CADAC USER DOCUMENTATION

VOLUME II

Release 3.1
June 2000

Prepared For:
Air Force Research Laboratory (AFRL/MNG)
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 (Concluded)

List of Figures

List of Tables

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 CADAC web 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*** (this report) 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*** 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 What's New - Release 3.1

Several programs have been altered or combined to simplify the CADAC process. The changes are as follows:

CADX - The CADAC executive routine has been modified: CADAC checks to ensure that the user has correctly set up a SWEEP or single run execution. The MINIT variable as a SWEEP run designation was also eliminated. A new utilities file has been generated, UTL3.FOR.
CADIN – The program has been modified to allow full length comment records.

DFHEAD – The program has been modified to include comments in the HEAD.ASC for any executive variables found equivalenced in the modules.
GLOBE - The terrain display option was modified to allow the legend display simultaneously.
KPLOT – A new log and trajectory file search is now included for all programs associated with all the KPLOT programs. This capability allows for the automatic opening of a log or trajectory file within the local directory. A preferences menu has also been added to the KPLOT interface and the DrawOLE graphics engine. This preferences option allows the user to define the default value for many graphics options. The following upgrades have been made to each program:

- **2DIM:** Modifications were made to correct problems encountered with version 3.0.
- **PITA:** A user entered scaling option was added to the interactive input screen. The user enters X, Y and Z minimums and maximums for the desired plot.
- **GLOBE:** Modifications were made to the terrain display option to simultaneously display the color code and terrain.
- **BIVAR:** Modifications were made to correct problems encountered with version 3.0.

INPUT – The program has been modified to allow full length comment records.

1.2 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 as well as advanced post CADAC utilities.

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.

1.3 CADAC Installation

The CADAC Installation is described in the following sections.

~~1.1.1~~ 1.3.1 Web Site Installation

Formatted: Bullets and Numbering

For files downloaded from the CADAC website, the user will begin with a single file in a self-extracting zip format (*.EXE). The self-extracting zip files should be downloaded to a temporary directory (i.e. \WINDOWS\TEMP). After the program has been expanded the user can execute the program named SETUP.EXE. This program will begin installing the proper files on the hard drive. The install program will create all necessary directories. Some windows *.DLL and *.OLE files will be installed onto the computer system (see file list). The original zipped file should be retained for archive proposes; the expanded programs in the temporary directory can be deleted to recover disk space.

~~1.1.2~~ 1.3.2 CD-ROM Installation

Formatted: Bullets and Numbering

The data residing on the CADAC web site is duplicated on the CADAC CD-ROM with a few exceptions. The CD-ROM directory follows:

CADAC\DOC – Documentation in 4 volumes
CADAC\CADX – CADAC executive and utility FORTRAN programs, test case
CADAC\KPLOT – Plotting programs: 2DIM, PITA, GLOBE, installation required
CADAC\SWEEP – Footprint and launch envelop generation programs, installation required
CADAC\CHKINT – Integration variable checking program, installation required
CADAC\REALTIME – Converter program for CADAC real-time code generation
CADAC\TESTSIM – 6 DOF air-to-air missile simulation (SR1S)

The application directories (KPLOT, SWEEP and CHKINT) on the CD-ROM contain the installation files for each program. Program installation begins by executing the setup file (SETUP.EXE) located in the application directory.

1.4 Running the CADAC Test Case Using Digital Visual FORTRAN

To run the test case included in the CADAC zip files using Digital Visual FORTRAN (DVF), install the files as described in Section 1.4. Within the CAD\TEST directory, execute CADIN to convert the INPUT.ASC to a CADIN.ASC file. Load DVF from within Windows. Select the **File, New** menu option and select the **Projects** tab. Highlight the Win32 Console Application item. If the current directory is CAD\TEST, enter the name CADAC in the project name box. If the current directory is not CAD\TEST, use the **BROWSE** option and change the directory to CAD\TEST and then enter CADAC in the file name prompt box. Select **OK**. This will create a new subdirectory CADAC. Source code files are loaded into the project by using the project menu options. Select **Add to Project** option and choose **Files** from the menu. The files necessary for the CADAC project will be found in the CAD\TEST directory and the CAD directory. The files to be added to the project from the CAD\TEST directory are MODULE.FOR and DUMMY.FOR. The files to be added to the project from the CAD directory are: CADX3.FOR and UTL3.FOR. From the **Build** menu option, select **Build CADAC.EXE** and then select **Execute CADAC.EXE**.

2. CADAC ENVIRONMENT

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 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 files in the CADAC environment can have a .FOR extension, indicating a FORTRAN file; a .EXE extension, indicating an executable file; a .OBJ extension, indicating an object file; a .BIN extension, indicating a binary or a .ASC indicating an ASCII file.

2.1 CADAC Program Files

The CADAC program files consist of the following files: CADX.FOR, UTL.FOR, MODULE.FOR and DUMMY.FOR. The structure of the CADAC program is shown in Figure 1. CADX.FOR, called the CADAC Executive, forms the top level simulation control and calls the user-supplied modules, located in the MODULE.FOR file. The CADX.FOR and UTL.FOR files contain pre-written FORTRAN code for use by the simulation programmer. The MODULE.FOR and DUMMY.FOR files contain FORTRAN code generated by the programmer and are specific to the missile system or vehicle being simulated. When these files are compiled and linked together, the resulting executable file performs the simulation.

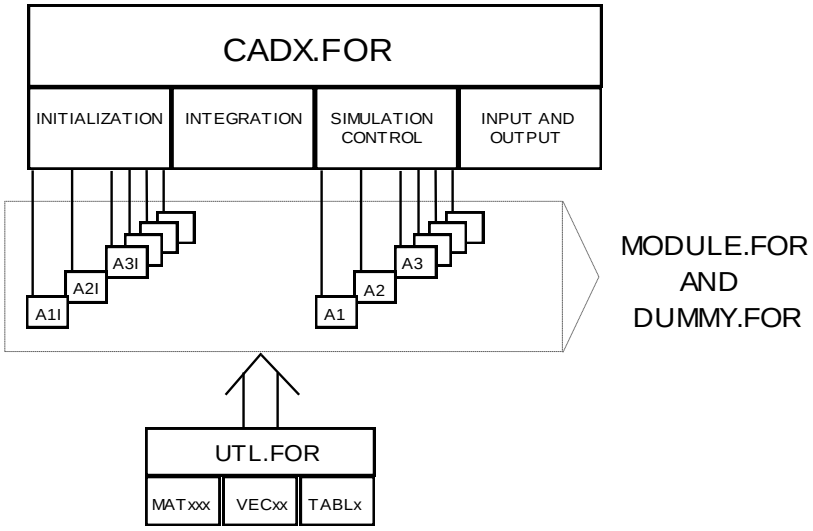

Figure 1. CADAC Program Overview.

2.1.1 CADX.FOR

The CADX.FOR file is called the CADAC program executive; this is the driver for the CADAC simulation and is an existing, compilable program file. The CADAC executive controls input of the simulation scenario, data output, state variable integration via a predictor/corrector methodology and data processing. In most typical simulations, the CADAC executive should **not** be modified by the simulation programmer.

2.1.2 UTL.FOR

The UTL.FOR file contains several pre-written, generic subroutines that perform array processing and table interpolation. These subroutines are available for access by any module of the simulation. Appendix A provides a list of the subroutines provided by this file and their function. Modification of these subroutines by the simulation programmer does not occur in typical simulation development. However, the programmer may elect to place in this file any similar, generic subroutines written to support the simulation. The utility subroutine names usually have the following format depending on the function:

"MATxxx" - Matrix operations
"VECxx" - Vector operations
"TABLx" - Table look-up operations
“CADxx” - Special Utilities

2.1.3 MODULE.FOR and DUMMY.FOR

The MODULE.FOR and the DUMMY.FOR files contain the FORTRAN subroutines generated by the programmer that are specific to the missile system/vehicle simulation. The MODULE.FOR contains a coded set of initialization subroutines and a set of main subroutines that perform the simulation. The DUMMY.FOR contains SUBROUTINE, RETURN and END statements for any of the initialization and main subroutines **not** used in the simulation.

2.2 Programming Conventions

Since the executive portion of CADAC is given and the code providing the simulation modeling is supplied by the programmer, a set of programming conventions exists to provide communication between the separate code entities as well as assist in communications between various CADAC users.

~~1.1.1~~ 2.2.1 Module Programming Conventions

Formatted: Bullets and Numbering

The simulation subroutines supplied by the user have specific names assigned by the CADAC Executive. These names are: A1, A2, A3, A4, C1, C2, C3, C4, D1, D2, D3, D4, G1, G2, G3, G4, S1, S2, S3 and S4. These major subroutines are termed "modules". Typically each module is coded to provide a model of a particular subsystem within the simulation. The

CADAC programming structure defines the functions of some of the modules. These module functions are listed in Table 1. Due to the two step integration methodology, each module is called by the CADAC executive twice per time step. Therefore, the modules should be coded to provide the state of the subsystem simulation at a given time in the trajectory.

Table 1. Suggested Names and Function Definitions.

MODULE NAME	MODULE FUNCTION	MODULE NUMBER	COMMENTS
A1	Aero Coefficients	2	
A2	Propulsion Model	3	
A3	Forces & Moments Calculations	4	
A4		5	Available for Growth
C1	Guidance Model	7	
C2	Auto pilot Model	8	
C3	Thrust Vector Control	9	
C4	Actuator Model	10	
D1	Newton's Equations	17	
D2	Euler Equations	18	
D3		19	
D4		20	
G1	Target	22	Two Point Targets
G2	Air Data	23	Atmospheric, Wind and Gravity Modeling
G3	Kinematics	24	
G4	Terminal	Not a module	Provides End-of-Trajectory Testing
S1	Seeker Model	28	
S2	Sensor Model	29	GPS, AI Radar
S3	NAV Filter Model	30	
S4	INS Model	31	

To facilitate structured programming, a module may have one or more sub-level modules associated with it. However, extensive levels of subroutine calls should be avoided. These sub-level modules should only be called from the associated module. The name of the sub-level module must contain the name of the associated module name as the first two characters. For example, the S1 module provides the seeker model for the system and may be coded to provide options between several types of seekers. A sub-level module may provide the model for a kinematic seeker; an example of the sub-level module name is S1KINEM.

Associated with each module is an initialization subroutine, which is also assigned a specific subroutine name. The initialization subroutine names have the format "xxI" where "xx" is the two character name of the associated module. For example, S1I is the name of the initialization module associated with the S1 module. The CADAC executive calls these subroutines once, at the beginning of the trajectory.

2.2.2 Naming Convention

The module programming convention includes a set of naming conventions for scalars, vectors and matrices. Table 2 provides a set of recommended naming conventions for scalar variables used within the module code. Examples of using these naming conventions include:

- REARTH - the radius of the earth (a range variable)
- DBT1 - the absolute value (distance) of the center of mass of missile B with respect to the target T1
- SBEL1 - the displacement vector of the center of mass of the missile B, with respect to the earth reference point E, in the Local Coordinate System, L; also the first directional component of the SBEL(3) vector

Table 2. Programming Names for Scalars.

Convention	Recommended Use	Convention	Recommended Use
Axxxxx	Acceleration variables	xxxLIM	Limiting Values
AKxxxx	Constants	Mxxxxx	Mode Indicators
BIASxx	Bias	Nxxxxx	Counters
Cxxxxx	Aero Coefficients	PHIxxx	Third Rotation Angles
DUMxxx	Dummy Variables	PSIxxx	First Rotation Angles
DTIMxx	Time Differences	Rxxxxx	Ranges
Exxxxx	Error Values	RANDxx	Random Noise
EPCHxx	Epoch	Txxxxx	Time Constants
Fxxxxx	Forces	THTxxx	Second Rotation Angles
FACTxx	Multiplying Factors	TIMExx	Time Clock
FLGxxx	Flags	Uxxxxx	Updates
FMxxxx	Moment Variables	xxxxxM	Stored Values (Minus)
FRAxxx	Fractional Increases	xxxxxP	New Values (Plus)
Gxxxxx	Gains	xxxxx1	First Vector Components
Hxxxxx	Heights	xxxxx2	Second Vector Components
ISETxx	Reset Switches	xxxxx3	Third Vector Components
Lxxxxx	Logic Variables		

Table 3 provides a set of recommended naming practices for vector and matrix variables. Examples of vector and matrix names using the recommended conventions include:

- SBEL(3) -vector with 3 elements; the displacement vector of the center of mass of the missile B, with respect to the earth reference point E, in the Local Coordinate System, L
- WBEB(3)-vector with 3 elements; the angular velocity vector of the body frame B, with respect to the earth frame E, in the Body Axes Coordinate System, B
- TBL(3,3) -3 x 3 matrix; a coordinate transformation matrix of body coordinate axes B, with respect to the Local Coordinate System, L

Table 3. Sample Programming Names for Vectors and Matrices.

VARIABLE TYPE	SUB & SUPERSCRIPT	COORDINATE SYSTEMS	DIRECTION
1st Letter	Followed by	Followed by	Optional
	POINTS:	COORDINATE SYSTEM:	
A - Linear Acceleration	A - Sensor Aimpoint	A - Air Mass	1
D - Absolute Value	B - Missile Center of Mass	B - Body Fixed	
E - Error Perturbation	C - Earth-Fixed Reference	E - Earth	2
R - Rotation	E - Start of Final Phase	G - Geographic	
S - Displacement	F - Start of Final Phase	I - Inertial	3
T - Transformation	I - INS Stored Reference Point	L - Local: N, E, Down	
U - Update, Unit Vector	L - Launch Point	N - Navigation	
V - Linear Velocity	N - Start Navigation Line	O - Line of Sight	
W - Angular Velocity	P - Predicted Intercept Point	S - Seeker	
	Q - Seeker Aimpoint	T1 - Target	
	S - Intercept LOS and Target	T2 - Launch Aircraft	
	T1 - Target	U - Geographic Velocity	
	T2 - Launch Aircraft	V - Velocity of Air	
		W - Wind	
	FRAMES:	H	
	A - Air Mass	Q	
	B - Missile Rigid Body	Y	
	E - Earth	Z	
	I - Inertial	MODIFIER:	
	O - Line-of-Seeker	C - Computed	
	S - Seeker Inner Gimbal	D - Derivative	
	U - Geographic Flight Path	K - Seeker	
	V - Inertial Flight Path	P - Predicted	
		R - Recorded	
		X - Non-standard Units	
		M - Stored in Memory	

2.2.3 Coding Conventions

Several conventions are recommended in the coding of the CADAC modules to remain consistent with the modular design of the program. These conventions also provide compatibility with several of the utility programs available with CADAC. Figure 2 displays a sample initialization module. The initialization modules include initializing the state variables used in the main modules. Coding recommendations for state variable initializations are displayed in Figure 3. Figure 4 displays the associated module code. These coding conventions are:

- Begin every SUBROUTINE with a brief statement of its purpose, the name of the programmer with phone number and the date of the completed check-out.
- Document all variables and group them into the following classifications: INPUT DATA, INITIALIZATION, INPUT FROM OTHER MODULES, OUTPUT TO OTHER MODULES, STATE VARIABLES and DIAGNOSTICS.
- Use the EQUIVALENCE statements to define module variables and constants and place the EQUIVALENCE statements into the proper block format. These statements are grouped into blocks corresponding to the variable classifications. These blocks are composed of the EQUIVALENCE statements and documentation for the equivalenced variables. Examples of the EQUIVALENCE blocks can be seen in Figures 2 and 3.

```
      SUBROUTINE C4I
C*************************************************************
C***  *   Actuator Initialization Module
C***  *
C***  *   This subroutine performs the following functions:
C***  *     (1) Initializes the state variables
C***  *
C***  *   MODIFICATION HISTORY
C***  *     940819 Created by Peter Zipfel
C***  *
C*************************************************************
C
      COMMON C(3510)
C
      DIMENSION IPL(100),IPLV(100)
C
C***      INPUT FROM EXECUTIVE
C
      EQUIVALENCE (C(2561),NIP)
      EQUIVALENCE (C(2562),IPL(1))
      EQUIVALENCE (C(2867),IPLV(1))
C
C NIP = E The number of variables being integrated
C IPL(100) = E The location of the state variable derivative
C IPLV(100) = E The location of the sate variable
C
C***      STORAGE OF STATE VARIABLE LOCATIONS
C
      IPL(NIP)=1110
      IPLV(NIP)=1111
      NIP=NIP+1
      IPL(NIP)=1112
      IPLV(NIP)=1113
      NIP=NIP+1
      IPL(NIP=1114
      IPLV(NIP)=1115
      NIP=NIP+1
      IPL(NIP)=1116
      IPLV(NIP)=1117
      NIP=NIP+1
C
      RETURN
      END
```

Figure 2. Sample Initialization Module.

```
Example 1, Single scalar Integration:

       IPL(NIP) = 1610  <= C-location
       IPLV(NIP) = 1611 <= C-location
       NIP=NIP+1

Example 2, Multiple Scalar Integration (6 integration variables):

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

Example 3 State Vector Integration (3 components):

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

       NOTE:  The IPLV line could be omitted

Example 4, Alternate 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

Example 5, Legacy Single State Integration:

       J = 1610  <= C-location
       IPL(NIP) = J
       IPLV(NIP) = J + 3
       NIP=NIP+1

       NOTE:  The IPLV line could be omitted

Example 6, Legacy 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
```

Figure 3. State Variable Initialization Code Conventions.

```
      SUBROUTINE C4
C*************************************************************
C***  *   Actuator Module
C***  *
C***  *   This subroutine performs the following functions:
C***  *
C***  *   (1) Converts from control deflections to fin deflections
C***  *   (2) Calls actuator dynamic subroutine
C***  *             MACT   =      0 No dynamics
C***  *                    =      1 First order dynamics
C***  *   (3) Limits fins excursions and converts back to
C***  *             control deflections
C***  *
C***  *   MODIFICATION HISTORY
C***  *
C***  *   940819 Created by Peter Zipfel
C***  *********************************************************
C
C***   COMMON LOCATIONS ARE 1100-1199
C
      COMMON C(3510)
C
C***   INPUT DATA
C
      EQUIVALENCE (C(1100),MACT)
      EQUIVALENCE (C(1102),DLIMX)
C
C MACT = D Actuator flag =0:no dynamics, =1:first order
C DLIMX = D Control flap limiter - deg
C
C***   INPUT FROM OTHER MODULES
C
      EQUIVALENCE (C(0919),DPCX)
      EQUIVALENCE (C(0920),DQCX)
      EQUIVALENCE (C(0921),DRCX)
C
C DPCX = O Commanded roll control deflection - deg
C DQCX = O Commanded pitch control deflection - deg
C DRCX = O Commanded yaw control deflection - deg
C
C***   OUTPUT TO OTHER MODULES
C
      EQUIVALENCE (C(1119),DPX)
      EQUIVALENCE (C(1120),DQX)
      EQUIVALENCE (C(1121),DRX)
C
C DPX = O Roll control deflection – deg
C DQX = O Pitch control deflection - deg
C DRX = O Yaw control deflection - deg
C
C***   DIAGNOSTICS
```

Figure 4. Sample Module Code.

```
      EQUIVALENCE (C(1122),D1X)
      EQUIVALENCE (C(1123),D2X)
      EQUIVALENCE (C(1124),D3X)
      EQUIVALENCE (C(1125),D4X)
C
C D1X = G Fin #1 defl subject to mech limit DLIMX - deg
C D2X = G Fin #2 defl subject to mech limit DLIMX - deg
C D3X = G Fin #3 defl subject to mech limit DLIMX - deg
C D4X = G Fin #4 defl subject to mech limit DLIMX - deg
C
C***   CONVERSION TO FOUR FIN DEFLECTIONS
C
      D1CX=-DPCX+DQCX-DRCX
      D2CX=-DPCX+DQCX+DRCX
      D3CX=+DPCX+DQCX-DRCX
      D4CX=+DPCX+DQCX+DRCX
C
C***   NO ACTUATOR DYNAMICS
C
      IF(MACT.EQ.0.) THEN
            D1X=D1CX
            D2X=D2CX
            D3X=D3CX
            D4X=D4CX
      ENDIF
C
C***   CALL FIRST ORDER ACTUATOR
C
      IF(MACT.EQ.1)CALL C4FIRST(D1X,D2X,D3X,D4X,D1CX,D2CX,D3CX,D4CX)
C
C***   LIMITERS
C
      IF(ABS(D1X).GE.DLIMX) D1X=SIGN(DLIMX,D1X)
      IF(ABS(D2X).GE.DLIMX) D2X=SIGN(DLIMX,D2X)
      IF(ABS(D3X).GE.DLIMX) D3X=SIGN(DLIMX,D3X)
      IF(ABS(D4X).GE.DLIMX) D4X=SIGN(DLIMX,D4X)
C
C***   CONVERSION TO CONTROL DEFLECTIONS
C
      DPX=(-D1X-D2X+D3X+D4X)/4.
      DQX=(+D1X+D2X+D3X+D4X)/4.
      DRX=(-D1X+D2X-D3X+D4X)/4.
C
      RETURN
      END
```

Figure 4. Sample Module Code (Continued).

```
C***************************************************************
      SUBROUTINE C4FIRST(D1X,D2X,D3X,D4X,D1CX,D2CX,D3CX,D4CX)
C***************************************************************
C***  * First Order Actuator Model
C***  *
C***  * This subroutine performs the following functions:
C***  *   (1) Models first order lags of all four fins
C***  *
C***  * Argument Input:
C***  *
C***  * Argument Output:
C***  *          D1X,D2X,D3X,D4X=Four actuator fin deflections
C***  *
C***  *          D1CX,D2CX,D3CX,D4CX=Four actuator fin commands
C***  *
C***  * MODIFICATION HISTORY
C***  *   940819 Created by Peter Zipfel
C***  ***************************************************************
C
      COMMON C(3510)
C
C*** INPUT DATA
C
       EQUIVALENCE (C(1103),TACT)
C
C TACT = D Actuator time constant - 1/s
C
C*** STATE VARIABLES
C
      EQUIVALENCE (C(1110),D1D)
      EQUIVALENCE (C(1111),D1)
      EQUIVALENCE (C(1112),D2D)
      EQUIVALENCE (C(1113),D2)
      EQUIVALENCE (C(1114),D3D)
      EQUIVALENCE (C(1115),D3)
      EQUIVALENCE (C(1116),D4D)
      EQUIVALENCE (C(1117),D4)
C
C D1 = S Fin #1 deflection - deg
C D2 = S Fin #2 deflection - deg
C D3 = S Fin #3 deflection - deg
C D4 = S Fin #4 deflection - deg
C
C*** FIRST ORDER LAG
C
      D1D=(D1CX-D1)/TACT
      D2D=(D2CX-D2)/TACT
      D3D=(D3CX-D3)/TACT
      D4D=(D4CX-D4)/TACT
C
      D1X=D1
      D2X=D2
      D3X=D3
      D4X=D4
C
      RETURN
      END
```

Figure 4. Sample Module Code (Concluded).

- Format the EQUIVALENCE blocks in the following manner for compatibility with CADAC utility programs:
  - Dimension all vectors and matrices named in EQUIVALENCE statements in a single dimension statement prior to the first EQUIVALENCE block.
  - Precede each EQUIVALENCE block with a comment indicating the classification. The comment should have the following format:

```
C
C***  OUTPUT TO OTHER MODULES
C
```

  - Consecutively list all EQUIVALENCE statements within a block, preferably on separate lines. Do NOT separate EQUIVALENCE statements with comment lines.
  - Spacing is critical within the EQUIVALENCE statement. Extra spaces are permitted only where the arrows indicate:

```
EQUIVALENCE  (C(  1340),  MYVAR  )
                  ^    ^        ^           ^
```

    No extra spacing can exist between the first open parenthesis and the name of the C array.
  - Follow the EQUIVALENCE statements by comment lines documenting each variable within the EQUIVALENCE block. Include at least one comment line after the last EQUIVALENCE statement, before the variable documentation. Do NOT separate the documentation lines with comment lines.
  - Only one documentation line per variable is permitted. On the documentation line, include the variable name followed by " = " with blank spaces before and after the equal sign. Then include the letter code corresponding to the classification: D (input Data), I (Initialization), O (Output to other modules), S (State variable) and G (diaGnostic). Follow with the variable definition and units.
- In sub-level modules, the argument list input and output variables are documented with their units.
- Use plenty of comments within the code.
- IF-THEN blocks are indented by three blanks, nested consecutively.
- Avoid the use of GOTO statements.
- Standard FORTRAN 77 conventions are used for naming REAL, INTEGER and LOGICAL variables. Do not use the IMPLICIT NONE statement.

2.2.4 Executive and Module Communication

The method of communication between all of the modules (Executive and user-supplied) is through an array, C(3510), located in a blank common. Only in special cases are labeled common communications permitted between modules. The C array is divided into blocks of 100 variable locations and a block of locations is reserved for each module. Table 4 shows the

common location assignment for each module. All of the variables utilized in a module are then equivalenced to locations within the assigned C-array block numbers. Duplication of equivalence variable NAMES is not permitted.

Table 4. Module Names and Associated Common Locations.

MODULE	MODULE NAME	RESERVED COMMON LOCATIONS	MODULE NUMBER
A1	Aero Coefficients	1200 - 1299	2
A2	Propulsion	1300 - 1399	3
A3	Forces & Moments	1400 - 1499	4
A4	Growth	1500 - 1599	5
C1	Guidance	800 - 899	7
C2	Auto pilot	900 - 999	8
C3	TVC	1000 - 1099	9
C4	Actuator	1100 - 1199	10
D1	Newton Eqs.	1600 - 1699	17
D2	Euler Eqs.	1700 - 1749	18
D3	SWEEP	1800 - 1899	19
D4	Growth	1900 - 1999	20
G1	Target	100-199	22
G2	Air Data	200-299	23
G3	Kinematics	300-399	24
G4	Terminal	1750-1799	25
S1	Seeker Model	400-499	28
S2	Sensor Model	500-599	29
S3	NAV Filter	600-699	30
S4	INS Model	700-799	31
CADAC EXECUTIVE	Controlling Methodology	1 - 99, 1772, 2000 - 2999	N/A
SWEEP MODULES	Sweep Methodology	1800 - 1824	N/A
Unassigned Locations		3000 - 3510	N/A

Locations 1 - 100 and 2000 - 3510 in the C array are reserved for use by the CADAC Executive. The user should **not** re-define these locations within the modules. However, the user may access these locations with the CADAC Executive definition. Table 5 gives the C locations and associated CADAC definitions; unassigned locations are available for use by the user.

Due to the maturity of the CADAC environment, several pre-written module codes exist and are commonly used. This usage has lead to the integration of several module-defined variables into the CADAC Executive. These variables and their corresponding C locations are:

TRCOND C(1772) Terminate Condition Code set by many G4 modules and output by the Executive.

These C locations should not be re-defined by any module code due the utilization of these variable definitions by the Executive.

The equivalence variables are divided into the following classifications: INPUT DATA, INITIALIZATION, INPUT FROM OTHER MODULES, OUTPUT TO OTHER MODULES,

STATE VARIABLES and DIAGNOSTICS. Preferably, the variables are arranged within the code under comments bearing the category name, in numerical order, as seen in Figures 2 and 3.

Table 5. Executive Definitions for C Locations.

0001	ERRVAL	The maximum integration step error value
0002	ERRN	IPL location of variable causing ERRVAL
0003	AERR	C location of variable causing ERRVAL
0004	PRELOC	
0051	REARTH	Radius of the Earth = 6370987 meters
0052	CRAD	Conversion factor = 57.29577951 (Degrees/Radians)
0053	OPTMET	Units of measure 1 = metric 0 = English (Default)
0054	AGRAV	Acceleration due to gravity @ sea level =9.8066 meters/s**2
0055	CFTM	Conversion factor = 0.3048006 (Meters/Feet)
0056	CKFPS	Conversion factor = 1.6878 (Knots / (ft/s))
0057	AMU	Gravitational parameter = 3.986005E+14 (meters**3 / sec**2)
0058	WEII3	Earth Angular rotation = 7.2921152E-5 (rad/sec)
0059	OPNORO	Flag: 0 = rotating earth model: WEII = 7.2921154E-05 1 = Non-rotating earth model: WEII3 = 0.0
0090	RANSEED	Random function generator initialization
1772	TRCOND	Terminate Condition Codes from right to left.
1800	ISWEEP	Sweep option flag (0 through 5)
1801	CRITNO	Critical variable C location
1802	CRITVAL	Minimum test for critical variable
1803	SEARNO	Number of binary searches runs (opt 4, 5)
1804	NUMR	The number of trajectory runs
1805	CRITMAX	The maximum test for critical variable
1811	ANGLNO	The C location of the angular variable
1812	ANGMIN	The minimum angle value
1813	ANGMAX	The maximum angle value
1814	ANGDEL	The delta angle
1815	ANGUNT	The units of the input data: radians or degrees
1821	RANGNO	The C location of the range variable
1822	RANMIN	The minimum range value
1823	RANMAX	The maximum range value
1824	RANDEL	The delta range value
1837	ANGX	The polar angle from target
1838	RANG	The range (distance) from target
2000	TIME	Trajectory time - s.

Table 5. Executive Definitions for C Locations (Concluded).

2001	TSTAGE	Time in current stage - s.
2003	PCNT	Time of the next print to TABOUT.ASC
2004	PPNT	Time of next print to the plot files
2005	PPP	Time interval writing to TRAJ.BIN or TRAJ.ASC - s.
2006	ITAP90	Flag: 0 = No CSAVE.ASC 1 = trajectory started from data saved to CSAVE.ASC (Used by D3I)
2011	KSTEP	Controls flow after an integration step
2014	ITCNT	Print flag counter
2015	CPP	Time interval writing to screen or TABOUT data - s.
2016	PGCNT	Page counter flag
2020	LCONV	Flag indicating end of trajectory run: 0 = start of trajectory ≥2 = stop trajectory calculations
2127-2196	PMIN(70)	Array of minimum plot variable values
2280	NV	The number of variables in the plot list
2285	NJ	Number of state variables to be integrated
2361	NOMOD	The number of modules to be called.
2362-2460	XMODNO(99)	The list of the module numbers to be called by the Executive, in the calling order.
2461	NOSUB	The number of the output and auxiliary subroutines to be called by the executive.
2462-2560	SUBNO(99)	The list of output and auxiliary subroutine numbers to be called by the Executive.
2561	NIP	The number of variables being integrated.
2562-2661	IPL(100)	The locations of the derivative of the state variable.
2662	HMIN	Not used
2663	HMAX	Not used
2664-2764	DER(101)	DER(1) = The integration interval.
2765-2865	V(101)	
2866	ICOOR	Flag: -2 = Executive is in module initialization mode -1 = Executive is in trajectory initialization mode 0 = Integration predictor cycle 1 = Integration corrector cycle.
2867-2967	IPLV(100)	The location of the state variable; corresponding to the derivative in the IPLV array

Communication between modules and the associated sub-level modules is through the argument list; these variables should be divided into INPUT and OUTPUT categories and so documented within the code. Variables of the categories INPUT DATA, INPUT FROM OTHER MODULES, STATE VARIABLES and DIAGNOSTICS may be passed through the argument list to sub-level modules, but not variables of the INITIALIZATION and OUTPUT categories.

2.2.5 Initialization of Variables and Constants

At the start of each trajectory, the C array is initialized to 0.0. The executive provides a set of pre-defined constant definitions within the C array; the executive then re-loads these

constants in preparation for the new trajectory. The constants provided by the executive are: REARTH (0051), CRAD (0052), OPTMET (0053), AGRAV (0054), CFTM (0055), CKFPS (0056), AMU (0057), WEII3 (0058) and OPNORO (0059). The default units for these variables are the units in the English system. By properly setting the value of the OPTMET variable in the input file, INPUT.ASC, the executive will convert these constants to the metric unit system.

After the C array is reset and the constants are re-initialized, the initial stage of the input data file, INPUT, is processed and then the initialization modules are executed. If initial values other than 0.0 are desired for variables, these variables must be initialized either in the INPUT file, or by assignment statements in the initialization modules. Constants used in the simulation may be initialized either in the INPUT file or by assignment or PARAMETER statements in the initialization modules. DATA statements should only be used for initializing tables and the values in these tables should never be modified by the module code. Every variable should have an initial value assigned. Do not rely on the computer initializing variables (i.e. variables not assigned a C location) to 0.0.

2.2.6 The G4 Module

The G4 module does not follow the typical module definitions. The function of this module is the detection of the end of the simulation. When the end of the simulation is detected, the user must assign the LCONV (2020) variable to have a value greater than or equal to 2. This variable signals the Executive when the end of the trajectory simulation has been reached. The CADAC Executive calls this module only once at the end of each integration time step.

2.2.7 Defining State Variables and Vectors

Since the CADAC Executive module performs the state variable integration and the state variables are defined in the user-written module code, information on these variables must be communicated between these code sets. The IPLV, IPL and NIP variables are used to communicate information on state variables between the module and Executive code. The IPLV array is a list of the C locations that are defined as state variables. The IPL array is a list of the C locations that are defined as the derivatives of the state variable. There must be a one to one correspondence between the IPLV and IPL arrays; IPLV(i) contains the C location of the state variable whose derivative is located in the C location stored in IPL(i). The NIP variable is the maximum number entries in the IPLV and IPL arrays. These arrays are loaded in the initialization module associated with the module where the state variables are defined.

Figures 5 and 6 show an example of initializing a scalar state variable and a three-element vector state variable. Figure 5 shows the initialization module and Figure 6 shows the corresponding main module. The C locations are defined within the EQUIVALENCE statements of the main module. The scalar state variable, PHIT1L, is defined at C location 0111 and the corresponding derivative of this variable is defined at C location 0110, PHIT1LD. A three element vector state variable (0119-0121) and corresponding derivative (0116-0118) are also defined in this module. If the IPL and IPLV arrays are not loaded with these C locations, then the CADAC Executive will not integrate these variables. The loading of the IPLV and IPL arrays is performed in the initialization module corresponding to the module defining the variables. The example shows the scalar state variable being loaded into the IPL and IPLV arrays

first. The IPLV array is loaded with the location of the state variable and the IPL array is loaded with the location of the state variable derivative. Since one state variable has been added to the list, the NIP variable is incremented by 1. The NIP variable is properly initialized by the CADAC Executive and must be incremented after each entry to the IPL/IPLV lists. The example then shows the addition of the vector state variable to the lists. The local variable, ILOC, is used to point to the C location of the first element of the vector state variable.

```
      SUBROUTINE G1I
C*****************************************************************
C***   *      Target Initialization Module
C***   *
C***   *      This subroutine performs the following functions:
C***   *        (1) Identifies the state variables for integration
C**    *        (2) Initializes the target and shooter velocity vectors
C***   *
C***   *      MODIFICATION HISTORY
C***   *      941129 Created by Peter Zipfel
C***   ***************************************************************
C
C***   COMMON LOCATIONS ARE 100-199
      COMMON C(3510)
C
      DIMENSION IPL(100),IPLV(100),VT1EL(3),VT2EL(3)
C
C***   INPUT FROM EXECUTIVE
C
      EQUIVALENCE (C(2561),NIP)
      EQUIVALENCE (C(2562),IPL(1))
      EQUIVALENCE (C(2867),IPLV(1))
C
C NIP = E The number of variables being integrated
C IPL(100) = E The location of the state variable derivative
C IPLV(100) = E The location of the sate variable
C
C***   STORAGE OF SCALAR STATE VARIABLE LOCATIONS
C
      IPL(NIP)=110
      IPLV(NIP)=111
      NIP=NIP+1
      IPL(NIP)=112
      IPLV(NIP)=113
      NIP=NIP+1
      IPL(NIP)=114
      IPLV(NIP)=115
      NIP=NIP+1
C
C***   STORAGE OF VECTOR STATE VARIABLE LOCATIONS
C
      ILOC=116
      DO I=0,2
            IPL(NIP)=ILOC+I
            IPLV(NIP)=ILOC+I+3
            NIP=NIP+1
      ENDDO
C
      ILOC=122
      DO I=0,2
            IPL(NIP)=ILOC+I
            IPLV(NIP)=ILOC+I+3
            NIP=NIP+1
      ENDDO
C
      RETURN
      END
```

Figure 5. Defining State Variable Sample (Initialization Module).

```
      SUBROUTINE G1
C*****************************************************************
C***   *      Target Module
C***   *
C***   *      This subroutine performs the following functions:
C***   *   (1) Simulates target trajectories
C***   *
C***   *      MODIFICATION HISTORY
C***   *      941129 Created by Peter Zipfel
C***   ***************************************************************
C
C***   COMMON LOCATIONS ARE 100-199
C
      COMMON C(3510)
C
      DIMENSION  VT1ELD(3),VT1EL(3),ST1ELD(3),ST1EL(3)
C
C***   INPUT DATA
C


C
C***   STATE VARIABLES
C
      EQUIVALENCE (C(0110),PHIT1LD)
      EQUIVALENCE (C(0111),PHIT1L)
      EQUIVALENCE (C(0112),AX1D)
      EQUIVALENCE (C(0113),AX1)
      EQUIVALENCE (C(0114),AN1D)
      EQUIVALENCE (C(0115),AN1)
      EQUIVALENCE (C(0116),VT1ELD(1))
      EQUIVALENCE (C(0119),VT1EL(1))
      EQUIVALENCE (C(0122),ST1ELD(1))
      EQUIVALENCE (C(0125),ST1EL(1))
C
C PHIT1L = S Bank angle of normal load factor plane of T1 - rad
C AX1 = S Acceleration along the target velocity vector of T1 - g's
C AN1 = S Normal load factor (normal to velocity vector) of T1 - g's
C VT1EL(3) = S Velocity of T1 wrt earth in local level coord - m/s
C ST1EL(3) = S Position of T1 wrt earth ref point E in L coor - m
C
C***   OUTPUT TO OTHER MODULES
C


C
C***   CODE DEFINING THE DERIVATIVES
C


C
RETURN
END
```

Figure 6. Defining State Variable Sample (Main Module).

3. CADAC DATA FILES

Table 6 provides a list of all the input and available output files. The CADAC simulation requires two data files as input. Up to 12 data files may be generated by a CADAC simulation as output. Up to 8 of these files, the IMPACT.ASC, IMPACT7.ASC, IMPACT10.ASC, CSAVE.ASC, INIT.ASC(BIN) and TRAK.ASC(BIN) files, are generated only when special execution options are selected by the user. More information on these files and the options required to generate them is available in ***CADAC Program Documentation, Vol III***.

Table 6. CADAC Filenames and Definitions.

FILENAME	FILE USAGE	FILE DEFINITION	UNIT NUMBER
CADIN.ASC	INPUT	Lead Card Input Deck	35
HEAD.ASC	INPUT	Variable Definitions	3
TABOUT.ASC	OUTPUT	Tabular Output Data	6
TRAJ.BIN	OUTPUT	Trajectory Plot Data in Binary Format	11
TRAJ.ASC	OUTPUT	Trajectory Plot data in ASCII Format	12
STAT.BIN	OUTPUT	Statistical Plot Data in Binary Format	44
STAT.ASC	OUTPUT	Statistical Plot Data in ASCII Format	45
IMPACT.ASC	OUTPUT	Sweep Plot Data	22
CSAVE.ASC	IN/OUT	Saved State Data	90
IMPACT10.ASC	IN/OUT	Scratch; Saved End Data	10
IMPACT7.ASC	IN/OUT	Scratch; Sweep Plot Data	7
RANVAR.ASC	OUTPUT	Stochastic Variable Assignment Values	37
INIT.BIN	OUTPUT	Real-time CADAC initialization file for target track data in Binary format	51
INIT.ASC	OUTPUT	Real-time CADAC initialization file for target track data in ASCII format	50
TRAK.BIN	OUTPUT	Real-time CADAC target track time history data in Binary format	61
TRAK.ASC	OUTPUT	Real-time CADAC target track time history data in ASCII format	60

3.1 Input Files

The CADAC simulation requires two input files for execution. These files are referred to in this document by the names: INPUT and HEAD.

~~1.1.1~~ 3.1.1 INPUT.ASC

Formatted: Bullets and Numbering

The INPUT file provides information on the flight scenario for the simulation. This free-formatted file provides maximum flexibility in the simulation applications. By manipulating this single file, the user may modify the flight scenario without modifying the actual simulation code. 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 file must be contained in the current HEAD.ASC file. Table 7 summarizes the statements found

in an INPUT.ASC file. For a complete description of the statements found in an INPUT.ASC, refer to ***CADAC Program Documentation, Vol III***.

The statements in the INPUT.ASC file are grouped into a specific structure. Figure 7 displays a sample INPUT file generating a single trajectory. By using various statement combinations, a single INPUT file can generate multiple trajectory simulations.

Table 7. 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
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

Table 7. INPUT.ASC Statement Summary (Concluded).

STATEMENT	DEFINITION
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-Runs see Vol III 3.1.1.4 for example
MONTE NumOfRuns	Indicates the number of trajectories to be made
RANDOM(Seed,RandomOption)	assigns the initial seed value to the variable RANSEED and sets the method to use for setting the random seed, see Vol III 3.1.2.3.6 for example
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
SWEEP MODE ModeType LIMIT CritVal< CritVar 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 Vol III 3.1.2.4.3 for details
WEATHER	the WEATHER block allows the user to input weather data for the atmospheric modules, see Vol III 3.1.2.4.5 for details

The first non-comment statement in the INPUT file is the TITLE command. This statement may contain any alphanumeric title the user desires. Following the TITLE statement is the block of module definitions dictating the auxiliary modules to be used in the simulation. In the module block, the names of the modules are indicated by two characters. When the modules (and initialization modules) are executed, only those listed in the module block are executed, and they are executed in the order listed in the block. However, the G4 module is an exception; it is not listed in the module block and will always be executed by CADAC.

The remaining statements in the INPUT file are divided into "stages". Many statement types may be included in a stage, the last statement in a stage is typically an IF statement which defines a stage criteria that must be satisfied for stage completion. The last stage in the trajectory simulation is terminated by the RUN statement. In the sample file, there is only one stage, the initialization stage, which is terminated by the RUN statement. Processing of the INPUT file occurs on a stage by stage basis. All of the statements in a stage are processed and the trajectory simulation is executed until the criteria dictated by the IF statement is satisfied. Then the next stage(s) are processed in the same manner until the end of the trajectory is reached. The statement types, definitions and examples of their usage may be found in ***CADAC Program Documentation, Vol III***. Prior to executing CADAC, the free-formatted INPUT file should be converted to a CADIN.ASC file by executing the CADAC utility program CADIN. CADIN.ASC has the fixed-format that CADAC expects.

```
TITLE CADAC Single Trajectory Input File
!
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 7. Sample INPUT File for a Single Trajectory.

3.1.1.1 Staging Procedures

Staging conditions are inserted into the INPUT.ASC file to simulate mode changes of the vehicle, like staging of rockets, mass property and aerodynamic changes, guidance mode switching, control command modulation and end-of-run conditions. The first set of input data is identified as the 0th Stage. They initialize the run. A staging criteria is introduced by an "IF" statement. Once it becomes true the input data for the next stage are read into the program. The stage after the first IF statement is labeled as the 1st Stage and so on. Figure 8 shows an example. After reading the initial conditions, the first Stage is initiated at 200 sec to change the angle-of-attack command. At 400 sec the throttle setting of the engine is changed (Stage 2) and once the fuel is exhausted the simulation is terminated. For details on multiple staging criteria see Section 3.1.1.2 in ***CADAC Program Documentation, Vol III***.

```
TITLE SSTO Launched from B747 above Cape Canaveral
MODULES
 G2 ENVIRONMENT
 A2 PROPULSION
 A1 AERODYNAMICS
 A3 FORCES
 D1 NEWTONS LAW
END
OPTMET =  1.0       ! E =0: English units, =1: SI units
OPNORO = 0.0        ! E =0: Rotating earth, =1: Non-rotating earth
! *** Control Input ***
ALPHA  =  .4        ! D Angle of attack - rad
! *** Propulsion ***
MPROP = INT(1)      ! D =0: Motor off, =1:Motor on
SPI  = 495.4        ! D Specific impulse - 1/s
THRSL  = 1.51E6     ! D Rocket thrust at sea level - N
THROTL = .9         ! D Throttle setting ( 0 - 1 )
        .
        .
        .
! *** Printing and Plotting ***
PPP = 1             ! E Plotting interval for data written to TRAJ - s
CPP = 10            ! E Display interval on screen or written to TABOUT - s
DER = .1            ! E Integration interval - s
IF TIME > 200       !### Stage 1: Change Alpha Command ###
  ALPHA = .1        ! D Angle of attack - rad
IF TIME > 400       !### Stage 2: Reduce Throttle ###
  THROTL = .5       ! D Throttle setting ( 0 - 1 )
IF FMASS > 156194   !### End-of-Run: Motor Burn-Out ###
RUN
STOP
```

Figure 8. Sample INPUT File Showing Staging Procedures.

3.1.2 HEAD.ASC

The HEAD.ASC file contains information for the output requirements for the simulation. By modifying the contents of this file, the user may modify the contents of the output generated by the CADAC simulation without modifying the simulation code. The HEAD 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 has a distinctive format and must occur on the file in the order listed. A brief discussion on the format of each section follows. Figure 9 shows a sample HEAD input file.

```
      SCROLL ECHOIN INTMSG TRAJBIN TRAJASC STATBIN NOSTATASC TABOUT STGMSG  RANVAR
  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 HEADER file for the report only
*
*  0001        ERRVAL
*I 0002        IERR
*  0003        AERR
    0051       REARTH
    0052       CRAD
    0115       ST1EL1  Position of the T1 wrt E
    0117       ST1EL3  Position of the T1 wrt E
  I 0200       MAIR    Atmospheric selection Flag
  I 0400       MSEEK   Seeker Option Flag
  I 0401       MS1DYN  Seeker Selection: Kine/Dynam
    0407       RACQ    Acquisition Range
      •
      •
      •
    0902       THTVLC  Pitch flight path angle command
    0904       HCOM    Altitude Command
    0907       PSIVLC  Yaw Flight path angle 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
```

Figure 9. Sample HEAD.ASC File.

3.1.2.1 HEAD Option List

The option list in the HEAD file is a single line at the top of the file. On this line, the user enters up to fourteen keywords indicating output options available in program CADAC. The keywords are not column dependent; they may be entered anywhere on the 100 column line. For clarity, the keywords should be separated by at least one space. Each keyword consists of two toggle phrases; one phrase that activates and one that suppresses the function. The valid keywords are:

SCROLL/NOSCROLL	-	controls the display of the scroll variable data to the TABOUT file.
ECHOIN/NOECHOIN	-	controls the display of the input card data on the TABOUT file.
INTMSG/NOINTMSG	-	controls the display of integration messages generated by the AMRK module to the TABOUT file.
STGMSG/NOSTGMSG	-	controls the display of staging messages to the TABOUT file. The staging message occurs when a stage criteria is satisfied and indicates the stage number, criteria number that was satisfied, stage variable C location and the time of the stage.
TRAJBIN/NOTRAJBIN	-	controls the creation of the binary trajectory plot data file
TRAJASC/NOTRAJASC	-	controls the creation of the ASCII trajectory plot data file
STATBIN/NOSTATBIN	-	controls the creation of the binary statistical plot data file
STATASC/NOSTATASC	-	controls the creation of the ASCII statistical plot data file
TABOUT/NOTABOUT	-	controls the direction of the tabular output; if TABOUT is included in the option list, the output is directed to the file TABOUT.ASC; if NOTABOUT is included, the output is directed to the screen
RANVAR/NORANVAR	-	controls the creation of the RANVAR output file which contains the values selected by the random distribution generators for stochastic variables defined in INPUT.
INITASC/NOINITASC	-	controls the creation of the ASCII target track initialization data file
INITBIN/NOINITBIN	-	controls the creation of the binary target track initialization data file
TRACKASC/NOTRACKASC	-	controls the creation of the ASCII target track time history data file
TRACKBIN/NOTRACKBIN	-	controls the creation of the binary target track time history data file

SWEEP/NOSWEEP — controls whether the current execution is a SWEEP run or a single mode run.

3.1.2.2 HEAD Scroll Variable List

The HEAD scroll variable list is the list of variables whose data values are to be printed to the TABOUT file/screen at the user-selected print rate. The scroll variable list may contain up to 16 records. Each record represents one variable whose data is to be displayed to the TABOUT file/screen. The format of these records is given in Table 8.

Table 8. 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 first character in the record is used to flag the end of the scroll list. The scroll list must be ended by 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 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 the TABOUT file. The user may enter up to 8 characters.

3.1.2.3 HEAD Comment List

The comment list follows the scroll variable list on the HEAD file. The comment list is for the user's benefit only; the data entered on these records is 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 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.

3.1.2.4 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 target track time history data files. These records follow the format shown in Table 9.

Table 9. 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

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.

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.

3.2 Output Files

The CADAC output files discussed in this section are those most commonly used by the programmer. A brief synopsis on the file contents and usage is presented. More detailed information on these files can be found in ***CADAC Program Documentation, Vol III***.

~~1.1.1~~ 3.2.1 TABOUT

Formatted: Bullets and Numbering

The general output produced by the CADAC simulation may be either scrolled interactively to the screen as the program executes, or may be written to a file with the file name TABOUT.ASC. The user indicates the direction of this data by using the keywords TABOUT (file output) or NOTABOUT (screen output) in the HEAD file option list. The TABOUT.ASC file is an ASCII text file suitable for printing. The exact contents of this file are dependent on user selected options. These options are selected through the HEAD input file. This file can contain a copy of the CADIN file used to execute the simulation and data at a user-specified print rate for up to 16 C array locations.

~~1.1.2~~ 3.2.2 TRAJ

Formatted: Bullets and Numbering

TRAJ is a file that contains data for up to 70 C locations, printed at a user-defined print rate. The file option list in the HEAD.ASC file determines the creation of this file and the format. If a binary TRAJ file is to be created, the keyword TRAJBIN should be included in the option list. If the user omits the keyword TRAJBIN from the HEAD.ASC option list or includes the keyword NOTRAJBIN in the HEAD.ASC option list, the binary TRAJ file is not created. In addition, the user can indicate an ASCII TRAJ file is created by including the keyword TRAJASC. If TRAJASC is omitted from the HEAD.ASC option list or NOTRAJASC is included in the HEAD.ASC option list, the ASCII TRAJ file is not created. Creation of these files may be inhibited if the print rate variable, PPP (C(2005)), is set to 0.0. This file is compatible with the QPRINT program for producing tabular output of the file contents and the 2DIM, CHARTS and HIST options of KPLOT, PITA and GLOBE graphics programs for producing plotted output.

3.2.3 STAT

STAT is a file that contains data for up to 70 C locations. The file option list in the HEAD file determines the creation of this file and the format. If a binary STAT file is to be created, the keyword STATBIN should be included in the option list. If STATBIN is not included in the HEAD.ASC option list or if the user includes the keyword NOSTATBIN in the HEAD.ASC option list, the binary STAT file is not created. In addition, the user can create an ASCII STAT file by including the keyword STATASC in the HEAD.ASC option list. If the STATASC keyword is omitted from the HEAD.ASC option list or if the keyword NOSTATASC is included in the HEAD.ASC option list, the ASCII STAT file is not created. This data is printed at each stage in the simulation and at the end of the simulation. This file is compatible with the QPRINT program for producing tabular output of the contents. The file is also compatible with the BIVAR and HISTO options of the KPLOT program for data analysis and plot production.

3.2.4 RANVAR

By using assignment statements, initial values for variables can be defined as a distribution with user-specified parameters. The CADAC Executive will then assign an initial value for these stochastic variables at execution time. The RANVAR file contains a list of the actual data values selected by the CADAC Executive for these assignment stochastic variables. This file is an ASCII file. The creation of this file is determined by the option list in the HEAD.ASC file. If the keyword RANVAR is included in the option list, the file is created. If the keyword RANVAR is not included in the option list or if the keyword NORANVAR is included in the option list, the RANVAR.ASC file is not created. The file may contain only header records if **no** stochastic variables are defined.

3.2.5 IMPACT

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 mode options except TEST and TEST2, this file will contain NUM sets of data for each outer trajectory value. For the TEST and TEST2 mode, this file may contain from 0 sets up to (NUM + 1) sets of data for each outer loop value depending on whether the trajectory criteria is satisfied.

3.2.6 INIT

INIT is a file that contains data for up to 70 C locations. The file option list in the HEAD file determines the creation of this file and the format. If a binary INIT file is to be created, the keyword INITBIN should be included in the option list. If INITBIN is omitted, or if the usr includes the keyword NOINITBIN in the option list, the binary INIT file is not created. In addition, including the INITASC keyword in the HEAD option list can create an ASCII INIT file. If the INITASC keyword is omitted or the keyword NOINITASC is included, the ASCII INIT file is not created. This data is printed at the beginning of the trajectory after the initialization

modules have been processed. The data provides initialization for target track data used in the real-time CADAC application.

3.2.7 TRACK

TRACK is a file that contains data for up to 70 C locations, printed at a user-defined interval. The interval is the same as used for the TABOUT file and should be set to the approximate MIL frame time step. The file option list in the HEAD file determines the creation of this file and the format. If a binary TRACK file is to be created, the keyword TRACKBIN should be included in the option list. If TRACKBIN is omitted or if the user includes the keyword NOTRACKBIN in the option list, the binary TRACK file is not created. In addition, including the TRACKASC keyword in the HEAD option list can create an ASCII TRACK file. If the TRACKASC keyword is omitted or the keyword NOTRACKASC is included, the ASCII TRACK file is not created. The data contained on this file is a time history of the target track data and provides data link updates for use in the real-time CADAC application.

4. CADAC UTILITIES

Several utility programs exist to support the CADAC environment. These utilities can be divided into two classes: 1) CADAC simulation and input file development and 2) debug and output processing and analysis.

~~1.1~~ 4.1 Development Utilities

Formatted: Bullets and Numbering

The developmental utilities assist the CADAC programmer in the creation of the CADAC simulation modules and program the input files. These programs include CADIN, MKHEAD, DFHEAD, AHEAD, FRLOC, INPUT, EQMAP and CHKINT.

Note: The file name for all FORTRAN utilities must not exceed 8 characters.

4.1.1 CADIN

The CADIN (CADAC Input) program converts the free-formatted input file INPUT.ASC into the fixed-formatted file CADIN.ASC that CADAC expects. CADIN prompts the user for the name of the input file with the default being INPUT.ASC. The program requires that a valid HEAD.ASC exist in the current directory. CADIN checks for valid command statements and creates an ERROR.ASC file if invalid statements are found. The following files are utilized in a successful CADIN execution:

INPUT.ASC	-	an updated version of the input file that includes definitions of variables extracted from HEAD.ASC
INPUT.BAK	-	a copy of the input file
CADIN.ASC	-	the fixed formatted CADIN output file for input into CADAC
INPUT.TP5	-	a copy of the CADIN.ASC for archiving purposes(TP5=TAPE5)

Note: If the user inputs a file with a name other than INPUT.ASC, then "INPUT" in the above output file names is replaced by the name entered by the user. More information on CADIN may be found in ***CADAC Program Documentation, Vol III***.

4.1.2 MKHEAD

The MKHEAD (MaKe HEAD) utility program assists the user in creating the HEAD input file. The utility reads the CADAC modules generated by the user and searches for variable names that have been equivalenced to C array locations. A list of these names (and any associated array dimension) is generated and written to an output file HEAD.ASC. This output file provides a starting point for creating the HEAD input file. MKHEAD also provides some error checking for the module code; these error conditions are:

1. The same variable name is equivalenced to different C array locations
2. The same C array location equivalenced to different variable names

3. The number of dimensions in the EQUIVALENCE and DIMENSION statements do not match
4. The locations assigned to an array variable overlap with other defined C locations

If the CADAC Executive code is included in the code checked by MKHEAD, MKHEAD may produce an error indicating that the IC array location 0001 overlaps the ERRN variable; this error is acceptable and can be ignored by the user.

Since MKHEAD searches the actual CADAC module code file, the coding conventions discussed in Section 2.2 must be strictly followed, especially those governing the format of the EQUIVALENCE statements and variables. The MKHEAD program may not work as expected if these coding conventions are not followed. More information on MKHEAD can be found in ***CADAC Program Documentation, Vol III***. **NOTE: the FORTRAN statement EQUIVALENCE must be entered in the source code as all uppercase characters**.

4.1.3 DFHEAD

The DFHEAD (DeFine HEAD) utility provides documentation for the HEAD input file. The input files required by DFHEAD are the user-written CADAC module file and a HEAD.ASC input file. The HEAD.ASC input file may be either the output file generated by MKHEAD or the HEAD input file used by CADAC executions. The documentation is produced by reading the CADAC module file, extracting the variable documentation from the blocks of EQUIVALENCE statements, matching them to the C locations and writing them to an output file named HEAD.ASC. If the variables found in the HEAD.ASC are CADAC executive variables, an appropriate definition is supplied by DFHEAD.

As the program searches the user-written module code for the variable definitions, the coding conventions discussed in Section 2.2 must be strictly followed, especially those governing the format of the variable definitions within the blocks of EQUIVALENCE statements. The DFHEAD program may not work as expected if these coding conventions are not followed. More information on DFHEAD can be found in ***CADAC Program Documentation, Vol III***.

4.1.4 AHEAD

The AHEAD (Alphabetize HEAD) utility produces an alphabetical listing of the variable names. The listing is written to the file AHEAD.ASC. The input file may be either the file generated by MKHEAD or a HEAD input file. The list of variables, associated C array location and variable definition provides documentation for the CADAC module simulation. More information on AHEAD may be found in ***CADAC Program Documentation, Vol III***.

4.1.5 FRLOC

The FRLOC (FRee LOCation) utility generates a list of unused C locations. The input file may be either the file generated by MKHEAD or a HEAD input file. The list of unassigned locations is written to a file named FRLOC.ASC. Blocks of unused locations may be listed as inclusive intervals. FRLOC also writes an error message if an incomplete array dimension is

detected in the input file. More information on FRLOC may be found in ***CADAC Program Documentation, Vol III***.

4.1.6 DFMOD

The DFMOD (DeFine MODules) utility generates variable documentation for the modules by taking variable definitions from the HEAD.ASC input file and inserting them into the user-written CADAC module file. DFMOD only transfers definitions for variables classified as "INPUT FROM OTHER MODULES". Old comments will be replaced with the set of new comments. The HEAD input file may be either the file generated by MKHEAD or a user created HEAD file. More information on DFMOD may be found in ***CADAC Program Documentation, Vol III***.

4.1.7 INPUT

The INPUT program converts the fixed-formatted input file CADIN.ASC into the free-formatted file INPUT.ASC. INPUT prompts the user for the name of the input file with the default being CADIN.ASC. The program requires that a valid HEAD.ASC exist in the current directory. INPUT checks for valid command statements and creates an ERROR.ASC file if invalid statements are found. If no errors occur in the CADIN.ASC file the program creates the following files:

INPUT.ASC	-	the free-formatted file including definitions of variables extracted from HEAD.ASC
INPUT.BAK	-	a copy of the INPUT.ASC for archiving purposes
CADIN.TP5	-	a copy of the CADIN.ASC file for archiving purposes
CADIN.ASC	-	the fixed formatted input file for input into CADAC

Note: More information on INPUT may be found in ***CADAC Program Documentation, Vol III***.

4.1.8 EQMAP

EQMAP (EQuivalence Map) 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 (or AHEAD.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 the variable is equivalenced to
- 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

Table 10 lists the symbols found in the output file generated by EQMAP and what they indicate. More information on EQMAP may be found in ***CADAC Program Documentation, Vol III***.

Table 10. 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 right side of the subroutine name; the variable is being used in the assignment of another variable if the "=" is on the side left of the subroutine name
C(Integer Val)	the C location the variable is equivalanced to
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

4.1.9 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 variables 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. For more information on CHKINT, see the online help provided with the application.

4.2 Advanced Output Processing Utilities

These utility programs assist the user in the debugging process of CADAC simulations, the analysis of CADAC generated data, displaying sweep type analysis and the documentation of

tasks utilizing CADAC. The advanced programs include: CONTOUR, CONVRT, ELLIPSE, GLOBE, KPLOT, MCAP, PITA, QPRINT, SWEEP and WinDRAW.

4.2.1 CONTOUR

The CONTOUR program provides a methodology to generate contours of user selected data from the sweep option of the CADAC program. The contours can be generated manually by interactive mouse movements or by the programs automatic methods. The IMPACT.ASC file generated by the CADAC execution is the required input for the program. The output of CONTOUR is a *.WDR file for input to WinDRAW . For more information on executing the program refer to the on line HELP included in the program or to the ***CADAC Program Documentation, Vol III***.

4.2.2 CONVRT

The CONVRT program provides a real-time application of a CADAC modeled weapon. The input files required by CONVRT are the CADIN.ASC, HEAD.ASC and INIT.ASC files which are used and created from a set of user-supplied modules. The module file, defaulted MODULE.FOR, is also a required input file. The real-time output is a CADRTxx.FOR file. This file needs to be compiled and linked with the CADAC matrix utilities file, UTL.FOR, to be executed within a MIL frame or as a stand-alone testbed. For more information see the ***Real-Time CADAC Documentation, Vol IV***.

4.2.3 ELLIPSE

The ELLIPSE program provides calculated error ellipses imposed over target icons. The input to this program is the SWEEP analysis of a CADAC sweep case and if desired user supplied icon files. The output of ELLIPSE is a *.WDR file for input to WinDRAW. For more information on executing the program refer to the on line HELP included in the program or to the ***CADAC Program Documentation, Vol III***.

4.2.4 GLOBE

The GLOBE application provides the ability to display trajectories over the GLOBE. The application supports the display of ground traces, time traces, stringers, time hacks and color coding of the trajectories. In addition the program displays continents, rivers, political borders, major cities, lines of latitude and longitude and terrain. For more information on the GLOBE application, refer to the on-line HELP included in the program.

4.2.5 KPLOT

The KPLOT program provides the graphical interface for all CADAC post-processing graphics utilities. These include generating two-dimensional plots, strip charts, histograms and

bivariate data plots, as well as 3-D plots using PITA and world plots using GLOBE for both analysis and task documentation. The TRAJ file generated by the CADAC execution is the required input for the two-dimensional and strip charts options of the KPLOT program. The STAT file generated by the CADAC execution is the required input for the histogram and the bivariate data plots options of KPLOT. For more information on the KPLOT program, refer to the on-line HELP included in the program.

4.2.6 MCAP

Program MCAP (Monte Carlo Analysis Program) performs analysis of the TRAJ file 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. For a complete description of the methodology used in the analysis performed by the MCAP program, refer to the ***CADAC Program Documentation, Vol III***.

4.2.7 PITA

The PITA (Program for Interactive Trajectory Animation) application creates three-dimensional plots for analysis and task documentation. The PITA application provides display of up to twelve strip charts, projections onto the x, y and z axes, stringers and time hacks. The application can also be used to animate the trajectory in three different speeds. For more information on the PITA program, refer to the on-line HELP included in the program.

4.2.8 QPRINT

Program QPRINT (quick print) allows the user to view the time-phased TRAJ and STAT data files produced by CADAC simulations. The user may select to view the data on the screen, write the data to a text file or both. QPRINT accepts either TRAJ.BIN, TRAJ.ASC, STAT.BIN, or STAT.ASC as input to the program. For information on the execution of this program, refer to the HELP included in the program menu.

4.2.9 SWEEP

The SWEEP program provides statistical analysis for multi-run/Monte Carlo or deterministic CADAC cases. The IMPACT.ASC generated by the CADAC execution is the required input for the program. For more information on executing the program refer to the on line HELP included in the program or to the ***CADAC Program Documentation, Vol III***.

4.2.10 WinDRAW

The WinDRAW program provides graphical displays of CADAC spatial data on a windows based PC. WinDRAW can be used both to analyze data and to prepare output for

presentation. For more information on executing the program refer to the on line HELP included in the program or to the ***CADAC Program Documentation, Vol III***.

5. SIMULATION DEVELOPMENT

This section provides an outline for developing a model with CADAC.

1. Requirements
   - Define the problem
   - Know your customer
   - What output is required and in what form

2. Modeling
   - Analyze the dynamic system
   - Determine the level of modeling required
     - Identify the subsystem structure and thus the Module break-out
     - Identify the input and output signals of each Module
     - Determine the Module calling sequence
     - Determine the flight mode and staging events
     - Derive the mathematical equations
     - Assign numerical values through input, DATA tables or PARAMETER statements
     - Use consistent units

3. Programming
   - Develop top level flow diagram
   - Decide the subroutine structure within a Module
   - Determine the input and output variables of the Modules
   - Determine the utility subroutines required
   - Code using FORTRAN and following the suggestions for naming variables and coding conventions
   - Compile and correct errors

4. Debugging
   - Run MKHEAD to produce a HEAD.ASC file and check for four types of programming discrepancies:
     1. Same variable name is equivalence to multiple C array locations
     2. Same C array location is equivalence to multiple variable names
     3. Mismatch of DIMENSION and EQUIVALENCE array dimensioning
     4. Overlap of C array variables
   - Create a cross reference list by executing the utility EQMAP. Check the listing for consistent C equivalence's, choice of integer and logical type, array dimensioning
   - Run CHKINT to check state variable integration and obtain the list of state variable c-locations.

5. Execution
   - Create INPUT.ASC file and convert it to CADIN.ASC
   - Provide DUMMY.FOR for dummy return statements of unused Modules
   - LINK all compiled modules, the Executive routine, CADX.FOR, the UTL.FOR and the DUMMY.FOR files
   - Have HEAD.ASC file in default directory to provide:
     1. Identification of variables to scroll on-screen during execution.
     2. Identification of variables to be written to TRAJ.BIN (or TRAJ.ASC)
   - RUN the executable file
   - After completion of run, use QPRINT or KPLOT to look at TRAJ.BIN (or TRAJ.ASC) output

6. Validation
   - Run test cases from simple to complicated
   - Compare with results from other sources
   - Have other experts look at the results

7. Analysis
   - Design Trades: Change input variables
   - Sensitivities: Use multi-run capability
   - Performance: Use Sweep methodology to generate launch envelopes and footprints
   - Accuracy: Introduce stochastic variables and use Monte Carlo option

8. Program Documentation
   - Define C array variables in Modules
   - Run DFHEAD to document HEAD.ASC with definitions in Modules
   - Run DFMOD to document "INPUT FROM OTHER MODULES" in Module code
   - Run AHEAD to alphabetize the variable in HEAD.ASC
   - Run FRLOC to determine unused C array locations
   - Sketch program flow
   - Summarize derivation of equations

9. Documentation of Results
   - Execute QPRINT to generate a time-phased list of variables
   - Execute:
     1. KPLOT for 2-D plotted output, or PITA for 3-D plotted output
     2. BIVAR option of KPLOT to generate and display error ellipses
   - Generate hard copies
   - Use Sweep methodology to generate launch envelopes and footprints

10. Simulation Maintenance
    - Charter CADACer User Group:
        - Assign responsibility for modules
        - Identify modules by version number
    - Publish bulletins to inform users of:
        - Errors in modules
        - New modules and their capabilities
    - Maintain documentation

APPENDIX A
CADAC Utility Subroutines

This appendix documents the subroutines of the UTL.FOR utility file. The CADAC program developer is encouraged to use these subroutines whenever possible. MATxxx utilities apply to two dimensional arrays of any order, except when the operation limits the order to three. DMATxxx are the corresponding double precision utilities. TABLE, TABL2, TABL3 are linear interpolation routines of one, two and three dimensional discrete tables with one dimensional output values. Extrapolation beyond the tables maintains the last table value. Dummies must be used as entry variables into the table because they are left modified after the subroutine call is returned. VECxxx utilities are vector operations in three space. Special CADAC simulations, like CADAC5 (which incorporates a round, rotating earth), may require additional utility subroutines mostly related to coordinate transformations. These routines are labeled CADxxx but are not documented here.

MATABS (D, V, N)
D is the absolute value of vector V(N)

MATADD (C, A, B, N1, N2)
C(N1,N2) is obtained by adding A(N1,N2) to B(N1,N2)

MATCAR (V, D, A, E)
Cartesian coordinates V(3) are generated from polar coordinates D (range), A (azimuth), E (elevation):

$$V = D * \begin{bmatrix} \cos E * \cos A \\ \cos E * \sin A \\ -\sin E \end{bmatrix}$$

MATCHO (A, P, N)
Lower triangular Cholesky matrix A(N,N) is obtained from square root of P(N,N)

MATCON (C, CON, A, N1, N2)
C(N1,N2) is obtained by multiplying constant CON by A(N1,N2)

MATDIA (A, D, N)
A(N,N) is the diagonal matrix obtained from vector D(N)

MATEQL (C, A, N1, N2)
C(N1,N2) is set equal to A(N1,N2)

MATFS (EPSIL, MX, M, A, IER, DET)
MATFS is a single-precision matrix inversion subroutine called from MATINV

MATINV (AI, A, D, N)
Compute the inverse of a non-singular matrix:
AI - Resulting inverse matrix (may be the same as A)
A - Input matrix
D - Determinant of A
N - Number of rows (and columns) of A

MATMUL (C, A, B, N1, N2, N3)
C(N1,N3) is obtained by multiplying A(N1,N2) times B(N2,N3)

MATPOL (D, A, E, V)
Polar coordinates D (range), A (azimuth) and E (elevation) are obtained from Cartesian coordinates V(3):

$$D = |V|$$
$$A = \text{Atan2}(V(2), V(1))$$
$$E = \text{Atan2}\left(V(3), \sqrt{V(1)^2 + V(2)^2}\right)$$

MATROT (R, A, U)
Rotation tensor, R(3,3), generated from angle of rotation, A, and unit vector of rotation, U(3).

MATSCA (D, V1, V2, N)
D is the scalar product of vector V1(N) and V2(N)

MATSKS (C, V)
Skew-symmetric matrix C(3,3) is obtained from vector V(3):

$$\begin{bmatrix} 0 & -V(3) & V(2) \\ V(3) & 0 & -V(1) \\ -V(2) & V(1) & 0 \end{bmatrix} \text{ FROM } \begin{bmatrix} V(1) \\ V(2) \\ V(3) \end{bmatrix}$$

MATSUB (C, A, B, N1, N2)
C(N1,N2) is obtained by subtracting B(N1,N2) from A(N1,N2)

MATTRA (C, A, N1, N2)
C(N2,N1) is the transpose of A(N1, N2).

MATTRF (C, T, A, N)
C(N,N) is obtained from A(N,N) by the orthogonal transformation matrix T(N,N):

$$C = T * A * \bar{T}$$

MATUNI (A, N)
Creation of a unit matrix A(N, N)

MATVDI (D, A, N)
Vector D(N) is the diagonal of matrix A(N,N)

MATVSK (V, A)
Vector V(3) is obtained from the skew-symmetric matrix, A(3,3)

MATZER (A, N1, N2)
Creation of a zero matrix A(N1, N2)

MAT2TR (T, PSI, THT)
T is the transformation matrix obtained from two angles PSI and THT. (Sequence of rotation: PSI then THT)

$$[T] = \begin{bmatrix} \cos THT * \cos PSI & \cos THT * \sin PSI & -\sin THT \\ -\sin PSI & \cos PSI & 0 \\ \sin THT * \cos PSI & \sin THT * \sin PSI & \cos THT \end{bmatrix}$$

MAT3TR (T, PSI, THT, PHI)
Transformation matrix for the three Euler angles, PSI, THT, PHI (in this sequence). These are the Euler angles of flight mechanics, NOT gyrodynamics.

MAT3EU (T, PSI, ALAM, OME)
Transformation matrix for the three Euler angles, PSI, ALAM, OME (in this sequence). These are the Euler angles of gyrodynamics, NOT flight mechanics.

DMATABS (D, V, N)
Double precision MATABS

DMATADD (C, A, B, N1, N2)
Double precision MATADD

DMATDBL (D, S, N1, N2)
Double precision array D(N1, N2) from single precision S(N1,N2)

DMATINV (AI, A, D, N)
Double precision MATINV

DMATMUL (C, A, B, N1, N2, N3)
Double precision MATMUL

DMATSGL (S, D, N1, N2)
Single precision array S(N1,N2) is from double precision D(N1,N2)

DMATSUB (C, A, B, N1, N2)
Double precision MATSUB

DMATTRA (C, A, N1, N2)
Double precision MATTRA

FNGAUS (XMEAN, SIGMA)
Construct Gaussian distribution given mean, XMEAN and SIGMA

FNRSS (R,N)
Constructs root-sum-square of N-vector R

RANF Random number function generator

TABLE (X, XI, YI, NX, Y)
Table interpolation (look-up) routine for a table with one independent variable:

- X - INPUT. The continuous independent variable. Must be a dummy variable and not the variable used in the program.
- XI - Discrete table entries of the independent variable, XI(NX)
- YI - Discrete table elements of the dependent variable, YI(NX)
- NX - The dimension of the table arrays.
- Y - OUTPUT. The continuous numerical value dependent on X.

TABL2 (X, Y, XY, ZI, NXY, Z)
Table interpolation routine for a table with two independent variables:

- X - INPUT. The first independent variable
- Y - INPUT. The second independent variable. Both must be dummy variables
- XY - Discrete table entries; a one dimensional array of length (NXY(1) + NXY(2)) containing discrete points of the first variable followed by the second variable.
- ZI - Table. A one dimensional array of size (NXY(1)*NXY(2)), containing discrete dependent values of the discrete independent table entries. The packing of data into the ZI array is by strings of consecutive values of the first variable for each increasing value of the second variable
- NXY - A two element array containing the dimension of the first variable and the dimension of second variable
- Z - OUTPUT. The continuous numerical value dependent on X and Y.

TABL3 (X, Y, Z, XYZ, ZW, NXYZ, W)
Table interpolation routine for a table with three independent variables:

- X - INPUT. The first independent variable.

Y - INPUT. The second independent variable.
Z - INPUT. The third independent variable.
All input variables must be dummies
XYZ - Discrete table entries; a one dimensional array of length (NXYZ(1) + NXYZ(2) + NXYZ(3)) containing discrete points of the first variable followed by the second and then the third variable.
ZW - Table. A one dimensional array of size (NXYZ(1) * NXYZ(2) * NXYZ(3)), containing discrete dependent values of discrete independent table entries.
NXYZ - A three element array containing the dimensions of the first, second and third variable
W - OUTPUT. The continuous numerical value dependent on X, Y and Z.

VECANG (A, B, C)
Computes the angle A (radians), between vectors B(3) and C(3).

VECUCR (Z, X, Y)
Performs unit cross product of two 3 dimensional vectors, namely: Z = UNIT(X cross Y)

VECUVC (U, A, B, C)
Constructs unit vector U(3) from scalar components A, B, C.

VECVEC (V, A, B, C)
Constructs vector V(3) from elements A, B, C.

APPENDIX B
Program Synopsis

Program Name:	Program Synopsis:
1. AHEAD3.FOR -	Produces alphabetical listing of equivalence variables.
2. CADX3.FOR -	Executive package of subroutines.
3. CADIN3.FOR -	Converts the free-formatted file INPUT.ASC to the formatted file (CADIN.ASC) that CADAC expects.
4. CHKINT12.EXE -	Provides the user with the locations of the C Array locations in the MODULE.FOR. It also checks the state variable location in the header file.
5. CONTOUR35.EXE-	Provides a method of displaying target centered launch envelopes from CADAC data. Requires WinDRAW for output display.
6. CONVRT24.FOR-	Creates a real-time CADAC application code file.
7. DFHEAD3.FOR -	Documents HEAD.ASC with the definitions of variables given in source code.
8. DFMOD3.FOR -	Takes definitions of variables from HEAD.ASC and inserts them into the source code.
9. DRAWOLE.EXE -	The graphics engine for the KPLOT, PITA, GLOBE and WinDRAW programs.
10. DUMMY.FOR -	Dummy RETURN statements for unused Modules.
11. ELLIPSE35.EXE-	Generates graphic displays of error ellipses over target icons. Requires WinDRAW for output display
12. EQMAP3.FOR -	Indicates how the variables in a HEAD.ASC (or AHEAD.ASC) file are used in a MODULE.FOR.
13. FRLOC3.FOR -	Identifies unused C-locations for Module identification.
14. GLOBE40.EXE -	Windows graphics display program, requires the DrawOLE application.
15. INPUT3.FOR -	Converts the formatted file CADIN.ASC into the free-form file INPUT.ASC

16. KPLOT20.EXE - Windows graphics display program, requires the DrawOLE application.

17. MCAP35.EXE - Performs analysis of the TRAJ file generated by CADAC multi-run/Monte Carlo simulations.

18. MKHEAD3.FOR Generates HEAD.ASC; sorts CADAC source code on C-locations. Provides some error checking.

19. PITA20.EXE - Windows 3-D graphics display program, requires the DrawOLE application.

20. QPRINT35.EXE - Post-listing of all variables that were written to TRAJ.BIN (or TRAJ.ASC).

21. SWEEP35.EXE - Statistics Analysis and Smoothing Program utilizing sweep type CADAC data.

22. UTL3.FOR - Utility subroutines for array processing and table look-ups.

23. WINDRAW.EXE - Windows graphics display program.