NASA Contractor Report 165802

NASA-CR-165802
19820023894

NECAP - NASA'S ENERGY COST ANALYSIS PROGRAM - Operations Manual

David L. Miner
Computer Sciences Corporation
Hampton, Virginia

Contract NAS1-16078
June 1982

National Aeronautics and
Space Administration
Langley Research Center
Hampton, Virginia 23665

FOREWORD

NECAP has been run on NASA Langley's CDC Cyber 170 series computers. Field length of 270K and CPU time of 3000 seconds are typical when running NECAP. Engineers using NECAP could find that specific needs for larger studies may require modification to the program dimensions. The NECAP USER'S MANUAL lists the current program array sizes.

Additions or modifications may be made to NECAP to keep pace with changes or discoveries in the energy field. NASA is continuing work to upgrade certain areas of the program and is conducting verification studies. Additionally, modifications and enhancements may be made to the computer equipment or its operating system at Langley which might require changes in the method of NECAP operation. This document explains the use of NECAP 4.1 on the NOS 1.4 operating system.

This manual is intended to outline basic information required to run NECAP on the CDC CYBER 170 Series computers at Langley Research Center. Information about the operating system is provided only with respect to NECAP and its execution and may not be suitable for other computer programs. Please note that in the examples of this document, all underlined parameters represent user defined information.

Operational guidance or assistance to non-government agencies in running the program cannot be provided. If funds and time permit, a review will be conducted and updating of the program and its manuals may be made.

Use of trade names or names of manufactures in this report does not constitute an official endorsement of such products or manufacturers, either expressed or implied, by the National Aeronautics and Space Administration.

TABLE OF CONTENTS

TABLE OF CONTENTS
(continued)

INTRODUCTION

In 1975, NASA published NASA's Energy Cost Analysis Program (NECAP). The program is documented in two manuals, Part I entitled Users Manual and Part II entitled Engineering Manual (see contractor report No. CR-2590). The algorithms used for determining thermal loads are from the American Society of Heating, Refrigerating, and Air-Conditioning Engineers, Inc., (see "Procedures for Determining Heating and Cooling Loads for Computerized Energy Calculation"). NASA uses the program for building heating and cooling design loads and energy analysis. NECAP has been used as a reference for development of several other computerized energy programs.

NECAP is maintained by Computer Sciences Corporation (CSC) for NASA's Langley Research Center (NAS1-16078). NECAP has been maintained at Langley Research Center since its installation in 1975. Program modifications which were performed by CSC are inculded in the NECAP ENGINEERING MANUAL (NASA TM-83240) and the NECAP USER'S MANUAL (NASA TM-83238).

Other items performed under the maintenance of NECAP include:

- Development of NECAP WEATHER DATA BASE
- Development of NECAP PROCEDURE FILES
- Validation of NECAP ENHANCEMENTS

NECAP 4.1 is documented in the following manuals:

TM 83238 NECAP Users Manual - Describes the input procedures, provides examples and output from the program.

TM 83239 NECAP Input Manual - Details the input requirements.

TM 83240 NECAP Engineering Manual - Provides the algorithms for the program.

TM 83241 NECAP Fast Input Manual and Example - Provides a simple method of preparing NECAP input.

TM 83242 NECAP Engineering Flowcharts Manual - Provides flowcharts of routines outlined in the Engineering Manual.

CR-165802 NECAP Operations Manual - Provides specific operating instruction for CDC computer system operation of NECAP at Langley Research Center.

SECTION II

NECAP PROGRAMS AVAILABLE AT LARC

2.1 NECAP 4.1

NECAP is composed of four individual computer programs. The new program NECAP 4.1 is the major system used at LaRC. The three main NECAP modules are described below.

1. NIPP - NECAP INPUT PROCESSOR PROGRAM
2. TLAP - THERMAL LOADS ANALYSIS PROGRAM
3. SESP - SYSTEMS ENERGY SIMULATION PROGRAM

These modules use the newer card format, which provide increased default capabilities, then performs Design day, Hourly loads, System Simulation, Energy Usage, and Economic analysis calculations.

A flow chart of NECAP 4.1. is shown in Figure 1.

NASA's ENERGY ANALYSIS PROGRAM
Version 4.1

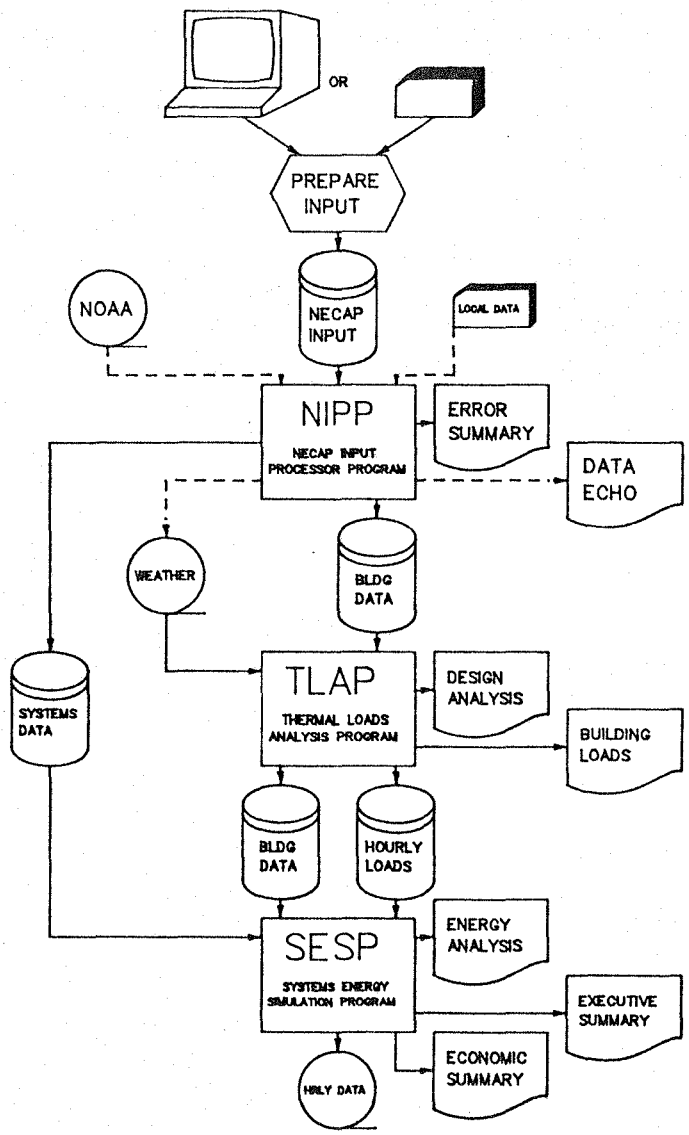


Figure 1

2.2 NECAP INPUT PROCESSOR PROGRAM (NIPP)

NIPP uses the free formatted input for cards as documented in NASA TM 83238 "NECAP Users Manual" and NASA TM 83239 "NECAP Input Manual." NIPP converts the flexible input to the rigid input format required for the TLAP and SESP programs listed below. NIPP provides default values and basic data required by TLAP and SESP. NIPP is also used to verify the formatted input to both the TLAP and SESP programs that were prepared independent of the NECAP INPUT PROCESSOR PROGRAM.

2.3 THERMAL LOAD ANALYSIS PROGRAM (TLAP)

TLAP performs hourly transient heat transfer calculations for each building space using acutal hourly weather recorded from various sources, the modeled geometry, scheduled internal loads, and construction of the building.

2.4 SYSTEMS ENERGY SIMULATION PROGRAM (SESP)

SESP performs hourly equipment and thermostat simulations. This program incorporates equipment limits to the variable temperature routines based on weather. The thermostats for operation and process loads are simulated by the use of more efficient scheduling techniques. SESP prints an Executive Summary, and prints an Economic Summary.

2.5 OWNING AND OPERATION COST PROGRAM (ECON)

ECON calculates the anticipated uniform annual expenditure to own and operate the building HVAC system(s) for the life expectancy of the building.

2.6 WEATHER PROCESSOR PROGRAM (WEATHER)

WEATHER is used to convert both the NOAA 1440 and the NOAA TEST REFERENCE YEAR (TRY) weather tape formats into the NECAP weather input format.

2.7 RESPONSE FACTOR PROGRAM (RESFAC)

RESFAC is used to generate response factors for wall or roof structures differing from the standard delayed heat transfer surfaces built into NECAP. RESFAC employs a layer by layer description of the surface to calculate a set of response factors required to perform transient heat transfer analysis. This program also exists as a subroutine in the TLAP and SESP programs.

SECTION III

RUNNING NECAP

3.0 NECAP OPERATION

NECAP is stored and run on the CDC (Control Data Corporation) computers at NASA's Langley Research Center. All of the programs outlined in Section II are maintained and updated to keep pace with state-of-the-art techniques in both engineering and computer programming.

NECAP uses various forms of input and output to provide the engineer with numerous options. To use these options the user should have a basic knowledge of NOS (CDC's Network Operating System) and XEDIT (a special CDC editor). Before running a job on the CDC NOS system, the user must secure a user number and a charge number.

A limited explanation of NOS and XEDIT are contained in Sections VIII and IX of this manual. These guides incorporate only those aspects of NOS and XEDIT that are necessary to execute NECAP.

NECAP should be run in batch mode. However, the interactive terminal can be used to prepare and modify the input data. The interactive terminal may be used to set up the job control card stream needed to run NECAP. A simple guide to the terminal is outlined in Section VII.

From this point on it will be assumed that the user has a working knowledge of NOS 1.4, an editor, the interactive terminal, and the NECAP INPUT DATA FORMAT.

3.1 MODES OF NECAP OPERATION

NECAP is accessed via a CYBER Control Language (CCL) procedure file called NPC (NECAP Procedures). The NECAP user must remember the file name (NPC) and the file name of the input data. All of the procedures that run NECAP are stored on the file NPC.

The input data may originate from a card deck which should be prepared and formatted as specified in THE NECAP USER'S MANUAL. Once the data is formatted and punched into cards, the user has the option of either copying the cards onto a permanent file or running NECAP directly from the cards.

When using small to moderate sized input files, it may be more advantageous to enter the NECAP data directly into an input file via the interactive terminal. This will eliminate the time required for keypunch and allows for easier data checking.

3.2 USING NECAP FILES ON THE NOS SYSTEM

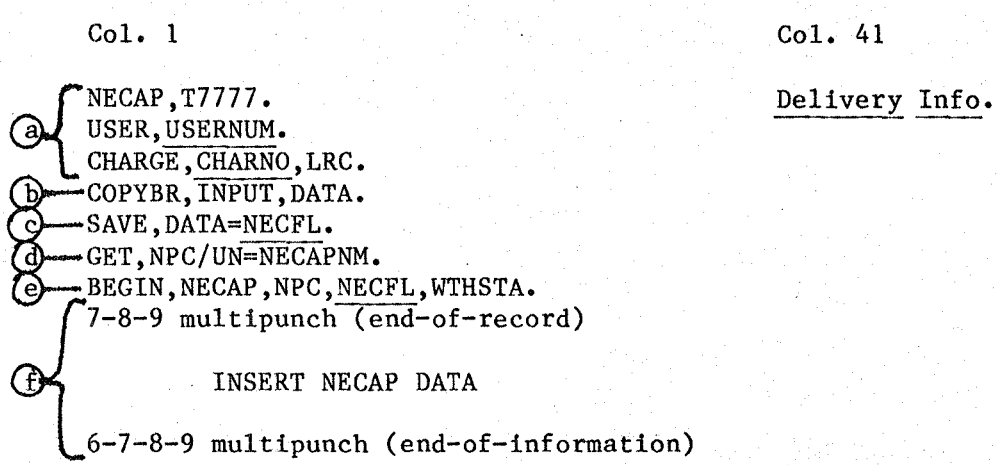
Permanent files containing the NECAP data increase the user's options primarily by offering an interactive or batch working environment. With the aid of an editor, it is possible to prepare and submit several case studies within a short period of time. These files are accessed by NOS control cards which are entered from a terminal or by job submit files.

NECAP operational procedures reside on the permanent file NPC. To access these procedures the user should bring NPC into the local file working space, then activate them with the CCL BEGIN command. The user should have all of the the neccesary files stored as a permanent file on the NOS system. NECAP is not set up to be run interactively, so the user should create a job submit file which can be submitted from an interactive terminal.

3.3 NECAP FROM CARDS

When running NECAP from cards, the user should ensure that the data and job cards are in proper order and format. The main advantage of using card decks is that the run is not dependent on a particular machine. The disadvantage is the handling of cards.

The deck should appear in the following manner:



NECAP 4.1 INPUT CARD DECK

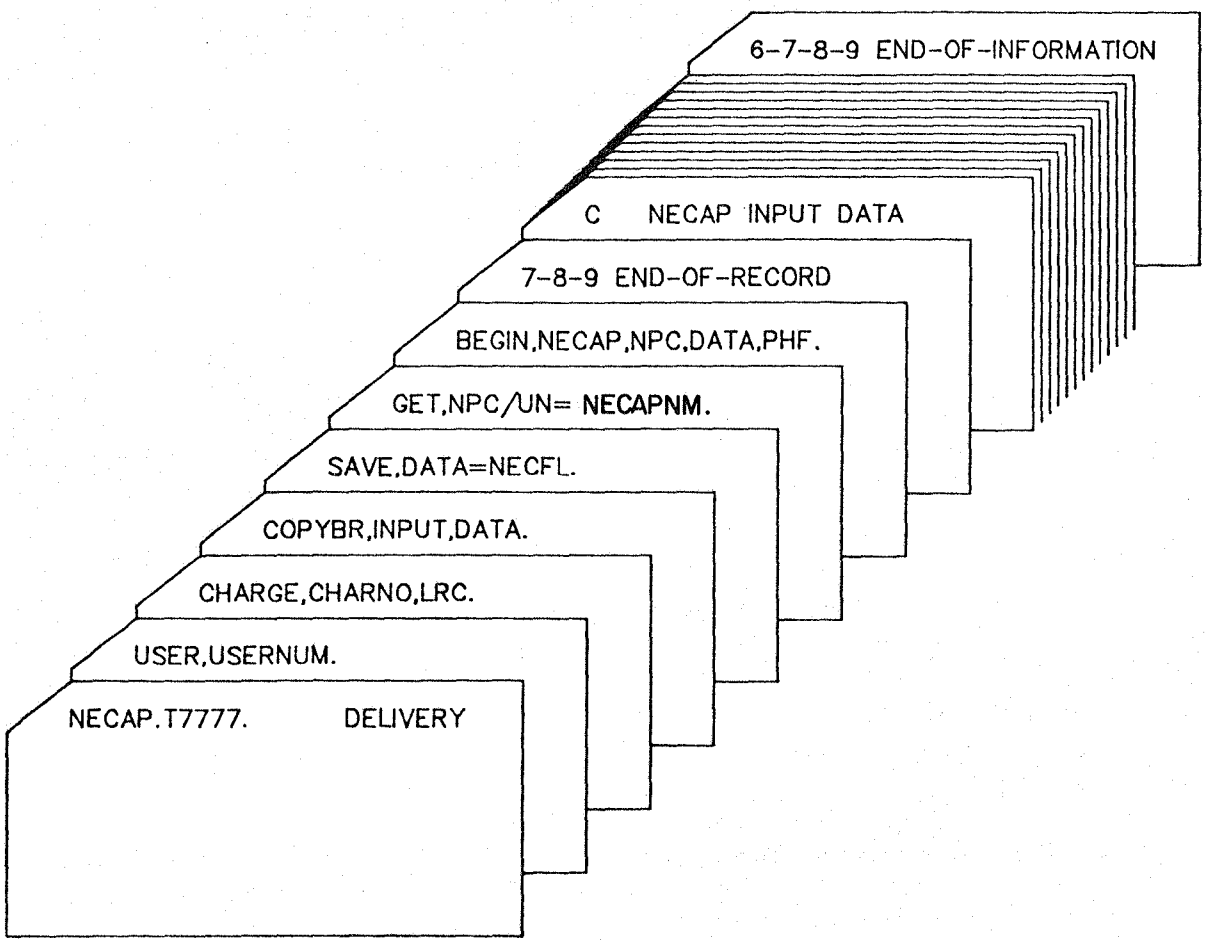


Figure 2

ⓐ Accounting Information

The accounting information is used in monitoring resource costs and file usage. For example, NECAP is the name of the job to be run. The job name can be any given 7 character name.

T7777 gives a timing limit of 7777 decimal seconds. This amount of time is adequate for most runs.

The core space can be specified next but is unnecessary as shown. For running NECAP, 270000 octal words is recommended.

User number identifies the assigned users. This is a number which must be obtained from LaRC computer management.

Charge (number,LRC) is the assigned charge for the computer run. This number must be assigned by computer management and is given to specific user organizations.

The delivery information is also part of this section. This is required and any error will result in job termination with no output. If the delivery information is missing or incorrect, the output will be delayed indefinitely. The delivery information should be as follows:

Col. 41 (7 characters or less) Col. 51

Building number (where your office is) Your name
-or-
Bin number (assigned to your division)

ⓑ Copy in Data

Data on cards is copied from a dedicated local file called INPUT to a local disk file. These local file names are temporary and remain only while the job is in execution.

ⓒ Store Data on System

The local file is saved as a permanent disk file. In order to do this the user must select the file name in alphanumeric characters (1-7). This part is optional but recommended because it is slightly faster to access and execute files from the terminal. This step enables the user to set up files that can be accessed from the terminal.

ⓓ Access the NECAP Procedures

The NECAP procedure file (NPC) is retrieved from storage and brought into the NECAP job stream. NPC remains with the job files to access the appropriate NECAP file for execution.

ⓔ Execute NECAP

The NECAP procedure is activated to execute the NECAP program. This is the standard NECAP 4.1 consisting of the NIPP, TLAP, and SESP programs.

The BEGIN,NECAP,NPC, must be entered to tell the NOS operating system to execute the NECAP procedure on the file NPC.

NECFL is a local file containing the NECAP input data. WTHSTA is the three letter code that tells the procedure NECAP which weather station from the NECAP WEATHER DATA BASE is to be used.

ⓕ Input Data on Cards

The job control cards deck is followed by the NECAP Input deck. The 7-8-9 multipunch cards separates the cards and must follow the job card deck. The 6-7-8-9 card must follow the NECAP input data deck. Correct order of these cards is essential for proper execution.

3.4 SUBMITTING NECAP FROM THE TERMINAL

NECAP may be run as a BATCH job which is submitted via the interactive terminal. The card images are put on a "Job Submit File", rather than a card deck. When the computer receives a job submit file, it treats the file as if it were a card deck. The format of a Job Submit File is the same as a card deck except for the following:

1. A "/JOB" card precedes the first job card.
2. End-of-record (7-8-9 multipunch) is denoted by "/EOR".
3. End-of-information (6-7-8-9 multipunch) is denoted by "/EOF".

A typical NECAP job submit file looks like this:

```
   Col. 1                                          Col. 41

ⓐ  /JOB                                           Delivery Info.
   NECAP,T7777.
ⓑ  USER(USERNUM)
   CHARGE(CHARNO,LRC)
ⓒ  GET(NPC/UN=NECAPNM)
ⓓ  BEGIN,NECAP,NPC,NCFL,WTHSTA.
ⓔ  /EOF
```

ⓐ /JOB Card

The "/JOB" card which signals to the system that this is a job submit file.

ⓑ Accounting Information

The accounting information is used to monitor resource and file usage.

ⓒ Accessing NPC

A copy of NPC is brought into the local file area from the permanent storage.

ⓓ Executing NECAP

NECAP is executed using the specified input files.

BEGIN,NECAP,NPC,p1,p2.

The actual procedures will be explained in Section IV of this document. To initiate the procedure the user must type BEGIN, the name of the procedure, and the name of the file on which that procedure resides, and the name of the file where the data resides.

ⓔ End-of-File

The "/EOF" card is used as a file marker which marks the end of information. Once the job is submitted, the "/EOF" card is converted to a physical end-of-file mark.

SECTION IV

NECAP PROCEDURES

4.1 THE NECAP PROCEDURE FILE

NECAP procedure files are available to give simplicity and depth to NECAP case studies. The user simply needs to know how to access and execute these files. The BEGIN statement activates all of these files and will be demonstrated for each file.

To access the NECAP procedure files the user must enter the following control card:

```
GET,NPC/UN=NECAPNM.
```

This need only be done once per job unless NPC is returned to the system. NECAPNM is the user number under which NECAP is stored. This number will be furnished to the user prior to running NECAP at LaRC.

PROCEDURES ON THE FILE NPC

SECTION	PROCEDURE NAME	PURPOSE
4.2	NECAP	To run standard NECAP
4.3	RUNNIP	To run NIPP as a single program
4.4	RUNTL	To run TLAP as a single program
4.5	RUNSP	To run SESP as a single program
4.6	SAVTAP	To save 'I' and 'A' or 'N' tapes
4.7	USETAP	To retrieve 'I' and 'A' or 'N' tapes
4.8	DAY	To store dayfile on a permanent file
4.9	GETIT	To make sure a file is local

The parameters will be discussed for each file.

4.2 NECAP

PROCEDURE NECAP runs the standard NECAP (Version 4.1) and is maintained to execute all of NECAP with a single card. It will get the user defined input files and activate the procedure for executing each program in the correct order providing defaults where neccesary. To use this procedure the input data should be stored on a permanent file and the weather input selected from the list of available stations given in Section VI. When the data is in order, the user should insert the following cards into the NECAP job stream:

```
GET,NPC/UN=NECAPNM.
BEGIN,NECAP,NPC,NCFL,WTHSTA,LDFL,SPFL,TEST,ECHO.
                ⓐ    ⓑ    ⓒ    ⓓ    ⓔ    ⓕ
```

This procedure will invoke the procedures needed to execute NECAP's components. The user's parameters are defined as follows:

ⓐ NCFL is the input file to NIPP. If a zero is inserted for this parameter, NIPP will start execution at the DATAV Routines. (Defaults to fn=DATA)

ⓑ WTHSTA is the weather station code used to access the NECAP WEATHER data. The codes are given in Section VI. (Defaults to Langley weather)

ⓒ LDFL is the input file to TLAP. If NCFL (a) is not set to zero, then this file will be created by NIPP and left as a local file upon NECAP's completion. If NCFL is set to zero, then the LDFL file will be accessed and used as input to NECAP's TLAP DATA Routines. (Defaults to fn=TAPE7)

ⓓ SPFL is the input file to SESP. This parameter is affected by NCFL (a) in the same manner as LDFL(c) except that it is used for input to NECAP's SESP DATA Routines. (Defaults to fn=TAPE8)

ⓔ TEST is the file that a copy of the dayfile is to be written to. (Defaults to pfn=TEST)

ⓕ ECHO is the flag which generates a NIPP echo of the TLAP & SESP data. If ECHO is set to a number, NIPP will generate an echo. If no echo is desired, let the parameter default. (Defaults to no NIPP ECHO)

The procedure accesses the files as the user specifies; therefore, the file names must be spelled correctly on the BEGIN card and stored under the same user number that is on the USER card.

This procedure also accesses and rewinds all of the files used in a NECAP run. Illustration 1 provides a list of NECAP's files which are used in the standard NECAP execution.

ILLUSTRATION 1 - NECAP 4.1 FILES

PROGRAM	LOCAL FILE NAME	TYPE	ACCESS	READ/WRITE
NIPP	DATA (TAPE4)	F	IA	R
	OUTPUT (TAPE6)	F	LO	W
	TAPE1	U	MT	W
	TAPE2	F	IA	R
	TAPE7	F	LO (IA)	W/R
	TAPE8	F	LO (IA)	W/R
	TAPE9	F	LO	*R/W
	TAPE11	F	MT	R
TLAP	WEATHER (TAPE1)	U	MT	R
	INIDT (TAPE2)	U	LO (MT)	W
	ANADT (TAPE3)	U	LO (MT)	W
	TAPE7 (TAPE5)	F	LO (IA)	R
	OUTPUT (TAPE6)	F	LO	W
SESP	INIDT (TAPE1)	U	LO (MT)	R
	ANADT (TAPE2)	U	LO (MT)	R
	NANDT (TAPE3)	U or F	LO (DA)(MT)	W
	TAPE8 (TAPE5)	F	LO (IA)	R
	OUTPUT (TAPE6)	F	IA	W

KEY:
U = Unformatted Binary
F = Formatted Binary
IA = Indirect Access
MT = Magnetic Tape
LO = Local Disk File
DA = Direct Access

* a scratch file, read in one overlay and written in another

4.3 RUNNIP

PROCEDURE RUNNIP runs the NECAP INPUT from a local file and generates two local files to be used for TLAP and SESP respectively. When running NIPP, do not set the core value to less than 270 thousand octal words. To use the procedure RUNNIP the following cards should be inserted into the jobstream.

```
GET,NPC/UN=NECAPNM.
BEGIN,RUNNIP,NPC,DATA,TAPE7,TAPE8,OUTPUT,ECHO.
                 (a)   (b)   (c)    (d)   (e)
```

where:

(a) DATA = Input data to be run (defaults to lfn=DATA)

(b) TAPE7 = File produced by NIPP to be used as input to TLAP (Defaults to lfn=TAPE7)

(c) TAPE8 = File produced by NIPP to be used as input to SESP (Defaults to lfn=TAPE8)

(d) OUTPUT = Output listing (Defaults to lfn=OUTPUT)

(e) ECHO = ECHO flag. If ECHO flag is set to 1, then NIPP will not echo the formatted TLAP & SESP data (Defaults to ECHO)

4.4 RUNTL

PROCEDURE RUNTL will activate the TLAP program. To use this routine the user must have the DATA file generated by NIPP. RUNTL will access the weather for the user. A list of the available weather stations which are in the NECAP format is contained in Section VI. The DATA file may be either local or permanent.

To run TLAP the user should have NPC as a local file and insert the following card into the job stream:

```
BEGIN,RUNTL,NPC,DATA,WTHSTA.
                (a)  (b)
```

where:

ⓐ DATA is the name of the input to TLAP. Defaults to lfn = DATA.

ⓑ WTHSTA is the weather station code used to access the NECAP weather input into TLAP. The codes are given in Section VI.

* lfn is the local file name.

The procedure will return all of the input data and leave the output data as local files. The output files are assigned these names and should not be changed locally because they are used in other NECAP components. These files are:

OUTPUT: contains the output reports from TLAP.

INIDT: contains the building initialization data.

ANADT: contains the hourly analysis.

4.5 RUNSP (SESP)

PROCEDURE RUNSP is designed to run SESP. SESP is run by using a relocatable binary file that executes SESP, input data generated by NIPP, initialization data (INIDT) and Hourly Analysis (ANADT) generated by TLAP. The binaries and input data need not be local prior to calling this procedure, but INIDT and ANADT must be. To run SESP the file NPC should be local and insert the following card into the jobstream:

BEGIN,RUNSP,NPC,TAPE8,INIDT,ANADT,TAPE3.
ⓐ ⓑ ⓒ ⓓ

where:

ⓐ TAPE8 is the input file to SESP
(defaults to fn=TAPE8)

ⓑ INIDT is the initialization data generated by TLAP
fn=INIDT should be used

ⓒ ANADT is the file containing the Hourly Data generated by TLAP.
fn=ANADT should be used.

ⓓ TAPE3 is an optional output file that can be used for other programs.

The output reports will be written to the OUTPUT file which is automatically sent to the printer.

4.6 SAVTAP

PROCEDURE SAVTAP is a procedure that will write the INIDT and ANADT/NANDT data to magnetic tape to save for later use or multiple SESP runs from a single TLAP run. It may be used to save any data on tape but is designed for files too large to be stored on indirect access files. The files used and produced by NECAP are listed on Illustration 1 and a NECAP Job Flow Diagram is shown in Figure I. To save the data on tape the user must secure a NOS 9-track tape. SAVTAP will access the tape for the user. The user should check to be sure that no one else is using that tape.

To run SAVTAP the user should have the procedure file NPC as a local file and insert the following card:

BEGIN,SAVTAP,NPC,TAPENM,INIDT,ANADT,OTHER.
(a) (b) (c) (d)

where:

(a) TAPENM is the NOS tape number for that tape.

(b) INIDT is the building initialization data.

(c) ANADT is the revised hourly analysis.

(d) OTHER is available for a possible third file.

4.7 USETAP

PROCEDURE USETAP accesses and reads in the data written by SAVTAP. The user must use the same tape number that he used for SAVTAP. If the user cannot remember the tape number, check the dayfile for the run that created the tape. USETAP assigns the tape to the local space and copies it to the specified files. To run USETAP, the procedure file NPC, must be local and insert the following card into the job stream:

BEGIN,USETAP,NPC,TAPENM,INIDT,ANADT,OTHER.
(a) (b) (c) (d)

where:

ⓐ TAPENM is the NOS tape number for that tape.

ⓑ INIDT is the building initialization data.

ⓒ ANADT is the revised hourly analysis data.

ⓓ OTHER is any file written by SAVTAP that may be read in and stored under a local file name.

4.8 DAY

PROCEDURE DAY writes the dayfile of a job execution to a permanent file. This is helpful if the user is sending a job to the 'R', 'T', 'D', or 'Z' machines. This enables the user to monitor the progress of a NECAP run. To use this procedure the procedure file NPC must be local, and the following statement should be placed before or after each of the procedures to be monitored:

```
BEGIN,DAY,NPC,TEST.
          (a)
```

where:

ⓐ TEST is the name of the dayfile. Defaults to 'pfn'* = TEST.

* 'Pfn' is the permanent file name.

4.9 GETIT

PROCEDURE GETIT is designed to insure that a file is in the local working space prior to its use. This procedure is activated for the user by the NECAP procedures.

GETIT checks each of the file systems to see if the file is stored on that system. Once GETIT has found the file it brings it into the job as a local file. The search is done on the following hierarchy, local, permanent indirect access, permanent archive, then the card reader. The file will be rewound and stored under the name given in the BEGIN statement.

If the user wishes to insert GETIT in a special job stream, then NPC should be a local file and the following command entered:

```
BEGIN,GETIT,NPC,FILE.
                (a)
```

where:

(a) FILE is the file to be secured into the local workspace. (This procedure will not bring NPC into local working space.)

SECTION V

LaRC ENHANCEMENTS

5.1 RUNNING NIPP FROM THE TERMINAL

NIPP is too large to be run from an interactive terminal under normal operation. However, at Langley there is a utility available for executing large batch programs with high priority. This is designed to provide interactive priority for large programs.

The procedure is called Express Submit. It resides on LaRC's "D" machine only. The "D" machine is a CDC 6600 which is used primarily for computer aided design. To log onto the "D" machine, use the steps outlined in Section VII, and use "7D" as the resource code.

Once logged onto the "D" machine, the next step is to get the EXPRESS SUBMIT utility from the library. This is done by:

```
GET,XSUBMIT/UN=LIBRARY.
```

The user should have the input on a NOS file. Then a job file should be created to execute the programs. Below is an example of how the XSUBMIT job file should be set up.

```
GO,T277,CM270000.
USER(usernum)
CHARGE(charge,LRC)
GET,NPC/UN=NECAPNM.
BEGIN,PREP,NPC,DATA,TAPE7,TAPE8,OUTFIL.
SKIP,OVER.
EXIT.
ENDIF,OVER.
REPLACE,TAPE7,TAPE8,OUTFIL.
DAYFILE,TEST.
REPLACE,TEST.
```

The next step is to submit GO into the EXPRESS BATCH QUEUE. To do this enter

```
XSUBMIT,GO.
```

The computer will respond with a 7 character "HASH" code. The progress of the job is monitored by entering:

```
STATUS,JN.
```

Various "HASH" codes will be displayed on the screen. The status of the submitted file is printed out for each job in the machine under the user number that is being used. When the "HASH" code of the submitted job no longer appears, then the job is complete.

To check the output of the run get OUTFIL and XEDIT as follows:

```
XEDIT,OUTFIL,P,W=137.
```

Use XEDIT to locate errors or desired information. If a printout is desired, then enter the following after completing XEDIT procedures.

```
COPY,OUTFIL,PUTOUT.
DELIVER.Delivery Information
DAYFILE,PUTOUT.
ROUTE,PUTOUT,DC=LP.
```

If OUTFIL is not on file, then:

```
XEDIT,TEST,P
```

This will allow the user to look at the dayfile and determine which control card is in error, or which program "blew up".

If TEST is not on file, then check the submit file "GO". There is an error in the job, user, or charge card that must be corrected.

For more information refer to Sections VII, VIII and IX for Using the Terminal, NOS Commands, and XEDIT.

5.2 RUNNING NIPP FROM CARDS

Due to the high core storage used by NIPP, it may be beneficial to run NIPP from cards on the initial run. This will provide the user with permanent files and a listing of any errors that may be found in the input data. In order to run NIPP the data should be prepared as specified in the NECAP User's Manual and set up as a job card deck as shown below.

```
     Col. 1                                          Col. 41

     Jobname,T77.                                    Delivery Info.
(a)  USER(USERNUM)
     CHARGE(CHARNO,LRC)
     COPYBR,INPUT,NCFL.
(b)  SAVE(NCFL=NECAPFL)
     REWIND(NCFL)
     GET,NPC/UN=NECAPNM.
(c)  BEGIN,RUNNIP,NPC,NCFL.
     REWIND,TAPE7.
(d)  COPYSBF,TAPE7,OUTPUT.
     SAVE(TAPE7=LOADSFL)
     REWIND,TAPE8.
(e)  COPYSBF,TAPE8,OUTPUT.
     SAVE(TAPE8=SESPFL)
     7-8-9 multipunch (end-of-record)

(f)       NECAP DATA

     6-7-8-9 multipunch (end-of-information)
```

where:

(a) is the accounting information

(b) copies the input data to a local file and saves a copy as a permanent file

(c) executes NIPP

(d) copies the TLAP file to output and saves a copy as a permanent file

(e) copies the SESP file to output and saves a copy as a permanent file

(f) is the data to be run

5.3 RESPONSE FACTOR PROGRAM FROM THE TERMINAL

NECAP uses the response factor technique in determining loads, both in the thermal loads and variable temperature programs. The interactive version allows the user to design or redesign a surface at the terminal and determine that surface's "U" factor from a remote terminal. To exercise this option the user should have the working knowledge of LaRC's XEDIT.

5.3.1 INPUT File Setup

To setup a run to the interactive response factor program, the user should create a response factor INPUT file as described in the NECAP INPUT MANUAL.

The RESFAC Input data is put into NECAP on the L9 and L10 cards. RESFAC will read and process those cards as documented.

5.3.2 Accessing RESFAC

The following commands will get and run the RESPONSE FACTOR PROGRAM:

```
GET,INTRSFC/UN=NECAPNM.
BEGIN,INTRSFC,INTRSFC,DATA FILE.
```

The output is 136 characters per line. Therefore it is recommended that a TEKTRONIX 4014 or 4015 terminal be used to avoid "wrap around".

5.3.3 Hard Copy From Terminal

Should the user decide that a hard copy is desired, the following commands should be entered:

```
BEGIN,INTRSFC,DATA FILE,PUTOUT.
DELIVER.DELIVERY INFORMATION
DAYFILE,PUTOUT.
ROUTE(PUTOUT,DC=LP)
```

These commands will send a copy of the file to the printer which will be delivered as the user specified. The input data must be stored on a local file. If the user needs to rerun a case, then the file should be rewound prior to the next execution. A text editor may be used to make any needed input modifications.

5.4 SYSTEMS OUTPUT FILES

Card S3 in SESP contains an option which allows for an output file to be generated by NECAP. This file can be used as input for another program that the user may wish to run.

5.4.1 SESP Input Parameter

SESP uses card number S3 to set the output file flag. The fifth field on the card is the parameter for this option. To set the flag use the following:

```
IOTWF = 0.0   No file (default)
IOTWF = 1.0   Formatted (display code)
IOTWF = 2.0   Unformatted binary
```

IOTWF is the variable in the program that determines when to write to the output file.

The two types used are for two unique purposes. The first file, the unformatted binary, is used because it uses less CP seconds as it reads and writes the data bits directly to disk, thus leaving the real numbers in numeralized computer words. The second file is easier to read because the bits are masked into display coded characters.

IOTWF = 2 Unformatted Binary

To use unformatted reads the user must know FORTRAN and use the Langley CDC computers to process the data. The advantages, however, are speed and simpler programming.

The data is stored on the file as follows:

Variables on Record	No. of Words	Frequency
FAC	35	Once
CITY	35	Once
ENGR	35	Once
PROJ	35	Once
DATE	35	Once
ISYS,KMAX,IHSRT,ISTP	4	Once
IHAR,IMOY,IDOM,IHOD,ITOA,ITWB,WOA, PATM,DOA,TNFBP,TABCD,BGAS,OILH	12	Once for each hour
KFAN(K),QKC(K),QKH(K),QKKW(K), PWLK(K),H2O(K),ZERO,THKL(K), TKCD(K)	9	For each hour 1 to K times where K is the number of system spaces

A definition of variables may be found in the NECAP ENGINEERING MANUAL.

IOTWF = 1 Formatted Output

This file was generated primarily for input to other progams, for example, TRNSYS. The data will be written out in display code for 80 column card images. The formatted file contains only hourly data file if set up as follows:

FOR EACH HOUR:

```
VARIABLES   IHOUR,IMOY,IDOM,IHOD,ITOA,1TWB,RDN,BS,PATM,WOA,DOA
FORMAT:     I4,3(1X,I20),2(1X,I3),2(1X,F6.1),3(1X,F6.2)
```

FOR EACH SYSTEMS SPACE FOR EACH HOUR:

```
VARIABLES   KFAN(K),QKC(K),QKH(K),QKKW(K),PWLK(K),H2O(K),
            TKHL(K),TKCD(K)
FORMAT:     I2,7F10.1
```

Since there are 8760 hours in a non-leap year, then a building of five (5) zones would generate 52560 cards for a year's run. Should the user wish to store the data, magnetic tape would provide the most compact storage with the greatest portability. All of the variables are defined in the NECAP USER'S MANUAL and the NECAP ENGINEERING MANUAL.

5.4.2 Methods to Store Output Data

The data may be stored on a permanent file provided that there is ample storage available. These files are very large and may be stored by one of three methods.

Magnetic Tape

The SAVTAP procedure may be used to write the hourly data out to tape (Section 4.6). It is activated by inserting the following card after the BEGIN,NECAP; or BEGIN,RUNSP; card. For this process the card is formatted as follows:

```
BEGIN.....
BEGIN,SAVTAP,NPC,TAPENM,TAPE3,INIDT.
                 (a)   (b)   (c)
```

where:

(a) TAPENM is the VSN number of the tape to be written.

(b) TAPE3 is the output file to be saved.

(c) INIDT is the initialization data from loads (optional)

When using the data at a later time, procedure USETAP (Section 4.7) may be used to retrieve the information from the tape.

Direct Access Files

Since TAPE3 is a large file, it is recommended that if a permanent file is to be generated, it should be stored on a direct access file. In order to do this SESP should be run via the BEGIN RUNSP card in a special NECAP job stream. The NECAP run that generates the file should run on the R, T, Z or D machine.

The following commands should be inserted immediately before the BEGIN RUNSP card:

```
PURGE(TAPE3/ST=LPF,NA)
DEFINE(TAPE3/M=W)
BEGIN,RUNSP,NPC,TAPE8, INIDT,ANADT,TAPE3.
```

TAPE3 is the name of the file to receive the data. As SESP writes the data, it will be stored to the file. When the job has terminated, the file will be in permanent storage.

Indirect Access Files

The data can be stored as an indirect access file, but the user should be aware of the following:

- Should the file be too large, the job will terminate
- Only a few large files can be stored
- If the files are not accessed within 30 days, they will be archived (sooner if the system needs disc space).

The data can be saved by inserting the following card after the system's EXECUTION control card (RUNSP):

```
BEGIN,RUNSP,...
SAVE,TAPE3=pfn.
```

where

TAPE3 is the name of the local file and pfn is the name of the permanent file.

SECTION VI

NECAP WEATHER STATION CODES

The NECAP Weather Input originally came directly from a NOAA 1440 tape. NECAP would access the NOAA tape directly during execution, processing the data during each run. To eliminate this repetition, the weather processing was put into a pre-processor program called WEATHER.

WEATHER was expanded to read two 1440 formats and the TRY (Test Reference Year) formats. The weather year and station number were also put onto the NECAP weather input file.

The weather year record was expanded to include design day conditions and ground temperatures. This provided defaults for those input items which are necessary to run NECAP.

Currently, there are 63 weather stations (which have the hourly data, design day conditions, and ground temperatures data) converted to the NECAP Weather format. The data was compiled from NOAA weather tapes and the 1977 ASHRAE Fundamentals Handbook. The stations are accessed by the FAA's 3 letter code which is used world-wide to identify airports and their weather stations. The 63 stations are listed alphabeticaly by state then by city.

LIST OF NECAP WEATHER STATION CODES

CITY	STATE	INPUT CODE
BIRMINGHAM	AL	BHM
HUNTSVILLE	AL	HSV
PHOENIX	AR	PHX
FRESNO	CA	FAT
LOS ANGELES	CA	LAX
SACRAMENTO	CA	SAC
SAN DIEGO	CA	SAN
SAN FRANCISCO	CA	SFO
WASHINGTON	DC	DCA
JACKSONVILLE	FL	JAX
MIAMI	FL	MIA
TAMPA	FL	TPA
ATLANTA	GA	ATL
BOISE	ID	BOI
CHICAGO	IL	ORD
INDIANAPOLIS	IN	IND
DODGE CITY	KA	DDC
LOUISVILLE	KY	SDF
LAKE CHARLES	LA	LCH
NEW ORLEANS	LA	NEW
BOSTON	MA	BOS
PATUXENT RIVER	MD	PAX
PORTLAND	ME	PWM
DETROIT	MI	DET
MINNEAPOLIS	MN	MSP
COLUMBIA	MO	COU
KANSAS CITY	MO	MCI
ST. LOUIS	MO	STL
JACKSON	MS	JAN
GREAT FALLS	MT	GTF
RALEIGH	NC	RDU
BISMARCK	ND	BIS
OMAHA	NE	OMA
ALBUQUERQUE	NM	ABQ
ALBANY	NY	ALB
BUFFALO	NY	BUF
NEW YORK	NY	LGA
CINCINNATI	OH	CVG
CLEVELAND	OH	CLE
OKLAHOMA CITY	OK	OKC
TULSA	OK	TUL
MEDFORD	OR	MFR
PORTLAND	OR	PDX
PHILADELPHIA	PA	PHL
PITTSBURGH	PA	PIT
CHARLESTON	SC	CHS
MEMPHIS	TN	MEM
NASHVILLE	TN	BNA
AMARILLO	TX	AMA
BROWNSVILLE	TX	BRO
EL PASO	TX	ELP
FORT WORTH	TX	FWH
HOUSTON	TX	HOU
LUBBOCK	TX	LBB
SAN ANTONIO	TX	SAT
SALT LAKE CITY	UT	SLC

LIST OF NECAP WEATHER STATIONS (cont)

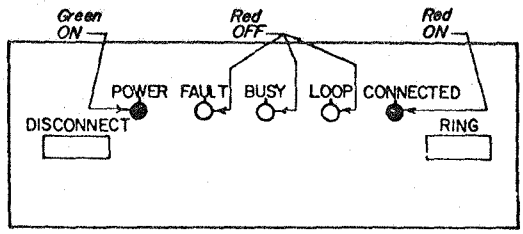
CITY	STATE	INPUT CODE	CITY	STATE	INPUT CODE
HAMPTON	VA	PHF	SEATTLE	WA	SEA
NORFOLK	VA	ORF			
RICHMOND	VA	RIC	MADISON	WI	MSN
BURLINGTON	VT	BTV	CHEYENNE	WY	CYS

SECTION VII

USING THE INTERACTIVE TERMINAL

7.1 LOG IN PROCEDURE

To gain access to NOS from a remote terminal, perform the following steps:



TRAN BOX PANEL (On Line)

1. Locate the TRAN box connected to your terminal and press the button labeled "RING." When NOS recognizes your request, the red light labeled "CONNECTED" will come on.

2. When the "CONNECTED" light comes on, press the terminal key "e."

3. NOS will then respond with:

   NASA LANGLEY CENTRAL COMPUTERS

   ENTER RESOURCE CODE:

   ADR
   /

   For most NECAP applications, the resource code should be 1A or 4Y.

4. NOS will then ask for a user number and a charge number which the user must supply and each response followed by a carriage return (cr) (see NOTE) The transmission listed below should appear on the terminal screen.

   80/02/04. 10:06:22 TERM4
   NASA/LRC CY173-Y NOS 1.4 (Y10) NOS 1.4 531/528Y
   FAMILY: (cr)
   USER NAME: XXXXXX
   PASSWORD: XXXXXX
   APPLICATION: IAF

   TERMINAL 40,NAMIAF
   RECOVER/CHARGE: CHARGE,XXXXXX,LRC.

   NOTE: (cr) designates carriage return, meaning that the user should press the carriage return at this point.

If user number and charge number are validated, the system will respond with:

```
$CHARGE,XXXXXX,LRC.
/
```

5. You are now in communication with NOS. Various "modes of operation" exist as subsystems of NOS. The user will go into the "batch mode."

7.2 TEKTRONIX TERMINALS

The TEKTRONIX terminals are storage screen terminals that have graphics capabilities. Listed below are several other capabilities:

1. Wider screens
2. More lines per screen
3. Change of character size

In order to clear the screen (page), the RESET PAGE key should be used. It is located in the upper left hand corner of the keyboard.

As mentioned, the TEKTRONIX will also vary the size of the characters shown on the screen. To change the size of the characters, the following procedure should be used:

1. Switch to local mode (upper left hand side of the console)
2. Depress "ESC" key (left hand side of keyboard, 2nd row of keys)
3. Depress an '8' for 72 cpl* and 30 lines per page
   '9' for 80 cpl* and 38 lines per page
   ':' for 120 cpl* and 53 lines per page
   ';' for 132 cpl* and 60 lines per page

   * characters per line
4. Switch back to line mode

7.3 CDC 713

There are also CONTROL DATA CORPORATION type 713 terminals that are smaller but easier to use. These terminals will print 80 characters per line and 15 lines per screen. The screen will roll up automatically but once a line leaves the screen, it cannot be seen again unless computer commands are used. The keyboard is somewhat easier to type on since it more closely resembles a typewriter keyboard.

The backspace key has an arrow pointing left (located on the right hand side of the keyboard). "Control/H" will also backspace.

7.4 USING THE TERMINAL FOR JOB CONTROL

The interactive terminal is the communication link between the user and the operating system. It allows the user to give the job control commands* such as:

GET,file (cr)
SAVE,file (cr)
SUBMIT,file b (cr)
etc

It also allows the user to create and/or change the contents of his/her files via the editor. XEDIT is a powerful and simple to use utility, any reference in this paper to an editor should be assumed to be speaking of XEDIT.

* Complete information or job control language can be found in the "NOS USER'S GUIDE." Complete information on XEDIT can be found in the "XEDIT USER'S GUIDE."

7.5 TERMINAL COMMANDS FOR PROCESS CONTROL

Commands may be entered from the terminal that affect program execution: for example, INTERRUPT. On NOS 1.4 these commands are as follows:

Cancel input line: (formerly ESCAPE)
**CTRL/X (cr)

Interrupt:
BREAK Key
**CTRL/P (cr)
then (cr) to resume

Terminate Program OUTPUT:
BREAK Key
**CTRL/T (cr)

Terminate Program EXECUTION:
**CTRL/T (cr)

** Depress Control Key and letter simultaneously, then press carriage return.

SECTION VIII

JOB CONTROL LANGUAGE AND NOS FILES

NETWORK OPERATING SYSTEM (NOS)

The LaRC computer complex utilizes the Network Operating System (NOS) developed by Control Data Corporation. On the LaRC system the current version of NOS is NOS 1.4, level 531. The user must understand three basic concepts:

- NOS is a file oriented system
- NOS commands are put into a separate job file
- NOS uses CYBER Control Language

The following information will be discussed and explained in this section.

- NOS Files
- Accessing Files
- Creating Files
- Removing Files
- Local File Usage
- Job Files
- CYBER Control Language

8.1 NOS FILES FILES

Files* can be classified in many ways and can exist in many different forms. In this manual only "access type" files will be discussed. Given this limitation, the two basic types of files available to NOS users are indirect access files and direct access files.

INDIRECT ACCESS FILES

When the user requests an indirect access file from NOS, the operating system does not get the actual file. A copy of the file is made and this copy is given to the user. This copy is called a "local file" and the actual file is referred to as a "permanent file." This feature is protective in nature. If the user improperly alters the local file, the permanent file still exists in unaltered form.

DIRECT ACCESS FILES

When the user requests a direct access file from NOS, the user alters the actual file and any changes made to the file are permanent.

NOS CONTROL STATEMENTS

The following sections will introduce the most frequently used NOS control statements for the system. The reader is cautioned that the description of the NOS control statements in this document is very brief and incomplete. For a detailed explanation of the NOS control options refer to the NOS USER'S MANUAL.

* For our purposes a "file" consists of any collection of information (such as data) saved on mass storage and accessed via the user created file name.

8.2 ACCESSING FILES

GET

The 'GET' control statement enables the user to request an indirect access file. Two common formats are:

GET,pfn.*

which instructs the operating system to get 'pfn,' make a copy of the file, and bring that copy to the user as a local file with the name 'pfn.'

or

GET,lfn=pfn.

which instructs the operating system to get 'pfn,' make a copy of the file, then bring that copy to the user as a local file with the name 'lfn.'

ATTACH

The 'ATTACH' command is used to obtain a direct access file. The format used is as follows:

ATTACH(pfn)

which brings the direct access file 'pfn' to the user.

or

ATTACH(lfn=pfn)

which brings the direct access file 'pfn' to the user under the name of 'lfn.'

* pfn designates a user supplied "permanent file name"
  lfn designates a user supplied "local file name"

NOTE: 'Lfn' is the actual file, not a copy. This option does not provide the "protective copy feature" of indirect access files. Instead, the user is permitted to refer to the direct access file by a name other than which it is saved. For example, the user has a direct access file saved under the name DAFILE and gives the command:

```
ATTACH(DATA=DAFILE)
```

If the user subsequently alters DATA, DAFILE is also altered.

8.3 CREATING FILES

Files are created in a number of ways, but the three most common methods are:

	Method	Discussed in
1.	Via the editor	Section IX
2.	Copying from card input	Section VIII
3.	Output from a program	Section VIII

CREATION FROM CARD INPUT

The two basic types of files are local and permanent. NOS provides a default file, INPUT, for the user. When in a local batch operation (via cards), this file uses the card reader. If the user is at the terminal, the file INPUT originates at the terminal keyboard. Thus, in NOS the file name INPUT is a reserved name. INPUT is always a local file. NOS automatically will put the data into a local disk file called INPUT. To do this, a job card deck should be punched and submitted to be run. This procedure is covered in Section E-6. The actual card deck is as follows:

```
    Col. 1                                          Col. 41

(a) JOBNAME,TIME,CORE MEMORY.                       Delivery Info.
    USER(USERNUM)
    CHARGE(CHARGE,LRC)
(b) COPYBR(INPUT,FILE1)
(c) SAVE(FILE1)
(d) 7-8-9 multipunch (end-of-record)
(e)          INSERT DATA
(f) 6-7-8-9 multipunch (end-of-information)
```

where:

- (a) is the accounting information. These three lines are required for all batch jobs.
- (b) copies the data from INPUT (the card reader) to a file designated by the user.
- (c) makes a permanent copy of the file.
- (d) is the end-of-record card (in this case denoting the end of the job card deck).
- (e) is the data to be copied in.
- (f) is the end-of-information which is the last card in the deck.

Parts b, c, d, and e are repeated if multiple files are to be input. For example, if three files are to be created, the steps would be bbb, ccc, d, e, d, e, d, e (i.e., the commands follow each other and the data is separated by an 7-8-9 multipunch card).

OUTPUT FROM A PROGRAM

To create a file from the output of another program, the user must know the name of the local file that the program is to produce. This file may be made permanent by using any of the following four commands.

SAVE

SAVE(lfn=pfn/FILE SECURITY) will save 'lfn' under the name 'pfn' as a permanent file with the user specified security. If a file is already saved under 'pfn,' the system will not save the file. The user must either 'REPLACE' the file or 'SAVE' it under another name. The file security is set by the parameters used in the 'SAVE' command. The user must access the file according to the security in force.

CT =	P	private file	*default
	S	semi-private file	
	PU	public file	
M =	W	read or write file	*default
	R	read only file	

PW = PASSWORD (default is no password)

REPLACE

REPLACE(lfn=pfn) replaces the old 'pfn' with 'lfn' and stores the local file under 'pfn.'

CHANGE

'CHANGE' changes any of the parameters originally set. The parameters are the same as the 'SAVE' command.

DEFINE

'DEFINE' creates a direct access file. The use of direct access files should be limited only to large files. This is because the system may purge direct access files on the CPFS system in order to save file space.

8.4 REMOVING OR DESTROYING FILES

This section will involve removing unwanted files. Unwanted files should be removed to keep file space availability at a maximum. Once the user is through with a file, it should be removed from the system. Any of the procedures discussed in this section may be done interactively or batch.

REMOVING LOCAL FILES

'RETURN' removes a file from the local working space. If the file was a permanent direct access file, that file will be returned to permanent storage. Otherwise, the file space is returned to the system. (Indirect access files locally, are copies of the permanent file.)

REMOVING PERMANENT FILES

'PURGE' removes a file from permanent file storage. If a copy is in the local file space, it remains intact until returned or the working session is terminated. However, the permanent copy is destroyed immediately. The 'PURGE' command will only purge a file that meets all of the requirements. If a file is semi-private and is a read only file, the purge command would be:

PURGE(pfn)

The 'PURGE' command will effect both indirect access and direct access files.

FILE ARCHIVAL

File archival will occur if files are not used within 30 days. The system will remove these files from the permanent file space and archive them on tape. If this occurs, the user need not worry about the file. An archived file may be brought back up by the following sequence of commands:

RELOAD,FILENAM/S,PO=S.

This will tell the system to reload the file and this process will usually be completed within the hour.

The file names are obtained by using the 'ARCLIST' command. To access the list the user should do the following:

```
ARCLIST.
```

This will list all of the files on archive and their corresponding tape number.

8.5 LOCAL FILE USAGE

On the NOS system, local files will perform a task or will be modified. It is in the local working space that files are created. Therefore, the user should be familiar with how the system handles local files. This section will discuss the following:

8.5.1 FILE POSITIONING

File positioning is important to most of the NOS commands. If the 'GET' or 'ATTACH' command is used, the pointer is positioned at the top of the file. If a program generates a file, the file position is dependent upon the program action. If a file is copied in from cards, the pointer is always at the end of the file. Unfortunately, the problem may arise that the pointer is not at the correct location. To make sure that the pointer is at the top of the file, the 'REWIND' statement is used:

```
REWIND,lfn.
```

The user should have the pointer at the top of all input files, program files, and executable files. The pointer should be at the bottom of all files being sent to the printer. To make sure that the pointer is at the bottom of a file, the copy commands may be used.

8.5.2 COPY COMMANDS

Copy commands move data from one file to another. There are several copy commands, each having its own specific purpose. Each command is covered in detail in the NOS User's Manual. The commands useful to NECAP are:

8.5.2.1 COPY
8.5.2.2 COPYBR
8.5.2.3 COPYBF
8.5.2.4 COPYEI
8.5.2.5 COPYSBF

For all copy commands, the first two parameters are (lfn1,lfn2), where 'lfn1' is the original file (defaults to INPUT) and 'lfn2' is the copied version (defaults to OUTPUT).

8.5.2.1 COPY

The copy command is used to copy a file from one file to another. This command has many parameter options, but the ones pertaining to NECAP are as follows:

COPY(lfn1,lfn2,V=X,M=C,L=OUTFL)
ⓐ ⓑ ⓒ

where:

ⓐ 'V' is the verify parameter which tells the system to rewind and verify the files. This defaults to no verify.

ⓑ 'M' specifies the coded mode. If 'M=C1,' input file is coded. If 'M=C2,' output file is coded (this should be used for stranger tapes). Both files are coded when 'M=CODED.' Defaults to binary files. The defaults are recommended for NOS to NOS copies.

ⓒ 'L' is the file on which the error messages are listed. This defaults to 'L=0,' no error listing.

8.5.2.2 COPYBR

'COPYBR' copies binary records. The format and parameters are:

COPYBR(lfn1,lfn2,NUM,C)
ⓐ ⓑ

where:

ⓐ is the number of records to be copied (defaults to 1).

ⓑ is copied in coded mode (defaults to binary).

The binary record is denoted by:

7-8-9	multipunch for cards
/EOR	for submit batch
--EOR--	for disk files

8.5.2.3 COPYBF

'COPYBF' copies binary files. The parameters are the same as 'COPYBR.' The system defines a binary file as a file separated by:

6-7-8-9	multipunch for cards
/EOF	for submit batch
--EOF--	for disk files

8.5.2.4 COPYEI

'COPYEI' copies the file until the end-of-information is encountered. The command has an optional third parameter, the verify/rewind parameter. End-of-information is denoted by:

6-7-8-9	multipunch for cards
/EOI	for submit batch
--EOI--	for disk files

Unless the verify/rewind option is invoked, neither of the files will be rewound.

8.5.2.5 COPYSBF

The 'COPYSBF' shifts the data from column 1 (the carriage control) to column 2. The 'COPYSBF' command should be used for all files that do not have carriage control. If this is not done, the printout may contain blank pages.

8.5.3 FILE OR SYSTEM STATUS

The 'ENQUIRE' command allows the user to determine various aspects of the computer actions. More detail and options are covered in the NOS USER'S REFERENCE MANUAL. Explanations of the three most useful commands follows:

8.5.3.1 SYSTEM STATUS

'ENQUIRE,A' outputs the status of the computer as it applies to the user. The local files, the limits, and the resources are reported. This is helpful for general information.

8.5.3.2 FILE STATUS

'ENQUIRE,F' outputs local file information. The following information is given: the file names, their PRU size, and whether a file is stored on tape, direct or indirect access. Also, this command indicates whether it is a read or write file.

8.5.3.3 JOB STATUS

'ENQUIRE,JN' outputs the progress of all jobs submitted under the logged user number. It reports the hash code and the queue the file is in.

8.5.4 DAYFILE

Dayfile is a file that the system writes, reporting the progress of the job as it executes. To look at this file, the 'DAYFILE' command is used. The dayfile will be written to the output file. If the user wants to write the dayfile to another file, the user should enter:

```
DAYFILE,lfn.
```

where 'lfn' is the file to receive the dayfile. Since the dayfile reports the actual work done by the computer, it is one of the most valuable tools the user has available.

8.5.5 FILE DISPOSITION

File disposition is especially useful when the user is working interactively. The user has the following options:

8.5.5.1 Dispose files to the printer
8.5.5.2 Submit a job from the terminal
8.5.5.3 Send a file to another computer

8.5.5.1 DISPOSING A FILE TO THE PRINTER

To send a file to the printer, the 'ROUTE' command is used. In addition, the 'DELIVER' and 'DAYFILE' commands must accompany the 'ROUTE' command. This information describes what was done and where to send the output. These three commands are entered upon completion of the file to be disposed. The sequence should be:

```
DELIVER.BIN OR BLDG. NUMBER   NAME
DAYFILE,lfn.
ROUTE(lfn,DC=LP)
```

8.5.5.2 JOB FILE SUBMIT

'SUBMIT' command submits a batch job from the interactive terminal onto the computer. The job file should begin with a '/JOB' card and the records are separated by '/EOR.' This allows the user to create several jobs and submit them from the terminal, i.e., without using cards. The command is as follows:

SUBMIT,lfn,B.

'Lfn' must be a job file and have the '/' cards in addition to the job cards. The 'B' must follow the file name. Failure to do so will result with job execution, but with no output.

8.5.5.3 JOB FILE SEND

The 'SEND' command will send the file to another machine. This is done by the following command:

SEND,lfn,M=machine,DC=DISPOSITION CODE.
ⓐ ⓑ ⓒ

where:

ⓐ 'Lfn' is the file that is sent. It may be a job file or an output file to the printer. In either case, the rules are the same for all machines. For job files make sure the file is in the proper format, and for output files make sure the dayfile and delivery information is present.

ⓑ This parameter indicates which machine the file is sent to. The user should specify the machine according to these guidelines:

Machine	Type of Job to be run
A	Overnight jobs
C	Jobs using magnetic tapes
D	Jobs using computer graphics
R	Large batch jobs
T	Large batch jobs
Y	Overnight jobs
Z	Jobs using the CYBER 203 computer

Any of the machines will do NECAP jobs, except CYBER 203.

ⓒ 'DC' is the disposition code. If the file is to be sent to the machine's printer, the parameter is 'DC=LP.'

8.6 NOS JOB FILES

NOS job files are files that are used to control a computer run. They are formatted as follows:

Job card
User number
Charge number
NOS commands
Data (if needed)

This is required for all batch jobs either from cards or the terminal. The only difference between terminal submit and card batch is that a '/JOB' card appears before the job card in the terminal submit file. The commands are the same, but the record terminations differ. The terminal submit file uses '/EOR,' where a 7-8-9 multipunch card goes into a job card deck. These are the only differences. In both cases, the commands and the data are submitted in the same format.

THE JOB CARD

The job card tells the system the following:

1. Job name (1 to 7 character alphanumeric name)
2. Time needed
3. Core memory needed
4. Delivery information

The format is shown on the following page.

```
Col.#1      TIME,  CORE MEMORY.                    Col.#41 Col.#51

                                        BIN or
JOBNAME,    T177,  CM300000.            BLDG.  NO.  NAME
   (a)       (b)       (c)               (d)        (e)
```

(a) is the Job Name, any 1-7 alphanumeric name may be given, but the first character must be a letter. This must be specified.

(b) is the time needed in cp seconds octal. This defaults to 100 which is not enough time for NECAP jobs. NECAP requires between 2000 and 5000 cp seconds.

(c) CM is the Core Memory required for the job. This defaults to the entire machine. If the default is used, the priority is the lowest. The lower the CM needed the higher the priority. NECAP usually uses approximatly 270000 octal words of memory.

(d) is the Bin or Building Number where the output listing to be delivered.

(e) is the name of the person to receive the output.

THE USER CARD

The user number is a 7-character code assigned to the user by ACD at LaRC. The card format is as follows:

```
USER(USERNUM)
```

THE CHARGE CARD

The charge number is the account number that the job costs are to be charged against. This number is assigned to the user. If the user is not authorized for a charge number, the job will be kicked off the system. The card format is as follows:

```
CHARGE,NUMBER,LRC.
```

THE NOS COMMANDS

The NOS commands are the NOS utilities used for the job. These immediately follow the charge card and are terminated by an end-of-record mark. The end-of-record mark is a /EOR for a job submit file and a 7-8-9 multipunch for card decks.

DATA SETUP

Each section of data should be separated by an end-of-record mark. The end-of-record mark is a /EOR for a job submit file and a 7-8-9 multipunch in column one for card decks. The entire job is terminated by an end-of-information mark. The end-of-information mark is a /EOF for submit files and a 6-7-8-9 mulitpunch card for card decks.

8.7 CYBER CONTROL LANGUAGE

CYBER Control Language (CCL) provides a method for executing a given string of NOS commands by issuing a single command. CCL allows the user to specify parameters and will test for system attributes. As with any language there is a structure which must be used.

THE CCL PROCEDURE

The CCL procedure is the set of NOS cards arranged to perform a specific task. The Proc is stored as a record of the file. To access the procedure both the name of the proc and the file on which it is stored must be specified. A procedure is started by a ".PROC" card which is formatted as follows:

.PROC,Procname,keyl,...,keyn.

Where:

Procnam is a 1 to 7 alphanumeric NOS compatible name.

Keyl,...,Keyn is a key word which may be changed upon execution to name a file or set a register. These are very powerful and add to the flexibility of the procedures.

The Revert Card informs the system to revert back to the calling procedure. The Revert Card allows a proc to call another proc and return to the calling procedure. The Revert Card is as follows:

REVERT. COMMENT

ACTIVATING A CCL PROCEDURE

A CCL Procedure is activated by the BEGIN command. The BEGIN command is as follows:

BEGIN,Procnam,Procfil,Keyl,...,Keyn.

Where:

Procnam is the name of the procedure; this defaults to the name of the file.

Procfil is the file containing the procedure; this defaults to a file called "PROCFIL."

Keyl,...,Keyn are the keywords used for this specific run. These may be order dependent; if they are not order dependent, then an equal sign (=) must be used specifying which parameter is being changed.

SECTION IX

XEDIT

This section describes simple useage of the editor XEDIT. It is not to be taken as a complete guide to XEDIT, but it will be sufficient information to use with NECAP. The guide is in three sections:

9.1 ACCESSING XEDIT

XEDIT is a Direct Access file residing on the NOS library. To use XEDIT the Direct Access file must be local. This is done as follows:

```
ATTACH,XEDIT/UN=LIBRARY
```

Once XEDIT is a local file, it can be initiated with the "XEDIT" command.

9.1.1 THE XEDIT COMMAND

The XEDIT command is very powerful and is capable of many uses. The following format will explain the uses.

```
              c
XEDIT,lfn,  ,w=n.
              p
```

XEDIT	tells NOS to begin EXECUTION
lfn	tells XEDIT the name of the file to be edited
c	tells XEDIT that the file is to be created
(blank)	tells XEDIT that the file is local
p	tells XEDIT that the file is on permanent storage
w="n"	tells XEDIT to set the width to "n". If omitted, XEDIT sets the width to 80 characters per line. TRUNCATION will occur beyond this width. W=137 is recommended for output formatted files.

Once the XEDIT command is entered, the user is in XEDIT and XEDIT responds with:

```
XEDIT 2.1.7
??
```

9.1.2 XEDIT MODES

XEDIT has two modes of operation: "EDIT" and "INPUT."

EDIT uses ?? as the prompt character
INPUT uses ? as the prompt character

Whenever the ?? prompt character appears, it tells the user to enter an XEDIT command. The ? prompt character tells the user to enter input.

9.1.3 LEAVING XEDIT

Once changes are made to a file, it then becomes necessary to return the file to the file space for proper use. There are three possible ways of leaving XEDIT associated with three different reasons.

1. Editing is complete and the file is ready; no more copies are needed.

2. Editing is not complete but the file needs to be stored as it is.

3. Incorrect changes were made during the editing session and the file needs to be restored to its original state.

9.1.3.1 Editing Completed

Editing has been completed, the file is to be stored as specified and control is to be returned to the user to perform another task. The END command performs this function. The format is:

END,filenam,storage

where:

filenam is the file that was created and is to be saved (defaults to original file name)

storage designates where the file is to be stored

l means local
r means replace with the new version
s means save it as a new permanent file (defaults to local only)
rs cannot be used together but rl and sl are permitted.

9.1.3.2 Editing is not Completed

Editing is not complete but the file is in an acceptable form. This is advantageous when a long editing session is necessary and there is a possibility the system may fail, or several copies are generated from a file at one time. The command that performs this task is the FILE command. It is formatted as:

```
FILE,filenam,storage
```

(see above for explanation of filenam and storage)

After the FILE command is executed, a message of the storage appears on the terminal screen. The ?? prompt character then tells the user he may enter his next XEDIT command. NOTE: The pointer is now at the top of the file.

9.1.3.3 Incorrect Changes Made

If incorrect changes were made during the editing session, XEDIT provides the user with an escape method that terminates all changes made in that editing session. The STOP command will return the file to its original condition prior to editing and terminate the XEDIT session.

9.2. XEDIT COMMANDS

XEDIT commands do the actual file manipulation. These commands move throughout the file sequentially using line pointers, character strings, and sometimes the cursor. Simple XEDIT commands can be categorized by their movement.

9.2.1 POINTER MOVEMENT

Pointer Movement commands move the editor up and down the file, acting as an entire line of code as they execute. They are used to reposition the pointer, display lines, delete lines, and insert lines.

9.2.1.1 PRINT (P)

Prints n lines. If n is not specified, only the current pointer location is printed. If n is specified, then n lines are printed and the pointer is repositioned at the nth line.

9.2.1.2 NEXT (N)

n moves the pointer down x lines. If x is not specified, the pointer goes to the next line. If x has a negative value, the pointer is positioned before the current line. When the NEXT command has repositioned the pointer, it prints the line out.

9.2.1.3 TOP (T)

Moves the pointer to the top of the file but the line is not printed out.

9.2.1.4 ^ (prefix)

When the is used as a prefix to a command, it signals XEDIT to go to the top of the file and execute the command. For example: ^p25 tells XEDIT to go to the top of the file and print 25 lines.

9.2.1.5 SCAN (SC)

Prints out n lines but the line pointer is not repositioned. The SC command cannot pass through the buffer limits, else the pointer position would be lost. When this occurs, XEDIT does scan to the end of the buffer then informs the user with an END OF BUFFER message. (Remember the pointer has not moved on the SC command.) To get past the end of buffer, the PRINT command should be used.

9.2.1.6 DELETE (D)

Deletes n lines from the file. As XEDIT deletes the lines, they are echoed out to the screen. If n is not specified, only the line of the current location is deleted. After the delete command has completed execution, the pointer is positioned on the line after the last deleted lines.

9.2.1.7 INSERT (I)

Inserts n lines into the file immediately following the line at which the pointer is positioned. INSERT will insert exactly n lines: no more and no less. If n is not specified, only one line is inserted. If no text is entered, a blank line will be inserted. Upon completion of the INSERT command, the pointer is positioned on the last line inserted.

9.2.1.8 INSERTB (IB)

Inserts a line above the current pointer location. The pointer is not repositioned to the new line.

9.2.1.9 * (suffix)

If * is used as n or x, it means execute the command preceding it throughout the entire file, from the current pointer location.

9.2.2 CHARACTER STRINGS

These commands move the pointer or change characters as the defined string is located in a file. This allows the user to "home" in on a specific unique part of a file.

To define a string the user must use a delimiter. A delimiter is simply a pair of like symbols used to denote the beginning and end of a string. In XEDIT any non-alphanumeric symbol may be used. (The exception to this is the : or ;.)

XEDIT can locate a string and change a string to a newly defined string.

9.2.2.1 LOCATE /str/ (L/str/)

Locates a string and then prints out the line containing that string. The pointer is repositioned at that line. If the string cannot be located, XEDIT will hit the END OF FILE and position the pointer to the top of the file. If "n" is used, it will print at n lines on which the string is found and the pointer is at the nth line printed. If the * suffix is used, the pointer is positioned at the top of the file.

9.2.2.2 CHANGE /st1/st2/ (C/st1/st2/)

Changes string one to string two. If string one is not on the line at which the pointer is positioned, XEDIT issues a message STRING NOT FOUND. String one must be on the line at which the pointer is located. There are other features to the CHANGE command which are explained in the XEDIT User's Manual.

9.2.3 CURSOR DEPENDENT COMMANDS

Combining the line pointer and the cursor enables XEDIT to locate a given character within a file. There are two primary commands: MODIFY and QMOD. These commands are similar but QMOD has more features.

9.2.3.1 MODIFY (M)

Modifies the line at which the pointer is positioned. MODIFY prints out the line as it is and provides the ? prompt character signifying that it is ready for input. The cursor is positioned under the character to be modified and the replacement character is typed at that location. After all of the characters have been keyed in, hit Carriage Return and XEDIT echoes out the corrected/modified line.

There are three characters that cannot be entered as text as they serve a specific function for the MODIFY command. These are the #, &, and ^.

- # Tells MODIFY to remove the character from the line (blanks are characters). All characters to the right shift to the left.

- & Tells MODIFY to put a blank at that location. A blank will appear and characters to the right do not shift.

- ^ Tells MODIFY to insert the characters following the before the character to which it is pointed. The string of characters following the must be terminated by a #. (Blanks may be inserted as blanks.)

9.2.3.2 QMOD (QM)

Works the same way as the MODIFY command but with two extra features. One is that it displays the column numbers beneath the characters and two, it can execute on "n" lines. These features are nice for rigidly formatted or repetitive files (NECAP uses both).

9.2.4 OTHER COMMANDS AVAILABLE

There are many other commands available to the user and they are explained in detail in the XEDIT USER'S GUIDE. The ones explained here are basic, and as stated previously, will be sufficient to handle NECAP editing.

9.3. NECAP APPLICATIONS

NECAP often requires modifying the input data, but usually only a few lines in an entire file. XEDIT is the fastest way to accomplish this. The most commonly used commands are the MODIFY and QMOD commands. They enable the user to change only one field on a given card. The user may also use the LOCATE and CHANGE command but often there are many duplicated numbers. EACH CARD SHOULD HAVE A SEPARATE IDENTIFIER. This enables the user to locate a specific card. These tips should increase the efficiency of editing a file. As the user becomes accustomed to XEDIT editing, the editing session itself will become faster.

Many times separate models must be generated for comparison studies. Here again XEDIT proves to be a valuable tool. This is done by three steps.

1. XEDIT,oldfile,p
   (oldfile is the base run)

2. Make changes using XEDIT

3. Then end session with:
   END,newfile,sl

This provides two files: the old file is on the permanent file storage under the original name and a modified copy is created and stored as a permanent file. This eliminates the repetitive card punching that would otherwise be required.

APPENDIX A

NECAP FILES

ECONSC	FORTRAN Source for ECON Program
NIPBIN	NECAP INPUT PROCESSOR PROGRAM Relocatable Binaries
NPC	NECAP PROCEDURE FILE
RESBIN	RESPONSE FACTOR PROGRAM Relocatable Binaries
SESP	SYSTEMS ENERGY SIMULATION PROGRAM Relocatable Binaries
TLBN41	THERMAL LOADS ANALYSIS PROGRAM Relocatable Binaires
WEATHER	FORTRAN Source to Process NOAA Weather
WPC	PROCEDURE FILE TO ACCESS NECAP WEATHER

APPENDIX B

NECAP MAGNETIC TAPES

NM0853 NECAP FORMATTED TRY TAPE WEATHER - TRY01

NM0854 NECAP FORMATTED TRY TAPE WEATHER - TRY02

NM0855 NECAP FORMATTED TRY TAPE WEATHER - TRY03

NH0850 NECAP FORMATTED 1440 for LANGLEY, HUNTSVILLE, and PAX RIVER

NH0860 NECAP Programs

NJ1274 - Scratch Tapes (there are six available)

NJ1279

APPENDIX C

NECAP PROCEDURE FILE (NPC)

APPENDIX C

Procedure to run NIPP

```
.PROC,RUNNIP,DATA,TAPE7,TAPE8,OUTPUT,ECHO.
.*
.*   PROCEDURE TO RUN NIPP
.*
.*   DATA = NIPP INPUT FILE
.*   TAPE7 = TLAP INPUT FILE
.*   TAPE8 = SESP INPUT FILE
.*   OUTPUT = MESSAGE FILE
.*
REWIND,NPC.
.*
.*   IF ECHO IS A NUMBER
.*   THEN TURN ON DATA ECHO.
.*   IN NIPP VIA SSW
.*
IFE,NUM(ECHO).EQ.0,NOECHO.
ONSW(1)
ELSE,NOECHO.
OFFSW(1)
ENDIF,NOECHO.
BEGIN,GETIT,NPC,DATA.
BEGIN,GETFL,NPC,NIPBIN.
LDSET(MAP=SBEX/PUTOUT,PRESET=ZERO)
NIPBIN(DATA,OUTPUT,,,TAPE7,TAPE8,PL=15000)
RETURN,NIPBIN,DATA.
REWIND,TAPE7,TAPE8.
OFFSW(1)
REVERT.  NIPP EXECUTION COMPLETE
```

APPENDIX C

Procedure to run TLAP

```
.PROC,RNTLAP,DATA,PHF,TAPENUM.
.*
.*    GET WEATHER
.*
GET,WPC/UN=NECAPNM.
IFE,NUM(PHF)=0,WETIT.
BEGIN,PHF,WPC.
ELSE,WETIT.
BEGIN,TAPE,NPC,TAPENUM.
ENDIF,WETIT.
.*
.*    CHECK TO SEE IF DATA IS LOCAL
.*
BEGIN,GETFL,NPC,TLBN41.
REWIND(WEATHER,DATA)
LDSET(MAP=SBEX/PUTOUT,PRESET=ZERO)
TLBN41(DATA,,,,WEATHER,INIDT,ANADT,PL=50000)
SKIP,THERE.
EXIT.
DMD.
DMD(300000)
BEGIN,D3.
ENDIF,THERE.
RETURN(WEATHER,TLAPBIN)
REVERT.  TLAP EXECUTION END
```

Procedure to run NECAP

```
.PROC,NECAP,DATA,PHF,TAPE7,TAPE8,TEST,ECHO,TAPENUM.
*
*   THIS PROCEDURE EXECUTES NECAP 4.1
*
*   DATA = NIPP INPUT FILE
*   TAPE7 = TLAP INPUT FILE
*   TAPE8 = SESP INPUT FILE
*
*
*   EXECUTE NIPP
*
.*   IF ECHO IS A NUMBER
.*   RUN NECAP WITH DATAV ECHO
.*
IFE,NUM(ECHO).EQ.1,DAECHO.
BEGIN,RUNNIP,NPC,DATA,TAPE7,TAPE8.
ELSE,DAECHO.
.*
.*   RUN WITH NO DATAV ECHO UNLESS
.*   INPUT ERRORS ARE DETECTED
.*
BEGIN,RUNNIP,NPC,DATA,TAPE7,TAPE8,,1.
ENDIF,DAECHO.
REWIND,TAPE7,TAPE8.
REWIND,NPC.
BEGIN,DAY,NPC,TEST.
IFE,EFG=0,ERROUT.
*
*   EXECUTE TLAP
*
BEGIN,RNTLAP,NPC,TAPE7,PHF.
REWIND,NPC.
BEGIN,DAY,NPC,TEST.
IFE,EFG=0,ERROUT.
*
*   EXECUTE SESP
*
BEGIN,RUNSP,NPC,TAPE8,INIDT,ANADT.
REWIND,NPC.
BEGIN,DAY,NPC,TEST.
ELSE,ERROUT.
*
*   AN ERROR HAS OCCURED
*   PUTOUT WILL BE DUMPED
*
DISPLAY,EFG.
ROUTE,PUTOUT,DC=LP)
ENDIF,ERROUT.
BEGIN,DAY,NPC,TEST.
REVERT.  NECAP EXECUTION COMPLETE
```

APPENDIX C

Procedure to SAVE file on Magnetic Tape

```
.PROC,SAVTAP,TAPNUM,INIDT,ANADT.
.*
.*    SAVE NECAP FILES ON TAPE
.*
REWIND,INIDT,ANADT.
VSN(TAPE=TAPNUM)
LABEL(TAPE,NT,D=PE,FI=NECAP FILES,PO=W,W)
COPYBF(INIDT,TAPE,1)
COPYBF(ANADT,TAPE,1)
REWIND,TAPE.
RETURN,TAPE.
REWIND,INIDT,ANADT.
REVERT.  END PROC FILE TO SAVE NECAP DATA
```

Procedre to RETRIEVE file from Magnetic Tape

```
.PROC,USETAP,TAPNUM,INIDT,ANADT.
.*
.*    READ NACAP FILES FROM TAPE
.*
VSN(TAPE=TAPNUM)
LABEL(TAPE,NT,D=PE,FI=NECAP FILES,PO=R)
COPYBF(TAPE,INIDT,1)
COPYBF(TAPE,ANADT,1)
REWIND,TAPE.
RETURN,TAPE.
REWIND,INIDT,ANADT.
REVERT.  END PROC FILE TO READ NECAP DATA
```

APPENDIX C

Procedure to run SESP

```
.PROC,RUNSP,INP,TAPE1,TAPE2,TAPE3.
.*
.*   THIS PROCEDURE RUNS SESP
.*
BEGIN,GETFL,NPC,SESP.
RENAME,LGO=SESP.
.*
.*   IF INP IS NOT LOCAL GETIT
.*
BEGIN,GETIT,NPC,INP.
REWIND(TAPE1,TAPE2)
LDSET(MAP=SBEX/PUTOUT,PRESET=ZERO)
LOAD(LGO)
NOGO(SESP)
SESP(INP,,,,TAPE1,TAPE2,TAPE3,PL=50000)
RETURN(SESP,LGO)
SKIP,THERE.
EXIT.
DMD.
DMD(270000)
ENDIF,THERE.
REVERT.  SESP EXECUTION END
```

APPENDIX C

Procedure to GET files for use in job stream

```
.PROC,GETIT,FN.
NOEXIT.
.*
.*    THIS PROC CHECKS TO SEE IF FN
.*    IS LOCAL: IF NOT, THEN GETS IT
.*
IFE,FILE(FN,LO)=0,GETIT.
GET,FN/NA.
IFE,FILE(FN,LO)=0,ONEMTM.
.*
.*    CHECK IF FILE IS ARCHIVED
.*
ARCLIST,FN.
DISPLAY,EF.
IFE,EF=0,LODIT.
.*
.*    FILE IS ON ARCHIVE RELOAD IT
.*
RELOAD,FN/NA.
REWIND,FN.
ELSE,LODIT.
.*
.*    FILE IS NOT ON ARCHIVE
.*    COPY FROM INPUT
.*
COPYBR,INPUT,FN.
REWIND,FN.
.*
.*    RESET EF TO ZERO
.*
SET,EF=0.
ENDIF,LODIT.
ENDIF,ONEMTM.
ENDIF,GETIT.
ONEXIT.
REVERT.  FN IS NOW LOCAL
```

APPENDIX C

Procedure to GET NECAP files for job stream

```
.PROC,GETFL,FN.
.*
.*    THIS PROC CHECKS TO SEE IF FN
.*    IS LOCAL: IF NOT, THEN GETS IT
.*
IFE,FILE(FN,LO)=0,GETIT.
GET,FN/UN=NECAPNM,NA.
IFE,FILE(FN,LO)=0,GOTAP.
BEGIN,FN,NPC.
ENDIF,GOTAP.
ENDIF,GETIT.
REVERT.  FN IS NOW LOCAL
```

Procedure to MONITOR JOB PROGRESS

```
.PROC,DAY,TEST.
SRUCOMP.
DAYFILE,TEST.
REPLACE,TEST.
REVERT.
```

BIBLIOGRAPHY

Jensen, R. N., NECAP 4.1, NASA's Energy Cost Analysis Program Engineering Manual. TM 83239 National Aeronautics and Space Administration, Hampton, VA, 1982.

Jensen, R. N., NECAP 4.1, NASA's Energy Cost Analysis Program Input Manual. TM 83240 National Aeronautics and Space Administration, Hampton, VA, 1982.

Jensen, R. N., NECAP 4.1, NASA's Energy Cost Analysis Program User's Manual. TM 83238 National Aeronautics and Space Administration, Hampton, VA 1982. (In preparation)

NOS User's Guide. Langley Research Center, Hampton, VA, 1981.

NOS Reference Manual, Vol. 1 pub. no. 60435400, Control Data Corporation, St. Paul, MN, 1981.

NOS Reference Manual, Vol. 2 pub. no. 60445300, Control Data Corporation, St. Paul, MN, 1981.

Reference Manual WBAN Hourly Surface Ovservations 1440. National Climatic Center, Asheville, NC, April, 1966.

Tape Reference Manual. Airways Surface Observations, TDF 14, Asheville, NC

XEDIT Version 3 Reference Manual, pub. no. 60455730, Control Data Corporation, St. Paul, MN, 1979.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-165802		

4. Title and Subtitle	5. Report Date
NECAP - NASA's Energy Cost Analysis Program - Operations Manual	June 1982
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
David L. Miner	
	10. Work Unit No.
9. Performing Organization Name and Address	11. Contract or Grant No.
Computer Sciences Corporation 3217 N. Armistead Avenue Hampton, VA 23666	NAS1-16078
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Contractor Report
National Aeronautics and Space Administration Washington, DC 20546	14. Sponsoring Agency Code

15. Supplementary Notes

Langley technical monitors: John E. Hogge and Ronald N. Jensen
Final Report

16. Abstract

This reference manual is written for the engineer/user of NASA'S ENERGY COST ANALYSIS PROGRAM (NECAP) at the Langley Research Center (LaRC) and describes the use of the NECAP program on the LaRC computer complex. It also provides supplementary information on new capabilities and program options that can be used.

Sections VII, VIII, and IX pertain to the Control Data Corporation (CDC) NETWORK OPERATING SYSTEM (NOS) which is in use at the NASA Langley Research Center. These sections provide the basic CDC NOS instructions which are required to successfully operate NECAP at LaRC.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Energy Analysis Air Conditioning and Heating	Unclassified - Unlimited Subject Category 44

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	84	A05

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

End of Document