NASA-CR-172,414 #

NASA Contractor Report 172414

ACCOUNTING UTILITY FOR DETERMINING INDIVIDUAL USAGE OF PRODUCTION LEVEL SOFTWARE SYSTEMS

NASA-CR-172414
19840023889

Stacey C. Garber

Kentron International, Inc.
Hampton, Virginia 23666

LIBRARY COPY

SEP 7 1984

LANGLEY RESEARCH CENTER
LIBRARY, NASA
HAMPTON, VIRGINIA

NASA Contract NAS1-16000
June 1984

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

Summary

An accounting package has been developed which determines the computer resources utilized by a user during the execution of a particular program and updates a file containing accumulated resource totals. The accounting package is divided into two separate programs. The first program determines the total amount of computer resources utilized by a user during the execution of a particular program. The second program uses these totals to update a file containing accumulated totals of computer resources utilized by a user for a particular program. This package is useful to those persons who have several other users continually accessing and running programs from their accounts. The package provides the ability to determine which users are accessing and running specified programs along with their total level of usage.

N84-31959#

Introduction

Much consideration has been given by software developers to the problem of obtaining accumulated totals for the computer resources utilized by different users during the execution of particular programs. A set of accounting programs has been developed to aid in solving this problem. The first program in the accounting package determines the computer resources utilized during the execution of a program. The second program in the accounting package updates a file containing the user table of accumulated usage. So, by the use of this package, a software manager can determine which of his programs are most frequently accessed by other users and the level of usage of the programs for each user.

The accounting package of the present study is set up to enable a software manager to determine the total amount of system resource units (SRU) and central processing unit (CPU) time that other users utilize while running a program that is maintained on the software manager's account. A system resource unit is a unit of accounting used at the NASA Langley Research Center's computer center, which is calculated by a charging algorithm (ref. 1) and is a composite value of Input/Output activity, memory usage, and central processing time. Central processing unit time, in seconds, is concerned only with the actual amount of CPU time required to execute the program. Input/Output activity is not a factor in determining CPU time (ref. 2). A few minor modifications would be needed to use the accounting package for other types of computer resources such as mass storage and magnetic tapes.

The logic of the two accounting programs and their use are described in detail in this report. The examples presented in this report demonstrate the use of the accounting package on a particular applications program system containing four independent modules (i.e., pre- and post-processors, analysis modules). The second program in the accounting package has been written specifically for one application software system, but with a few modifications, the accounting programs can be adapted to another software manager's needs. Appendix A contains the flowcharts for both programs in the accounting package, appendix B contains the source listings of these programs, and appendix C contains some examples.

PROGRAM ACCTPRG

The first program in the accounting package is a program called ACCTPRG. This program determines the total amount of SRU and CPU time used during the execution of a particular module of the application software system. Program ACCTPRG is called twice during a job stream. The first call occurs before the execution of the module for which the accounting is being performed, and once again after the execution of the module has been completed. The first call of ACCTPRG retrieves the starting values of the SRU and CPU time. The second call of ACCTPRG retrieves the ending SRU and CPU time, and determines the total amount of SRU and CPU time used during the execution of the module by taking the difference of these values.

Input/Output

Program ACCTPRG needs only one variable from the user as input. The variable, called TITLE, is an identification title that designates the module for which the accounting information is being accumulated. The variable TITLE must be right-justified in the first ten columns of the input deck. This input for ACCTPRG is set up through the use of the CYBER control language command statement .DATA (ref. 1). All other data used by ACCTPRG is gathered from the operating system.

The output for program ACCTPRG is written on a local file, TAPE99 and includes the following information:

Title User number
SRU CPU

This sequence may be repeated as many times as needed during a particular job stream, depending on how many modules are executed. Example 1 of appendix C is an example of the output from the program ACCTPRG.

General Procedure for Program ACCTPRG

Program ACCTPRG performs two tasks: initialization and termination. The initialization task retrieves the SRU and CPU time from the system before any program execution takes place. The termination task retrieves the SRU and CPU time from the system after the program execution is completed. The use of the local input file, TAPE3, helps the program determine if the call is the first or second call for the program. IF ACCTPRG is able to read a title from TAPE3, then the initialization is performed. If ACCTPRG is not able to read a title from TAPE3, then the termination is performed.

During the first access of ACCTPRG, several initialization steps are followed. First, a call to the system utility JPARAMS (ref. 3) is made. This utility retrieves the accounting information from the system, and returns it to the main program in an array. The needed initial information is then written to the local file TAPE90. The user number under which the job is running is retrieved from the system using the system utility USERNUM, and stored in the variable USER1. The title, read from TAPE3, and the user number, retrieved from the system, are then written to the local file TAPE99.

During the second call to ACCTPRG, the termination steps are followed. First, a call to the system routine JPARAMS is made. The ending accounting information is retrieved from the system, used as the ending information, and returned to the main program in another array. The initial information retrieved in the first call to ACCTPRG is read from TAPE90. Next, the total number of SRUs and the total amount of CPU time used during execution of the module for which the accounting information is being accumulated is determined. This is done by subtracting the initial SRU and CPU time from the ending SRU and CPU time. These results are then written to TAPE99.

A flowchart of the program ACCTPRG is given in appendix A.

Program ACCTUPD

The second program in the accounting package is a program called ACCTUPD. This program updates the accounting file that contains the accumulated accounting information and can be set up to handle the accumulated SRU and CPU time totals for many unique applications programs. To add a new program, modifications must be made to this program and to the accounting file.

Input/Output

Input for program ACCTUPD comes from two sources. One source is the local file TAPE99 which was created by program ACCTPRG. This tape contains the SRU and CPU times for the execution of a particular module. Below is a table containing the input format for TAPE99, and an example can be found in appendix C, example 1. The format statements used for the input to program ACCTUPD from TAPE99 are:

FORMAT (1X,A10,5X,A7)
and
FORMAT (1X,F10.3,F10.3).

Record No.	Variable	Format	Description
1	TITLE	A10	Identification title
1	USER	A7	User number under which job was run
2	SR	F10.3	SRU units used during execution
2	CP	F10.3	CPU time used during execution

These cards may be repeated, depending on how many modules were run during one job stream.

The other source of input for ACCTUPD comes from the accounting file TAPE98 which already exists as a permanent file in the software manager's directory. Below is a table containing the input format for TAPE98, and an example can be found in appendix C, example 2. The format statements used for the input to program ACCTUPD from TAPE98 are:

FORMAT (132A1)

and

FORMAT (1X,A7,2X,F10.2,2X,F10.2)

Record No.	Variable	Format	Description
1	HEAD1	132A1	first set of headings
2	HEAD2	132A1	second set of headings
3	USERNUM	A7	user number
3	CPUNIT	F10.2	accumulated CPU time
3	SRUNIT	F10.2	accumulated SRU units

Records 1 and 2 contain any headings that are used in the accounting file. The variable USERNUM found on record 3, contains the user numbers that already exist in the accounting file. The variables CPUNIT and SRUNIT are used to store the accumulated totals representing the level of usage by the given user. The CPUNIT and SRUNIT variables may be repeated several times on one record, depending on how many module totals are being accumulated. Also, record 3 may be repeated depending on how many users have accessed the modules.

The output of program ACCTUPD is written on the local file TAPE97, which is used to replace the old accounting file, TAPE98. TAPE97 contains all the latest updates made on the accounting file. Example 2 in appendix C is an example of a generated accounting file.

General Procedure for Program ACCTUPD

Upon entering program ACCTUPD, the program first prepares the new accounting file (TAPE97) by writing the headings. The program is then ready to begin the search to locate the user number for which the accumulated SRU and CPU totals will be updated. Three situations have been considered for this study and each situation will be studied individually.

If the accounting file has never been accessed, program ACCTUPD enters this first entry and updates the appropriate columns, depending on which column title found in STITLE compares with the TITLE read from TAPE99. The new entry is written to the accounting file and then the program terminates. Control is then returned to the job stream.

If the accounting file exists and contains one or more entries, but the user for which the updates should be made is a new user, the program first searches through all entries in the accounting file for a user number corresponding to the one obtained by program ACCTPRG. Once the end of the accounting file has been reached, without finding the corresponding user number, the program assumes it is a new entry. The procedure for adding a new entry is used.

If the accounting file exists and the user for whom the updates should be made is an existing entry on the accounting file, ACCTUPD searches the accounting file until the corresponding user number is found, and the appropriate SRU and CPU updates are made.

A flowchart of program ACCTUPD is given in appendix A.

Execution of the Accounting Package

To build and execute a procedure file which utilizes the accounting package, ACCTPKG, the software manager must perform the following steps:

1. Create absolutes (or executables) for ACCTPRG and ACCTUPD
2. Form an initial accounting file
3. Build a procedure file to drive the accounting programs
4. Create the job stream to execute the procedure file

Each step is described in detail below and an example is presented in appendix C.

Step 1: Create Absolutes

To execute the accounting package, it is necessary to form the absolutes of the program ACCTPRG and ACCTUPD and to use these absolutes in the procedure file. This prevents the compilation and loading of each program during each separate run. When retrieving a copy of the accounting package, three separate records are found on this file. The first record contains some brief documentation, the second record contains program ACCTPRG, and the third record contains program ACCTUPD. Example 3 in appendix C is an example job stream which creates the absolutes of program ACCTPRG and ACCTUPD.

Step 2: Form Initial Accounting File

To run the accounting package, the permanent accounting file, maintained by the software manager, must already exist. On the initialization run of program ACCTUPD, the accounting file need only contain two lines of headings. These headings may be set up by another user-written program, or the user may manually define the file using the system's editor. It is important that this permanent accounting file be set up to allow other users to replace it on the account from which it was retrieved so that any updates that are made will be saved. The software manager needs to specify that the accounting file be given write permission. Example 4 in appendix C is an example of an initial accounting file.

Step 3: Procedure File

A procedure file should be written to insure that all appropriate calls have been made for the execution of the accounting package. See example 5 in appendix C for a procedure file used in the execution of a STAGSC-1 module that incorporates the accounting programs.

There are several things to note when building the procedure file. First, the absolutes for program ACCTPRG need to be called both before and after the module being executed (in the example case the module is STAGS1). This insures that all the accounting information has been retrieved from the system, and the SRU and CPU totals determined. Secondly, TAPE90 must be rewound before the second call to ACCTPRG to ensure that the initial accounting information can be read by the program. Third, TAPE99 must be rewound after the second call of the absolutes for ACCTPRG so that the data may be read by the absolutes for program ACCTUPD. Finally, TAPE3 must not be rewound before the second call to the absolutes of ACCTPRG. If this happens, the ending information is never retrieved because ACCTPRG sees the second call as the first, since a title can be read.

Step 4: Job Stream

Once the procedure file has been completed, the actual use of it is very easy. Other users are only responsible for retrieving the written procedure file, and executing the appropriate procedure thereby insuring that all steps are being followed and completed in the appropriate order by all users. Example 6 in appendix C is an example of a job stream used to execute the procedure in example 5.

Concluding Remarks

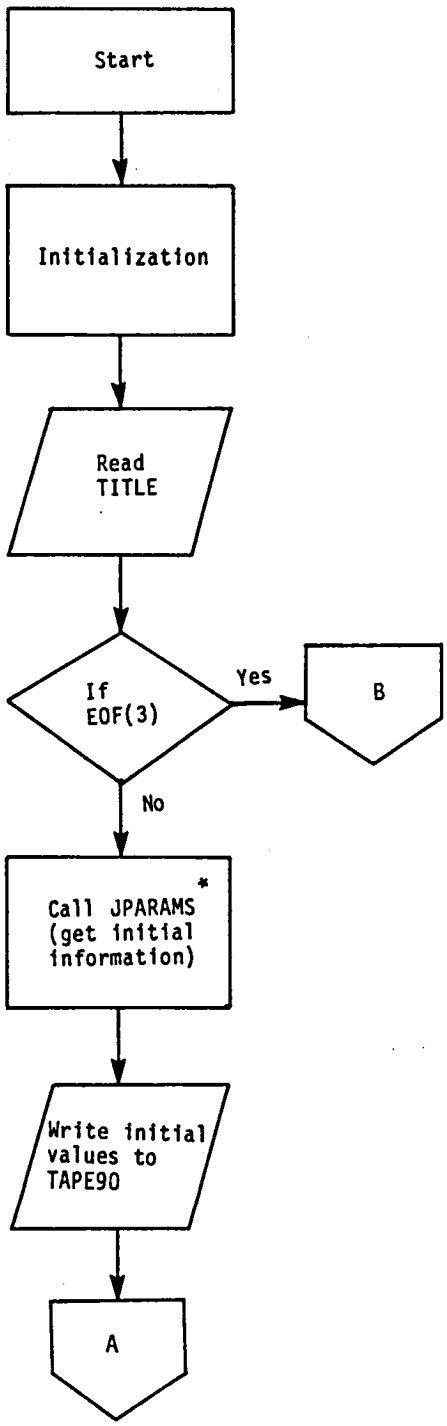
An accounting package has been written to determine and accumulate the computer system resources used during the execution of particular applications programs. This package consists of two programs. The first program determines the amount of SRU and CPU time that has been used during the execution of a program, and the second program uses these values and updates the accumulated totals. This accounting package allows a software manager to determine what users are accessing various programs from his account, and the level of usage of each user.

The accounting package was developed and tested on the NASA Langley Research Center computer system using the available system software utilities. Usage of this package at another computer installation may require significant modifications depending on the computer system (i.e., different system software routines or different version of FORTRAN).

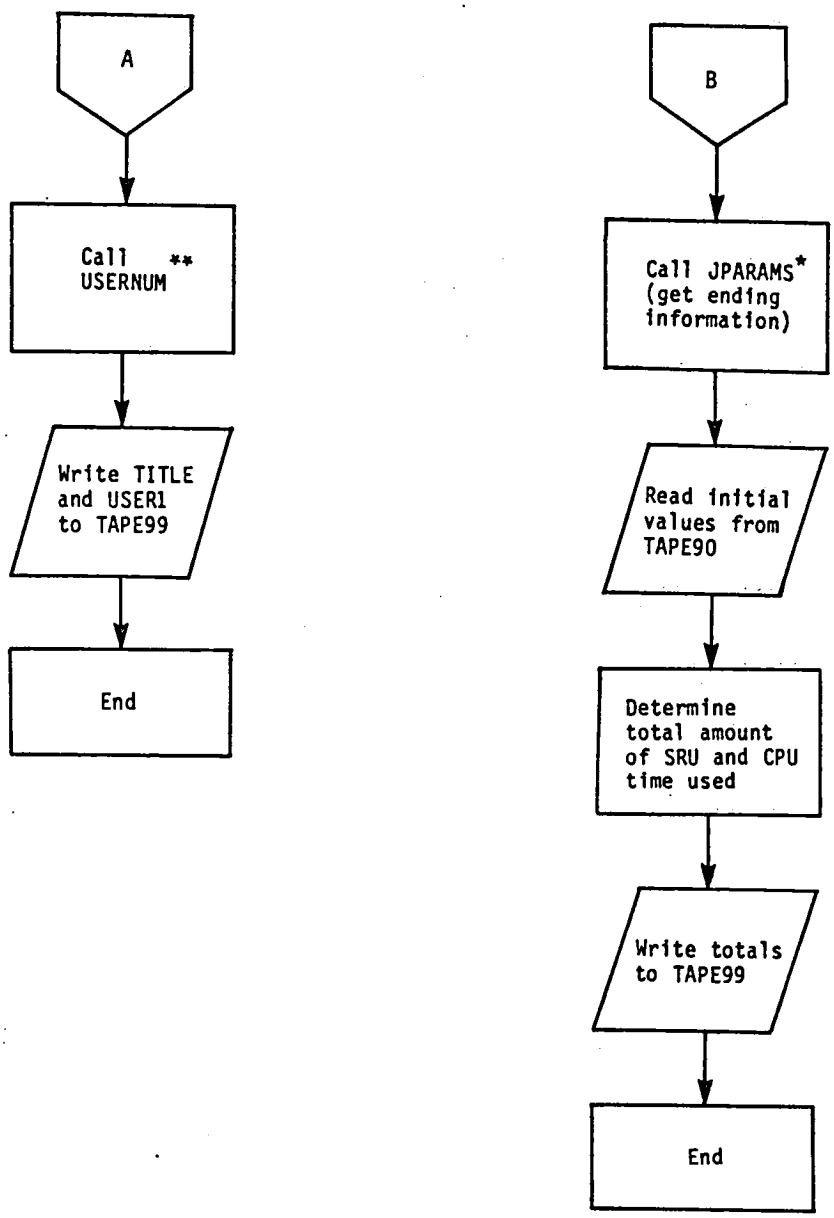
Appendix A: Flowcharts

This appendix contains the flowcharts for the two programs in the accounting package.

Flowchart for Program ACCTPRG



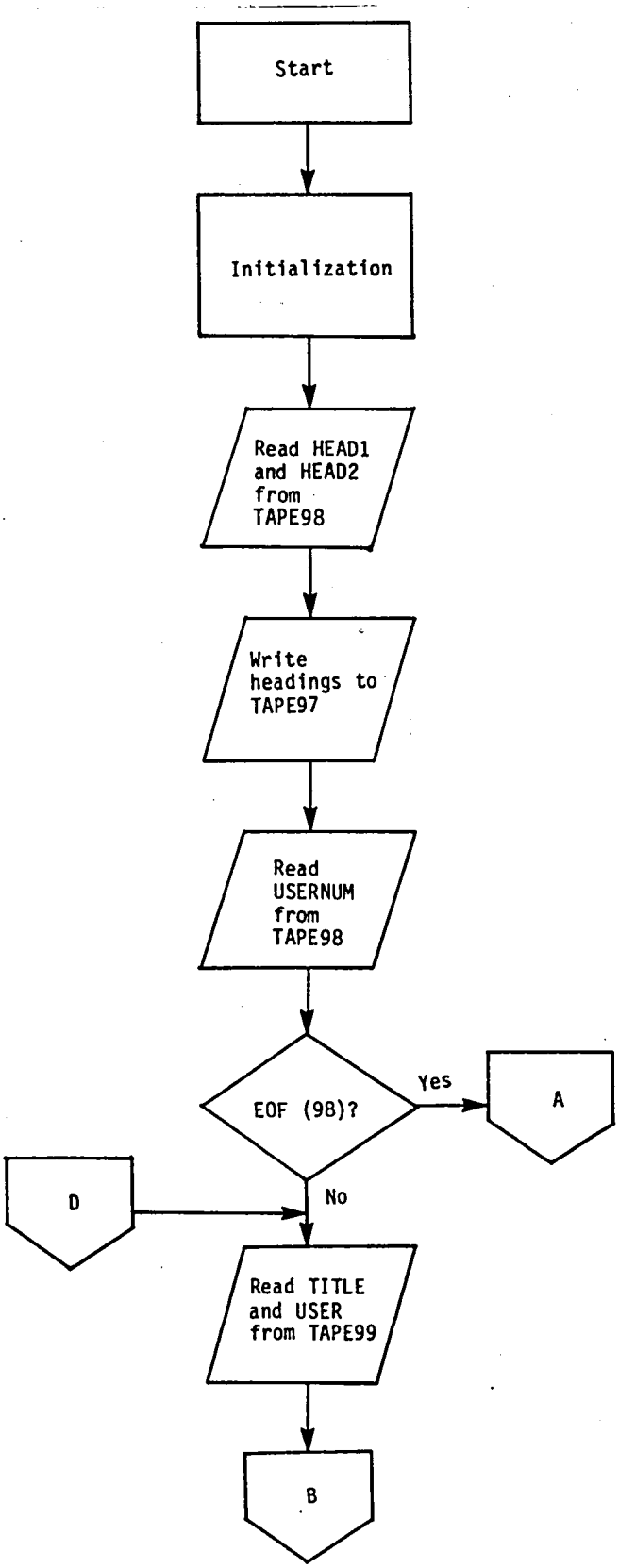
Flowchart for Program ACCTPRG (concluded)



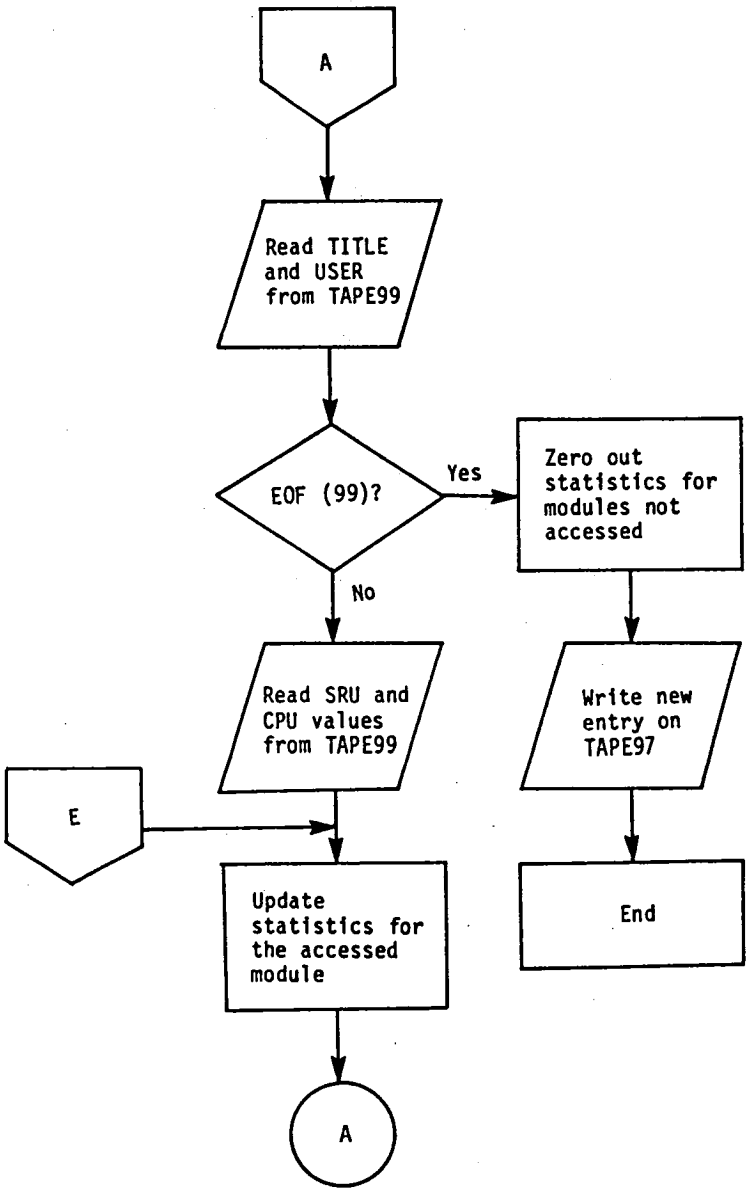
*No flowchart provided for this routine. The routine already exists and can be found on NMACFTN/UN=LIBRARY.

**No flowchart provided for this routine. The routine already exists and can be found on UTLIB/UN=UTIL.

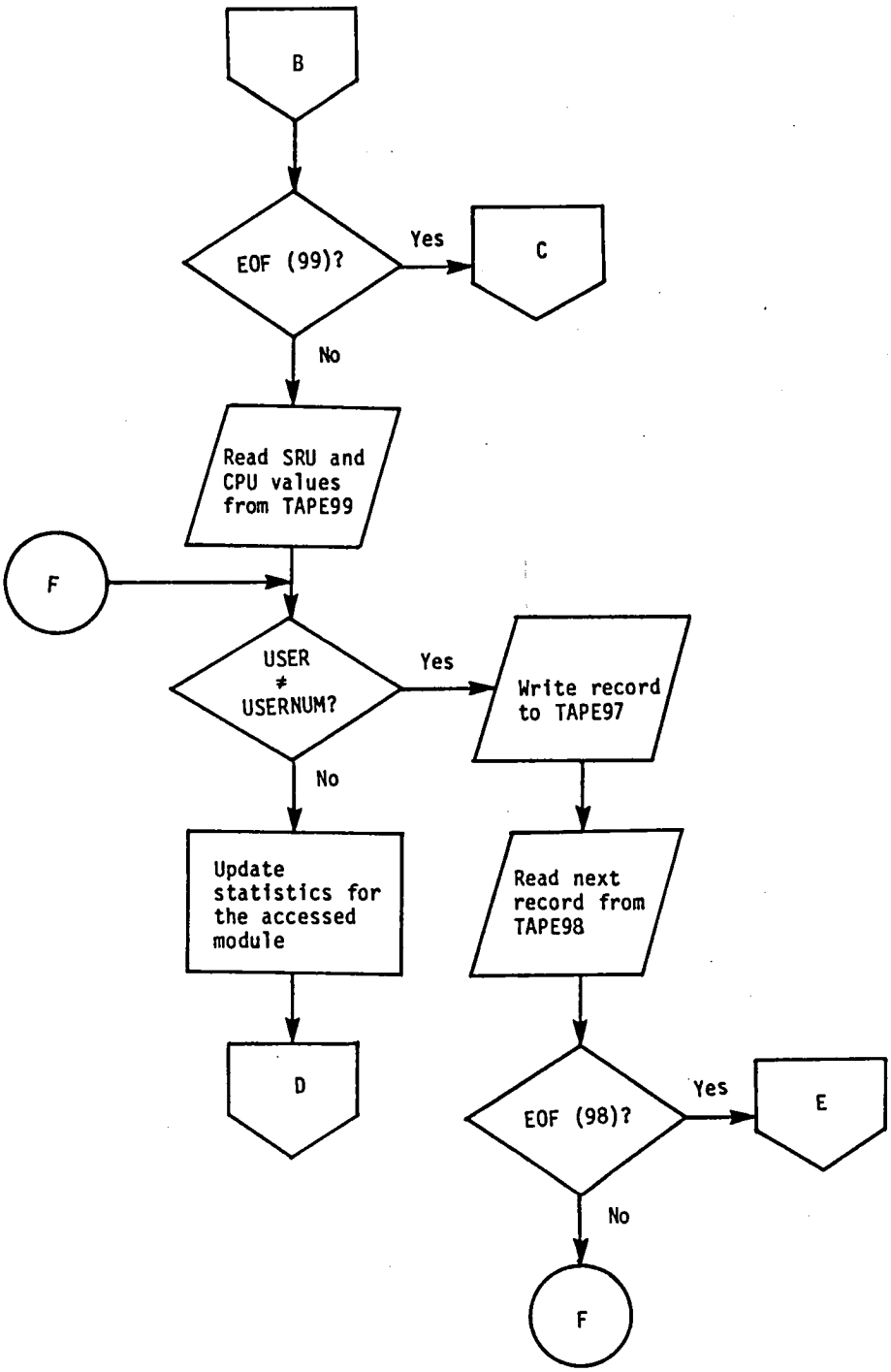
Flowchart for Program ACCTUPD



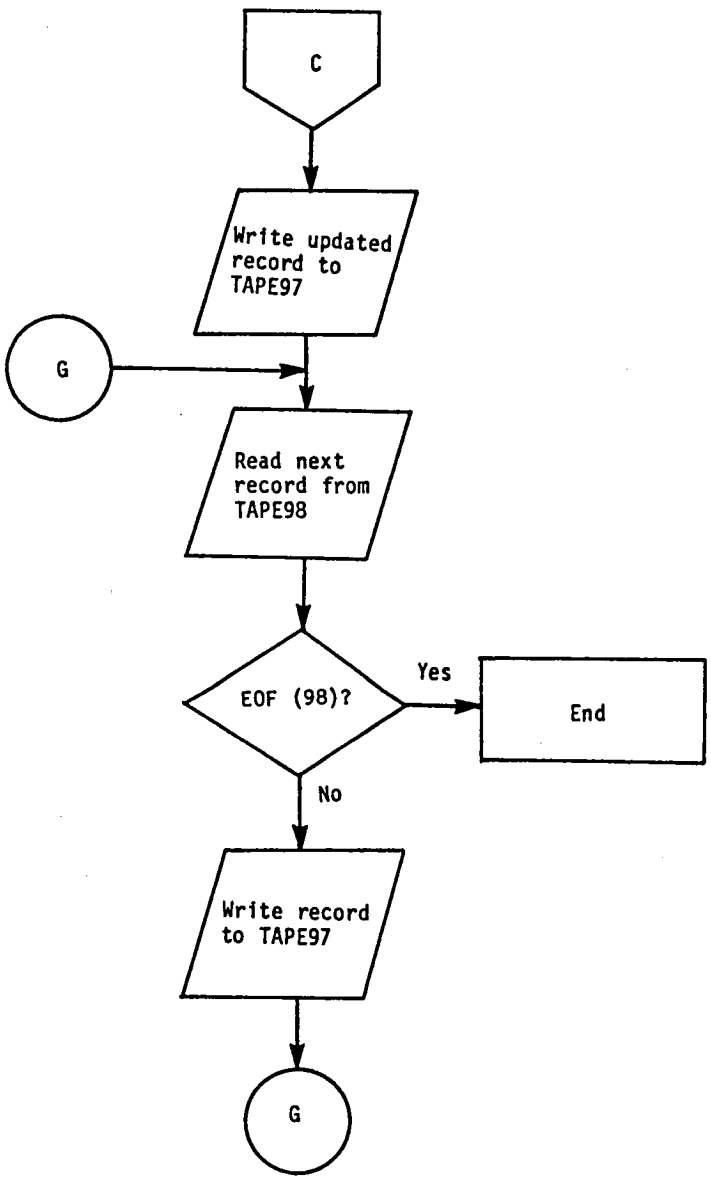
Flowchart for Program ACCTUPD (cont.)



Flowchart for Program ACCTUPD (cont.)



Flowchart for Program ACCTUPD (concluded)



Appendix B: Source Listing of the Accounting Package

This appendix contains the source listings of both programs in the accounting package. These programs were written in FORTRAN IV for a CYBER 170 series computer with NOS 1.4. Loading the accounting package requires 19,049 words of memory.

Source Code for Program ACCTPRG

```
      PROGRAM ACCTPRG(INPUT,OUTPUT,TAPE3,TAPE90,TAPE99)
C
C
C  THIS PROGRAM USES THE SYSTEM ROUTINE JPARAMS TO RETRIEVE
C  THE SRU AND CPU TIME FROM THE SYSTEM. THE VARIABLES
C  USED IN THIS PROGRAM ARE:
C       P = ARRAY PASSED TO JPARAMS
C       RVALUE1 = CPU VALUE RETRIEVED FROM JPARAMS
C       RVALUE2 = SRU VALUE RETRIEVED FROM JPARAMS
C       TITLE = IDENTIFICATION TITLE
C       USER1 = USER NUMBER RETRIEVED FROM THE SYSTEM
C       BVALUE1 = INITIAL CPU VALUE
C       BVALUE2 = INITIAL SRU VALUE
C       EVALUE1 = TOTAL CPU TIME
C       EVALUE2 = TOTAL SRU TIME
C
C
      INTEGER P(30)
      EQUIVALENCE (RVALUE1,IVALUE1)
      EQUIVALENCE(RVALUE2,IVALUE2)
C
C  TRY TO READ FROM TAPE3
C
      READ(3,100) TITLE
      IF (EOF(3)) 12,3
C
C  TITLE READ, SO RETRIEVE BEGINNING INFORMATION
C
    3 CALL JPARAMS(P)
      IVALUE1=P(12)
      IVALUE2=P(13)
C
C  WRITE BEGINNING VALUES TO TAPE90
C
      WRITE(90,300) RVALUE2,RVALUE1
C
C  RETRIEVE USER NUMBER FROM SYSTEM
C
      CALL USERNUM(USER1)
      WRITE(99,200) TITLE,USER1
      GO TO 50
C
```

Source Code for Program ACCTPRG (concluded)

```
C  TITLE NOT READ, SO RETRIEVE ENDING INFORMATION
C
   12 CALL JPARAMS(P)
      IVALUE1=P(12)
      IVALUE2=P(13)
C
C  READ BEGINNING VALUES FROM TAPE90
C
      READ(90,300) BVALUE2,BVALUE1
C
C  DETERMINE TOTAL AMOUNT OF SRU AND CPU TIME USED
C
      EVALUE2=RVALUE2-BVALUE2
      EVALUE1=RVALUE1-BVALUE1
      WRITE(99,210) EVALUE2,EVALUE1
   50 CONTINUE
C
C  FORMATS
C
  100 FORMAT(A10)
  200 FORMAT(1X,A10,5X,A7)
  210 FORMAT(1X,2F10.3)
  300 FORMAT(1X,2F10.3)
C
C  EXIT
C
      STOP
      END
```

Source Code for Program ACCTUPD

```
      PROGRAM ACCTUPD(INPUT,OUTPUT,TAPE6=OUTPUT,TAPE97,TAPE98,TAPE99)
C
C
C  THIS PROGRAM TAKES THE ACCOUNTING INFORMATION GENERATED FROM
C  THE ACCTPRG RUN AND UPDATES THE PERMANENT ACCOUNTING FILE,
C  STAGACT.  THE TAPES ARE DEFINED AS FOLLOWS:
C      TAPE97 = NEW ACCOUNTING FILE CREATED
C      TAPE98 = OLD ACCOUNTING FILE
C      TAPE99 = ACCOUNTING INFORMATION GENERATED BY ACCTPRG
C  THE VARIABLES USED ARE AS FOLLOWS:
C      HEAD1 = FIRST LINE OF HEADINGS
C      HEAD2 = SECOND LINE OF HEADINGS
C      USERNUM = USER NUMBER FROM ACCOUNTING FILE
C      CPUNIT = ACCUMULATED CPU TIME FROM ACCOUNTING FILE
C      SRUNIT = ACCUMULATED SRU TIME FROM ACCOUNTING FILE
C      TITLE = TITLE OF THE MODULE RUN
C      USER = USER NUMBER FROM THE TAPE99 RUN
C      SP = SRUS USED BY THE USER ON THE TAPE99 RUN
C      CP = CPUS USED BY THE USER ON THE TAPE99 RUN
C
C
      DIMENSION HEAD1(132),HEAD2(132),CPUNIT(4),SRUNIT(4),IDUM(4),
     .          STITLE(4)
      DATA STITLE/10H    STAGS1,10H    STAGS2,10H      STAPL,
     .           10H     POSTP/
C
C  ZERO OUT DUMMY ARRAY AND SET COUNTER TO ZERO
C
      DO 10 I=1,4
   10   IDUM(I)=0
      J=0
C
C  READ THE HEADINGS FROM THE OLD ACCOUNTING FILE AND WRITE
C  THEM TO THE NEW ACCOUNTING FILE
C
      READ (98,200) (HEAD1(I),I=1,132)
      READ (98,200) (HEAD2(I),I=1,132)
      WRITE(97,300) (HEAD1(I),I=1,132)
      WRITE(97,300) (HEAD2(I),I=1,132)
C
C  ATTEMPT TO READ FIRST ENTRY ON OLD ACCOUNTING FILE
C
```

```
      READ(98,210) USERNUM,(CPUNIT(I),SRUNIT(I),I=1,4)
C
C  CHECK FOR EOF ON OLD ACCOUNTING FILE
C      T - FIRST ACCESS EVER OF THIS CODE
C      F - REGULAR UPDATE
C
      IF (EOF(98)) 130,20
C
C  CODE FOR REGULAR UPDATE
C
C
C  READ THE INFORMATION OFF OF THE GENERATED ACCOUNTING
C  INFORMATION FROM THE STAGSC1 RUN
C  (IF EOF TRUE, THEN THE UPDATE IS COMPLETE -
C   IF EOF FALSE, MORE UPDATING IS NECESSARY)
C
   20 READ(99,220) TITLE,USER
      IF(EOF(99)) 80,25
   25 READ(99,230) SR,CP
C
C  TEST TO SEE IF THE USER NUMBER FROM THE ACCOUNTING FILE
C  IS THE ONE NEEDED
C
   30 IF(USER.NE.USERNUM) GO TO 110
C
C  CORRECT USER INFORMATION FOUND, NOW UPDATE THE RECORD
C  (LOCATE THE APPROPRIATE MODULE COLUMN AND UPDATE THE
C   MODULE STATISTICS)
C
   35 J=J+1
      IF(TITLE.NE.STITLE(J)) GO TO 35
      CPUNIT(J)=CPUNIT(J)+CP
      SPUNIT(J)=SRUNIT(J)+SR
      J=0
      GO TO 20
   80 CONTINUE
C
C  WRITE UPDATED RECORD, READ AND WRITE REST OF RECORDS FROM
C  THE OLD ACCOUNTING FILE
C
      WRITE(97,310) USERNUM,(CPUNIT(I),SRUNIT(I),I=1,4)
   90 READ(98,210) USERNUM,(CPUNIT(I),SRUNIT(I),I=1,4)
```

```
      IF(EOF(98)) 500,100
  100 WRITE(97,310) USERNUM,(CPUNIT(I),SRUNIT(I),I=1,4)
      GO TO 90
C
C  CORRECT USER NUMBER NOT FOUND - WRITE CURRENT RECORD, READ
C  NEW RECORD AND REPEAT THE ABOVE PROCEDURE - IF EOF,
C  THEN NEW ENTRY
C
  110 WRITE(97,310) USERNUM,(CPUNIT(I),SRUNIT(I),I=1,4)
      READ(98,210) USERNUM,(CPUNIT(I),SRUNIT(I),I=1,4)
      IF(EOF(98)) 140,30
C
C  CODE FOR FIRST ACCESS EVER OF THE CODE OR NEW ENTRY FOR
C  ACCOUNTING FILE
C
  130 READ(99,220) TITLE,USER
      IF(EOF(99)) 190,135
  135 READ(99,230) SR,CP
C
C  FORM THE NEW RECORD
C  (LOCATE THE APPROPRIATE MODULE AND ASSIGN STATISTICS -
C   IDUM=2 IF A MODULE IS ACCESSED AND UPDATED)
C
  140 J=J+1
      IF(TITLE.NE.STITLE(J)) GO TO 140
      IDUM(J)=2
      CPUNIT(J)=CP
      SRUNIT(J)=SR
      J=0
      GO TO 130
C
C  ZERO THE VARIABLES THAT WERE NOT UPDATED WITH THIS NEW ACCESS
C
  190 DO 195 I=1,4
        IF(IDUM(I).EQ.2) GO TO 195
        CPUNIT(I)=0.0
        SRUNIT(I)=0.0
  195 CONTINUE
C
C  WRITE THE NEW ENTRY ON THE ENDING OF THE ACCOUNTING FILE
C
      WRITE(97,310) USER,(CPUNIT(I),SRUNIT(I),I=1,4)
```

Source Code for Program ACCTUPD (concluded)

```
      500 CONTINUE
C
C  FORMAT STATEMENTS
C
      200 FORMAT(132A1)
      210 FORMAT(1X,A7,2X,F10.2,2X,F10.2,3X,F10.2,2X,F10.2,3X,
     *       F10.2,2X,F10.2,3X,F10.2,2X,F10.2)
      220 FORMAT(1X,A10,5X,A7)
      230 FORMAT(1X,F10.3,F10.3)
      300 FORMAT(132A1)
      310 FORMAT(1X,A7,2X,F10.2,2X,F10.2,3X,F10.2,2X,F10.2,3X,
     *       F10.2,2X,F10.2,3X,F10.2,2X,F10.2)
C
C  EXIT
C
      STOP
      END
```

Appendix C: Examples

This appendix contains all the output examples for both programs ACCTPRG and ACCTUPD. Also, any examples that are needed for the proper execution of the accounting package are also found in this appendix.

```
   STAGS1      163709C
205.038       24.735
    STAPL      163709C
 58.479        5.850
```

Example 1 - Output of Program ACCTPRG

USERNUM	STAGS1		STAGS2		STAPL		POSTP	
	CPU	SRU	CPU	SRU	CPU	SRU	CPU	SRU
163709C	24.73	204.02	0.00	0.00	5.77	57.51	0.00	0.00
489200N	15.58	185.76	0.00	0.00	0.00	0.00	5.88	25.33

Example 2 - Output of Program ACCTUPD

```
STREM.
USER,XXXXXXX,XXXXXX.
CHARGE,XXXXXX,LRC.
DELIVER.  XXXXXXX
FETCH,ACCTPKG/UN=163709C.
COPYBR,ACCTPKG,DOC.
COPYBR,ACCTPKG,ACCTPRG.
COPYBR,ACCTPKG,ACCTUPD.
REWIND,ACCTUPD,ACCTPRG.
GET,LIB1=NMACFTN/UN=LIBRARY.
GET,LIB2=UTLIB/UN=UTIL.
FTN,I=ACCTPRG,R=3,L=OUTPUT,B=ACCT.
FTN,I=ACCTUPD,R=3,L=OUTPUT,B=UPACCT.
REWIND,ACCT,UPACCT.
LDSET(PRESET=ZERO)
LOAD,UPACCT.
NOGO,STACT.
REPLACE,STACT.
LDSET(PRESET=ZERO,LIB=LIB1/LIB2)
LOAD,ACCT.
NOGO,RACCT.
REPLACE,RACCT.
```

Example 3 - Job Stream for Generating the Absolutes for the Accounting Package

USERNUM	STAGS1		STAGS2		STAPL		POSTP	
	CPU	SRU	CPU	SRU	CPU	SRU	CPU	SRU

Example 4 - Initial Accounting File

```
.PROC,LSTAGS1,FL.
*
*  PROCEDURE TO RUN STAGSC1 AND GENERATE ACCOUNTING INFO
*
FETCH,STAGS1/UN=XXXXXXX.
*
*  GET ABSOLUTES FOR PROGRAM ACCTPRG
*
FETCH,RACCT/UN=XXXXXXX.
REWIND,TAPE3.
*
*  FIRST CALL OF ACCTPRG
*
RACCT,TAPE3.
RFL(FL)
REDUCE(-)
*
*  EXECUTE MODULE
*
STAGS1.
REWIND,RACCT.TAPE90.
*
*  SECOND CALL OF ACCTPRG
*
RACCT,TAPE3.
REWIND,TAPE99.
*
*  GET ABSOLUTES FOR PROGRAM ACCTUPD
*
FETCH,STACT/UN=XXXXXXX.
*
*  GET OLD ACCOUNTING FILE
*
FETCH,TAPE98=STAGACT/UN=XXXXXXX.
*
*  PERFORM UPDATES
*
STACT.
*
*  REPLACE NEW ACCOUNTING FILE
*
REPLACE,TAPE97=STAGACT/UN=XXXXXXX.
RETURN,TAPE90,TAPE99.
*
*  SET UP DATA FOR TAPE 3
*
.DATA,TAPE3
 .   STAGS1
.EOF
REVERT.
```

Example 5 - Procedure File to Run the Accounting Package

```
STAGSC1.
USER,XXXXXXX,XXXXXX.
CHARGE,XXXXXX,LRC.
DELIVER.  XXXXXXX
GET,STGPROC/UN=XXXXXXX.
BEGIN,LSTAGS1,STGPROC,160000.
```

Example 6 - Job Stream to Run the Procedure File

REFERENCES

1. Control Data Corporation: NOS Version 1 Reference Manual, Volume 1 of 2. September, 1982, CDC Publication No. 60435400.

2. Control Data Corporation: NOS Version 1 Reference Manual, Volume 2 of 2. September, 1982, CDC Publication No. 60445300.

3. Computer Programing Manual, Volume 2 of 4, Section K5.3, NASA Langley Research Center, Report No. ACD-CAB N-948, October 1980.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-172414		

4. Title and Subtitle	5. Report Date
Accounting Utility for Determining Individual Usage of Production Level Software Systems	June 1984
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
Stacey C. Garber	
	10. Work Unit No.
9. Performing Organization Name and Address Kentron International, Inc. Aerospace Technologies Division 3221 N. Armistead Avenue Hampton, VA 23666	11. Contract or Grant No. NAS1-16000
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546	13. Type of Report and Period Covered Contractor Report
	14. Sponsoring Agency Code 505-33-53-10

15. Supplementary Notes

Technical Monitor: Norman F. Knight, Jr., NASA Langley Research Center, Mail Stop 190, Hampton, VA 23665
Final Report - Task P17-P333

16. Abstract

An accounting package has been developed which determines the computer resources utilized by a user during the execution of a particular program and updates a file containing accumulated resource totals. The accounting package is divided into two separate programs. The first program determines the total amount of computer resources utilized by a user during the execution of a particular program. The second program uses these totals to update a file containing accumulated totals of computer resources utilized by a user for a particular program. This package is useful to those persons who have several other users continually accessing and running programs from their accounts. The package provides the ability to determine which users are accessing and running specified programs along with their total level of usage.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
computer programming accounting	Unclassified - Unlimited Subject Category 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	30	A03

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