimates of SD_x and SD_y for the Near Miss (NM) impacts. R_{GRAN} is the Normalization Error Bound (NEB).
- PC_{ce} - Confidence level for CEPs, DEPs, REPs.
- PC_{ee} - Confidence level for the containment ellipse.

C.1.2 Output

The subroutine NORMLZ outputs the following:

- R_{pnm} - Radial Distance Associated with the Probability of Near Miss. This is the circle of radius R_{pnm} about the aimpoint which contains 99.97% of the NM impacts.
- R_{bias} - The radial bias for the NM impacts.
- MPI_x - X-coordinate (crossrange) of the MPI of the NM impacts.
- MPI_y - Y-coordinate (downrange) of the MPI of the NM impacts.
- CEP_m - Radius of the circle about the MPI for the NM impacts corresponding to a confidence level of PC_{ce}.
- CEP_a - Radius of the circle about the aimpoint (0,0) for the NM impacts corresponding to a confidence level of PC_{ce}.
- REP_m - "Radius" of the interval about MPI_y for the y-coordinates of NM impacts corresponding to a confidence level of PC_{ce}.
- REP_a - "Radius" of the interval about 0 for the y-coordinates of NM impacts corresponding to a confidence level of PC_{ce}.
- DEP_m - "Radius" of the interval about MPI_x for the x-coordinates of NM impacts corresponding to a confidence level of PC_{ce}.
- DEP_a - "Radius" of the interval about 0 for the x-coordinates of NM impacts corresponding to a confidence level of PC_{ce}.
- E_{min} - "Radius" of the minor axis of the containment ellipse about the MPI for the NM impacts.
- E_{maj} - "Radius" of the major axis of the containment ellipse about theMPI for the NM impacts.
- E_{az} - Angle (in radians) corresponding to a coordinate frame rotation which uncorrelates the x- and y-coordinates of the NM impacts.

- PC_{dh} - Percentage of direct hits (DH).
- PC_{ge} - Percentage of gross errors (GE).
- PC_{ce} - Confidence level for the CEP, DEP and REP.
- PC_{ee} - Confidence level for the containment ellipse.

The subroutine also outputs matrices DH and GE containing the direct hits and gross error impacts, respectively. The radial miss of each of these impacts is also output (each matrix has three columns). Gross error impacts are determined as those for which at least one of the coordinates is at least 4 standard deviations away from the corresponding MPI coordinate.

C.1.3 Determination Of The Standard Deviation For The NM Impacts

In reality, some of the DH impacts are NM impacts. The problem is to estimate SD_x and SD_y for the NM impacts without having any sample data within a distance R_T from the aimpoint. The situation is depicted in Figure C-1.

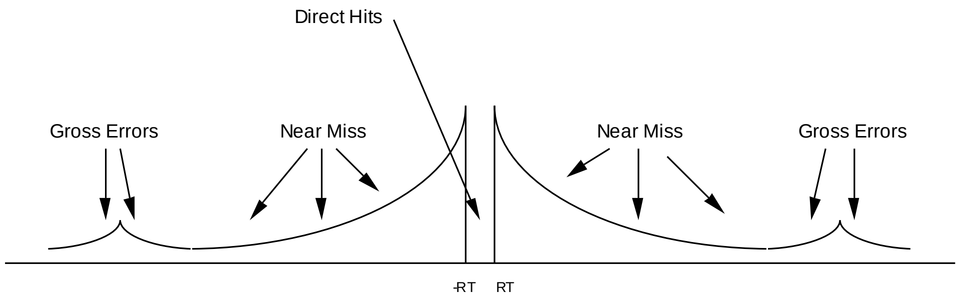

Figure C-1. Delivery Accuracy Impact Case.

The algorithm used to estimate SD_x and SD_y compares the empirical distribution function with the theoretical distribution function computed using the Gaussian Cumulative Distribution Function (GCDF). GCDF is evaluated using the sample mean and a candidate sample SD. For a candidate SD, comparisons are made at each point in the NM sample and summed over all points. The candidate SD which minimizes the difference between the empirical and theoretical distributions is the best estimate of the SD for the NM impacts. It is necessary to expand the sample size of the NM impacts otherwise the algorithm will always overestimate SDx and SDy (it does anyway, but to a lesser degree than if the sample size was not expanded). The expansion factor is determined by estimating the percentage, p, of NM impacts within the target radius. The estimate is computed as:

$$p = 2*(GCDF(0.0,SD,R_T) - 0.5)$$

where SD is the candidate sigma. The situation is depicted in Figure C.2. The quantity q = (1-p) is the percentage of NM impacts beyond RT. Thus the number of NM impacts must be expanded from K_{NM} to $(1/q)*K_{NM}$ to provide a more accurate computation of empirical probabilities.

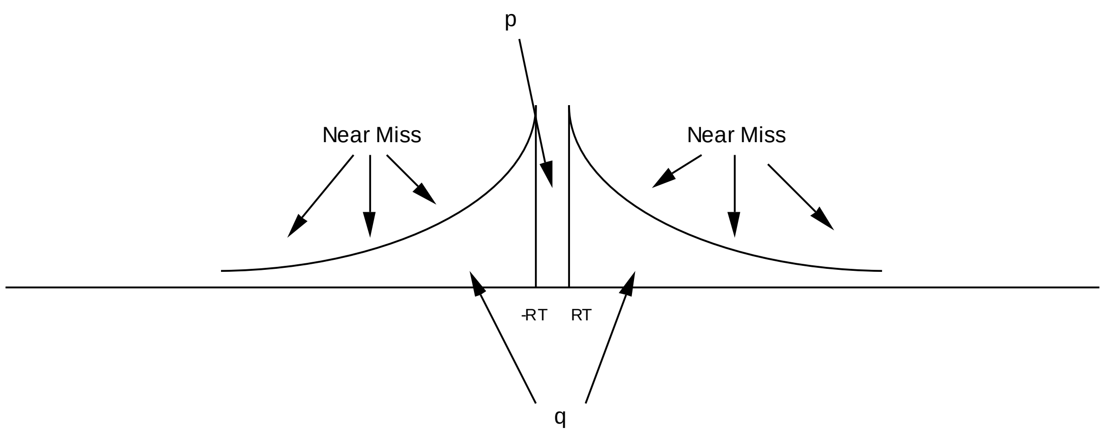

Figure C-2. Delivery Accuracy Normalization.