CR 151031

JSC-10917
Revision 1

CODE USAGE ANALYSIS SYSTEM
(CUAS)

Job Order 81-337

(NASA-CR-151031) CODE USAGE ANALYSIS SYSTEM (CUAS) (Lockheed Electronics Co.) 203 p HC A10/MF A01 CSCL 09B

N77-10809

Unclas
G3/61 08970

Prepared by

Lockheed Electronics Company, Inc.
Aerospace Systems Division
Houston, Texas

Contract NAS 9-12200

For

MISSION PLANNING AND ANALYSIS DIVISION

National Aeronautics and Space Administration

LYNDON B. JOHNSON SPACE CENTER

Houston, Texas

September 1976

LEC-8119
Revision 1

JSC-10917
Revision 1

CODE USAGE ANALYSIS CENTER
(CUAS)

Job Order 81-337

PREPARED BY

P.H. Horsley

P. H. Horsley
Support Software Section

J.D. Oliver

J. D. Oliver
Support Software Section

APPROVED BY

J. Philip Davis

J. P. Davis, Supervisor
Support Software Section

F N Barnes

F. N. Barnes, Manager
Dynamic Systems Department

Prepared By

Lockheed Electronics Company, Inc.

For

Mission Planning and Analysis Division

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
LYNDON B. JOHNSON SPACE CENTER
HOUSTON, TEXAS
September 1976

LEC-8119
Revision 1

TECHNICAL REPORT INDEX/ABSTRACT

(See instructions on reverse side.)

1. TITLE AND SUBTITLE OF DOCUMENT

Code Usage Analysis System (CUAS)

2. JSC NO.

JSC- 10917, Revision 1

3. CONTRACTOR/ORGANIZATION NAME

Lockheed Electronics Company, Inc.

4. CONTRACT OR GRANT NO.

NAS 9-12200

5. CONTRACTOR/ORIGINATOR DOCUMENT NO.

LEC-8119, Revision 1

6. PUBLICATION DATE (THIS ISSUE)

September 1976

7. SECURITY CLASSIFICATION

Unclassified

8. OPR (OFFICE OF PRIMARY RESPONSIBILITY)

M. A. Goodwin

9. LIMITATIONS

GOVERNMENT HAS UNLIMITED RIGHTS ☒ YES ☐ NO

IF NO, STATE LIMITATIONS AND AUTHORITY

10. AUTHOR(S)

P. H. Horsley
J. D. Oliver

11. DOCUMENT CONTRACT REFERENCES

WORK BREAKDOWN STRUCTURE NO.

N/A

CONTRACT EXHIBIT NO.

DRL NO. AND REVISION

DRL LINE ITEM NO.

12. HARDWARE CONFIGURATION

SYSTEM

N/A

SUBSYSTEM

MAJOR EQUIPMENT GROUP

13. ABSTRACT

The Code Usage Analysis System is a set of computer programs which are designed to aid the user in the interpretation of the execution characteristics of his application programs.

14. SUBJECT TERMS

CONTENTS

Section Page

APPENDIX

TABLES

FIGURES

CODE USAGE ANALYSIS SYSTEM

(CUAS)

1. INTRODUCTION

This document provides information related to the design, operation, and use of the Code Usage Analysis System (CUAS). Included in this document are descriptions of the system capabilities, method of operation, and a user reference guide.

The CUAS is intended to aid the user in developing and performing evaluation of application programs. This objective is realized by providing the user with reports of subroutine usage, program errors, and segment loading which occurred during the execution of an application program.

The CUAS was evolved specifically to aid in the development and validation of the Space Vehicle Dynamics Simulation (SVDS).

2. CUAS DESCRIPTION

2.1 DEFINITION OF TERMS

Contingency subroutine — A contingency is an abnormal condition, often associated with an interrupt, which may occur during execution of a program. EXEC 8 allows a user to register a subroutine to process contingencies and transfers control to the registered contingency subroutine should any abnormal condition occur.

CPU — Central Processing Unit, a hardware device capable of interpreting instructions and performing the indicated operation

CUAS — Code Usage Analysis System

Element — A named grouping of data, typically manipulated as a unit; as used in this document contains a logical program part such as a subroutine

External definition — An address within a closed subroutine which may be referenced from code which is not a part of the closed subroutine

Interrupt — A hardware facility that causes a CPU to suspend execution, save the machine state, and transfer control to a specific address

Invalid code — Also IOPR (illegal operator); a machine instruction code which is not a member of the set of valid machine instruction codes

J — Jump Keys, a mnemonic for a valid UNIVAC-1100 processor instruction which transfers control to a specific address

JHS — Jump History Stack, a name given to a file of data created by the CUAS contingency routine during the execution of an application program

LMJ — Load Modifier and Jump, a mnemonic for a valid UNIVAC-1100 processor instruction which saves the current value of the P register and transfers control to a specific address

Master bit notation	A notation for representing execution options in one cell. Within the cell, bit 25 is set on if option A was specified and bit 0 on if option Z was specified
PCT	Program Control Table
Primary Storage	General purpose constant access time storage directly addressable by the CPU and serving generally to contain executing programs
PSR	Processor State Register
Secondary storage	General purpose nonconstant access time storage typically available to the CPU via a peripheral processor or channel, and serving generally to contain nonexecuting programs and data
SUP	Standard Unit of Processing
SVDS	Space Vehicle Dynamics Simulation
TCURS	Test Case Usage Reporting System

2.2 SYSTEM CAPABILITIES

The objective of the CUAS is to apply software technology for questions concerning software performance and quality which have historically been answered by detailed and laborious manual techniques costing considerable time and expense. The typical method proposed and implemented for automating program performance analysis through software technology involves source code modification, which may become burdensome for the user to apply and impose a considerable primary storage overhead penalty, making the technique difficult or impossible to apply. The technique used in the CUAS does not involve source code modification, has a small constant primary storage overhead which for most application programs should be negligible, and affords the user more information than is possible with a typical source code modification technique.

Specifically, the CUAS provides the user the following information concerning the execution of his application program.

2.2.1 EXTERNAL USAGE REPORT

This report consists of three alphabetically ordered lists: (1) Every user-supplied external definition name included in the application program; (2) Every user-supplied external definition name referenced as a subroutine during program execution and optionally the number of references and execution time; and (3) Every user-supplied external definition name not referenced as a subroutine during program execution.

2.2.2 ERROR LOCATION REPORT

This report locates for the user the element name, relative location within the element, and overlay segment name within which a program error (such as a floating point overflow) occurred. A walk back which traces the calling sequence from the main program element to the element in which the error occurred is also provided at the option of the user.

2.2.3 SEGMENT LOADING REPORT

This report informs the user how many times each segment of an overlaid program was transferred from secondary storage into primary storage during the execution of a segmented and overlaid program. An itemized report of which subroutines were called to result in the segment begin loaded is provided for each segment loaded at least once.

2.2.4 TEST CASE ELEMENT GENERATION REPORT

This report is generated only when specifically requested by the user. When requested, the CUAS creates in an EXEC 8 program file a source element containing the names of those subroutine elements called at least once and those not called. The report merely informs the user that the source element has been created. The source element created may be used by the Test Case Usage Reporting System (TCURS) in producing a cross-reference listing of subroutines used and not used within a program file from which several absolute elements may be generated.

2.3 METHOD OF SOLUTION

2.3.1 EXTERNAL USAGE REPORT

The external usage reporting technique involves modification of the machine instructions in an absolute program element after it has been prepared by the UNIVAC MAP processor and prior to its execution. A typical Central Processing Unit (CPU) of a computer functions in this way: The machine instructions of a program are loaded into sequential contiguous primary storage cells and the CPU is given the address of the primary storage cell at which instruction execution is to begin. The CPU proceeds from the starting address, executing the instructions one at a time in sequence, broken only when the CPU encounters a jump-type instruction.

The jump instruction directs the CPU to execute an instruction at an address contained within the jump instruction as the next step rather than the next sequential instruction. The CPU will execute the instruction at the address contained in the jump instruction and proceed sequentially until another jump instruction is encountered, as illustrated in figure 2-1.

The CPU provides the user one other way of changing sequential instruction execution flow besides the jump-type instruction, i.e., the CPU interrupt. The interrupt can change the address of the next instruction to be executed by suspending the CPU at any point in its sequential instruction processing.

Usually, when the interrupt occurs, the address of the next sequential instruction to be executed is saved, and the address of a fixed location in primary storage becomes the address of the next instruction to execute. The interrupt may be caused by a stimulus external or internal to the CPU. The sequential nature of a computer program and the CPU interrupt are both incorporated into the CUAS technique.

One would realized immediately that in a program containing no jump instructions, the CPU executes every instruction in the program one time only. When jump instructions are introduced into the code, it is necessary to know when

FORTRAN SOURCE CODE

```
I = 1
CALL SUBA(I)
STOP
END
SUBROUTINE SUBA(J)
J = J + 1
RETURN
END
```

UNIVAC 1100 MACHINE CODE CORRESPONDING TO FORTRAN SOURCE CODE

```
ADDRESS    INSTRUCTION    OPERAND       COMMENT

   0          LA,U        A0,1          Set A0 = 1
   1          SA          A0,6          Store A0 at address 6
   2          LMJ         X11,7         Set X11=3, jump to address 7
   3          +           6             Parameter address
   4          +           0             Vacant cell
   5          ER          EXIT$         Program stop
   6          +           0             Vacant cell
   7          LA          A0,*0,X11     Load parameter into A0
  10          AA,U        A0,1          Add 1 to A0
  11          SA          A0,*0,X11     Store A0 at parameter address
  12          J           2,X11         Jump to address 5
```

Figure 2-1.- Sample UNIVAC-1100 programming code.

a jump is executed and the address jumped to for determination of a program execution path. If the execution of each jump-to-subroutine type instruction is recorded into a jump history stack, this run statistic may be examined to determine the execution frequency of subroutines within a program. If the amount of computer time expended in a subroutine is to be determined, the execution of each subroutine return type instruction must also be recorded into a Jump History Stack (JHS) as well as the amount of time expended between the subroutine call and return.

The CUAS records the fact that Load Modifier and Jump (LMJ) and Jump Keys (J) type instructions would have been executed in the following way.

1. Prior to execution, the instructions of an absolute element are scanned for LMJ and J type instructions. Any LMJ instruction which transfers control to an address external to the element in which it is contained is changed to the invalid code $(7700)_8$, unless control is transferred to a FORTRAN library routine. Any J instruction which uses indirect addressing or in which the address field of the instruction is external to the element in which it is contained is changed to the invalid code $(7701)_8$. These conventions for instruction modification avoid modifying the majority of LMJ or J instructions which are not associated with a subroutine call or return.

2. When the CPU attempts to execute an invalid instruction, it interrupts itself and passes control to an interrupt call in EXEC 8.

3. EXEC 8 executes a user-supplied contingency routine which examines the invalid code that the CPU attempted to execute. The instruction type and the address to which the original instruction would have transferred control are determined and recorded into a JHS on secondary storage. If subroutine timing is being performed, the CPU, executive, and I/O (input/output) times that have elapsed since the last invalid instruction trap are recorded into the JHS. A software execution of the original instruction is performed and program execution continues exactly as if no valid instruction contingency had occurred. The execution of a nonmodified and

a modified LMJ instruction is illustrated in figure 2-2. The technique used by the CUAS for determining elapsed computer time is described in detail in Appendix A.

2.3.2 ERROR LOCATION REPORT

In the course of the execution of an application program under EXEC 8, the occurrence of any of the following errors will result in a CPU interrupt.

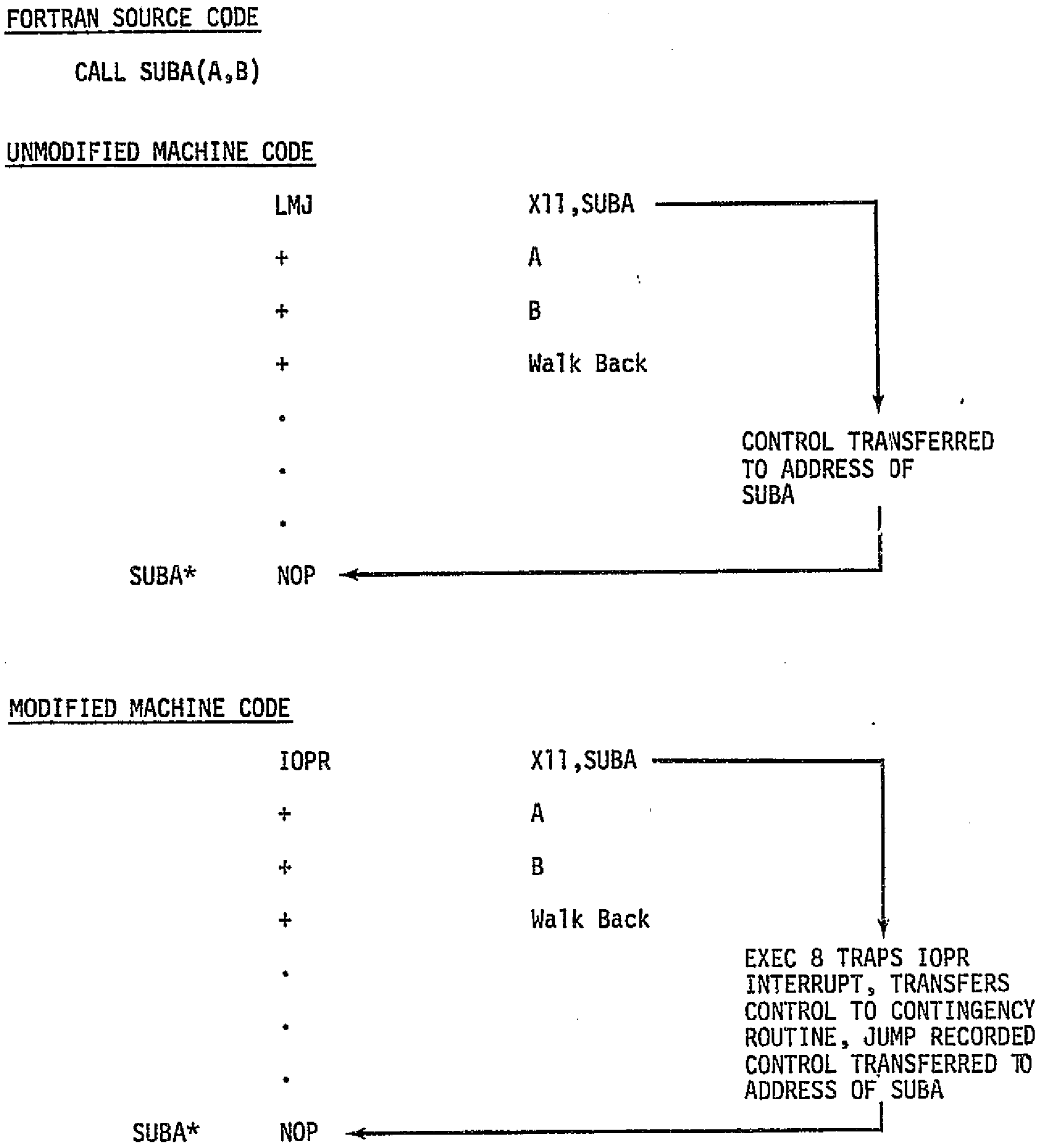
1. Storage limits violation (Guard Mode)
2. Floating-point overflow
3. Floating-point underflow
4. Divide fault

Upon the occurrence of the interrupt, EXEC 8 will execute a user-supplied contingency subroutine within which the user may take action as he deems necessary to overcome the error condition and then either continue or terminate the program execution. In addition to the above hardware-detected errors, two software-detected errors, ERROR MODE and user-requested ABORT, result in EXEC 8 executing the user-supplied contingency subroutine.

The CUAS uses a contingency subroutine to record the type and location of the error into the JHS and then continues the program execution from the interrupted point for arithmetic-type errors, or terminates the program execution for all other type errors.

2.3.3 SEGMENT LOADING REPORT

When an application program is segmented and overlaid and segment loading is to occur by the indirect method, that is, load on call, a system-supplied subroutine performs the segment loading automatically. In the CUAS, the indirect segment loading routine is a part of the contingency subroutine and has the expanded capability of recording into the JHS the index of the segment loaded as well as loading the segment. The segment loading procedure follows the normal method of inspection of the segment load table and actual segment loading by request to EXEC 8.



FORTRAN SOURCE CODE

```
     CALL SUBA(A,B)
```

UNMODIFIED MACHINE CODE

```
                    LMJ             X11,SUBA
                    +               A
                    +               B
                    +               Walk Back
                    .
                    .
                    .
          SUBA*     NOP
```

CONTROL TRANSFERRED TO ADDRESS OF SUBA

MODIFIED MACHINE CODE

```
                    IOPR            X11,SUBA
                    +               A
                    +               B
                    +               Walk Back
                    .
                    .
                    .
          SUBA*     NOP
```

EXEC 8 TRAPS IOPR INTERRUPT, TRANSFERS CONTROL TO CONTINGENCY ROUTINE, JUMP RECORDED CONTROL TRANSFERRED TO ADDRESS OF SUBA

Figure 2-2.- Operation of unmodified/modified code.

2.3.4 TEST CASE ELEMENT GENERATION REPORT

The external usage report generates the data for a source element of those subroutines called and not called, and the UNIVAC program file element maintenance package SOR is utilized to create the element. The element created may be manipulated with the UNIVAC FURPUR commands exactly as any other UNIVAC SDF source element. The format of the element is covered in detail in Appendix B.

2.3.5 CUAS COMPONENTS

The CUAS consists of a preprocessor, a contingency subroutine, and a postprocessor. The first step in the application of the system is for the user to collect his program, including the CUAS contingency subroutine. The first executable statement of every FORTRAN program under EXEC 8 is a subroutine call to the external reference NINTR$. The FORTRAN library contains a standard contingency subroutine named NINTR$ which will satisfy this external reference. The CUAS contingency subroutine also will satisfy the external reference NINTR$ and so will override the library-supplied routine.

The absolute element thus obtained will execute and operate identically the same as if the library-supplied NINTR$ subroutine had been included in the program. The CUAS preprocessor is used to prepare the absolute element such that the JHS will be created during the program execution. The CUAS preprocessor scans the code of each user-supplied subroutine in the absolute element and changes the operation field of LMJ and J type instructions to an invalid code, as required to generate the report detail requested by the user through execution options. The preprocessor uses the diagnostic tables from the absolute element (which are shown in detail in Appendix C) to locate the I-BANK code for each user-supplied subroutine within the absolute element. The preprocessor sets a sentinel cell within the contingency routine so that this routine may determine that the code has been modified and the JHS is to be created.

When the absolute element thus prepared by the preprocessor is executed, the jump history file will be created. The creation of this file is completely transparent to the user, and his program will apparently operate identically as if the code had not been modified.

Once the execution of the user's program is complete, the CUAS postprocessor is executed to prepare the reports of the execution from the JHS. The postprocessor also uses the diagnostic tables from the absolute element to correlate the addresses contained in the jump history stack file to user-recognizable element, segment, and external reference names.

The total operation of the CUAS is depicted in figure 2-3, and the content and format of the jump history stack file is described in Appendix D. The Test Case Usage Reporting System is described in Appendix E.

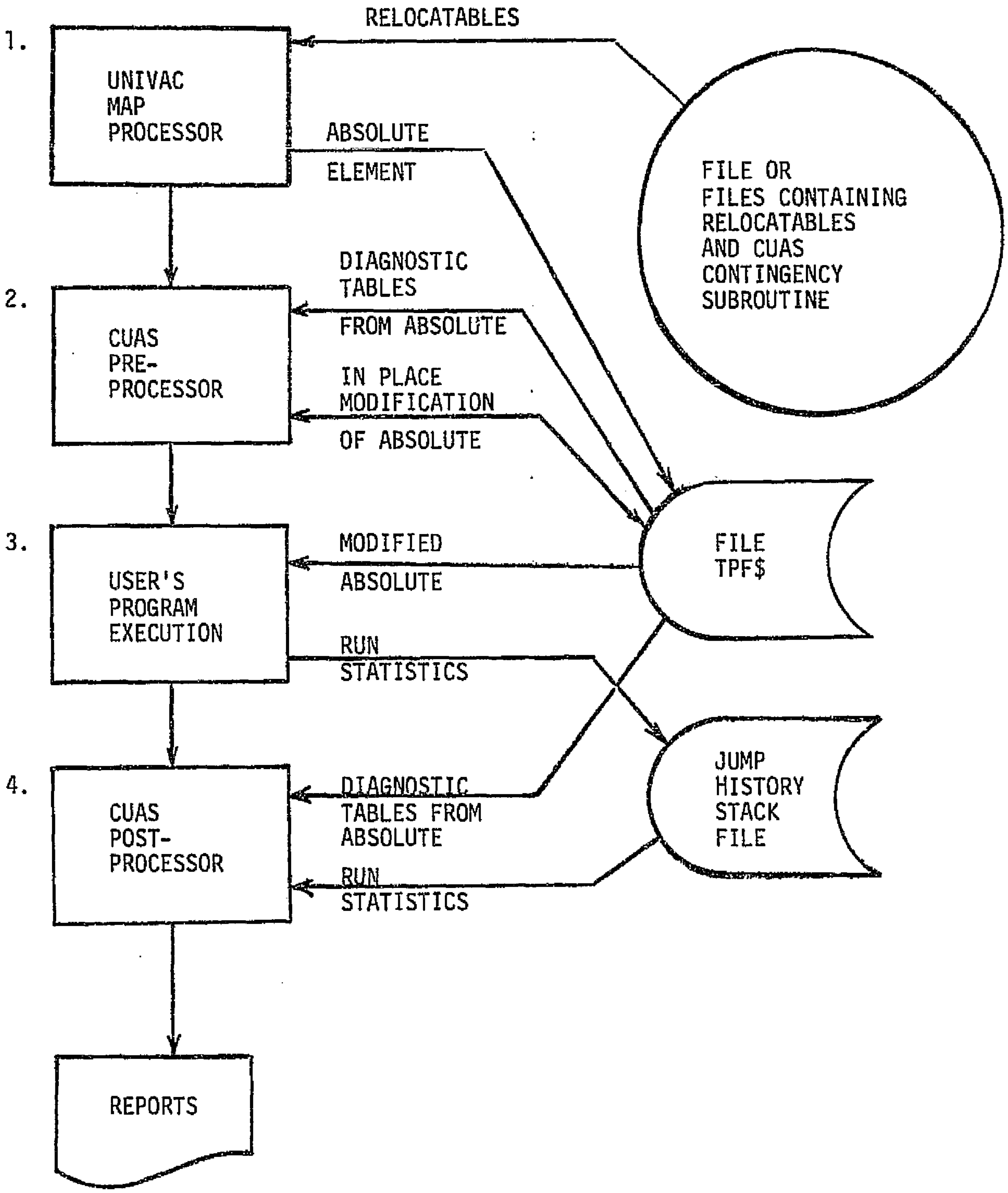


Figure 2-3.- CUAS technique.

3. USER REFERENCE GUIDE

3.1 METHOD OF USE

The CUAS has been implemented on the UNIVAC 1110 Series, EXEC 8 operating system and may be accessed from the secure file FML-L79351*PHPA., hereafter referred to as file X. In order for a JHS to be created when a program is executed, the collection of the absolute program element must include the element IICONT from file X. This is easily achieved by copying the element IICONT into the file TPF$ just prior to collection of the absolute element. Once the absolute element is collected, the CUAS preprocessor element CUAPREPRO must be executed to prepare the absolute element to create the jump history stack. The CUAS preprocessor automatically prepares the last absolute element inserted into the file TPF$. The absolute element thus prepared may now be executed in the normal manner, and during its execution a file named JHS will be produced containing the jump history stack for the program. Once the execution is complete, the CUAS postprocessor element CUAPSTPRO must be executed to produce the code usage reports. The CUAS postprocessor automatically reads the file named JHS and compares it with data tables from the last absolute element inserted into the file TPF$. For proper execution, the absolute element producing the file named JHS must be the last absolute element inserted into the file TPF$ at the time the postprocessor is executed. A typical run stream including the utilization of the CUAS is depicted in figure 3-1.

3.1.1 CUAS PREPROCESSOR OPTIONS

When the preprocessor is executed with no option, the postprocessor external usage report will contain the names of those subroutines called and the total number of calls to each. If the user wishes the execution time of each subroutine in addition to the calling frequency, the option C must be inserted on the preprocessor execute statement in the following way.

```
@XQT,C  X.CUAPREPRO
```

RUN STREAM

```
1.  @RUN

2.  @USE X.,FML-L79351*PHPA.

3.  @COPY,[illegible] X.IICONT,TPF$.

4.  @MAP,S  USERFILE.USERMAP,ABSELT

5.  @XQT    X.CUAPREPRO

6.  @XQT    ABSELT

7.  @XQT    X.CUAPSTPRO

8.  @FIN
```

RUN STREAM DESCRIPTION

1. Required statement to initiate run on EXEC 8.
2. Assign reference name X to the CUAS file.
3. Copy the element IICONT into the run's TPF$ file.
4. Collection of absolute in normal way and saved in run's TPF$ file.
5. The CUAS preprocessor is executed to prepare the absolute element ABSELT for producing a JHS.
6. The absolute element ABSELT executed in the normal way.
7. The CUAS postprocessor is executed to produce its reports once the execution of ABSELT is complete.
8. Required statement to end run on EXEC 8.

Figure 3-1.- Typical run stream including CUAS.

Use of this option gives the user the maximum reporting capability of the CUAS and also requires the greatest amount of execution time overhead. The times reported will not include CUAS execution overhead, and will reflect the time required to execute the subroutines in a normal untraced execution.

If the user is interested in determining only if an external definition has been referenced or not during execution, run time overhead may be reduced by inserting the option A on the preprocessor execute statement in the following way.

```
@XQT,A  X.CUAPREPRO
```

When the user's absolute element is then executed, only the first occurrence of an LMJ to an external definition will be trapped and all successive occurrences will operate normally. This option trades off less reporting capability for less execution time overhead and a small jump history stack file. When this option is used, the postprocessor will report only if an external name was referenced or not, and no error walk backs will be produced in the error location report. The error location report will still be available but will be limited to the location of the element and segment in which each unique error occurred.

If the user is not interested in any external usage report and desires only error location and/or segment loading reports, run time overhead may be further reduced by inserting option B on the preprocessor execute statement in the following way.

```
@XQT,B  X.CUAPREPRO
```

When the user's absolute element is then executed, LMJ instructions will be processed in the normal manner and only error conditions and segment loads will be trapped. When this option is used, the postprocessor will produce no external usage report and no error walk backs will be produced in the error location report.

3.1.2 CUAS POSTPROCESSOR OPTIONS

The CUAS postprocessor options afford the user the capability to limit the reports produced by the postprocessor to those he desires. The user may choose none or one or more of the following options to be included on the execute card for the postprocessor.

Option — Meaning

X — Do not produce the external usage report.

E — Do not produce the error location report.

S — Do not produce the segment loading report.

W — Include a walk back for each error located in the error location report. Walk backs are not possible if either option A or B was specified on the CUAS preprocessor execution.

G — Generate a source element for the TCURS, and report the generation of this element. Generation of this element will not be possible if option B was specified on the CUAS preprocessor. When this option is used, following the postprocessor execution card must be one additional card on which is specified in columns 1-8 the name for the element to be created. The element is created in a program file with the internal file name DBF which the CUAS postprocessor will automatically assign and free so that the user need only equate the name DBF to his secure file name with an EXEC 8 @USE directive. Options X and G must not be used together, since the external usage

Option	Meaning
	report must be generated if the test case element is to be created. The CUAS post-processor will always include the version name TESTCASE on every test case element it generates in the file DBF.
blank	When no option is specified, all possible reporzs will be generated, with the exception of the test case element generation report.

As an example, the following postprocessor execution would generate a test case source element and would not produce the segment loading report.

```
@XQT,GS  X.CUAPSTPRO
```

3.1.3 CUAS TIMING OPTION CONSIDERATIONS

When the option C is specified to the CUAS preprocessor, a report of charge time by subroutine will be generated. The CUAS does not trace calls to elements from the UNIVAC SYS$*RLIB$ library file or to any element which includes one or more $ characters in its name. The time spent within these elements is accrued to the user-supplied element from which they were called. The charges reported in this manner accurately reflect the time necessary to execute the users-supplied FORTRAN source code, but do not provide a breakdown of charges for the execution of UNIVAC-supplied support subroutines. The timing of UNIVAC-supplied support routines would probably be only of academic interest to the application programmer since typically it is not within the scope of his job or interest to revise or modify such software. Such software may be indirectly evaluated by the CUAS by creating a FORTRAN source element that only references a library support software function, such as the FORTRAN WRITE statement. The time charged to such a routine would then be a charge evaluation of those routines which support the WRITE statement.

The CUAS should not be used to evaluate a long-running iterative subroutine calling sequence program unless an execution test case is chosen which invokes only one or two loops through the calling sequence. Test cases which invoke several loops will not enhance the timing data but will cost the user a considerable amount of computer time for timing overhead.

3.1.4 USER CONTROL OF OVERHEAD

The file JHS in which the execution statistics of a program are stored is automatically created by the CUAS contingency routine IICONT. This file has a default maximum size of 10,000 tracks which are allocated in increments of 100 tracks as needed, with each track containing 1792 36-bit cells of data. In general, the user should estimate for each track of data stored by subroutine IICONT an additional 1-second SUP charge above and beyond the normal running time for his program when not being analyzed by the CUAS. The CUAS postprocessor requires approximately a 1-second SUP charge for processing each track of data produced by subroutine IICONT, so that the approximate total overhead per track is 2 SUP seconds. The above overhead estimates are for the maximum overhead case, which is the subroutine timing report generation capability of the CUAS; when other reporting options are chosen, the overhead per track will be somewhat less.

The user may limit the number of tracks of data created and processed by the CUAS by overriding the default maximum size of the file JHS. This is done by assigning the file JHS with an EXEC 8 assign statement prior to the execution of the user's program which is to be traced. An assignment statement which limits the maximum JHS file size to 50 tracks might be prepared in the following way.

```
@ASG,T  JHS.,F4/50//50
```

When the maximum size of the file JHS has been reached, subroutine IICONT discontinues all tracing and the application program continues its processing from that point in the normal way. When the JHS file is filled and closed prior to the end of program execution, the CUAS postprocessor will so inform the user in its report, and the G option will be disabled. The production of an element containing the names of subroutines used and not used may not correct when program tracing is discontinued prior to program completion.

3.1.5 CUAS OPERATIONAL OPTIONS

If the file named JHS and the absolute element creating the file are both saved, the CUAS postprocessor may be executed in a job run some time after the one in which the file JHS was created. Multiple executions of the postprocessor may also be done in the same job or a later job. The following two requirements are necessary and sufficient for the execution of the CUAS postprocessor.

1. The absolute element which created the file JHS is present in the file TPF$ and is the last absolute element inserted into the file TPF$.
2. The jump history stack file is attached to the job, has the internal file name JHS, and contains a jump history stack for the execution of the identical absolute element in the file TPF$.

If both of the above conditions are not met, the postprocessor will provide a diagnostic message informing the user of the problem and terminate without producing any reports.

If the CUAS contingency subroutine IICONT has been included in an absolute element but the CUAS preprocessor is not executed prior to the execution of that absolute element, no jump history stack file will be created, no execution time overhead will be incurred, and the element will operate identically as if the standard FORTRAN contingency routine were present. Once an absolute element has been modified by the CUAS preprocessor, each successive execution of that absolute will produce a new jump history stack file without further execution of the CUAS preprocessor. Once an absolute element has been created which includes the CUAS contingency subroutine IICONT, any sp cific execution of that absolute may be analyzed by copying it into the file TPF$ and executing the CUAS preprocessor just prior to the execution of the absolute.

3.2 ERROR MESSAGES

When the CUAS detects an error or option conflict condition, a message is generated which fully describes the condition, giving enough information to allow the user to take remedial action. The CUAS utilizes dynamic core expansion and may be executed from an EXEC 8 demand terminal. The CUAS will not expand core above 20K when it is executed as a demand job, and an error message is provided. Both the pre- and postprocessor require approximately 10K static core, leaving 10K for data area expansion.

4. EXECUTION CHARACTERISTICS

4.1 RESTRICTIONS

The CUAS is implemented for operation with FORTRAN programs including any UNIVAC 1100 ASSEMBLER programs which conform to FORTRAN conventions. The concept of the CUAS should be applicable to any EXEC 8 single activity application program in which code and data have been separated so that each may be recognized, i.e., I-BANKs and D-BANKs. The CUAS, as implemented, assumes that all the code for a user-supplied subroutine may be found in the absolute element I-BANK. The CUAS will not operate properly with programs which involve multi-activites, reentrant code, multibanking, or common banks, and intermixed code and data in the program's I-BANK.

The above restrictions were designed into the CUAS after identifying the potential users of the CUAS as FORTRAN application programs. The restrictions seemed reasonable in view of the fact that the CUAS will perform its intended purpose within the environment of FORTRAN application programs and globally expanding the capability of the CUAS would significantly increase the execution and primary storage overhead associated with the technique.

4.2 EXECUTION OVERHEAD

The execution overhead of the CUAS involves additional primary storage for the contingency subroutine above that which would be used for the standard contingency subroutine NINTR$, and additional execution time to perform software executions of an LMJ instruction. The primary storage overhead is always a fixed constant, approximately 800 storage cells above the version of NINTR$ which would have otherwise been used by the application program. The user may arrive at an exact figure by comparing the length of the NINTR$ version which he would normally use with the CUAS version of NINTR$. The execution time overhead has been found to be approximately 1 millisecond total Standard Unit of Processing (SUP) additional charge for each contingency interrupt trapped. A SUP is used on UNIVAC EXEC 8 systems for determining computer use charges and is comprised of CPU time, I/O time, and executive time. The 1 millisecond figure was arrived at by comparing execution SUP charges for the execution of a program without the CUAS applied and the identical execution of that program with the CUAS applied. Additional SUP charges accrued when the CUAS was applied were divided by the number of contingency interrupts to arrive at an average charge per interrupt. This technique was applied to several application programs to project an overall average expected execution time overhead.

When the CUAS timing report is invoked by option C on the preprocessor execution card, a very large JHS file will very likely be created. The size is dependent on the number of subroutine calls and returns trapped. The size of the JHS will be an order of seven times larger than it is when only subroutine frequency is requested by no option on the preprocessor execution card. The amount of data captured in a timing analysis will require a considerable amount of executive and I/O time and typically the user should expect an increase of three to four times in the total charges to execute the program. An absolute value cannot be placed on the increase since it is dependent entirely on the number of subroutine calls in the program being traced. In general, the CUAS should not be used to time long-running iterative calling loop programs unless the execution test case invokes only one or two loops through the program. Such analysis will supply the user with the cost data for each subroutine and will not involve a large overhead.

4.3 ACCURACY/VALIDITY

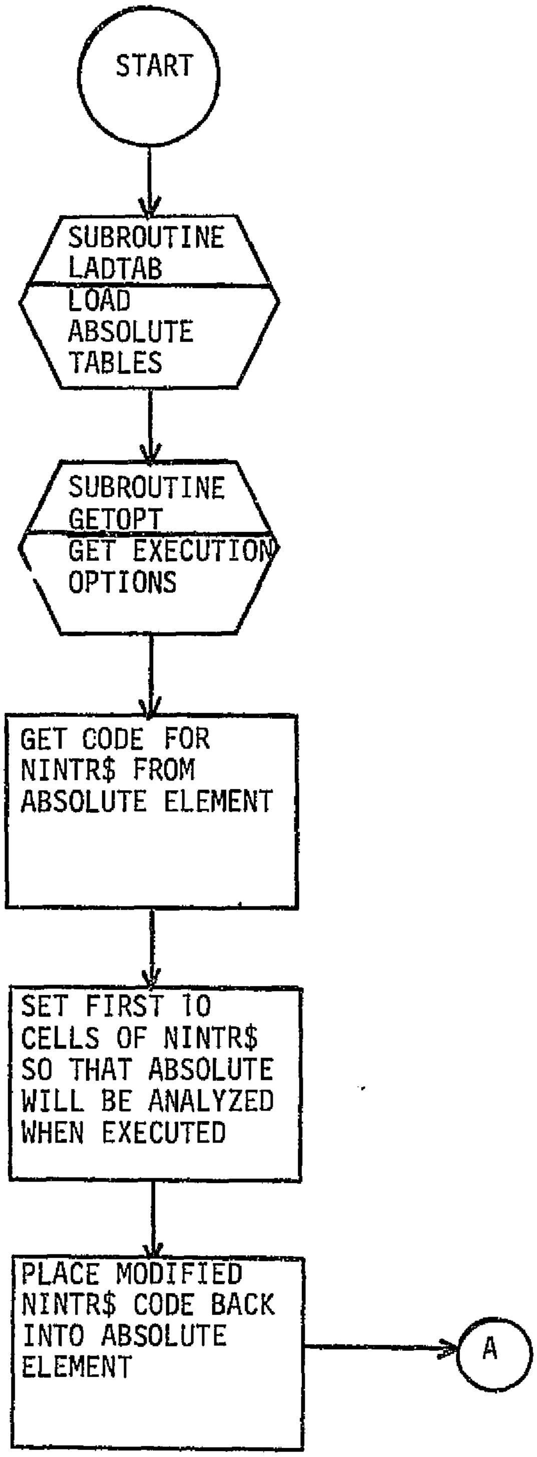
The CUAS was verified by applying it in the analysis of application programs whose execution characteristics were known. The results of the CUAS reports for such programs were desk checked to ensure that they reflected accurately what was known to have happened during the execution of the program.

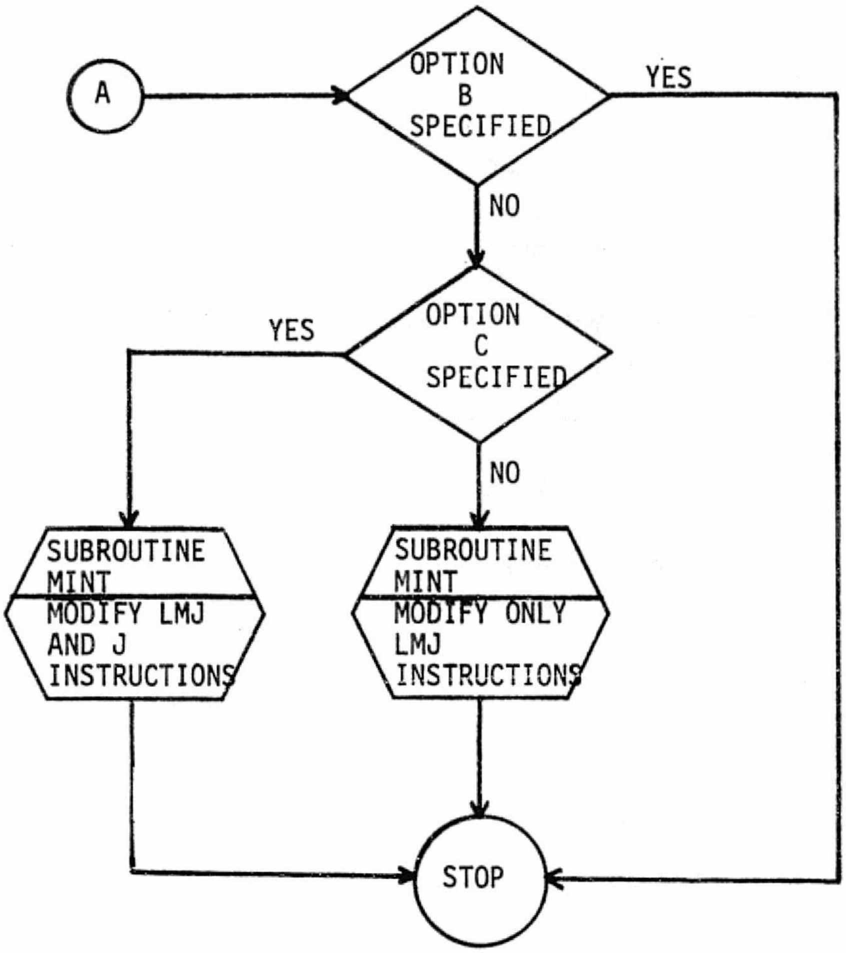
4.4 COMPUTER IMPLEMENTATION

The CUAS is designed and programmed to operate only on UNIVAC 1100 series computers which are operated with the UNIVAC EXEC 8 operation system. As of this writing, the CUAS is compatible with UNIVAC EXEC 8 level 31.244, update level D, UNIVAC FORTRAN V level E3 and UNIVAC MAP level 27.1. The entire CUAS is coded in the FORTRAN and ASSEMBLER languages with approximately 95 percent of the code in FORTRAN. The reference version of the CUAS was implemented at the NASA/JSC computing center in June 1976.

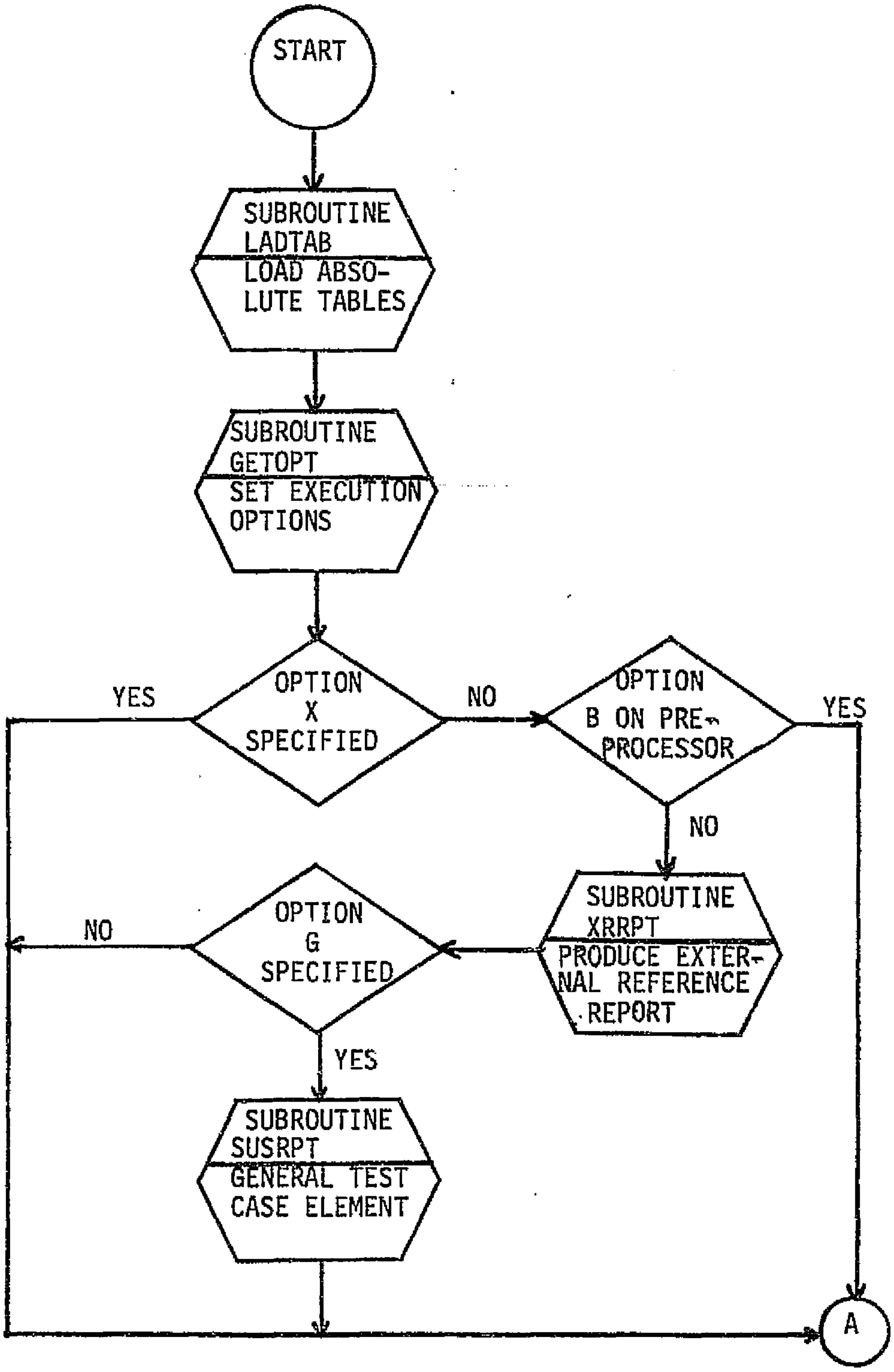
5. REFERENCE INFORMATION

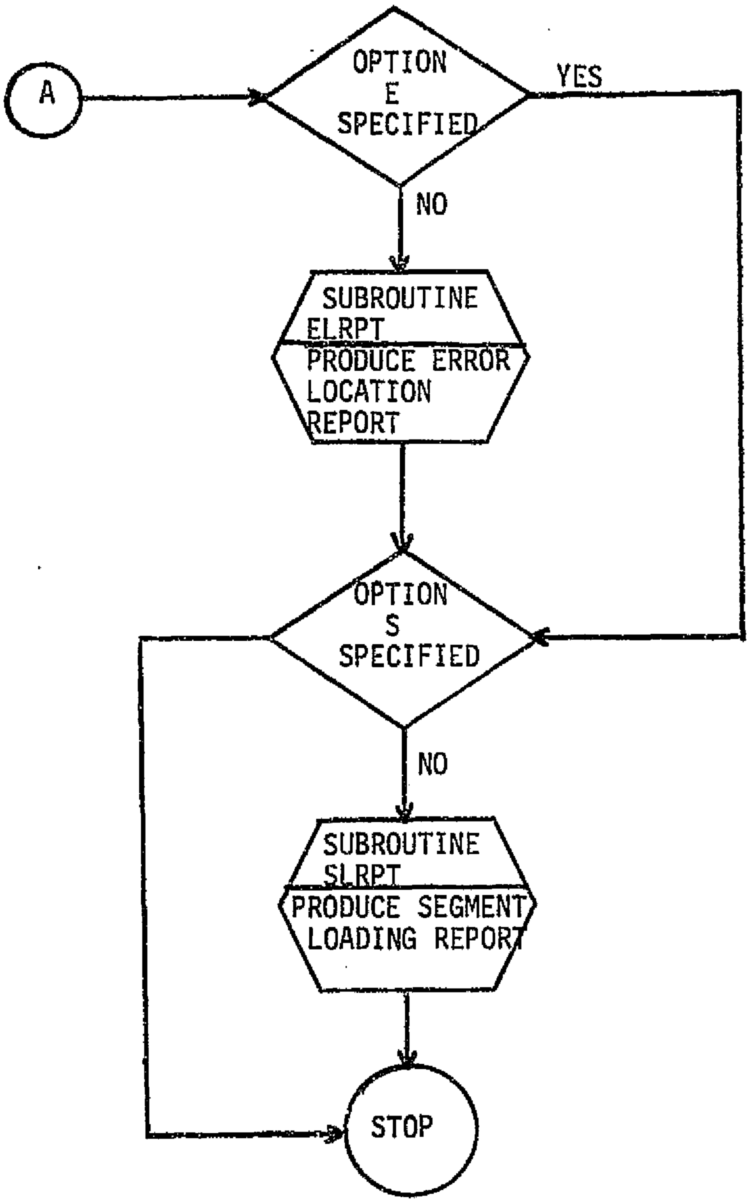
5.1 FUNCTIONAL FLOWCHART OF CUAS PREPROCESSOR


START
SUBROUTINE LADTAB
LOAD ABSOLUTE TABLES
SUBROUTINE GETOPT
GET EXECUTION OPTIONS
GET CODE FOR NINTR$ FROM ABSOLUTE ELEMENT
SET FIRST 10 CELLS OF NINTR$ SO THAT ABSOLUTE WILL BE ANALYZED WHEN EXECUTED
PLACE MODIFIED NINTR$ CODE BACK INTO ABSOLUTE ELEMENT
A

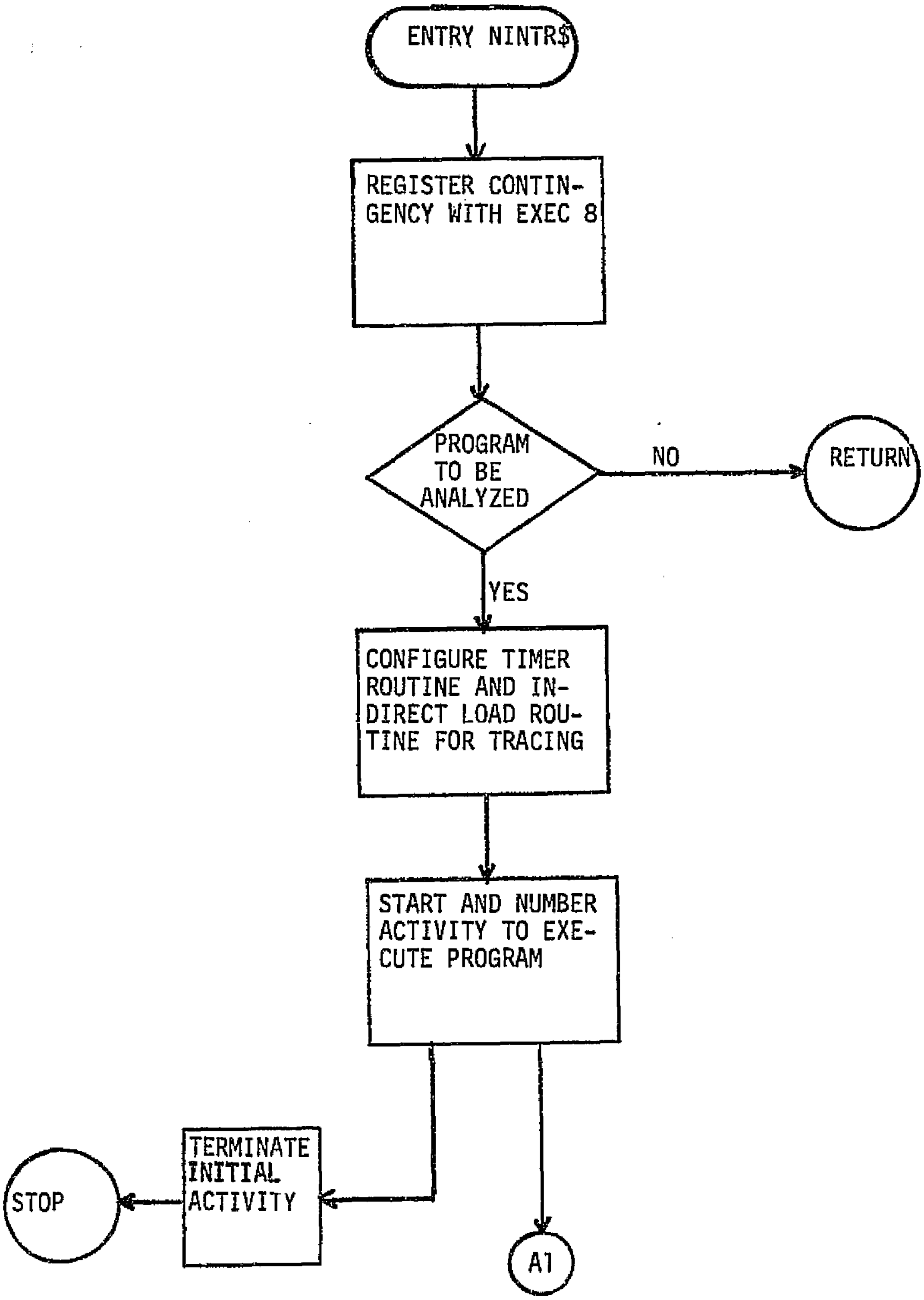

A
OPTION B SPECIFIED
YES
NO
OPTION C SPECIFIED
YES
NO
SUBROUTINE MINT
MODIFY LMJ AND J INSTRUCTIONS
SUBROUTINE MINT
MODIFY ONLY LMJ INSTRUCTIONS
STOP

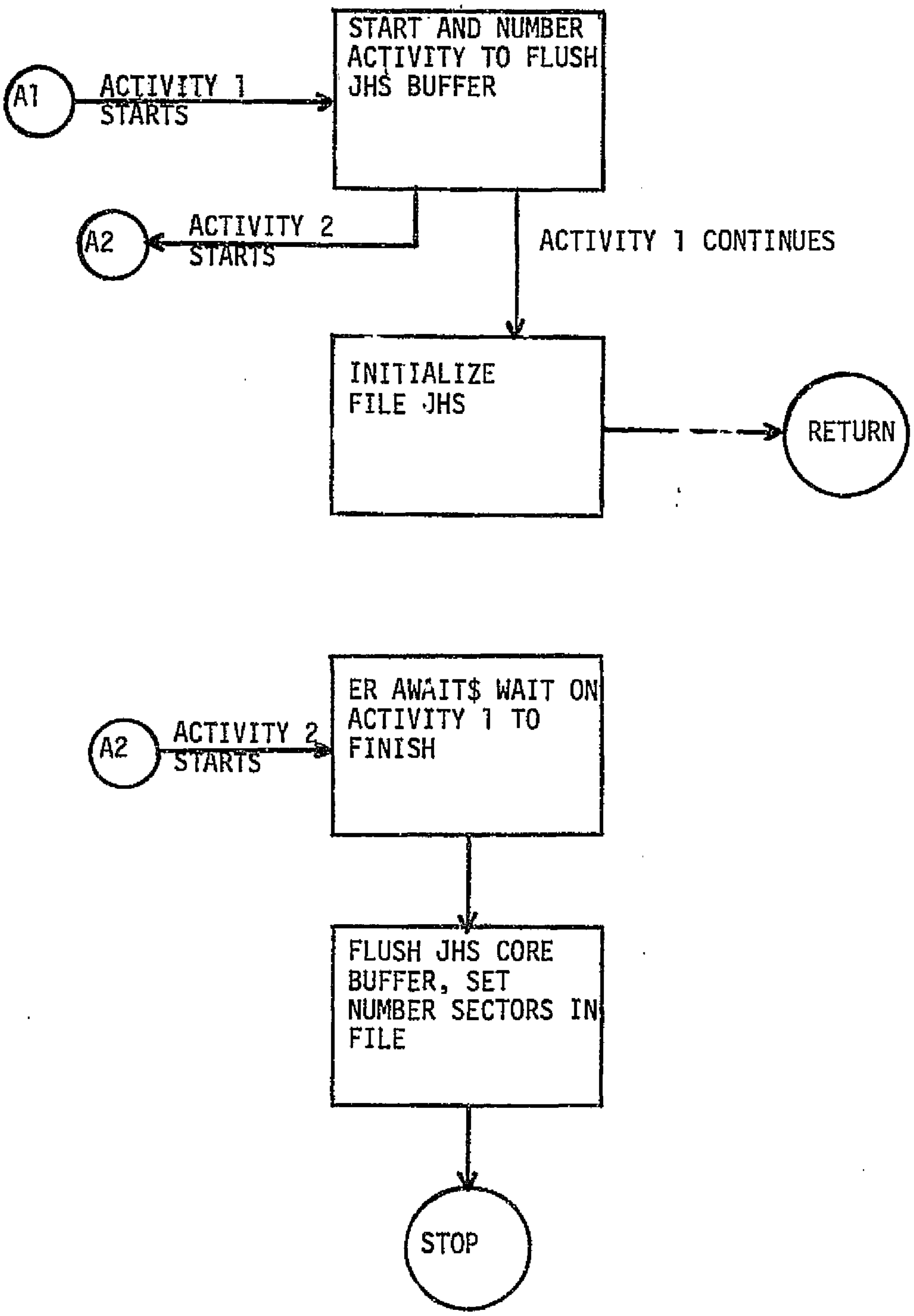
5.2 FUNCTIONAL FLOWCHART OF CUAS POSTPROCESSOR



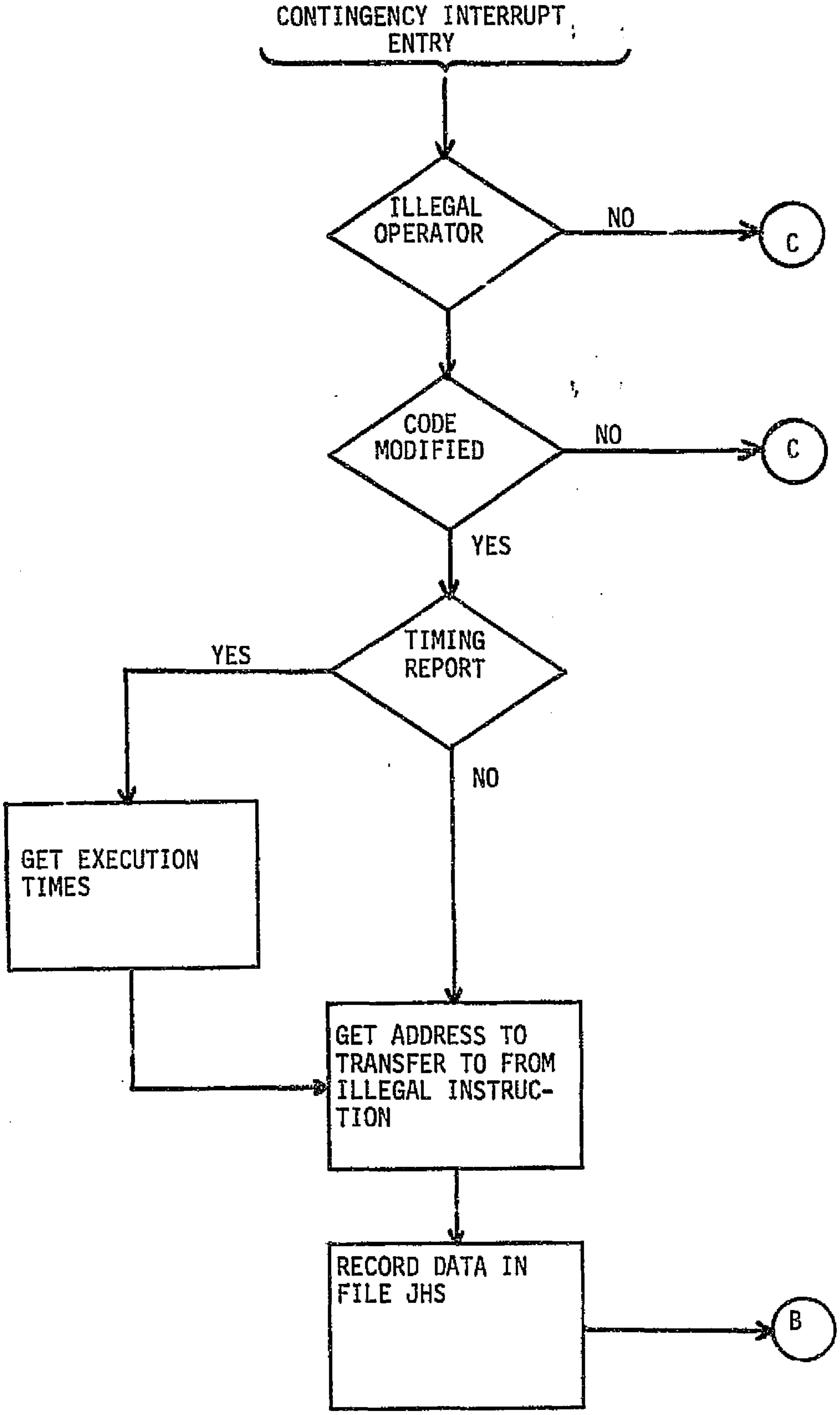

A
OPTION E SPECIFIED
YES
NO
SUBROUTINE ELRPT
PRODUCE ERROR LOCATION REPORT
OPTION S SPECIFIED
NO
SUBROUTINE SLRPT
PRODUCE SEGMENT LOADING REPORT
STOP

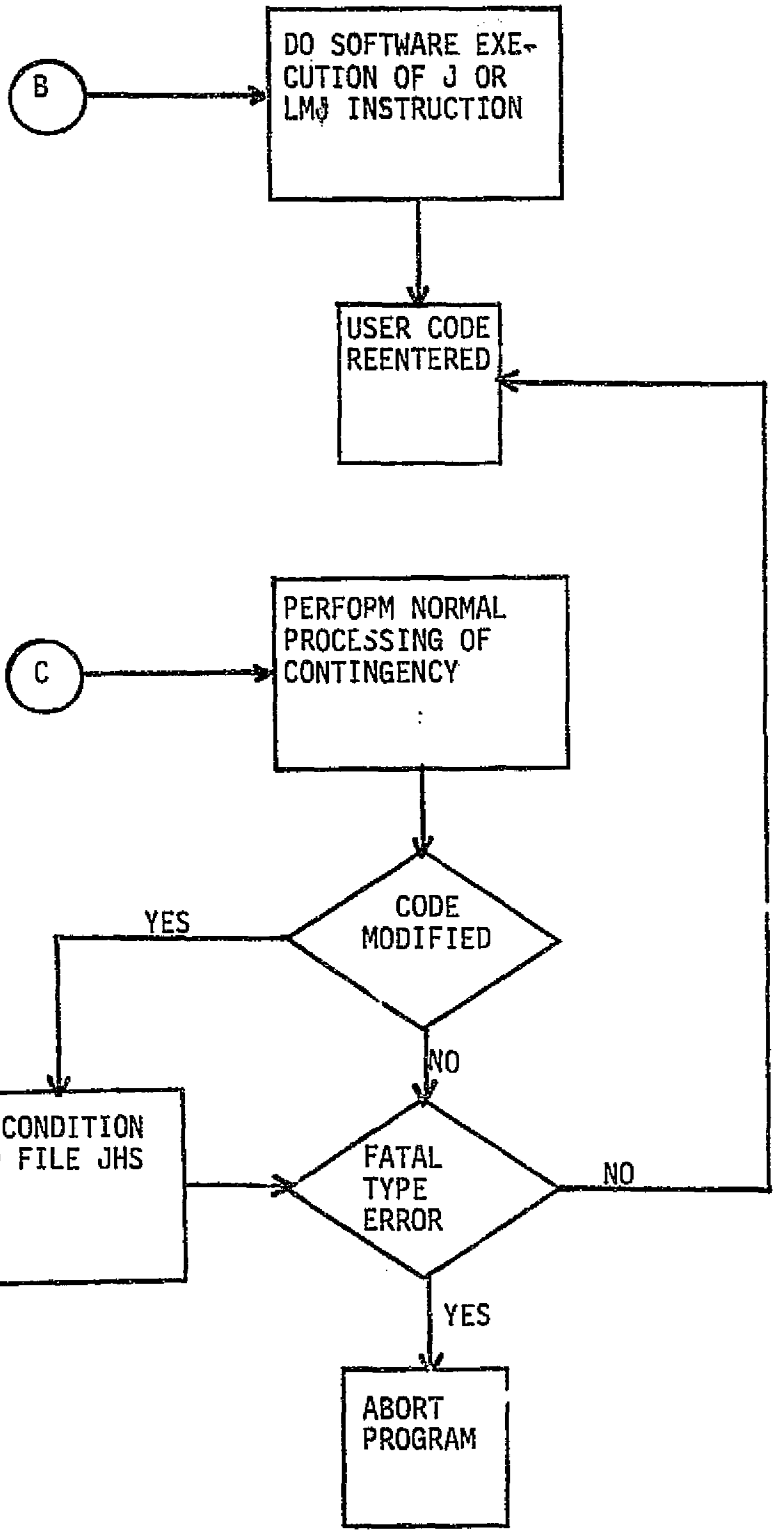
5.3 FUNCTIONAL FLOWCHART OF CONTINGENCY ROUTINE IICONT

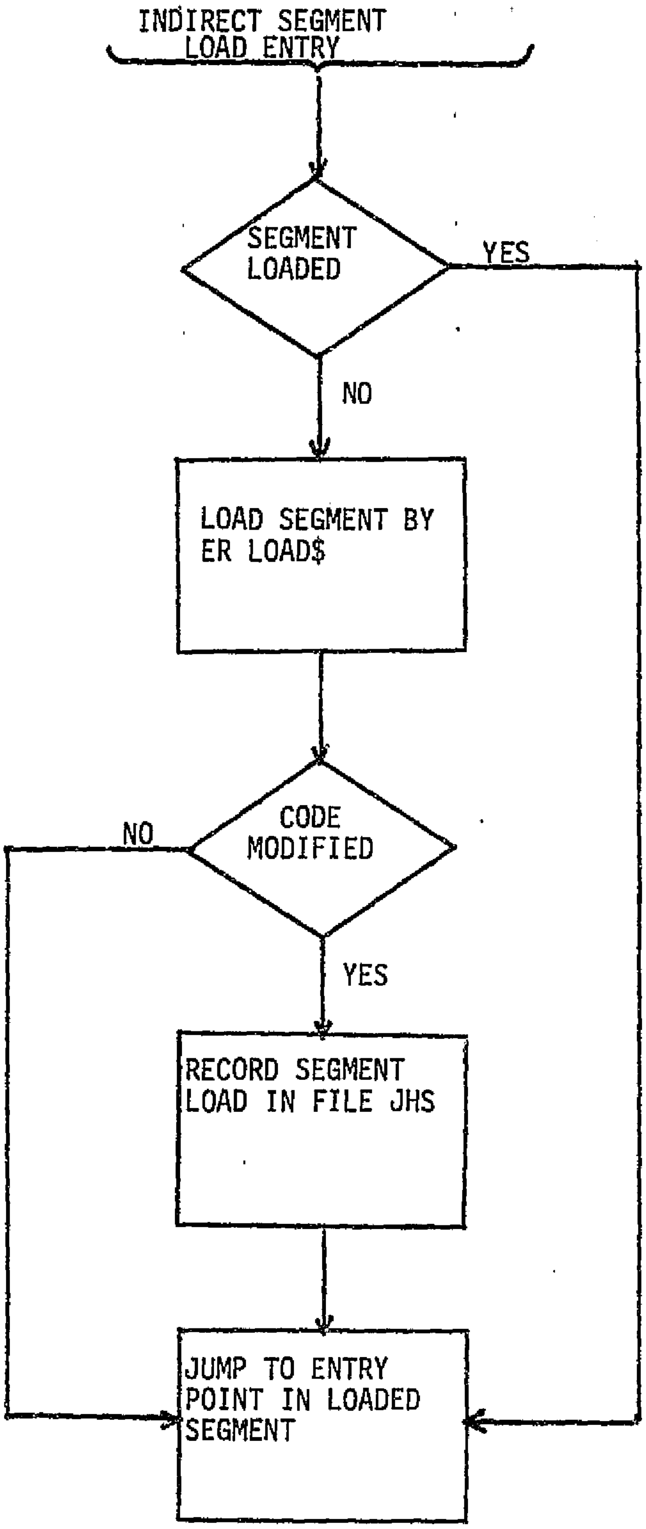

ENTRY NINTR$
REGISTER CONTINGENCY WITH EXEC 8
PROGRAM TO BE ANALYZED
NO
RETURN
YES
CONFIGURE TIMER ROUTINE AND INDIRECT LOAD ROUTINE FOR TRACING
START AND NUMBER ACTIVITY TO EXECUTE PROGRAM
TERMINATE INITIAL ACTIVITY
STOP
A1


A1
ACTIVITY 1 STARTS
START AND NUMBER ACTIVITY TO FLUSH JHS BUFFER
A2
ACTIVITY 2 STARTS
ACTIVITY 1 CONTINUES
INITIALIZE FILE JHS
RETURN
A2
ACTIVITY 2 STARTS
ER AWAIT$ WAIT ON ACTIVITY 1 TO FINISH
FLUSH JHS CORE BUFFER, SET NUMBER SECTORS IN FILE
STOP

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR


CONTINGENCY INTERRUPT ENTRY
ILLEGAL OPERATOR
NO
C
CODE MODIFIED
NO
C
YES
TIMING REPORT
YES
NO
GET EXECUTION TIMES
GET ADDRESS TO TRANSFER TO FROM ILLEGAL INSTRUCTION
RECORD DATA IN FILE JHS
B


B
DO SOFTWARE EXE-
CUTION OF J OR
LMJ INSTRUCTION
USER CODE
REENTERED
C
PERFOPM NORMAL
PROCESSING OF
CONTINGENCY
CODE
MODIFIED
YES
NO
ERROR CONDITION
OUT TO FILE JHS
FATAL
TYPE
ERROR
NO
YES
ABORT
PROGRAM


INDIRECT SEGMENT LOAD ENTRY
SEGMENT LOADED
YES
NO
LOAD SEGMENT BY ER LOAD$
CODE MODIFIED
NO
YES
RECORD SEGMENT LOAD IN FILE JHS
JUMP TO ENTRY POINT IN LOADED SEGMENT

5.4 SYMBOL DEFINITIONS

Table 5-I describes all variables used in labeled COMMON blocks within the CUAS preprocessor and postprocessor. BLANK COMMON is not used in either program, and the CUAS contingency routine IICONT uses no COMMON.

Table 5-II describes constants defined by DATA statements in the CUAS preprocessor main program (CUAPREPRO).

Table 5-III describes constants defined by DATA statements in the CUAS postprocessor main program (CUAPSTPRO).

TABLE 5-I.- VARIABLES IN LABELED COMMON

- COMMON BLOCK NAME: ACWCNT

DESCRIPTION: ACWCNT is used for the communication of code access channel words from the subroutine FCODE to the subroutine calling FCODE. ACWCNT is used only in the CUAS preprocessor.

LOCATION	NAME	DIMENSION	TYPE	DESCRIPTION
1-8	KACW	8	I	Cells 1 and 2 contain the base offset and length of the access channel words. Cells 3-8 are used to contain up to two channel word directives. This array is dynamically expanded in more locations as needed.

- COMMON BLOCK NAME: CONTRL

DESCRIPTION: CONTRL is used to retain general control information for the operation of the CUAS postprocessor. CONTRL is used only in the CUAS postprocessor.

LOCATION	NAME	DIMENSION	TYPE	DESCRIPTION
1	IPSTOP	1	I	The @XQT options, in master bit notation, from the postprocessor execution
2	IPREOP	1	I	The @XQT options, in master bit notation, from the preprocessor execution
3	IFWJHS	1	I	The number of the cell in the jump history stack file where the first even cell is located

- COMMON BLOCK NAME: JHSCNT

DESCRIPTION: JHSCNT contains information relating to the reading of the Code Usage Analysis System (CUAS) Jump History Stack File (JHSF). JHSCNT is used only in the CUAS postprocessor.

LOCATION	NAME	DIMENSION	TYPE	DESCRIPTION
1	JHS	1	I	The six-character fieldata internal file name used to reference the JHSF
2	NSIJHS	1	I	The length of the JHSF in 28-cell sectors
3	IBIMN	1	I	The number, relative 0, of the 112-cell block of the JHSF now contained in the I/O buffer array IBUF
4	LBIMN	1	I	The length, in cells, of the block of the JHSF now contained in the I/O buffer array IBUF
5-116	IBUF	112	I	An I/O buffer array for containing four contiguous sectors of the JHSF

- COMMON BLOCK NAME: TABLES

DESCRIPTION: TABLES is used to retain the location and length of the diagnostic tables from an absolute element, and is used in both the CUAS preprocessor and postprocessor. Primary storage for the diagnostic tables is dynamically allocated, and TABLES does not contain any of the data, but only pointers to the table locations in core and the length of the tables.

LOCATION	NAME	DIMENSION	TYPE	DESCRIPTION
1	IUN	1	I	The name of the file which contains absolute element diagnostic tables
2-11	IADE	10	I	The program file directory entry for the absolute element
12-39	IAHD	28	I	The first sector of the absolute element text
40-41	MSNT	2	I	The base offset and length of the segment name table
42-43	MENT	2	I	The base offset and length of the element name table
44-45	MBNT	2	I	The base offset and length of the bank name table
46-47	MSET	2	I	The base offset and length of the location counter table
48-49	MEPT	2	I	The base offset and length of the entry point table
50-51	MSLT	2	I	The base offset and length of the segment load table

TABLE 5-II.- DATA STATEMENT CONSTANTS IN CUAS PREPROCESSOR MAIN PROGRAM

PARAMETER NAME	VALUE	DESCRIPTION
IOPTA	$(00020000000)_8$	A bit mask used to extract the option A bit position from an option cell in master bit notation
IOPTB	$(00010000000)_8$	A bit mask used to extract the option B bit position from an option cell in master bit notation
IOPTC	$(000040000000)_8$	A bit mask used to extract the option C bit position from an option cell in master bit notation
LTRA	'A'	The alpha letter A
LTRB	'B'	The alpha letter B
LTRC	'C'	The alpha letter C
NINTR	'NINTR$'	The alpha letters NINTR$
LMASK	$(000000777777)_8$	A bit mask used to extract the lower 18 bits of a cell
IUN	'TPF$'	The alpha letters 'TPF$'
MSENT	$(666666666666)_8$	A sentinel used for insertion into the contingency subroutine IICONT

TABLE 5-III.- DATA STATEMENT DEFINED CONSTANTS IN CUAS POSTPROCESSOR MAIN PROGRAM

PARAMETER NAME	VALUE	DESCRIPTION
JHSASG	-	An array containing the character string '@ASG,A JHS.'
IFRMSK	$(004112302700)_8$	A bit mask for checking a facility status cell
LTRS(1)	'NO'	The alpha letters NO
LTRS(2)	'C'	The alpha letter C
LTRS(3)	'B'	The alpha letter B
LTRS(4)	'A'	The alpha letter A
IOPTW	$(000000000010)_8$	A bit mask used to extract the option W bit position from an option cell in master bit notation
IOPTG	$(000002000000)_8$	A bit mask used to extract the option G bit position from an option cell in master bit notation
IGOFF	$(777775777777)_8$	A bit mask used to clear the option G bit position from an option cell in master bit notation
IOPTX	$(000000000004)_8$	A bit mask used to extract the option X bit position from an option cell in master bit notation
IOPTE	$(000010000000)_8$	A bit mask used to extract the option E bit position from an option cell in master bit notation

PARAMETER NAME	VALUE	DESCRIPTION
IOPTS	$(000000000200)_8$	A bit mask used to extract the option S bit position from an option cell in master bit notation
IOPTB	$(000100000000)_8$	A bit mask used to extract the option B bit position from an option cell in master bit notation
IFWSJHS	8	The integer value 8
IUN	'TPF$'	The alpha letters TPF$
JHS	'JHS'	The alpha letters JHS
NSIJHS	0	The integer value 0

5.5 SUBROUTINE DOCUMENTATION

Individual documentation of the non-UNIVAC EXEC 8 supplied subroutines used in the CUAS appears in alphabetical order on the following pages.

SUBROUTINE BD2FD

IDENTIFICATION

Name (Title)	- BD2FD (Binary Data to Fieldata)
Programmer, Date	- P. H. Horsley, March 1976
Machine Identification	- UNIVAC 1100 - Series
Source Language	- FORTRAN V

PURPOSE

Subroutine BD2FD converts the data and time entry from an EXEC 8 program file directory item to a four-cell string of fieldata characters.

USAGE

- Calling Sequence

  CALL BD2FD(IBDAT, IFDDAT)

  Arguments:

Parameter name	In/Out	Dimension	Type	Description
IBDAT	In	1	I	Time in seconds past midnight in bits 18-35, month in bits 12-17, day in bits 6-11, year MOD(64) in bits 0-5
IFDDAT	Out	4	I	Four-cell array containing 'MM/DD/YYbbbb HH:MM:SSbbbb' where MM=month, DD=day, YY=year, HH=hour, MM=minutes, SS=seconds, b=fieldata blank

METHOD

- Model

The binary values of the date and time are separated from the input parameter IBDAT, and a fieldata character string of the corresponding numbers is constructed in the output parameter array IFDDAT.

RESTRICTIONS

- Operational

Subroutines ICLSFT and ILLSFT are required.

SUBROUTINE BUBSRT

IDENTIFICATION

Name (Title)	- BUBSRT (Bubble Sort)
Programmer, Date	- P. H. Horsley, September 1975
Machine Identification	- UNIVAC 1100 - Series
Source Language	- FORTRAN V

PURPOSE

Subroutine BUBSRT sorts a numeric integer array into ascending sequence and optionally reorders two other arrays in the same sequence as the sorted array.

USAGE

- Calling Sequence

CALL BUBSRT (IA, IP, NC, L, N)

Arguments:

Parameter name	In/Out	Dimension	Type	Description
IA	In	Variable	I	The integer array to sort
IP	In	Variable	I	The first array to be reordered same as IA. See parameter N
NC	In	Variable	I	The second array to be reordered same as IA. See parameter N
L	In	1	I	The length of the array lA. If IP and/or NC are used, then their lengths are assumed the same as IA
N	In	1	I	If 1, do not reorder IP or NC If 2, reorder IP only If 3, recrder both IP and NC
IA	Out	Variable	I	The sorted array
IP	Cut	Variable	I	Reordered array
NC	Out	Variable	I	Reordered array

METHOD

- Model

Subroutine BUBSRT uses a bubble sort technique to reorder the integer array IA into ascending sequence numerically. The arrays IP and NC are not sorted but strictly reordered in the same sequence as the array IA. For example, if the sort of the array IA requires swapping the values in position i and j, the values in position i and j of the arrays IP and NC are also swapped. If the reordering of either IP, NC, or both is not desired, these arguments in the calling sequence need not be arrays but may be merely undimensioned dummy arguments.

SUBROUTINE CKDOLR

IDENTIFICATION

Name (Title)	- CKDOLR (Check Dollar)
Programmer, Date	- P. H. Horsley, April 1976
Machine Identification	- UNIVAC 1100 - Series
Source Language	- FORTRAN V

PURPOSE

Subro tine CKDOLR determines if a fieldata $ character is included in a two-cell array of fieldata characters.

USAGE

- Calling Sequence

  CALL CKDOLR (NAME, IRET)

  Arguments:

Parameter name	In/Out	Dimension	Type	Description
NAME	In	2	I	The array of fieldata characters to check for a $ character
IRET	Out	1	I	0 if no $ character found; if 1, array contains at least one $ character

METHOD

- Model

  The two-cell array of characters is inspected from left to right, with the scan terminating either when a $ character is found or when all characters have been inspected.

RES;RICTIONS

- Operational

  Subroutine IRRSFT is required.

SUBROUTINE CLSEOR

IDENTIFICATION

Name (Title)	- CLSEOR (Close Output Routine)
Programmer, Date	- D. M. Braley, May 1973
Author, Date	- P. H. Horsley, June 1976
Machine Identification	- UNIVAC 1100-Series
Source Language	- UNIVAC 1100-Series Assembler

PURPOSE

Subroutine CLSEOR closes a symbolic element in an EXEC 8 program file which was opened with subroutine OPNEOR.

USAGE

- Calling Sequence

  CALL CLSEOR ($LAB)

  Arguments:

Parameter name	In/Out	Dimension	Type	Description
$LAB	-	-	-	The label number to return to if element is not successfully closed

METHOD

- Model

  The UNIVAC-supplied package SOR is used to close the source element. SOR is documented in U.P. 4144, Rev. 3.

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

RESTRICTIONS

- Operational

Subroutine OPNEOR must be called to open the source element, and usually subroutine OUTIMG will be called to insert source lines into the element before subroutine CLSEOR is called to close the element.

SUBROUTINE CSFER

IDENTIFICATION

NAME (Title)	- CSFER (CSF EXEC Request)
Programmer, Date	- P. H. Horsley, May 1976
Machine Identification	- UNIVAC 1100-Series
Source Language	- UNIVAC 1100-Series Assembler

PURPOSE

Subroutine CSFER performs an executive request to the EXEC 8 entry point CSF$.

USAGE

- Calling Sequence

  CALL CSFER (ICC, ISTAT)

  Arguments:

Parameter name	In/Out	Dimension	Type	Description
ICC	In	Variable	I	Array containing control card image for CSF$ formatted as described in U.P. 4144, Rev. 3, pp. 4-41, 43
ISTAT	Out	1	I	Status from register A0 following completion of request

METHOD

- Model

  Subroutine CSFER does an executive request to the EXEC 8 entry point CSF$. The address of the parameter array ICC is supplied as the control card image address, and the content of the register A0 is stored in parameter ISTAT upon return from the request.

SUBROUTINE DISKIO

IDENTIFICATION

Name (Title)	- DISKIO (Disk Input Output)
Programmer, Date	- P. H. Horsley, September 1975
Machine Identification	- UNIVAC 1100 - Series
Source Language	- UNIVAC 1100 - Series Assembler

PURPOSE

Subroutine DISKIO will either read or write FASTRAND formatted secondary storage on UNIVAC 1100 EXEC 8 operating systems.

USAGE

- Calling Sequence

  CALL DISKIO (IUN, IOP, ISS, INS, MRAY, ISTAT)

  Arguments:

Parameter name	In/Out	Dimension	Type	Description
IUN	In	1	I	A six-character fieldata name of the file name to read/write. The name is expanded to 12 characters by subroutine DISKIO by adding six blanks.
IOP	In	1	I	A 1 for write operation or 2 for read operation.
ISS	In	1	I	The sector number relative to 0 at which to start the read or write
INS	In	1	I	The number of 28 cell sectors to read/write starting at sector ISS
MRAY	In/Out	28*INS	I	If IOP = 1, the primary storage array is transferred to secondary storage. If IOP = 2, the primary storage array is filled from secondary storage. The length of this array must be defined

PRECEDING PAGE BLANK NOT FIL

Parameter name	In/Out	Dimension	Type	Description
				by the caller and must be adequate to contain the number of 28 cell sectors to be transferred.
ISTAT	Out	1	I	The postive actual number of cells transferred if the operation completes normally, a negative number, same as I/O error messages from UNIVAC Publication 4144, Rev. 3, Appendix C, if the I/O was not completed normally. A positive number returned but less than 28*INS indicates end of allocated mass storage was reached during the transfer and only the indicated number of cells was transferred.

- Error Messages

  None - errors indicated only by value of the parameter ISTAT upon return to caller. ISTAT will be set to -100 if the value of the parameter IOP is neither 1 or 2.

METHOD

- Model

  Subroutine DISKIO creates an I/O packet for the read or write request and then performs the actual I/O operation with an Executive Request (ER) to the EXEC 8 entry point IOW$. The variable ISTAT is set directly from the I/O packet status cell before returning to the caller.

  Subroutine DISKIO saves and restores all registers used internally.

- Reference

  UNIVAC Publication 4144, Revision 3, Section 6.3.5, pages 6-11 and Appendix C.

SUBROUTINE ELRPT

IDENTIFICATION

Name (Title)	- ELRPT (Error Location Report)
Programmer, Date	- P. H. Horsley, December 1975
Machine Identification	- UNIVAC 1100 - Series
Source Language	- FORTRAN V

PURPOSE

Subroutine ELRPT produces the CUAS error location report and is intended for use only within the CUAS postprocessor.

USAGE

- Calling Sequence

  CALL ELRPT (IERR, ISTAT)

  Arguments:

Parameter name	In/Out	Dimension	Type	Description
IERR	Out	1	I	An error indicator, set nonzero to indicate an error and type
ISTAT	Out	1	I	A status flag associated with the error type

- Labeled COMMON

  All of the labeled COMMON blocks TABLES and CONTRL are required (refer to labeled COMMON block description in table 5-I. The data in both COMMON blocks are used strictly as reference data and no output from the subroutine occurs through the COMMON blocks.

- Error Messages

  None - all errors are indicated by a nonzero value of the parameter IERR. IERR=901 indicates an I/O error has occurred while attempting to read file

JHS, and IERR=902 indicates that the required dynamic core could not be allocated. In the second case, parameter ISTAT contains the amount of core needed.

METHOD

- Model

Subroutine ELRPT scans the CUAS JHS to locate error event cells entered in the file by the CUAS contingency routine IICONT. The content and format of the JHS is presented in Appendix D of the CUAS program documentation. The type of the error is determined from the upper 18 bits of an error event cell and the address at which it occurred is determined from the lower 18 bits of the cell. The element within which the address is located is determined by inspection of the location counter table in COMMON block TABLES. As the JHS is scanned, segment load event cells are used to maintain which segments of an overlaid program are loaded. This is done by calling subroutine SLTSET once for each segment load event cell encountered. The segment load table thus maintained is then used to resolve address ambiguities should any be enountered.

A walk back to the main routine is provided for each error located if the trace has been requested and then only depending on the CUAS preprocessor options. The preprocessor options are determined by inspection of cell 2 of COMMON block CONTRL. If either option A or B was specified to the CUAS preprocessor, the walk back is not performed in any case. If neither option A or B was specified to the CUAS preprocessor, cell 1 of COMMON block CONTRL is inspected to determine if the user desires the walk back report. This cell reflects the options specified to the CUAS postprocessor.

The walk back is performed by proceeding backwards in the JHS from the point where an error event cell was encountered. Subroutine call event cells are inspected to determine the calling sequence to the element within which the error occurred. During this process, the external name table and element name table from COMMON block TABLES are used to correlate addresses to names.

Subroutine ELRPT dynamically allocates the temporary storage needed in order to produce the error location report, and then dynamically releases this core once the report is finished. This is accomplished by using the subroutine GCORE and RCORE. The amount of dynamic core allocated is dependent solely on the size of the diagnostic tables from the absolute element for the program under analysis.

RESTRICTIONS

- Operational

  Subroutines SLTCLR, SLTSET, GCORE, and RCORE and functions ILLSFT, IRRSFT, and NGET are required.

SUBROUTINE FASLT

IDENTIFICATION

Name (Title)	- FASLT (Fetch Absolute Segment Load Table)
Programmer, Date	- P. H. Horsley, May 1976
Machine Identification	- UNIVAC 1100-Series
Source Language	- FORTRAN V

PURPOSE

Subroutine FASLT fetches the segment load table from an absolute element in an EXEC 8 program file.

USAGE

- Calling Sequence

  CALL FASLT (IERR, ISTAT)

  Arguments:

Parameter name	In/Out	Dimension	Type	Description
IERR	Out	1	I	Error condition indicator
ISTAT	Out	1	I	Value associated with error condition

- Labeled COMMON

  All of the labeled COMMON block TABLES are required (refer to labeled COMMON block description in table 5-I). The core to contain the segment load table is dynamically allocated with the location and length of the table filled into array MSLT in labeled COMMON block TABLES.

- Error Messages

  None - all errors are indicated by a nonzero value of the parameter IERR. IERR=801 if the dynamic core needed to load the segment load table cannot

PRECEDING PAGE BLANK NOT FI..

be allocated, in which case the parameter ISTAT contains the amount of core needed. IERR=802 if the segment load table could not be located in the absolute element.

METHOD

- Model

The segment load table is a part of an absolute element as prepared by the UNIVAC MAP under EXEC 8. The content and format of an absolute element is described in Appendix C. The segment load table is extracted from the control bank of the absolute element and read into a block of core which is dynamically allocated. The program file and absolute element are determined from the data in labeled COMMON block TABLES.

RESTRICTIONS

- Operational

Subroutines DISKIO and GCORE and function IRRSET are required.

SUBROUTINE FCODE

IDENTIFICATION

Name (Title)	- FCODE (Fetch Code)
Programmer, Date	- P. H. Horsley, August 1975
Machine Identification	- UNIVAC 1100-Series
Source Language	- FORTRAN V

PURPOSE

Subroutine FCODE locates the start of the I-BANK code of an element within an absolute element residing in an EXEC 8 program file.

USAGE

- Calling Sequence
  CALL FCODE (IWP, IES, IEL, IERR, ISTAT)
  Arguments:

Parameter name	In/Out	Dimension	Type	Description
IWP	In	1	I	The word number, relative 0, of the segment name table entry describing the segment which contains the element
IES	In	1	I	The primary storage address of the first location in the element's I-BANK
IEL	In	1	I	The primary storage length of the element's I-BANK
IERR	Out	1	I	An error condition indicator
ISTAT	Out	1	I	Value associated with error indicator IERR

- Labeled COMMON

All of the labeled COMMON blocks TABLES and ACWCNT are required (refer to the labeled COMMON block description in table 5-I). The data in COMMON block TABLES is used strictly as reference data and no output of data occurs through the COMMON block. The array KACW in COMMON block ACWCNT is used to return to the calling routine the location of the element's I-BANK code.

- Error Messages

None - all errors are indicated by a nonzero value of the parameter IERR. IERR=401 if the dynamic core needed to expand the array KACW is not available, and the parameter ISTAT contains the amount of core needed.

METHOD

- Model

The parameter IWP is used to retrieve the segment data from the segment name in table COMMON block TABLES, describing the segment which contains the desired element's I-BANK code. The access channel words (ACW's) for that segment are inspected to determine the disk location of the element's I-BANK code. Disk access directives are returned in COMMON block ACWCNT which will enable the calling routine to retrieve the actual code for the element.

RESTRICTIONS

- Operational

Subroutines DISKIO, GCORE, and function IRRSFT are required.

SUBROUTINE FNDELT

IDENTIFICATION

Name (Title)	- FNDELT (Find Element)
Programmer, Date	- P. H. Horsley, April 1976
Machine Identification	- UNIVAC 1100-Series
Source Language	- FORTRAN V

PURPOSE

Subroutine FNDELT locates (within the location counter table from an absolute element) the entry describing the address of the program element.

USAGE

- Calling Sequence

  CALL FNDELT (IADR, ISP, ISET, LENSET, IRET)

  Arguments:

Parameter name	In/Out	Dimension	Type	Description
IADR	In	1	I	The address for which the location counter table entry is to be found
ISP	In	1	I	Segment name table index of segment within which address is located
ISET	In	Variable	I	Location counter table array
LENSET	In	1	I	Length of location counter table
IRET	Out	1	I	Index of entry in location counter table describing element within which address is contained

- Error Messages

  None - the parameter IRET is set to -1 if the location counter table entry could not be found.

METHOD

- Model

  The location counter table (LCT) is inspected with a linear search for an entry (within the specified segment) which contains the specified address.

RESTRICTIONS

- Operational

  The function IRRSFT is required.

SUBROUTINE GCORE

IDENTIFICATION

Name (Title)	- GCORE (Get Core)
	- RCORE (Release Core)
	- LCORE (Limits of Core)
Programmer, Date	- P. H. Horsley, March 1976
Machine Identification	- UNIVAC 1100-Series
Source Language	- UNIVAC 1100-Series Assembler

PURPOSE

The purpose of each of the three entry points follows:

Entry point GCORE dynamically allocates additional primary storage at the end of a program's D-BANK, or extends the length of a block of primary storage previously allocated by subroutine GCORE.

Entry point RCORE releases blocks of core which were dynamically allocated by GCORE.

Entry point LCORE supplies information as to the core in use and the mode of the job executing the program, either demand or batch.

USAGE

- Calling Sequence
  CALL GCORE (ITYP, LEN, A, LOC, $LAB)
  Arguments:

Parameter name	In/Out	Dimension	Type	
ITYP	In	1	I	Set to 1 to request a new block of core be allocated, set to 0 to request extension of last block allocated
LEN	In	1	I	The length, in cells, of the core to be allocated
A	In	Variable	I	The array name to be used as the base address for referencing the dynamic core
LOC	Out	1	I	The offset used from array A to reference the dynamic core. A(1+LOC) will reference the first cell and A(LEN+LOC) will reference the last cell. If parameter ITYP=0, LOC is not set on return
$LAB	In	-	-	The FORTRAN statement label number to return to if an error occurs

CALL RCORE(N)

Arguments:

Parameter name	In/Out	Dimension	Type	Description
N	In	1	I	The number of dynamically allocated core blocks to release. The blocks released are the last N blocks allocated by GCORE. If N is a negative number, all dynamic core blocks are released.

CALL LCORE (INFO)

Arguments:

Parameter name	In/Out	Dimension	Type	Description
INFO	Out	5	I	INFO(1)=Job type, 4 for demand, 5 for deadline batch, 6 for normal batch INFO(2)=Total core in use by program at time of call INFO(3)=Maximum core which GCORE will allocate for caller. A demand job is limited to 20K, a batch job 65K. INFO(4)=The number of blocks which have been allocated by a type 1 call to GCORE. INFO(5)=The maximum number of blocks which GCORE will allocate

- Error Messages

None - error conditions are detected only by entry point GCORE, and are indicated by returning a negative value of the parameter LOC and returning to the statement number indicated by the parameter $LAB rather than a normal return. The error condition is indicated by the value of the parameter LOC, which may take on values as follow:

-1 = The length of the requested block would exceed maximum core restrictions.

-2 = The maximum number of blocks which GCORE can handle are currently allocated. As of this writing, the maximum number is 20.

-3 = An add core to last block request was made, but no blocks have been allocated by GCORE.

-4 = The parameter LEN is negative, no core was allocated.

METHOD

- Model

EXEC 8 allocates core to a user program in pages of 512 cells each, with the programs I-BANK and D-BANK in disjoint sets of pages. The GCORE subroutine extends a program's D-BANK only by allocating additional pages of core at the end of the static D-BANK pages. Subroutine GCORE controls the dynamic core by means of logical blocks which are of variable lengths as specified by the calling routine. Once a block is allocated, it may be extended only if it is the last block allocated by subroutine GCORE. A nested block may be made the last block allocated by releasing all blocks which were allocated after it.

New pages are allocated or released dynamically by means of the EXEC 8 entry points MCORE$ and LCORE$, respectively. New pages are allocated only if a block will not fit within the unused core in the last page in use by the program.

RESTRICTIONS

- Operational

This subroutine may not be used to exceed the core limitation requirements in force at the NASA/JSC 1100/EXEC 8 computing facility. A request for more core than allowed a demand job will result in an error return from subroutine GCORE and no additional core will be allocated. Subroutine GCORE is designed to aid the user in installing demand programs and conserving storage resources through dynamically allocating only that storage required and/or allocating and releasing temporary storage arrays as needed, so that temporary storage may share a common address space.

SUBROUTINE GETOPT

IDENTIFICATION

Name (Title) - GETOPT (Get Options)

Programmer, Date - P. H. Horsley, December 1975

Machine Identification - UNIVAC 1100 - Series

Source Language - UNIVAC 1100 - Series Assembler

PURPOSE

Subroutine GETOPT returns the @XQT options specified when the program calling subroutine GETOPT was executed.

USAGE

- Calling Sequence

  CALL GETOPT (IOPT)

  Argument:

Parameter name	In/Out	Dimension	Type	Description
IOPT	Out	1	I	The options in master bit notation

METHOD

- Model

  Subroutine GETOPT does an executive request to the EXEC 8 entry point OPT$. EXEC 8 returns the option word in register A0, which is then stored in parameter IOPT. Within IOPT, the options are specified in master bit notation, that is, bit 25 is set on if option A was present and bit 0 is set on if option Z was present.

SUBROUTINE IICONT

IDENTIFICATION

Name (Title)	- IICONT (Contingency Routine)
Programmer, Date	- P. H. Horsley, April 1976
Machine Identification	- UNIVAC 1100 - Series
Source Language	- UNIVAC 1100 - Series Assembler

PURPOSE

Subroutine IICONT registers a contingency handler routine with EXEC 8 and traps program contingencies when they occur during the execution of a program. Subroutine IICONT is intended for use with the CUAS for creating a jump history stack file of a FORTRAN application program execution. This subroutine contains no FORTRAN callable entry points.

USAGE

- Calling Sequence (Assembly Language)

```
LMJ   X11,NINTR$
+     0
```

Arguments:

One cell must follow the LMJ instruction as the return jump is to X11 + 1.

METHOD

- Model

Every UNIVAC EXEC 8 FORTRAN main program includes a reference to the external NINTR$ as the first executable instruction of the program. The purpose of the subroutine containing the external name NINTR$ is to register a contingency handler routine with EXEC 8. A contingency routine is described in UNIVAC Publication 4144, Revision 3, Section 4.9.

A standard contingency handler routine is provided in the UNIVAC FORTRAN library. Subroutine IICONT performs the same functions as the standard,

PRECEDING PAGE BLANK NOT FI

but has the expanded capability to create a JHS of a FORTRAN program execution. The JHS content and format as created by subroutine IICONT is described in Appendix D. The tracing of program errors is done directly by trapping the errors in the normal way and then inserting this information into the JHS. The tracing of subroutine calls and jumps is done by interpreting an illegal operator (IOPR) interrupt as an expected event which signals that control is to be transferred to or from a subroutine. The transfer is performed by a software execution of the instruction, and the fact that the transfer was made is inserted into the JHS.

Subroutine IICONT also contains the code for performing segment loads when segment loading is by the indirect method. This code has the expanded capability to record into the JHS the fact that the segment was loaded.

RESTRICTIONS

- Operational

Ten cells are reserved immediately preceding the entry cell NINTR$. These 10 cells are used as a communication area between the CUAS preprocessor and subroutine IICONT. Any change in the length, location, or format of these 10 cells will require compatible changes be done in the CUAS preprocessor. The 10 cells contain the following information:

Cell	Description
1-2	The fieldata name of the absolute element within which IICONT is contained
3-4	The fieldata version name of the absolute element within which IICONT is contained
5	The time and date of creation of the absolute element
6	A sentinel cell set to the octal value (666666666666) by the CUAS preprocessor

Cell	Description
7	The start address of the indirect load table in this absolute element
8	The last address of the indirect load table in this absolute element
9	The CUAS preprocessor options in master bit notation
10	A sentinel cell set to the octal value (666666666666)

When the CUAS preprocessor is executed, it examines cell 10 of the area and if it does not contain the specified sentinel, the preprocessor assumes that the incorrect version of NINTR$ has been included in the absolute. The first nine cells of the area are filled in by the preprocessor as part of its task of modifying the absolute code. Cell 6 of the area is used by subroutine IICONT to determine if the absolute element has been modified by the preprocessor. If the specified sentinel is not found in cell 6, it is assumed that the absolute has not been modified. If the code has been modified, cells 1-5 and cell 9 of the area are passed on the the CUAS postprocessor by inserting them into the jump history stack file.

Any change of the location, order, format, or length of this 10-cell area without compatible changes in the CUAS preprocessor will result in improper operation of the entire CUAS.

The timing subroutine contained within subroutine IICONT is designed for operation on the EXEC 8 operating system in use at NASA/JSC as of this writing. The formulas used by this routine in computing timing charges may be found in Appendix A. The technique for determining time is, in general, EXEC 8 installation dependent.

FUNCTION ILLSFT

IDENTIFICATION

Name (Title)	- ILLSFT (Logical Left Shift) - IRRSFT (Logical Right Shift) - ICLSFT (Circular Left Shift) - ICRSFT (Circular Right Shift)
Programmer, Date	- P. H. Horsley, August 1975
Machine Identification	- UNIVAC 1100 - Series
Source Language	- UNIVAC 1100 - Series Assembler

PURPOSE

The purpose of each of the four entry points follows:

Entry point ILLSFT performs a logical left shift on a single cell. Bit positions vacated are zero filled, bits shifted out are lost.

Entry point IRRSFT performs a logical right shift on a single cell. Bit positions vacated are zero filled, bits shifted out are lost.

Entry point ICLSFT performs a circular left shift on a single cell. Bits shifted out of the upper left position are reentered at the lower right.

Entry point ICRSFT performs a circular right shift on a single cell. Bits shifted out of the lower right position are reentered at the upper left.

USAGE

- Calling Sequence

  I = ILLSFT (IVAR, NUB)

  Arguments:

Parameter name	In/Out	Dimension	Type	Description
IVAR	In	1	I	The cell containing the bits to shift left

Parameter name	In/Out	Dimension	Type	Description
NUB	In	1	I	The number of bit positions to shift left in IVAR
I	Out	1	I	The shifted cell as the function value. IVAR is unchanged

I = IRRSFT (IVAR, NUB)

Arguments:

Parameter name	In/Out	Dimension	Type	Description
IVAR	In	1	I	The cell containing the bits to shift right
NUB	In	1	I	The number of bit positions to shift right in IVAR
I	Out	1	I	The shifted cell as the function value. IVAR is unchanged

I = ICLSFT (IVAR, NUB)

Arguments:

Parameter name	In/Out	Dimension	Type	Description
IVAR	In	1	I	The cell containing the bits to circular shift left
NUB	In	1	I	The number of bit positions to shift left in IVAR
I	Out	1	I	The shifted cell as the function value. IVAR is unchanged

I = ICRSFT (IVAR, NUB)

Arguments:

Parameter name	In/Out	Dimension	Type	Description
IVAR	In	1	I/R	The cell containing the bits to circular shift right

Parameter name	In/Out	Dimension	Type	Description
NUB	In	1	I	The number of bit positions to shift right in IVAR
I	Out	1	I	The shifted cell as the function value. IVAR is unchanged.

- Error Messages

  None - for all four entry points, the value of the parameter NUB is checked and if it is less than or equal to zero, or greater than 36, no shift is performed and a return to the caller is done.

METHOD

- Model

  All four entry points insert the value of the parameter NUB into the appropriate machine instruction and then that instruction is executed to perform the shift. The shifted value is in register A0 when the subroutine returns to the caller. The calling sequence to all four entry points conform to FORTRAN conventions in that they are referenced by an LMJ instruction using X11, and the return point allows for the FORTRAN walk back word. Subroutine SHIFTY uses only registers A0 and X11.

SUBROUTINE LADTAB

IDENTIFICATION

Name (Title) - LADTAB (Load Absolute Tables)
Programmer, Date - P. H. Horsley, March 1976
Machine Identification - UNIVAC 1100-Series
Source Language - FORTRAN V

PURPOSE

Subroutine LADTAB allocates primary storage for the diagnostic tables and then reads them into that area from the last absolute element inserted into a program file.

USAGE

- Calling Sequence

  CALL LADTAB (IERR, ISTAT)

  Arguments:

Parameter name	In/Out	Dimension	Type	Description
IERR	Out	1	I	An error condition indicator
ISTAT	Out	1	I	A status value associated with the error condition

- Labeled COMMON

  All of COMMON block TABLES is required (refer to the labeled COMMON description in table 5-I). The size of the COMMON block is dynamically expanded to exactly contain the tables from an absolute element.

- Error Messages

None - all error conditions are indicated by a nonzero value of the parameter IERR. IERR=101 if the core for the tables could not be allocated, in which case parameter ISTAT contains the amount of core needed. IERR=102 if the file referenced is not a program file. IERR=103 if no absolute element exists in the referenced file. IERR=104 if the absolute element does not contain diagnostic tables.

METHOD

- Model

The sequence number of the last absolute element inserted into the program file is determined from the program file header table. The sector address and size of the diagnostic tables is determined from the header table for the absolute element. Core is expanded to hold the tables by calling subroutine GCORE and then the tables are read into that area by calling subroutine DISKIO.

RESTRICTIONS

- Operational

Subroutines GCORE, RCORE, and DISKIO, and function IRRSFT are required.

SUBROUTINE MINT

IDENTIFICATION

Name (Title)	- MINT (Modify Instruction)
Programmer, Date	- P. H. Horsley, April 1976
Machine Identification	- UNIVAC 1100-Series
Source Language	- FORTRAN V

PURPOSE

Subroutine MINT scans the I-BANK code of an absolute element and modifies the operation code of LMJ and J instructions to illegal operation codes.

USAGE

- Calling Sequence

  CALL MINT (IOPTWD, NIWD, IERR, ISTAT)

  Arguments:

Parameter name	In/Out	Dimension	Type	Description
IOPTWD	In	1	I	The @XQT options in master bit notation
NIWD	Out	1	I	The index of the third cell in the external name table for the name NINTR$
IERR	Out	1	I	An error condition indicator
ISTAT	Out	1	I	A status value associated with IERR

- Labeled COMMON

  All of the labeled COMMON blocks TABLES and ACWCNT are required (refer to the labeled COMMON block description in table 5-I). The COMMON blocks are used strictly as reference data, and no output of data occurs through the COMMON blocks.

- Error Messages

  None - all error conditions are indicated with a nonzero value of the parameter IERR. IERR=201 if dynamic core cannot be allocated for local working arrays, in which case parameter ISTAT contains the amount of core needed.

METHOD

- Model

  The I-BANK code for each user-supplied element of the absolute is scanned for LMJ instructions to an external reference and J instructions which will transfer control outside of the element being scanned. These instructions are then modified by inserting into the operation field an octal 7700 for an LMJ and an octal 7701 for a J instruction. Any element which was obtained from the SYS$*RLIB$. file or any element which contains a $ character within its name is not scanned and modified. Similarly, any LMJ instruction which will transfer control to an address in such an element is not modified. The element IICONT is not modified and any LMJ to the entry point NINTR$ is not changed to an illegal operator. A J instruction is always modified if an index register is used in forming the jump to address. When no index register is used, the instruction is modified only if the jump to address is outside the range of the element containing the J instruction, or if the address is indirect.

RESTRICTIONS

- Operational

  Subroutines GCORE, RCORE, CKDOLR, FNDELT, and FCODE and functions IRRSFT, ICLSFT, and ICRSFT are required.

FUNCTION NGET

IDENTIFICATION

Name (Title) - NGET (Next Get)

Programmer, Date - P. H. Horsley, December 1975

Machine Identification - UNIVAC 1100 - Series

Source Language - FORTRAN V

PURPOSE

Function NGET returns to the caller a single cell of the Code Usage Analysis System (CUAS) Jump History Stack File (JHSF) as the value of an integer function. The desired cell number is furnished to the function in a calling argument.

USAGE

- Calling Sequence

  I = NGET (NB, $LABEL)

  Arguments:

Parameter name	In/Out	Dimension	Type	Description
NB	In	1	I	The number of the desired cell from the JHSF
$LABEL	In	1	I	The label of the statement returned to if the requested cell is outside the JHSF range, that is, an end-of-file condition
NGET	Out	1	I	The content of the requested cell from the JHSF

- Labeled COMMON

  All of COMMON block JHSCNT is required (refer to the labeled COMMON description in table 5-I). COMMON block JHSCNT is used strictly for reference and no OUTPUT of data occurs through the COMMON block.

- Error Messages

  None - if the caller attempts to reference a cell outside the JHSF limits, a nonstandard return through parameter 2 is performed.

METHOD

- Model

  When function NGET is called, the variable NSIJHS is examined and if zero, NGET assumes this is the initial call to the routine. The first four sectors of the JHSF are retrieved and the variables NSIJHS, IBIMN, and LBIMN are initialized. Once these variables are initialized, the first call and succeeding calls determine the block number within which the requested cell, in parameter NB, is contained. A block is defined as four 28-cell sectors as 112 total cells. If the necessary block is currently in the I/O buffer array IBUF, as indicated by the variable IBIMN, the requested cell is retrieved and returned as the value of the function. If the necessary block is not in the I/O buffer array IBUF, the block is retrieved from the JHSF and the variables IBIMN, LBIMN are updated. The variable LBIMN reflects the actual length of each I/O transfer which could be fewer than four sectors if a request exceeds the length of the JHSF but starts within it.

  The first four cells of COMMON block JHSCNT must be initialized prior to the first call to function NGET.

RESTRICTIONS

- Operational

  The subroutine DISKIO is required. The named COMMON block JHSCNT is considered the sole property of this function and no other subroutines of a program which use the function should alter the content of the COMMON block in any way except to set initial values in it prior to the first call to function NGET. Failure to adhere to this restriction will result in unpredictable operation of function NGET.

SUBROUTINE OPNEOR

IDENTIFICATION

Name (Title)	- OPNEOR (Open Output Routine)
Programmer, Date	- D. M. Braley, May 1973
Author, Date	- P. H. Horsley, June 1976
Machine Identification	- UNIVAC 1100-Series
Source Language	- UNIVAC 1100-Series Assembler

PURPOSE

Subroutine OPNEOR oepns a symbolic element for line image output into an EXEC 8 program file.

USAGE

- Calling Sequence
  CALL OPNEOR ($LAB)
  Arguments:

Parameter name	In/Out	Dimension	Type	Description
$LAB	-	-	-	The label number to return to if element is not successfully opened

METHOD

- Model

  The UNIVAC-supplied package SOR is utilized to open the source element. SOR is documented in U.P. 4144, Rev. 3.

RESTRICTIONS

- Operational

  The subroutine PTABLE must be called to initialize the element directory packet before subroutine OPNEOR is called.

SUBROUTINE OUTIMG

IDENTIFICATION

Name (Title)	- OUTIMG (Out Image)
Programmer, Date	- D. M. Braley, May 1973
Author, Date	- P. H. Horsley, June 1976
Machine Identification	- UNIVAC 1100-Series
Source Language	- UNIVAC 1100-Series Assembler

PURPOSE

Subroutine OUTIMG inserts a single line image into a source element which has been opened in an EXEC 8 program file.

USAGE

- Calling Sequence

  CALL OUTIMG (IMAGE, $LAB)

  Arguments:

Parameter name	In/Out	Dimension	Type	Description
IMAGE	In	14	I	The line image to insert into the element, containing six fieldata characters per call
$LAB	-	-	-	The label number to return to if the line is not successfully inserted

METHOD

- Model

  The UNIVAC-supplied package SOR is used to insert a source image into an open element in a program file. SOR is documented in U.P. 4144, Rev. 3.

RESTRICTIONS

- Operational

  The subroutine OPNEOR must be called to open a source element in a program file before the initial call to subroutine OUTIMG.

SUBROUTINE PARTBL

IDENTIFICATION

NAME (Title)	- PARTBL (Part Table)
Programmer, Date	- D. M. Braley, May 1973
Author, Date	- P. H. Horsley, June 1976
Machine Identification	- UNIVAC 1100-Series
Source Language	- UNIVAC 1100-Series Assembler

PURPOSE

Subroutine PARTBL defines the array PARTBL for use by the UNIVAC SOR package in creating a source element in an EXEC 8 program file.

USAGE

This routine contains no FORTRAN callable entry points. It should be in an absolute element if subroutine PTABLE is used in the absolute element.

METHOD

- Model

  The symbolic name PARTBL is defined as an external address, and 40 cells of primary storage under location counter 0 are reserved, starting at the address. The format of the array PARTBL is documented in U.P. 4144, Rev. 3, p. 9-21.

SUBROUTINE PREERR

IDENTIFICATION

Name (Title)	- PREERR (Pre-Error)
Programmer, Date	- P. H. Horsley, April 1976
Machine Identification	- UNIVAC 1100-Series
Source Language	- FORTRAN V

PURPOSE

Subroutine PREERR prints all error messages for the Code Usage Analysis System (CUAS) preprocessor.

USAGE

- Calling Sequence
  CALL PREERR (IERR, ISTAT, IUN)
  Arguments:

Parameter name	In/Out	Dimension	Type	Description
IERR	In	1	I	A number indicating the routine in which an error occurred and the error type
ISTAT	In	1	I	Descriptive information about the error
IUN	In	1	I	The name of the program file containing absolute elements being processed by CUAPREPRO

FRECEDING PAGE BLANK NOT

- Error Messages

This routine prints fully descriptive error messages for the total CUAS preprocessor.

METHOD

- Model

Each error-producing subroutine of the CUAS preprocessor is assigned a unique routine number. When such a routine detects an error condition, an error cell is set with the detecting routine number (DRN) and the error type (ET). The two numbers are combined in the one cell by the formula IERR=DRN*100+ET. Subroutine PREERR decodes the error cell, which is passed to it as parameter IERR, and prints the appropriate error message format. The parameter ISTAT is used to pass any additional needed information to be printed in the error message. This technique of handling error messages allows for all error formats to be included in one element which may be inserted in an overlay segment and is thus never loaded unless an error is detected during processing.

RESTRICTIONS

- Operational

The subroutine LCORE is required.

SUBROUTINE PSTERR

IDENTIFICATION

Name (Title)	- PSTERR (Post Error)
Programmer, Date	- P. H. Horsley, May 1976
Machine Identification	- UNIVAC 1100-Series
Source Language	- FORTRAN V

PURPOSE

Subroutine PSTERR prints all error messages for the Code Usage Analysis System (CUAS) postprocessor.

USAGE

- Calling Sequence

  CALL PSTERR (IERR, ISTAT)

  Arguments:

Parameter name	In/Out	Dimension	Type	Description
IERR	In	1	I	A number indicating the routine in which an error occurred and the error type
ISTAT	In	1	I	Descriptive information about the error

- Labeled COMMON

  All of the labeled COMMON blocks TABLES, JHSCNT and CONTROL is required. The data in these COMMON blocks is used strictly as reference information, and no output occurs through the COMMON blocks.

- Error Messages

  This routine prints error messages for the total CUAS postprocessor, with the messages fully descriptive of the error which has occurred.

METHOD

- Model

  Each error-producing subroutine of the CUAS postprocessor is assigned a unique routine number. When such a routine detects an error condition, an error cell is set with the detecting routine number (DRN) and the error type (ET). The two numbers are combined in the one cell by the formula IERR=DRN*100+ET. Subroutine PSTERR decodes the error cell, which is passed to it as parameter IERR, and prints the appropriate error message format. The parameter ISTAT is used to pass any additional needed information to be printed in the error message. This technique of handling error messages allows for all error formats to be included in one element which may be inserted into an overlay segment and is thus never loaded unless an error is detected during processing.

RESTRICTIONS

- Operational

  Subroutines BD2FD and LCORE are required.

SUBROUTINE PTABLE

IDENTIFICATION

Name (Title)	- PTABLE (Part Table)
Programmer, Date	- D. M. Braley, May 1973
Author, Date	- P. H. Horsley, June 1976
Machine Identification	- UNIVAC 1100-Series
Source Language	- UNIVAC 1100-Series Assembler

PURPOSE

Subroutine PTABLE initializes an externalized array named PARTBL for the UNIVAC source element creation package SOR.

USAGE

- Calling Sequence

  CALL PTABLE (IO, IOPT, INOUT, IPKT)

  Arguments:

Parameter name	In/Out	Dimension	Type	Description
IO	In	1	I	Selection option, 0 = insert options word into the packet 1 = Source input 2 = Source output
IOPT	In	1	I	1 for data input if file name, 2 if element packet, 3 if both
INOUT	In	1	I	0 = move data from PARTBL into IPKT 1 = move data from IPKT into PARTBL

Parameter name	In/Out	Dimension	Type	Description
IPKT	In/Out	12	I	The packet which describes the element, which becomes cells 2-13 of array PARTBL.

METHOD

- Model

  Subroutine PTABLE initializes an externalized array of data which the UNIVAC SOR package uses to create a new source element in an EXEC 8 program file. The format of the array is described in U.P. 4144, Rev. 3, p. 9-21.

RESTRICTIONS

- Operational

  The subroutine PARTBL is required which contains the externalized array PARTBL. This routine is normally used in conjunction with subroutines OPNEOR, OUTIMG, and CLSEOR, with the following calling sequence.

```
CALL PTABLE (IO, IOPT, INOUT, IPKT)
CALL OPNEOR ($100)
CALL OUTIMG (IDAT, $100)     {1 call per line}
.
.
.
CALL OUTIMG (IDAT, $100)
CALL CLSEOR ($100)
```

At this point, a new source element has been inserted into a program file in accordance with the data in array PARTBL.

SUBROUTINE SLRPT

IDENTIFICATION

Name (Title)	- SLRPT (Segment Loading Report)
Programmer, Date	- P. H. Horsley, December 1975
Machine Identification	- UNIVAC 1100-Series
Source Language	- FORTRAN V

PURPOSE

Subroutine SLRPT produces the CUAS segment loading report and is intended for use only with the CUAS postprocessor.

USAGE

- Calling Sequence
  CALL SLRPT (IERR, ISTAT)
  Arguments:

Parameter name	In/Out	Dimension	Type	Description
IERR	Out	1	I	An error indicator, set nonzero to indicate an error and type
ISTAT	Out	1	I	A status flag associated with the error type

- Labeled COMMON

  All of the labeled COMMON blocks TABLES and CONTRL are required (refer to labeled COMMON block description in table 5-I). The data in both COMMON blocks is used strictly as reference data and no output occurs through the COMMON blocks.

- Error Messages

None - all errors are indicated by a nonzero value of the parameter IERR. IERR=1001 if an I/O error occurred on file JHS. IERR=1002 if dynamic core is not available for local storage arrays, in which case ISTAT contains the amount of core needed.

METHOD

- Model

Subroutine SLRPT scans the CUAS JHS to locate segment load event cells entered in the file by the CUAS contingency routine IICONT. The content and format of the JHS is presented in Appendix D. A sum of the total loads for each segment is maintained and once the entire JHS has been scanned, a report of the total loads for each segment is reported. The subroutine which when called resulted in a segment being loaded is determined from the subroutine call event cell in the JHS immediately following a segment load event cell. This information is also included in this report.

The names of the segments and subroutines are determined by correlating indexes and addresses respectively with the segment name table and external name table in COMMON block TABLES.

Subroutine SLRPT dynamically allocates the temporary storage needed in order to produce the segment loading report, and then dynamically releases the storage once the report is finished. This is accomplished by using the subroutines GCORE and RCORE. The amount of dynamic storage allocated is dependent solely on the size of the diagnostic tables from the absolute element for the program under analysis.

RESTRICTIONS

The subroutines GCORE and RCORE, and functions NGET, ILLSFT, and IRRSFT are required.

SUBROUTINE SLTSET

IDENTIFICATION

Name (Title)	- SLTSET (Segment Load Table Set) - SLTCLR (Segment Load Table Clear)
Programmer, Date	- P. H. Horsley, September 1975
Machine Identification	- UNIVAC 1100-Series
Source Language	- FORTRAN V

PURPOSE

Entry point SLTSET updates the segment load table such that it reflects the loaded segments of a segmented program.

Entry point SLTCLR resets the segment load table to initial load conditions, that is, only the root segment of a segmented program is loaded.

USAGE

- Calling Sequence

  CALL SLTSET (ISLT, LSLT, ISEG)

  CALL SLTCLR (ISLT, LSLT)

  Arguments:

Parameter name	In/Out	Dimension	Type	Description
ISLT	In/Out	LSLT	I	The segment load table which is updated and returned to the caller
LSLT	In	1	I	The length, in cells, of the segment load table
ISEG	In	1	I	The index of the segment which is to be indicated as loaded in the segment name table

METHOD

- Model

In order to resolve addressing ambiguities, the CUAS postprocessor must be able to determine which segments of a segmented program were in primary storage at any point during execution of a program. Subroutine SLTSET maintains the segment load table such that this determination may be made. The table is maintained exactly in format and content as it is by the EXEC 8 entry point LOAD$ during the execution of a program. The table may thus be examined by any other subroutine to determine which segments of an overlaid program were loaded at any time.

Subroutine SLTCLR merely clears the segment load table such that it reflects only the root segment loaded.

RESTRICTIONS

- Operational

Function IRRSFT is required.

SUBROUTINE SUPTIM

IDENTIFICATION

Name (Title)	- SUPTIM (SUP Time)
Programmer, Date	- P. H. Horsley, February 1976
Machine Identification	- UNIVAC 1100-Series
Source Language	- UNIVAC 1100-Series Assembler

PURPOSE

Subroutine SUPTIM retrieves the total Standard Unit of Processing (SUP) charges for an EXEC 8 job at the time of the call to the subroutine.

USAGE

- Calling Sequence

  CALL SUPTIM (ICAU, ICCER, IO)

  Arguments:

Parameter name	In/Out	Dimension	Type	Description
ICAU	Out	1	I	The CAU time in 200 micro-second increments
ICCER	Out	1	I	The executive time in 200 microsecond increments
IO	Out	1	I	The I/O time in 200 micro-second increments

METHOD

- Model

  The charges for a job running under EXEC 8 are maintained in an EXEC 8 controlled list called the Program Control Table (PCT). Subroutine SUPTIM references the PCT to retrieve the time charges for a job. The PCT is available for reference by an application program as a read-only D-BANK.

Subroutine SUPTIM reads the PCT by dropping the control D-BANK and basing the PCT on the primary Processor State Register (PSR). The control D-BANK is re-based on the primary PSR prior to returning to the caller.

RESTRICTIONS

- Operational

The UNIVAC LDJ instruction is used to base the PCT and the address of the PCT is supplied by the UNIVAC MAP in the external address tag RPCTA$. The effective PCT address is RPCTA$-1 on the 1108 hardware and the RPCTA$ on the 1110 hardware, thus a different version of this routine is required for each machine. The CAU time is also maintained in a different manner on the 1110 than it is on the 1108. Two versions of subroutine SUPTIM are available, one for the 1110 and one for the 1108, and are distinguished by the version names U1108 for the 1108 version and U1110 for the 1110. The formulas used by subroutine SUPTIM for calculating SUP charges may be found in Appendix A.

ՏUBROUTINE SUSRPT

IDENTIFICATION

Name	- SUSRPT (Subroutine Usage Report)
Programmer, Date	- P.·H. Horsley, May 1976
Machine Identification	- UNIVAC 1100-Series
Source Language	- FORTRAN V

PURPOSE

Subroutine SUSRPT creates a source element in a UNIVAC EXEC 8 program file. The element created contains the names of the subroutine elements used and not used during the execution of the program which created file JHS for analysis by the CUAS postprocessor.

USAGE

- Calling Sequence

  CALL SUSRPT (IERR, ISTAT)

  Arguments:

Parameter name	In/Out	Dimension	Type	Description
IERR	Out	1	I	An error condition indicator
ISTAT	Out	1	I	A status value associated with IERR

- Labeled COMMON

  All of the labeled COMMON block TABLES is required (refer to labeled COMMON block description in table 5-I). The data in the COMMON block is used strictly as reference information, anu no output of data occurs through the COMMON block.

- Error Messages

None - all errors are indicated by a nonzero value of the parameter IERR. IERR=1101 if file DBF was not successfully assigned, in which case ISTAT contains the facility status bits for rejection. IERR=1102 if no name was supplied for the source element to be created. IERR=1003 if an I/O error occurred on file DBF.

METHOD

- Model

Subroutine SUSRPT initially attempts to dynamically assign file DBF exclusively. If the file cannot be assigned exclusively, execution is suspended for 30,000 milliseconds and then another attempt is made to assigned exclusively. A source element is then opened in the file, and the names of subroutines used and not used are inserted into the file, one names of subroutines used and not used are inserted into the file, one source line per subroutine name. Once the element is created, the file DBF is dynamically freed so that other jobs running concurrently may assign and use it.

RESTRICTIONS

- Operational

Subroutines CSFER, WAITER, LCORE, PTABLE, OPNEOR, OUTIMG, CLSEOR, and BD2FD are required.

SUBROUTINE WAITER

IDENTIFICATION

Name (Title)	- WAITER (Wait Exec Request)
Programmer, Date	- P. H. Horsley, May 1976
Machine Identification	- UNIVAC 1100-Series
Source Language	- UNIVAC 1100-Series Assembler

PURPOSE

Subroutine WAITER performs an executive request to the EXEC 8 entry point TWAIT$, which will delay executive for a specified interval of time.

USAGE

- Calling Sequence

  CALL WAITER (ITIME)

  Argument:

Parameter name	In/Out	Dimension	Type	Description
ITIME	In	1	I	The time, in milliseconds, for program execution to be delayed. The upper limit is 30,000 milliseconds.

METHOD

- Model

  An executive request is performed to the EXEC 8 entry point TWAIT$ with the value of parameter ITIME in register A1. Upon return from the request, control is immediately returned to the caller.

SUBROUTINE XRRPT

IDENTIFICATION

Name (Title)	- XRRPT (External Reference Report)
Programmer, Date	- P. H. Horsley, December 1975
Machine Identification	- UNIVAC 1100-Series
Source Language	- FORTRAN V

PURPOSE

Subroutine XRRPT produces the CUAS external reference report and is intended for use only with the CUAS postprocessor.

USAGE

- Calling Sequence

  CALL XRRPT (IERR, ISTAT)

  Arguments:

Parameter name	In/Out	Dimension	Type	Description
IERR	Out	1	I	An error indicator, set nonzero to indicate an error and type
ISTAT	Out	1	I	A status flag associated with the error type

- Labeled COMMON

  All of the labeled COMMON blocks TABLES and CONTRL are required (refer to labeled COMMON block description in table 5-I). The data in both COMMON blocks is used strictly as reference data and no output of data occurs through the COMMON blocks.

PRECEDING PAGE BLANK NOT

- Error Messages

None - all errors are indicated by a nonzero value of the parameter IERR. IERR=1201 if the dynamic core for local storage arrays is not available, in which case parameter ISTAT contains the amount of core needed. IERR=1202 if an I/O error occurred on file JHS.

METHOD

- Model

The preprocessor execution options, which are available in COMMON block CONTRL, are examined to determine the type of report to be produced. The JHS is scanned to obtained data about the program execution under analysis. The content and format of the JHS is presented in Appendix D.

If option G was specified on the postprocessor execution, bits 16 and 17 of the date cell of each element name entry are used to indicate if the element was used (01), not used (10) or nonuser-supplied element (00). This data is used by subroutine SUSRPT to generate a source element containing the names of subroutines used and not used.

Subroutine XRRPT dynamically allocates the temporary storage needed in order to produce the external reference report, and then dynamically releases the storage once the report is finished. This is accomplished by using the subroutines GCORE and RCORE. The amount of dynamic storage allocated is dependent solely on the size of the diagnostic tables from the absolute element for the program under analysis.

RESTRICTIONS

- Operational

The subroutines GCORE, RCORE, SLTCLR, SLTSET, CKDOLR, BD2FD, FNDELT, and BUBSRT, and functions NGET, IRRSFT, and ILLSFT are required.

5.6 SAMPLE RUN STREAM/OUTPUT

The following pages contain the reports generated by the CUAS postprocessor during the execution of the run stream below.

```
 1.  @RUN
 2.  @ADD,P  FM9*SVDSDECKS.SETUP/MS37
 3.  @MAP,S  FM9*QQINPT.SVDS,RO.,SO.
 4.  @XQT  .SVDS
 5.  @ADD,P  TCL.DECK3
 6.  @USE  SI.,SV.
 7.  @COPY,R  FML-L79351*PHPA.IICONT,TPF$.
 8.  @MAP,S  SI.SVDS1,RO.,SO.
 9.  @XQT,C  FML-L79351*PHPA.CUAPREPRO
10.  @ASG,T  JHS.,F4/10//10
11.  @XQT  .SVDS1
12.  @XQT  FML-L79351*PHPA.CUAPSTPRO
13.  @FIN
```

Note that statements 7, 9, 10, and 12 are included solely for the use of the CUAS on the absolute element SVDS1 in file TPF$. The C option on statement 9 invokes the subroutine timing option of the CUAS. Statement 10 overrides the default size of the file JHS and specifies a maximum size of 10 tracks. Once 10 tracks of data have been collected, program analysis will terminate automatically.

```
CUAS POST-PROCESSOR

CODE USAGE ANALYSIS FOR EXECUTION OF ABSOLUTE SVD51          /
CREATED ON 08/12/76     AT 21:14:45

FILE JHS WAS FILLED AND CLOSED PRIOR TO END OF
SVD51         /                EXECUTION
G OPTION DIABLED

SVD51         /                WAS MODIFIED BY CUAS PRE-PROCESSOR
WITH C  OPTION SPECIFIED
```

```
                                  E X T E R N A L   U S A G E   R E P O R T

                          LIST OF EXTERNAL DEFINITIONS INCLUDED IN ABSOLUTE
          EXTERNALS DEFINED IN ELEMENT WITH S IN NAME OR FROM SYS$*RLIB$ NOT INCLUDED

 EXT.    IN ELEMENT     CREATION CREATION      EXT.    IN ELEMENT      CREATION CREATION
 NAME       NAME          DATE     TIME        NAME       NAME           DATE     TIME
ABINOI  ABINDI          08/12/76 21:12:49     ACS     ACS             08/12/76 21:12:49
ACSINT  ACSINT          08/12/76 21:12:49     ADAMS   INTEGR          09/09/75 18:10:47
ADSET   ADSET           08/12/76 21:12:50     ADSOP   ADSOP           08/12/76 21:12:50
AERO    AERO            08/12/76 21:12:51     ALIGN   ALIGN           03/02/76 16:31:48
ALTBLK  ALTBLK          08/12/76 21:12:51     ALTDLL  ALTDLL          04/07/76 16:28:42
AROBLK  AROBLK          09/09/75 18:04:39     AROCAL  AROCAL          06/02/76 14:03:03
AROLOK  AROLOK          04/07/76 16:29:25     ARO356  ARO356          03/02/76 16:32:07
ARTAN   ARTAN           09/05/75 18:29:31     ARI4UC  ARI40C          08/12/76 21:12:51
ASIGN   ASIGN           01/09/76 10:03:59     ATDISP  ATDISP          08/12/76 21:12:51
ATMOS   ATMOS           01/09/76 09:44:31     ATMOSI  ATMOSI          07/08/76 08:37:45
ATMSPL  ATMSPL          08/12/76 21:12:51     ATTITD  ATTITD          08/12/76 21:12:51
AUTLND  AUTLND          08/12/76 21:12:51     AUTO    AUTO            08/12/76 21:12:51
AVELOC  AVELOC          09/05/75 18:30:28     BALIGN  BALIGN          08/12/76 21:12:52
BDE74C  BDE74C          08/12/76 21:12:52     BDE74H  BDE74H          08/12/76 21:12:52
BDE75R  BDE75R          08/12/76 21:12:52     BDGLOB  BDGLOB          08/12/76 21:12:52
BDJACH  BDJACH          08/12/76 21:12:52     BDP63C  BDP63C          08/12/76 21:12:52
BDP63H  BDP63H          08/12/76 21:12:52     BDSP63  BDSP63          08/12/76 21:12:52
BDS30C  BDS30C          08/12/76 21:12:52     BDS30H  BDS30H          08/12/76 21:12:52
BDS60C  BDS60C          08/12/76 21:12:52     BDS60H  BDS60H          08/12/76 21:12:53
BDS62R  BDS62R          08/12/76 21:12:53     BDV71R  BDV71R          08/12/76 21:12:53
BDV73C  BDV73C          08/12/76 21:12:53     BDV73H  BDV73H          08/12/76 21:12:53
BD30C   BD30C           08/12/76 21:12:53     BD30H   BD30H           08/12/76 21:12:53
BD60C   BD60C           08/12/76 21:12:53     BD60H   BD60H           08/12/76 21:12:53
BD62R   BD62R           09/16/75 12:46:36     BEND    BEND            08/12/76 21:12:54
BENDI   BENDI           08/12/76 21:12:54     BLDA    BLDA            06/02/76 14:03:35
BMATRX  BMATRX          09/05/75 18:34:23     BMSTER  BMSTER          07/08/76 08:41:07
BMTRXT  BMTRXT          07/08/76 08:41:32     BNDRES  BNDRES          08/12/76 21:12:54
CASEIN  CASEIN          04/07/76 16:31:15     CATBLD  CATBLD          07/08/76 08:41:56
CLOCK   CLOCK           04/26/73 08:14:44     CMATRA  CMATRX          03/02/76 16:33:03
CONGID  CONGID          08/12/76 21:12:56     CONINT  CONINT          08/12/76 21:12:56
CORADD  CORADD          05/22/76 12:16:33     CROSS   CROSS           09/05/75 18:37:06
CSF     CSF             07/08/76 08:45:10     DAP30   DAP30           05/05/76 13:05:16
DATE    DATE            06/02/76 14:04:34     DELVT4  DELVT4          08/12/76 21:12:57
DISP    DISP            08/12/76 21:12:57     DNTERP  DNTERP          07/08/76 08:46:19
DOT     DOT             09/05/75 18:39:10     DRGCMP  DRGCMP          08/12/76 21:12:57
DTAPE   DTAPE           08/12/76 21:12:57     DTCALC  DTCALC          09/05/75 18:40:35
DTCHEK  DTCHEK          09/05/75 18:40:40     DTMIN   DTMIM           11/05/75 11:09:01
DTPDIS  DTPDIS          07/08/76 08:46:56     DTYET   DTYET           01/09/76 09:47:00
DVSOP   DVSOP           08/12/76 21:12:57     DYNAMC  DYNAMC          05/05/76 13:06:27
DYNTCC  DYNTCC          07/08/76 08:47:06     ECA74   ECA74           08/12/76 21:12:58
ECIALN  ECIALN          09/05/75 18:42:44     EFNAME  EFNAME          05/30/73 07:16:19
EHA74   EHA74           08/12/76 21:12:58     ENGDOT  ENGDDT          08/12/76 21:12:58
ENTIO   ENTID           07/08/76 08:47:31     EWCOM   EWCOM           08/12/76 21:12:58
ERA75   ERA75           08/12/76 21:12:58     ETG     ETG             08/12/76 21:12:58
ETGX    ETGX            08/12/76 21:12:58     ETGXI   ETGXI           08/12/76 21:12:58
ETOXYZ  ETOXYZ          09/05/75 18:43:45     ETPLMI  ETPLMI          08/12/76 21:12:58
FCSA    FCSA            08/12/76 21:12:58     FCSAI   FCSAI           08/12/76 21:12:58
FDATA   FDATA           08/12/76 21:12:58     FILINI  FILINT          08/12/76 21:12:59
FILSET  FILSET          08/12/76 21:12:59     FILTER  FILTER          08/12/76 21:12:59
FLZERO  FLZERO          01/09/76 09:29:17     FMCOM   FMCOM           08/12/76 21:12:59
```

```
FNAME   FNAME        04/23/75  11:38:14      FREAD   FILEIO       05/10/75  00:53:44
FSTRCS  FSTRCS       08/12/76  21:13:00      FSWDE   FSWDE        07/08/76  08:49:42
GEOD    ALTOLL       04/07/76  16:28:42      GETDAT  GETDAT       03/02/76  16:41:15
GIMBAL  GIMBAL       05/05/76  13:15:07      GLOAD   GLOAD        08/12/76  21:13:00
GLOBAL  GLOBAL       08/12/76  21:13:00      GMTIME  GMTIME       09/05/75  18:47:46
GNTERP  GNTERP       07/08/76  08:51:09      GRAVTY  GRAVTY       04/20/76  23:44:39
GTURN   GTURN        04/07/76  16:34:27      HEAD    HEAD         07/30/76  05:46:29
HEAT    HEAT         05/05/76  13:15:23      HPIC    HPIC         08/12/76  21:13:00
HSTDIF  HSTDIF       08/12/76  21:13:00      HTRATE  HTRATE       07/08/76  08:51:43
IDJS    IICONT       08/05/76  13:40:37      IMPACT  IMPACT       08/12/76  21:13:01
INITAL  INITAL       06/02/76  14:09:23      INIT3   INIT3        08/12/76  21:13:01
INTEGA  INTEGA       08/12/76  21:13:01      INTEGI  INTEGI       03/02/76  16:36:03
INTEG3  INTEG3       08/12/74  21:13:01      INTEG4  INTEG4       08/12/76  21:13:01
INTEG5  INTEG5       08/12/76  21:13:01      INTEG6  INTEG6       08/12/76  21:13:01
INTERP  INTERP       07/08/76  08:52:54      INTGIA  INTGIA       08/12/76  21:13:01
INTGI1  INTGI1       09/09/75  18:11:16      INTGI3  INTGI3       08/12/76  21:13:01
INTGI5  INTGI5       08/12/76  21:13:01      INVERT  INVERT       09/08/75  01:01:18
ITEREX  ITEREX       08/12/76  21:13:01      IVRSTI  IVRSTI       08/12/76  21:13:01
JSLTHD  JSLTHD       08/12/76  21:13:02      JUMP    JUMP         11/05/75  11:12:45
KILLER  TRACER       01/09/76  10:04:10      KKLOC   KKLOC        09/08/75  01:04:44
LIMSET  LIMSET       08/12/76  21:13:02      LINECT  LINECT       01/09/76  09:57:39
LINEUP  LINECT       01/09/76  09:57:39      LINKI   LINKI        08/12/76  21:13:02
LLOAD   LLOAD        07/08/76  08:54:19      LNCHI   LNCHI        08/12/76  21:13:03
LNCHID  LNCHID       08/12/76  21:13:03      LOAD    LOAD         07/08/76  08:54:27
LOADIT  LSDATA       06/02/76  15:51:57      MAENGI  MAENGI       08/12/76  21:13:03
MATE07  MATE07       08/12/76  21:13:03      MATRIX  MATHSB       01/09/76  09:32:34
MATROT  MATROT       09/05/75  08:59:32      MAXQ    MAXQ         08/12/76  21:13:03
MCBLD2  MCBLD2       08/12/76  21:13:03      MCDISP  MCDISP       08/12/76  21:13:03
MEASI   MEASI        08/12/76  21:13:03      MMDALT  MMDALT       08/12/76  21:13:03
MMDE    MMDE         07/08/76  08:55:48      MMDGE   MMDGE        08/12/76  21:13:04
MMDGN   MMDGN        08/12/76  21:13:04      MMDGO   MMDGO        08/12/76  21:13:04
MMDI    SETJ         07/08/76  09:03:31      MMDL    MMDL         08/12/76  21:13:04
MMDO    MMDO         08/12/76  21:13:05      MMDS    MMDS         08/12/76  21:13:06
MODELI  MODELI       07/08/76  08:56:06      MONITR  MONITR       08/12/76  21:13:06
MONIT2  MONIT2       05/05/76  13:27:24      MOVEIT  LSDATA       06/02/76  15:51:57
MOVER   MOVER        05/26/73  16:20:27      MTRXMP  MTRXMP       04/07/76  16:38:56
MULT    MULT         09/05/75  09:04:36      MYRIAH  MYRIAH       08/12/76  21:13:06
NAVESU  NAVESU       08/12/76  21:13:06      NAVEVT  NAVEVT       08/12/76  21:13:06
NAVI    NAVI         08/12/76  21:13:06      NAVMON  NAVMON       08/12/76  21:13:06
NAVQUN  NAVQUN       05/05/76  13:27:42      NINTRS  IICONT       08/05/76  13:40:37
NROTI   NROTI        08/12/76  21:13:06      OBSINT  OBSINT       08/12/76  21:13:07
OMPDAT  OMPDAT       08/12/76  21:13:07      ONSTEP  ONSTEP       11/05/75  11:16:06
ORBDRV  ORBDRV       08/12/76  21:13:07      ORBID   ORBID        08/12/76  21:13:07
OREBLK  OREBLK       08/12/76  21:13:07      ORIENT  ORIENT       07/15/76  00:06:52
OUTPRC  OUTPRC       07/08/76  08:57:00      OUT31   OUT31        08/12/76  21:13:07
PARALB  PARALB       09/05/75  09:12:18      PCHPLN  PHSPLN       07/08/76  08:58:29
PCTD    PCTD         02/24/76  14:37:00      PERTSI  PERTSI       08/12/76  21:13:08
PHSCG   PHSCG        07/08/76  08:58:12      PHSEXC  PHSEXC       07/08/76  08:58:16
PHSINT  PHSINT       07/08/76  08:58:24      PHSPLN  PHSPLN       07/08/76  08:58:29
PLTINT  PLTINT       08/12/76  21:13:08      PLTSGM  PLTSGM       08/12/76  21:13:08
PMATRX  PMATRX       06/02/76  14:23:28      POSIT   POSIT        03/02/76  16:37:51
PRA63C  PRA63C       08/12/76  21:13:08      PRA63H  PRA63H       08/12/76  21:13:08
PRBMMD  PRBMMD       09/05/75  09:15:13      PREADR  INTEGR       09/09/75  18:10:47
PREADT  INTEGR       09/09/75  18:10:47      PREIGM  PREIGM       08/12/76  21:13:08
PREQUA  PREQUA       08/12/76  21:13:08      PRHEAD  PRHEAD       07/08/76  08:58:43
PRINT   PRINT        09/05/75  09:16:12      PRTIME  PRTIME       01/09/76  09:40:56
PS      PSBMMD       09/05/75  09:19:03      PSERS     THFN       09/05/75  08:58:10
```

```
LTADR   INTEGR     09/09/75  18:10:47      PSTADI  INTEGR     09/09/75  18:10:47
PWEIGH  PWEIGH     09/05/75  09:19:08      PYRANG  PYRANG     06/02/76  14:28:39
QWEXIT  QWEXIT     03/11/76  09:25:31      RADAR   RADAR      08/12/76  21:13:09
RANGEP  RANGEP     08/12/76  21:13:09      RANGF   RANGF      08/12/76  21:13:09
RCSENT  RCSENT     08/12/76  21:13:09      RCSINT  RCSINT     08/12/76  21:13:09
RDER31  RDER31     08/12/76  21:13:09      READI1  LSDATA     06/02/76  15:51:57
REDIMU  REDIMU     08/12/76  21:13:09      RELATV  RELATV     08/12/76  21:13:09
RESET   CLOCK      04/26/73  08:14:44      RGB3    RGB3       08/12/76  21:13:09
RIDMAP  RIDMAP     05/07/76  14:35:10      RKG1    INTEGR     09/09/75  18:10:47
RKG2    INTEGR     09/09/75  18:10:47      RKG3    INTEGR     09/09/75  18:10:47
RKG4    INTEGR     09/09/75  18:10:47      RNDDAT  RNDDAT     08/12/76  21:13:10
RNDINT  RNDINT     08/12/76  21:13:10      RNGERR  RNGERR     08/12/76  21:13:10
ROLPLN  PHSPLN     07/08/76  08:58:29      ROTBLK  ROTBLK     08/12/76  21:13:10
ROTDER  ROTDER     09/05/75  09:26:38      ROTMAT  ROTMAT     09/05/75  09:26:55
ROTRES  INTEGR     09/09/75  18:10:47      ROTR31  INTEGR     09/09/75  18:10:47
RWNC    RWNC       08/12/76  21:13:10      RXAAP   RXAAP      08/12/76  21:13:10
SDGLDT  SDGLDT     08/12/76  21:13:10      SDRIVE  SDRIVE     08/12/76  21:13:10
SEARCH  SEARCH     04/07/76  16:57:05      SENDE   SENDE      08/12/76  21:13:10
SEPO    SEPO       08/12/76  21:13:11      SEPDI   SEPDI      08/12/76  21:13:11
SEPI    SEPI       08/12/76  21:13:11      SETJ    SETJ       07/08/76  09:03:31
SINIT   SINIT      08/12/76  21:13:11      SLOSH   SLOSH      08/12/76  21:13:11
SLOSHI  SLOSHI     08/12/76  21:13:11      SLOSKS  INTEGR     09/09/75  18:10:47
SNAVO   SNAVO      08/12/76  21:13:11      SNAVI   SNAVI      08/12/76  21:13:11
SPGPRT  SPGPRT     08/12/76  21:13:11      STEER   STEER      04/07/76  16:47:04
STGIO   STGIO      08/12/76  21:13:12      STIME   STIME      02/26/76  19:41:49
STORE   STORE      07/08/76  09:05:39      STORIT  LSDATA     06/02/76  15:51:57
STPOVR  STPOVR     06/02/76  14:30:22      SUBLST  SUBLST     08/12/76  21:13:12
SUMF    SUMF       09/09/75  18:19:13      TA      TA         09/05/75  09:37:19
TABLEP  TABLEP     07/08/76  09:06:03      TABPRI  TABPRT     04/07/76  16:48:21
TAEM    TAEM       08/12/76  21:13:13      TAPEND  TAPEND     06/02/76  14:31:04
TDERI   TDERI      09/05/75  09:39:29      TDER31  TDER31     08/12/76  21:13:13
TERMIN  TERMIN     06/02/76  14:31:16      TERMPT  TERMPT     06/02/76  14:31:27
TERMX   TERMX      06/02/76  14:31:34      TGUESS  TGUESS     03/15/76  21:35:11
TI      TI         01/09/76  09:55:55      TIMEPF  TIMERS     11/05/75  11:20:16
TIMEPH  TIMERS     11/05/75  11:20:16      TIMERS  TIMERS     11/05/75  11:20:16
TIMERU  TIMERS     11/05/75  11:20:16      TIMESF  TIMERS     11/05/75  11:20:16
TIMESH  TIMERS     11/05/75  11:20:16      TIMESN  TIMERS     11/05/75  11:20:16
TITLE   TITLE      07/30/76  05:48:59      TLOKUP  TLOKUP     07/08/76  09:07:32
TOPBLK  TOPBLK     08/12/76  21:13:13      TOPODT  TOPODT     09/05/75  09:43:04
TPS     TPS        07/08/76  09:07:59      TPS#R   TPS#R      03/02/76  16:40:41
TRACE   TRACER     01/09/76  10:04:10      TRACER  TRACER     01/09/76  10:04:10
TRANS   MATHSB     01/09/76  09:32:34      TRIM    TRIM       08/12/76  21:13:13
TRJBLK  TRJBLK     06/02/76  14:31:43      TRJCAL  TRJCAL     06/02/76  14:31:51
TRNDER  TRNDER     09/09/75  18:19:39      TRNRES  INTEGR     09/09/75  18:10:47
TRNRSI  INTEGR     09/09/75  18:10:47      TRNR31  INTEGR     09/09/75  18:10:47
UNIT    UNIT       09/05/75  09:47:50      UNVEC   UNVEC      09/05/75  09:48:01
UPDATE  UPDATE     07/08/76  09:08:25      VARMAS  VARMAS     08/12/76  21:13:13
VECBLK  VECBLK     01/09/76  19:13:45      VECMG   VECMG      09/05/75  09:49:25
VELOC   VELOC      01/09/76  09:56:37      VPLOT   VPLOT      04/07/76  16:49:19
VPRINT  VPRINT     08/12/76  21:13:13      VRA71   VRA71      08/12/76  21:13:13
VRA73C  VRA73C     08/12/76  21:13:13      VRA73H  VRA73H     08/12/76  21:13:13
VSIPRT  VSIPRT     07/08/76  09:09:33      WAIT30  WAIT30     07/08/76  09:09:51
WRITEX  WRITEX     09/05/75  09:53:25      XDATE   XDATE      08/12/76  21:13:13
XQTOPS  XQTOPS     05/14/76  13:56:13      ZERO11  LSDATA     06/02/76  15:51:57
```

```
                 LIST OF EXTERNAL DEFINITIONS REFERENCED BY LHJ DURING EXECUTION
        EXTERNALS DEFINED IN ELEMENT WITH $ IN NAME OR FROM SYS$*RLIBS NOT INCLUDED
                                 ALL TIMES IN SUP SECONDS

  EXT.     NO.      TOTAL     MAX SUPS   MIN SUPS   TOTAL    TOTAL    TOTAL     AVG.     AVG.     AVG.
  NAME     REFS.    SUPS      ANY REF.   ANY REF.   CAU      CC/ER    I/O       CAU      CC/ER    I/O
 FMAIN$      1      .0013     .0013      .0013     .0003    .0010    .0000     .0003    .0010    .0000
 ADSET       1      .0000     .0000      .0000     .0000    .0000    .0000     .0000    .0000    .0000
 ALIGN       2      .0008     .0005      .0003     .0008    .0000    .0000     .0004    .0000    .0000
 ALTDLL     14      .0022     .0004      .0001     .0022    .0000    .0000     .0002    .0000    .0000
 AROBLK      1      .0001     .0001      .0001     .0001    .0000    .0000     .0001    .0000    .0000
 AROCAL      7      .0015     .0006      .0001     .0015    .0000    .0000     .0002    .0000    .0000
 AROLOK     13      .0014     .0004      .0000     .0014    .0000    .0000     .0001    .0000    .0000
 ARO356      7      .0007     .0002      .0000     .0007    .0000    .0000     .0001    .0000    .0000
 ARTAN      14      .0004     .0002      .0000     .0004    .0000    .0000     .0000    .0000    .0000
 ASIGN       1      .8463     .8464      .8464     .0002    .8462    .0000     .0002    .8462    .0000
 ATMOS       7      .0021     .0006      .0001     .0021    .0000    .0000     .0003    .0000    .0000
 ATMOSI      1      .0000     .0000      .0000     .0000    .0000    .0000     .0000    .0000    .0000
 AVELOC      1      .0002     .0002      .0002     .0002    .0000    .0000     .0002    .0000    .0000
 BD62R       1      .0000     .0000      .0000     .0000    .0000    .0000     .0000    .0000    .0000
 BLOA        1      .0006     .0006      .0006     .0006    .0000    .0000     .0006    .0000    .0000
 BHTRXT      5      .0013     .0006      .0001     .0013    .0000    .0000     .0002    .0000    .0000
 CASEIN      1      .0002     .0002      .0002     .0002    .0000    .0000     .0002    .0000    .0000
 CATBLU      1      .2618     .2618      .2618     .1461    .0716    .0442     .1461    .0716    .0442
 CLOCK       1      .0003     .0003      .0003     .0001    .0002    .0000     .0001    .0002    .0000
 CHATRX      1      .0176     .0176      .0176     .0104    .0072    .0000     .0104    .0072    .0000
 CORADO     63      .0578     .0021      .0000     .0578    .0000    .0000     .0009    .0000    .0000
 CROSS      26      .0019     .0003      .0000     .0019    .0000    .0000     .0001    .0000    .0000
 DAP3D       5      .0003     .0002      .0000     .0003    .0000    .0000     .0001    .0000    .0000
 DNTERP      7      .0066     .0021      .0004     .0066    .0000    .0000     .0009    .0000    .0000
 DOT        78      .0016     .0002      .0000     .0016    .0000    .0000     .0000    .0000    .0000
 DTCALC      1      .0003     .0003      .0003     .0003    .0000    .0000     .0003    .0000    .0000
 DTCHEK      2      .0002     .0002      .0001     .0002    .0000    .0000     .0001    .0000    .0000
 DTPDIS      1      .0914     .0914      .0914     .0720    .0194    .0000     .0720    .0194    .0000
 DYNAMC     16      .0276     .0273      .0000     .0006    .0070    .0200     .0070    .0004    .0013
 DYNTCC      3      .0030     .0000      .0000     .0000    .0000    .0000     .0000    .0000    .0000
 ECIALN      1      .0000     .0000      .0000     .0000    .0000    .0000     .0000    .0000    .0000
 EFNAME      1      .0011     .0012      .0012     .0002    .0010    .0000     .0002    .0010    .0000
 ENGDOT      1      .0000     .0000      .0000     .0000    .0000    .0000     .0000    .0000    .0000
 ENTIO       1      .0169     .0169      .0169     .0113    .0056    .0000     .0113    .0056    .0000
 FLZERO      1      .0003     .0003      .0003     .0003    .0000    .0000     .0033    .0000    .0000
 FNAME       1      .0000     .0000      .0000     .0000    .0000    .0000     .0000    .0000    .0000
 FREAD     426     2.5992     .0167      .0056     .0860    .4260   2.0872     .0002    .0010    .0049
 FSHDE       2      .0002     .0002      .0001     .0002    .0000    .0000     .0001    .0000    .0000
 GEVD        1      .0005     .0005      .0005     .0005    .0000    .0000     .0005    .0000    .0000
 GETDAT      1      .0003     .0003      .0003     .0001    .0002    .0000     .0001    .0002    .0000
 GIMBAL      2      .0008     .0005      .0004     .0008    .0000    .0000     .0004    .0000    .0000
 GMTIME      4      .0002     .0001      .0000     .0002    .0000    .0000     .0000    .0000    .0000
 GNTERP      7      .0002     .0001      .0000     .0002    .0000    .0000     .0000    .0000    .0000
 GRAVTY      6      .0020     .0007      .0002     .0020    .0000    .0000     .0003    .0000    .0000
 HEAD        4      .0683     .0178      .0169     .0587    .0096    .0000     .0147    .0024    .0000
 HTRATE      2      .0037     .0023      .0015     .0029    .0008    .0000     .0015    .0004    .0000
 INITAL      1      .0003     .0003      .0003     .0003    .0000    .0000     .0003    .0000    .0000
 INTEGI      2      .0501     .0501      .0000     .0005    .0070    .0426     .0003    .0035    .0213
 INTGII      1      .0213     .0214      .0214     .0004    .0070    .0140     .0004    .0070    .0140
```

```
              LIST OF EXTERNAL DEFINITIONS REFERENCED BY LMJ DURING EXECUTION
         EXTERNALS DEFINED IN ELEMENT WITH $ IN NAME OR FROM SYS$*RLIBS NOT INCLUDED
                              ALL TIMES IN SUP SECONDS

 EXT.     NO.     TOTAL    MAX SUPS   MIN SUPS   TOTAL    TOTAL    TOTAL    AVG.     AVG.     AVG.
 NAME    REFS.    SUPS     ANY REF.   ANY REF.   CAU      CC/ER    I/O      CAU      CC/ER    I/O
INVERT      1     .0000     .0000      .0000     .0000    .0000    .0000    .0000    .0000    .0000
JUMP        4     .0001     .0001      .0000     .0001    .0000    .0000    .0000    .0000    .0000
KKLOC     370     .0078     .0004      .0000     .0078    .0000    .0000    .0000    .0000    .0000
LINECT     17     .0003     .0002      .0000     .0003    .0000    .0000    .0000    .0000    .0000
LINEUP     75     .0019     .0003      .0000     .0019    .0000    .0000    .0000    .0000    .0000
LLUAO      70     .0030     .0004      .0003     .0030    .0000    .0000    .0000    .0000    .0000
LOAD        1     .3240     .3241      .3241     .0133    .0144    .2964    .0133    .0144    .2964
LOADIT     70     .0008     .0002      .0000     .0008    .0000    .0000    .0000    .0000    .0000
MATROT      6     .0002     .0001      .0000     .0002    .0000    .0000    .0000    .0000    .0000
MMDE       16     .0981     .0777      .0000     .0011    .0078    .0892    .0001    .0005    .0056
MMUI       16     .0001     .0001      .0000     .0001    .0000    .0000    .0000    .0000    .0000
MODELI      1     .0000     .0003      .0000     .0000    .0000    .0000    .0000    .0000    .0000
MONIT2      4     .0001     .0001      .0000     .0001    .0000    .0000    .0000    .0000    .0000
MOVEIT      4     .0000     .0000      .0000     .0000    .0000    .0000    .0000    .0000    .0000
MOVER       9     .0003     .0002      .0000     .0003    .0000    .0000    .0000    .0000    .0000
MIRXMP     37     .0097     .0007      .0001     .0097    .0000    .0000    .0003    .0000    .0000
MULT        5     .0006     .0002      .0001     .0006    .0000    .0000    .0001    .0000    .0000
NAVGUN      2     .0006     .0005      .0002     .0006    .0000    .0000    .0003    .0000    .0000
NINTRS      1     .0002     .0002      .0002     .0000    .0002    .0000    .0000    .0002    .0000
ONSTEP      2     .0000     .0000      .0000     .0000    .0000    .0000    .0000    .0000    .0000
ORIENT      1     .0001     .0001      .0001     .0001    .0000    .0000    .0001    .0000    .0000
OUTPRC      2     .0093     .0090      .0003     .0085    .0008    .0000    .0042    .0004    .0000
PARALB    105     .0023     .0003      .0000     .0023    .0000    .0000    .0000    .0000    .0000
PCID        5     .0012     .0003      .0003     .0003    .0010    .0000    .0001    .0002    .0000
PHSEXC      1     .0711     .0712      .0712     .0002    .0078    .0632    .0002    .0078    .0632
PHSINT      1     .0004     .0004      .0004     .0004    .0000    .0000    .0004    .0000    .0000
PHSPLN      4     .0003     .0002      .0000     .0003    .0000    .0000    .0001    .0000    .0000
PMATRX      4     .0005     .0004      .0000     .0005    .0000    .0000    .0001    .0000    .0000
POSIT       1     .0003     .0003      .0003     .0003    .0000    .0000    .0003    .0000    .0000
PRHEAD      2     .0068     .0035      .0034     .0036    .0032    .0000    .0018    .0016    .0000
PRINT       2     .0416     .0287      .0130     .0352    .0064    .0000    .0176    .0032    .0000
PRTIME      1     .0208     .0208      .0208     .0160    .0048    .0000    .0160    .0048    .0000
PSERS      21     .0048     .0007      .0001     .0048    .0000    .0000    .0002    .0000    .0000
PHEIGH     14     .0002     .0001      .0000     .0002    .0000    .0000    .0000    .0000    .0000
PYRANG      4     .0003     .0001      .0001     .0003    .0000    .0000    .0001    .0000    .0000
READIT      4     .1118     .0848      .0006     .0204    .0026    .0886    .0051    .0007    .0221
RESET       2     .0009     .0006      .0004     .0003    .0006    .0000    .0002    .0003    .0000
RKG1        2     .0002     .0002      .0001     .0002    .0000    .0000    .0001    .0000    .0000
RKG2        1     .0003     .0003      .0003     .0003    .0000    .0000    .0003    .0000    .0000
RKG3        1     .0001     .0001      .0001     .0001    .0000    .0000    .0001    .0000    .0000
RKG4        1     .0001     .0001      .0001     .0001    .0000    .0000    .0001    .0000    .0000
RNDDAT      1     .0000     .0000      .0000     .0000    .0000    .0000    .0000    .0000    .0000
ROTMAT      5     .0008     .0003      .0000     .0008    .0000    .0000    .0002    .0000    .0000
SEARCH     14     .0011     .0003      .0000     .0011    .0000    .0000    .0001    .0000    .0000
SETJ        2     .0000     .0000      .0000     .0000    .0000    .0000    .0000    .0000    .0000
SNAVO       1     .0000     .0000      .0000     .0000    .0000    .0000    .0000    .0000    .0000
STEER       2     .0002     .0002      .0000     .0002    .0000    .0000    .0001    .0000    .0000
STIME       2     .0001     .0001      .0000     .0001    .0000    .0000    .0001    .0000    .0000
STORE     116     .0044     .0004      .0000     .0044    .0000    .0000    .0000    .0000    .0000
```

```
          LIST OF EXTERNAL DEFINITIONS REFERENCED BY LMJ DURING EXECUTION
EXTERNALS DEFINED IN ELEMENT WITH $ IN NAME OR FROM SYS$*RLIB$ NOT INCLUDED
                          ALL TIMES IN SUP SECONDS

EXT.      NO.      TOTAL     MAX SUPS   MIN SUPS    TOTAL     TOTAL     TOTAL     AVG.      AVG.      AVG.
NAME     REFS.     SUPS      ANY REF.   ANY REF.    CAU       CC/ER     I/O       CAU       CC/ER     I/O
STURIT    116      .0019      .0003      .0000      .0019     .0000     .0000     .0000     .0000     .0000
STPOVR      2      .0000      .0000      .0000      .0000     .0000     .0000     .0000     .0000     .0000
SUMF        6      .0002      .0001      .0000      .0002     .0000     .0000     .0000     .0000     .0000
TABLEP      2      .0001      .0001      .0001      .0001     .0000     .0000     .0001     .0000     .0000
TDERI       5      .0008      .0003      .0001      .0008     .0000     .0000     .0001     .0000     .0000
TERMIN      2      .0001      .0001      .0000      .0001     .0000     .0000     .0000     .0000     .0000
TERMX       2      .0000      .0000      .0000      .0000     .0000     .0000     .0000     .0000     .0000
TI          2      .0035      .0020      .0016      .0035     .0000     .0000     .0018     .0000     .0000
TIMEPH      3      .0000      .0000      .0000      .0000     .0000     .0000     .0000     .0000     .0000
TIMERS      1      .0000      .0000      .0000      .0000     .0000     .0000     .0000     .0000     .0000
TIMERU      1      .0000      .0000      .0000      .0000     .0000     .0000     .0000     .0000     .0000
TIMESH      3      .0001      .0001      .0000      .0001     .0000     .0000     .0000     .0000     .0000
TIMESN      3      .0000      .0000      .0000      .0000     .0000     .0000     .0000     .0000     .0000
TITLE       1      .0875      .0875      .0875      .0567     .0308     .0000     .0567     .0308     .0000
TLOKUP      4      .0005      .0002      .0000      .0005     .0000     .0000     .0001     .0000     .0000
TPS         2      .0047      .0026      .0021      .0047     .0000     .0000     .0023     .0000     .0000
TRANS       1      .0000      .0000      .0000      .0000     .0000     .0000     .0000     .0000     .0000
TRJBLK      1      .0000      .0000      .0000      .0000     .0000     .0000     .0000     .0000     .0000
TRJCAL      2      .0008      .0006      .0002      .0008     .0000     .0000     .0004     .0000     .0000
TRNRS1      5      .0002      .0001      .0000      .0002     .0000     .0000     .0000     .0000     .0000
UNIT       16      .0008      .0002      .0000      .0008     .0000     .0000     .0000     .0000     .0000
UNVEC      27      .0012      .0002      .0000      .0012     .0000     .0000     .0000     .0000     .0000
UPDATE      2      .0002      .0002      .0001      .0002     .0000     .0000     .0001     .0000     .0000
VECMG      74      .0032      .0002      .0000      .0032     .0000     .0000     .0000     .0000     .0000
VELOC       1      .0008      .0008      .0008      .0008     .0000     .0000     .0008     .0000     .0000
VPLOT       3      .0076      .0035      .0020      .0054     .0022     .0000     .0018     .0007     .0000
VSIPRT      1      .0318      .0318      .0318      .0222     .0096     .0000     .0222     .0096     .0000
XUTOPS      1      .0003      .0003      .0003      .0001     .0002     .0000     .0001     .0002     .0000

COLUMN TOTALS      4.9641                           .7163    1.5024    2.7454
```

```
              LIST OF EXTERNAL DEFINITIONS NOT REFERENCED BY LMJ DURING EXECUTION
       EXTERNALS DEFINED IN ELEMENT WITH $ IN NAME OR FROM SYS$*RLIBS NOT INCLUDED
            Δ INDICATES EXTERNAL IN ELEMENT WITH I-BANK LENGTH LESS THAN 8 CELLS

      EXT.       EXT.       EXT.       EXT.       EXT.       EXT.       EXT.
      NAME       NAME       NAME       NAME       NAME       NAME       NAME
    ΔABINUI    ΔACS       ΔACSINI     AUAMS     ΔADSOP     ΔAERU      ΔALTBLK
    ΔARI40C    ΔATDISP    ΔATMSPL    ΔATTITO    ΔAUTLND    ΔAUTU      ΔBALIGN
    ΔBDE74C    ΔBDE74H    ΔBDE75R    ΔBDGLOB    ΔBDJACH    ΔBDP63C    ΔBDP63H
    ΔBDSP63    ΔBDS30C    ΔBDS30H    ΔBDS60C    ΔBDS60H    ΔBDS62R    ΔBDV71R
    ΔBDV73C    ΔBDV73H    ΔBD30C     ΔBD30H     ΔBD60C     ΔBD60H     ΔBEND
    ΔBENDI      BMATRX     BMSTEK    ΔBNDRES    ΔCONGID    ΔCONINT    ΔCSF
     DATE      ΔDELVT4    ΔDISP      ΔDRGCMP    ΔDTAPE      DTMIM      DTYET
    ΔDVSOP     ΔECA74     ΔEHA74     ΔEWCOM     ΔERA75     ΔETG       ΔETGX
    ΔETGXI      ETUXYZ    ΔETPLMI    ΔFCSA      ΔFCSAI     ΔFDATA     ΔFILINT
    ΔFILSET    ΔFILTER    ΔFMCOM     ΔFSTRCS    ΔGLOAD     ΔGLOBAL     GTURN
     HEAT      ΔHPIC      ΔHSTDIF     IDJS      ΔIMPACT    ΔINIT3     ΔINTEGA
    ΔINTEG3    ΔINTEG4    ΔINTEG5    ΔINTEG6     INTERP    ΔINTGIA    ΔINTGI3
    ΔINTGI5    ΔITEREX    ΔIVRSTI    ΔJSLTHD     KILLER    ΔLIMSET    ΔLINKI
    ΔLNCHI     ΔLNCHID    ΔMAENGI    ΔMATE07     MATRIX    ΔMAXW      ΔMCBLD2
    ΔMCDISP    ΔMEASI     ΔMMDALT    ΔMHDGE     ΔMMDGN     ΔMMDGO     ΔMMDL
    ΔMMDO      ΔMMDS      ΔMONITR    ΔMYRIAH    ΔNAVESU    ΔNAVEVT    ΔNAVI
    ΔNAVMON    ΔNROTI     ΔOBSINT    ΔOMPDAT    ΔORBDRV    ΔORBID     ΔOREBLK
    ΔOUT31      PCHPLN    ΔPERTSI     PHSCG     ΔPLTINT    ΔPLTSGM    ΔPRA63C
    ΔPRA63H     PRBMMD     PREADR     PREADT    ΔPREIGM    ΔPREQUA     PSBMMD
     PSTADR     PSTADT    ΔQQEXIT    ΔRADAR     ΔRANGEP    ΔRANGF     ΔRCSENT
    ΔRCSINT    ΔRDER31    ΔREDIMU    ΔRELATV    ΔRG83       RIDMAP    ΔRNDINT
    ΔRNGERR     RULPLN    ΔROTBLK     RUTDER     ROTRES     ROTR31    ΔRWNC
    ΔRXAAP     ΔSDGLDT    ΔSDRIVE    ΔSENDE     ΔSEPD      ΔSEPDI     ΔSEPI
    ΔSINIT     ΔSLOSH     ΔSLOSHI     SLOSRS    ΔSNAVI     ΔSPGPRT    ΔSTGID
    ΔSUBLST     TA         TABPRT    ΔTAEM       TAPEND    ΔTDER31     TERMPT
     TGUESS     TIMEPF     TIMESF    ΔTOPBLK     TOPODT     TPSWR      TRACE
     TRACER    ΔTRIM       TRNDER     TRNRES     TRNR31    ΔVARMAS     VECBLK
    ΔVPRINT    ΔVRA71     ΔVRA73C    ΔVRA73H    ΔWAIT30     WRITEX    ΔXDATE
     ZEROIT
```

```
ERROR LOCATION REPORT

DIVIDE FAULT AT ADDRESS 055754 AT RELATIVE ADDRESS
000671 OF ELEMENT HTRATE       IN SEGMENT E
```

```
 S E G M E N T   L O A D I N G   R E P O R T

SEGMENT MAIN          LOADED        1 TIMES TOTAL


SEGMENT PHSSEG        LOADED        1 TIMES TOTAL
          1 LOADS BY CALL TO SUBROUTINE LOAD

SEGMENT DYNSEG        LOADED        1 TIMES TOTAL
          1 LOADS BY CALL TO SUBROUTINE PHSEXC

SEGMENT ORBSEG        LOADED        0 TIMES TOTAL


SEGMENT COASEG        LOADED        0 TIMES TOTAL


SEGMENT TARGET        LOADED        0 TIMES TOTAL


SEGMENT INTI          LOADED        1 TIMES TOTAL
          1 LOADS BY CALL TO SUBROUTINE INTEGI

SEGMENT INTII         LOADED        1 TIMES TOTAL
          1 LOADS BY CALL TO SUBROUTINE INTGII

SEGMENT INTA          LOADED        0 TIMES TOTAL


SEGMENT INTIA         LOADED        0 TIMES TOTAL


SEGMENT MMDSEG        LOADED        1 TIMES TOTAL
          1 LOADS BY CALL TO SUBROUTINE DYNAMC

SEGMENT E             LOADED        1 TIMES TOTAL
          1 LOADS BY CALL TO SUBROUTINE MMDE
```

```
SEGMENT L              LOADED        0 TIMES TOTAL

SEGMENT O              LOADED        0 TIMES TOTAL

SEGMENT S              LOADED        0 TIMES TOTAL

SEGMENT GE             LOADED        0 TIMES TOTAL

SEGMENT GN             LOADED        0 TIMES TOTAL

SEGMENT GO             LOADED        0 TIMES TOTAL

SEGMENT GA             LOADED        0 TIMES TOTAL

SEGMENT ITRSEG         LOADED        0 TIMES TOTAL

SEGMENT MCOSEG         LOADED        0 TIMES TOTAL

END CUAS POST-PROCESSOR
CHARGES FOR CUAPSTPRO EXECUTION IN SUP SECONDS
CAU =    4.962 CC/ER =    1.841 I/O =    2.784

@ FIN
```

```
RUNID: PHLXIA    ACCT: 3421-E255-C     PROJECT: FML-L7935
TIME:      TOTAL: 00:04:13.088     CHARGE: 00:04:13.088
           CAU:   00:01:35.751     I/O:    00:01:34.560
           CC/ER: 00:01:02.777     WAIT:   00:00:01.665
IMAGES  READ:     364        PAGES:  140
START:     20:41:10 AUG 12,1976     FIN:  21:32:04 AUG 12,1976
```

APPENDIX A

EXEC 8 STANDARD UNIT OF PROCESSING CHARGES AT NASA/JSC

A.1 APPLICATION

This appendix covers the technique used by EXEC 8 at NASA/JSC for maintaining Standard Unit of Processing charges for an application program. The accumulation of these charges is the basis for charging application programs for the computer resources needed during their execution. The data presented in this appendix is EXEC 8 installation dependent.

A.2 DEFINITIONS

SUP Standard Unit of Processing, a unit devised to provide a consistent measure of computer service as viewed by an application program running under EXEC 8.

PCT Program Control Table, a table maintained by EXEC 8 and containing control information about a specific job running under EXEC 8.

CAU Control/Arithmetic Unit, the UNIVAC hardware equivalent to a Central Processing Unit (CPU).

ER/CC Executive Request/Control Card, a designation for a computer service which the operating system performs for the user directly at his request.

I/O Input/Output, a computer service in which data is moved between primary storage and a peripheral device.

A.3 GENERAL DESCRIPTION

An application program running under EXEC 8 may accrue computer service charges in any or all of three categories which are charges for CAU, charges for ER/CC, and charges for I/O. The charges in each category are expressed in a common unit called a SUP second. EXEC 8 maintains the SUP charges accrued for a job in the job's PCT, which the job may reference but not change. A user job may reference his PCT by performing an executive request to the EXEC 8 entry point PCT$, or by basing the PCT as a read-only D-BANK by means of the LDJ instruction, both of which are documented in U. P. 4144, Revision 3. The total SUP charges for a job are calculated by independently determining the charges for each of CAU, ER/CC, and I/O and then adding the three charge values together. The CAU time is determined differently on 1108 and 1110 hardware but the ER/CC and I/O times are determined in the same way on both the 1108 and 1110.

A.4 CAU TIME ON UNIVAC 1110

When configured on UNIVAC 1110 hardware, EXEC 8 maintains a storage reference counter for primary storage in PCT location 118 and for extended storage in PCT location 119. The SUP time is calculated from these counters as indicated by the following equation.

$$T_{CAU} = (S_P + S_e) * 7/3200$$

where

T_{CAU} = SUP charge for CAU in 200 microsecond increments

S_P = Primary storage reference counter from PCT cell 118

S_e = Extended storage reference counter from PCT cell 119

A.5 CAU TIME ON UNIVAC 1108

When configured on UNIVAC 1108 hardware, EXEC 8 maintains CAU time in cell 22 of the PCT. The time value in this cell is the SUP charge for CAU in 100-microsecond increments. The SUP time is calculated from the following equation.

$$T_{CAU} = P_t/2$$

where

T_{CAU} = SUP charge for CAU in 200-microsecond increments

P_t = The SUP charge value from PCT cell 22

A.6 ER/CC TIME

On both the 1108 and 1110, EXEC 8 maintains the ER/CC time in cell 134 of the PCT. The time value in this cell is the SUP charge for ER/CC in 200-microsecond increments, and the SUP time is simply calculated from the following equation.

$$T_{ER/CC} = E_t$$

where

$T_{ER/CC}$ = SUP charge for ER/CC in 200-microsecond increments

E_t = The SUP charge value from PCT cell 134

A.7 I/O TIME

On both the 1108 and 1110, EXEC 8 maintains I/O charges in an identical fashion in the PCT. The I/O time is maintained in cells 124 through 133 of the PCT in 10 device group reference counter cells, and the time charges for each device group is the product of the device group counter and a group rate factor as indicated in the following equation.

$$DT_i = IOG_J * GF_i$$

where

DT_i = Device group SUP charges in 100-nanosecond increments for i=1, 10

IOG_j = The device group factor from the PCT, where j varies from PCT cell 124 to 133

GF_i = The device group rate factor, for i=1, 10. This factor is defined in the following table.

i	Factor	Description
1	22	Transfer rate for extended storage
2	82	Transfer rate for 432 drum
3	82	Transfer rate for 1782 drum
4	229	Transfer rate for 8440 disk
5	140	Transfer rate for 8414 disk
6	200	Transfer rate for U16C tape drive
7	229	Transfer rate for 8460 disk
8	625	Transfer rate for U8C tape drive
9	700	Transfer rate for U12C tape drive
10	3000	Transfer rate for all other devices

The transfer rates for each factor are in 100-nanosecond increments. The total I/O time is calculated from the following equation.

$$T_{I/O} = \left(\sum_{i=1}^{10} DT_i\right)/2000$$

where

$T_{I/O}$ = SUP charges for I/O in 200-microsecond increments

DT_i = Device group SUP charges in 100-nanosecond increments

A.8 TOTAL SUP CHARGES

The total SUP charges for a job may be calculated by the following equation.

$$T_{SUP} = T_{CAU} + T_{ER/CC} + T_{I/O}$$

where

T_{SUP} = Total SUP charges in 200-microsecond increments

T_{CAU} = Total CAU SUP charges in 200-microsecond increments

$T_{ER/CC}$ = Total ER/CC SUP charges in 200-microsecond increments

$T_{I/O}$ = Total I/O SUP charges in 200-microsecond increments

APPENDIX B

TEST CASE ELEMENT FORMAT

B.1 APPLICATION

This appendix covers the content and format of the test case usage source element as created by the Code Usage Analysis System (CUAS) postprocessor. The element includes the names of subroutine elements called and not called for the execution of a specific set of test case data by an application program.

B.2 GENERAL DESCRIPTION

The test case element is created as a source element in a UNIVAC program file with the UNIVAC program file maintenance package SOR utilized to generate the element. The element is created in the internal file name 'DBF' with the element name assigned by the user and the version name of 'TESTCASE'. The first record in the element is the text 'SUBROUTINES USED BY TEST CASE XXXXXXXX' where XXXXXX represents the element name supplied by the user. The next n records of the element each contain the name, date, and time of creation of a subroutine element used by the test case, where n is the number of subroutines used. Following the n records of subroutines used is a record with the character '*' in positions 1-36 and then a record with the text 'SUBROUTINES NOT USED BY TEST CASE XXXXXXXX' where XXXXXXXX represents the element name supplied by the user. The next j records of the element each contain the name, date, and time of creation of a subroutine element not used by the test case where j is the number of subroutines not used. Following the j records of subroutines not used is a record with the character '*' in positions 1-36. The format of the test case element is depicted in figure B-1.

Record no.	Description
1	SUBROUTINE USED BY TEST CASE XXXXXXXX
2	SUBNAM 1 MM/DD/YY HH:MM:SS
3	SUBNAM 2 MM/DD/YY HH:MM:SS
.	
.	
.	
n	SUBNAMN MM/DD/YY HH:MM:SS
n+1	******************************
n+2	SUBROUTINES NOT USED BY TEST CASE XXXXXXXX
n+3	SUBNAM1 MM/DD/YY HH:MM:SS
n+4	SUBNAM2 MM/DD/YY HH:MM:SS
.	
.	
.	
n+j	SUBNAMJ MM/DD/YY HH:MM:SS
n+j+1	*********************** 36*

where

XXXXXXXX = User-supplied name

MM = Month

DD = Day

YY = Year

HH = Hours

MM = Minutes

SS = Seconds

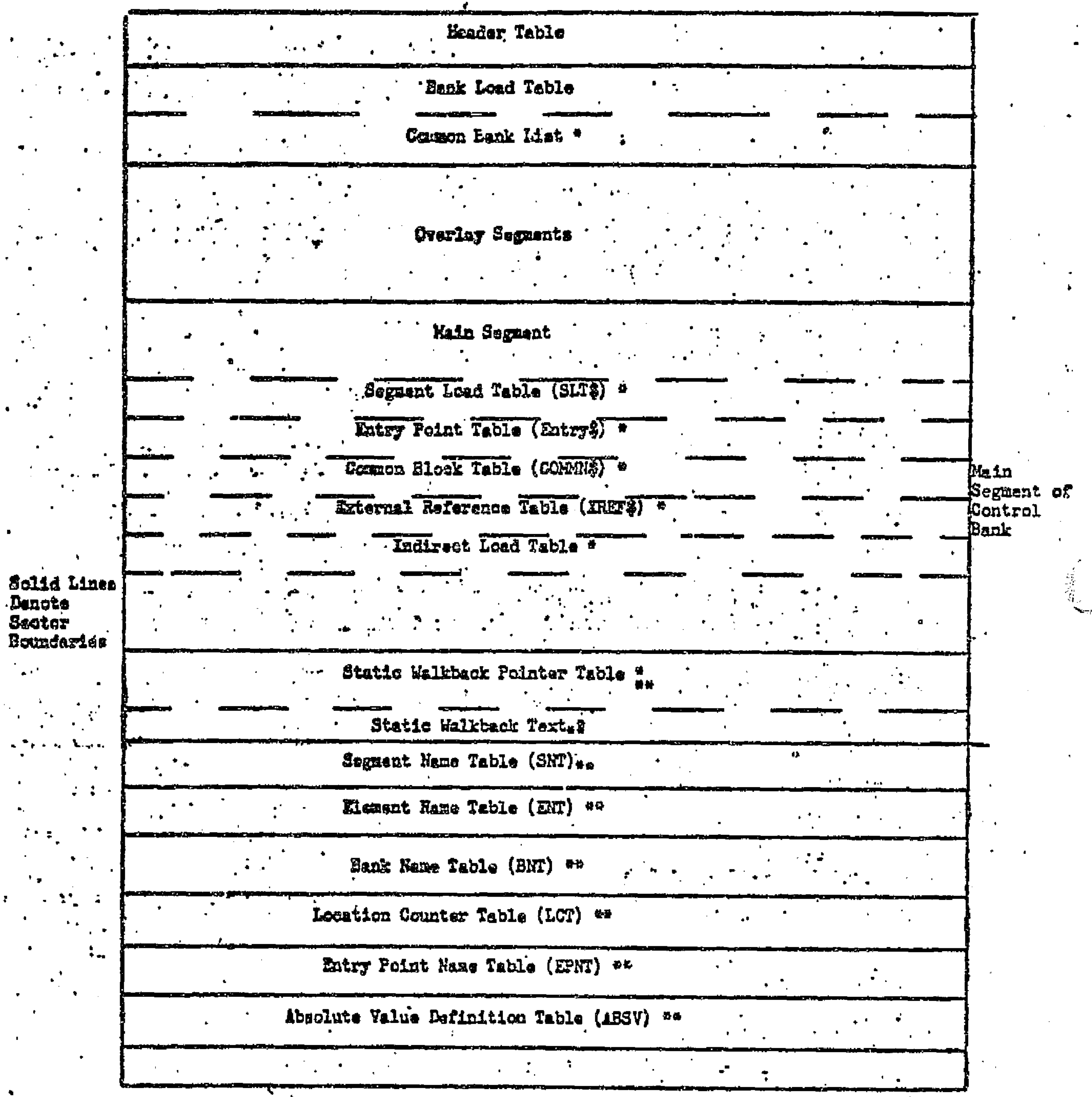
Figure B-1.- Test case element format.

APPENDIX C

UNIVAC EXEC 8 ABSOLUTE ELEMENTS

This appendix covers the format of an absolute element as produced by the UNIVAC 1100 series Memory Allocation Processor (MAP), level 24 or later, and is compatible only with EXEC 8 level 30 or later. The format presented is that of an absolute element in an EXEC 8 program file as placed there by the UNIVAC MAP. The presentation assumes a general knowledge of the function and use of the UNIVAC MAP and EXEC 8 program files as presented in UNIVAC Publication 4144, Revision 3. The remainder of this appendix is a presentation of the format of an absolute element as it is produced on the EXEC 8 systems in use at NASA/JSC as of this writing.

Absolute Element Format


Header Table
Bank Load Table
Common Bank List *
Overlay Segments
Main Segment
Segment Load Table (SLT$) *
Entry Point Table (Entry$) *
Common Block Table (COMMN$) *
External Reference Table (XREF$) *
Indirect Load Table *
Static Walkback Pointer Table *
**
Static Walkback Text**
Segment Name Table (SNT)**
Element Name Table (ENT) **
Bank Name Table (BNT) **
Location Counter Table (LCT) **
Entry Point Name Table (EPNT) **
Absolute Value Definition Table (ABSV) **
Main
Segment of
Control
Bank
Solid Lines
Denote
Sector
Boundaries

* Table may or may not be present
** Table present unless e option on map

Header Table Format

Ø	Sentinel (Ø44444444444)					
1	E	Ø	BDI-Initial PSRMI	E	Ø	BDI-Initial PSRMD
2	E	Ø	BDI-Initial PSRUI	E	Ø	BDI-Initial PSRUD
3	Ø					
4	E	Ø	BDI-Control Bank	Program Start Address		
5	Sector Address of Main Segment			ACWs for Initially Loaded Banks		
6	No. of 4 Word SLT Entries			Control Bank SLT Length		
7	Min. Absolute Addr for (H2)			Initial I Size in PS (Storage Blocks)		
Ø1Ø	*			Initial I Size - No Preference		
Ø11	*			Initial I Size In ES		
Ø12	*			Initial D Size In PS		
Ø13	*			Initial D Size - No Preference		
Ø14	*			Initial D Size in ES		
Ø15	Time and Date of Absolute Element Creation					
Ø16	Fieldata RLIB$ ID					
Ø17	Options on @MAP in Master Bit Notation					
Ø2Ø	Sector Address of Diagnostic Tables					
Ø21	Walkback Pointer Table Word Length			Control Bank SLT$ Address		
Ø22	No. of Entry$ Entries			Control Bank Entry$ Address		
Ø23	No. of COMMN$ Entries			Control Bank COMMN$ Address		
Ø24	No. of XREF$ Entries			Control Bank XREF$ Address		
Ø25	No. of Indirect Load Table Entries			Control Bank ILL Table Address		
Ø26	Word Length of LCT			Word Length of BNT		
Ø27	Word Length of SNT			Word Length of ENT		
Ø3Ø	Word Length of EPNT			Word Length of Walkback Text		
Ø31	Word Length of ABSV			Word Length of BLT		
Ø32	C	Total Mass Storage Word Length of Element				
Ø33	Ø					

E - Set if the BDI is for an EXEC BDT	ENT - Element Name Table
SLT - Segment Load Table	EPNT - Entry Point Name Table
LCT. - Location Counter Table	ABSV - Absolute Value Table
BNT - Bank Name Table	BLT - Bank Load Table
SNT - Segment Name Table	C - Set by Partial Checkpoint

Bank Load Table Format

BDI		SECTOR ADDR. OF BANK'S MAIN SEGMENT ACWS	
TYPE	PREF.	BLOCK SIZE	SLR LOWER (12 BIT FIELD)

Type bits are set as follows:

Bit 35 - BANK IS DYNAMIC

Bit 34 - BANK IS A DBANK

Bit 33 - WRITE PROTECT SET FOR BANK

PREF. is as follows:

0 - Requires Primary Storage

1 - Requires Extended Storage

2 - Prefers Primary Storage

3 - Prefers Extended Storage

4 - No Storage Preference

A. There is one BLT entry for each bank in the program. Entries are ordered by ascending BDI beginning with BDI 4.

B. For bank implied collections, the BLT always contains 2 entries, with a BDI = 4 for the IBANK and BDI = 5 for the DBANK.

C. For V-option banks (assign addresses but strip code), block size - ∅.

D. BLT is immediately followed by the common bank list if it exists.

Common Bank List Format

FIRST BDI	SECOND BDI	THIRD BDI

This table contains the BDI for the EXEC BDT of all COMMON BANKS referenced within the program.

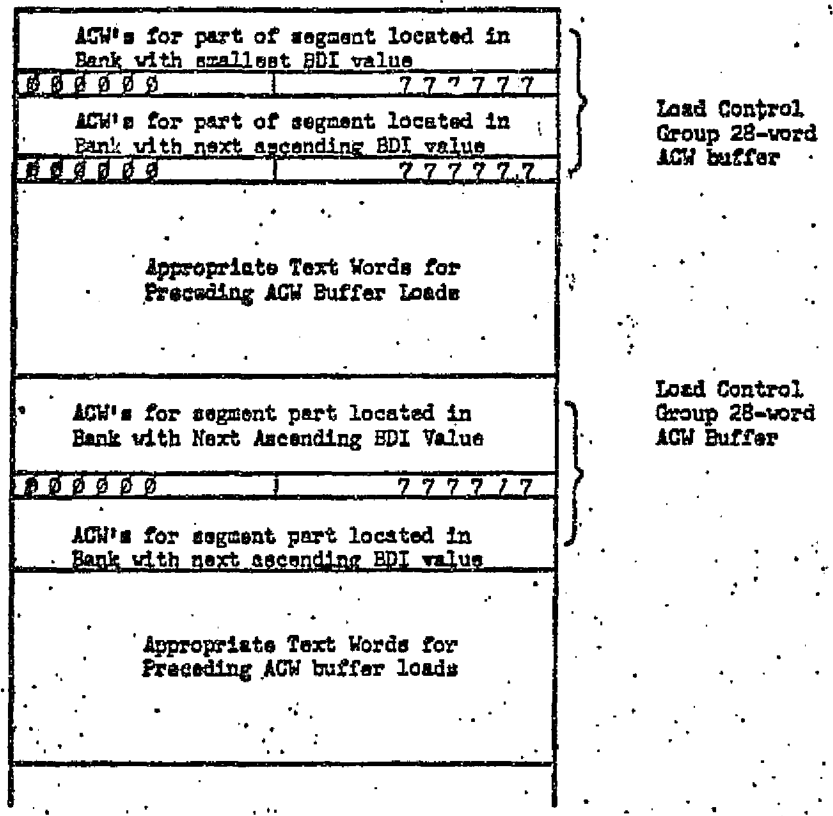
BDIs of COMMON BANKS specified for initial basing will be included within this list.

T2 of word 6 of the ABSOLUTE ELEMENT TABLE contains the number of third word values contained in this list.

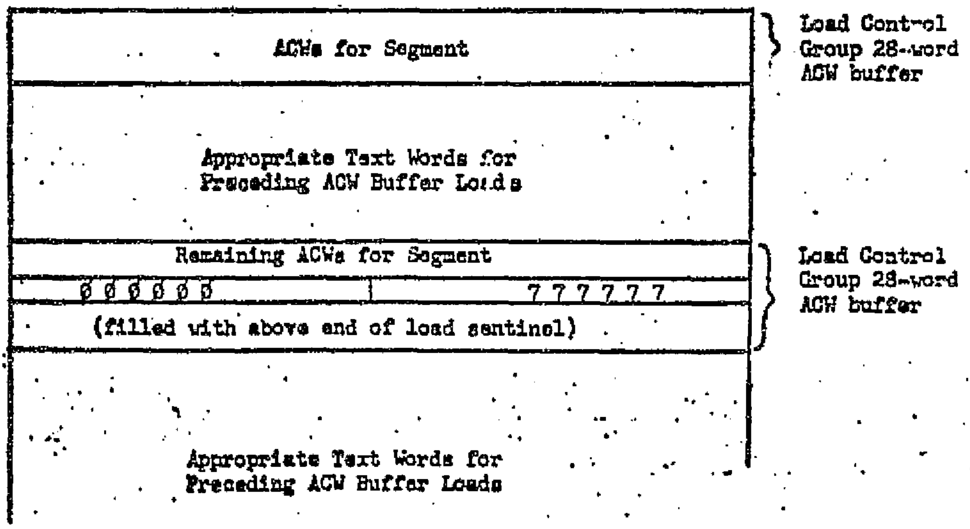
This table immediately follows the BANK LOAD TABLE.

Text and ACW Formats

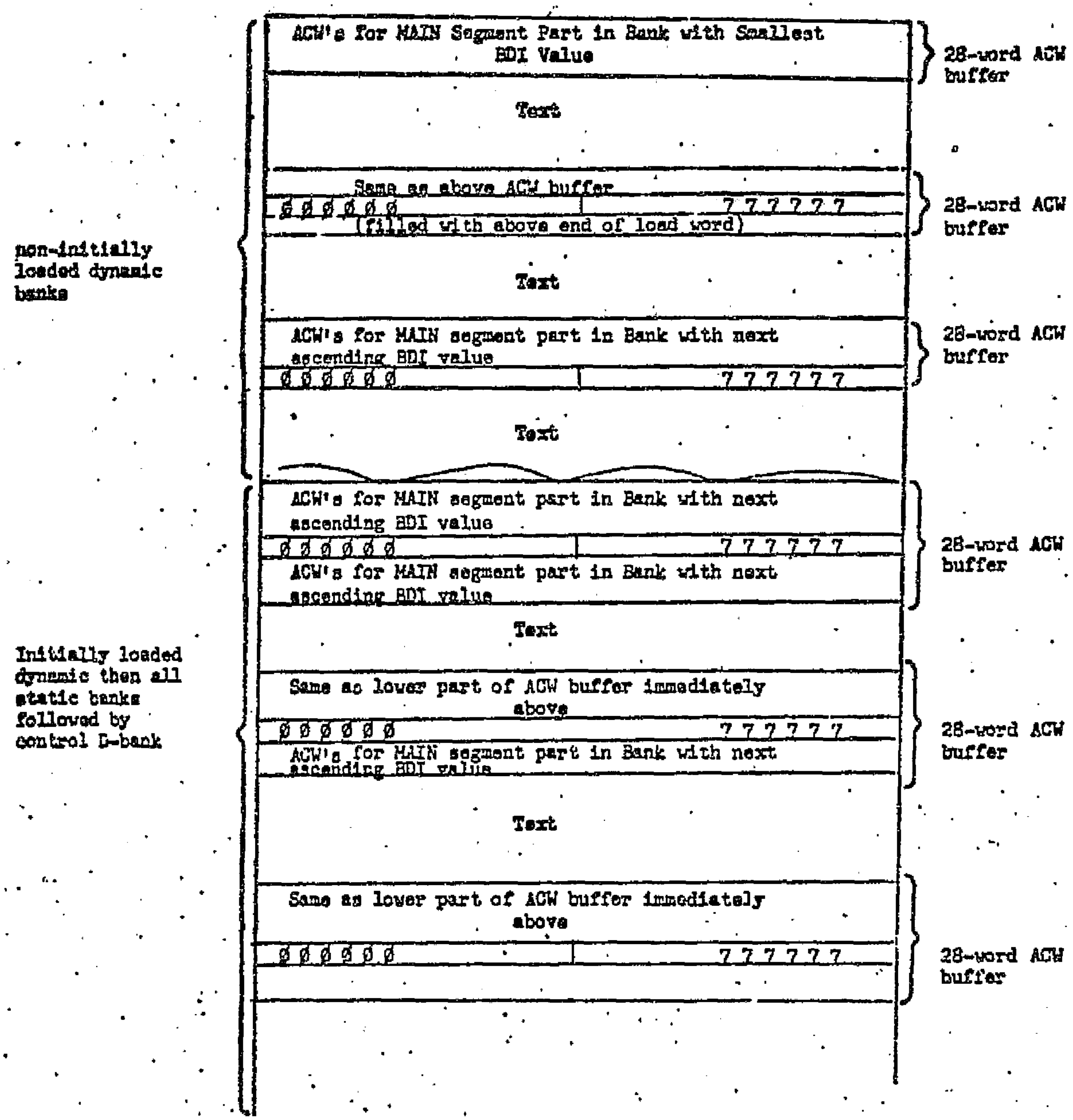
Absolute element text format for overlay segments which span banks.


ACW's for part of segment located in Bank with smallest BDI value
ØØØØØØ
777777
ACW's for part of segment located in Bank with next ascending BDI value
ØØØØØØ
777777
Load Control Group 28-word ACW buffer
Appropriate Text Words for Preceding ACW Buffer Loads
ACW's for segment part located in Bank with Next Ascending BDI Value
ØØØØØØ
777777
ACW's for segment part located in Bank with next ascending BDI value
Load Control Group 28-word ACW Buffer
Appropriate Text Words for Preceding ACW buffer loads

Absolute Element Text Format for Overlay Segments which do not span Banks


ACWs for Segment
Load Cont-ol Group 28-word ACW buffer
Appropriate Text Words for Preceding ACW Buffer Loads
Remaining ACWs for Segment
ØØØØØØ
777777
(filled with above end of load sentinel)
Load Control Group 28-word ACW buffer
Appropriate Text Words for Preceding ACW Buffer Loads

Absolute Element Text Format for the MAIN Segment


ACW's for MAIN Segment Part in Bank with Smallest BDI Value
28-word ACW buffer
Text
Same as above ACW buffer
Ø Ø Ø Ø Ø Ø
7 7 7 7 7 7
(filled with above end of load word)
28-word ACW buffer
non-initially loaded dynamic banks
Text
ACW's for MAIN segment part in Bank with next ascending BDI value
Ø Ø Ø Ø Ø Ø
7 7 7 7 7 7
28-word ACW buffer
Text
ACW's for MAIN segment part in Bank with next ascending BDI value
Ø Ø Ø Ø Ø Ø
7 7 7 7 7 7
ACW's for MAIN segment part in Bank with next ascending BDI value
28-word ACW buffer
Text
Initially loaded dynamic then all static banks followed by control D-bank
Same as lower part of ACW buffer immediately above
Ø Ø Ø Ø Ø Ø
7 7 7 7 7 7
ACW's for MAIN segment part in Bank with next ascending BDI value
28-word ACW buffer
Text
Same as lower part of ACW buffer immediately above
Ø Ø Ø Ø Ø Ø
7 7 7 7 7 7
28-word ACW buffer

Note that the only case where ACW's for a segment spanning banks are not packed within any ACW buffer is when the ACW's are for the MAIN segment of a non-initially loaded dynamic bank. That is, the first ACW for a non-initially loaded dynamic bank's MAIN segment will always start on a sector boundary, as will its first text word.

ACW Formats

Normal ACW

NO. OF WORDS TO LOAD	PROGRAM START ADDR. FOR LOAD

Above is normal ACW format.

Zero Fill ACW

35	34		
1	0	NO. OF WORDS TO ZERO FILL	START ADDRESS

Above ACW is used to zero fill areas of core into which no text is initially loaded.

Each BANK's main segment ACWs will contain a zero fill ACW to fill core with zeroes from the end of the main segment to the end of the last core block assigned to that bank.

End-of-Load Sentinel

ø	777777

Above ACW is an end-of-load sentinel. It marks the end of the set of normal and zero fill ACWs for each segment part within a bank.

RSEG Relocation-Bits-ACW

NO. OF RELOCATION WORDS	ø

Above ACW is for the relocations bits for an RSEG and immediately follows the end-of-load sentinel for the actual RSEG text ACWs.

NOP ACW

35	34	
ø	1	ø

Above ACW is a NOP ACW. It is recognized and bypassed by the loader.

NOP Common Block ACW

35	34		
0	1		BDI

Above ACW is a NOP Common Block ACW. H2 contains the BDI of the bank into which common block text is to be loaded. The ACWs immediately following the NOP Common Block ACW specify the locations where the common block text is loaded within the indicated bank. An end-of-load sentinel will be found following the last ACW for the common block text load.

Segment Load Table Entry Formats

Non-Extended SLT Entry Format

Ø	A	TYPE	Ø	FORWARD LINK TO ACTIVE SEGMENT
1	LAST I-BANK ADDRESS			FIRST I-BANK ADDRESS
2.	LAST D-BANK ADDRESS			FIRST D-BANK ADDRESS
3	Ø		SECTOR ADDR OF FIRST LOAD CONTROL GROUP	

A: Bit 35 set if segment is not loaded.

TYPE: ØØØ main segment for non-extended SLT format.

ØlØ dynamic segment

Ø27 relocatable segment

Ø24 overlay segment

If the S2 value of word 0 of the first SLT entry (SLTØ) is equal zero, the table only contains four word entries as formatted above.

If S2 of word Ø = Ø22 in the first SLT entry, the table format contains extension entries, in addition to four word entries.

Extended SLT Entry Format

0	A	TYPE	Ø	FORWARD LINK TO ACTIVE SEGMENT
1	LAST BANK ADDRESS			FIRST BANK ADDRESS
2	BDI		Ø	EXTENSION INDEX
3	Ø		SECTOR ADDR OF FIRST LOAD CONTROL GROUP	

A: Bit 35 set if segment is not loaded

TYPE: Ø22 main segment for extended SLT format.

Ø11 dynamic segment

Ø27 relocatable segment

Ø24 overlay segment

BDI: BDI value for bank into which segment part is loaded

When H2 word 2 is non-zero, it contains a link to the next SLT extension for the segment. Note: The index points to the word immediately preceeding the first word of the extension entry.

Index

ø	LAST BANK ADDRESS		FIRST BANK ADDRESS
1	EDI	ø	EXTENSION INDEX

Each 4 word entry and its extensions are linked in order of ascending EDI value.

The SLT 4 word entry for the main segment never contains an extension index as no extension entries are built for the main segment.

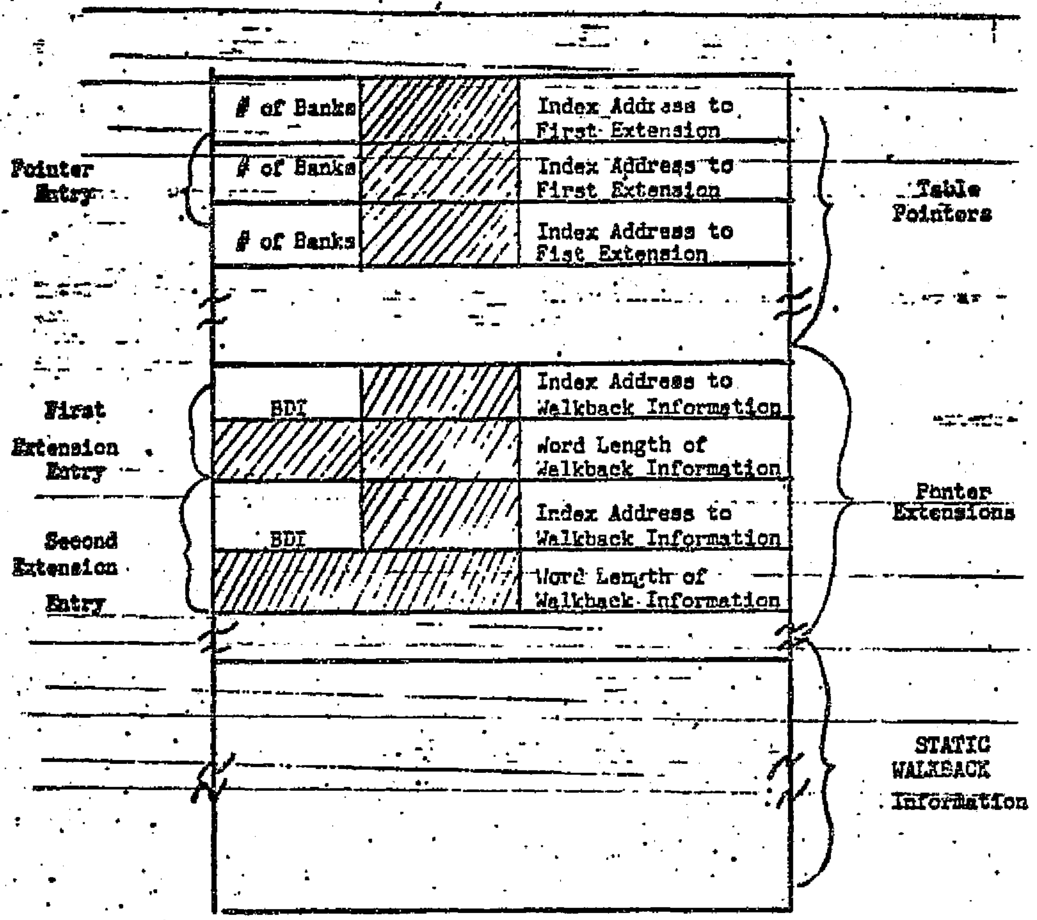
RSEG SLT Entry Format

For RSEG SLT entries, the format is the same in both the non-extended and extended segment load table. The format is as follows:

A	ø27	ø	FORWARD LINK TO ACTIVE SEGMENT
LAST RSEG ADDRESS			ø
ø			NO. OF RELOCATION WORDS
ø		SECTOR ADDR OF FIRST LOAD CONTROL GROUP	

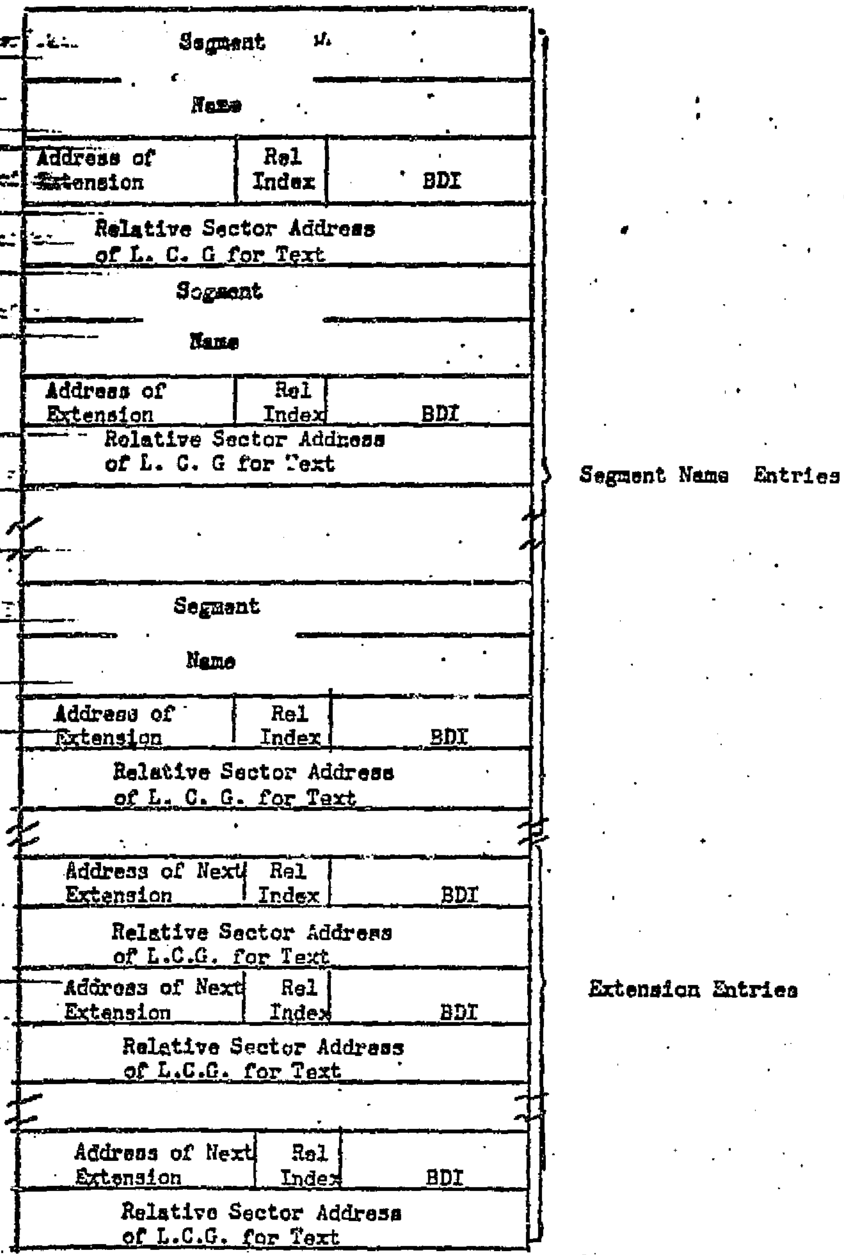
Diagnostic Table Formats

Static Walkback Pointer Table and Walkback Information


# of Banks
Index Address to First Extension
# of Banks
Index Address to First Extension
# of Banks
Index Address to Fist Extension
BDI
Index Address to Walkback Information
Word Length of Walkback Information
BDI
Index Address to Walkback Information
Word Length of Walkback Information
Pointer Entry
First Extension Entry
Second Extension Entry
Table Pointers
Ponter Extensions
STATIC WALKBACK Information

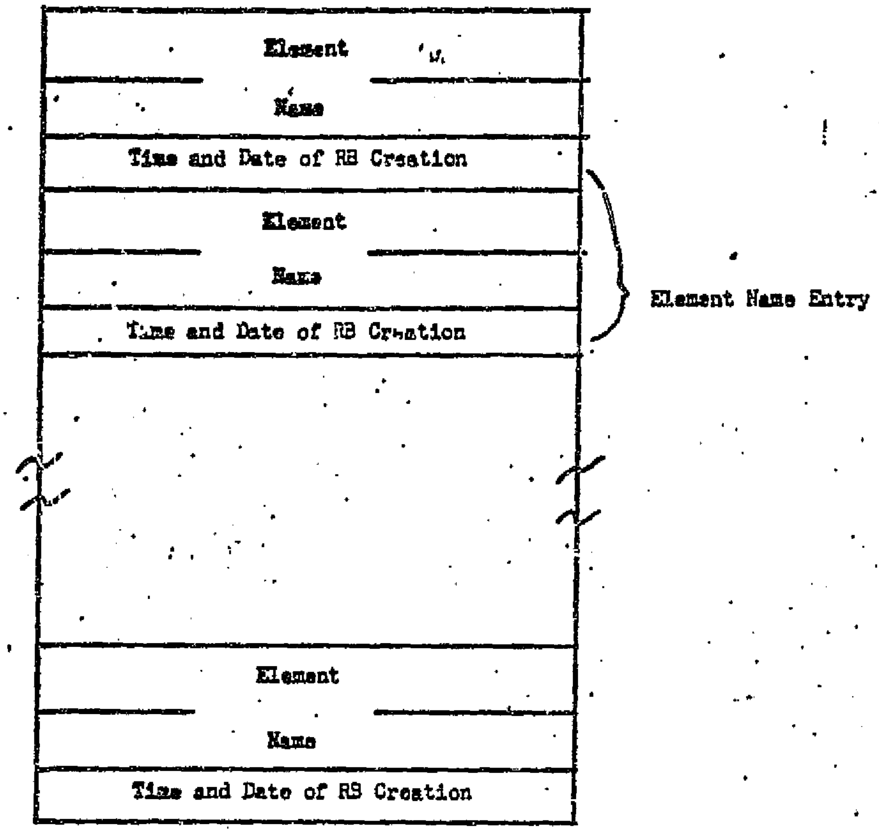
The pointer table is organized by ascending segment index. All index addresses are relative to the start table.

Segment Name Table (SNT)


Segment
Name
Address of Extension
Rel Index
BDI
Relative Sector Address of L. C. G for Text
Segment
Name
Address of Extension
Rel Index
BDI
Relative Sector Address of L. C. G for Text
Segment
Name
Address of Extension
Rel Index
BDI
Relative Sector Address of L. C. G. for Text
Segment Name Entries
Address of Next Extension
Rel Index
BDI
Relative Sector Address of L.C.G. for Text
Address of Next Extension
Rel Index
BDI
Relative Sector Address of L.C.G. for Text
Extension Entries
Address of Next Extension
Rel Index
BDI
Relative Sector Address of L.C.G. for Text

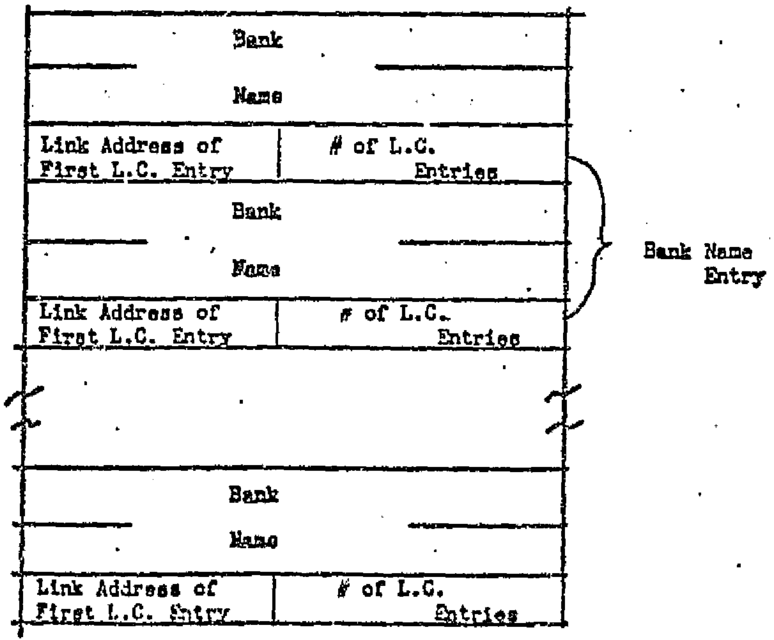
Rel Index - Word in the Load Control Group where this segment description begins.

Element Name Table (ENT)


Element
Name
Time and Date of RB Creation
Element
Name
Time and Date of RB Creation
Element Name Entry
Element
Name
Time and Date of RB Creation

Time and Date of RB Creation - The time and date when the Relocatable Binary (RB) element was created.

Bank Name Table (BNT)


Bank
Name
Link Address of First L.C. Entry
# of L.C. Entries
Bank
Name
Link Address of First L.C. Entry
# of L.C. Entries
Bank Name Entry
Bank
Name
Link Address of First L.C. Entry
# of L.C. Entries

Location Counter Table Format

<table>
<tr><td>Ø</td><td colspan="6">SEGMENT INDEX</td><td colspan="2">ELEMENT INDEX</td></tr>
<tr><td>1</td><td>D</td><td>L</td><td>C</td><td>X</td><td>R</td><td></td><td>LC #</td><td>BDI</td></tr>
<tr><td>2</td><td colspan="6">L. C. WORD LENGTH</td><td colspan="2">L.C. PROGRAM START ADDRESS</td></tr>
<tr><td>Ø</td><td colspan="6">SEGMENT INDEX</td><td colspan="2">ELEMENT INDEX</td></tr>
<tr><td>1</td><td>D</td><td>L</td><td>C</td><td>X</td><td>R</td><td></td><td>LC #</td><td>BDI</td></tr>
<tr><td>2</td><td colspan="6">L. C. WORD LENGTH</td><td colspan="2">L. C. PROGRAM START ADDRESS</td></tr>
<tr><td></td><td colspan="8"></td></tr>
<tr><td>Ø</td><td colspan="6">SEGMENT INDEX</td><td colspan="2">ELEMENT INDEX</td></tr>
<tr><td>1</td><td>D</td><td>L</td><td>C</td><td>X</td><td>R</td><td></td><td>LC #</td><td>BDI</td></tr>
<tr><td>2</td><td colspan="6">L. C. WORD LENGTH</td><td colspan="2">L. C. PROGRAM START ADDRESS</td></tr>
</table>

D-SET IF LC IS IN A DBANK

L - SET IF LC IS FROM AN RLIB$ ROUTINE

C - SET IF LC TEXT IS FOR A COMMON BLOCK

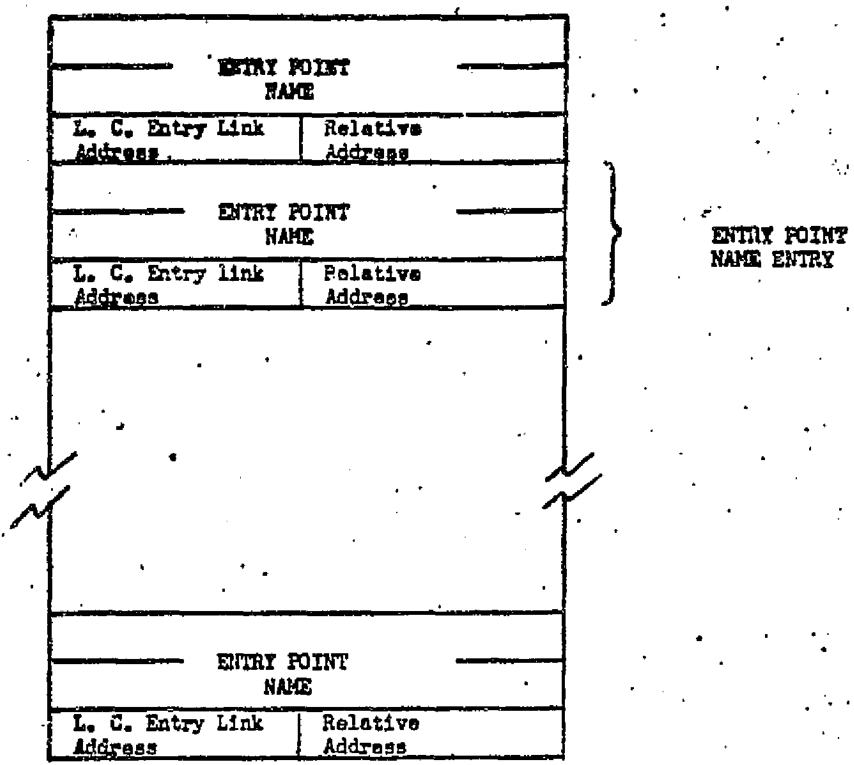
X - SET IF LC IS THE ACTUAL COMMON BLOCK

R - SET IF LC TEXT IS IN AN RSEG

If C is set, H2 of word 2 contains an index to the LCT entry for the final LC assigned to the common block in the absolute element.

If X is set, the LC # is Ø.

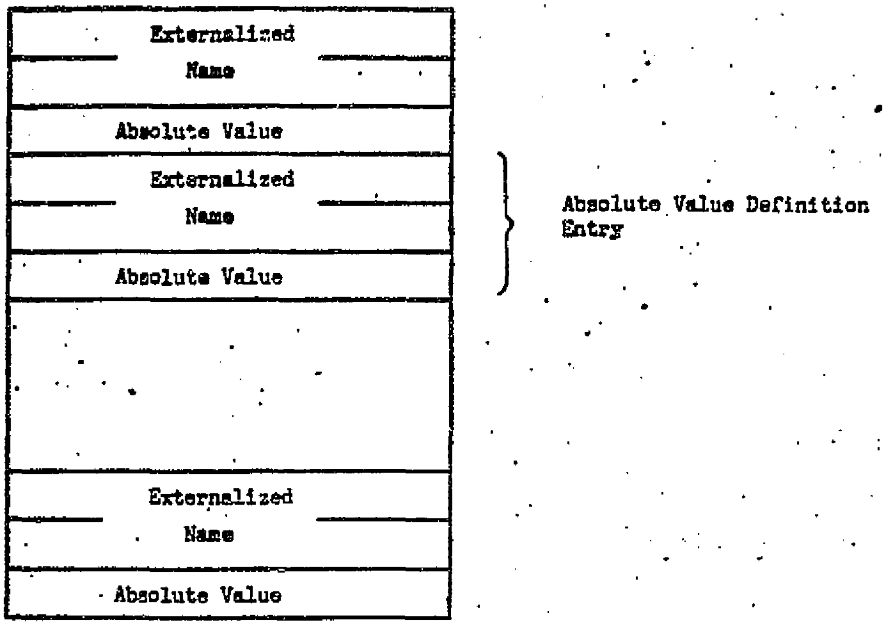
Entry Point Name Table (EPNT)



L. C. Entry Link Address - Address of Location Counter Entry which includes this Entry Point.

Relative Address - The address of the Entry Point relative to the start of the Location Counter.

Absolute Value Definition Table (ABSV)



APPENDIX D

CODE USAGE ANALYSIS SYSTEM JUMP HISTORY STACK FILE

D.1 APPLICATION

This appendix covers the content and format of the Jump History Stack File (JHSF) as created by the Code Usage Analysis System (CUAS) contingency subroutine IICONT. This file is used as a data input file by the CUAS postprocessor program in order to produce reports concerning the execution characteristics of an application program executed under the UNIVAC EXEC 8 operating system.

D.2 DEFINITIONS

CAU	Control/Arithmetic Unit, the UNIVAC hardware equivalent to a Central Processing Unit (CPU)
CELL	36 binary digits treated as a unit of primary storage and directly addressable by the CPU as a unit
CUAS	Code Usage Analysis System
ER/CC	Executive Request/Control Card, a designation for a computer service which the operating system performs for the user directly at his request
H1	The most significant 18 bits of a cell, or bits 18-35 of the cell
H2	The least significant 18 bits of a cell, or bits 0-17 of the cell
I/O	Input/Output, a computer service in which data is moved between primary storage and a peripheral device
J	A mnemonic for a valid UNIVAC-1100 processor instruction which transfers control to a specified address
JHSF	Jump History Stack File
LMJ	A mmemonic for a valid UNIVAC-1100 processor instruction which saves the current value of the P register and transfers control to a specific address

$(nnnnnn)_8$ The representation used for an octal natural number, that is, a number whose base is 8

Q1 The most significant 9 bits of H1 of a cell, or bits 27-35 of the cell

Q2 The least significant 9 bits of H1 of a cell, or bits 18-26 of the cell

Q3 The most significant 9 bits of H2 of a cell, or bits 9-17 of the cell

Q4 The least significant 9 bits of H2 of a cell or bits 0-8 of the cell

SUP Standard Unit of Processing, a unit devised to provide a consistent measure of computer service as viewed by an application program running under EXEC 8

D.3 GENERAL DESCRIPTION

The Jump History Stack File (JHSF) is created as a UNIVAC EXEC 8 FASTRAND formatted mass storage file on a secondary storage device. A FASTRAND formatted file is one which is addressable by units of 28 cells which are defined as sectors. Such a file may be accessed by a basic I/O executive request, which is documented in UNIVAC Publication 4144, Revision 3, Chapter 6 or by the FORTRAN I/O routine NTRAN, which is documented in UNIVAC Publication 7876, Section 4.18.

D.4 JUMP HISTORY STACK FILE DESCRIPTION

The JHSF may be viewed as simply a sequential string of 36-bit cells occupying as many sequential and contiguous sectors as are required to contain the string. With the exception of the first seven cells of the string, each of the cells contains data indicating the occurrence of some event during the execution of an application program, and the sequence of the cells in the file indicates the sequence of event occurrence. The first seven cells of the file contain the following data:

Cell 1 - The first six fieldata characters of the absolute element name which when executed created this file

Cell 2 - The last six fieldata characters of the absolute element name

Cell 3 - The first six fieldata characters of the version name of the absolute element

Cell 4 - The last six fieldata characters of the version name

Cell 5 - The date and time of creation of the absolute element, formatted exactly as it appears in a program file

Cell 6 - The @XQT options present on the CUAS preprocessor execute card when the absolute element was modified. The options are represented in master bit notation, that is, bit 0 on for Z and bit 25 on for A

Cell 7 - The total number of sectors written in the JHSF, a 36-bit binary integer

Beginning with cell 8 of the file, each cell contains a sentinel in bits 34 and 35 which indicates the type of data represented by the remaining bits 0-33 of the cell. The two-bit sentinel is defined as follows:

Sentinel binary value	Description
00	The lower 16 bits of H1 of the cell contain the address of an LMJ instruction which was executed. H2 of the cell contains the address to which the LMJ instruction transferred control
01	The lower 16 bits of H1 of the cell contain the address of a J instruction which was executed. H2 of the cell contains the address to which the J instruction transferred control
10	Bits 0-33 of the cell contain a SUP time value
11	H1 of the cell contains a sentinel indicating the occurrence of an error event and H2 of the cell contains the description of the event

D.4.1 SUP TIME CELL INTERPRETATION

SUP time value cells (sentinel 10) are included in the JHSF only when option C was specified to the CUAS preprocessor, which may be determined by inspection of cell 6 of the JHSF. When this option is specified, each LMJ event cell (sentinel 00) and J event cell (sentinel 01) is followed by three SUP time cells. The three cells contain the amount of CAU, ER/CC, and I/O time, repectively, which was used since the last such occurrence in the JHSF of an LMJ or J cell followed by the three time cells. The CAU time is in microsecond increments, and the ER/CC and the I/O times are in 200-microsecond increments.

D.4.2 ERROR EVENT CELL INTERPRETATION

Error event cells (sentinel 11) are included in the JHSF for the indication of program errors, segment loads, and end of file. The total value of H1 of an error event cell indicates the event type, defined as follows:

H1 octal value	Description
$(777777)_8$	If H2 of the cell is also $(777777)_8$, this cell indicates the end of the file. If not, this is a segment load sentinel with Q4 of the cell containing the index of the segment loaded and Q3 of the cell containing the index of the segment loaded just prior to it.
$(777776)_8$	A guard mode sentinel with the address at which the guard mode occurred in H2 of the cell.
$(777775)_8$	An arithmetic overflow sentinel with the address at which the overflow occurred in H2 of the cell.
$(777774)_8$	An arithmetic underflow sentinel with the address at which the overflow occurred in H2 of the cell.
$(777773)_8$	An arithmetic divide fault sentinel with the address at which the divide fault occurred in H2 of the cell.
$(777772)_8$	An ER ABORT$ sentinel, that is, the user did an executive request to ABORT$ to terminate his program execution. The address at which the request was made is contained in H2 of the cell.

HI octal value	Description
$(777771)_8$	An ERROR MODE sentinel with the address at which the program was placed in error mode contained in H2 of the cell.

D.5 USE OF THE JHSF

The JHSF represents both events that occurred during a program execution and the sequence in which those events occurred. The amount of detail included in the file is determined by the way the subroutine IICONT is initialized. This initialization is normally performed by the CUAS preprocessor and is controlled by the user with execution options. J instruction and SUP time event cells are included in the JHSF only if option C was specified to the CUAS preprocessor. LMJ instruction cells are included in the JHSF unless option B was specified to the CUAS preprocessor. Error event cells are always included in the JHSF. A scan, normally done by the CUAS postprocessor to produce reports, allow determination of the execution path and characteristics of a program.

The JHSF is intended to be used in conjunction with the diagnostic tables from an absolute element as prepared by the UNIVAC MAP under EXEC 8. The content and format of an absolute element is described in Appendix C. The format of the JHSF is depicted in figure D-1.

CELL			
1	ABSOLUTE ELEMENT NAME		
2			
3	ABSOLUTE ELEMENT VERSION NAME		
4			
5	DATE/TIME OF ABSOLUTE ELEMENT CREATION		
6	CUAS PREPROCESSOR @XQT OPTIONS		
7	TOTAL NUMBER OF SECTORS S, IN FILE		
8			FIRST EVENT DESCRIPTOR CELL
9			SECOND EVENT DESCRIPTOR CELL
⋮			⋮
⋮			⋮
n	777777	777777	LAST EVENT DESCRIPTOR CELL, (E.O.F. SENTINEL)

Figure D-1.- Jump history stack file.

APPENDIX E

TEST CASE USAGE REPORTING SYSTEM

E1. INTRODUCTION

This document provides information related to the operation and use of the Test Case Usage Reporting System (TCURS). Descriptions of the system capabilities, method of operation, and user's information are included.

The TCURS is intended to aid the user in determining those subroutines used and not used within a program file from which several different absolute elements may be generated. Cross-reference reports are provided from which the user may determine for any specific absolute element those subroutines which were included in the absolute element and called at least once; or conversely, for any specific routine, those absolute elements in which it was included but not called at least once.

E2. TEST CASE USAGE REPORTING SYSTEM DESCRIPTION

E2.1 SYSTEM CAPABILITIES

The TCURS was designed to generate reports of all subroutines used by a set of test cases and consists of two separate programs, the File Relocatable Element Directory (FRED) program and the Test Case Report Generator (TCRGEN) program.

The FRED program analyzes a user's program file directory and inserts into a Data Base File (DBF) an EXEC 8 System Data Format (SDF) source element containing the names of all the user relocatable elements. The TCRGEN program produces the cross-reference reports of subroutine usage by test case. The Code Usage Analysis System (CUAS) is used to generate elements containing the names of the subroutines used by each test case.

Specifically, the FRED and TCRGEN programs provide the user the following information concerning the execution of his application programs.

E2.1.1 MASTER ELEMENT DIRECTORY REPORT

This report is generated when the FRED program is executed. The FRED program provides the user with a report containing the subroutine element names and version names, including creation date and time.

E2.1.2 TEST CASE ELEMENT REPORT

The test case element report which is generated by the TCRGEN program contains an alphabetized listing of all element names included in the master file directory. This is an exception report which is included to indicate to the user that the master file director is incomplete and should be updated.

E2.1.3 TEST CASE USAGE REPORT

The test case usage report, which is generated by the TCRGEN program, is displayed in matrix-like form with an alphabetized listing of each test case name, including creation date, and time, appearing horizontally at the top of the report page. The alphabetized element names including version, creation date, and time, are listed vertically on the left side of the report page. When an element was used by a particular test case, this is indicated by an "X" in the element name row and test case name column intersection. An asterisk is placed adjacent to and preceding the "X" to indicate that the master element directory time and date for the element does not coincide with the creation date and time for the element used by the test case.

E2.1.4 ELEMENT USAGE REPORT

The element usage report displays the test case usage by element name, alphabetized on the left side of the page; opposite are the names of all test cases in which the element was included and called at least once.

E2.2 METHOD OF SOLUTION

The TCURS maintains all test case and subroutine data as source elements in a UNIVAC EXEC 8 program file. Standard UNIVAC software is used to create and read these data elements. The use of this technique for maintaining the data allows for the use of standard EXEC 8 control statements to maintain the file and inspect the contents.

The following sections describe the reports generated by the FRED and TCRGEN programs.

E2.2.1 MASTER ELEMENT DIRECTORY REPORT

The master element directory report consists of a list of relocatable element names and version names, including creation date and time. The FRED program scans the user-specified EXEC 8 program file directory for relocatable element names and combines all such names and associated information into a source element. This source element becomes the master file directory and is inserted into a file with the internal name DBF. The FRED program then prints the report under the appropriate heading information.

E2.2.2 TEST CASE ELEMENT REPORT

The test case element report consists of an alphabetical list of all element names included in a test case but not included in the master file directory. The master file directory elements are obtained from the DBF. The test case elements are obtained from source elements created by the CUAS postprocessor, which are inserted into the DBF. The elements specified in the master file directory are sorted into alphabetical order and each element used by the test case and not included in the master file element directory is listed. The TCRGEN program then prints the report under the appropriate heading information.

E2.2.3 TEST CASE USAGE REPORT

The test case usage report consists of an alphabetized listing of each test case name, including creation date and time, appearing horizontally at the top of the report page. The alphabetized element names including version name, creation date, and time are listed vertically on the left side of the report page. The TCRGEN program scans the DBF directory for test case entries and sorts the names into alphabetical order. When an element was used by a particular test case, and the master element directory time and date are the same as those for the element used by the test case, this is indicated by an "X" in the element name row and test case name column intersection. An asterisk is placed adjacent to and preceding the "X" to indicate that the master element directory time and date for the element is not the same as the creation date and time for the element used by the test case.

E2.2.4 ELEMENT USAGE REPORT

The element usage report consists of an alphabetized list of element names. For each element name, the names of all test cases which called the element are listed in alphabetical order.

The test case usage report is prepared in the same manner as the test case element report except that the element usage report displays test case usage with the element name on the left side of the report page. The elements and test case names are listed under the appropriate heading.

The total operation for the FRED and TCRGEN programs is depicted in figure 1.

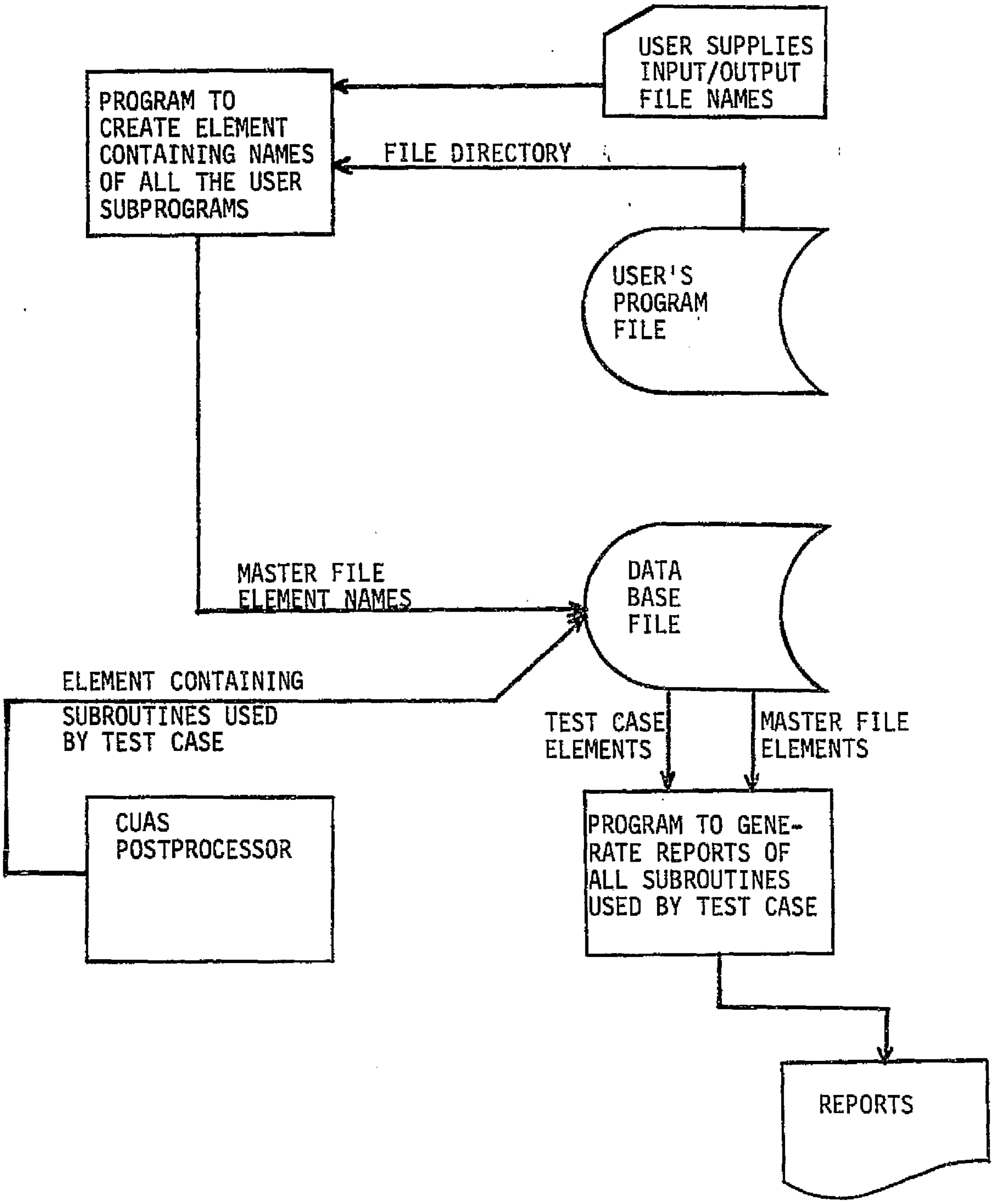

USER SUPPLIES INPUT/OUTPUT FILE NAMES
PROGRAM TO CREATE ELEMENT CONTAINING NAMES OF ALL THE USER SUBPROGRAMS
FILE DIRECTORY
USER'S PROGRAM FILE
MASTER FILE ELEMENT NAMES
DATA BASE FILE
ELEMENT CONTAINING SUBROUTINES USED BY TEST CASE
TEST CASE ELEMENTS
MASTER FILE ELEMENTS
CUAS POSTPROCESSOR
PROGRAM TO GENERATE REPORTS OF ALL SUBROUTINES USED BY TEST CASE
REPORTS

Figure 1. FRED and TCRGEN programs technique.

E3. USER'S GUIDE

E3.1 METHOD OF USE

The FRED and TCRGEN absolute elements have been implemented on the UNIVAC 1108, EXEC 8, processors 7 and 8, and may be accessed from the secure file FML-L79351*PHPA. The FRED program uses an internal file name, hereafter referred to as Relocatable Input (RIP), which is assigned to the user program file from which the master file directory is to be created.

The user must provide EXEC 8 @USE directives to indicate which file is to be scanned for relocatable elements (file RIP) and which file is to receive the source element created (file DBF). The TCRGEN program utilizes the master file directory created as described above in the generation of the TCRGEN reports. The data for the subroutines used and not used by the test case is obtained from source elements created by the CUAS postprocessor. The test case elements created by the CUAS must be in the same file with internal name DBF, as the master file directory. The method of use of the CUAS for the creation of test case elements is described in the CUAS user's guide and will not be detailed here. Before executing the TCRGEN, the user must create in a common program file the master file directory using the FRED program and one or more test case elements using the CUAS.

E3.1.1 RUN STREAM FOR THE FRED PROGRAM

```
1.  @RUN ...
2.  @USE   DBF.,OUTPUT.
3.  @USE   RIP.,INPUT.
4.  @XQT   FML-L79351*PHPA.FRED
5.  @FIN
```

1. Required statement to initiate run on EXEC 8.
2. Assign internal name DBF to OUTPUT.
3. Assign internal name RIP to INPUT.

4. The FRED program is executed to analyze the program file directory and insert into the DBF an element containing the name of all the user relocatable elements.

5. Required statement to end run on EXEC 8.

E.3.1.2 RUN STREAM FOR THE TCRGEN PROGRAM

```
1. @RUN ...
2. @USE DBF.,USER FILE
3. @XQT FML-L79351*PHPA.TCRGEN
4. @FIN
```

1. Required statement to initiate run on EXEC 8.
2. Assign internal name DBF to the SVDSTCL file. DBF must contain the user master file directory and the test case elements.
3. The TCRGEN program to produce the reports.
4. Required statement to end run on EXEC 8.

See the CUAS user's guide for a sample CUAS run stream.

E3.2 OUTPUT DESCRIPTION

E3.2.1 NORMAL OUTPUT FOR THE FRED PROGRAM

Normal output for the FRED program consists of one basic display:

1. This report is initiated by the heading, "MASTER ELEMENT DIRECTORY CREATION PROGRAM." The following information is printed for each element.

   - Element name
   - Version name
   - Creation date
   - Creation time

E3.2.2 NORMAL OUTPUT FOR THE TCRGEN PROGRAM

Normal output for the TCRGEN program consists of three basic displays:

1. Test Case Element Name Report - This report is initiated by the heading, "TEST CASE USAGE REPORTING SYSTEM." The next line indicates a legend for the information to follow.

   The following information is printed for each test case.

   - Test case names in alphabetical order
   - Element names used by the test case but not listed in master file directory

   This report is displayed if the master file directory is incomplete and should be updated.

2. Test Case Usage Report - This report is initiated by the heading, "TEST CASE USAGE REPORTING SYSTEM." The next line provides a legend for the asterisk. An asterisk is placed adjacent to and preceding the "X" to inform the user that the master element directory time and date for the element does not coincide with the creation date and time for the element used by the test case. The following information is printed for each element:

   - The test case names in alphabetical order, with creation date and time
   - The element names including version name in alphabetical order, and creation date and time
   - When an element was used by a particular test case, this is indicated by an "X" in the element name row and test case name column intersection. The "X" is preceded by an asterisk under the conditions specified in the legend.

3. Element Usage Report - This report is initiated by the heading, "ELEMENT" and "TEST CASE NAMES USING EACH ELEMENT." The following information is printed for each element:

- The element names in alphabetical order
- The names of all test cases which called the element are listed in alphabetical order. An example of the FRED program report is shown in section E5.4.2. Examples of the TCRGEN reports are shown in section E5.4.4.

E3.2.3 ERROR MESSAGES

Error messages are provided in the TCURS to inform the user of any condition that prohibits the normal operation of the system. There are two forms of error messages; the first form identifies and describes the error condition directly in the printed output, and the second form provides error numbers which are described in the following table.

The error format is 'ERR n ERROR' where n may take on values as follows:

Diagnostic (n)	Program	Description
1	FRED	Error indicates a FASTRAND read error. This error condition is documented in Univac Publication 4144, Rev. 3.
2	FRED	The element table is empty; this error will terminate program execution
3	FRED	I/O error on file RIP
4	FRED	See diagnostic 1
5	FRED	See diagnostic 1
6	FRED	I/O error on file RIP
7	FRED	I/O error on file RIP
8	FRED	See diagnostic 1
9	TCRGEN	The element table is empty; this error will terminate program execution
10	TCRGEN	See diagnostic 1
11	TCRGEN	ERTRAN read error. This error is documented in Univac Publication 4144, Rev. 3.

E4. EXECUTION CHARACTERISTICS

E4.1 RESTRICTIONS

The TCRGEN program has these limitations:

1. The program is valid only if there exist a program file directory.
2. The maximum number of test case names allowed is 50.
3. The maximum number of relocatable element names allowed in the master file directory is 700.

E4.2 RUNNING TIME

The run time for the TCRGEN program will vary depending on the size of the program file directory being evaluated. Approximately 2.18 minutes were required to evaluate the Space Vehicle Dynamics Simulation (SVDS) program file directory. The FRED program requires approximately 13.7 seconds to evaluate the SVDS program file.

E4.3 ACCURACY/VALIDITY

Validation of the FRED and TCRGEN programs has been accomplished primarily by the comparison of contents from the normal SVDS output and by desk checking the results of the SVDS reports.

E5. REFERENCE INFORMATION

E5.1 FUNCTIONAL FLOWCHART OF THE FRED PROGRAM

Figure 2 illustrates the flow of the FRED program logic.

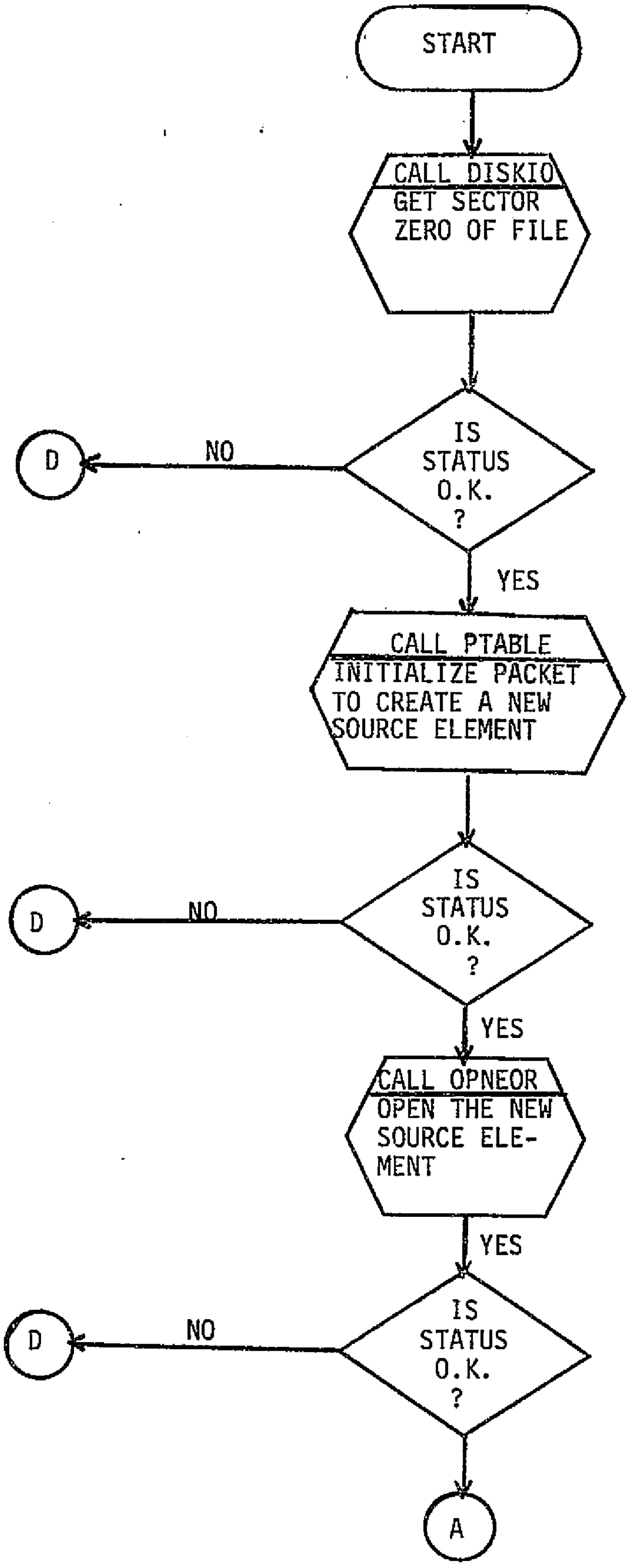
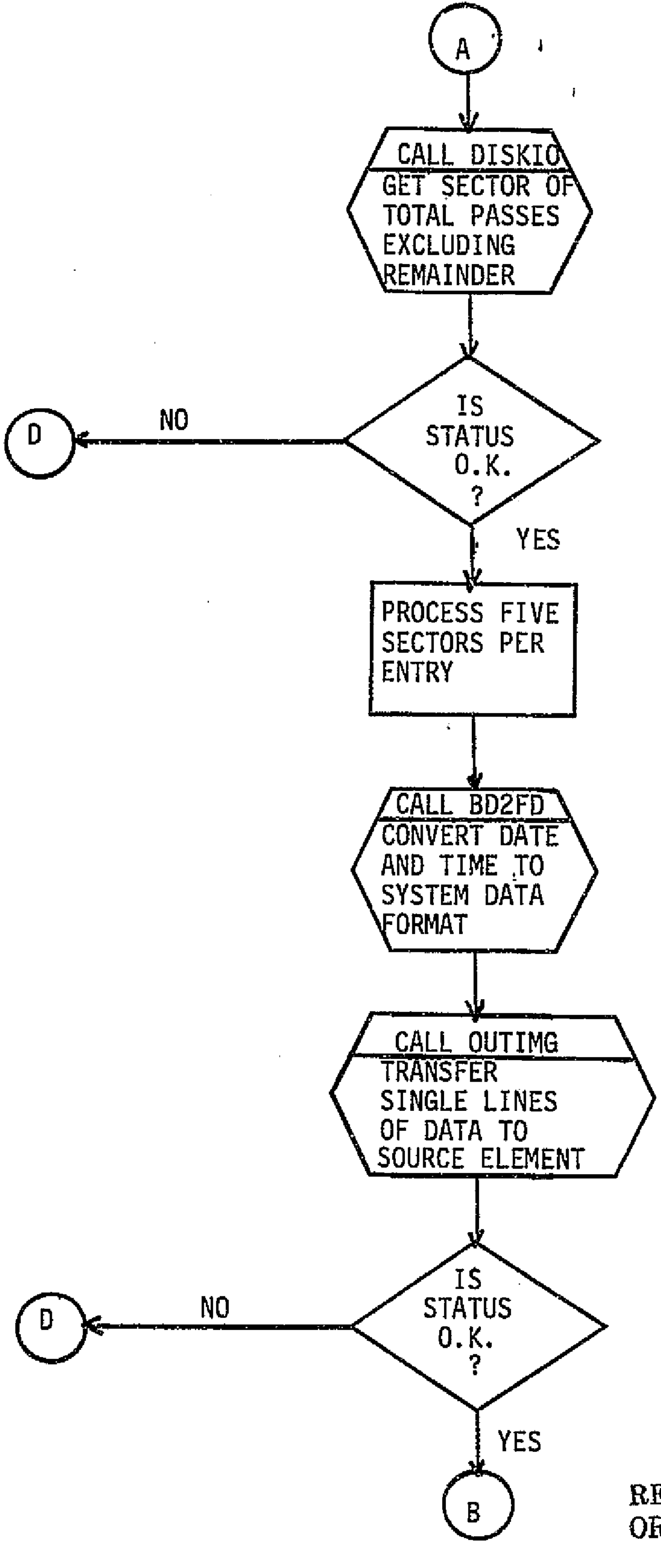


Figure 2. - FRED program functional flowchart.


A
CALL DISKIO
GET SECTOR OF TOTAL PASSES EXCLUDING REMAINDER
IS STATUS O.K. ?
NO
D
YES
PROCESS FIVE SECTORS PER ENTRY
CALL BD2FD
CONVERT DATE AND TIME TO SYSTEM DATA FORMAT
CALL OUTIMG
TRANSFER SINGLE LINES OF DATA TO SOURCE ELEMENT
IS STATUS O.K. ?
NO
D
YES
B

REPRODUCIBILITY OF TH
ORIGINAL PAGE IS POOR

Figure 2. - Continued.

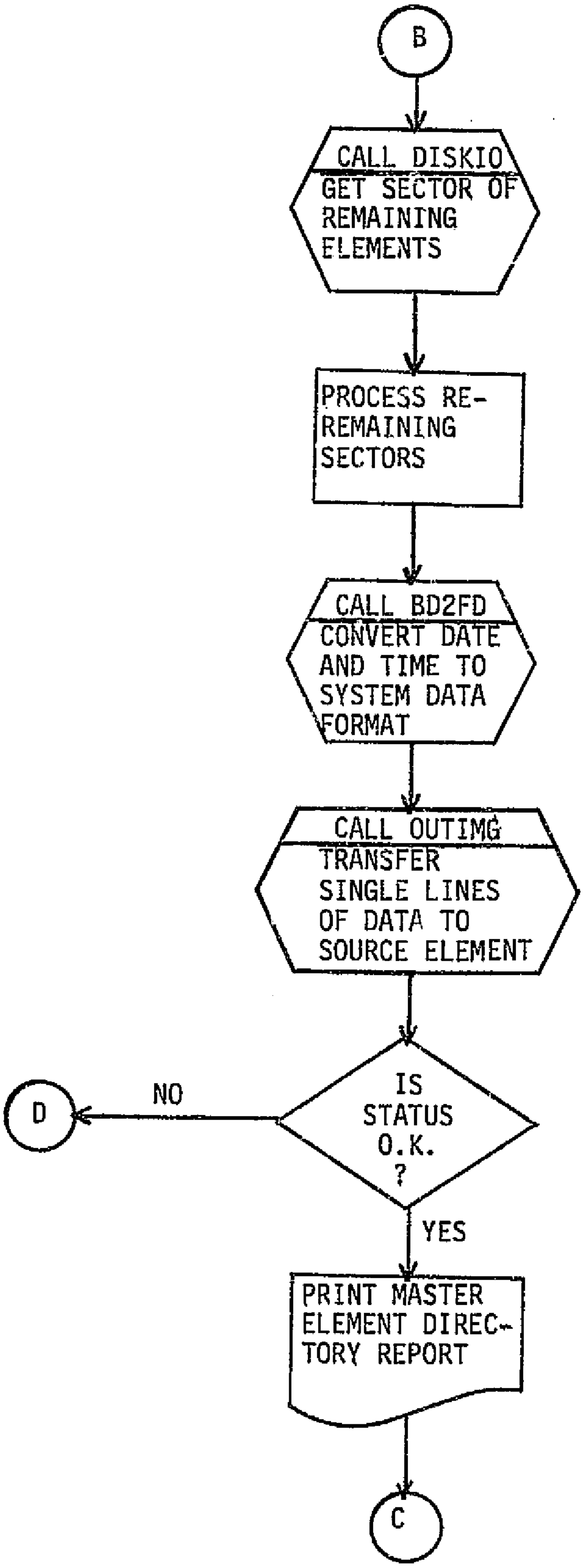

B
CALL DISKIO
GET SECTOR OF REMAINING ELEMENTS
PROCESS RE-REMAINING SECTORS
CALL BD2FD
CONVERT DATE AND TIME TO SYSTEM DATA FORMAT
CALL OUTIMG
TRANSFER SINGLE LINES OF DATA TO SOURCE ELEMENT
IS STATUS O.K. ?
NO
D
YES
PRINT MASTER ELEMENT DIRECTORY REPORT
C

Figure 2. - Continued.

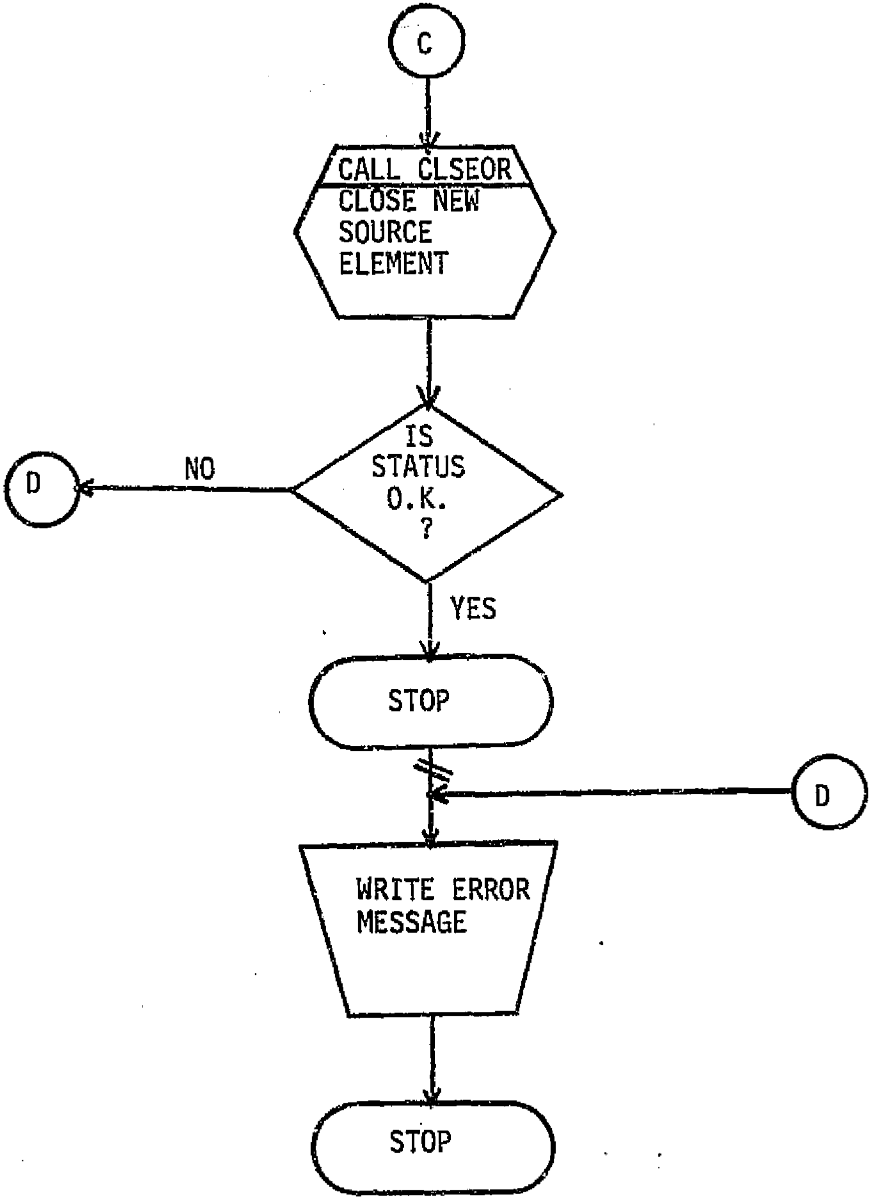


Figure 2.- Concluded.

E.5.2 FUNCTIONAL FLOWCHART OF THE TCRGEN PROGRAM

Figure 3 illustrates the flow of the main program (TCRGEN) logic.

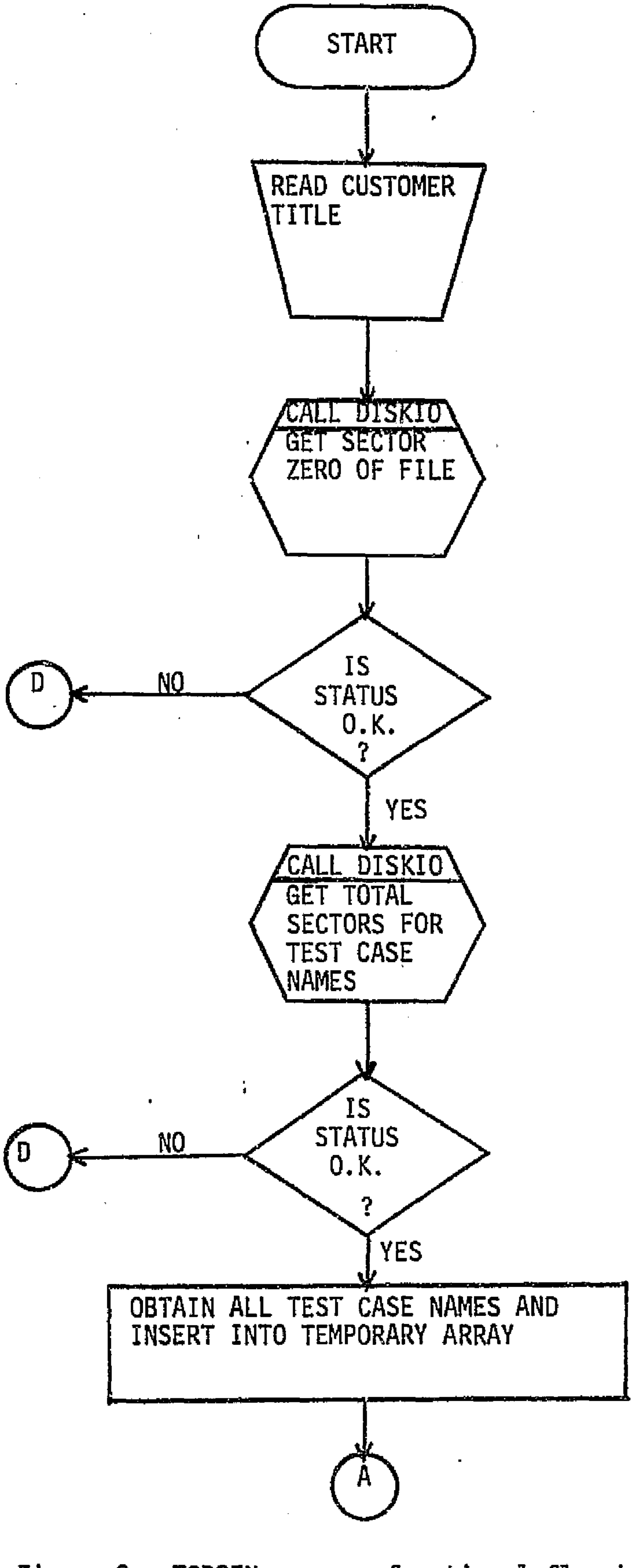


Figure 3.- TCRGEN program functional flowchart.

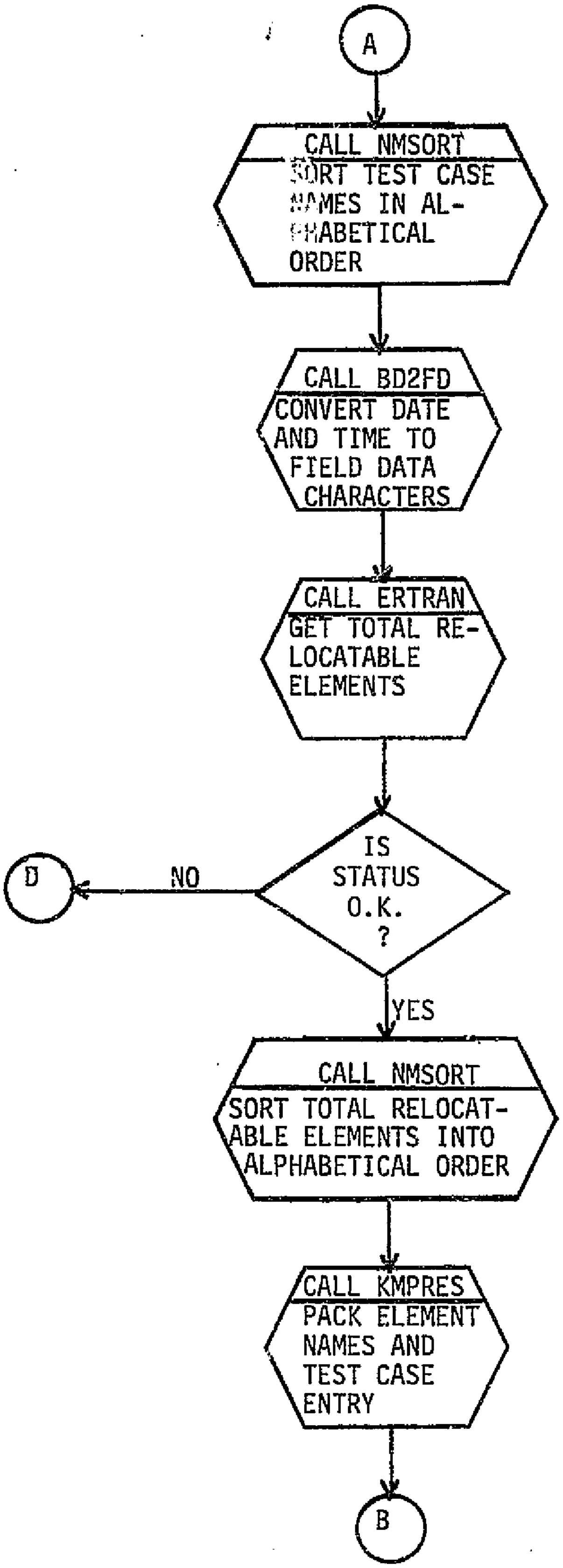

A
CALL NMSORT
SORT TEST CASE NAMES IN AL-PHABETICAL ORDER
CALL BD2FD
CONVERT DATE AND TIME TO FIELD DATA CHARACTERS
CALL ERTRAN
GET TOTAL RE-LOCATABLE ELEMENTS
IS STATUS O.K. ?
NO
D
YES
CALL NMSORT
SORT TOTAL RELOCAT-ABLE ELEMENTS INTO ALPHABETICAL ORDER
CALL KMPRES
PACK ELEMENT NAMES AND TEST CASE ENTRY
B

Figure 3.- Continued.

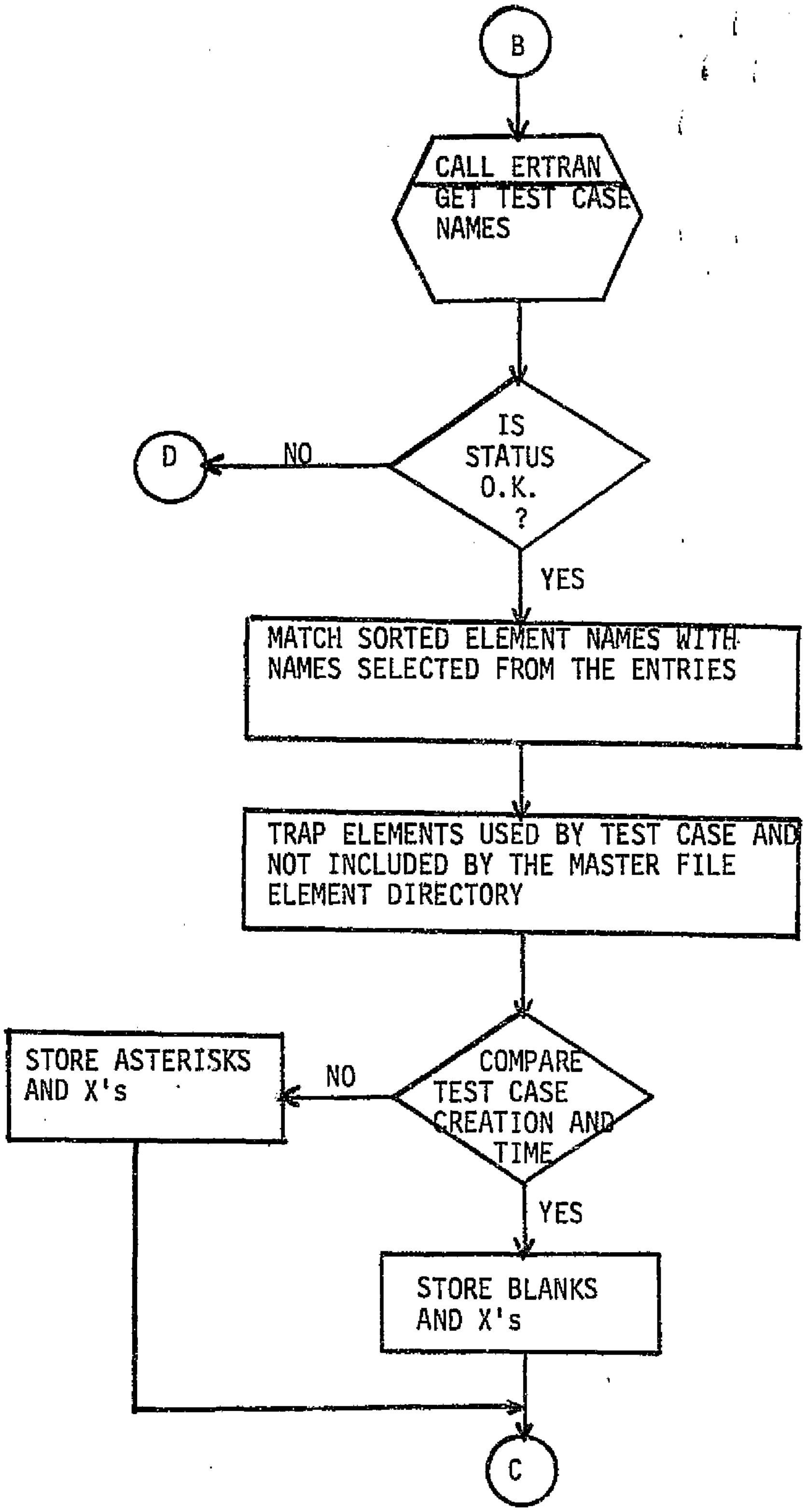


Figure 3.- Continued.

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

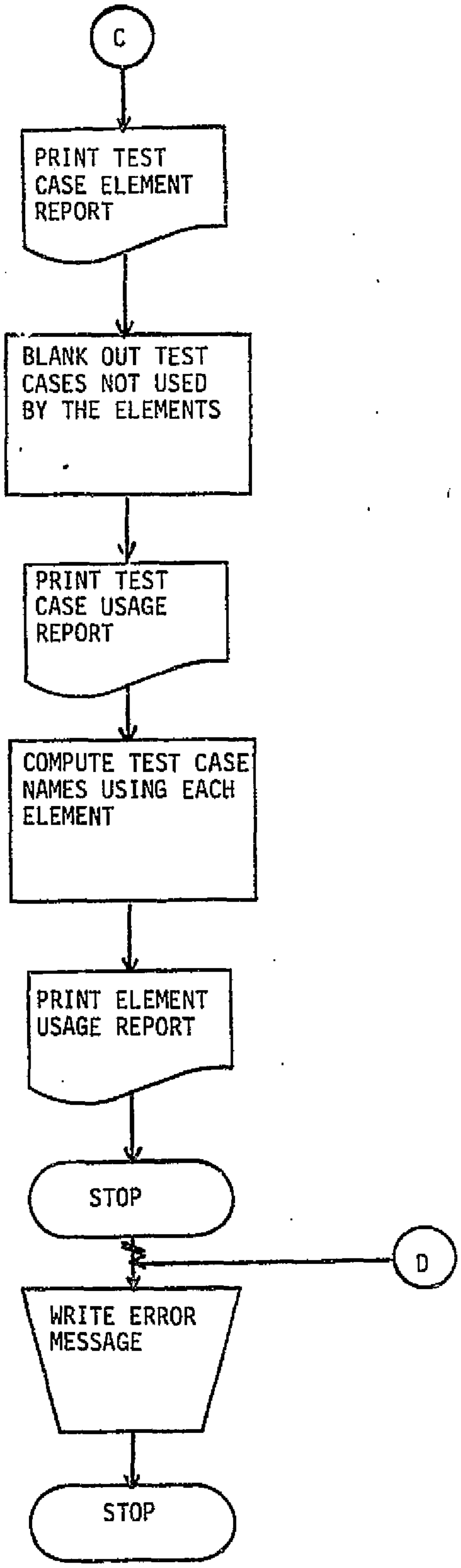

C
PRINT TEST CASE ELEMENT REPORT
BLANK OUT TEST CASES NOT USED BY THE ELEMENTS
PRINT TEST CASE USAGE REPORT
COMPUTE TEST CASE NAMES USING EACH ELEMENT
PRINT ELEMENT USAGE REPORT
STOP
D
WRITE ERROR MESSAGE
STOP

Figure 3.- Concluded.

E5.3 SUBROUTINE DOCUMENTATION

The JLALOC, KMPRES, and NMSORT subprograms, unique to the TCURS, are documented on the following pages.

Subprograms BD2FD, DISKIO, and ILLSFT, common to both the TCURS and the CUAS, are documented in section 5.5 of the CUAS document.

SUBROUTINE JLALOC

IDENTIFICATION

Name (Title)	- JLALOC (Locate Word)
Programmer, Date	- J. D. Oliver, June 1976
Machine Identification	- UNIVAC 1100-Series
Source Language	- FORTRAN V

PURPOSE

Subroutine JLALOC locates the word and position within a word where the nth character of a string would be.

USAGE

- Calling Sequence

```
CALL JLALOC (N, NW, NL)
```

Arguments:

Parameter name	In/Out	Dimension	Type	Description
N	In	1	I	The character number
NL	Out	1	I	The beginning bit containing this character
NW	Out	1	I	The word containing this character

METHOD

- Model

Subroutine JLALOC computes the word location and bit location of a string. In general, the bit location is computed using a mathematical intrinsic function.

The word location is given by:

$$NW = (N-1)/6 + 1$$

The bit location is given by:

$$NL = MOD(N-1,6) * 6$$

where

N = The character number
NW = The word containing this character
NL = The beginning bit containing this character.

FUNCTION KMPRES

IDENTIFICATION

Name (Title)	- KMPRES (Compress Blanks)
Programmer, Date	- J. D. Oliver, June 1976
Machine Identification	- UNIVAC 1100-Series
Source Language	- FORTRAN V

PURPOSE

Function KMPRES moves a substring from a string containing blanks to the array containing the compressed string.

USAGE

- Calling Sequence
  J=KMPRES (A, I, N, B, J)

Arguments:

Parameter name	In/Out	Dimension	Type	Description
A	In	1	R	The string containing blanks
B	Out	1	R	The array to contain the compressed string
I	In	1	I	The starting character in the string containing blanks
J	In	1	I	The beginning character in the array to contain the compressed string
N	In	1	I	The number of characters

METHOD

- Model

  Starting with the beginning character, function KMPRES is set to the number of nonblank characters in a substring containing blanks to compress the characters of the array.

RESTRICTIONS

- Operational

  Subroutine JLALOC is required.

SUBROUTINE NMSORT

IDENTIFICATION

Name (Title)	- NMSORT (Name Sorter)
Programmer, Date	- P. H. Horsley, August 1975
Machine Identification	- UNIVAC 1100-Series
Source Language	- FORTRAN V

PURPOSE

Subroutine NMSORT sorts an array of alphanumeric names into alphabetical order according to the collating sequence of fieldata characters on UNIVAC EXEC 8.

USAGE

- Calling Sequence

  CALL NMSORT(NAM, IX, NE)

  Arguments:

Parameter Name	In/Out	Dimension	Type	Description
NAM	In	Variable	I	The array of fieldata names to be sorted. Each name occupies two contiguous cells in the array, or is 12 characters long
NE	In	1	I	The number of two-cell entries in array NAM, or half the length of the array NAM in cells
IX	Out	Variable		Array of NE entries containing the index order of the original NAM array needed to place it in alphabetical order
NAM	Out	Variable	I	The array of fieldata names sorted into alphabetical order

- Error Messages

  None - if the parameter NE is less than or equal to 1, no sort is performed and a return is done.

METHOD

- Model

  The fieldata entries in the parameter array NAM are considered as 72 bit, unsigned integers by subroutine NMSORT. A bubble sort technique is used to place the 72 bit unsigned integers into ascending numerical sequence which effectively results in an alphabetic sort based on the UNIVAC EXEC 8 fieldata characters collating sequence. The parameter array IX is returned with the index position of each name in the array NAM on entry. For example, if name number 5 was moved to position 1 in the course of the sort, the first cell of IX would contain the number 5.

E5.4 SAMPLE INPUT/OUTPUT

E5.4.1 SAMPLE INPUT FOR THE FRED PROGRAM

The job stream given below indicates the operations necessary to execute the FRED program. This sample input illustrates the application of the FRED program to the SVDS file directory. For a general description of the input, refer to the run stream description provided in section E3.1.1.

```
@RUN ...
@USE         DBF.,FML-L79351*SVDSTCL.
@USE         RIP.,FM9*SVDS.
@XQT         FML-L79351*PHPA.FRED
@FIN
```

E5.4.2 SAMPLE OUTPUT FOR THE FRED PROGRAM

A portion of the printed output generated by the FRED program execution follows.

```
  MASTER ELEMENT DIRECTORY CREATION PROGRAM

     ELEMENT        VERSION        CREATION      CREATION
     NAME           NAME             DATE           TIME
   1*  LOSS                        09/05/75      08:56:03
   2*  MATHFN                      09/05/75      08:58:10
   3*  MATOPS                      09/05/75      08:58:28
   4*  MATROT                      09/05/75      08:59:32
   5*  MINV                        09/05/75      09:00:43
   6*  MULT                        09/05/75      09:04:36
   7*  MULT3                       09/05/75      09:04:39
   8*  MULT4                       09/05/75      09:04:43
   9*  M509I                       09/05/75      09:04:52
  10*  NOMCON                      09/05/75      09:05:08
  11*  ONEDM                       09/05/75      09:10:12
  12*  ORDER                       09/05/75      09:10:55
  13*  OREBLK                      09/05/75      09:11:03
  14*  OUT31                       09/05/75      09:12:00
  15*  PARALB                      09/05/75      09:12:18
  16*  PHSANG                      09/05/75      09:12:39
  17*  PLM509                      09/05/75      09:13:50
  18*  PMATCH                      09/05/75      09:13:59
  19*  PRA63C                      09/05/75      09:14:49
  20*  PRA63H                      09/05/75      09:15:10
  21*  PRBMMD                      09/05/75      09:15:13
  22*  PROCT                       09/05/75      09:15:20
  23*  PREGLT                      09/05/75      09:15:27
  24*  PRINT                       09/05/75      09:16:12
  25*  PROJCT                      09/05/75      09:17:00
  26*  PSBMMD                      09/05/75      09:19:03
  27*  PWEIGH                      09/05/75      09:19:08
  28*  RANDN                       09/05/75      09:19:57
  29*  RDER31                      09/05/75      09:22:53
  30*  RES                         09/05/75      09:24:46
  31*  RN2S                        09/05/75      09:26:04
  32*  ROTDER                      09/05/75      09:26:38
  33*  ROTMAT                      09/05/75      09:26:55
  34*  RPNMAT                      09/05/75      09:27:58
  35*  RTMATX                      09/05/75      09:29:17
  36*  RWNCIO                      09/05/75      09:29:34
  37*  SEARMT                      09/05/75      09:31:56
  38*  SKEW                        09/05/75      09:33:20
  39*  SLOSHI                      09/05/75      09:33:37
  40*  SPLNI                       09/05/75      09:34:47
  41*  SUM                         09/05/75      09:36:42
  42*  TA                          09/05/75      09:37:19
  43*  TABLE                       09/05/75      09:37:23
  44*  TAB509                      09/05/75      09:37:54
  45*  TDER1                       09/05/75      09:39:29
  46*  TDER2                       09/05/75      09:39:37
  47*  TDER31                      09/05/75      09:39:58
  48*  TINORM                      09/05/75      09:41:48
  49*  TINTEG                      09/05/75      09:41:51
  50*  TLAG                        09/05/75      09:41:54
  51*  TOPCON                      09/05/75      09:42:52
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
         MASTER ELEMENT DIRECTORY CREATION PROGRAM

           ELEMENT       VERSION       CREATION     CREATION
           NAME          NAME            DATE         TIME
       52*  TOPDOT                     09/05/75     09:43:04
       53*  TRIVAR                     09/05/75     09:45:10
       54*  TVBEC                      09/05/75     09:47:30
       55*  TWASH                      09/05/75     09:47:46
       56*  UNIT                       09/05/75     09:47:50
       57*  UNVEC                      09/05/75     09:48:01
       58*  VECMG                      09/05/75     09:49:25
       59*  VECOPS                     09/05/75     09:49:29
       60*  VRA73C                     09/05/75     09:50:08
       61*  VRA73H                     09/05/75     09:50:16
       62*  WRITEX                     09/05/75     09:53:25
       63*  XLIMIT                     09/05/75     09:53:35
       64*  XYZTOE                     09/05/75     09:53:39
       65*  ABINDI                     09/05/75     18:22:52
       66*  ANSWER                     09/05/75     18:27:52
       67*  ARCOS                      09/05/75     18:28:36
       68*  ARODOT                     09/05/75     18:29:01
       69*  ARSIN                      09/05/75     18:29:27
       70*  ARTAN                      09/05/75     18:29:31
       71*  AVELOC                     09/05/75     18:30:28
       72*  AZTTAR                     09/05/75     18:30:34
       73*  BDE74C                     09/05/75     18:31:02
       74*  BDE74H                     09/05/75     18:31:08
       75*  BDP63C                     09/05/75     18:31:28
       76*  BDP63H                     09/05/75     18:31:34
       77*  BDSP63                     09/05/75     18:31:43
       78*  BDS30C                     09/05/75     18:31:54
       79*  BDS30H                     09/05/75     18:32:03
       80*  BDS60C                     09/05/75     18:32:11
       81*  BDS60H                     09/05/75     18:32:20
       82*  BDS62R                     09/05/75     18:32:30
       83*  BDV71R                     09/05/75     18:32:37
       84*  BDV73C                     09/05/75     18:32:42
       85*  BDV73H                     09/05/75     18:32:48
       86*  BEND                       09/05/75     18:33:38
       87*  BENDAT                     09/05/75     18:33:46
       88*  BENDI                      09/05/75     18:33:53
       89*  BINTRP                     09/05/75     18:34:00
       90*  BMATRX                     09/05/75     18:34:23
       91*  BNDRES                     09/05/75     18:34:52
       92*  BNDVAL                     09/05/75     18:35:07
       93*  BPROC                      09/05/75     18:35:15
       94*  CMDFIL                     09/05/75     18:35:28
       95*  COE                        09/05/75     18:35:33
       96*  COMELE                     09/05/75     18:35:36
       97*  CONTAC                     09/05/75     18:36:39
       98*  CROSS                      09/05/75     18:37:06
       99*  DAYZ                       09/05/75     18:38:25
      100*  DEPORO                     09/05/75     18:38:45
      101*  DETPTR                     09/05/75     18:38:49
      102*  DOT                        09/05/75     18:39:10
```

E5.4.3 SAMPLE INPUT FOR THE TCRGEN PROGRAM

The job stream given below indicates the operations necessary to execute the TCRGEN program. This sample input illustrates the application of the TCRGEN program to the SVDS file directory. For a general description of the input, refer to the run stream description provided in section E3.1.2.

```
@RUN ...
@USE        DBF.,FML-L79351*SVDSTCL
@XQT        FML-L79351*PHPA.TCRGEN
@FIN
```

E5.4.4 SAMPLE OUTPUT FOR THE TCRGEN PROGRAM

A portion of the printed output generated by the TCRGEN program execution follows.

```
                              TEST CASE USAGE REPORTING SYSTEM

TEST CASE      ELEMENT USED BY TEST  CASE AND NOT INCLUDED IN THE MASTER FILE ELEMENT DIRECTORY

ALT1       CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME

ALT2       CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME

DECK1      CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME
           TRACER
DECK10     CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME

DECK11     CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME
           TRACER
DECK12     CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME

DECK2      CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME

DECK3      CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME

DECK4      CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME

DECK5      CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME
           TRACER
DECK6      CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME
           TRACER
DECK7      CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME
           TRACER
DECK8      CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME

DECK9      CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME
           TRACER
ENTRY1     CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME

ENTRY3     CLOCK      EFNAME     FILEIO     FNAME      GETDAT     IICONT     MOVER      PCTD       SNAVD
           STIME
ENTRY4     CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME
```

```
                              T E S T   C A S E   U S A G E   R E P O R T I N G   S Y S T E M

TEST CASE      ELEMENT USED BY TEST  CASE AND NOT INCLUDED IN THE MASTER FILE ELEMENT DIRECTORY

ENTRY5     CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME

GNCORB     CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME

GNC1       CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME

GNC2       CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME

GNC3       CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME

GNC4       CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME

GNC5       CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME

LAUNCH5    CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME

LAUNCH4    CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME

LAUNCH3    CLOCK      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME      TRACER
LAUNCH2    CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME

LAUNCH1    CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME
           TRACER
NAVE1      CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME

NAVE2      CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME

NAVL1      CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME
           TRACER
OMS1       CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME
           TRACER
ORBIT1     CLOCK      EFNAME     FNAME      FILEIO     GETDAT     MOVER      IICONT     PCTD       STIME
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
                                    TEST CASE USAGE REPORTING SYSTEM
TEST CASE     ELEMENT USED BY TEST  CASE AND NOT INCLUDED IN THE MASTER FILE ELEMENT DIRECTORY

ORBIT2    CLOCK       EFNAME      FNAME       FILEIO      GETDAT      MOVER       IICONT      PCTD        STIME

RCS1      CLOCK       EFNAME      FNAME       FILEIO      GETDAT      MOVER       IICONT      PCTD        STIME

RCS2      CLOCK       EFNAME      FNAME       FILEIO      GETDAT      MOVER       IICONT      PCTD        STIME

SRBDKI    CLOCK       EFNAME      FNAME       FILEIO      GETDAT      MOVER       IICONT      PCTD        STIME
          TRACER
TBAPTC    CLOCK       EFNAME      FNAME       FILEIO      GETDAT      MOVER       IICONT      NSTOP$      NWEF$
          PCTD        STIME
WINDI     CLOCK       EFNAME      FNAME       FILEIO      GETDAT      MOVER       IICONT      PCTD        STIME
```

T E S T   C A S E   U S A G E   R E P O R T I N G   S Y S T E M

(* INDICATES MASTER ELEMENT DIRECTORY TIME AND DATE DO NOT COINCIDE WITH THE ELEMENT USED BY TEST CASE TIME AND DATE)

				ALT1	ALT2	DECK1	DECK10	DECK11	DECK12	DECK2	DECK3	DECK4
ELEMENT	VERSION	CREATION	CREATION	05/14/76	05/11/76	05/11/76	05/11/76	05/15/76	05/11/76	05/11/76	05/11/76	05/11/76
NAME	NAME	DATE	TIME	16:32:15	15:08:43	07:46:13	09:00:00	09:58:34	08:17:09	09:21:57	13:56:25	12:33:38
AABTAR		05/05/76	12:38:46									
AAINIT		04/07/76	16:55:55									
AASNSP		03/02/76	16:30:57									
ABIND		01/09/76	09:34:46									
ABINDI		09/05/75	18:22:52									
ABIND1		01/09/76	10:06:58									
ACCALC		03/02/76	16:30:58									
ACCGRP		05/05/76	12:39:06									
ACCUPD		05/05/76	12:39:46									
ACMAUP		05/05/76	12:42:35									
ACMEAS		05/05/76	12:43:35									
ACSINT		11/05/75	11:05:00	X	X							
ACSUP		05/05/76	12:44:30									
ACSUP2		05/05/76	12:45:56									
ACS15		05/05/76	12:49:34	X	X							
ACTTVC		04/07/76	16:27:05									
ADSEN		04/07/76	16:27:16									
ADSET		01/09/76	09:35:24	*X	*X.	*X	*X	*X	*X	*X	*X	*X
ADSOP		04/07/76	16:27:34									
AEPHEM		01/09/76	09:56:57									
AERALT		04/07/76	16:27:58	X	X							
AERAUX		05/05/76	12:51:19									
AERO		05/05/76	12:51:55									
AGIN		04/07/76	16:28:25									
AGSUP		01/09/76	09:37:34									
AGUID		05/05/76	12:53:10									
ALDENS		05/05/76	12:55:24									
ALFC		03/02/76	16:31:51									
ALIGN		03/02/76	16:31:48	X	X	X	X	X	X	X	X	X
ALOCVF		04/07/76	16:55:59									
ALPRD		01/09/76	09:38:07									
ALPRNT		03/02/76	16:31:54									
ALTBLK		01/09/76	09:43:50	X	X							
ALTDLL		04/07/76	16:28:42	X	X	X	X	X	X	X	X	X
ALTIMR		05/05/76	12:55:36									
ANOM		01/09/76	09:34:12	X	X	X			X			
ANSUP		01/09/76	09:43:56									
ANSWER		09/05/75	18:27:52									
AOATAR		05/05/76	12:5[illegible]									
AOMS		05/05/76	12:55:54									
AOMSG		05/12/76	21:23:09									
AOMSIN		04/07/76	16:28:54									
ARCOS		09/05/75	18:28:36	X	X	X						
ARDSPF		01/09/76	10:06:16									
ARELST		05/05/76	12:56:17									
ARINTP		04/07/76	16:56:46									
ARMEAS		01/09/76	09:44:06									
AROBLK		09/09/75	18:04:39	X	X						X	X
AROCAL		04/07/76	16:29:03	X	X						X	X
AROCOM		04/07/76	16:29:12									

```
                                    TEST CASE USAGE REPORTING SYSTEM

 ( * INDICATES MASTER ELEMENT DIRECTORY TIME AND DATE DO NOT COINCIDE WITH THE ELEMENT USED BY TEST CASE TIME AND DATE)

                                             ALT1     ALT2     DECK1    DECK10   DECK11   DECK12   DECK2    DECK3    DECK4
ELEMENT      VERSION      CREATION CREATION  05/14/76 05/11/76 05/11/76 05/11/76 05/15/76 05/11/76 05/11/76 05/11/76 05/11/76
NAME         NAME             DATE   TIME    16:32:15 15:08:43 07:46:13 09:00:00 09:58:34 08:17:09 09:21:57 13:56:25 12:33:38
XESTAB                    05/05/76 13:31:28
XLIMIT                    09/05/75 09:53:35     X        X
XQTOPS                    05/14/76 13:56:13     X       *X       *X       *X        X       *X       *X       *X       *X
XYZTOE                    09/05/75 09:53:39     X        X        X                          X
```

TEST CASE USAGE REPORTING SYSTEM

(* INDICATES MASTER ELEMENT DIRECTORY TIME AND DATE DO NOT COINCIDE WITH THE ELEMENT USED BY TEST CASE TIME AND DATE)

```
                                              DECK5     DECK6     DECK7     DECK8     DECK9     ENTRY1    ENTRY3    ENTRY4    ENTRY5
ELEMENT       VERSION       CREATION CREATION 05/11/76  05/11/76  05/11/76  05/11/76  05/11/76  05/11/76  05/20/76  05/13/76  05/11/76
NAME          NAME            DATE     TIME   09:42:00  09:16:39  10:17:03  07:35:23  10:27:52  23:50:34  14:57:34  22:06:51  20:13:45
AABTAR                      05/05/76 12:38:46
AAINIT                      04/07/76 16:55:55
AASNSP                      03/02/76 16:30:57
ABIND                       01/09/76 09:34:46
ABINDI                      09/05/75 18:22:52
ABIND1                      01/09/76 10:06:58
ACCALC                      03/02/76 16:30:58
ACCGRP                      05/05/76 12:39:06
ACCUPD                      05/05/76 12:39:46
ACMAUP                      05/05/76 12:42:35
ACMEAS                      05/05/76 12:43:35
ACSINT                      11/05/75 11:05:00                                                  X                   X
ACSUP                       05/05/76 12:44:30
ACSUP2                      05/05/76 12:45:56
ACS15                       05/05/76 12:49:34                                                  X                   X
ACTTVC                      04/07/76 16:27:05
ADSEN                       04/07/76 16:27:16                                                                      X         X
ADSET                       01/09/76 09:35:24   *X        *X        *X        *X        *X        *X        *X       X         X
ADSOP                       04/07/76  6:27:34                                                                      X         X
AEPHEM                      01/09/76 09:56:57
AERALT                      04/07/76 16:27:58
AERAUX                      05/05/76 12:51:19                                                  X                   X
AERO                        05/05/76 12:51:55                                                  X                   X
AGIN                        04/07/76 16:28:25
AGSUP                       01/09/76 09:37:34
AGUID                       05/05/76 12:53:10
ALDENS                      05/05/76 12:55:24
ALFC                        03/02/76 16:31:51                                                                                X
ALIGN                       03/02/76 16:31:48    X         X         X         X         X         X         X         X         X
ALOCVF                      04/07/76 16:55:59
ALPRD                       01/09/76 09:38:07
ALPRNT                      03/02/76 16:31:54
ALTBLK                      01/09/76 09:43:50
ALTDLL                      04/07/76 16:28:42    X         X         X         X         X         X         X         X         X
ALTIMR                      05/05/76 12:55:36
ANOM                        01/09/76 09:34:12
ANSUP                       01/09/76 09:43:56
ANSWER                      09/05/75 18:27:52
AOATAR                      05/05/76 12:55:48
AOMS                        05/05/76 12:55:54
AOMSG                       05/12/76 21:23:09
AOMSIN                      04/07/76 16:28:54
ARCOS                       09/05/75 18:28:36
ARDSPF                      01/09/76 10:06:16
ARELST                      05/05/76 12:56:17
ARINTP                      04/07/76 16:56:46
ARMEAS                      01/09/76 09:44:06
AROBLK                      09/09/75 18:04:39
AROCAL                      04/07/76 16:29:03                                                  X        *X         X         X
AROCOM                      04/07/76 16:29:12
```

```
                                T E S T   C A S E   U S A G E   R E P O R T I N G   S Y S T E M

  ( * INDICATES MASTER ELEMENT DIRECTORY TIME AND DATE DO NOT COINCIDE WITH THE ELEMENT USED BY TEST CASE TIME AND DATE)

                                                     DECK5    DECK6    DECK7    DECK8    DECK9    ENTRY1   ENTRY3   ENTRY4   ENTRY5
ELEMENT         VERSION         CREATION CREATION  05/11/76 05/11/76 05/11/76 05/11/76 05/11/76 05/11/76 05/20/76 05/13/76 05/11/76
NAME            NAME              DATE     TIME    09:42:00 09:16:39 10:17:03 07:35:23 10:27:52 23:50:34 14:57:34 22:06:51 20:13:45
XESTAB                          05/05/76 13:31:28
XLIMIT                          09/05/75 09:53:35
XQTOPS                          05/14/76 13:56:13     *X       *X       *X       *X       *X       *X        X       *X       *X
XYZTOE                          09/05/75 09:53:39
```

III

```
                         T E S T   C A S E   U S A G E   R E P O R T I N G   S Y S T E M

     ( * INDICATES MASTER ELEMENT DIRECTORY TIME AND DATE DO NOT COINCIDE WITH THE ELEMENT USED BY TEST CASE TIME AND DATE)

                                             GNCORB   GNC1     GNC2     GNC3     GNC4     GNC5     LAUNCH5  LAUNCH4  LAUNCH3
ELEMENT      VERSION      CREATION CREATION  05/19/76 05/15/76 05/11/76 05/11/76 05/11/76 05/14/76 05/14/76 05/12/76 05/20/76
NAME         NAME           DATE     TIME    15:46:34 18:02:53 08:03:36 08:44:39 14:36:44 00:43:58 00:57:31 13:07:07 02:12:12
AABTAR                    05/05/76 12:38:46                                                  X
AAINIT                    04/07/75 16:55:55                                                  X
AASNSP                    03/02/76 16:30:57
ABIND                     01/09/76 09:34:46
ABINDI                    09/05/75 18:22:52
ABIND1                    01/09/76 10:06:58
ACCALC                    03/02/76 16:30:58
ACCGRP                    05/05/76 12:39:06
ACCUPD                    05/05/76 12:39:46     X
ACMAUP                    05/05/76 12:42:35                       X        X        X        X
ACMEAS                    05/05/76 12:43:35
ACSINT                    11/05/75 11:05:00
ACSUP                     05/05/76 12:44:30
ACSUP2                    05/05/76 12:45:56
ACS15                     05/05/76 12:49:34
ACTTVC                    04/07/76 16:27:05
ADSEN                     04/07/76 16:27:16
ADSET                     01/09/76 09:35:24    *X       *X       *X       *X       *X       *X       *X       *X       *X
ADSOP                     04/07/76 16:27:34
AEPHEM                    01/09/76 09:56:57
AERALT                    04/07/76 16:27:58
AERAUX                    05/05/76 12:51:19
AERO                      05/05/76 12:51:55
AGIN                      04/07/76 16:28:25                       X        X        X        X
AGSUP                     01/09/76 09:37:34                       X        X        X        X
AGUID                     05/05/76 12:53:10                       X        X        X        X
ALDENS                    05/05/76 12:55:24
ALFC                      03/02/76 16:31:51
ALIGN                     03/02/76 16:31:48     X        X        X        X        X        X        X        X        X
ALDCVF                    04/07/76 16:55:59                                                  X
ALPRD                     01/09/76 09:38:07
ALPRNT                    03/02/76 16:31:54
ALTBLK                    01/09/76 09:43:50
ALTDLL                    04/07/76 16:28:42     X        X        X        X        X        X        X        X        X
ALTIMR                    05/05/76 12:55:36
ANOM                      01/09/76 09:34:12     X        X                          X        X        X
ANSUP                     01/09/76 09:43:56                       X        X        X        X
ANSWER                    09/05/75 18:27:52
AOATAR                    05/05/76 12:55:48                                                  X
AOMS                      05/05/76 12:55:54                       X        X                 X
AOMSG                     05/12/76 21:23:09                      *X       *X                 X
AOMSIN                    04/07/76 16:28:54                       X        X                 X
ARCOS                     09/05/75 18:28:36
ARDSPF                    01/09/76 10:06:16
ARELST                    05/05/76 12:56:17
ARINTP                    04/07/76 16:56:46
ARMEAS                    01/09/76 09:44:06
AROBLK                    09/09/75 18:04:39                       X        X        X        X        X        X
AROCAL                    04/07/76 16:29:03                       X        X        X        X        X        X
   COM                    04/07/76 16:29:12
```

```
                          T E S T   C A S E   U S A G E   R E P O R T I N G   S Y S T E M

( * INDICATES MASTER ELEMENT DIRECTORY TIME AND DATE DO NOT COINCIDE WITH THE ELEMENT USED BY TEST CASE TIME AND DATE)

                                            GNCORB   GNC1     GNC2     GNC3     GNC4     GNC5     LAUNCH5  LAUNCH4  LAUNCH3
ELEMENT      VERSION     CREATION CREATION  05/14/76 05/15/76 05/11/76 05/11/76 05/11/76 05/14/76 05/14/76 05/12/76 05/20/76
NAME         NAME          DATE     TIME    15:46:34 18:02:53 08:03:36 08:44:39 14:36:44 00:43:58 00:57:31 13:07:07 02:12:12
%ESTAB                   05/05/76 13:31:28
%LIMIT                   09/05/75 09:53:35
%QTOPS                   05/14/76 13:56:13     X        X       *X       *X       *X       *X                *X         X
%YZTOE                   09/05/75 09:53:39     X        X                          X        X        X
```

TEST CASE USAGE REPORTING SYSTEM

(* INDICATES MASTER ELEMENT DIRECTORY TIME AND DATE DO NOT COINCIDE WITH THE ELEMENT USED BY TEST CASE TIME AND DATE)

				LAUNCH2	LAUNCH1	NAVE1	NAVE2	NAVL1	OMS1	ORBIT1	ORBIT2	RCS1
ELEMENT	VERSION	CREATION	CREATION	05/11/76	05/11/76	05/17/76	05/12/76	05/11/76	05/15/76	05/11/76	05/12/76	05/11/76
NAME	NAME	DATE	TIME	08:06:33	09:16:13	14:26:49	15:59:01	08:58:13	10:43:28	13:18:21	16:00:28	15:06:28
AABTAR		05/05/76	12:38:46									
AAINIT		04/07/76	16:55:55									
AASNSP		03/02/76	16:30:57			X	X					
ABIND		01/09/76	09:34:46									
ABINDI		09/05/75	18:22:52									
ABIND1		01/09/76	10:06:58									
ACCALC		03/02/76	16:30:58						X			X
ACCGRP		05/05/76	12:39:06						X			X
ACCUPD		05/05/76	12:39:46									
ACMAUP		05/05/76	12:42:35									
ACMEAS		05/05/76	12:43:35			X	X					
ACSINT		11/05/75	11:05:00									
ACSUP		05/05/76	12:44:30									
ACSUP2		05/05/76	12:45:56									
ACS15		05/05/76	12:49:34									
ACTTVC		04/07/76	16:27:05									
ADSEN		04/07/76	16:27:16			X	X					
ADSET		01/09/76	09:35:24	*X	*X	X	X	*X	*X	*X	*X	*X
ADSOP		04/07/76	16:27:34				X					
AEPHEM		01/09/76	09:56:57									
AERALT		04/07/76	16:27:58									
AERAUX		05/05/76	12:51:19									
AERO		05/05/76	12:51:55									
AGIN		04/07/76	16:28:25									
AGSUP		01/09/76	09:37:34									
AGUID		05/05/76	12:53:10									
ALDENS		05/05/76	12:55:24									
ALFC		03/02/76	16:31:51									
ALIGN		03/02/76	16:31:48	X	X	X	X	X	X	X	X	X
ALOCVF		04/07/76	16:55:59									
ALPRD		01/09/76	09:38:07				X					
ALPRNT		03/02/76	16:31:54									
ALTBLK		01/09/76	09:43:50									
ALTDLL		04/07/76	16:28:42	X	X	X	X	X	X	X	X	X
ALTIMR		05/05/76	12:55:36				X					
ANOM		01/09/76	09:34:12						X	X	X	
ANSUP		01/09/76	09:43:56									
ANSWER		09/05/75	18:27:52									
AOATAR		05/05/76	12:55:48									
AOMS		05/05/76	12:55:54									
AOMSG		05/12/76	21:23:09									
AOMSIN		04/07/76	16:28:54									
ARCOS		09/05/75	18:28:36							X	X	
ARDSPF		01/09/76	10:06:16									
ARELST		05/05/76	12:56:17									
ARINTP		04/07/76	16:56:46									
ARMEAS		01/09/76	09:44:06									
AROBLK		09/09/75	18:04:39		X					X	X	
AROCAL		04/07/76	16:29:03		X	X	X	X				
AROCOM		04/07/76	16:29:12									

```
                              TEST CASE USAGE REPORTING SYSTEM

( * INDICATES MASTER ELEMENT DIRECTORY TIME AND DATE DO NOT COINCIDE WITH THE ELEMENT USED BY TEST CASE TIME AND DATE)

                                             LAUNCH2  LAUNCH1  NAVE1    NAVE2    NAVL1    OMS1     ORBIT1   ORBIT2   RCS1
ELEMENT      VERSION       CREATION CREATION 05/11/76 05/11/76 05/17/76 05/12/76 05/11/76 05/15/76 05/11/76 05/12/76 05/11/76
NAME         NAME           DATE     TIME    08:06:33 09:16:13 14:26:49 15:59:01 08:58:13 10:43:28 13:18:21 16:00:28 15:06:28
XESTAB                     05/05/76 13:31:28                                                          X        X
XLIMIT                     09/05/75 09:53:35                                                 X                 X
XQTOPS                     05/14/76 13:56:13   *X       *X        X       *X       *X        X       *X       *X       *X
XYZTOE                     09/05/75 09:53:39                                                 X        X        X
```

```
                 T E S T   C A S E   U S A G E   R E P O R T I N G   S Y S T E M

( * INDICATES MASTER ELEMENT DIRECTORY TIME AND DATE DO NOT COINCIDE WITH THE ELEMENT USED BY TEST CASE TIME AND DATE)

                                             ACS2     SABDK1   TBAPTC   WIND1
ELEMENT      VERSION      CREATION CREATION 05/11/76 05/11/76 05/11/76 05/17/76
NAME         NAME          DATE      TIME   14:37:41 10:22:32 11:32:40 16:57:34
AABTAR                    05/05/76 12:38:46
AAINIT                    04/07/76 16:55:55
AASNSP                    03/02/76 16:30:57
ABIND                     01/09/76 09:34:46
ABINDI                    09/05/75 18:22:52
ABIND1                    01/09/76 10:06:58
ACCALC                    03/02/76 16:30:58
ACCGRP                    05/05/76 12:39:06
ACCUPD                    05/05/76 12:39:46
ACMAUP                    05/05/76 12:42:35
ACMEAS                    05/05/76 12:43:35
ACSINT                    11/05/75 11:05:00     X
ACSUP                     05/05/76 12:44:30
ACSUP2                    05/05/76 12:45:56
ACS15                     05/05/76 12:49:34     X
ACTTVC                    04/07/76 16:27:05
ADSEN                     04/07/76 16:27:16
ADSET                     01/09/76 09:35:24    *X       *X       *X       *X
ADSOP                     04/07/76 16:27:34
AEPHEM                    01/09/76 09:56:57
AERALT                    04/07/76 16:27:59
AERAUX                    05/05/76 12:51:19     X
AERO                      05/05/76 12:51·55     X
AGIN                      04/07/76 16:28:2
AGSUP                     01/09/76 09:37:34
AGUID                     05/05/76 12:53:10
ALDENS                    05/05/76 12:55:24
ALFC                      03/C2/76 16:31:51
ALIGN                     03/02/76 16:31:48     X        X                 X
ALOCVF                    04/07/76 16:55:59
ALPRD                     01/09/76 09:38:07
ALPRNT                    03/02/76 16:31:54
ALTBLK                    01/09/76 09:43:50
ALTDLL                    04/07/76 16:28:42     X        X                 X
ALTIMR                    05/05/76 12:55:36
ANOM                      01/09/76 09:34:12
ANSUP                     01/09/76 09:43:56
ANSWER                    09/05/75 18:27:52
AOATAR                    05/05/76 12:55:48
AOMS                      05/05/76 12:55:54
AOMSG                     05/12/76 21:23:09
AOMSIN                    04/07/76 16:28:54
ARCOS                     09/05/75 18:28:36
ARDSPF                    01/09/76 10:06:16
ARELST                    05/05/76 12:56:17
ARINTP                    04/07/76 16:56:46
ARMEAS                    01/09/76 09:44:06
AROBLK                    09/09/75 18:04:39                                X
AROCAL                    04/07/76 16:29:03     X        X                 X
AROCOM                    04/07/76 16:29:12
```

```
                                          TEST CASE USAGE REPORTING SYSTEM

( * INDICATES MASTER ELEMENT DIRECTORY TIME AND DATE DO NOT COINCIDE WITH THE ELEMENT USED BY TEST CASE TIME AND DATE)

                                                   RCS2     SABDK1   TBAPTC   WIND1
ELEMENT       VERSION       CREATION CREATION 05/11/76 05/11/76 05/11/76 05/17/76
NAME          NAME            DATE     TIME   14:37:41 10:22:32 11:32:40 16:57:34
XESTAB                      05/05/76 13:31:28
XLIMIT                      09/05/75 09:53:35
XQTOPS                      05/14/76 13:56:13   *X       *X       *X        X
XYZTOE                      09/05/75 09:53:39
```

```
ELEMENT                          T E S T   C A S E   N A M E S   U S I N G   E A C H   E L E M E N T

AABTAR        GNC5

AAINIT        GNC5

AASNSP        NAVE1   NAVE2

ABIND

ABINDI

ABIND1

ACCALC        OMS1    RCS1

ACCGRP        OMS1    RCS1

ACCUPD        GNCORB

ACMAUP        GNC2    GNC3    GNC4    GNC5

ACMEAS        NAVE1   NAVE2

ACSINT        ALT1    ALT2    ENTRY1  ENTRY4  RCS2

ACSUP

ACSUP2

ACS15         ALT1    ALT2    ENTRY1  ENTRY4  RCS2

ACTTVC

ADSEN         ENTRY4  ENTRY5  NAVE1   NAVE2

ADSET         ALT1    ALT2    DECK1   DECK10  DECK11  DECK12  DECK2   DECK3   DECK4   DECK5   DECK6   DECK7   DECK8
              DECK9   ENTRY1  ENTRY3  ENTRY4  ENTRY5  GNCORB  GNC1    GNC2    GNC3    GNC4    GNC5    LAUNCH5 LAUNCH4
              LAUNCH3 LAUNCH2 LAUNCH1 NAVE1   NAVE2   NAVL1   OMS1    ORBIT1  ORBIT2  RCS1    RCS2    SRBDK1  TBAPTC
              WIND1

ADSOP         ENTRY4  ENTRY5  NAVE2

AEPHEM

AERALT        ALT1    ALT2

AERAUX        ENTRY1  ENTRY4  RCS2

AERO          ENTRY1  ENTRY4  RCS2

AGIN          GNC2    GNC3    GNC4    GNC5
```

```
ELEMENT                            TEST CASE NAMES USING EACH ELEMENT

WRITEX

XDATE          DECK1   NAVE2   ORBIT1

XESTAB         ORBIT1  ORBIT2

XLIMIT         ALT1    ALT2    OMS1    ORBIT2

XQTOPS         ALT1    ALT2    DECK1   DECK10  DECK11  DECK12  DECK2   DECK3   DECK4   DECK5   DECK6   DECK7   DECK8
               DECK9   ENTRY1  ENTRY3  ENTRY4  ENTRY5  GNCORB  GNC1    GNC2    GNC3    GNC4    GNC5    LAUNCH4 LAUNCH3
               LAUNCH2 LAUNCH1 NAVE1   NAVE2   NAVL1   OMS1    ORBIT1  ORBIT2  RCS1    RCS2    SRBDK1  TBAPTC  WIND1

XYZTOE         ALT1    ALT2    DECK1   DECK12  GNCORB  GNC1    GNC4    GNC5    LAUNCH5 OMS1    ORBIT1  ORBIT2
```

E6. REFERENCES

1. UNIVAC Programmer Reference Manual, UNIVAC Publication 4144 (Rev.3), 1973.

2. CAD Procedures Manual-JSC, Part 20, October 1973.

C3