NASA TM-86367

NASA-TM-86367 19850016410

NASA Technical Memorandum 86367

USERS GUIDE: THE LaRC HUMAN-OPERATOR-SIMULATOR-BASED PILOT MODEL

Edward H. Bogart and Marvin C. Waller

APRIL 1985

LIBRARY COPY

APR 29 1985

LANGLEY RESEARCH CENTER
LIBRARY, NASA
HAMPTON, VIRGINIA

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

3 1176 01359 9718

SUMMARY

A Human Operator Simulator (HOS) based pilot model has been developed for use at NASA LaRC for analysis of flight management problems. The model is currently configured to simulate piloted flight of an advanced transport airplane. The generic HOS operator and machine model was originally developed under U.S. Navy sponsorship by Analytics, Inc. and through a contract with LaRC was configured to represent a pilot flying a transport airplane.

A version of the HOS program runs in batch mode on LaRC's (60-bit-word) central computer system. This document provides a guide for using the program and describes in some detail the assortment of files used during its operation.

1.0 INTRODUCTION

The Human Operator Simulator (HOS) described in this document is a version of the generic HOS operator and system model developed under U.S. Navy sponsorship by Analytics, Inc. The LaRC version, developed under contract (NASA 15983, ref. 1), models piloted flight of the Transport Systems Research Vehicle (ref. 2) during instrument approach to landing operations. HOS can be modified to simulate different flight scenarios, pilot characteristics, and piloting strategies. Characteristics of the aerodynamics, controls system, and displays of the airplane are represented in the model through the input data set. The LaRC version of HOS described in this document retains its generic nature and, through changes to the input data set, an operator of any system could be simulated. Two scenarios are currently programmed in the Transport Systems Research Vehicle (TSRV) model simulation - an ILS straight-in approach and a Microwave Landing System (MLS) curved approach. The straight-in approach, presented as an example in this document, is described in appendix A.

The purpose of this document is to describe the files used in execution of the LaRC HOS simulation programs, the nature of the input data, and the types of output records and analysis available from HOS as it currently exists. No attempt will be

N85-24721#

made to provide a detailed explanation of how a simulation of a particular system can be developed since such a description is given in reference 1. It is assumed that the user will have acquired a copy of the magnetic tape containing the referenced files and wishes to initially operate the HOS TSRV simulation as formated on the tape. The user is also assumed to have a working knowledge of the CDC computer system and the associated NOS operating system. Detailed documentation of the generic HOS programs can be obtained from Analytics, Inc., 2500 Maryland Road, Willow Grove, PA 19090.

Throughout this document, file names will be printed using ANSI standard for ASCII characters with the number Ø and the letter O, with the exception of some reproduced computer listings. Names of HOS created files referenced within the text of this document will be printed with fixed characters in upper case and the variable characters represented by a dollar sign ($) when the reference is made to the file name in the unedited procedure file. An example selected for use throughout much of this discussion will be referred to by the character string "EX". The dollar sign ($) in the various file names in the unedited procedure files will be replaced by the string "EX" when the discussion is intended to specifically address the example. Also, when listings of files are presented in figures, the variable character string is represented by "$NAME$".

LIST OF ACRONYMS AND ABBREVIATIONS

ASCII	American Standard Code for Information Interchange
CCL	CYBER Control Language
CDC	Control Data Corporation
CDF	Crewstation Data File
CPFS	Common Permanent File System
HAL	HOPROC Assembler Loader
HODAC	Human Operator Data Analyzer and Colator

HOS	Human Operator Simulator
ILS	Instrument Landing System
LaRC	Langley Research Center
MLS	Microwave Landing System
NOS	Network Operating System
NPL	New Program Library file
OPL	Old Program Library file
TCV	Terminal Configured Vehicle
TSRV	Transport Systems Research Vehicle, alternate name for TCV
PROC	Procedure file

LIST OF REFERENCED FILE NAMES

C1$	UPDATE directive file output from HAL
C2$	UPDATE directive file output from HAL
CRUISE	CREWSTATION data file for curved flight scenario
DATA30	Atmospheric data file for aircraft submodel
D1$	Pseudo machine instruction file for HAL
D2$	Pseudo machine instruction file from step HAL
EXEC	File containing all the Procedures used in the execution of a simulation
G1$	Packed binary plot data file generated in step HOS, used in step HODAC
HAL	Submit file for step HAL
HALHOD	Submit file for step HALHOD
HALHOS	Submit file for step HALHOS
HODAC	Submit file for step HODAC
HOS	Submit file for step HOS
HOSMOD	Submit file for step HOSMOD
H7MOD	UPDATE directive file for step HOSMOD
IFILE	Default name of UPDATE directive input file

LAND	CREWSTATION data file for straight flight scenario
LHODAC	FORTRAN source code for HODAC program (in UPDATE format)
LHOS	FORTRAN source code for HOS program (in UPDATE format)
M1$	Machine readable time history generated by step HOS, input to step HODAC
O1$	Binary load module executed in step HOS
O2$	Binary load module executed in step HODAC
PLOT	Submit file for execution of HOS plot
PLOTLIB	Runtime library for HOS plot execution
PLOTMOD	UPDATE directive file for file PLOTSYS
PLOTSYS	FORTRAN data conversion program used by PLOT
S1$	UPDATE new program library file from step HALHOS
S2$	UPDATE new program library file from step HOSMOD
T3$	Binary plot data output from HOS; input to PLOT
TAPE45	Binary input data for aircraft submodel
TCVLIB	Runtime library for aircraft submodel in HOS execution
WGUSLIB	Runtime library for atmosphere submodel in HOS execution

2.0 BATCH OPERATING PROCEDURES

The steps involved in execution of a flight simulation using HOS are summarized in figure 1. To complete a simulation of a system, three FORTRAN computer programs are normally compiled and executed - HAL, HOS, and HODAC. (Program names may vary, e.g., LHALO, LHOS, and LHODAC.) Other steps represented in figure 1 are related to constructing the file environment necessary to operate these three programs.

In the first major step in developing a simulation, the hardware to be simulated is analyzed and coded as mathematical and logical functions using a special superset of FORTRAN called HOPROC. Similarly, the strategies and rules used by the pilot are analyzed and coded using HOPROC. The resulting file is the input required for execution of the first in the sequence of three programs listed above, HAL. HAL is

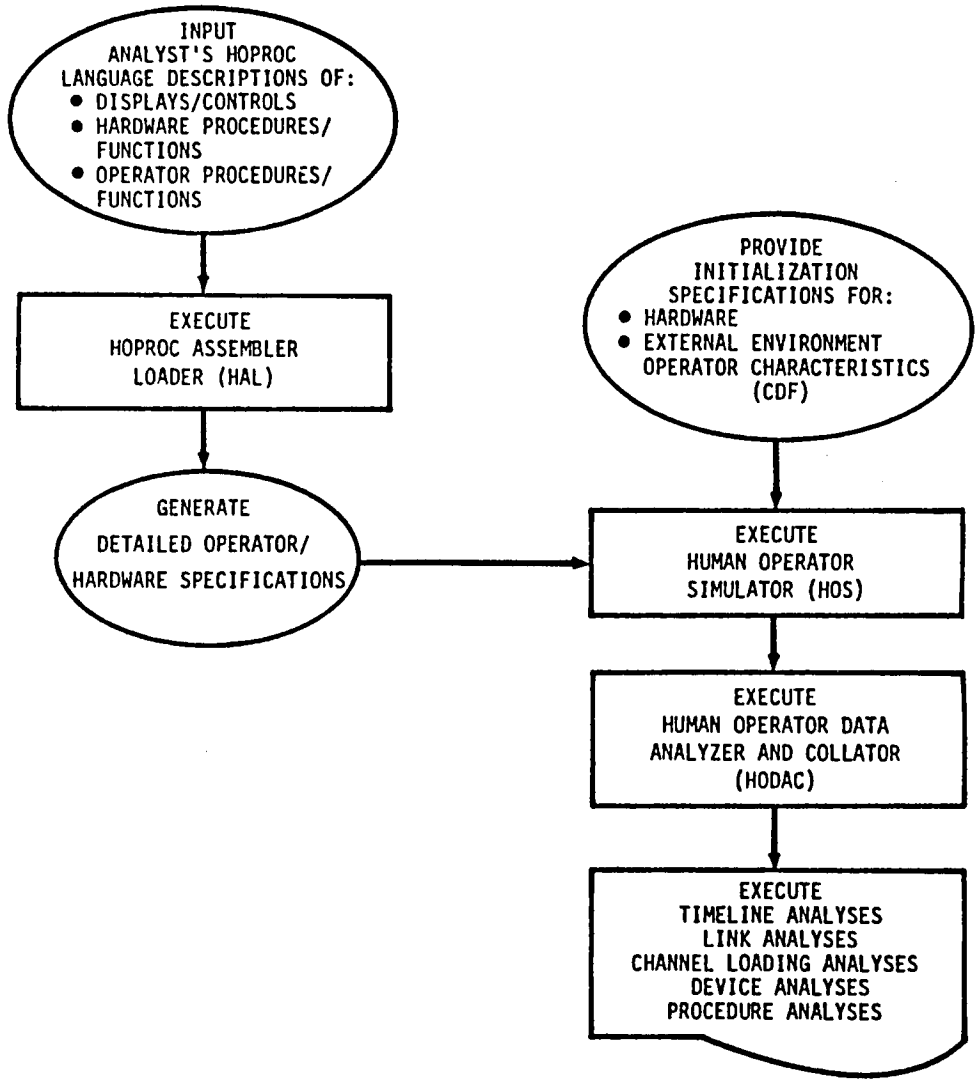


Figure 1. - Simulator Steps.

described as an assembly/loader and functions to format the description of the specific system under study into an UPDATE MOD file used to edit the generic HOS program. HAL also generates a numerically coded procedure file that is used as input during the HOS step.

The second major step in simulating a system is to operate the HOS program. This step comprises the actual execution of the simulation. The Crew Data File (CDF) provides input data to support the HOS step. It consists of descriptions of the locations, initial values, and operational characteristics of the state variables, display and control devices used during the simulation.

Operation of HODAC, the Human Operator Data Analyzer and Collator, is the third major step in completing a simulation. This program uses files generated in the HOS step to develop several analyses. These include a timeline analysis, a link analysis, channel loading report, a device analysis, and a procedure analysis.

In practice, the operations necessary to complete a simulation and the associated data reduction and analysis have been organized into the following eight submit files:

1. CLEAR
2. HAL
3. HALHOS
4. HOSMOD
5. HOS
6. PLOT
7. HALHOD
8. HODAC

Operation of CLEAR, HAL, HALHOS, HOSMOD, and HOS is normally necessary to accomplish the actual system simulation. Details of operation of these five submit files will be discussed in sections 2.1, 2.2, 2.3, 2.4, and 2.5, respectively. PLOT generates time history plots of selected variables associated with the flight simulation. Its

operation will be discussed in section 3.0. The submit files HODAC and the associated submit file HALHOD generate data reduction and analysis reports. Details of their operation will be presented in section 4.0. Appendix B presents a list of all of the files used along with a brief verbal description of each.

2.1 CLEAR

The first step required to complete an HOS simulation is to operate the submit file CLEAR. It initializes the files used by subsequent simulation steps.

Before CLEAR can be submitted, the string "??" in the sixth line of the file must be replaced by the characters that will become the variable part of the file names. In all the examples in this document, the string "EX" (short for example) will be used. After editing, CLEAR should appear as shown in Figure 2.

```
/JOB
CLEAR,T5ØØ,CM12ØØØØ.              DESTINATION INFORMATION
USER,#######.
CHARGE,#######,LRC.
ATTACH,XEDIT/UN=LIBRARY.
XEDIT,EXEC,P.;XC/$NAME$/EX/*;XEND.
RETURN,XEDIT.
BEGIN,1CLEAR,EXEC.
BEGIN,1EXIT,EXEC.
```

Figure 2.- Submit file for step CLEAR.

When CLEAR executes, it gets EXEC and, using XEDIT, replaces all occurrences of $NAME$ with the string "EX". PROC 1CLEAR, as shown in figure 3, purges all the indirect access temporary files used by HOS and defines and saves one direct access file, M1EX.

```
.PROC,1CLEAR
PURGE,M1$NAME$/NA.
PURGE,$NAME$D/NA.
PURGE,$NAME$O/NA.
PURGE,C1$NAME$/NA.
PURGE,C2$NAME$/NA.
PURGE,D1$NAME$/NA.
PURGE,D2$NAME$/NA.
PURGE,G1$NAME$/NA.
PURGE,T1$NAME$/NA.
PURGE,T2$NAME$/NA.
PURGE,S2$NAME$/NA.
PURGE,O2$NAME$/NA.
DEFINE,M1$NAME$.
RETURN,M1$NAME$.
REVERT.
EXIT.
REVERT.
--ERO--
```

Figure 3.- PROC for step CLEAR.

PROC 1EXIT copies and saves the dayfile as EXD. A sample dayfile is shown in figure 4.

Execution of step "CLEAR" does not produce output on file $O or OUTPUT. The output returned to the user consists of the dayfile only.

```
11.31.17.CLEAR,T5Ø,CM12ØØØØ.          DESTINATION INFORMATION
11.31.17. USERNAME.
11.31.17.USER,#####
11.31.17.CHARGE,####,LRC.
11.31.17.ATTACH,XEDIT/UN=LIBRARY.
11.31.18.XEDIT,EXEC.P.;XC/$NAME$/EX/*;XEND.
11.31.21.RETURN,XEDIT.
11.31.21.BEGIN,1CLEAR,EXEC.
11.31.22.PURGE,M1EX/NA.
11.31.23. M1EX NOT FOUND, AT 121.
11.31.23.PURGE,EXD/NA.
11.31.26.PURGE,EXO/NA.
11.31.27.PURGE,C1EX/NA.
11.31.28.PURGE,C2EX/NA.
11.31.28.PURGE,D1EX/NA.
11.31.29.PURGE,D2EX/NA.
11.31.29.PURGE,G1EX/NA.
11.31.30.PURGE,T1EX/NA.
11.31.31.PURGE,T2EX/NA.
11.31.31.PURGE,S2EX/NA.
11.31.32.PURGE,O2EX/NA.
11.31.32.DEFINE, M1EX
11.31.33.EXIT.
11.31.33.REVERT.
11.31.33.BEGIN,1EXIT,EXEC.
11.31.33.DAYFILE,EXD.
```

Figure 4.- Dayfile from step CLEAR.

2.2 Execution of Submit File HAL

Submit file HAL shown in figure 5 contains a series of CCL statements that perform NOS system level operations. Prior to submitting, the sixth line is edited to replace the string "??" with a one to three character string, the string "EX" has been inserted in this example. HAL gets a local copy of the PROC file EXEC and replaces all occurrences of the string $NAME$ with the one to three character file set name (e.g., EX) previously selected by the operator and used previously in step CLEAR.

```
/JOB
HALEX,T5ØØ,CM23ØØØØ.              DESTINATION INFORMATION
USER,#######.
CHARGE,######,LRC.
ATTACH,XEDIT/UN=LIBRARY.
XEDIT,EXEC,P.;XC/$NAME$/EX/*;XEND.
RETURN,XEDIT.
BEGIN,1HAL,EXEC,INPUT.
BEGIN,1EXIT,EXEC.
/NOSEQ
/EOR
SYSTEM                    MASTER-----TCV
SETTING SECTION
     ON   OFF'
     FO F1 F5 F15 F25 F3Ø F4Ø
     RIGHT LEFT STRAIGHT
```

Figure 5.- Submit file for step HAL with appended HOPROC code.

The edited local copy of PROC 1HAL (fig. 6) in file EXEC is executed by the eighth line of HAL, "BEGIN,1HAL,EXEC,INPUT.". This same line passes the HOPROC data file to PROC 1HAL. The HOPROC file is normally appended to HAL after the /EOR line. It becomes the input file for the execution step 1HAL.

```
.PROC,1HAL,HOPROC.
GET,LHALO.
PURGE,$NAME$D/NA.
PURGE,$NAME$O/NA.
MAP=OFF.
LDSET,PRESET=ZERO.
LHALO,HOPROC,$NAME$O,C1,C2,,D1,D2.
REPLACE,C1=C1$NAME$/NA.
REPLACE,C2=C2$NAME$/NA.
REPLACE,D1=D1$NAME$/NA.
REPLACE,D1=D2$NAME$/NA.
REVERT.
EOR
```

Figure 6.- PROC for step HAL.

PROC 1HAL gets file LHALO and purges the dayfile and output file (if they exist) from a previous execution. References to HOPROC are replaced by INPUT. The load module LHALO executes reading data from INPUT. LHALO translates the instructions and data in INPUT into forms that will be used by later execution steps and writes the

files C1$NAME$, C2$NAME$, D1$NAME$, and D2$NAME$. The output file $O consists of a listing of input file HOPROC and any diagnostic and execution information.

2.2.1 File HOPROC

The file passed with name INPUT to PROC 1HAL is the HOPROC file that contains program code specific to the simulation being conducted. It is written in Human Operator Procedures Language (HOPROC), a superset of CDC FORTRAN-IV Extended.

The HOPROC file consists of three major sections: a title declaration section, a functions section, and a procedures section.

2.2.1.1 Title

The title section contains the names of all devices to be used in the simulation and the names of the legal settings, scale factors, and symbols to be used. The title section is processed by LHALO into the Data Dictionary and is the basis for the entries in the crew station data file.

2.2.1.2 Functions

The function section contains both Operator Functions and Hardware Functions. Operator Functions describe the mental calculations that the pilot is to make based on the information available to him. Each of the Operator Functions consists of a function name followed by HOPROC code that defines the mental operations performed to accomplish the tasks named. Hardware Functions describe the manipulation of information by the hardware in the simulation. They consist of a function name in quotes followed by HOPROC code defining what takes place.

2.2.1.3 Procedures

The procedure section contains Operator Procedures and Hardware Procedures. Operator Procedures embody the pilots knowledge of the steps required to perform a task, their order of execution, and the criteria used to determine when to start a task or when a task is completed. Hardware Functions describe the hardware

consequences of actions taken by the pilot as well as independent actions taken by the systems being simulated.

The Analytics in-house Technical Report 14ØØ.22E contains a complete description of rules and syntax of HOPROC and describes how a HOPROC coded file is generated.

When execution of PROC lHAL is complete, control returns to submit file HAL and the PROC lEXIT is executed by the line "BEGIN,lEXIT,EXEC.".

2.2.2. PROC lEXIT

PROC lEXIT copies the dayfile to $D, copies $D and $O to OUTPUT and saves $D and $O.

2.2.2.1 File $D

File $D contains the system execution history, dayfile, of the job. An example is shown in figure 7. It will be identical with the dayfile printed at the end of the printed program output but it is available via a terminal immediately after job completion.

```
12.51.32.HAL,T5ØØ,CM23ØØØØ.
12.51.32.USERNAME.
12.51.32.USER,####.
12.51.32.CHARGE,####,LRC.
12.51.33.ATTACH,XEDIT/UN=LIBRARY.
12.51.35.XEDIT,EXEC,P.;XC/$NAME$/EX/*;XEND.
12.52.Ø6.RETURN,XEDIT.
12.52.Ø6.BEGIN,lHAL,EXEC,INPUT.
12.52.Ø7.GET,LHALO.
12.52.19.IFE,FILE(INPUT,ASA)=Ø,GLOM.
12.52.21.ENDIF,GLOM.
12.52.21.PURGE,EXD/NA.
12.52.29.PURGE,EXO/NA.
12.52.3Ø.PURGE,OlEX/NA.
12.52.3Ø. OlEX NOT FOUND, AT 121.
12.52.31.MAP=OFF.
12.52.31.LDSET,PRESET=ZERO.
12.52.31.LHALO,INPUT,EXO,Cl,C2,,Dl,D2.
12.56.46.       STOP
12.56.46.       1411ØØ MAXIMUM EXECUTION FL.
12.56.46.       25.Ø4Ø CP SECONDS EXECUTION TIME.
12.56.46.REPLACE,Cl=ClEX/NA.
```

Figure 7.- Sample dayfile from step HAL.

```
12.56.49.REPLACE,C2=C2EX/NA.
12.56.51,REPLACE,D1=D1EX/NA.
12.56.53.REPLACE,D2=D2EX/NA.
12.56.55.REVERT.
12.56.55.BEGIN,1EXIT,EXEC.
12.56.56.REWIND,OUTPUT.
12.56.56.SKIPEI,EXO.
12.56.58.COPYEI,OUTPUT,EXO.
12.56.59. EOI ENCOUNTERED.
12.56.59.REWIND,EXD,EXO,OUTPUT.
12.56.59.COPYEI,EXO.
12.57.ØØ. EOI ENCOUNTERED.
12.57.ØØ.REPLACE,EXO.
12.57.Ø6.DAYFILE,EXD.
```

Figure 7. Concluded.

2.2.2.2 File $O

File $O contains the output of the execution of LHALO. It contains a listing of the INPUT file, a dictionary of variable names, and set of HOS data input forms.

The listing of the input file could be slightly expanded and contains some explanatory and diagnostic statements such as those shown in figure 8.

The dictionary portion of $O contains a list of all names of symbols, devices, functions, and procedures that are to be used by HOS and the index number or ID that LHALO has assigned to each.

The data entry forms in $O list the dictionary entry name and provides space to fill in all the information about each name that is required to run the simulation. These data entry forms are only used when a new simulation is being defined and a new Crewstation Data File (CDF) required. See Appendix B of this document for a discussion of the CDF.

```
                    HAL -- THE HOPROC ASSEMBLER/LOADER

          MASTER------TCV                                   82/Ø6/Ø4.

HARDWARE PROCEDURES

*******************************************************************************
     DEFINE THE PROCEDURE AC-UPDATE.

               IF HOR-PATH-LIGHT IS ON THEN COMPUTE AUTO-BROLLY-ROLL.

               COMPUTE NEW-SPEED, NEW-BANK,

               NEW-GAMMA, NEW-LOCATION, PATH-ERRORS.

$ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $ $

                                   WARNING

        HARDWARE PROCEDURES HAVE NOT BEEN DEFINED FOR THE FOLLOWING CONTROLS

                                   AUTO SWITCH

                                   HOR PATH SWITCH

                                   VERT PATH SWITCH

                                   SPEED BREAK LEVER

                                   THROTTLE LEVER

                                   DH REF KNOB

           THE FOLLOWING DEFAULT HARDWARE PROCEDURE WILL BE ASSUMED --

         END       CHANGE THE ACTUAL VALUE OF <CONTROL> TO THE DESIRED VALUE
                   OF <CONTROL>
```

Figure 8.- HOS diagnostics from output file $0.

2.3 Execution of Submit File HALHOS

Submit file HALHOS, as shown in figure 9, gets a local copy of file EXEC and changes all occurrences of the string "$NAME$" to the file name previously selected (the character string "EX" in this example).

```
/JOB
HALHOS,T5ØØ,CM33ØØØØ.                    DESTINATION INFORMATION
USER,#######.
CHARGE,######,LRC.
ATTACH,XEDIT/UN=LIBRARY.
XEDIT,XEC,P.;XC/$NAME$/EX/*;XEND.
RETURN,XEDIT.
BEGIN,1HALHOS,EXEC,A1234,EX7MOD.
BEGIN,1EXIT,EXEC.
END OF FILE
```

Figure 9.- Submit file for step HALHOS.

The statement "BEGIN,1HALHOS,EXEC,A1234,EXMOD." begins the PROC 1HALHOS, shown in figure 1Ø, from the edited copy of EXEC, passes the UPDATE list parameters (A1234) and passes the name of the UPDATE directive file (EXMOD) to be used.

```
.PROC,1HALHOS,LIST,IFILE.
GET,S1=LHOS.
GET,C1=C1$NAME$.
GET,P1=B737PLA.
IFE,FILE(IFILE,AS)=Ø,GETIT.
GET,B73MOD.
GET,IFILE.
ENDIF,GETIT.
UPDATE,O=Ø,C=Ø,P=P1,N=P2,F,S,I=Ø.
UPDATE,O=Ø,C=Ø,P=S1,F,N=N1,L=LIST,I=IFILE.
UPDATE,O=Ø,C=Ø,P=N1,F,N=N2,L=LIST,I=C1.
UPDATE,O=/Ø,C=COMPILE,F,P=N2,N=NEW,L=LIST,I=B73MOD.
RETURN,S1,C1,N1.
FTN,I=COMPILE,L=Ø,A,OPT=Ø,R=3.
REPLACE,NEW=S1$NAME$.
REPLACE,COMPILE=$NAME$C1.
REPLACE,LGO=O1$NAME$.
REVERT.
```

Figure 1Ø.- PROC for step HALHOS.

PROC 1HALHOS loads files LHOS, C1$, B737PLA, B73MOD, and $MOD. File LHOS is a permanent library file of HOS source code in UPDATE format. File C1$ is the UPDATE directive file generated by step HAL. B73MOD and $MOD are user generated UPDATE directive files containing changes to be made to LHOS.

File LHOS is loaded as S1 and is modified with directive file $MOD in the second UPDATE step to produce the UPDATE New Program Library (NPL) file N1. Directives in

file C1$ add COMMON's and the subroutines HFUNC and MFUNC to N1 producing an NPL file N2 in the third update step.

File B737PLA is loaded as file P1. The first UPDATE step:

```
UPDATE,O=Ø,P=P1,N=P2,F,S,I=Ø.
```

creates New Program Library file P2. The fourth UPDATE step:

```
UPDATE,O=Ø,C=COMPILE,F,P=N2,N=NEW,L=Ø,I=B73MOD.
```

modifies P2 and adds it as a group of subroutines onto the end of N2, creating the NPL file NEW and the compiler source file COMPILE.

File NEW is saved as file S1$ and can be used as the input file in the optional step HOSMOD.

The NPL file COMPILE is compiled and the resulting LGO file is saved as file O1$ which is used in steps HOSMOD and HOS. The compiler source file COMPILE is saved as $C1.

At the end of the execution of PROC 1HALHOS, control returns to submit file HALHOS. The next line:

```
BEGIN,1EXIT,EXEC.
```

begins execution of PROC 1EXIT. Files $O, $D, and OUTPUT are rewound and copied onto OUTPUT for printing. Files $O and $D are saved as indirect access files. Figure 11 shows the dayfile, file $D, for a typical HALHOS execution.

If HALHOS executes with no UPDATE or FTN errors, no output is produced, other than the usual job header and dayfile. If output is desired from any of the UPDATE steps, the output option O=Ø can be changed to O=OUTPUT or O=$O. A printed listing of the entire FORTRAN source file at compile time can be obtained by modifying L=Ø to L=OUTPUT in the FTN step.

```
14.39.40.HALHOS,T50Ø,CM33ØØØØ.                    RM 2138
14.39.40.#######.
14.39.40.USER.#######.
14.39.40.CHARGE,######,LRC.
14.39.41.ATTACH,XEDIT/UN=LIBRARY.
14.40.11.XEDIT,EXEC,P.;XC/$NAME$/EX/*;XEND.
14.40.14.RETURN,XEDIT.
14.40.15.BEGIN,1HALHOS,EXEC,A1234,EXMOD.
14.40.15.GET,S1=LHOS.
14.40.18.GET,C1=C1EX.
14.40.19.GET,P1=B737PLA.
14.40.21.IFE,FILE(EXMOD,AS)=0,GETIT.
14.40.21.GET,B73MOD.
14.40.22.GET.EXMOD.
14.40.23.ENDIF,GETIT.
14.40.40.UPDATE,O=Ø,C=Ø,P=P1,N=P2,F,S,I=Ø.
14.40.50. UPDATE COMPLETE.
14.40.50.UPDATE,O=Ø,C=Ø,P=S1,F,N=N1,L=A1234,I=EXMOD.
14.41.10.DECK STRUCTURE CHANGED
14.41.17. UPDATE COMPLETE.
14.41.17.UPDATE,O=Ø,C=Ø,P=N1,F,N=N2,L=A1234,I=C1.
14.41.32. UPDATE COMPLETE.
14.41.33.UPDATE,O=Ø,C=COMPILE,F,P=N2,N=NEW,L=A1234,I=B73MOD.
14.42.05. UPDATE COMPLETE.
14.42.05.RETURN,S1,C1,N1.
14.42.05.FTN,I=COMPILE,L=Ø,A,OPT=Ø,R=3.
14.45.36.   147.055 CP SECONDS COMPILATION TIME
14.45.37.REPLACE,NEW=S1EX.
14.45.53.REPLACE,COMPILE=EXC1.
14.46.31.REPLACE,LGO=O1EX.
14.46.41.REVERT.
14.46.41.BEGIN,1EXIT,EXEC.
14.46.42.REWIND,OUTPUT.
14.46.42.SKIPEI,EXO.
14.46.42.COPYEI,OUTPUT,EXO.
14.46.42. EOI ENCOUNTERED.
14.46.42.REWIND,EXD,EXO.OUTPUT.
14.46.42.COPYEI,EXO.
14.46.43. EOI ENCOUNTERED.
14.46.43.REPLACE,EXO.
14.47.11.DAYFILE,EXD.
```

Figure 11.- Sample dayfile from step HALHOS.

2.4 Execution of Submit File HOSMOD

HOSMOD is the only optional step in the sequence from CLEAR to HOS. It is a relatively inexpensive and quick method of making changes in the executable HOS file O1$. It uses UPDATE and LIBEDIT to make changes in source file S1$ but only

those subroutines in which changes are made are recompiled thus reducing the field length and CPU time required.

In general, HOSMOD is used as a troubleshooting and coding aid. Changes can be made and tested without the cost of rerunning HALHOS. After a change has been tested and verified using HOSMOD, the UPDATE directive can be included in $MOD or B737MOD and HALHOS can be rerun.

The user is cautioned against the pitfall of making a large number of changes in the model using HOSMOD only to find that later the changes must be reprogrammed for inclusion in the HAL step.

Submit file HOSMOD (fig. 12) gets and edits a local copy of EXEC, changing all occurrences of the string "$NAME$" to the selected file name (the string "EX" in this example). HOSMOD executes PROC 1HOSMOD from the edited copy of EXEC. The line

BEGIN,1HOSMOD,EXEC,H7MOD.

begins the PROC and passes the UPDATE directive file name H7MOD. PROC 1HOSMOD (fig. 13) purges the old output file $O and the old dayfile $D if they exist. The UPDATE Old Program Library (OPL) file S1$ is loaded as local file S1 (File S1$ is the complete FORTRAN source for HOS in UPDATE format that was created in step HALHOS). The load module O1$ (the compiled version of S1) is loaded as local file NEW.

```
/JOB
HOSMOD,T5ØØ,CM33ØØØØ.                    DESTINATION INFORMATION
USER,#######.
CHARGE,######,LRC.
ATTACH,XEDIT/UN=LIBRARY.
XEDIT,EXEC,P.;XC/$NAME$/EX/*XEND.
RETURN,XEDIT.
BEGIN,1HOSMOD,EXEC,A1234,H7MOD.
BEGIN,1EXIT,EXEC.
```

Figure 12.- Submit file for step HOSMOD.

```
.PROC,1HOSMOD,IFILE.
GET,S1=S1$NAME$.
GET,OLD=O1$NAME$.
IFE,FILE(EDIT,AS)=Ø,GETIT1.
GET,EDIT.
ENDIF,GETIT1.
IFE,FILE(EDIT,AS)=Ø,GETIT2.
GET,EDIT.
ENDIF,GETIT2.
UPDATE,O=Ø,P=S1,N=N1,L=LIST,I=IFILE.
FTN,I=COMPILE,L=Ø.
LIBEDIT,LO=$NAME$L,I=EDIT.
REPLACE,N1=S1$NAME$.
REPLACE,NEW=O1$NAME$.
REVERT.
EXIT.
REVERT.
```

Figure 13.- PROC for step HOSMOD.

The UPDATE step in 1HOSMOD,

```
UPDATE,O=Ø,P=S1,N=N1,L=Ø,I=IFILE.
```

modifies file S1 using the directives in $MOD. Since the UPDATE parameter F is not specified, only those subroutines that have been changed are included in the UPDATE NPL File N1.

The FTN step recompiles the subroutines that have been UPDATE'd and LIBEDIT replaces those sections of local file NEW that have been changed.

The new source file N1 is saved as S2$ and the load file NEW is saved as O1$. Control returns to submit file HOSMOD.

HOSMOD executes the PROC 1EXIT which rewinds files $O, $D, and OUTPUT and copies them to OUTPUT to be printed. Files $O and $D are saved as indirect access files.

The argument A1234 is passed as the UPDATE parameters L=LIST. This will cause error messages and diagnostics to be printed if any errors occur. In a normal execution of 1HOSMOD, no output is produced. If a complete UPDATE history is desired, the O option may be changed to O=OUTPUT.

Setting the FTN parameter L to L=OUTPUT produces a complete source listing of those subroutines that have been modified and recompiled. Changing the parameter to L=Ø will suppress the listing. (It is usually desirable to leave L=OUTPUT.)

THE LIBEDIT parameter L=OUTPUT produces a record of which the subroutines and procedures in O1$ have been replaced. This output can be suppressed by changing to L=0.

2.5 Execution of Submit File HOS

Submit file HOS, shown in Figure 14, controls the actual execution of the simulation using the files that have been created and modified by steps CLEAR, HAL, HALHOS, and HOSMOD. Submit file HOS gets a local copy of PROC file EXEC and replaces all occurrences of the string "$NAME$" with file name character string previously selected (the character string "EX" in this example). The line:

BEGIN,lHOS,EXEC,LAND.

executes the PROC lHOS, shown in figure 15, from the edited copy of the file EXEC.

PROC lHOS gets the FORTRAN binary load file O1$, the LGO file created by step HALHOS. It gets D1$, one of the pseudo machine code instruction files created by step HAL, and the Crew Data file (LAND in this example). PROC lHOS attaches the direct access FORTRAN run time math library FTNMLIB, and gets the subroutine library files TCVLIB and WGUSLIB, and the input data files T30 and T45. The four LIB files are combined by the first LDSET into the run time LIB for the execution of O1$.

File O1$ is executed producing output in file $O and the previously defined direct access files M1$, T1$, T3$ and G1$. The direct access files are returned and control is passed back to submit file HOS.

File HOS executes PROC lEXIT which rewinds files $O and $D and copies them to OUTPUT to be printed.

```
/JOB
HOS,T5ØØ,CM33ØØØØ.                          DESTINATION INFORMATION
USER,#######.
CHARGE,######,LRC.
ATTACH,XEDIT/UN=LIBRARY.
XEDIT,EXEC,P.;XC/$NAME$/EX/*;XEND.
RETURN,XEDIT.
BEGIN,1HOS,EXEC,LAND.
BEGIN,1EXIT,EXEC.
```

Figure 14.- Submit file for step HOS.

```
.PROC,1HOS,CREW.
GET,O1=O1$NAME$.
GET,D1=D1$NAME$.
GET,T30=DATA30,TCVLIB/UN=11953ØC.
GET,WGUSLIB,T45=TAPE45/UN=34925ØC.
IFE,FILE(TAPE7,AS)=Ø,GOTIT1.
GET,TAPE7/NA.
ENDIF,GOTIT1.
IFE,FILE(TAPE8,AS)=Ø,GOTIT2.
GET,TAPE8/NA.
ENDIF,GOTIT2.
IFE,FILE(CREW,AS)=Ø,GOTIT3.
GET,CREW/NA.
ENDIF,GOTIT3.
PURGE,M1$NAME$/NA,ST=LPF.
DEFINE,M1=M1$NAME$/M=W.
PURGE,T1$NAME$/NA,ST=LPF.
DEFINE,T1=T1$NAME$/M=W.
PURGE,T3$NAME$/NA,ST=LPF.
DEFINE,T3=T3$NAME$/M=W.
PURGE,G1$NAME$/NA,ST=LPF.
DEFINE,G1=G1$NAME$/M=W.
ATTACH,FTNMLIB/UN=LIBRARY.
LDSET,LIB=FTNMLIB/TCVLIB/WGUSLIB.
LDSET(MAP=SBEX).
LDSET,PRESET=ZERO.
O1,INPUT,$NAME$O,TAPE7,TAPE8,T2,D1,M1,G1,CREW,T1,T3,T3Ø,T4Ø,T45.
DISPLAY,EFG.
RETURN,M1,G1,T3,T1.
REVERT.
ENDIF,STILLOK.
EXIT.
RETURN,M1.
RETURN,G1.
REVERT.
--EOR--
```

Figure 15.- PROC for step HOS.

Normal execution of step HOS produces six files as output. They are the Plot Data file T3$, the HODAC Data files M1$, G1$, T1$ and T2$, and the printer output file $O. Two direct access scratch files, TAPE7 and TAPE8, are defined and used by HOS but are not saved. Figure 16 shows a sample dayfile from a typical HOS execution.

Files T1$, T2$, T3$, M1$ and G1$ are direct access files that are created and saved by step HOS. See Appendix B of this document for a discussion of the contents of each of these files.

Output file $O and dayfile $D become the printed simulation history. The output file $O consists of three things: a listing of instruction array (input file D1), a listing of the Crew Data File (input file CREW), and the step-by-step simulation history produced by the execution of the simulation. A section of a typical HOS output, a discrete simulation history, is shown in figure 17. The simulation history is a complex file containing, among other things, a time history of every step of the simulation, an aircraft status update record printed every 10 seconds and printouts of the HOS Dictionary arrays (the complete set of simulation variables) at user specified points and at the end of the simulation.

```
13.54.17.HOS,T5ØØ,CM3ØØØØØ.
13.54.17. ED BOGART
13.54.18.USER.
13.54.18.CHARGE.
13.54.19.ATTACH,XEDIT/UN=LIBRARY.
13.54.21.XEDIT,EXEC,P.;XC/$NAME$/EX/*;XEND.
13.54.35.RETURN,XEDIT.
13.54.35.BEGIN,1HOS,EXEC,LAND.
13.54.37.GET,O1=O1EX.
13.54.48.GET,D1=D1EX.
13.54.49.GET,T30=DATA30,TCVLIB.
13.54.54.GET,WGUSLIB,T45=TAPE45.
13.55.Ø3.IFE,FILE(LAND,AS)=Ø,GOTIT3.
13.55.Ø3.GET,LAND/NA.
13.55.Ø5.ENDIF,GOTIT3.
13.55.Ø5.DEFINE,M1=M1EX/M=W.
13.55.Ø5.DEFINE,T1=T1EX/M=W.
13.55.Ø6.DEFINE,T3=T3EX/M=W.
13.55.Ø6. T3EX NOT FOUND, AT 000121.
13.55.Ø6.DEFINE,T3=T3EX/M=W.
13.55.Ø6.DEFINE,G1=G1EX/M=W.
13.55.Ø6.ATTACH,FTNMLIB/UN=LIBRARY.
13.55.Ø9.LDSET,LIB=FTNMLIB/TCVLIB/WGUSLIB.
13.55.Ø9.LDSET(MAP=SBEX).
13.55.Ø9.LDSET,PRESET=ZERO.
13.55.Ø9.Ø1,INPUT,EXO,TAPE7,TAPE8,T2,D1,M1,G1,LAND,T1,T3,T3Ø,T4Ø,T45.
14.39.11     2274ØØ MAXIMUM EXECUTION FL.
14.39.11     5.649 CP SECONDS EXECUTION TIME.
14.39.11.EXIT.
14.39.12.RETURN,M1.
14.39.12.RETURN,G1.
14.39.12.REVERT.
14.39.12.BEGIN,1EXIT,EXEC.
14.39.13.REWIND,OUTPUT.
14.39.13.SKIPEI,EXO.
14.39.13.COPYEI,OUTPUT,EXO.
14.39.14. EOI ENCOUNTERED.
14.39.14.REWIND,EXD,EXO,OUTPUT.
14.39.15.COPYEI,EXO.
14.39.16. EOI ENCOUNTERED.
14.39.16.REPLACE,EXO.
14.39.35.DAYFILE,EXD.
```

Figure 16.- Dayfile from step HOS.

OPERATOR			BODY RH LH RF LF E	HARDWARE
22.64	ABSORB ROLL INDICATOR			AC UPDATE
			X	
22.80	ROLL INDICATOR	= 2.5		AC UPDATE 2.5
22.81	STEP END			
22.81	ALTER 805			
22.81	ALTER 807			
22.81	ALTER 809			
22.81	END ROLL OUT			
22.81	&STRAIGHT VECTOR			
22.81	STEP STOP			
22.81	ALTER 892			
22.81	IF 896			
22.85	ABSORB LOCALIZER			
			X	
23.09	LOCALIZER	= -.2		AC UPDATE -.2
23.10	&COMMAND GAMMA			
23.13	ABSORB COMMAND GAMMA			
			X	
23.22	COMMAND GAMMA	= .1		AC UPDATE .1
23.24	ABSORB GAMMA WEDGES			
			X	
23.33	GAMMA WEDGES	= .6		AC UPDATE .1
23.33	IF 71			
23.58	FLY TO WAYPOINT			AC UPDATE
23.58	STEP CRIT			
23.58	ALTER 166			
23.58	COMPUTE FLY TO CRIT			
23.65	ABSORB AIRCRAFT POSITION			
			Y	
23.89	AIRCRAFT POSITION	= (-54808.8, -171.2)		AC UPDATE (-54705.0, -168.7)
24.03	ABSORB WAYPOINT 1 POSITION			AC UPDATE
			X	
24.27	WAYPOINT 1 POSITION	= (-55083.3, 0.0)		AC UPDATE (-55084.3, 0.0)
24.28	FLY TO CRIT	= 2.0		
24.28	IF 168			
24.28	COMPUTE DIST TO WAYPOINT			
24.41	ABSORB AIRCRAFT POSITION			AC UPDATE
24.45	AIRCRAFT POSITION	= (-54542.8, -163.8)	X	(-54539.5, -163.9)
24.91	DIST TO WAYPOINT	= 563.8		AC UPDATE
24.91	COMPUTE WAYPOINT CLOSE			
25.01	WAYPOINT CLOSE	= 2000.0		AC UPDATE
25.02	ALTER 172			
25.02	IF 174			
25.09	ABSORB WAYPOINT 1 NAME			
			X	
25.29	WAYPOINT 1 NAME	= QUARY		AC UPDATE QUARY

Figure 17.- Typical output page from step HOS.

3.0 DATA PLOTTING

PLOT is a complex file that contains a submit file (NEWPLOT) with two PROCs (C and SIM), a MODIFY input file (.DATA,SIM) and a plot directive file (.DATA,DATAPLT).

When PLOT is submitted, it gets, modifies and executes PLOTSYS and PLOTLIB.

3.1 Execution of PLOT

Prior to submitting PLOT, DATAPLT should be edited to select the desired data channels, data scaling, and other plot parameters. A careful examination of the sample plot (fig. 18), the data reduction file SIM, and the plot directives contained in DATAPLT will facilitate this. Figure 19 presents a sample dayfile from an execution of PLOT. The plot data file name T3$ in the line:

```
ATTACH,TAPE1Ø=T3$/M=W.
```

must be edited to agree with the name of the plot data file created by HOS (T3EX in the sample dayfile, fig. 19).

Figure 18 is a portion of a plot output generated by PLOT. The variable names and scales are printed on the Y axis and simulation time is printed on the X axis.

Normally, two plot files are maintained to plot HOS output: PTLROLL to plot lateral axis data and PLTPIT to plot longitudinal axis data. The two files are the same except for their names and the data contained in DATAPLT. Each program plots those flight data channels appropriate for that axis.

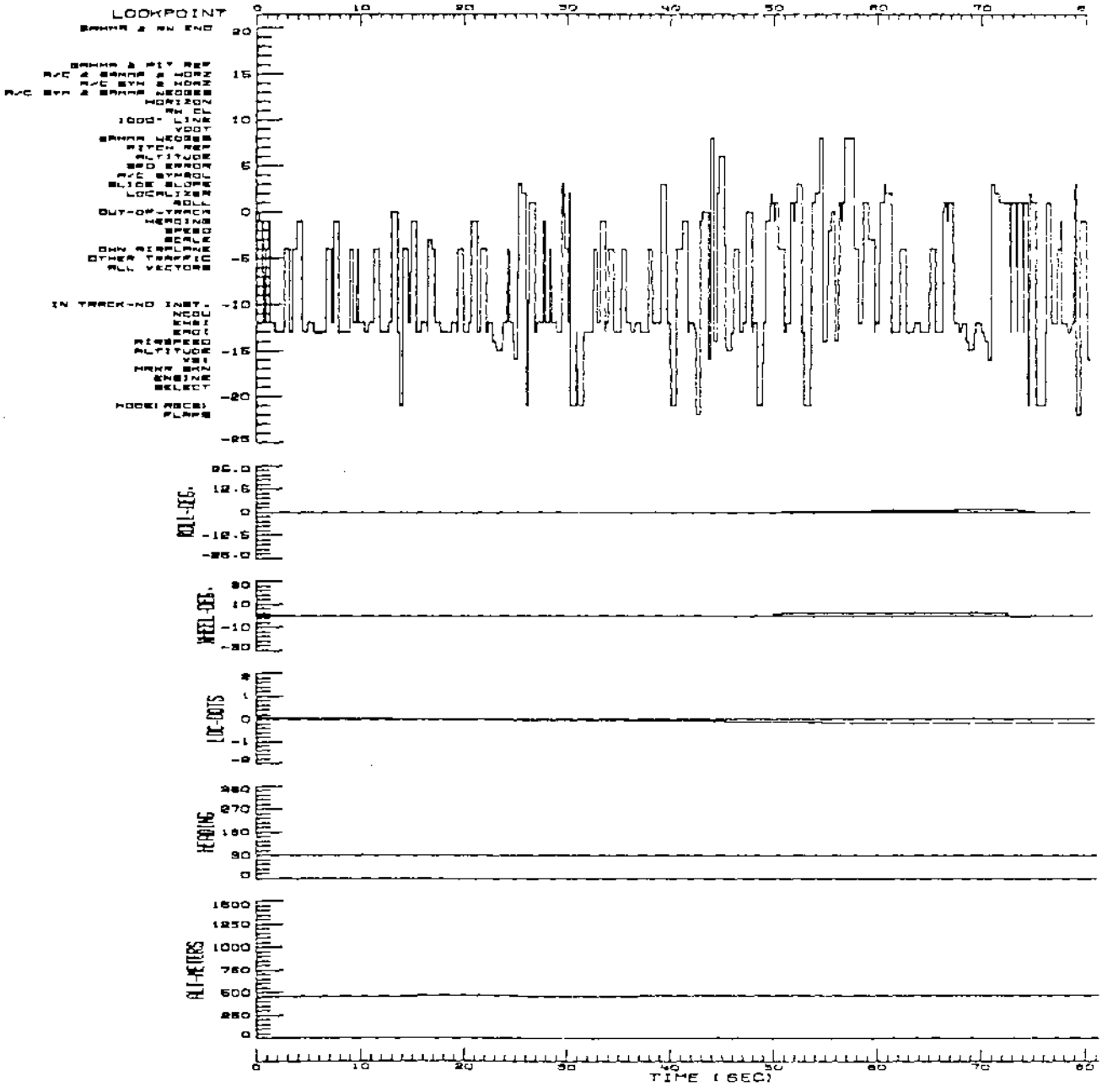


Figure 18.- Typical plot of HOS data.

```
11.38.53.PLOTROL,T2ØØØ,CM1ØØØØØ.          DELIVERY INFORMATION
11.38.53. ED BOGART
11.38.53.USER,XXXXXXX.
11.38.53,CHARGE,######,LRC.
11.38.54.BEGIN,C,PLOTROL,MDS=SIM.
11.38.57.GET,PLOTMOD.
11.38.59.MODIFY,F,I=SIM,P=PLOTMOD.
11.39.Ø1. MODIFICATION COMPLETE.
11.39.Ø1.FTN,I,A,R=3,L=Ø.
11.39.22.      9.Ø5Ø CP SECONDS COMPILATION TIME
11.39.22.REVERT.
11.39.22.BEGIN,SIM,PLOTROL.
11.39.24.ATTACH,TAPE1Ø=T3EX/M=W.
11.39.24.REWIND,TAPE1Ø.
11.39.24.COPYEI,TAPE1Ø,TAPE8.
11.39.28. EOI ENCOUNTERED.
11.39.29.RETURN,TAPE1Ø.
11.39.29.REWIND,TAPE8.
11.39.29.BEGIN,X,PLTROLL.
11.39.31.ATTACH,FTNMLIB/UN=LIBRARY.
11.39.31.ATTACH,LRCGOSF/UN=LIBRARY.
11.39.31.GET,PLOTMOD.
11.39.33.GET,PLOTSYS.
11.39.36.LDSET,LIB=PLOTLIB/PLOTSYS/LRCGOSF/FTNMLIB,PRESETA=NGINF.
11.39.36.LGO,INPCRDS.
11.41.Ø9.     Ø634ØØ MAXIMUM EXECUTION FL.
11.41.Ø9.     49.18Ø CP SECONDS EXECUTION TIME.
11.41.Ø9.RETURN,TAPE8.
11.41.Ø9.PLOT.VARIAN(XM=1.ØØ,YM=.45,YO=7.Ø)
11.41.14.VØØ2
11.41.48.     2 FRAMES/       .89 METERS GENERATED.
11.41.48.PICTURE IMAGE FILE WILL BE SAVED ON DISK
11.41.51. *****  PLOT OUTPUT COMPLETED  *****
11.41.52.REVERT.
11.41.52.REVERT.
11.41.53.DAYFILE,PROLD.
```

Figure 19.- Dayfile from step PLOT.

4.0 HODAC REPORT GENERATION

The Human Operator Data Analysis/Collector (HODAC) is a FORTRAN program that converts data from a HOS simulation into graphs and/or reports suitable for use by a human factors analyst. The user can select from ten report formats. Analytics HODAC User's Guide contains descriptions of the report types available and describes the input directives required for each.

4.1 HALHOD Execution

Two steps are required to produce any HODAC report or graph. The first step, HALHOD (see fig. 20), executes PROC 1HALHOD from file EXEC. PROC 1HALHOD shown in figure 21 gets library file LHODAC as local file P2 and gets the UPDATE directive file C2$ as local file C2. The UPDATE step modifies LHODAC, incorporating the changes contained in C2. The COMPILE file is compiled in the FTN step producing the load file LGO. The new program file N2 is saved as S2$ and LGO is saved as O2$.

```
/JOB
HALHOD,T3ØØ,CM15ØØØØ.                    DELIVERY INFORMATION
USER,#######.
CHARGE,######,LRC.
ATTACH,XEDIT/UN=LIBRARY.
XEDIT,EXEC,P.;XC/$NAME$/EX/*;XEND.
RETURN,XEDIT.
BEGIN,1HALHOD,EXEC,A1234.
BEGIN,1EXIT,EXEC.
```

Figure 20.- Submit file for step HALHOD.

```
.PROC,1HALHOD,LIST.
GET,C2=C2$NAME$.
GET,P2=LHODAC.
UPDATE,O=Ø,P=P2,I=C2,F,N=N2,L=LIST.
FTN,I=COMPILE,L=Ø,OPT=Ø,R-3,A.
REPLACE,N2=S2$NAME$.
REPLACE,LGO=O2$NAME$.
REVERT.
```

Figure 21.- PROC for step HALHOD.

Control returns to submit file HALHOD (fig. 20) which begins PROC 1EXIT saving $D and $O. If no UPDATE or FTN errors occur during execution of 1HALHOD no output is written into $O and the printed listing for the job will contain only the JOB header and dayfile. Figure 22 is the dayfile from a typical execution of HALHOD.

```
Ø9.35.15.HALHOD,T3ØØ,CM15ØØØØ.          DELIVERY INFORMATION
Ø9.35.15.USER,#####.
Ø9.35.15.CHARGE,#####,LRC.
Ø9.35.15.ATTACH,XEDIT/UN=LIBRARY.
Ø9.35.16.XEDIT,EXEC,P.;XC/$NAME$/EX/*;XEND.
Ø9.35.23.RETURN,XEDIT.
Ø9.35.23.BEGIN,1HALHOD,EXEC,A1234.
Ø9.35.24.GET,C2=C2EX.
Ø9.35.45.GET,P2=LHODAC.
Ø9.35.57.UPDATE,O=Ø,P=P2,I=C2,F,N=N2,L=A1234.
Ø9.36.09. UPDATE COMPLETE.
Ø9.36.09.FTN,I=COMPILE,L=Ø,OPT=Ø,R=3,A.
Ø9.37.08.    45.295 CP SECONDS COMPILATION TIME
Ø9.37.08.REPLACE,N2=S23X.
Ø9.37.17.REPLACE,LGO=02EX.
Ø9.37.25.REVERT.
Ø9.37.25.BEGIN,1EXIT,EXEC.
Ø9.37.26.REWIND,OUTPUT.
Ø9.37.26.SPIPEI,EXO.
Ø9.37.26.COPYEI,OUTPUT,EXO.
Ø9.37.26. EOI ENCOUNTERED.
Ø9.37.26.REWIND,EXD,EXO,OUTPUT.
Ø9.37.26.COPYEI,EXO.
Ø9.37.26. EOI ENCOUNTERED.
Ø9.37.26.REPLACE,EXO.
Ø9.37.26.REPLACE,EXO.
Ø9.37.29.DAYFILE,EXD.
```

Figure 22. Dayfile from step HALHOD

5.2 HODAC Execution

The second step required to generate a HODAC report is the execution of submit file HODAC (shown in fig. 23). When HODAC is submitted, it gets and edits a local copy of file EXEC, replacing all occurrences of the string $NAME$ with the desired file name (the string "EX" in this example). The next command line in HODAC:

```
BEGIN,1HODAC,EXEC,INPUT.
```

begins PROC 1HODAC from the edited copy of EXEC.

```
/JOB
HODAC,T3ØØ,CM15ØØØØ.                DELIVERY INFORMATION
USER,#######.
CHARGE,######,LRC.
ATTACH,XEDIT/UN=LIBRARY.
XEDIT,EXEC,P.;XC/$NAME$/EX/*;XEND.
RETURN,XEDIT.
BEGIN,1HODAC,EXEC,INPUT.
BEGIN,1EXIT,EXEC.
/NOSEQ
/EOR
DEVICES BY PARTS TIMELINE EVERY 5 SECOND:
FROM Ø.ØØ SECONDS TO 3ØØ SECONDS.
LABELS CHANNEL-LOAD EVERY 5 SECOND:
FROM Ø.ØØ SECONDS TO 3ØØ SECONDS.
LINKS SYSTEM CENTER-PANEL = CAS-ENG-DISPLAY THRU TKA-SEL-LIGHT;
SYSTEM EADI = EADI THRU TRACK-POINTER,
    AIRCRAFT-SYMBOL THRU WAYPOINT-5-POSITION;
SYSTEM EHSI = ENHSI THRU WIND-VELOCITY;
SYSTEM PILOTS-PANEL = BARO-ALTIMETER-THOU THRU MIDDLE-BEACON-LIGHT;
SYSTEM CENTER-CONTROLS = VEL-CWS-SWITCH THRU VERT-PATH-SWITCH,
  WPT-ALT THRU TRACK-UP-SWITCH,
  SPEED-BRAKE-LEVER THRU AUTO-MANUAL-SWITCH;
SYSTEM PILOTS-CONTROLS = BROLLY-PITCH THRU START-SIMULATION:
FROM Ø.ØØ SECONDS TO 3ØØ SECONDS.
/EOF
```

Figure 23.- Submit file for step HODAC with appended HODAC directions.

PROC 1HODAC (shown in fig. 24) gets the execution load module O2$, created by step HALHOD, as local file O2. It gets the pseudo-machine-instruction file D2$, created by HAL, as local file D2. The HODAC directives are passed as file INPUT.

```
.PROC,1HODAC,IFILE.
GET,D2=D2$NAME$.
GET,O2=O2$NAME$.
IFE,FILE(IFILE,AS)=Ø,SKIP1.
GET,IFILE.
ENDIF,SKIP1.
ATTACH,M1=M1$NAME$.
ATTACH,G1=G1$NAME$.
LDSET,PRESET=ZERO.
O2,D2,G1,M1;IFILE,$NAME$O.
RETURN,M1.
RETURN,G1.
REVERT.
EXIT.
REVERT.
--EOR--
```

Figure 24.- PROC for step HODAC.

Two direct access data files, M1$ and G1$, both created during execution of the HOS step, are attached by 1HODAC. File M1$ is a binary version of the simulation output listing but without any headings, error messages, or pagination. M1$ is read as input data during execution of 02. File G1$ is a binary file that contains plot data. It is read as input by 02 if a Time Graph is being produced.

When execution of 02 is complete, direct access files M1 and G1 are returned and control returns to submit file HODAC. HODAC begins 1EXIT which copies the dayfile to $D, copies $O and the dayfile to OUTPUT, and saves $O and $D.

Figure 25 shows a typical page of HODAC output. It is the first page of the Device-by-Parts-Timeline report created by the first command line in the HODAC Command File

DEVICES BY PARTS TIMELINE EVERY 5 SECONDS

shown in figure 23.

REFERENCES

1. Glenn, F. A., III, Doane, S. M.: A Human Operator Simulator Model of the NASA Terminal Configured Vehicle (TCV). NASA-CR 15983, May 1981.

2. Anon: Terminal Configured Vehicle Program Test Facilities Guide. NASA-SP 435, 1980.

83/03/04. 2/23/93 FINAL (2) VCWS MODE PAGE 1

HODAC BODY PART TIMELINE ANALYSIS (5.0 SECOND SNAPSHOTS)

TIME	EXECUTING	EYES ARE	RIGHT HAND IS	LEFT HAND IS	RIGHT FOOT IS	LEFT FOOT IS
0.0	FLY TO WAYPOINT	ABSORBING FROM MAP SCALES	MANIPULATING CAS ENG KNOB	MANIPULATING START SIMULATION		
5.0	PREVIEW MONITORS	ABSORBING FROM WAYPOINT 5 POSITION	RELAXING			
10.0	STRAIGHT VECTOR	ABSORBING FROM AIRCRAFT PITCH	. . . . .	MANIPULATING BROLLY ROLL		
15.0	. . . . .	ABSORBING FROM TRACK BOX	. . . . .			
20.0	ROLL OUT	ABSORBING FROM WAYPOINT 1 POSITION	. . . . .	MANIPULATING BROLLY ROLL		
25.0	FLY TO WAYPOINT	ABSORBING FROM WAYPOINT 2 SPEED	MANIPULATING CAS ENG KNOB			
30.0	AIRSPEED CHECK	ABSORBING FROM VSI	RELAXING			
35.0	STRAIGHT VECTOR	ABSORBING FROM AIRCRAFT POSITION	. . . . .			
40.0	. . . . .	ABSORBING FROM AIRCRAFT PITCH	. . . . .			
45.0	FLY TO WAYPOINT	ABSORBING FROM GROUND SPEED	. . . . .			
50.0	. . . . .	ABSORBING FROM WAYPOINT 3 SPEED	MOVING TO CAS ENG KNOB			
55.0	AIRSPEED CHECK	ABSORBING FROM AIRCRAFT PITCH	MANIPULATING CAS ENG KNOB			
60.0	STRAIGHT VECTOR	ABSORBING FROM WAYPOINT 3 POSITION	RELAXING	MANIPULATING BROLLY ROLL		
65.0	. . . . .	ABSORBING FROM TRACK BOX	. . . . .	. . . . .		
70.0	AIRSPEED CHECK	ABSORBING FROM WIND VELOCITY	. . . . .			
75.0	FLY TO WAYPOINT	ABSORBING FROM AIRCRAFT POSITION	. . . . .			
80.0	STRAIGHT VECTOR	ABSORBING FROM WAYPOINT 3 POSITION	. . . . .	MANIPULATING BROLLY ROLL		

Figure 25.- Typical output page from HODAC timeline analysis report.

APPENDIX A

STRAIGHT-IN APPROACH SCENARIO

The straight-in approach scenario (fig. 26) presents the nominal flight path for a straight-in Instrumentation Landing System (ILS) approach with the locations of the waypoints marked and with altitude and airspeed schedules shown.

The following description of the tasks was taken from NASA CR 3421 since it provided an appropriate description of the sequence of activities being simulated.

Flight From Waypoint MERCI to Waypoint QUARY:

The aircraft begins the simulation at an altitude of 151Ø feet, an airspeed of 21Ø knots, with flaps set at 1°, and holding a flight path angle of Ø.Ø°. Immediately after passing waypoint MERCI, the pilot calls for a flap setting of 5° (nominally implemented by the co-pilot, however, implement in this simulation by software since the co-pilot is not actually modeled), and selects an airspeed of 185 knots.

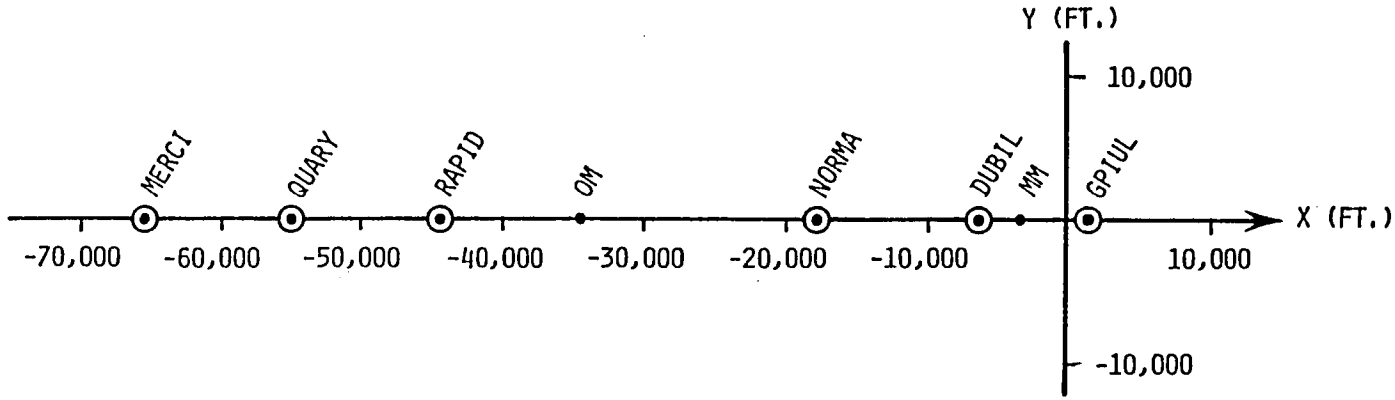
The pilot will monitor the gamma wedges, the track angle pointer, and the localizer deviation indicator. As each waypoint is approached, the pilot monitors the EHSI, airspeed indicator, and the altimeter.

Flight From Waypoint QUARY to Waypoint RAPID:

QUARY is approached at an airspeed of 185 knots and an altitude of 151Ø feet is maintained. Immediately after passing QUARY, the pilot calls for a flap setting of 15° and selects an airspeed of 16Ø knots. He maintains these conditions in straight and level flight through RAPID.

Flight From Waypoint RAPID to Waypoint NORMA:

The pilot selects an airspeed of 14Ø knots, calls for the flaps to be set at 25°, and the landing gear lowered. After passing RAPID he begins monitoring for the outer marker which is indicated by the flashing of the outer beacon indicator light



⊙ - WAYPOINT ON STRAIGHT-IN APPROACH

Figure A1. - Waypoint locations along the Straight-in Approach Path.

(the audio-signal is not currently simulated in the model). Upon reaching the outer marker, he begins monitoring the glideslope indicator, when the glideslope indicator is approximately centered, he changes the flight path angle to -3° with a column input.

During this phase of the flight, the pilot arms the speed brakes and calls for "landing checklist to flaps." Continued attention is given to the localizer deviation indicator, glideslope deviation indicator, airspeed indicator, EHSI, and IVSI.

Flight From Waypoint NORMA to Waypoint DUBIL:

Flying at 14Ø knots, the aircraft passes waypoint NORMA. The pilot calls for a flap setting of 3Ø° and selects an airspeed of 13Ø knots. The same instruments are monitored as between RAPID and NORMA, with a target altitude of 384 feet at waypoint DUBIL.

Flying at 13Ø knots, descent is continued while the pilot visually monitors for the middle marker signal which should occur at 188 feet altitude. Maintaining an airspeed of 13Ø knots, the pilot calls for completion of the landing checklist and a flap setting of 4Ø°.

APPENDIX B

FILE DESCRIPTIONS

All files required to execute a TSRV flight simulation are maintained in their current form on a master magnetic tape and also in the LaRC permanent file system. In the following discussion, the names of files and the names of CDC utility programs are printed in upper case characters for clarity. The names of files created and used during the simulation process are made up of a variable part consisting of one to four characters, used to identify the particular simulation run conditions, and a prefix or suffix of one to three characters identifying the function of the file in the execution of the simulation. As in the main text, the names of files used during the simulation will be referenced with the fixed characters in upper case and the variable characters represented by a dollar sign ($).

Several types of files are used during the simulation. These include a procedure file (EXEC), several submit files that execute sequences of procedures from EXEC, three large FORTRAN programs in UPDATE format (UPDATE is a NOS utility used to modify and maintain files), several supporting UPDATE input files and input data files. This appendix presents a verbal description of each of these files.

EXEC is the procedure file comprised of all of the PROC's used in the normal execution of a HOS simulation. Each PROC combines a number of Cyber Control Language (CCL) statements that can be executed by a single command in a submit file. In this document, the PROC for each step is referenced by separate submit files. In normal operation of a complete simulation, several or all of the necessary PROC's could be initiatd by a sequence of commands in a single submit file. EXEC consists of the following procedures:

```
1CLEAR
1HAL
1HALHOS
1HOSMOD
1HOS
1HALHOD
1HODAC
1EXIT.
```

The action of a PROC is transparent to the user during normal use of the HOS simulation program. Only when changes to HOS execution are desired is it necessary to modify an individual PROC. For example, a listing of the FORTRAN program LHOS can be obtained by changing the compile step in PROC 1HALHOS. The compile or FTN step which now specifies no listing, (L=Ø) can be changed to L=OUTPUT.

File CLEAR is a submit file containing NOS commands required to execute step CLEAR which purges all old output files remaining from a previous simulation that used the same file name variable part. It gets and edits a local copy of the PROC file EXEC and begins PROC 1CLEAR and PROC 1EXIT.

File HAL is a submit file containing NOS commands required to execute step HAL and an input data file in HOPROC code. The file gets and edits a local copy of the PROC file EXEC, begins PROC 1HAL passing the HOPROC input file as file INPUT, and begins PROC 1EXIT.

File HALHOS is a submit file containing NOS commands required to complete step HALHOS. It gets and edits a local copy of PROC file EXEC and begins PROC 1HALHOS and PROC 1EXIT.

File HOSMOD is a submit file containing NOS commands required to complete step HOSMOD. It gets and edits a local copy of PROC file EXEC and begins PROC 1HOSMOD and PROC 1EXIT.

File HOS is a submit file containing NOS commands required to complete step HOS. It gets and edits a local copy of PROC file EXEC and begins PROC 1HOS and PROC 1EXIT.

File HALHOD is a submit file containing NOS commands required to complete step HALHOD. It gets and edits a local copy of PROC file EXEC and begins PROC lHALHOD and PROC lEXIT.

File HODAC is a submit file containing NOS commands required to complete step HODAC and a file of HODAC report directives. It gets and edits a local copy of EXEC, begins PROC lHODAC with the name INPUT passed as the directive file name and it begins PROC lEXIT.

File PLOT contains the submit file that executes a plot of HOS output data, three PROC's (PROC's C, SIM, and X) that carry out various steps in the plot execution, an UPDATE directive file (DATA SIM) and a plot specification data file (DATA INPCRDS). PLOT produces output on a VARIAN plotter using files PLOTSYS, PLOTMOD, and PLOTLIB.

File PLOTSYS contains the executable load module of the plot program.

File PLOTMOD is a FORTRAN source program in UPDATE format. After an UPDATE and compile, PLOTMOD processes the plot data file from HOS into a form compatible with the plot program PLOTSYS.

File PLOTLIB contains the runtime library used in the execution of the plot program PLOTSYS.

File LHALO is the binary load module for the HOS program named HAL. In normal operation, LHALO is never modified by the user.

LHALO translates statements written in the Human Operator Procedures (HOPROC) language into CDC FORTRAN IV extended statements. Array sizes, variable labels and titles, and function entry points are calculated and used to generate the variable COMMON blocks and EQUIVALENCE's for program HOS. LHALO saves these in the form of UPDATE directive files C1$ and C2$. The procedures are translated into pseudo-machine code binary form and are saved as filed D1$ and D2$.

(See Analytics T.R. 14ØØ.22 for a detailed description of the structure and operation of LHALO.)

File LHOS is a large FORTRAN IV Extended source program in UPDATE format. It contains the generic portion of the Human Operator Simulator (HOS) program, and all the subroutines and functions necessary for simulation. LHOS is loaded as a local file S1 by PROC 1HALHOS. The UPDATE steps in PROC 1HALHOS add variable COMMONS to the subroutines, add subroutines HFUNC and MFUNC and add the aircraft modeling subroutines from B737PLA. Optional UPDATE steps in PROC 1HOSMOD make more operational changes. The result of these steps, file COMPILE, is compiled into 01$, the HOS execution load module.

(See Analytics T.R. 14ØØ.22 for a detailed description of the subroutines and variables in LHOS.)

File B737PLA is a FORTRAN program in UPDATE format. It models the flight dynamics of the Boeing 737 aircraft as configured for the NASA/LaRC T.C.V. It is loaded as file PLA by PROC 1HALHOS, UPDATE'd and added into HOS source file.

(Execution of B737PLA requires the presence of three other files; the library files TCVLIB and WGUSLIB and the data file TAPE45.)

File WGUSLIB is part of the B737-1ØØ simulator file library. It contains a number of functions used by B737PLA in the computations concerned with the atmosphere and winds. PROC 1HOS gets WGUSLIB and the first LDSET line:

```
LDSET,LIB=FTNMLIB/TCVLIB/WGUSLIB.
```

makes it part of the runtime library for the execution of load module 01$.

File TAPE45 is part of the B737-1ØØ simulator file library. It contains atmosphere and wind data used as input data for wind and atmospheric calculations.

File TCVLIB is a library file that contains run time subroutines and functions called by B737 subroutines. The first LDSET line in PROC 1HOS:

```
LDSET,LIB=FTNMLIB/TCVLIB/WGUSLIB.
```

makes TCVLIB part of the runtime library for the execution of load module 01$.

File DATA30 contains aircraft data which is used as input for the B737-1ØØ subroutines.

The files B737PLA, WGUSLIB, TAPE45, and TCVLIB are maintained in the LaRC central computer complex and local copies can be obtained by executing the following GET commands interactively or from BATCH.

```
GET,B737PLA,WGUSLIB,TALE45/UN=24925ØC.
GET,TAPE3Ø=DATA3Ø,TCVLIB/UN=11953ØC.
```

(See Sperry Systems Management Report No. SP71Ø-Ø21, dated March 1981, for a detailed description of the structures and functions of these files.)

CREWSTATION Data File is an input data file for the execution of HOS. It contains all the variable parameter data that are used to define the pilot, the aircraft and its characteristics, and scenario to be followed during the flight simulation. The file is divided into sections corresponding to the sections of the dictionary listing and data entry forms generated by step HAL. The divisions are:

DISPLAY SECTION

CONTROL SECTION

SYMBOL SECTION

OPERATOR FUNCTIONS

MODE SPECIFICATIONS

HUMAN OPERATOR SPEX

Each of these sections consists of dictionary names followed by lists of variable parameters that define the location, type, initial values, and other characteristics of the variables.

Two crew files are standard with the TCV version of HOS: LAND and CRUISE.

File LAND, used in all the examples in this document, defines a simulation of a straight-in approach and landing starting about 65 thousand feet from the runway threshold.

File CRUISE defines a simulation scenario for a curved, descending approach and landing starting from south of the runway threshold.

The following files contain data that are specific to a particular simulation scenario or condition. They, therefore, may change with each execution of a simulation. Some are created and/or modified by the operator prior to execution and contain variable data for the simulation. Others are output files that are recreated with each run and reflect changes in input data.

UPDATE Directive File $MOD contains modifications to be made in the FORTRAN program LHOS. These are user generated changes to the operation of HOS. This includes setting the number of lines per page of printed output and setting the minimum time increment for simulation update. The name of $MOD is passed from submit file HALHOS to PROC 1HALHOS and replaces the name IFILE.

UPDATE Directive File H7MOD used in step HOSMOD contains the modifications to be made in the FORTRAN source file S1$. The complete name of H7MOD is passed by submit file HOSMOD to PROC 1HOSMOD and replaces the dummy variable name IFILE.

File C1$ is an UPDATE directive file generated by the execution of LHALO in PROC 1HAL. It contains the HOPROC statements after conversion into CDC FORTRAN IV source statements. File C1$ is used as the directive file in the third UPDATE step of PROC 1HALHOS. The code in C1$ is inserted into the source code of file LHOS and becomes the subroutines HFUNC and MFUNC.

File C2$ is an update directive file generated in the execution of LHALO by PROC 1HAL. It contains the functions from file HOPROC after conversion by LHALO into FORTRAN.

The UPDATE step in PROC 1HALHOD uses C2$ as its directive file and inserts these source statements into file LHODAC, creating the UPDATE New Program Library file N2 and the compiler source file COMPILE. These files are part of the group of files

necessary for the production of the HODAC reports discussed in section 5.0 of this document.

File 01$ is the binary load module that is executed in step HOS. It is the compiled version of the FORTRAN source program file $C1. It is created by step HALHOS where COMPILE, generated by the last update step, is compiled and LGO is saved as 01$. If optional step HOSMOD is executed, all subroutines that are changed in COMPILE are recompiled. The new LGO code is used to replace the corresponding subroutines code in 01 using LIBEDIT. The new version of 01 is saved as 01$.

File 02$ is a binary load module that is executed in step HODAC. It is the compiled LGO file generated by the FTN line of step HALHOD. It is comprised of the FORTRAN source program LHODAC modified by the UPDATE directives in file C2$. File 02$ is a compiled version of N2$.

File N2$ is an UPDATE New Program Library file made up of the permanent file LHODAC and the modifications from the directive file C2$. It is the FORTRAN source file (in UPDATE format) for load file 02$. N2$ is saved at the completion of step HALHOD.

File S1$ is an UPDATE New Program Library (NPL) file created by the last UPDATE step in PROC,1HALHOS. It contains (in UPDATE format) the complete source code for the execution of HOS. This is comprised of the permanent file LHOS plus the compiled code from file HOPROC plus the aircraft simulation program B737PLA and the modifications in the directive files $MOD and B737MOD.

File S2$ is an UPDATE New Program Library file created by the optional step HOSMOD. It is the file S1$ edited according to the directives in the directive file $H7M.

File D1$ contains Hardware Procedure and Operator Procedure code from HOPROC that has been converted by HAL into pseudo-machine instructions. It is stored in octal format. File D1$ is read by HOS and becomes the instruction arrays HINSHW and HINSOP. (See Analytics T.R. 14ØØ.22B for more information.)

File D2$ is a data file generated by the execution of LHALO in step HAL. It is read as input data in step HODAC. D2$ contains information on dictionary titles and their associated data arrays. The information is stored in a packed binary format. (See Analytics T.R. 14ØØ.22D, pp. 1-3 for more information on the data and the storage format.)

File G1$ contains packed binary plotting data for the Time Graph option of HODAC. It is generated by the execution of O1$ in step HOS and contains a time history of the simulation, consisting of simulation times, activity and device identification codes and information on actual values, estimated values, desired values, and limits for specified HOS variables. (See Analytics T.R. 14ØØ.22D for more details of the record format.)

File M1$ is a data file generated in step HOS during the execution of O1$. It contains a record of all the actions that have occurred during the execution of the simulation. This is the same information that is output by HOS as file $O but is stored in packed, 6Ø bit, binary words. Each word contains the record of one action and consists of simulation time, an action identification code, and a dictionary reference number. (See Analytics T.R. 14ØØ.22D, pp. 1-5 for more information.) M1$ is read as input during the execution of the report generation step HODAC.

File T3$ is a direct-access plot data file created during the execution of O1$ by PROC,1HOS. At each step in a simulation, the current values of simulation variables are loaded into array VSTATE in subroutine HFUNC and written to the plot data file T3$ by subroutine LPTAPE. File T3$ may be as much as 25 thousand records long, depending on the length of the simulation.

File T3$ consists of one header record followed by a number of data records. The storage protocol is:

One header record with the following information:

1. Identification number for this run (integer)
2. Number of data channels (integer)
3. Names of each data channel (holerith)

4. Units for each data channel (holerith)
5. Run title (eight holerith words)

A number of data records of the following form:

1. Time (seconds)
2. X lookpoint coordinate (inches)
3. Y lookpoint coordinate (inches)
4. Stick trim (blank or not used)
5. Stick displacement (inches)
6. Wheel displacement (degrees)
7. Throttle lever position (degrees)
8. Rudder pedal (degrees)
9. Altitude (feet)
10. Glideslope deviation (dots)
11. Localizer deviation (dots)
12. Airspeed (knots)
13. Altitude rate (feet per second)
14. X position of aircraft with reference to runway (feet)
15. Y position of aircraft with reference to runway (feet)
16. Pitch attitude (degrees)
17. Roll attitude (degrees)
18. Yaw attitude (degrees)
19. Pitch rate (degress per second)
20. Roll rate (degrees per second)
21. Yaw rate (degrees per second)
22. Instrument lookpoint where:

    1 = other instruments
    2 = EHSI
    3 = EADI

23. An integer designation for each point within the EHSI, the EADI, or other instruments using the following codes:

    1. For "other instruments" (the preceding word, #22, has value 1).

        1. NCDU
        2. EHSI
        3. EADI
        4. Airspeed
        5. Altitude
        6. VSI
        7. Outer beacon
        8. Middle beacon
        9. Engine
        10. Select
        11. AGCS mode
        12. Flap

    2. For EHSI (word #22 has value 2)

        1. Heading
        2. Speed

3. Scale
4. Own airplane
5. Other traffic
6. All vectors

3. For EADI (word #22 has value 3)

1. Roll
2. Localizer
3. Glide slope
4. Aircraft symbol
5. Speed error
6. Altitude
7. Pitch reference
8. Gamma wedges
9. VDOT
10. Runway 1,ØØØ ft. line
11. Runway centerline
12. Horizon
13. Aircraft + gamma
14. Aircraft + horizon
15. Aircraft + gamma + horizon
16. Gamma + pitch reference
17. Gamma + runway end
18. Gamma + runway 1,ØØØ ft. line
19. Runway end
20. Gamma + horizon
21. Track symbol
22. Pitch scale

Data channels can be removed from or added to T3$ as required. A careful examination of HOS subroutines HFUNC and LPTAPE and the corresponding data input and data manipulation portions of file PLOT should enable the change.

File $O is a saved copy of the printed OUTPUT file. It is available for examination from the terminal as soon as job execution is completed. In some cases, file O$ may exceed the maximum length for files that can be saved by the system. This will not adversely affect the printing of the output or the execution of the balance of the steps.

File $D contains a saved copy of the job execution DAYFILE. It is available for examination from a terminal immediately after execution is completed.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-86367		

4. Title and Subtitle	5. Report Date
Users Guide: The LaRC Human-Operator-Simulator-Based Pilot Model	April 1985
	6. Performing Organization Code: 505-35-33-01
7. Author(s): Edward H. Bogart and Marvin C. Waller	8. Performing Organization Report No.
	10. Work Unit No.
9. Performing Organization Name and Address: NASA Langley Research Center, Hampton, VA 23665	11. Contract or Grant No.
	13. Type of Report and Period Covered: Technical Memorandum
12. Sponsoring Agency Name and Address: National Aeronautics and Space Administration, Washington, DC 20546	14. Sponsoring Agency Code

15. Supplementary Notes

Edward H. Bogart: Kentron International, Inc., Hampton, Virginia.
Marvin C. Waller: NASA Langley Research Center, Hampton, Virginia.

16. Abstract

A Human Operator Simulator (HOS) based pilot model has been developed for use at NASA LaRC for analysis of flight management problems. The model is currently configured to simulate piloted flight of an advanced transport airplane. The generic HOS operator and machine model was originally developed under U.S. Navy sponsorship by Analytics, Inc. and through a contract with LaRC was configured to represent a pilot flying a transport airplane.

A version of the HOS program runs in batch mode on LaRC's (60-bit-word) central computer system. This document provides a guide for using the program and describes in some detail the assortment of files used during its operation.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Pilot model Operator model HOS	Unclassified - Unlimited Subject Category 53

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	46	A03

N-305

For sale by the National Technical Information Service, Springfield, Virginia 22161

DO NOT REMOVE SLIP FROM MATERIAL

Delete your name from this slip when returning material to the library.

NAME	DATE	MS
~~P Arbuckle~~ cancel THIS copy of THIS	~~12-1-92~~	~~156A~~
~~Joey Brandon~~	~~11/05~~	~~153~~

NASA Langley (Rev. Dec. 1991) RIAD N-75