AD-A050 845 HARRY DIAMOND LABS ADELPHI MD F/G 9/2
EXTENDING AND INTERFACING THE MSEP SEMICONDUCTOR DAMAGE DATA BA--ETC(U)
DEC 77 C P RUZIC
UNCLASSIFIED HDL-TR-1821 NL

1 OF 2
AD
A050 845

HDL-TR-1821

Extending and Interfacing the MSEP Semiconductor Damage Data Bank for Analysis and Retrieval by DAMTRAC

December 1977

THIS WORK WAS SPONSORED BY THE DEPARTMENT OF THE ARMY UNDER PROJECT NO. 1W162118AH75/CA, MULTIPLE SYSTEMS EVALUATION PROGRAM.

U.S. Army Materiel Development
and Readiness Command
HARRY DIAMOND LABORATORIES
Adelphi, Maryland 20783

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: HDL-TR-1821

2. GOVT ACCESSION NO.:

3. RECIPIENT'S CATALOG NUMBER:

4. TITLE (and Subtitle): Extending and Interfacing the MSEP Semiconductor Damage Data Bank for Analysis and Retrieval by DAMTRAC.

5. TYPE OF REPORT & PERIOD COVERED: Technical Rep[redacted]t.

6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s): Charles P./Ruzic

8. CONTRACT OR GRANT NUMBER(s): DA: 1W162118AH75/CA

9. PERFORMING ORGANIZATION NAME AND ADDRESS: Harry Diamond Laboratories, 2800 Powder Mill Road, Adelphi, MD 20783

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Program Ele: 6.21.18.A

11. CONTROLLING OFFICE NAME AND ADDRESS: US Army Materiel Development and Readiness Command, Alexandria, VA 22333

12. REPORT DATE: Dec[redacted]77

13. NUMBER OF PAGES: 103

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):

15. SECURITY CLASS. (of this report): UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report):

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report):

18. SUPPLEMENTARY NOTES:

HDL Project: X756E2 DRCMS Code: 61211811H7500
This work was sponsored by the Department of the Army under project No. 1W162118AH75/CA, Multiple Systems Evaluation Program.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number):

Semiconductor model parameters
Semiconductor damage data
Semiconductor data bank
Damage data bank
TRAC parameters
DAMTRAC

20. ABSTRACT (Continue on reverse side if necessary and identify by block number):

Presented are four groups of programs to update, maintain, and list the diode and transistor data bases for use with circuit analysis codes. The data bases are specifically designed to work with the DAMTRAC code, but can work with other circuit analysis codes, too. Included in the data bases are the standard TRAC parameters, references to the sources of these parameters, and damage parameters with individual source references.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(14) (6) (9) (10) (16) (17) CA (11) (12) 105p.

163 050

CONTENTS

CONTENTS (Cont'd)

APPENDICES

FIGURES

1. INTRODUCTION

Electromagnetic pulse effects of nuclear weapons deployed at high altitudes (HEMP) can seriously degrade tactical weapon and communication systems vitally needed by the field Army prepared to fight a conventional and nuclear war. The Multiple Systems Evaluation Program (MSEP) was established to determine both the vulnerability and the means for hardening a large number of these Army tactical systems to a HEMP environment. An essential step in the program is to develop analytic tools (such as computer programs for predicting transient data and system response) to evaluate system susceptibility to HEMP. These computer programs have been gathered into an applications package titled "Generic Assessment Methods for a Priori Hardening of Systems" (GAMPHS).[1] The GAMPHS application for the vulnerability and hardness assessment of systems covered by the MSEP uses the programs described in this report in addition to other computer programs (fig. 1).

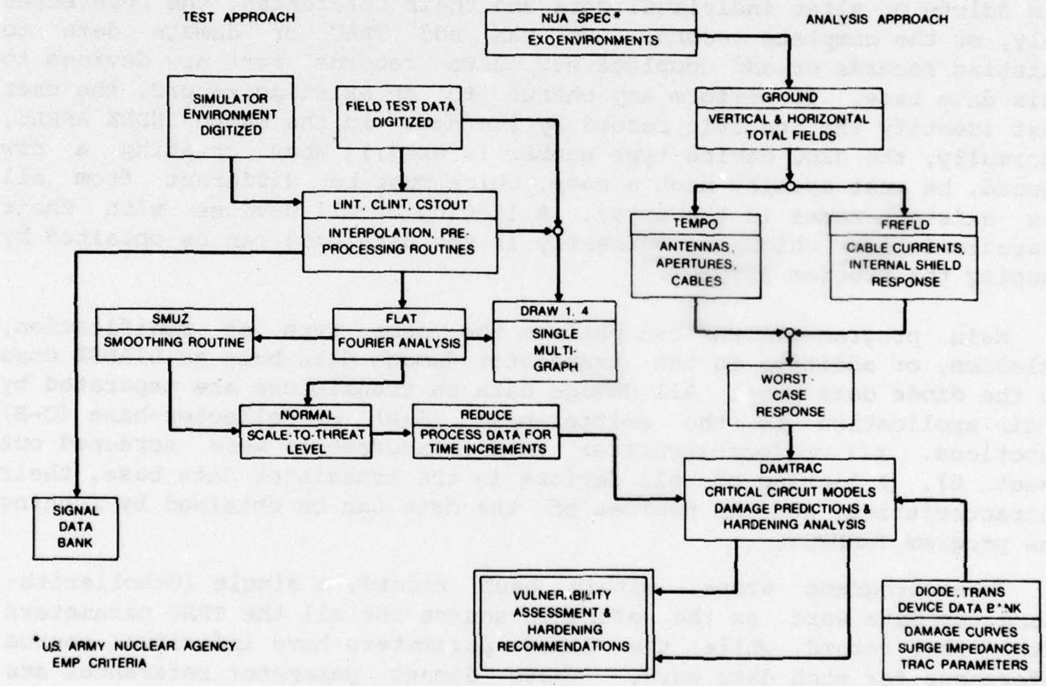

Figure 1. Application of generic assessment methods for a priori hardening of systems (GAMPHS) in vulnerability and hardness assessment.

---

[1] *George Gornak et al, EMP Assessment for Army Tactical Communications Systems: Transmission Systems Series No. 1, Radio Terminal Set AN/TRC-145 (U), Harry Diamond Laboratories TR-1746 (February 1976). (SECRET)*

This report describes four interrelated groups of computer programs that are used to update the diode and transistor data bases for use with DAMTRAC and to point out the contents of these data bases. These bases currently contain, in addition to standard Transient Radiation Analysis by Computer (TRAC) parameters (sect. 4), measured and calculated damage parameters (sect. 4), with references on the original source of each datum. This manual describes the purpose, applications, limitations, and possible future enlargements of the semiconductor data base. It is intended to be useful to scientists, engineers, and programmers working on the assessment of damage to circuits, with discrete semiconductor components, from an EMP environment. It can be of use also to designers and program managers in this subject as a readily available source of EMP damage testing data, with information on the sources of these data.

Main program DTABSE and its associated subroutines are used to update or alter the diode damage data base, residing on mass storage at a Control Data Corp. (CDC) 6600 computer, as required by the user. He can delete or alter individual data and their references, the references only, or the complete record. He can add TRAC or damage data to existing records or add complete new data records for new devices to this data base. To perform any change to an existing record, the user must identify the specific record by its name in the NAMED INDEX ARRAY. (Normally, the JEDC device type number is used.) When creating a new record, he must specify such a name, which must be different from all the existent names in the array. A listing of all devices with their characteristics which are currently in the data base can be obtained by running the program TSTUPDT.

Main program DTATRNS can perform the same type of modification, deletion, or addition to the transistor damage data base as DTABSE does to the diode data base. All damage data on transistors are separated by their application to the emitter-base (E-B) or collector-base (C-B) junctions. *All collector-emitter (C-E) measurements were* screened out (sect. 6). A listing of all devices in the transistor data base, their characteristics, and the sources of the data can be obtained by running the program TRNUPDT.

These programs store, within each record, a single 10-hollerith-character data word as the reference source for all the TRAC parameters within that record, while the damage parameters have individual source references for each data word. These damage parameter references are encoded into three octal characters. The detailed source for each of these octal references is given in appendix A.

Appendix B lists the main programs. Detailed descriptions of all the subroutines required to support these main programs are given in appendix C.

2. PURPOSE AND USES

The organizations and people working on the analysis of electrical systems in a nuclear EMP environment are trying to determine the response of discrete semiconductor devices within circuits exposed to this environment. Several organizations are measuring the characteristics of semiconductor devices being subjected to electrical pulses similar to those produced by an EMP. Unfortunately, standard test procedures for this analysis do not exist, so the sources of data and the procedures used may have to be evaluated before the data themselves can be applied in a specific instance. Furthermore, no coherent model uses all these measurements in the transient analysis circuit codes.

We are attempting to concentrate all the available data on damage testing of semiconductors into a single data base, with a uniform format, which can be used by existing circuit analysis codes. This document describes how we have created a data base on mass storage of a CDC 6600 that is in a format usable by DAMTRAC. (DAMTRAC[2] is a version of the TRAC[3] circuit analysis code, which has been modified by the Harry Diamond Laboratories (HDL) to accept free format inputs and, more importantly, has a model for determining the onset of damage in a semiconductor during a transient analysis run.)

3. PROGRAM RELATIONSHIPS AND FLOWCHART SYMBOL DEFINITIONS

Figure 2 defines the symbols used in the detailed logical flowcharts.

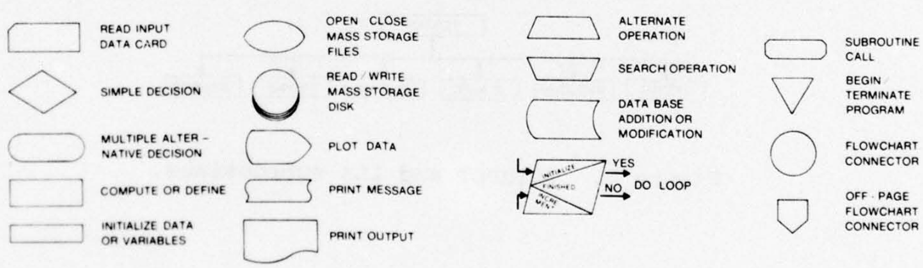

Figure 2. Flow chart symbol definitions for detailed logical flowcharts.

---

[2]*G. Baker, A. McNutt, B. Shea, and D. Rubenstein, Damage Analysis Modified TRAC Computer Program, Harry Diamond Laboratories TM-75-6 (May 1975).*

[3]*Transient Radiation Analysis by Computer Program (TRAC), Autonetics Division of North American Rockwell (June 1968).*

Figures 3 to 6 show how the four programs relate to each other.

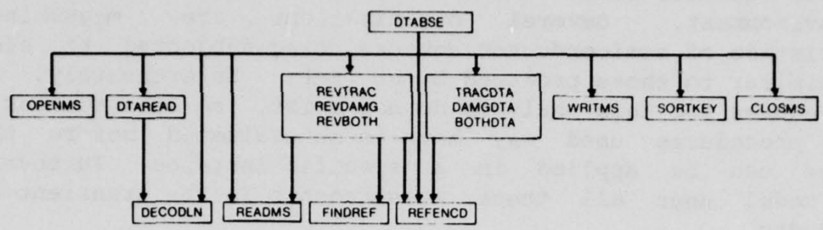

Figure 3. DTABSE and its subroutines.

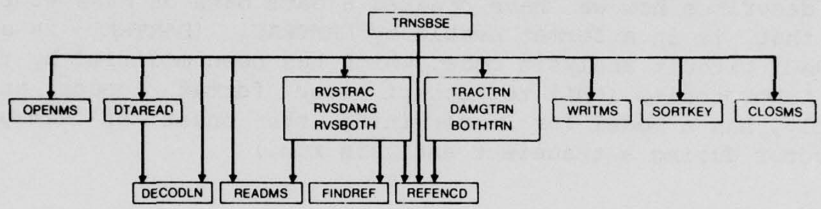

Figure 4. TRNSBSE and its subroutines.

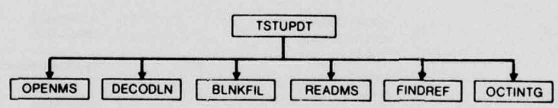

Figure 5. TSTUPDT and its subroutines.

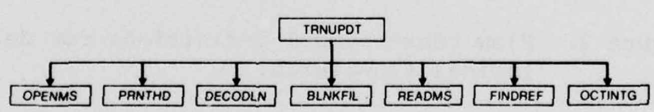

Figure 6. TRNUPDT and its subroutines.

4. DATA-BASE STORAGE FORMATS

4.1 Diode Data Base

For each record, these parameters apply:

Word	Parameter
1	Reverse saturation current
2	Diode proportionality constant
3	Leakage resistance
4	Junction capacitance at zero bias
5	Diffusion voltage
6	Diode time constant
7	Ambient photo current
8	Breakdown voltage
9	Reverse surge impedance (0.1-μs pulse width) (if data are available)
10	10-character reference source for first nine data words
11	Forward bulk resistance of diode
12	Reverse surge impedance for 1-μs square pulse
13	Reverse surge impedance for 10-μs square pulse
14	Measured reverse damage constant ($\tau < 50$ ns), $K = P\tau$
15	Measured reverse damage constant ($\tau > 50$ ns), $K = P\tau^{\frac{1}{2}}$
16	Measured forward damage constant by using $K = P\tau$
17	Optional damage constant data word
18	Optional damage constant data word
19	Optional damage constant data word
20	Optional damage constant data word
21	Optional damage constant data word
22	Optional damage constant data word

Words 1 to 9 are standard TRAC parameters for diodes. Words 11 to 16 have the reference for the source of the data encoded into the three least-significant octal characters of the data word. Words 17 to 22 have the reference encoded in the same fashion, when data exist for them.

4.2 Transistor Data Base

For each record, these parameters apply:

Word	Parameter
1	Normal common emitter current gain
2	Inverse common emitter current gain
3	Emitter time constant
4	Collector time constant
5	Collector reverse saturation current
6	C-B proportionality constant in exponent
7	C-B junction capacitance at zero bias
8	C-B junction diffusion potential
9	C-B leakage resistance
10	E-B reverse saturation current
11	E-B proportionality constant in exponent
12	E-B junction capacitance at zero bias
13	E-B junction diffusion potential
14	E-B leakage resistance
15	Primary photocurrent for C-B junction
16	Primary photocurrent for E-B junction
17	C-B breakdown voltage
18	E-B breakdown voltage

Word	Parameter
19	C-B reverse surge impedance (0.1-μs pulse width)
20	E-B reverse surge impedance (0.1-μs pulse width)
21	10-character reference source for first 20 data words
22	C-B junction forward bulk resistance
23	E-B junction forward bulk resistance
24	C-B reverse surge impedance (1-μs pulse width)
25	C-B reverse surge impedance (10-μs pulse width)
26	E-B reverse surge impedance (1-μs pulse width)
27	E-B reverse surge impedance (10-μs pulse width)
28	C-B reverse biased damage constant ($\tau < 50$ ns), $K = P\tau$
29	E-B reverse biased damage constant ($\tau < 50$ ns), $K = P\tau$
30	C-B reverse biased damage constant ($\tau > 50$ ns), $K = P\tau^{1/2}$
31	E-B reverse biased damage constant ($\tau > 50$ ns), $K = P\tau^{1/2}$
32	C-B forward biased damage constant, $K = P\tau$
33	E-B forward biased damage constant, $K = P\tau$
34	Optional C-B damage constant
35	Optional E-B damage constant
36	Optional C-B damage constant
37	Optional E-B damage constant

Words 1 to 20 are standard TRAC parameters for transistors. Words 21 to 33 have the reference for the source of the data encoded into the three least-significant octal characters of the data word. Words 34 to 37 have the reference encoded in the same fashion, when data exist for them.

5. PROGRAM DESCRIPTIONS

5.1 DTABSE

5.1.1 Flowchart

Figure 7 shows the DTABSE detailed logical flowchart (p. 13).

5.1.2 Variable Definitions

The DTABSE variables are defined as follows:

Variable	Definition
BOTHPRM	Array of new TRAC and damage data to be read from cards
CHANGE	Variable containing "CHANGE" for checking options
CHNGPRM	Array of data to be read from input mass storage unit
DAMG	Variable containing "DAMAGE" for checking options
DAMGPRM	Array of new damage data to be read from cards
DELETE	Variable containing "DELETE" for checking options
I, II ,IJ, J	Dummy DO loop indices
IBLANK(BLANK)	Variable containing 10 blank characters to create blank filled output words
IBOTH	Variable containing BOTH for checking options
ICNT	Variable containing number of words in record to be written to mass storage file
ID	Record of ID for comparison with identifiers in named index array
IDELETE	Count of number of records deleted from input mass-storage file on this run
IDFLG	Formal parameter for subroutine DTAREAD
IDXTRA	Record of ID on runs where only references are to be changed

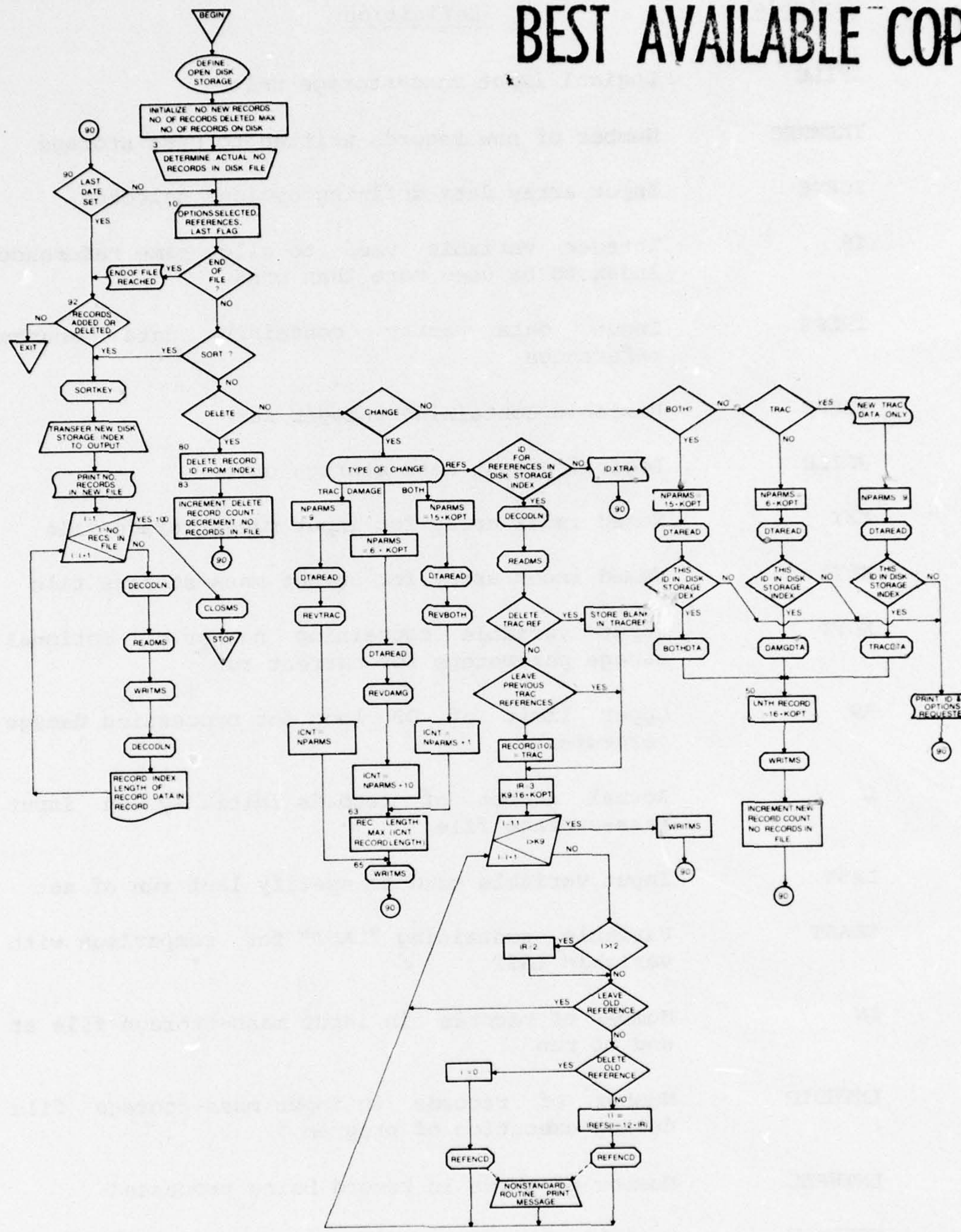

Figure 7. DATABSE detailed logical flowchart.

Variable	Definition
IFILE	Logical input mass-storage unit
INEWREC	Number of new records written to mass storage
IOPTS	Input array data defining options selected
IR	Integer variable used to allow same reference index to be used more than once
IREFS	Input data array containing data source references
IZERO	Variable containing integer zero
JFILE	Logical output mass-storage unit
KEY	Named index array for input mass-storage file
KEY1	Named index array for output mass-storage file
KOPT	Input variable containing number of optional damage parameters for current run
K9	Upper limit of DO loop for processing damage parameters
L	Actual number of records initially on input mass-storage file
LAST	Input variable used to specify last run of set
LLAST	Variable containing "LAST" for comparison with variable LAST
LN	Number of records in input mass-storage file at end of run
LNTHDIO	Number of records in input mass-storage file during execution of program
LNTHREC	Number of words in record being processed
NEGINDF	Variable containing octal 40007777777777777000 used when no datum is available for variable

Variable	Definition
NPARMS	Number of input time and data words to be read from cards
RECORD	Storage array for data to be written to mass-storage file
REFS	Variable containing "REFS" for checking options
SORT	Variable containing "SORT" for checking options
TRAC	Variable containing "TRAC" for checking options
TRACPRM	Array for new TRAC data to be read from input data cards
TRACREF	Variable containing 10 character references of source of TRAC data in record
XNEGIND	Real variable name equivalenced to NEGINDF

5.1.3 Input Data Formats

The diode reference data card has this format and content:

Card	Column	Variable	Definition
1	1	KOPTS	Number of optional K's included (5 max)
	2 to 7	IOPTS(1)	NEW, CHANGE, DELETE, or SORT
	8 to 11	IOPTS(2)	TRAC, DAMG, BOTH, or REFS (on CHANGE only)
	12 to 21	TRACREF	
	22 to 24	Octal R_F	Bulk reference
	26 to 28	Octal Z_{rev}	Surge reference
	30 to 32	Octal $K_{meas,rev}$ ($<$ 50 ns)	Reference
	34 to 36	Octal $K_{meas,rev}$ ($>$ 50 ns)	Reference

Card	Column	Variable	Definition
1	38 to 40	Octal $K_{meas,forward}$	Reference
	42 to 44	Octal K_1 optional	Reference
	46 to 48	Octal K_2 optional	Reference
	50 to 52	Octal K_3 optional	Reference
	54 to 56	Octal K_4 optional	Reference
	58 to 60	Octal K_5 optional	Reference
	65 to 69	LAST	Flag for last data set, when punched
	71 to 80	IDXTRA	Variable used only when references are changed

The diode data card has this format:

Card	Column	TRAC	DAMAGE	BOTH
1	1 to 10	ID	ID	ID
	11 to 20	TRAC(1)	$Z_{forward}$ (surge)	TRAC(1)
	21 to 30	TRAC(2)	Z_{rev} (1 μs)	TRAC(2)
	31 to 40	TRAC(3)	Z_{rev} (10 μs)	TRAC(3)
	41 to 50	TRAC(4)	$K_{meas,rev}$ (<50 ns)	TRAC(4)
	51 to 60	TRAC(5)	$K_{meas,rev}$ (>50 ns)	TRAC(5)
	61 to 70	TRAC(6)	$K_{meas,forward}$	TRAC(6)
	71 to 80	TRAC(7)	K optional	TRAC(7)
2	1 to 10	TRAC(8)	K_1 optional	TRAC(8)
	11 to 20	TRAC(9)	K_2 optional	TRAC(9)
	21 to 30		K_3 optional	$Z_{forward}$ (surge)

Card	Column	TRAC	DAMAGE	BOTH
	31 to 40		K_4 optional	Z_{rev} (1 μs)
	41 to 50		K_5 optional	Z_{rev} (10 μs)
	51 to 60			$K_{meas,rev}$ (<50 ns)
	61 to 70			$K_{meas,rev}$ (>50 ns)
	71 to 80			$K_{meas,forward}$
3	1 to 10			K optional
	11 to 20			K_1 optional
	21 to 30			K_2 optional
	31 to 40			K_3 optional
	41 to 50			K_4 optional
	51 to 60			K_5 optional

5.1.4 Program Explanation

Overview.--DTABSE starts by opening the disk storage file and then searching the named index array to determine the actual number of data items on the existing file. It then reads the first input data card for each device record to be altered. It checks the values on this card to determine which options were selected. DTAREAD is then called to read the actual data values to be added or the new values for data items to be altered. The ID in this data set is compared with ID's in the named index to ascertain if that device is in the current disk data file.

This main program then calls TRACPRM, DAMGPRM, or BOTHPRM, as appropriate, to process new records; or, alternatively, it calls REVTRAC, REVDAMG, or REVBOTH, as appropriate, to process changes in existing records. All possible changes, deletions, or additions to the damage references are effected by calling subroutine REFENCD to encode a three octal character reference into the three least-significant octal digits of the specified real data variable. This processed data record is then written to the extended disk file. After all changes to the file have been implemented, subroutine SORTKEY is called to resort the names in the named key index for the existing data file, on runs where records have been added or deleted.

If existing records have only been modified, the run then terminates. Otherwise, the program writes a new cycle on the disk storage file with the records written as they are ordered in the sorted named index array. The program then prints on the system output, for each record in the file: the named index, the length of the record, and the octal representation of the contents of the record. The program then ends.

Details.--DTABSE starts by defining and opening the mass-storage units, initializing the variables, and then, in the DO 3 loop, determining the actual number of data records currently resident in the existing file. The diode reference data card (sect. 5.1.3) is then read for the data set to be processed, at statement 10.

The end of file on input is tested before the options selected on this data card are analyzed. (This is a safeguard in case the LAST variable was not properly encoded on the data card for the last data set.) If an end of file was found, an appropriate message is printed, and the program goes to statement 92. The allowable options are as follows: NEW is used to add a record to the data base. CHANGE is used to modify an existing record. SORT is used to merely reorder the record identifiers in the named index array and then to rewrite the data on a new cycle of the mass-storage file. The option DELETE, which must be used with CHANGE, causes the entire specified record to be deleted from the file. The option TRAC is to be used when only TRAC data are to be processed in the record under consideration; similarly for DAMAGE, only damage data are to be processed. For the option BOTH, TRAC and damage data are to be processed for that record. Lastly is the option REFS, which is to be used when only the references in an existing record are to be changed. When REFS is selected, the record identifier must be punched in the field IDXTRA, rather than the first field of the succeeding data card; thus, only a single data card is required for this option and data set.

If the option NEW is selected, the number of new data variables to be read from data cards is calculated and DTAREAD is called to read these data. After this call, the formal parameter IDFLG, which was set within DTAREAD, is tested to see that the current record identifier does not already exist on the storage file. If IDFLG indicates that this identifier is already on the storage file, a message is printed, and processing goes to statement 90. Otherwise, TRACDTA, DAMGDTA, or BOTHDTA is called, as appropriate, to create the new data record. Then at statement 50, the length of this new record is defined, and the record is written to the existing, extended mass-storage file. The new record count and the count of the number of records in the file are incremented, and the processing goes to statement 90.

If CHANGE is selected, the program first checks whether REFS also was selected. If so, the program goes to statement 70. Otherwise, the number of input data words, NPARMS, is calculated as a function of KOPT and of which type of data is to be read. DTAREAD is then called to read the input data parameters for the current record to be changed, to determine the length of the existing record, and to test that the record identifier specified already exists. If it does not exist, an error message is printed, and processing goes to statement 90. If this error is not found, one of the following subroutines, as appropriate, is called: REVTRAC, REVDAMG, or REVBOTH. The subroutine changes the corresponding type of data within the specified record as required by the input data cards.

The next several statements, ending at statement 63, determine the length of the revised record. At statement 65, this revised record is written to the existent, extended mass-storage file, and then the program goes to statement 90.

At statement 70 is begun the treatment of reference changes only. The named index is searched to find the record specified by IDXTRA. If no match is found, an error message is printed, and the program goes to statement 90. Otherwise, subroutine DECODLN is called to determine the length of the existent specified record. This record is then read from the mass-storage file. Each reference field on the diode reference data card is then checked. A minus zero in any field causes the existing reference to be deleted.* (For a minus zero, the TRAC reference is replaced by a blank field, and a three octal digit *reference is replaced by three octal zeros.)* If the reference field on the data card is blank, the CDC 6600 stores a zero. Thus, we compare the fields read with zero; if a match occurs, the existing reference is left as it was, and processing continues on the next reference. If any reference field on the data card contains a value that is neither blank nor minus zero, then the existing corresponding reference is changed to the value given on the data card. This reference is changed, except for the TRAC reference, by calling subroutine REFENCD. If an error condition is found in this subroutine, an abnormal return occurs that causes an error message to be printed, but does not change the reference; then the program flows normally. After these changes to the references, the current corrected record is rewritten to the mass-storage file. The program then goes to statement 90.

If the option DELETE is selected for this record, implementation begins at statement 80. The DO 83 loop deletes the specified identifier from the named index array. If this specified identifier is not found, an error message is printed, and the processing

---

**The description in the rest of this section applies to TRNSBSE, also.*

flow jumps to statement 90. After this specified identifier is deleted, the count of the number of records deleted is incremented by one, and the count of the number of records in the file is decremented by one. The program then continues on at statement 90.

Statement 90 is the junction point in the processing flow for each record. After the different branches corresponding to different selected options have been executed, the program flow comes to this statement and tests whether the diode data card contained the flag LAST; if not, it jumps back to statement 10 to start on the next data set. If LAST was set, the program continues on at statement 92.

At statement 92, both the new record count and the deleted record count are tested. If these counts are both zero, the program ends, and any changes in the run were performed onto the extended existing storage file. Otherwise, subroutine SORTKEY is called to resort the named index array for the existing data file. This resorting is performed according to the standard CDC 6600 FORTRAN collating sequence. The even-indexed elements of this array are then copied to another array to be used as the named index for a new cycle of the data on the mass-storage file.

This new data file cycle is written to avoid problems with the record manager that are possible if the named index was merely resorted. A secondary reason is to insure that the devices were stored in a known order, so that the file contents could be printed out in a neat, logical order by other independent programs.

The DO 100 loop, for each record, calls subroutine DECODLN to determine the record length of the specified record on the existing data file, reads that record, and then writes that record using the new named index onto a new cycle of the mass-storage file. The named index, the record length, and the contents of the record, in octal format, are then printed onto the system output. After the end of this loop, both logical mass-storage files are closed, and the program ends.

5.1.5 <u>Output Data Format</u>

At the beginning, after the program searches the named index, the number of records then on the mass-storage file is printed. Just before the end of the program, and only if the data file has had records added or deleted or if a sort has been requested, the program prints the number of records on the storage file at that time. It prints out the contents of the file, using format 950, given in the program listing.

For each record is printed (1) the record identifier from the named index, (2) the record length in the extended existing file, (3) the record length on the new cycle of the file (these should be identical), and (4) the complete contents of this record in octal format (20 octal characters for each CDC 6600 60-bit word).

Under erroneous conditions, the program prints out six possible error messages:

- END OF FILE ENCOUNTERED WHILE TRYING TO READ NEXT DATA SET
  JOB ENDED BASED ON DATA ALREADY READ

- REFERENCE OUT OF RANGE FOR DATA ITEM NO. (I5) FOR DEVICE (A10) VALUE IS (E12.3) REF IS (O3)
  OLD REFERENCE NOT CHANGED

- NO MATCH FOUND IN NAMED INDEX ARRAY FOR DEVICE TYPE WHICH WAS TO ONLY HAVE REFERENCES CHANGED, ID WAS (A10)
  THESE CHANGES WERE THEREFORE IGNORED & PROCESSING CONTINUES

- EOF NOT REACHED AFTER (I5) DEVICE TYPES

- ATTEMPTED TO CREATE A NEW DATA REC WITH THE SAME ID AS AN OLD REC, THIS NEW DATA SET WAS IGNORED & PROCESSING CONTINUES
  ID WAS (A10) IOPTS(1) WAS (A6) IOPTS(2) WAS (A4)

- ATTEMPTED TO CHANGE AN EXISTING RECORD IN THE DATA FILE, AND THE SPECIFIED 'ID' WAS NOT FOUND IN THE NAMED INDEX ARRAY
  THIS DATA SET WAS IGNORED & PROCESSING CONTINUES
  ID WAS (A10) IOPTS(1) WAS (A6) IOPTS(2) WAS (A4)

5.1.6 Sample Run

The deck setup uses these control cards:

```
EMCPR,CM64000,T100
TASK,TNEM71603,PW*****,TRTS. RUZIC
ATTACH,TAPE11,DIODMG,ID=*******.
REQUEST,TAPE12,*PF.
EXTEND(TAPE11)
MAP(PART)
FTN(R=2,L)
LGO.
CATALOG,TAPE12,DIODMG,ID=*******,RP=777.
EXIT.
7/8/9
```

The CATALOGUE card *must be omitted* if there are no additions or deletions to the file. Card 7/8/9 is a standard CDC 6600 end-of-record mark. Numbers 7, 8, and 9 are all punched in column 1.

The deck setup uses these programs:

Program DTABSE
Subroutine REVTRAC with entries REVDAMG and REVBOTH
Subroutine TRACDTA with entries DAMGDTA and BOTHDTA
Subroutine SORTKEY
Subroutine DTAREAD
Subroutine REFENCD
Subroutine DECODLN
Subroutine FINDREF
7/8/9

For this sample run, the deck setup uses the data cards in listing 1 (p. 23). The listings of these programs are in appendix B, and the subroutines are described in appendix C. The reader can ask the author for the program output.

LISTING 1. DTABSE SAMPLE RUN DATA

```
1CHANGEBOTH PREV BASE034 034 060 060 060 020 063
1N277
              5.1E3    .89E0     7.6E3     7.6E3  3.95E-6  1.77E-2   3.7E-4
    2.7E-2    1.2E-1
1NEW   BOTH PREV BASE034 034 060 060 060 022
1N485B
              2.5E1   .463E0    962.E0    532.E0  1.13E-4  5.06E-1   9.6E-3
    .30E0
3CHANGEBOTH PREV BASE034 066 060 060 060 017 002 013
1N645
             1.75E3  2.25E-1     6.9E3     6.9E3  1.49E-4  6.65E-1   6.6E-2
     2.8E0   3.63E0   7.9E-1
1CHANGEDAMG PREV BASE034 062 060 060 060 017
1N746A        .28E0  3.35E-1   3.35E-1   1.91E-3   8.54E0   2.0E-2
2NEW   BOTH PREV BASE034 034 060 060 060 017 002
1N751A
               .7E0  1.85E-1      .7E0      .7E0  4.46E-4   1.99E0   9.3E-3
     1.1E0   3.87E0
1NEW   BOTH 1N1202   062 034 060 060       013
1N1202AR    .452E-9   1.62E0    41.3E9  .130E-10   1.0E0   1.0E-7   1.0E-4
     2.0E2   1.13E3   1.1E-1    1.13E3    2.95E3  2.26E-4   1.01E0
    4.67E0
1NEW   BOTH PREV BASE034 034 060 060 060 022
1N1731A
             1.45E3  2.65E-1    5.35E3    5.35E3  4.03E-4    1.8E0   1.9E-1
     3.2E0
1CHANGEBOTH PREV BASE034 062 060 060 060 017
1N3025B
             3.5E-1   8.5E-2    3.5E-1     6.E-1  8.49E-4    3.8E0   4.4E-2
     1.9E0
0NEW   BOTH PREV BASE034 034 060 060 060
M01054
              4.5E3   1.3E-1     7.1E3     1.5E4  6.72E-6  3.01E-2   6.7E-3
0NEW   BOTH PREV BASE034 034 060 060 060
MS1040
             5.75E1    6.4E0    5.75E1   3.925E1  8.49E-8   3.8E-4   2.1E-6
0NEW   BOTH PREV BASE034 034 060 060 060
1R696735
              1.0E0   2.2E-1     1.0E0     1.0E0  1.13E-3   5.06E0  8.08E-3
0NEW   DAMG PREV BASE062 071 060 060
1N2560        .17E0    4.4E1    4.25E1   1.13E-2   5.06E1
0CHANGEBOTH PREV BASE033 033
PC115
              4.7E1  1.55E-1     4.7E1    1.20E2
0CHANGEDAMG PREV BASE032 032
1N600                 1.6E-2               7.0E-2
0CHANGEDAMG PREV BASE025 025     022

1N1200       5.E-2               1.1E2              6.232E1
0CHANGEDAMG PREV BASE025 025     025
1N1202        1.0E0    6.5E1     6.5E1                 1.4E1
0CHANGEDAMG PREV BASE            022
1N1204A                                             4.611E1
1CHANGEDAMG PREV BASE            013     017
1N1614                                                4.88E0            .38E00
0CHANGEDAMG PREV BASE            017
1N191                                                 5.0E-3
0CHANGEDAMG PREV BASE            022
1N270                                                 2.2E-2
0CHANGEDAMG PREV BASE            022
1N276                                                 5.5E-3
0CHANGEDAMG PREV BASE            022
1N2823B                                               2.49E2
0CHANGEDAMG PREV BASE            017
1N2846B                                                1.5E1
0CHANGEDAMG PREV BASE            017
```

LISTING 1. DTABSE SAMPLE RUN DATA (Cont'd)

```
1N3024                                                   1.9E0
0CHANGEDAMG PREV BASE             017
1N3027B                                                  1.9E0
0CHANGEDAMG PREV BASE             017
1N3033B                                                  1.9E0
1CHANGEDAMG PREV BASE             013 013 017
1N3064                                                   .17E0   1.86E-4      .02E0
1CHANGEDAMG PREV BASE             021     002
1N338                                                   1.83E1               3.24E0
0CHANGEDAMG PREV BASE064 065 003 013 013
1N3600         .397E0   1.7E1                 .18E0      .19E0   3.04E-4
0CHANGEDAMG PREV BASE    025      025
1N4003                 1.65E1                            .78E0
0CHANGEDAMG PREV BASE             013
1N4006                                                   .26E00
0CHANGEDAMG PREV BASE             013 013
1N4148                                                1.94E-2    1.5E-4
0CHANGEDAMG PREV BASE             017
1N4249                                                   2.4E0
1CHANGEDAMG PREV BASE025 025      020     025
1N457            .4E0   6.0E1                            .12E0                .75E0
0CHANGEDAMG PREV BASE             017
1N458                                                    .5E0
1CHANGEDAMG PREV BASE    002      021     002
1N459A                  3.6E0                            .96E0                .96E0
0CHANGEDAMG PREV BASE    024      020
1N459                   8.3E2                            .59E0
0CHANGEDAMG PREV BASE             017
1N461                                                    .24E0
0CHANGEDAMG PREV BASE             017
1N462                                                    5.E-2
1CHANGEDAMG PREV BASE    024      020     002
1N482A                  7.6E2                            .96E0                .96E0
1CHANGEDAMG PREV BASE             020     002
1N537                   1.3E2                            .51E0                .51E0
0CHANGEDAMG PREV BASE             017
1N5364                                                   1.E0
1CHANGEDAMG PREV BASE             012     017
1N538                                                   8.53E0               1.0E00

1CHANGEDAMG PREV BASE    024      020     002
1N540                   9.4E1                            .93E0                .93E0
0CHANGEDAMG PREV BASE             020
1N547                                                   12.1E0
0NEW    DAMG PREV BASE            020
1N646                                                   2.29E0
0CHANGEBOTH PREV BASE025 025 025 007
1N647
              1.72E1    5.2E0                            4.2E0     3.9E0
0CHANGEDAMG PREV BASE             017
1N648                                                    2.8E0
0CHANGEDAMG PREV BASE             007
1N649                                                    2.9E0
0CHANGEDAMG PREV BASE             007
1N658                                                    .92E0
1CHANGEDAMG PREV BASE    012      012     017
1N660                   1.1E1                           2.80E0                .44E0
1CHANGEDAMG PREV BASE             007     020
1N661                                                    .46E0                .41E0
0CHANGEDAMG PREV BASE             022
1N706                                                   .268E0
0CHANGEDAMG PREV BASE    002      002
1N711A                  1.9E0                            2.1E0
1CHANGEDAMG PREV BASE             013 013 017
1N746A                                                   1.6E0   3.2E-3       1.1E0
0CHANGEDAMG PREV BASE             017
```

LISTING 1. DTABSE SAMPLE RUN DATA (Cont'd)

```
1N747A                                                      1.1E0
0CHANGEDAMG PREV BASE              017
1N748A                                                      1.1E0
2CHANGEDAMG PREV BASE              013 013 017 021
1N752A                                                      1.06E1   3.235E-2    1.1E0
    1.2E0
0CHANGEDAMG PREV BASE              017
1N752                                                       1.1E0
1CHANGEDAMG PREV BASE    024       013 013 020
1N753A                    .4E0                              14.8E0   2.34E-2     1.2E0
0CHANGEDAMG PREV BASE              020
1N753                                                       1.2E0
1CHANGEDAMG PREV BASE              013 013 017
1N754A                                                      1.12E0   6.43E-4     .63E0
1CHANGEDAMG PREV BASE              013     017
1N755A                                                      13.3E0               .63E0
1CHANGEDAMG PREV BASE              013 013 017
1N756                                                       2.04E1   6.87E-2     .63E0
0CHANGEDAMG PREV BASE              017
1N756A                                                      .63E0
1CHANGEDAMG PREV BASE              013 013 017
1N758A                                                      6.17E0   2.5E-2      .63E0
0CHANGEDAMG PREV BASE              017
1N758                                                       .63E0
0CHANGEDAMG PREV BASE              020
1N816W                                                      1.5E0
1CHANGEDAMG PREV BASE    024       002     020
1N823                     .79E0                             1.8E0                1.8E0
2CHANGEDAMG PREV BASE          070 012 067 007 020
1N914                     4.E1             .505E0           7.2E-2   1.91E0      9.6E-2
  .8.5E-1

1CHANGEDAMG PREV BASE              013 013 017
1N963B                                                      6.17E0   2.02E-3     1.0E0
1CHANGEDAMG PREV BASE              013     017
1N965B                                                      7.59E0               1.0E0
0CHANGEDAMG PREV BASE              020
1N967B                                                      .73E0
1CHANGEDAMG PREV BASE              013     017
1N973B                                                      4.27E1               1.0E0
1NEW   DAMG PREV BASE062 034 060 060     017                         LAST
1N2991B      1.05E-1     .55E0      .36E0   7.08E-3        3.16E1               1.5E1
0/6/7/8/9*
```

The 0/6/7/8/9 is a multipunch with all punches 0,6,7,8, and 9 punched in column 1. This is required as an end of file when utilizing a Mohawk simulator of a CDC 200 user terminal. The zero punch must be omitted when the deck is run on CDC equipment.

5.2 TRNSBSE

5.2.1 Flowchart

Figure 8 (p. 29) shows the TRNSBSE detailed logical flowchart.

5.2.2 Variable Definitions

All variables in TRNSBSE, except one, are identical in meaning and use with the variables of the same name in DTABSE. The one exception is that LNTHDIO in DTABSE is replaced by LNTHFIL.

5.2.3 Input Data Formats

The transistor reference data card has this format:

Card	Column	Variable
1	1	KOPT
	2 to 7	IOPTS(1)
	8 to 11	IOPTS(2)
	12 to 21	TRACREF
	22 to 24	Octal Ref $R_{forward\ bulk}$ C-B
	26 to 28	Octal Ref $R_{forward\ bulk}$ E-B
	30 to 32	Octal Ref $Z_{rev\ surge}$ C-B
	34 to 36	Octal Ref $Z_{rev\ surge}$ E-B
	38 to 40	Octal Ref $K_{meas,rev}$ (<50 ns) C-B
	42 to 44	Octal Ref $K_{meas,rev}$ (<50 ns) E-B
	46 to 48	Octal Ref $K_{meas,rev}$ (>50 ns) C-B
	50 to 52	Octal Ref $K_{meas,rev}$ (>50 ns) E-B
	54 to 56	Octal Ref $K_{meas,forward}$ C-B
	58 to 60	Octal Ref $K_{meas,forward}$ E-B
	62 to 64	Octal Ref K optional C-B

Column	Variable
66 to 68	Octal Ref K_1 optional E-B
70 to 72	Octal Ref K_2 optional C-B
74 to 76	Octal Ref K_2 optional E-B
77 to 80	LAST

The transistor data card has this format:

Card	Column	TRAC	DAMAGE	BOTH	REFS
1	1 to 10	ID	ID	ID	ID
	11 to 20	TRAC(1)	$R_{forward}$ C-B	TRAC(1)	
	21 to 30	TRAC(2)	$R_{forward}$ E-B	TRAC(2)	
	31 to 40	TRAC(3)	Z_{rev} (1 μs) C-B	TRAC(3)	
	41 to 50	TRAC(4)	Z_{rev} (10 μs) C-B	TRAC(4)	
	51 to 60	TRAC(5)	Z_{rev} (1 μs) E-B	TRAC(5)	
	61 to 70	TRAC(6)	Z_{rev} (10 μs) E-B	TRAC(6)	
	71 to 80	TRAC(7)	K_{rev} (<50 ns) C-B	TRAC(7)	
2	1 to 10	TRAC(8)	K_{rev} (<50 ns) E-B	TRAC(8)	
	11 to 20	TRAC(9)	K_{rev} (>50 ns) C-B	TRAC(9)	
	21 to 30	TRAC(10)	K_{rev} (>50 ns) E-B	TRAC(10)	
	31 to 40	TRAC(11)	$K_{forward}$ C-B	TRAC(11)	
	41 to 50	TRAC(12)	$K_{forward}$ E-B	TRAC(12)	
	51 to 60	TRAC(13)	K_1 optional C-B	TRAC(13)	
	61 to 70	TRAC(14)	K_1 optional E-B	TRAC(14)	
	71 to 80	TRAC(15)	K_2 optional C-B	TRAC(15)	
3	1 to 10	TRAC(16)	K_2 optional E-B	TRAC(16)	
	11 to 20	TRAC(17)		TRAC(17)	
	21 to 30	TRAC(18)		TRAC(18)	
	31 to 40	TRAC(19) = Z_{rev} (0.1 μs) C-B		TRAC(19) = Z_{rev} (0.1 μs) C-B	

Card	Column	TRAC	DAMAGE	BOTH	REPS
	41 to 50	TRAC(20) = Z_{rev} (0.1 μs) E-B		TRAC(20) = Z_{rev} (0.1 μs) E-B	
	51 to 60			$R_{forward}$ C-B	
	61 to 70			$R_{forward}$ E-B	
	71 to 80			Z_{rev} (1 μs) C-B	
4	1 to 10			Z_{rev} (10 μs) C-B	
	11 to 20			Z_{rev} (1 μs) E-B	
	21 to 30			Z_{rev} (10 μs) E-B	
	31 to 40			K_{rev} (<50 ns) C-B	
	41 to 50			K_{rev} (<50 ns) E-B	
	51 to 60			K_{rev} (>50 ns) C-B	
	61 to 70			K_{rev} (>50 ns) E-B	
	71 to 80			$K_{forward}$ C-B	
5	1 to 10			$K_{forward}$ E-B	
	11 to 20			K_1 optional C-B	
	21 to 30			K_1 optional E-B	
	31 to 40			K_2 optional C-B	
	41 to 50			K_2 E-B	

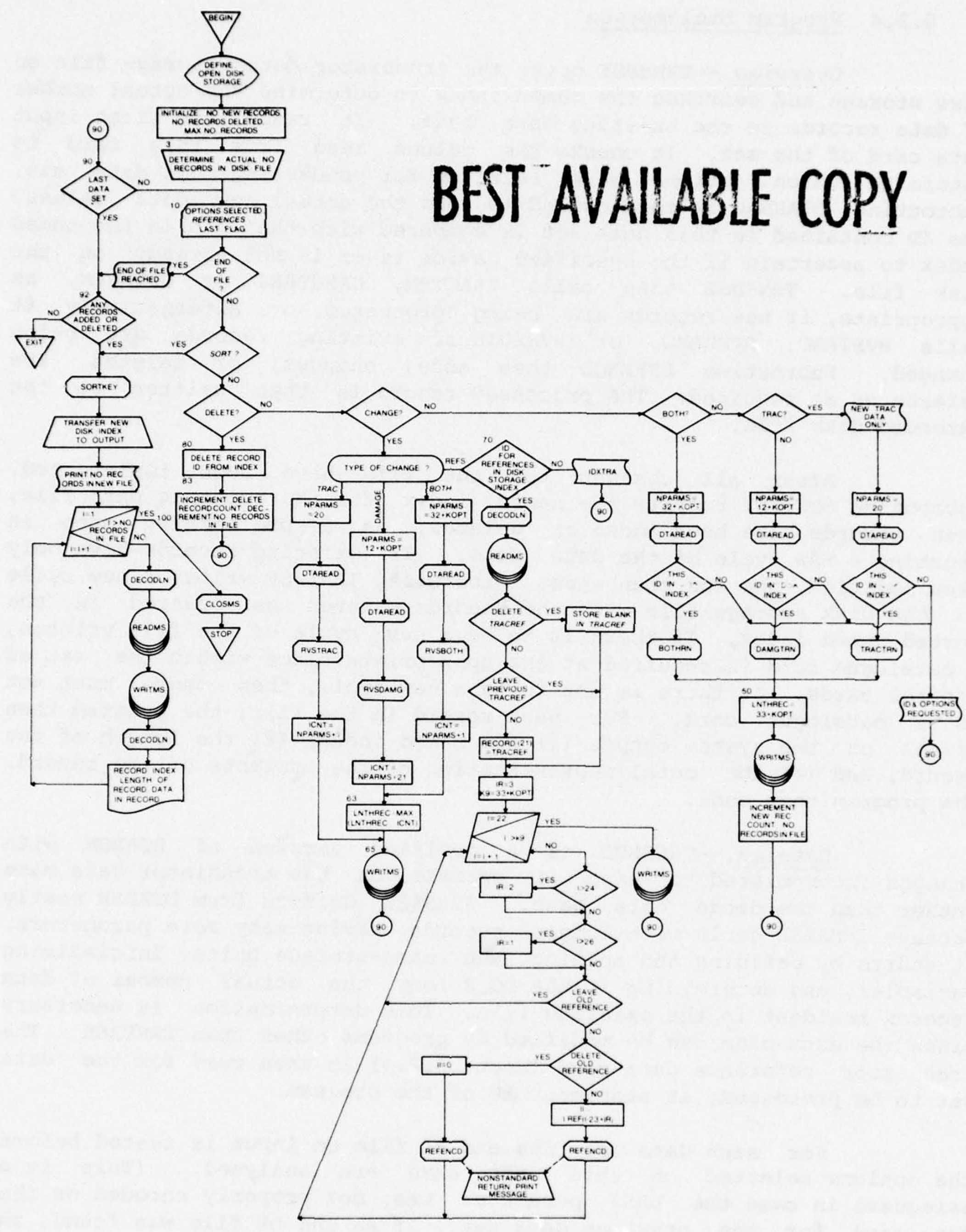

Figure 8. TRNSBSE detailed logical flowchart.

5.2.4 Program Explanation

Overview.--TRNSBSE opens the transistor data storage file on mass storage and searches the named index to determine the actual number of data records in the existing data base. It reads the first input data card of the set. It checks the values read from this card to determine which options were selected for processing this data set. Subroutine DTAREAD is then called to read the actual new data values. The ID contained in this data set is compared with the ID's in the named index to ascertain if the specified device is or is not already on the disk file. TRNSBSE then calls TRACTRN, DAMGTRN, or BOTHTRN, as appropriate, if new records are being processed; or, alternatively, it calls RVSTRAC, RVSDAMG, or RVSBOTH if existing records are being changed. Subroutine REFENCD then adds, changes, or deletes the references as required. The processed record is then written to the extended disk file.

After all changes to the file have been implemented, subroutine SORTKEY resorts the named index for the existing data file, when records have been added or deleted, as a preliminary step in creating a new cycle of the data file. If existing records have only been modified, the run then ends. Otherwise, TRNSBSE writes a new cycle on the disk storage file with the records written as ordered in the sorted named index. If there is to be a new cycle of the file written, a catalogue card is required at the appropriate place within the set of control cards. If there is not to be a new cycle, then there must not be a catalogue card. For each record in the file, the program then prints on the system output (1) the named index, (2) the length of the record, and (3) the octal representation of the contents of the record. The program then ends.

Details.--TRNSBSE is a modified version of DTABSE with changes incorporated to have it operate on the transistor data base rather than the diode data base. TRNSBSE differs from DTABSE mostly because TRNSBSE deals with longer records having many more parameters. It starts by defining and opening the mass-storage units, initializing variables, and determining in the DO 3 loop the actual number of data records resident in the existing file. This determination is necessary since the data base can be modified by programs other than TRNSBSE. The transistor reference data card (sect. 5.2.3) is then read for the data set to be processed, at statement 10 of the program.

For each data set, the end of file on input is tested before the options selected on this data card are analyzed. (This is a safeguard in case the LAST parameter was not properly encoded on the data card for the previous data set.) If an end of file was found, an appropriate message is printed, and the program goes to statement 92.

The allowable options on the transistor reference data card, with one exception, are the same as those for the diode data reference card (sect. 5.1.4). This one exception is that IDXTRA is not used here. Instead, a second data card with the ID must be read regardless of whatever option is selected.

If the option NEW was selected, the number of new data variables to be read from data cards is calculated, and DTAREAD is called to read these data. After this call, the formal parameter IDFLG, which was set within DTAREAD, is tested to see that the current record identifier does not already exist in the storage file. If it does exist, a message is printed, and the processing goes to statement 90. Otherwise, TRACTRN, DAMGTRN, or BOTHTRN is called, as appropriate, to create the new data record. Then at statement 50, the length of this new record is defined, and the record is written to the existing, extended mass-storage file. The new record count and the count of the number of records in the file are incremented, and the processing goes to statement 90.

If the option CHANGE is selected, the program first checks whether the option REFS also was selected. If so, the program goes to statement 70. Otherwise, the number of input data words, NPARMS, is calculated as a function of KOPT and of the type of data to be read. DTAREAD is then called to read the input data parameters for the current record to be changed, to determine the length of the existing record, and to test that the record identifier specified already exists. If it does not exist, an error message is printed, and the processing goes to statement 90. If this error is not found, one of the following subroutines, as appropriate, is called: RVSTRAC, RVSDAMG, or RVSBOTH. These subroutines change the corresponding type of data within the specified record as required by the input data cards.

The next several statements, ending at statement 63, determine the length of the revised record. At statement 65, this revised record is written to the existent, extended mass-storage file, and then the program goes to statement 90.

At statement 70 is begun the treatment of reference changes only. The named index is searched to find the record specified by ID. If no match is found, an error message is printed, and the program goes to statement 90. Otherwise, subroutine DECODLN is called to determine the length of the existent specified record. This record is then read from the mass-storage file. Each reference field on the transistor reference data card is then checked.*

**The description for DTABSE (sect. 5.1.4, Details) beginning with the asterisk (*) applies to TRNSBSE, also.*

5.2.5 Output Data Format

The output data format is the same as that described for DTABSE (sect. 5.1.5). The only difference is that since the records are longer, more words are printed out for each record of TRNSBSE.

5.2.6 Sample Run

The deck setup uses these control cards:

```
EMCPR,CM64000,T35
TASK,TN*******,PW*****,TRTS. RUZIC
ATTACH,TAPE11,TRANS,ID=*******.
FTN(R=2,L)
MAP(PART)
LGO.
7/8/9 (end of record punch)
```

The deck setup uses these programs:

```
Program TRNSBSE
Subroutine RVSTRAC
Subroutine TRACTRN
Subroutine DTAREAD
Subroutine REFENCD
Subroutine DECODLN
Subroutine FINDREF
7/8/9
```

For this sample run, the deck setup uses the data cards in listing 2 (p. 33). The listings of these programs are in appendix B, and the subroutines are described in appendix C. The reader can ask the author for the program output.

LISTING 2. TRNSBSE

```
OCHANGEBOTHPREV BASE  34   34   34   34   60   60   60   60   60   60
2N393
------------------------------------------------------------------------------------------
                                          1.22E+2     2.35E+2     4.40E+1     3.80E+1     2.45E+2
   1.47E+2   1.275E+3   1.275E+3          2.69E-4     3.11E-5     1.20E+0     1.39E-1     8.40E-4
   6.14E-4
ONEW   BOTH           34   34   34   34             60   60   60   60
2N396A
------------------------------------------------------------------------------------------
                                          1.31E+3     1.10E+3     4.25E-1     1.35E+0     1.175E+3
  1.175E+3    1.10E+3    1.40E+3                                  1.42E-1     1.56E-1     1.60E-2
   3.20E-3
ONEW   BOTH           34   34   34   34   60   60   60   60   60   60
2N428M
------------------------------------------------------------------------------------------
                                          4.40E+2     9.78E+2     5.60E-1     1.84E+0     9.75E+2
   9.75E+2    9.78E+2    9.78E+2          4.25E-5     6.51E-5     1.90E-1     2.91E-1     1.30E-2
   3.20E-3
ONEW   BOTH           34   34   34   34   60   60   60   60   60   60
2N466M
------------------------------------------------------------------------------------------
                                          6.80E+2     6.20E+2     1.30E+0     1.85E+0     6.80E+2
   2.41E+2    5.00E+2    2.80E+2          1.42E-4     1.34E-4     6.33E-1     6.01E-1     1.40E-2
   2.70E-3
OCHANGEBOTHPREV BASE  34   34   34   34   60   60   60   60   60   60
2N501A TD
------------------------------------------------------------------------------------------
                                          3.15E+1     4.20E+1     5.10E+0     8.40E+0     3.55E+2
   4.70E+2    2.60E+2    1.01E+3          3.40E-6     4.95E-6     1.52E-2     2.22E-2     3.90E-4
   1.40E-4
OCHANGEBOTHPREV BASE  34   34   34   34   60   60   60   60   60   60
2N705
------------------------------------------------------------------------------------------
                                          7.25E+1     1.65E+2     9.50E-1     2.20E+0     1.50E+2
   3.60E+2    3.40E+2    3.40E+2          3.18E-6     1.13E-6     1.42E-2     5.06E-3     7.90E-5
   3.60E-5
OCHANGEBOTHPREV BASE  34   34   34   34   60   60   60   60   60   60
2N706
------------------------------------------------------------------------------------------
                                          1.65E+1     2.50E+1     2.15E+0     2.45E+0     4.70E+1
   7.30E+1    2.50E+1    2.50E+1          4.25E-6     3.96E-6     1.80E-2     1.77E-2     4.30E-5
   8.80E-5
ONEW   BOTH           34   34   34   34   60   60   60   60   60   60
2N1042RA
------------------------------------------------------------------------------------------
                                          9.75E+1     1.42E+2     1.60E-1     2.95E-1     5.92E+2
  2.625E+2    2.55E+2    5.00E+2          3.11E-4     2.69E-4     1.39E+0     1.20E+0     1.60E-1
   1.40E-1
2CHANGEBOTHPREV BASE  62   34   34   67   60   60   60   60   60   60         17
2N1485
------------------------------------------------------------------------------------------
                                          7.75E+0     1.90E+0     5.75E-1     4.05E-1     6.90E+1
   6.90E+1    0.50E+0    3.30E-1          2.48E-4     2.05E-3     1.11E+0     9.18E+0
   4.50E-2               4.10E+0
2NEW   BOTH           34   34   34   34   60   60   60   60   60   60         17
2N1490
------------------------------------------------------------------------------------------
                                                      1.65E+0     3.00E-2     3.00E-2     2.15E+0
   1.60E+1    1.65E+0    1.65E+0          4.74E-4     8.49E-4     2.17E+0     3.80E+0

4CHANGEBOTHPREV BASE  34   34   34   34   60   60   60   60   60   60         17         25
2N1613
------------------------------------------------------------------------------------------
                                          6.05E+1     7.55E+0     5.50E-1     6.75E-1     6.05E+1
   6.05E+1    7.55E+0    7.55E+0          3.40E-4     6.30E-5     1.52E+0     2.82E-1     1.70E-2
   2.90E-3               2.70E-1                      1.00E+0
2CHANGEBOTH           33   33   33   33   60   60   60   60   60   60         22
2N2857NET
```

```
                                          6.50E+1    1.65E+1    2.40E+0    2.30E+0    6.70E+2
   5.70E+2    1.65E+1    5.50E+1          5.24E-6    5.38E-7    2.34E-2    2.41E-3    2.90E-5
   1.30E-5
OCHANGEBOTHPREV BASE   34   34   34   34   60   60   60   60   60   60
2N2894

                                          4.60E+0    1.60E+0    1.95E+0    2.30E+0    4.60E+1
   9.35E+2   1.305E+1   1.785E+1          7.78E-6    3.82E-6    3.48E-2    1.71E-2    2.30E-5
   2.10E-5
OCHANGEBOTHPREV BASE   34   34   34   34   60   60   60   60   60   60
2N3013

                                          1.51E+1    9.50E+0    1.45E+0    1.65E+0    3.85E+2
   7.50E+2    9.50E+0    9.50E+0          3.82E-6    5.80E-6    1.71E-2    2.59E-2    5.80E-5
   8.40E-5
OCHANGEBOTH            33   33   33   33   60   60   60   60   60   60
2N3375 T

                                          3.70E+3    9.60E-1    3.05E-1    4.50E-1    3.70E+1
   7.90E+1    6.00E-1    6.00E-1          1.56E-4    1.13E-4    6.97E-1    5.06E-1    4.10E-3
   1.80E-3
ONEW    BOTH           34   34   34   34   60   60   60   60   60   60
2N3439

                                          4.75E+3    1.85E+0    2.50E-1    2.95E-1    1.90E+4
   8.50E+4    1.85E+0    1.85E+0          5.38E-6    1.27E-4    2.41E-2    5.70E-1    3.70E-2
   1.30E-2
ONEW    BOTH           34   34   34   34   60   60   60   60   60   60
2N3584

                                          2.15E+3    1.30E+0    8.00E-2    2.60E-1    2.15E+3
   2.15E+3    1.30E+0    4.15E+0                                3.50E-1    2.30E+0    1.50E-2
   3.20E-2
ONEW    BOTH           34   34   34   34   60   60   60   60   60   60
2N5829

                                          9.25E+1    1.80E+1    2.65E+0    1.65E+0    6.25E+2
   1.50E+3    1.80E+1    1.80E+1          2.90E-6    2.05E-6    1.30E-2    9.18E-3    3.70E-5
   1.40E-5
ONEW    BOTH           34   34   34   34   60   60   60   60   60   60
SMB525517

                                                     3.85E+1    9.70E-1    1.57E+0    5.85E+2
   1.21E+3    3.85E+2    6.70E+1          2.05E-1    1.20E-1    8.54E-2    4.40E-2    5.00E-3
   4.10E-4

OCHANGEBOTHPREV BASE   34   34   34   34   60   60   60   60   60   60
CA3018 TD

                                          9.25E+1    1.05E+2    3.70E+0    8.90E+0    2.40E+2
   2.50E+3    1.05E+2   1.825E+2          4.46E-6    1.98E-6    1.99E-2    8.86E-3    1.70E-4
   4.80E-5
ONEW    DANG           32   32   32   32
2N600      2.50E-1    1.00E+0                        6.00E+2               4.30E+2

OCHANGEDANGPREV BASE   32   32   32   32                                                LAST
2N1184 T   2.00E-1    3.00E-1                        1.40E+2               2.00E+2

7/8/9             (end of file punch)
```

5.3 TSTUPDT

5.3.1 Flowchart

Figure 9 shows the TSTUPDT detailed logical flowchart.

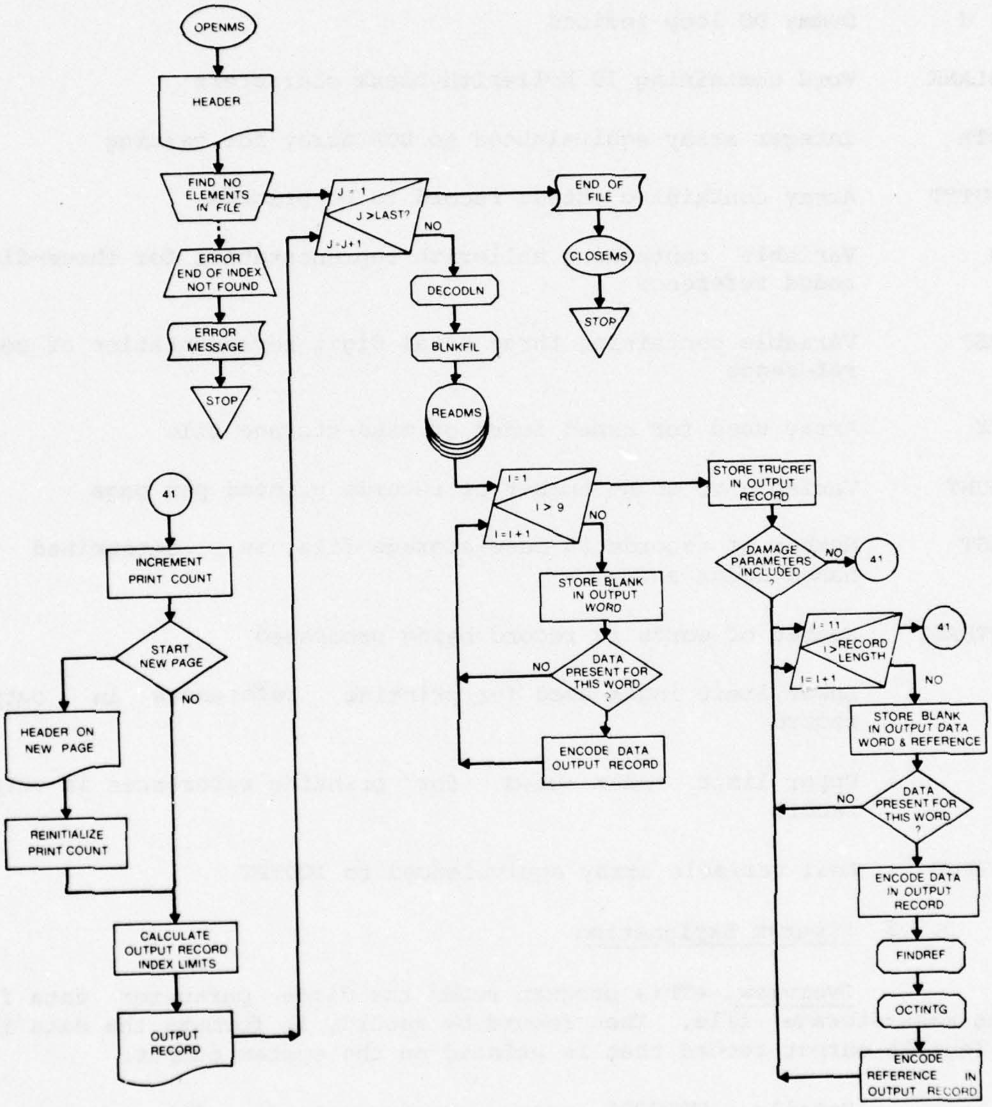

Figure 9. TSTUPDT detailed logical flowchart.

5.3.2 Variable Definitions

The TSTUPDT variables are defined as follows:

Variable	Definition
DTA	Array containing data record read from mass storage
I, J	Dummy DO loop indices
IBLANK	Word containing 10 hollerith blank characters
IDTA	Integer array equivalenced to DTA array for testing
IOUTPT	Array containing output record to be printed
IR	Variable containing hollerith representation for three-digit coded reference
IREF	Variable containing three octal digit representation of coded reference
KEY	Array used for named index of mass-storage file
KOUNT	Variable to count number of records printed per page
LAST	Number of records in mass-storage file, as determined via named index array
LNTHREC	Number of words in record being processed
L1	Lower limit index used for printing references in output record
L2	Upper limit index used for printing references in output record
OUTPUT	Real variable array equivalenced to IOUTPT

5.3.3 Program Explanation

Overview.--This program reads the diode parameter data from the mass-storage file. Then record by record, it formats the data into a legible output record that is printed on the system output.

Details.--TSTUPDT opens the mass-storage file, prints an initial header label on the system output, and then searches the named index to find the number of data records in the file. If the end of the named index is not found within the maximum specified length of this index, an error message is printed, and the job ends.

For each of the records in the file, TSTUPDT first calls DECODLN to determine the actual record length; then it calls subroutine BLNKFIL to blank out the memory area into which the record will be read. It calls READMS to read the data from the mass-storage file. The DO 30 loop processes the first nine words of the record, which are the TRAC parameters. For each of these words, a hollerith blank is initially stored into the corresponding word of the output record. If data exist for that word in the diode data record from the mass-storage file, these data are reformatted into a hollerith representation and stored in the corresponding word of the output record, overwriting the blanks previously stored there. The 10th word of the data record is then stored as the 10th word of the output record. This word contains the TRAC reference or a blank field if there is no TRAC reference. If there are no further data words in the record read from mass storage, the program goes to statement 41. Otherwise, the DO 40 loop is next executed to reformat the damage data words and their references for printing.

For each damage parameter, the output record storage area for the data and the reference is initialized with blank fields. The data word is then checked for the presence of data. If no data exist for the current word, the program skips to the end of the loop at statement 40. Otherwise, the data are encoded into the output record. Then FINDREF is called to obtain the reference, and OCTINTG is called to convert this octal coded reference into a hollerith field. The hollerith-formatted, coded reference is then placed in the output record.

After this loop is completed, the line print count is incremented. If the line limit for a page has been reached, another header label is printed at the top of the next page on the system output, and the line count is reinitialized. At statement 45, the lower and upper index limits for the references are calculated. The named index and the output data record are then printed followed by the three digit references. This process ends the loop at statement 50. After all the records have been processed through this iteration, end-of-file message is printed, and the job ends.

5.3.4 Output Data Format

The TSTUPDT output has a header label at the top of each page. The first line of this header label describes all the standard TRAC parameters and the reference for these parameters. These headings are descriptors for the quantities in the first of the three lines of data for each device, printed below the label. The second and third lines of this label contain the descriptors for the diode damage parameters. The corresponding damage parameters are in the second of the three lines of data for each device. The fourth line of the label merely indicates that the coded reference for each damage parameter is printed directly below the corresponding data word. The coded references are the items in the third line of printed output for each device. The diode data words are defined in section 4.1, and the coded references are defined in appendix A.

If no TRAC parameters are available for a given device, only the device name is printed on the first of the three lines. If no damage parameters exist for a given device, the second and third lines for that data set are blank.

At the end of the job, the message "TEST OF FILE COMPLETED" is printed on the system output.

For the error condition described in section 5.3.3, the message printed is "END OF INDEX NOT FOUND, JOB TERMINATED."

5.3.5 Sample Run

The deck setup uses these control cards:

```
EMCPR,CM64000,T35.
TASK,TN*******,PW*****,TRTS. RUZIC
ATTACH,TAPE11,DIODES,ID=*******.
FTN(R=2,L)
MAP(PART)
LGO.
7/8/9
```

The deck setup uses these programs:

```
Program TSTUPDT
Subroutine OCTINTG
Subroutine BLNKFIL
Subroutine DECODLN
Subroutine FINDREF
0/6/7/8/9
```

Card 0/6/7/8/9 is the end-of-file punch.* The listings of these programs are in appendix B, and the subroutines are described in appendix C. The reader can ask the author for the program output.

5.4 TRNUPDT

5.4.1 Flowchart

Figure 10 shows the TRNUPDT detailed logical flowchart.

**All punches 0, 6, 7, 8, and 9 must be punched in column 1. A normal CDC 6600 does not require the 0, but the CDC 200 user terminal emulator on a Mohawk 2400 terminal does require it.*

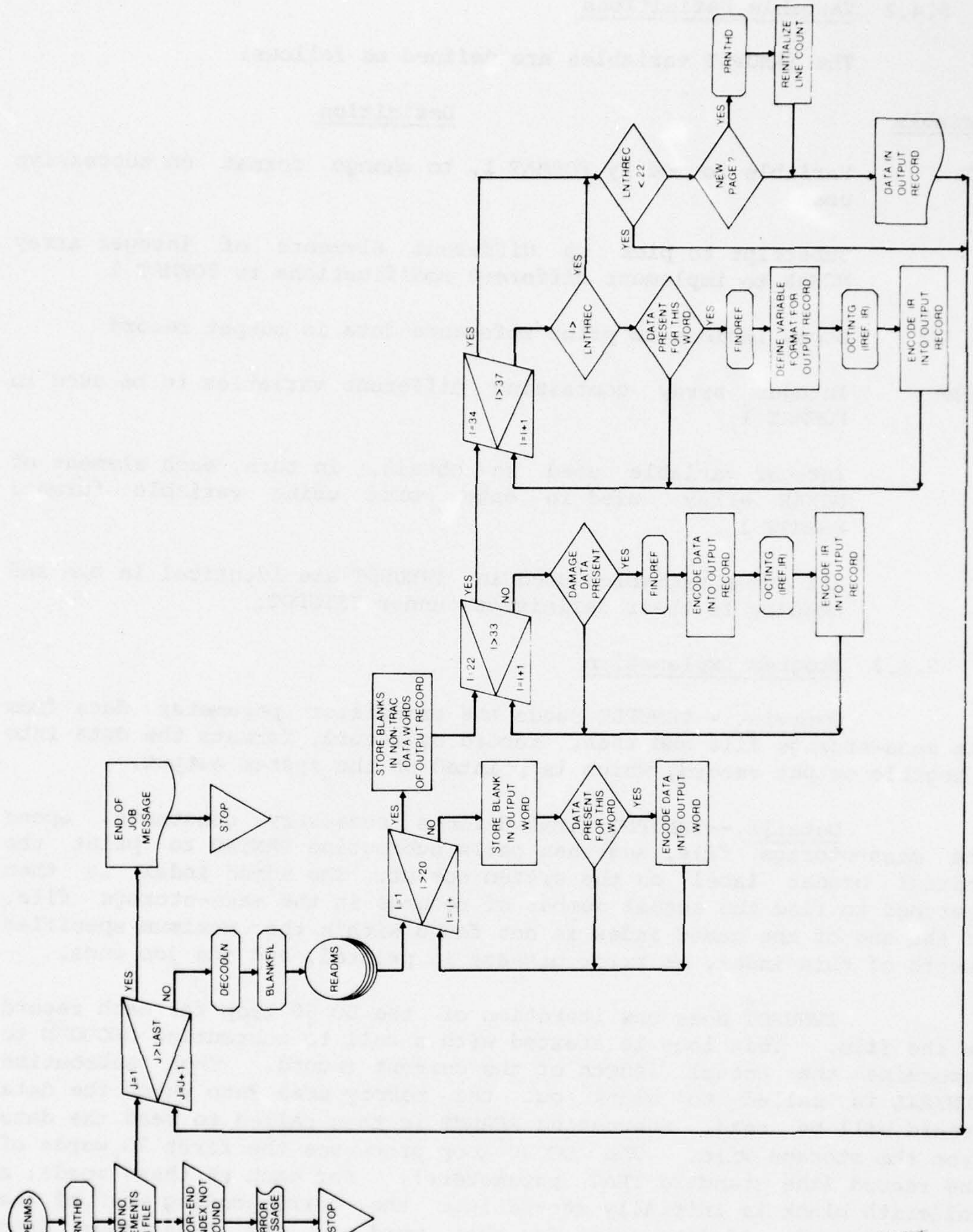

Figure 10. TRNUPDT detailed logical flowchart.

BEST AVAILABLE COPY

5.4.2 Variable Definitions

The TRNUPDT variables are defined as follows:

Variable	Definition
FMT	Variable to modify FORMAT 1, to change format on successive uses
JK	Subscript to pick up different elements of integer array NCHAR to implement different modifications to FORMAT 1
J1, J2	Dummy indices to print reference data in output record
NCHAR	Integer array containing different variables to be used in FORMAT 1
N	Integer variable used to obtain, in turn, each element of NCHAR array; used in data words using variable format: FORMAT 1

All other variables used in TRNUPDT are identical in use and meaning to their definitions under TSTUPDT.

5.4.3 Program Explanation

Overview.--TRNUPDT reads the transistor parameter data from its mass-storage file and then, record by record, formats the data into a legible output record, which is printed on the system output.

Details.--TRNUPDT first defines necessary constants, opens the mass-storage file, and then calls subroutine PRNTHD to print the initial header label on the system output. The named index is then searched to find the actual number of records in the mass-storage file. If the end of the named index is not found within the maximum specified length of this index, an error message is printed, and the job ends.

TRNUPDT does one iteration of the DO 50 loop for each record in the file. This loop is started with a call to subroutine DECODTN to determine the actual length of the current record. Then subroutine BLNKFIL is called to blank out the memory area into which the data record will be read. Subroutine READMS is then called to read the data from the storage file. The DO 30 loop processes the first 20 words of the record (the standard TRAC parameters). For each of these words, a hollerith blank is initially stored into the corresponding word of the output record. If data exist for that word in the transistor data record from the mass-storage file, then these data are reformatted into a hollerith representation and stored in the corresponding word of the output record, overwriting the blank field previously stored there. The 21st word of the data record is then stored as the 21st word of the output record. This word contains the reference for the TRAC data or a blank field if there is no reference.

If there are no further data words in the record read from mass storage, the program goes to statement 43. Otherwise, the DO 40 loop is executed to reformat the first 12 damage data words and their references for printing. Each of these words is checked for the presence of data. The program goes to statement 40, the end of the loop, if no data exist for the current word. Otherwise, the data are encoded into the output record, and FINDREF is called to obtain the encoded octal reference. OCTINTG is then called to convert this reference into a hollerith format, which is placed in the output record. At this step, the DO 42 loop is initiated to process the last four damage data words. These are processed in a separate loop solely to conserve space in the output listing. A variable format is used for these four words, and they are allocated less space on the listing.

TRNUPDT begins each iteration of this loop by checking if the data already processed are the entire record; if so, the program goes to statement 43. Otherwise, it tests that data exist for the current word. If they do, an integer constant is selected from the NCHAR array and encoded in the variable FORMAT 1. FORMAT 1 then encodes the current data word into the output record. Next, subroutine FINDREF is called to obtain the reference for this data word. OCTINTG converts this reference into a hollerith format, which is then placed in the output record. Its placement ends this iteration of the DO 42 loop.

At the completion of this loop, the line print count is incremented. If this count exceeds the preset page limit, subroutine PRNTHD is called again to print a label at the top of the next page. The line count is then reinitialized. The statements beginning at 45 then print the device identifier, the TRAC reference, The statements beginning at 45 then print the device identifier, the TRAC reference, the 20 standard TRAC parameters, and two of the damage parameters and their references. The print statement outputs the remainder of the damage parameters, and the last print statement outputs the individual references for these parameters to complete this iteration of the DO 50 loop.

After all iterations of this loop have been executed, the program prints a message that the job is completed and then halts.

5.4.4 Output Data Format

At the top of each page is a label. The first two lines contain the descriptors for the device identifier, the 20 standard TRAC parameters, the TRAC reference, and descriptors for the C-B and E-B bulk resistances. Next is a blank line for spacing and clarity. The remaining two lines of the label contain the descriptors for the various damage parameters. These parameters are defined in section 4.2, and the references are listed in appendix A.

The remainder of the page contains a listing of the contents of the data records, with each record occupying four lines. The first two lines of each set contain the actual data for the descriptors in the first two lines of the label. The third line of each data set contains the damage parameters, as described in the last two lines of the label, with the corresponding references just below in the fourth line.

If the end of the named index is not found within its maximum specified length, the following message is printed: "END OF INDEX NOT FOUND, JOB TERMINATED." At the end of the job, the message "TEST OF FILE COMPLETED" is written on the system output.

5.4.5 Sample Run

The deck setup uses these control cards:

```
EMCPR,CM64000,T35.
TASK,TN*******,PW*****,TRTS. RUZIC
ATTACH,TAPE11,TRANS,ID=*******.
FTN(R=2,L)
MAP(PART)
LGO.
7/8/9
```

Card 7/8/9 is the end-of-record punch.

The deck setup uses these programs:

```
Program TRNUPDT
Subroutine PRNTHD
Subroutine OCTINTG
Subroutine DECODLN
Subroutine BLNKFIL
Subroutine FINDREF
0/6/7/8/9
```

Card 0/6/7/8/9 is the end-of-file punch. The listings of these programs are in appendix B, and the subroutines are described in appendix C. The reader can ask the author for the program output.

6. CONCLUSIONS AND RECOMMENDATIONS

Several organizations and individuals have, with varied motivations, undertaken to measure and calculate parameters, to ascertain the level at which semiconductors will be damaged from electrical transients induced by an EMP. Many results of very early measurements were later found to be unusable, primarily because many of the transistors were measured C-E, whereas much of the theoretical damage modeling concerns the separate semiconductor junctions. Another problem was that no statistical analysis was presented with the early data to enable a user to know the precision and accuracy of the results published. Unfortunately, what has been called "engineering judgment" was the criterion used to select a resultant single data value from a series of measurements on a given device. (The published data value is not directly measurable.) As an even larger problem, other organizations have published some of these same data as their own. Thus, to create a credible data base, extreme care must be taken to weed out data whose reliability is unknown.

In this data base, we have eliminated all data that are not related to specific semiconductor junctions. We found all available information on the original sources of the measured data and ascertained whether the pulsing was performed on an individual junction. The actual data points were reviewed to see if they adequately substantiated the published damage constant that had been derived from the data. (The damage constant is derived from a line that is supposed to be a discriminant function of the most probable failure or nonfailure of the device junction due to a transient EMP.) Often the damage constant published was not clearly justified by the data points. Some data sets had too few points. Some had the points too clustered to correctly define a damage line over several orders of magnitude in time. Some sets had no failure points, so the damage discriminant line was a mere guess. Other sets had almost all failure points, so the stated discriminant was not confirmed.

Therefore, eliminated from the data base were many damage constants that were not well substantiated.

This data base has been instituted to comprise all useful semiconductor data that are available in a format that can be used by circuit analysis codes. Some circuit analysis codes[2] have been modified to incorporate EMP damage assessment and are being used at HDL for this purpose. A data base containing the necessary information and

---

[2]*G. Baker, A. McNutt, B. Shea, and D. Rubenstein, Damage Analysis Modified TRAC Computer Program, Harry Diamond Laboratories TM-75-6 (May 1975).*

conforming to the format requirements of these codes is then a basic requirement that is intended to be solved by the work documented here. Further work is required. Further information that would greatly enhance the usefulness of this data base is statistical parameters on the included data. Some suggested parameters are sample size, standard estimate of error, tolerance, standard deviation, and reliability of each of the data items. A careful assessment must be made to determine the minimum number of parameters that provide the maximum amount of useful information that could be used in analyzing a circuit. Also, since a set of statistics on each damage parameter could vastly enlarge the disk storage requirement, it is highly desirable to design a format by which these sets of data could be encoded into as few words as possible.

These possibilities are being explored for implementation in these data bases.

APPENDIX A.--EXPERIMENTAL SOURCES

The data contained in this data base were derived from a number of sources. The source and methodology used in obtaining the data could be crucial in ascertaining the degree of reliance to be placed on results derived from these data. The so-called damage constants are not physical constants, but rather a type of derived statistical average. Thus, the source and methodology are often important considerations in assessing the reliability and precision of the data.

Code	Report
001	D. Wunsch, Preliminary Report--Semiconductor Damage Study, Braddock, Dunn & McDonald (December 1968).
002	D. Wunsch and L. Marzetelli, BDM Final Report, Vol. 1, Semiconductor and Nonsemiconductor Damage Study, Braddock, Dunn & McDonald Report BDM-375-69-F-0168, Prepared for U.S. Army Mobility Equipment Research and Development Center, Fort Belvoir, VA (April 1969).
003	J. B. Singletary and D. Wunsch, Final Report on Semiconductor Damage Study, Phase II, Braddock, Dunn & McDonald Report BDM/A-66-70-TR (June 1970).
004	J. B. Singletary and D. Wunsch, Final Summary Report on Semiconductor Damage Study, Phase II, Braddock, Dunn & McDonald Report BDM/A-84-70-TR (February 1971).
005	D. Wunsch, R. Cline, and G. Case, Semiconductor Vulnerability, Phase II Report, Vol. I, Theoretical Estimates of Failure Levels of Selected Semiconductor Diodes and Transistors, Braddock, Dunn & McDonald Report AFWL-TR-73-119, Vol. 1 (July 1973).
006	D. Wunsch, R. Cline, and G. Case, Semiconductor Vulnerability, Phase II Report, Theoretical Estimates of Failure Levels of Selected Semiconductor Diodes and Transistors, Braddock, Dunn & McDonald Report BDM/A-42-69-R (December 1969).
007	J. B. Singletary, W. D. Collier, and J. A. Meyers, Semiconductor Vulnerability, Phase III Report, Vol. II, Theoretical Estimates of Failure Levels of Selected Semiconductor Diodes and Transistors, Braddock, Dunn and McDonald Report BDM/A-75-70-TR (July 1973).

APPENDIX A

Code	Report
010	D. Durgin, C. Jenkins, and G. Rimbert, Methods, Devices, and Circuits for the EMP Hardening of Army Electronics, Braddock, Dunn and McDonald Report BDM/A-119-72-TR, ECOM-0275-F (July 1972).
011	D. Alexander, J. Almassy, G. Brown, D. Durgin, C. Jenkins, R. Randal, A. Unwin, and J. Schwartz, EMP Susceptibility of Semiconductor Components, Boeing Aerospace Co. and Braddock, Dunn & McDonald, Boeing Report D224-13042-1 (September 1974).
012	D. Alexander, J. Almassy, G. Brown, D. Durgin, C. Jenkins, R. Randal, A. Unwin, and J. Schwartz, Addendum to EMP Susceptibility of Semiconductor Components, Boeing Aerospace Co. and Braddock, Dunn & McDonald, Boeing Report D224-13042-2 (July 1975).
013	D. Alexander, J. Almassy, G. Brown, D. Durgin, C. Jenkins, R. Randal, A. Unwin, and J. Schwartz, Electromagnetic Susceptibility of Semiconductor Components, Final Report, Boeing Aerospace Co. and Braddock, Dunn & McDonald, Boeing Report D224-13042-1, BDM Report A-110-74-TR (September 1975).
014	G. Brown, D. Duncan, J. Cooke, D. Joppa, Experimental Damage Constant Summary, Braddock, Dunn & McDonald Report BDM/A-99-74-TR-Rl (September 1974).
015	Diode and SCR D.A.T.A. Book, Derivation and Tabulation Associates, Inc. (1970).
016	Transistor D.A.T.A. Book, Derivation and Tabulation Associates, Inc. (1969).
017	DNA EMP Handbook (U), Vol. 2, Analysis and Testing, Chapter 13, data derived from SAP-1 computer listing, DASIAC, General Electric Co.--TEMPO, DNA Report 2114H-2 (November 1971). (CONFIDENTIAL)
020	DNA EMP Handbook (U), Vol. 2, Analysis and Testing, Chapter 13, data derived from experimental data in DASA EMP Handbook, DASIAC, General Electric Co.--TEMPO, DNA Report 2114H-2 (November 1971). (CONFIDENTIAL)
021	DNA EMP Handbook (U), Vol. 2, Analysis and Testing, Chapter 13, data derived from estimated data in DASA EMP Handbook, DASIAC, General Electric Co.--TEMPO, DNA Report 2114H-2 (November 1971). (CONFIDENTIAL)

Code	Report
022	DNA EMP Handbook (U), Vol. 2, Analysis and Testing, Chapter 13, data derived from calculations of Section III of same document, DASIAC, General Electric Co.--TEMPO, DNA Report 2114H-2 (November 1971). (CONFIDENTIAL).
023	DNA EMP Handbook (U), Vol. 2, Analysis and Testing, Chapter 13, tables, DASIAC, General Electric Co.--TEMPO, DNA Report 2114H-2 (November 1971). (CONFIDENTIAL)
024	J. Miletta, EMP Effects on Components--Preprint, Harry Diamond Laboratories, unpublished.
025	J. Miletta, Lance System Component Damage Characterizations, Vol. 1, Harry Diamond Laboratories, unpublished.
026	D. Tasca, Submicrosecond Pulse Power Failure Modes in Semiconductor Devices, General Electric Co. Re-Entry and Environmental Systems Division Report 70SD401 (January 1970).
027	D. Tasca, Energy-Time Dependence of Second Breakdown in Semiconductors for Submicrosecond Electrical Pulses, General Electric Co. Missile and Space Division Report 67SD7253 (October 1967).
030	D. Tasca, J. Peden, and J. Andrews, Theoretical and Experimental Studies of Semiconductor Device Degradation Due to High Power Electrical Transients, General Electric Co. Report 73SD4289 (December 1973).
031	B. Kalab, Analysis of Failure of Electronic Circuits from EMP-Induced Signals--Review and Contribution, Harry Diamond Laboratories Report HDL-TR-1615 (August 1973).
032	G. Baker, EMP Vulnerability Analysis of M109, M110 Self-Propelled Howitzers, Harry Diamond Laboratories, unpublished.
033	G. Baker, EMP Vulnerability Analysis of Radio Sets AN/PRC-77, AN/VRC-64, and AN/GRC-160 (U), Harry Diamond Laboratories Report HDL-TR-1747 (February 1976). (SECRET RESTRICTED DATA)
034	G. Gornak et al, EMP Assessment for Army Tactical Communications Systems: Transmission Systems, Series No. 1, Radio Terminal Set AN/TRC-145 (U), Harry Diamond Laboratories Report HDL-TR-1746 (February 1976). (SECRET RESTRICTED DATA)

APPENDIX A

Code	Report
035	B. Kalab, Harry Diamond Laboratories, unpublished, measured data on AN/TRC-145 semiconductor components; damage constants obtained by analyzing measured data via program SEMCOM.
050	J. D. Holder and V. Ruwe, Statistical Component Damage Study, U.S. Army Missile Command Report RG-TR-71-1 (January 1971).
051	P. H. Stadler, Failure Threshold and Resistance of the Protected and Unprotected 2N2222 Transistor in the Short Pulse Width Regime, Philco-Ford Corp. Report U-4976 (May 1972).
052	EMP Electronic Design Handbook, Boeing Aerospace Co. Report D224-10019-1 (April 1973).
053	C. Jenkins and J. Meyers, Integrated Circuits Test Program, Final Report, Braddock, Dunn & McDonald Report BDM/A-98-73-TR (July 1973).
054	D. Alexander, T. Zwolinski, and C. Jenkins, Integrated Circuits and Discrete Semiconductor Components Test Program, Braddock, Dunn & McDonald Technical Directive 4-6, Monthly Progress Reports (January, February, March 1974).
055	J. Smith, Pulse Power Testing of Microcircuits, Rome Air Development Command Report RADC-TR-71-59 (October 1971).
056	Pulse Damage Data from Integrated Circuits and Electronic Parts, Boeing Aerospace Co. Memorandum 2-6731-0000-C/S-102 (September 1973).
057	G. Rimbert et al, Resistor Modeling Program, Final Report, Braddock, Dunn & McDonald ASV Work Order 2-14 [n.d.].
060	A. Brandstein, Harry Diamond Laboratories, calculations performed on experimental data measured by B. Kalab for threshold of power to damage for pulses of 10-μs width.
061	Data values contained in previous computer data base, maintained by Harry Diamond Laboratories at U.S. Army Equipment Research and Development Center, Fort Belvoir, VA.
062	B. Kalab, Harry Diamond Laboratories, experimentally measured data affected by pulser limitations.
063	DNA Handbook (U), forward biased damage constants depicted graphically, DNA Report 2114H-2 (November 1971). (CONFIDENTIAL)

Code	Report
064	C. Ruzic, Harry Diamond Laboratories, weighted average of data published in reference for code 013.
065	C. Ruzic, Harry Diamond Laboratories, average of data published in reference for code 013.
066	G. Gornak et al, Harry Diamond Laboratories, data obtained from reference for code 034, but for which conflicting data exist in reference for code 025.
067	D. Tasca, J. Peden, and D. Nepreux, Pulsed Power Failure Modes, Conference Proceedings, Component Degradation from Transient Inputs, U.S. Army Mobility Equipment Research and Development Center, Fort Belvoir, VA (April 1970).
070	C. Ruzic, Harry Diamond Laboratories, data derived from averaging means of data presented in reference for code 067.
100	G. Brown, S. Jones, R. Randall, J. Schwartz, Discrete Semiconductor EMP Data Summary, Boeing Aerospace Co. and Braddock, Dunn & McDonald, Boeing Report D224-13043-1, BDM Report A-111-74-TR (September 1974).
777	Unknown.

APPENDIX B.--PROGRAM LISTINGS

This appendix lists the programs for DTABSE, TRNSBSE, TSTUPDT, and TRNUPDT.

APPENDIX B

```
EMCPR,CM64000,T100.
TASK,TNEM71603,PWPRMPT,TRTS. RUZIC
ATTACH,TAPE11,DIODE,ID=EM71603.
REQUEST,TAPE12,*PF.
EXTEND,TAPE11.
FTN(R=2,L)
MAP(PART)
LGO.
CATALOG,TAPE12,DIODE,ID=EM71603.
W
      PROGRAM DTABSE( INPUT,OUTPUT, TAPE5=INPUT,TAPE6=OUTPUT,TAPE11,
     1  TAPE12 )
      DIMENSION BOTHPRM(50), TRACPRM(50), DAMGPRM(50), RECORD(50),
     1 KEY1(501)
      COMMON /A/ID, LNTHDIO, KOPT, IOPTS(2), IREFS(15), IDXTRA, IBLANK,
     1 IZERO,IBOTH, TRAC, DAMG, XNEGIND, KEY(501), LNTHREC, CHNGE
      EQUIVALENCE( BOTHPRM(1), TRACPRM(1), DAMGPRM(1) ),(IBLANK,BLANK)
      EQUIVALENCE ( TRACREF, IREFS(1) ), ( XNEGIND, NEGINDF )
      INTEGER TRAC, DAMG, CHNGE, DELETE, REFS,SORT
      DATA LLAST,IBLANK, IZERO,IBOTH,TRAC,DAMG/10HLAST       ,10H
     1    , 0, 10HBOTH       , 10HTRAC      , 10HDAMG        /
      DATA XNEGIND,SORT / 40007777777777777000B, 10HSORT        /
      DATA DELETE,CHNGE,REFS /10HDELETE     ,10HCHANGE     ,10HREFS       /
C
C        LNTHDIO         IS THE NO. OF RECORDS IN THE FILE
C        LNTHREC         IS THE NO. OF WORDS IN THE RECORD CURRENTLY UNDER CONSID
C        IOPTS(1)        CAN BE
C                        DELETE          TO DELETE AN EXISTING RECORD
C                        CHANGE          TO CHANGE AN EXISTING RECORD
C                        NEW             TO CREATE A NEW RECORD FROM DATA CARDS
C                        SORT            TO RESORT + REWRITE THE EXISTING FILE
C        IOPTS(2)        ARE THE POSSIBLE OPTIONS WHEN  IOPTS(1) = "CHANGE"
C                        THESE OPTIONS ARE
C                        TRAC            TO CHANGE OR DELETE STD. TRAC PARAMETERS
C                        DAMG            TO CHANGE OR DELETE NON-STD TRAC OR DAMAG
C                        BOTH            CHANGE OR DELETE STD + NON-STD TRAC +DAMG
C                        REFS            TO CHANGE OR DELETE REFERENCES ONLY
C
C
C     ************************************************************************
C
C     IF A CATALOGUE IS DONE BY MISTAKE WITHOUT THE FILE BEING SORTED AND
C     REWRITTEN, A BLANK FILE, CONTAINING ONLY THE INDEX KEY, WILL RESULT
C
C     ************************************************************************
C
      IFILE = 11
      JFILE = 12
C  LNTHDIO INITIALLY SET TO CORRESPOND TO MAX. NO. OF RECS IN FILE AS
C  DETERMINED BY DIMENSIONED SIZE OF "KEY" ARRAY
      LNTHDIO = 249
      CALL OPENMS( IFILE, KEY, 499, 1 )
      CALL OPENMS ( JFILE, KEY1, 499, 1 )
      INEWREC = 0
      IDELETE = 0
C  SEARCH KEY ARRAY TO DETERMINE NUMBER OF RECORDS IN DATA FILE
      DO 3 I = 1, LNTHDIO
```

```
      IF( KEY(2*I) .NE. 0 ) GO TO 3
      L = I - 1
      PRINT 905, L
      GO TO 5
    3 CONTINUE
      PRINT 915, LNTHDIO
      STOP
    5 LNTHDIO = L
C   READ INPUT DATA OPTIONS + REFS FOR CHANGES TO DATA BASE FOR THIS DEVICE
   10 READ (5,800)KOPT,(IOPTS(J),J=1,2),( IREFS(I),I =1,12), LAST,IDXTRA
      IF( EOF(5) ) 84, 15
   15 IF ( IOPTS(1) .EQ. SORT ) GO TO 95
      IF( IOPTS(1) .EQ. DELETE )  GO TO 80
      IF( IOPTS(1) .EQ. CHNGE ) GO TO 60
C   IOPTS(1) ASSUMED TO BE "NEW RECORD"
      IF( IOPTS(2) .EQ.IBOTH ) GO TO 40
      IF( IOPTS(2) .NE. TRAC )  GO TO 30
C   NEW TRAC PARAMETER DATA ONLY
      NPARMS = 9
      PRINT 1
    1 FORMAT( 1H1, * STANDARD TRAC DATA ONLY * )
      CALL DTAREAD ( TRACPRM, NPARMS, IDFLG )
      IF( IDFLG .NE. 0 ) GO TO 55
C   STORE NEW TRAC DATA IN OUTPUT REC + FILL BAL. OF REC WITH NEG. INDF.
      CALL TRACDTA ( TRACPRM, RECORD, IFILE )
      GO TO 50
C   NEW DAMAGE DATA + NON-STD. TRAC PARAMETERS ONLY
   30 NPARMS = 6 + KOPT
      CALL DTAREAD( DAMGPRM, NPARMS, IDFLG )
      IF ( IDFLG .NE. 0 ) GO TO 55
      CALL DAMGDTA( BOTHPRM, RECORD, IFILE )
      GO TO 50
C   NEW RECORD WITH BOTH TRAC + ( DAMAGE + NON-STD. TRAC PARAMETERS)
   40 NPARMS = 15 + KOPT
      CALL DTAREAD( BOTHPRM, NPARMS, IDFLG )
      IF( IDFLG .NE. 0 ) GO TO 55
      CALL BOTHDTA(  BOTHPRM, RECORD, IFILE )
C   WRITE NEW OUTPUT RECORD AT END OF PREVIOUS DATA + INCREMENT NEW RECORD COUNT
   50 LNTHREC = 16 + KOPT
      CALL WRITMS( IFILE, RECORD, LNTHREC, ID, 0 )
      INEWREC = INEWREC + 1
      LNTHDIO = LNTHDIO + 1
      GO TO 90
C   PRINT ERROR MESSAGE THAT A RECORD ALREADY EXISTS WITH SUPPOSEDLY NEW ID
   55 PRINT 925, ID, IOPTS(1), IOPTS(2)
      GO TO 90
C   PRINT ERROR MESSAGE THAT NO RECORD ALREADY EXISTS WITH SUPPOSEDLY OLD ID
   58 PRINT 935, ID, IOPTS(1), IOPTS(2)
      GO TO 90
C
C   TREAT CASES OF CHANGES TO EXISTING RECORDS
   60 IF( IOPTS(2) .EQ. REFS ) GO TO 70
      NPARMS = 15 + KOPT
      IF( IOPTS(2) .EQ. TRAC ) NPARMS = 9
      IF( IOPTS(2) .EQ. DAMG ) NPARMS = 6 + KOPT
      CALL DTAREAD( CHNGPRM, NPARMS, IDFLG )
C   DTAREAD WILL CALL DECODLN WHICH WILL STORE THE REC LENGTH IN LNTHREC
```

```
      IF( IDFLG .EQ. 0 ) GO TO 58
      IF( IOPTS(2) .EQ. TRAC ) CALL REVTRAC(CHNGPRM,RECORD,IFILE)
      IF( IOPTS(2) .EQ. DAMG ) CALL REVDAMG(CHNGPRM,RECORD,IFILE)
      IF( IOPTS(2) .EQ.IBOTH ) CALL REVBOTH(CHNGPRM,RECORD,IFILE)
C  REWRITE EXISTING RECORD WITH NEW DATA
C  DETERMINE RECORD LENGTH FOR REVISED RECORD
      ICNT = NPARMS + 1
      IF( IOPTS(2) .EQ. TRAC ) GO TO 65
      IF( IOPTS(2) .EQ. IBOTH ) GO TO 63
      ICNT = NPARMS + 10
   63 LNTHREC = MAXO( LNTHREC, ICNT )
   65 CALL WRITMS( IFILE, RECORD, LNTHREC, ID,-1 )
      GO TO 90
C
C     ******************************************************************************
C  TREAT CHANGES TO REFERENCES ONLY
C     ******************************************************************************
C
   70 DO 72 I = 1, LNTHDIO
      IF( KEY(2*I) .EQ. IDXTRA ) GO TO 74
   72 CONTINUE
C  NO MATCH FOUND IN NAMED INDEX FOR SPECIFIED "ID"
      PRINT 910, IDXTRA
      GO TO 90
C  IMPLEMENT CHANGES TO REFERENCES
   74 CALL DECODLN( KEY(2*I+1), LNTHREC )
      CALL READMS( IFILE, RECORD, LNTHREC, IDXTRA )
C  MUST LINK INDICIES TO DATA WORD + NOT REF BECAUSE SINGLE REF CAN BE USED
C  FOR MORE THAN ONE DATA WORD
      IF( IREFS(1) .EQ. -0 ) RECORD(10) = BLANK
      IF( IREFS(1) .EQ. 0 ) GO TO 741
      RECORD(10) = TRACREF
  741 IR = 3
      K9 = 16 + KOPT
      DO 78 I = 11, K9
      IF( I .GT. 12 ) IR = 2
C  CHECK FOR NO CHANGE TO THIS REFERENCE
      IF( IREFS(I-12+IR) .EQ. 0 )  GO TO 78
C  CHANGE OR DELETE OLD REF + STORE THIS CHANGE IN OUTPUT RECORD
      IF( IREFS(I-12 + IR ) .NE. -0 ) GO TO 75
C  DELETE OLD REFERENCE
      II = IZERO
      CALL REFENCD( RECORD(I), II ), RETURNS (77)
      GO TO 78
C  CHANGE OLD REFERENCE
   75 II = IREFS(I-12 + IR )
      CALL REFENCD( RECORD(I), II ), RETURNS(77)
      GO TO 78
   77 PRINT 900, I, IDXTRA, RECORD(I), II
   78 CONTINUE
C  REWRITE EXISTING RECORD WITH REFERENCES CHANGED
      CALL WRITMS( IFILE, RECORD, LNTHREC, IDXTRA, 1 )
      GO TO 90
C  TREAT CASE OF ENTIRE RECORD TO BE DELETED
C  DELETE ID FROM NAMED INDEX ARRAY + INCREMENT COUNTER
   80 LN = LNTHDIO - 1
      DO 83 I = 1, LNTHDIO
```

```
      IF( KEY(2*I) .NE. IDXTRA ) GO TO 83
      DO 81 J = I, LN
      KEY(2*J) = KEY(2*J+2)
   81 KEY(2*J+1) = KEY(2*J+3 )
      GO TO 85
   83 CONTINUE
C  IF IT FALLS THRU END OF LOOP, NO MATCH FOUND FOR "ID"
      PRINT 910, IDXTRA
      GO TO 90
   84 PRINT 810
      GO TO 92
C  INCREMENT COUNTER OF RECORDS DELETED
   85 IDELETE = IDELETE + 1
      LNTHDIO = LNTHDIO - 1
C  CHECK FOR LAST DATA SET
   90 IF( LAST .NE. LLAST ) GO TO 10
C  IF THERE HAVE NOT BEEN ANY RECORDS ADDED OR DELETED, CALL EXIT
   92 IF( INEWREC .EQ. 0 .A. IDELETE .EQ. 0 ) CALL EXIT
C  SORT NAMED INDEX ARRAY
C  WRITE A NEW CYCLE ON THE FILE WITH ADDITIONS AND DELETIONS AS WELL AS CHANGES
   95 CALL SORTKEY( KEY,  LNTHDIO )
C  TRANSFER INDEX DATA TO OUTPUT FILE INDEX
      DO 98 I = 1,LNTHDIO
      KEY1(2*I) = KEY(2*I)
   98 CONTINUE
C   PRINT NO. OF DEVICES, CONTENTS OF INDEX ARRAY AND INDIVIDUAL REC LENGTHS
      PRINT 940, LNTHDIO
      DO 100 I = 1, LNTHDIO
      CALL DECODLN( KEY(2*I+1), LNTHREC )
      CALL READMS(IFILE, RECORD, LNTHREC, KEY(2*I) )
      CALL WRITMS( JFILE, RECORD, LNTHREC, KEY1(2*I), 0 )
      CALL DECODLN( KEY1(2*I+1), LN )
      PRINT 950, KEY1(2*I), LNTHREC, LN, ( RECORD(IJ),IJ= 1,LNTHREC )
  100 CONTINUE
      CALL CLOSMS(IFILE)
      CALL CLOSMS(JFILE)
      STOP
  800 FORMAT(I1, A6, A4, A10, 11(O3,1X), A4, 1X, A10 )
  810 FORMAT( 1H1, *END OF FILE ENCOUNTERED WHILE ATTEMPTING TO READ NEX
     1T DATA SET* / 1X, *JOB ENDED BASED ON DATA ALREADY READ * )
  900 FORMAT( 1H0,// * REFERENCE OUT OF RANGE FOR DATA ITEM NO. * I5,
     1 * FOR DEVICE * , A10, * VALUE IS * E12.3, * REF IS *, O3 / * OLD
     2REFERENCE NOT CHANGED * )
  905 FORMAT( 1H0, /// * NUMBER OF RECORDS IN THE FILE IS *, I5 )
  910 FORMAT( 1H0 // * NO MATCH FOUND IN NAMED INDEX ARRAY FOR DEVICE TY
     1PE WHIC WAS TO ONLY HAVE REFERENCES CHANGED, ID WAS *, A10, /
     2 * THESE CHANGES WERE THEREFORE IGNORED + PROCESSING CONTINUES * )
  915 FORMAT( 1H0,///*EOF NOT REACHED AFTER *, I5, * DEVICE TYPES*)
  925 FORMAT(1H0///* ATTEMPTED TO CREATE A NEW DATA REC WITH THE SAME ID
     1 AS AN OLD REC, THIS NEW DATA SET WAS IGNORED + PROCESSING CONTINU
     2ES* / * ID WAS *, A10, * IOPTS(1) WAS *, A6, * IOPTS(2) WAS *, A4)
  935 FORMAT(1H0///* ATTEMPTED TO CHANGE AN EXISTING RECORD IN THE DATA
     1FILE, AND THE SPECIFIED "ID" WAS NOT FOUND IN THE NAMED INDEX ARRA
     2Y *,/ *THIS DATA SET WAS IGNORED + PROCESSING CONTINUES */ * ID WA
     3S *, A10, * IOPTS(1) WAS *, A6, * IOPTS(2) WAS *, A4 //1H  )
  940 FORMAT(1H0, *THERE ARE NOW *, I4, * RECORDS ON THE MASS STORAGE FI
     1LE*, / 1X, *DEVICE*, 11X, *REC LENGTH* // 1H    )
```

```
  950 FORMAT( 1X, A10, 2X, I4, 2X, I4,5(2X,O20)/( 25X,O20,2X,O20,2X,O20,
     1 2X, O20, 2X, O20 ) )
      END
      SUBROUTINE REVTRAC( A, RECORD, IFILE )
C  THIS SUBROUTINE WILL CHECK FOR ADDITIONS OR DELETIONS TO THE EXISTING RECORD
C
C  A MINUS ZERO  (-0)  IN THE  A  ARRAY OR THE  IREFS  ARRAY WILL INDICATE THAT
C  THE CORRESPONDING DATA VALUE IS TO BE DELETED FROM THE EXISTING RECORD.
C  A  NON ZERO  (  NON BLANK  FOR IREFS(1) ) VALUE WILL INDICATE THAT ANEW VALUE
C  IS TO BE STORED IN THE DATA RECORD
C
C  THIS SUBROUTINE ASSUMES THAT "ID" HAS ALREADY BEEN CHECKED AGAINST THE
C  CONTENTS OF THE KEY NAMED INDEX ARRAY , AND A MATCH WAS FOUND
      COMMON /A/ID, LNTHDID, KOPT, IOPTS(2), IREFS(15), IDXTRA, IBLANK,
     1 IZERO,IBOTH, TRAC, DAMG, XNEGIND, KEY(501), LNTHREC, CHNGE
      DIMENSION RECORD(50), A(1)
      EQUIVALENCE ( TRACREF, IREFS(1) )
    1 CALL READMS( IFILE, RECORD(1), LNTHREC, ID )
      DO 10 I = 1,9
      IF( A(I) .EQ. 0. ) GO TO 10
      IF( A(I) .EQ. -0. ) GO TO 5
C  NEW VALUE TO BE PLACED IN THE RECORD
      RECORD(I) = A(I)
      GO TO 10
    5 RECORD(I) = XNEGIND
   10 CONTINUE
      RECORD(10) = TRACREF
   20 CONTINUE
      IF( IOPTS(2) .EQ. IBOTH ) GO TO 30
      RETURN
      ENTRY REVDAMG
      CALL READMS( IFILE, RECORD, LNTHREC, ID )
C  SET INDICIES FOR DO LOOP TO HANDLE DAMAGE + NON-STD. TRAC PARAMETERS
      IT1 = 1
      IT2 = 6
   30 IR = 2
      J = 10
      IT2 = IT2 + KOPT
      DO 60 I = IT1, IT2
      J = J + 1
C  IS DATA TO BE DELETED...IF SO, REF IS ALSO DELETED
      IF( A(I) .NE. -0. ) GO TO 33
      RECORD(J) = XNEGIND
      GO TO 60
   33 IF( I .GT. IT1 + 1 ) IR = 1
C  IS THERE A NEW VALUE FOR CURRENT DATA ITEM...
      IF( A(I) .EQ. 0. ) GO TO 38
C  IS THERE A NEW REFERENCE FOR THIS NEW DATA VALUE
      IF( IREFS(I-IT1+IR) .GT. 0 ) GO TO 36
C  IS THE OLD REFERENCE TO BE DELETED...
      IF( IREFS(I-IT1+IR) .NE. -0 ) GO TO 34
C  TREAT CASE OF NEW DATA WITHOUT REF + OLD REFERENCE TO BE DELETED
      CALL REFENCD( A(I), IZERO ), RETURNS(40)
      RECORD(J) = A(I)
      GO TO 60
C  TREAT CASE OF NEW DATA TO BE COMBINED WITH OLD REFERENCE
   34 CALL FINDREF( RECORD(J), IRF )
```

```
      CALL REFENCD( A(I), IRF ), RETURNS(42)
      RECORD(J) = A(I)
      GO TO 60
C  TREAT CASE OF NEW DATA VALUE + NEW REFERENCE
   36 CONTINUE
C     PRINT STATEMENTS FOR CHECKOUT
C     PRINT 800, ID, I, IT1, IR,  A(I), A(I),IREFS(I-IT1+IR)
C 800 FORMAT(1X, A10, 5X, *I =*, I3, 5X, *IT1 =*, I3, 5X, *IR = *, I3,
C    1 5X, O20, 5X, E13.3, 5X, *IREFS = *, O3, 5X, *CHANGE* )
C 801 FORMAT( 1X, O20, 5X, O20 )
      CALL REFENCD( A(I), IREFS(I-IT1+IR) ), RETURNS(44)
      RECORD(J) = A(I)
C     PRINT 801,A(I), RECORD(J)
      GO TO 60
C  IS NEW REFERENCE TO BE STORED IN THE OLD DATA VALUE...
   38 IF( IREFS(I-IT1+IR) .EQ. 0 ) GO TO 60
C  IS OLD REF TO BE DELETED FROM OLD DATA VALUE...
      IF( IREFS(I-IT1+IR) .EQ.-0 )   GO TO 50
C  TREAT CASE OF OLD DATA TO BE COMBINED WITH NEW REFERENCE
      CALL REFENCD( RECORD(J), IREFS(I-IT1+IR) ), RETURNS(40)
      GO TO 60
C  DELETE OLD REFERENCE + LEAVE OLD DATA VALUE
   50 CALL REFENCD( RECORD(J), IZERO ), RETURNS(40)
      GO TO 60
C  TREAT ERROR RETURNS FROM REFENCD
   40 PRINT 900, J, ID, A(I), IZERO
      GO TO 60
   42 PRINT 900, J, ID, A(I), IRF
      GO TO 60
   44 PRINT 900, J, ID, A(I), IREFS(I-IT1+IR)
      GO TO 60
   46 PRINT 900, J, ID, RECORD(J), IREFS(I-IT1+IR)
   60 CONTINUE
      RETURN
      ENTRY REVBOTH
      IT1 = 10
      IT2 = 15
      GO TO 1
  900 FORMAT( 1H0,// * REFERENCE OUT OF RANGE FOR DATA ITEM NO. * I5,
     1 * FOR DEVICE * , A10, * VALUE IS * E12.3, * REF IS *, O3 / * OLD
     2REFERENCE NOT CHANGED * )
      END
      SUBROUTINE TRACDTA( A, RECORD, IFILE )
C  SUBROUTINE TO ADD PARAMETERS FOR NEW DEVICE TYPES TO EXISTING TRAC + DAMAGE
C  DATA BASE
C                      A  IS TRAC AND/OR DAMAGE PARAMETER ARRAY
C
C                      N  IS IS THE NUMBER OF DATA WORDS READ FROM CARDS
C                      RECORD  IS OUTPUT RECORD FOR WRITING TO MASS STORAGE
C
C                      IREFS ARE REFERENCES ON THE DATA SOURCES
C                      IFILE  IS THE LOGICAL MASS STORAGE UNIT NUMBER
C
      COMMON /A/ID, LNTHDIO, KOPT, IOPTS(2), IREFS(15), IDXTRA, IBLANK,
     1 IZERO,IBOTH, TRAC, DAMG, XNEGIND, KEY(501), LNTHREC, CHNGE
      DIMENSION RECORD(50), A(1)
      EQUIVALENCE ( TRACREF, IREFS(1) ), (IBLANK, BLANK )
```

```
      IT3 = 15
   10 DO 20 I = 1,9
      RECORD(I) = A(I)
      IF( A(I) .EQ. 0 ) RECORD(I) = XNEGIND
   20 CONTINUE
C  IF THERE WAS NO TRAC REFERENCE, A BLANK IS STORED
      RECORD(10) = TRACREF
C  IS THIS A BOTHDTA ENTRY....
      IF(  IOPTS(2) .EQ.IBOTH ) GO TO 50
C  STORE NEGATIVE INDEFINITE IN DAMAGE AND NON-STD. TRAC PARAM PORTION OF RECORD
      DO 30 I = 11, IT3
   30 RECORD(I) = XNEGIND
      RETURN
      ENTRY DAMGDTA
      DO 40 I = 1,9
C  STORE NEGATIVE INDEFINITE IN STD. TRAC PARAMETER PORTION OF THE RECORD
   40 RECORD(I) = XNEGIND
      RECORD(10) =  BLANK
C  SET INDICIES FOR DO LOOP TO HANDLE DAMAGE + NON-STD. PARAMETERS
      IT1 = 1
      IT2 = 6
   50 IR = 2
      J = 10
      IT2 = IT2 + KOPT
      DO 60 I = IT1, IT2
      J = J + 1
C  TEST FOR PRESENCE OF DATA VALUE
      IF( A(I) .EQ. 0. ) GO TO 55
      IF( I .GT. IT1 + 1 ) IR = 1
C     PRINT STATEMENTS FOR CHECKOUT
C     PRINT 800, ID, I, IT1, IR,  A(I), A(I),IREFS(I-IT1+IR)
C 800 FORMAT(1X, A10, 5X, *I =*, I3, 5X, *IT1 =*, I3, 5X, *IR = *, I3,
C    1  5X, O20, 5X, E13.3, 5X, *IREFS = *, O3, 5X, *NEW* )
C 801 FORMAT( 1X, O20, 5X, O20 )
      CALL REFENCD( A(I), IREFS(I-IT1+IR) ), RETURNS(56)
   53 RECORD(J) = A(I)
C     PRINT 801,A(I), RECORD(J)
      GO TO 60
   55 RECORD(J) = XNEGIND
      GO TO 60
C  ERROR RETURN ON REFERENCE ID OUT OF RANGE, STORE ZERO FOR REF.
   56 IZERO = 0
      PRINT 900, J, ID,A(I), IREFS(I-IT1+IR)
      CALL  REFENCD( A(I), IZERO ), RETURNS(56)
      RECORD(J) = A(I)
   60 CONTINUE
      RETURN
      ENTRY BOTHDTA
      IT1 = 10
      IT2 = 15
      GO TO 10
  900 FORMAT( 1HO,// * REFERENCE OUT OF RANGE FOR DATA ITEM NO. * I5,
     1 * FOR DEVICE * , A10, * VALUE IS * E12.3, * REF IS *, O3 / * OLD
     2REFERENCE NOT CHANGED * )
      END
      SUBROUTINE SORTKEY( KEY, LENGTH )
C  SUBROUTINE TO SORT THE NAMED KEY ARRAY
```

```
      DIMENSION KEY(1)
      L1 = LENGTH - 1
      DO 30 I = 1, L1
      K2 = KEY(2*I)
      K3 = KEY(2*I+1)
      I1 = I + 1
      DO 20 J = I1, LENGTH
      IF( KEY(2*I) .LT. KEY(2*J) ) GO TO 20
      KEY(2*I) = KEY(2*J)
      KEY(2*I+1) = KEY(2*J+1)
      KEY(2*J) = K2
      KEY(2*J+1) = K3
      K2 = KEY(2*I)
      K3 = KEY(2*I+1)
   20 CONTINUE
   30 CONTINUE
      RETURN
      END
      SUBROUTINE DTAREAD ( A, N, IDFLG )
      DIMENSION A(1)
      COMMON /A/ID, LNTHDIO, KOPT, IOPTS(2), IREFS(15), IDXTRA, IBLANK,
     1 IZERO,IBOTH, TRAC, DAMG, XNEGIND, KEY(501), LNTHREC, CHNGE
      INTEGER CHNGE
C   READ INPUT DATA CARDS
      J = 7
      READ (5,850) ID, ( A(I), I= 1,J )
    5 IF( J.GE. N ) GO TO 10
      J1 = J + 1
      J =  J + 8
      READ (5,860 ) ( A(I), I = J1,J )
      GO TO 5
   10 DO 20 I = 1,LNTHDIO
      IF( KEY(2*I) .EQ. ID )  GO TO 40
   20 CONTINUE
C   NO MATCH IN NAMED KEY INDEX WITH CURRENT "ID"
      IDFLG = 0
      RETURN
C   MATCH FOUND IN NAMED KEY INDEX WITH CURRENT "ID"
   40 IDFLG = ID
      IF( IOPTS(1) .NE. CHNGE ) GO TO 50
      CALL DECODLN( KEY(2*I+1), LNTHREC )
   50 RETURN
  850 FORMAT( A10, 7E10.1 )
  860 FORMAT( 8E10.1 )
      END
      SUBROUTINE REFENCD( DTA, IREF ), RETURNS(I)
C   SUBROUTINE TO STORE AN ENCODED REFERENCE INTO THE THREE LEAST SIGNIFICANT
C   OCTAL  CHARACTERS OF A REAL VARIABLE DATA WORD
      DATA MASK1 / 77777777777777770000B /
      IF( IREF .GT. 777B ) GO TO 20
      DTA = ( AND( DTA, MASK1 )  .OR. IREF )
      RETURN
   20 RETURN I
      END
      SUBROUTINE DECODLN( K, LNTHREC )
C   SUBROUTINE TO DECODE THE RECORD LENGTH FROM THE DATA IN NAMED INDEX ARRAY
C   THIS DATA IS APPARENTLY CONTAINED IN THE 12TH THRU 19TH OCTAL CHARACTERS OF
```

APPENDIX B

```
C   THE ODD NUMBERED WORDS OF THE NAMED KEY  INDEX, EXCLUSIVE OF THE FIRST WORD
       DATA MASK / 07777777700000000000B /
C   ZERO OUT ALL OCTAL CHARACTERS EXCEPT THE RECORD LENGTH
       KK = AND( K, MASK )
C   SHIFT RECORD LENGTH TO LOW ORDER BITS OF THE DATA WORD, IN INTEGER FORMAT
       LNTHREC = SHIFT( KK, -33 )
       RETURN
       END
       SUBROUTINE FINDREF( DTA, IREF )
C   SUBROUTINE TO DECODE A 3 OCTAL DIGIT REFERENCE STORED IN THE 3 LEAST
C   SIGNIFICANT OCTAL CHARACTERS OF A REAL VARIABLE DATA WORD
       DATA MASK2 / 00000000000000000777B /
       IREF = AND(  DTA, MASK2 )
       RETURN
       END
```

```
EMCPR,CM64000,T100.
TASK,TNEM71603,PWPRMPT,TRTS. RUZIC
ATTACH,TAPE11,TRANS,ID=EM71603.
REQUEST,TAPE12,*PF.
EXTEND,TAPE11.
FTN(R=2,L)
MAP(PART)
LGO.
CATALOG,TAPE12,TRANS,ID=EM71603.
W
      PROGRAM TRNSBSE( INPUT,OUTPUT, TAPE5=INPUT,TAPE6=OUTPUT,TAPE11,
     1  TAPE12 )
      DIMENSION BOTHPRM(50), TRACPRM(50), DAMGPRM(50), RECORD(50),
     1 KEY1(501)
      COMMON /A/ID, LNTHFIL, KOPT, IOPTS(2), IREFS(15), IDXTRA, IBLANK,
     1 IZERO,IBOTH, TRAC, DAMG, XNEGIND, KEY(501), LNTHREC, CHNGE
      EQUIVALENCE( BOTHPRM(1), TRACPRM(1), DAMGPRM(1) ), ( BLANK,IBLANK)
      EQUIVALENCE ( TRACREF, IREFS(1) ), ( XNEGIND, NEGINDF )
      INTEGER TRAC, DAMG, CHNGE, DELETE, REFS,SORT
      DATA LLAST,IBLANK, IZERO,IBOTH,TRAC,DAMG/10HLAST      ,10H
     1    , 0, 10HBOTH      , 10HTRAC      , 10HDAMG      /
      DATA XNEGIND,SORT / 40007777777777777000B, 10HSORT      /
      DATA DELETE,CHNGE,REFS /10HDELETE    ,10HCHANGE    ,10HREFS      /
C
C        LNTHFIL        IS THE NO. OF RECORDS IN THE FILE
C        LNTHREC        IS THE NO. OF WORDS IN THE RECORD CURRENTLY UNDER CONSID
C        IOPTS(1)       CAN BE
C                       DELETE         TO DELETE AN EXISTING RECORD
C                       CHANGE         TO CHANGE AN EXISTING RECORD
C                       NEW            TO CREATE A NEW RECORD FROM DATA CARDS
C                       SORT           TO RESORT + REWRITE THE EXISTING FILE
C        IOPTS(2)       ARE THE POSSIBLE OPTIONS WHEN  IOPTS(1) = "CHANGE"
C                       THESE OPTIONS ARE
C                       TRAC           TO CHANGE OR DELETE STD. TRAC PARAMETERS
C                       DAMG           TO CHANGE OR DELETE NON-STD TRAC OR DAMAG
C                       BOTH           CHANGE OR DELETE STD + NON-STD TRAC +DAMG
C                       REFS           TO CHANGE OR DELETE REFERENCES ONLY
C
C
C     ********************************************************************
C
C     IF A CATALOGUE IS DONE BY MISTAKE WITHOUT THE FILE BEING SORTED AND
C     REWRITTEN, A BLANK FILE, CONTAINING ONLY THE INDEX KEY, WILL RESULT
C
C     ********************************************************************
C
      IFILE = 11
      JFILE = 12
C   LNTHFIL INITIALLY SET TO CORRESPOND TO MAX. NO. OF RECS IN FILE AS
C   DETERMINED BY DIMENSIONED SIZE OF "KEY" ARRAY
      LNTHFIL = 249
      CALL OPENMS( IFILE, KEY, 500, 1 )
      CALL OPENMS ( JFILE, KEY1, 499, 1 )
      INEWREC = 0
      IDELETE = 0
C   SEARCH KEY ARRAY TO DETERMINE NUMBER OF RECORDS IN DATA FILE
      DO 3 I = 1, LNTHFIL
```

```
      IF( KEY(2*I) .NE. 0 ) GO TO 3
      L = I - 1
      PRINT 905, L
      GO TO 5
    3 CONTINUE
      PRINT 915, LNTHFIL
      STOP
    5 LNTHFIL = L
C   READ INPUT DATA OPTIONS + REFS FOR CHANGES TO DATA BASE FOR THIS DEVICE
   10 READ (5,800)KOPT,(IOPTS(J),J=1,2),( IREFS(I),I =1,14), LAST
      IF( EOF(5) ) 84, 15
   15 IF ( IOPTS(1) .EQ. SORT ) GO TO 95
      IF( IOPTS(1) .EQ. DELETE )  GO TO 80
      IF( IOPTS(1) .EQ. CHNGE ) GO TO 60
C   IOPTS(1) ASSUMED TO BE "NEW RECORD"
      IF( IOPTS(2) .EQ.IBOTH ) GO TO 40
      IF( IOPTS(2) .NE. TRAC )  GO TO 30
C   NEW TRAC PARAMETER DATA ONLY
      NPARMS = 20
      PRINT 1
    1 FORMAT( 1H1, * TRAC PARAMETERS ONLY FOR THIS DEVICE * )
      CALL DTAREAD ( TRACPRM, NPARMS, IDFLG )
      IF( IDFLG .NE. 0 ) GO TO 55
C   STORE NEW TRAC DATA IN OUTPUT REC + FILL BAL. OF REC WITH NEG. INDF.
      CALL TRACTRN ( TRACPRM, RECORD, IFILE )
      GO TO 50
C   NEW DAMAGE DATA + NON-STD. TRAC PARAMETERS ONLY
   30 NPARMS = 12+ KOPT
      CALL DTAREAD( DAMGPRM, NPARMS, IDFLG )
      IF ( IDFLG .NE. 0 ) GO TO 55
      CALL DAMGTRN( BOTHPRM, RECORD, IFILE )
      GO TO 50
C   NEW RECORD WITH BOTH TRAC + ( DAMAGE + NON-STD. TRAC PARAMETERS)
   40 NPARMS = 32 + KOPT
      CALL DTAREAD( BOTHPRM, NPARMS, IDFLG )
      IF( IDFLG .NE. 0 ) GO TO 55
      CALL BOTHTRN(  BOTHPRM, RECORD, IFILE )
C   WRITE NEW OUTPUT RECORD AT END OF PREVIOUS DATA + INCREMENT NEW RECORD COUNT
   50 LNTHREC = 33 + KOPT
     .CALL WRITMS( IFILE, RECORD, LNTHREC, ID, 0 )
      INEWREC = INEWREC + 1
      LNTHFIL = LNTHFIL + 1
      GO TO 90
C   PRINT ERROR MESSAGE THAT A RECORD ALREADY EXISTS WITH SUPPOSEDLY NEW ID
   55 PRINT 925, ID, IOPTS(1), IOPTS(2)
      GO TO 90
C   PRINT ERROR MESSAGE THAT NO RECORD ALREADY EXISTS WITH SUPPOSEDLY OLD ID
   58 PRINT 935, ID, IOPTS(1), IOPTS(2)
      GO TO 90
C
C   TREAT CASES OF CHANGES TO EXISTING RECORDS
   60 IF( IOPTS(2) .EQ. REFS ) GO TO 70
      NPARMS = 32 + KOPT
      IF( IOPTS(2) .EQ. TRAC ) NPARMS = 20
      IF( IOPTS(2) .EQ. DAMG ) NPARMS = 12+ KOPT
      CALL DTAREAD( CHNGPRM, NPARMS, IDFLG )
C   DTAREAD WILL CALL DECODLN WHICH WILL STORE THE REC LENGTH IN LNTHREC
```

```
      IF( IDFLG .EQ. 0 ) GO TO 58
      IF( IOPTS(2) .EQ. TRAC ) CALL RVSTRAC(CHNGPRM,RECORD,IFILE)
      IF( IOPTS(2) .EQ. DAMG ) CALL RVSDAMG(CHNGPRM,RECORD,IFILE)
      IF( IOPTS(2) .EQ.IBOTH ) CALL RVSBOTH(CHNGPRM,RECORD,IFILE)
C  REWRITE EXISTING RECORD WITH NEW DATA
C  DETERMINE RECORD LENGTH FOR REVISED RECORD
      ICNT = NPARMS + 1
      IF( IOPTS(2) .EQ. TRAC ) GO TO 65
      IF( IOPTS(2) .EQ. IBOTH ) GO TO 63
      ICNT = NPARMS + 21
C  SELECT MAXIMUM OF OLD REC LENGTH + NUMBER OF DATA WORDS IN REVISION
   63 LNTHREC = MAX0( LNTHREC, ICNT )
   65 CALL WRITMS( IFILE, RECORD, LNTHREC, ID,-1 )
      GO TO 90
C
C     ********************************************************************
C  TREAT CHANGES TO REFERENCES ONLY
C     ********************************************************************
C
   70 READ ( 5, 801 ) ID
      DO 72 I = 1, LNTHFIL
      IF( KEY(2*I) .EQ. ID ) GO TO 74
   72 CONTINUE
C  NO MATCH FOUND IN NAMED INDEX FOR SPECIFIED "ID"
      PRINT 910, IDXTRA
      GO TO 90
C  IMPLEMENT CHANGES TO REFERENCES
   74 CALL DECODLN( KEY(2*I+1), LNTHREC )
      CALL READMS( IFILE, RECORD, LNTHREC, ID )
      IF( IREFS(1) .EQ. -0 ) RECORD(21) = BLANK
      IF( IREFS(1) .EQ. 0 ) GO TO 741
      RECORD(21) = TRACREF
C  MUST LINK INDICIES TO DATA WORD + NOT REF BECAUSE SINGLE REF CAN BE USED
C  FOR MORE THAN ONE DATA WORD
  741 IR = 3
C  K9 IS THE LAST OUTPUT DATA ELEMENT FOR WHICH THE REF IS TO BE CHANGED
      K9 = 33 + KOPT
      DO 78 I = 22, K9
      IF( I .GT. 24 ) IR = 2
      IF( I .GT. 26 ) IR = 1
C  CHECK FOR NO CHANGE TO THIS REFERENCE
      IF( IREFS(I-23+IR) .EQ. 0 )  GO TO 78
C  CHANGE OR DELETE OLD REF + STORE THIS CHANGE IN OUTPUT RECORD
      IF( IREFS(I-23 + IR ) .NE. -0 ) GO TO 75
C  DELETE OLD REFERENCE
      II = IZERO
      CALL REFENCD( RECORD(I), II ), RETURNS (77)
      GOTO 78
C  CHANGE OLD REFERENCE
   75 II = IREFS(I-23 + IR )
      CALL REFENCD( RECORD(I), II ), RETURNS(77)
      GO TO 78
   77 PRINT 900, I, ID, RECORD(I), II
   78 CONTINUE
C  REWRITE EXISTING RECORD WITH REFERENCES CHANGED
      CALL WRITMS( IFILE, RECORD, LNTHREC, ID, 1 )
      GO TO 90
```

```
C  TREAT CASE OF ENTIRE RECORD TO BE DELETED
C  DELETE ID FROM NAMED INDEX ARRAY + INCREMENT COUNTER
   80 LN = LNTHFIL - 1
      DO 83 I = 1, LNTHFIL
      IF( KEY(2*I) .NE. ID ) GO TO 83
      DO 81 J = I, LN
      KEY(2*J) = KEY(2*J+2)
   81 KEY(2*J+1) = KEY(2*J+3 )
      GO TO 85
   83 CONTINUE
C  IF IT FALLS THRU END OF LOOP, NO MATCH FOUND FOR "ID"
      PRINT 910, ID
      GO TO 90
   84 PRINT 810
      GO TO 92
C  INCREMENT COUNTER OF RECORDS DELETED
   85 IDELETE = IDELETE + 1
      LNTHFIL = LNTHFIL - 1
C  CHECK FOR LAST DATA SET
   90 IF( LAST .NE. LLAST ) GO TO 10
C  IF THERE HAVE NOT BEEN ANY RECORDS ADDED OR DELETED, CALL EXIT
   92 IF( INEWREC .EQ. 0 .A. IDELETE .EQ. 0 ) CALL EXIT
C  SORT NAMED INDEX ARRAY
C  WRITE A NEW CYCLE ON THE FILE WITH ADDITIONS AND DELETIONS AS WELL AS CHANGES
  95  CALL SORTKEY( KEY,  LNTHFIL )
C  TRANSFER INDEX DATA TO OUTPUT FILE INDEX
      DO 98 I = 1,LNTHFIL
      KEY1(2*I) = KEY(2*I)
   98 CONTINUE
C   PRINT NO. OF DEVICES, CONTENTS OF INDEX ARRAY AND INDIVIDUAL REC LENGTHS
      PRINT 940, LNTHFIL
      DO 100 I = 1, LNTHFIL
      CALL DECODLN( KEY(2*I+1), LNTHREC )
      CALL READMS(IFILE, RECORD, LNTHREC, KEY(2*I) )
      CALL WRITMS( JFILE, RECORD, LNTHREC, KEY1(2*I), 0 )
      CALL DECODLN( KEY1(2*I+1), LN )
      PRINT 950, KEY1(2*I), LNTHREC, LN, ( RECORD(IJ),IJ= 1,LNTHREC )
  100 CONTINUE
      CALL CLOSMS(IFILE)
      CALL CLOSMS(JFILE)
      STOP
  800 FORMAT(I1, A6, A4, A10, 13(O3,1X), O3, A4 )
  801 FORMAT( A10 )
  810 FORMAT( 1H1, *END OF FILE ENCOUNTERED WHILE ATTEMPTING TO READ NEX
     1T DATA SET* / 1X, *JOB ENDED BASED ON DATA ALREADY READ * )
  900 FORMAT( 1H0,// * REFERENCE OUT OF RANGE FOR DATA ITEM NO. * I5,
     1 * FOR DEVICE * , A10, * VALUE IS * E12.3, * REF IS *, O3 / * OLD
     2REFERENCE NOT CHANGED * )
  905 FORMAT( 1H0, /// * NUMBER OF RECORDS IN THE FILE IS *, I5 )
  910 FORMAT( 1H0 // * NO MATCH FOUND IN NAMED INDEX ARRAY FOR DEVICE TY
     1PE WHIC WAS TO ONLY HAVE REFERENCES CHANGED, ID WAS *, A10, /
     2 * THESE CHANGES WERE THEREFORE IGNORED + PROCESSING CONTINUES * )
  915 FORMAT( 1H0,///*EOF NOT REACHED AFTER *, I5, * DEVICE TYPES*)
  925 FORMAT(1H0///* ATTEMPTED TO CREATE A NEW DATA REC WITH THE SAME ID
     1 AS AN OLD REC, THIS NEW DATA SET WAS IGNORED + PROCESSING CONTINU
     2ES* / * ID WAS *, A10, * IOPTS(1) WAS *, A6, * IOPTS(2) WAS *, A4)
  935 FORMAT(1H0///* ATTEMPTED TO CHANGE AN EXISTING RECORD IN THE DATA
```

```
     1FILE, AND THE SPECIFIED "ID" WAS NOT FOUND IN THE NAMED INDEX ARRA
     2Y *,/ *THIS DATA SET WAS IGNORED + PROCESSING CONTINUES */ * ID WA
     3S *, A10, * IOPTS(1) WAS *, A6, * IOPTS(2) WAS *, A4 //1H  )
  940 FORMAT(1H0, *THERE ARE NOW *, I4, * RECORDS ON THE MASS STORAGE FI
     1LE*, / 1X, *DEVICE*, 11X, *REC LENGTH* // 1H   )
  950 FORMAT( 1X, A10, 2X, I4, 2X, I4,5(2X,O20)/( 25X,O20,2X,O20,2X,O20,
     1 2X, O20, 2X, O20 ) )
      END
      SUBROUTINE RVSTRAC( A, RECORD, IFILE )
C  THIS SUBROUTINE WILL CHECK FOR ADDITIONS OR DELETIONS TO THE EXISTING RECORD
C
C  A MINUS ZERO  (-0)   IN THE  A  ARRAY OR THE  IREFS  ARRAY WILL INDICATE THAT
C  THE CORRESPONDING DATA VALUE IS TO BE DELETED FROM THE EXISTING RECORD.
C  A  NON ZERO  (  NON BLANK  FOR IREFS(1) ) VALUE WILL INDICATE THAT ANEW VALUE
C  IS TO BE STORED IN THE DATA RECORD
C
C  THIS SUBROUTINE ASSUMES THAT "ID" HAS ALREADY BEEN CHECKED AGAINST THE
C  CONTENTS OF THE KEY NAMED INDEX ARRAY , AND A MATCH WAS FOUND
      COMMON /A/ID, LNTHFIL, KOPT, IOPTS(2), IREFS(15), IDXTRA, IBLANK,
     1 IZERO,IBOTH, TRAC, DAMG, XNEGIND, KEY(501), LNTHREC, CHNGE
      DIMENSION RECORD(50), A(1)
      EQUIVALENCE ( TRACREF, IREFS(1) )
    1 CALL READMS( IFILE, RECORD(1), LNTHREC, ID )
      DO 10 I = 1,20
      IF( A(I) .EQ. 0. ) GO TO 10
      IF( A(I) .EQ. -0. ) GO TO 5
C  NEW VALUE TO BE PLACED IN THE RECORD
      RECORD(I) = A(I)
      GO TO 10
    5 RECORD(I) = XNEGIND
   10 CONTINUE
      RECORD(21) = TRACREF
   20 CONTINUE
      IF( IOPTS(2) .EQ. IBOTH ) GO TO 30
      RETURN
      ENTRY RVSDAMG
      CALL READMS( IFILE, RECORD, LNTHREC, ID )
C  SET INDICIES FOR DO LOOP TO HANDLE DAMAGE + NON-STD. TRAC PARAMETERS
      IT1 = 1
      IT2 = 12
   30 IR = 2
      J = 21
      IT2= IT2 + KOPT
      DO 60 I = IT1, IT2
      J = J + 1
C  IS DATA TO BE DELETED...IF SO, REF IS ALSO DELETED
      IF( A(I) .NE. -0. ) GO TO 33
      RECORD(J) = XNEGIND
      GO TO 60
   33 IF( I .GT. IT1 + 2 ) IR = 1
      IF( I .GT. IT1 + 4 ) IR = 0
C  IS THERE A NEW VALUE FOR CURRENT DATA ITEM...
      IF( A(I) .EQ. 0. ) GO TO 38
C  IS THERE A NEW REFERENCE FOR THIS NEW DATA VALUE
      IF( IREFS(I-IT1+IR) .GT. 0 ) GO TO 36
C  IS THE OLD REFERENCE TO BE DELETED...
      IF( IREFS(I-IT1+IR) .NE. -0 ) GO TO 34
```

```
C   TREAT CASE OF NEW DATA WITHOUT REF + OLD REFERENCE TO BE DELETED
       CALL REFENCD( A(I), IZERO ), RETURNS(40)
       RECORD(J) = A(I)
       GO TO 60
C   TREAT CASE OF NEW DATA TO BE COMBINED WITH OLD REFERENCE
    34 CALL FINDREF( RECORD(J), IRF )
       CALL REFENCD( A(I), IRF ), RETURNS(42)
       RECORD(J) = A(I)
       GO TO 60
C   TREAT CASE OF NEW DATA VALUE + NEW REFERENCE
    36 CALL REFENCD( A(I), IREFS(I-IT1+IR) ), RETURNS(44)
       RECORD(J) = A(I)
       GO TO 60
C   IS NEW REFERENCE TO BE STORED IN THE OLD DATA VALUE...
    38 IF( IREFS(I-IT1+IR) .EQ. 0 ) GO TO 60
C   IS OLD REF TO BE DELETED FROM OLD DATA VALUE...
       IF( IREFS(I-IT1+IR) .EQ.-0 )   GO TO 50
C   TREAT CASE OF OLD DATA TO BE COMBINED WITH NEW REFERENCE
       CALL REFENCD( RECORD(J), IREFS(I-IT1+IR) ), RETURNS(40)
       GO TO 60
C   DELETE OLD REFERENCE + LEAVE OLD DATA VALUE
    50 CALL REFENCD( RECORD(J), IZERO ), RETURNS(40)
       GO TO 60
C   TREAT ERROR RETURNS FROM REFENCD
    40 PRINT 900, J, ID, A(I), IZERO
       GO TO 60
    42 PRINT 900, J, ID, A(I), IRF
       GO TO 60
    44 PRINT 900, J, ID, A(I), IREFS(I-IT1+IR)
       GO TO 60
    46 PRINT 900, J, ID, RECORD(J), IREFS(I-IT1+IR)
    60 CONTINUE
       RETURN
       ENTRY RVSBOTH
       IT1 = 21
       IT2 = 32
       GO TO 1
   900 FORMAT( 1H0,// * REFERENCE OUT OF RANGE FOR DATA ITEM NO. * I5,
     1 * FOR DEVICE * , A10, * VALUE IS * E12.3, * REF IS *, O3 / * OLD
     2REFERENCE NOT CHANGED * )
       END
       SUBROUTINE TRACTRN( A, RECORD, IFILE )
C   SUBROUTINE TO ADD PARAMETERS FOR NEW DEVICE TYPES TO EXISTING TRAC + DAMAGE
C   DATA BASE
C                      A   IS TRAC AND/OR DAMAGE PARAMETER ARRAY
C
C                      RECORD  IS OUTPUT RECORD FOR WRITING TO MASS STORAGE
C
C                      IREFS ARE REFERENCES ON THE DATA SOURCES
C                      IFILE  IS THE LOGICAL MASS STORAGE UNIT NUMBER
C
       COMMON /A/ID, LNTHDIO, KOPT, IOPTS(2), IREFS(15), IDXTRA, IBLANK,
     1 IZERO,IBOTH, TRAC, DAMG, XNEGIND, KEY(501), LNTHREC, CHNGE
       DIMENSION RECORD(50), A(1)
       EQUIVALENCE ( TRACREF, IREFS(1) ), (IBLANK, BLANK )
C   IT3 I THE MAX. NO. OF NON-OPTIONAL OUTPUT PARAMETERS
       IT3 = 37
```

```
   10 DO 20 I = 1,20
      RECORD(I) = A(I)
      IF( A(I) .EQ. 0 ) RECORD(I) = XNEGIND
   20 CONTINUE
C  IF THERE WAS NO TRAC REFERENCE, A BLANK IS STORED
      RECORD(21) = TRACREF
C  IS THIS A BOTHTRN ENTRY....
      IF(  IOPTS(2) .EQ.IBOTH ) GO TO 50
C  STORE NEGATIVE INDEFINITE IN DAMAGE AND NON-STD. TRAC PARAM PORTION OF RECORD
      DO 30 I = 22, IT3
   30 RECORD(I) = XNEGIND
      RETURN
      ENTRY DAMGTRN
      DO 40 I = 1,20
C  STORE NEGATIVE INDEFINITE IN STD. TRAC PARAMETER PORTION OF THE RECORD
   40 RECORD(I) = XNEGIND
      RECORD(21) =  BLANK
C  SET INDICIES FOR DO LOOP TO HANDLE DAMAGE + NON-STD. PARAMETERS
C  IT1 IS THE SUBSCRIPT OF THE FIRST DAMAGE DATA WORD IN INPUT A ARRAY
C  J IS THE INDEX FOR THE PARAMETERS IN THE OUTPUT RECORD
      IT1 = 1
      IT2 = 12
   50 IR = 2
      J = 21
      IT2 = IT2 + KOPT
      DO 60 I = IT1, IT2
      J = J + 1
C  TEST FOR PRESENCE OF DATA VALUE
      IF( A(I) .EQ. 0. ) GO TO 55
      IF( I .GT. IT1 + 2 ) IR = 1
      IF( I .GT. IT1 + 4 ) IR = 0
      CALL REFENCD( A(I), IREFS(I-IT1+IR) ), RETURNS(56)
   53 RECORD(J) = A(I)
      GO TO 60
   55 RECORD(J) = XNEGIND
      GO TO 60
C  ERROR RETURN ON REFERENCE ID OUT OF RANGE, STORE ZERO FOR REF.
   56 IZERO = 0
      PRINT 900, J, ID,A(I), IREFS(I-IT1+IR)
      CALL  REFENCD( A(I), IZERO ), RETURNS(56)
      RECORD(J) = A(I)
   60 CONTINUE
      RETURN
      ENTRY BOTHTRN
      IT1 = 21
      IT2 = 32
      GO TO 10
  900 FORMAT( 1H0,// * REFERENCE OUT OF RANGE FOR DATA ITEM NO. * I5,
     1 * FOR DEVICE * , A10, * VALUE IS * E12.3, * REF IS *, O3 / * OLD
     2REFERENCE NOT CHANGED * )
      END
      SUBROUTINE SORTKEY( KEY, LENGTH )
C  SUBROUTINE TO SORT THE NAMED KEY ARRAY
      DIMENSION KEY(1)
      L1 = LENGTH - 1
      DO 30 I = 1, L1
      K2 = KEY(2*I)
```

```
      K3 = KEY(2*I+1)
      I1 = I + 1
      DO 20 J = I1, LENGTH
      IF( KEY(2*I) .LT. KEY(2*J) ) GO TO 20
      KEY(2*I) = KEY(2*J)
      KEY(2*I+1) = KEY(2*J+1)
      KEY(2*J) = K2
      KEY(2*J+1) = K3
      K2 = KEY(2*I)
      K3 = KEY(2*I+1)
   20 CONTINUE
   30 CONTINUE
      RETURN
      END
      SUBROUTINE DTAREAD ( A, N, IDFLG )
      DIMENSION A(1)
      COMMON /A/ID, LNTHFIL, KOPT, IOPTS(2), IREFS(15), IDXTRA, IBLANK,
     1 IZERO,IBOTH, TRAC, DAMG, XNEGIND, KEY(501), LNTHREC, CHNGE
      INTEGER CHNGE
C   READ INPUT DATA CARDS
      J = 7
      READ (5,850) ID, ( A(I), I= 1,J )
    5 IF( J.GE. N ) GO TO 10
      J1 = J + 1
      J =  J + 8
      READ (5,860 ) ( A(I), I = J1,J )
      GO TO 5
   10 DO 20 I = 1,LNTHFIL
      IF( KEY(2*I) .EQ. ID )  GO TO 40
   20 CONTINUE
C   NO MATCH IN NAMED KEY INDEX WITH CURRENT "ID"
      IDFLG = 0
      RETURN
C   MATCH FOUND IN NAMED KEY INDEX WITH CURRENT "ID"
   40 IDFLG = ID
      IF( IOPTS(1) .NE. CHNGE ) GO TO 50
      CALL DECODLN( KEY(2*I+1), LNTHREC )
   50 RETURN
  850 FORMAT( A10, 7E10.1 )
  860 FORMAT( 8E10.1 )
      END
      SUBROUTINE REFENCD( DTA, IREF ), RETURNS(I)
C   SUBROUTINE TO STORE AN ENCODED REFERENCE INTO THE THREE LEAST SIGNIFICANT
C   OCTAL  CHARACTERS OF A REAL VARIABLE DATA WORD
      DATA MASK1 / 77777777777777777000B /
      IF( IREF .GT. 777B ) GO TO 20
      DTA = ( AND( DTA, MASK1 )  .OR. IREF )
      RETURN
   20 RETURN I
      END
      SUBROUTINE DECODLN( K, LNTHREC )
C   SUBROUTINE TO DECODE THE RECORD LENGTH FROM THE DATA IN NAMED INDEX ARRAY
C   THIS DATA IS APPARENTLY CONTAINED IN THE 12TH THRU 19TH OCTAL CHARACTERS OF
C   THE ODD NUMBERED WORDS OF THE NAMED KEY  INDEX, EXCLUSIVE OF THE FIRST WORD
      DATA MASK / 07777777700000000000B /
C   ZERO OUT ALL OCTAL CHARACTERS EXCEPT THE RECORD LENGTH
      KK = AND( K, MASK )
```

```
C   SHIFT RECORD LENGTH TO LOW ORDER BITS OF THE DATA WORD, IN INTEGER FORMAT
      LNTHREC = SHIFT( KK, -33 )
      RETURN
      END
      SUBROUTINE FINDREF( DTA, IREF )
C   SUBROUTINE TO DECODE A 3 OCTAL DIGIT REFERENCE STORED IN THE 3 LEAST
C   SIGNIFICANT OCTAL CHARACTERS OF A REAL VARIABLE DATA WORD
      DATA MASK2 / 00000000000000000777B /
      IREF = AND(  DTA, MASK2 )
      RETURN
      END
```

APPENDIX B

```
EMCPR,CM64000,T35.
TASK,TNEM71603,PWPRMPT,TRTS. RUZIC
ATTACH,TAPE11,DIODES,ID=EM71603.
FTN(R=2,L)
LGO.
W
      PROGRAM TSTUPDT  (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE11)
      DIMENSION DTA(40), OUTPT(60), IOUTPT(60), KEY(501), IDTA(40)
      EQUIVALENCE( OUTPT,IOUTPT ), ( DTA, IDTA )
      DATA IBLANK / 10H          /
C    PROGRAM TO READ + CHECKOUT THE UPDATED SEMICONDUCTOR DATA FILE
      CALL OPENMS( 11, KEY, 500, 1 )
      KOUNT = 0
C    PRINT OUTPUT LABEL
      PRINT 100
C    FIND NUMBER OF ELEMENTS IN DATA FILE
      DO 20 I = 2, 500, 2
      IF( KEY(I) .NE. 0 ) GO TO 20
      LAST = (I/2) -1
      GO TO 25
   20 CONTINUE
      PRINT 200
  200 FORMAT( * END OF INDEX NOT FOUND, JOB TERMINATED * )
      STOP
   25 DO 50 J = 1, LAST
      CALL DECODLN( KEY(2*J+1), LNTHREC )
C    BLANK OUT DATA AREA
      CALL BLNKFIL( DTA, 40 )
      CALL READMS( 11, DTA, LNTHREC, KEY(2*J) )
C    CHECK STD. TRAC DATA WORDS FOR EXISTENCE OF DATA
      DO 30 I = 1,9
      ENCODE(10, 901, IOUTPT(I) )IBLANK
      IF(IDTA(I).LT.40007777777777777000B)ENCODE(10,90,IOUTPT(I))IDTA(I)
   30 CONTINUE
      IOUTPT(10) = IDTA(10)
C    CHECK IF RECORD CONTAINS DAMAGE PARAMETERS
      IF(LNTHREC.GE.11)GO TO 35
      PRINT 800,KEY(2*J),(IOUTPT(I),I=1,LNTHREC)
      GO TO 50
   35 DO 40 I = 11, LNTHREC
      ENCODE(10,901,IOUTPT(I) ) IBLANK
      ENCODE(10,901,IOUTPT(I+LNTHREC-10) ) IBLANK
C    IS DATA PRESENT FOR THIS PARAMETER
      IF(IDTA(I) .GE. 40007777777777777000B ) GO TO 40
      CALL FINDREF ( DTA(I), IREF )
      ENCODE(10,  90, IOUTPT(I) ) IDTA(I)
      CALL OCTINTG( IREF, IR )
      ENCODE( 3, 903, IOUTPT(I+LNTHREC-10) ) IR
   40 CONTINUE
      KOUNT = KOUNT + 1
      IF( KOUNT .LT. 12 ) GO TO 45
      PRINT 100
      KOUNT = 1
   45 L1 = LNTHREC + 1
      L2 = 2* LNTHREC - 10
      PRINT 800,KEY(2*J), ( IOUTPT(I), I = 1, LNTHREC )
      PRINT 801, ( IOUTPT(I) , I = L1, L2 )
```

```
   50 CONTINUE
      PRINT 850
      STOP
C     $NUMBER OF CHARACTERS PER LINE IN FORMAT 100 ARE 129/126/79/134 *
  100 FORMAT( 1H1, *DEVICE*, 10X, *IS*, 10X, *MD*, 9X, *RDL*, 9X, *CDO*,
     1 9X, *VDB1*, 9X, *TD*, 9X, *IPPD*, 9X, *VB*, 7X,*SURGE Z*, 5X,
     2 *TRAC REF* / 15X,  *R BULK*,5X, *SURGE Z*, 5X, *SURGE Z*, 5X,
     3 *K DAMAGE*, 4X, *K DAMAGE*, 4X, *K DAMAGE*, 8X, *K*, 11X, *K*,
     4 11X, *K*,11X,*K*   / 14X, *FORWARD*, 3X, *REVRSE 1US*, 4X,
     5 *REV 10US*, 3X,*REV <50NS*,3X,*REV >50NS*, 4X, *FORWARD*/ 15X,
     6 */ REF.*,6X,*/ REF.*,6X,*/ REF.*,6X,*/ REF.*,6X,*/ REF.*,6X,
     7 */ REF.*,6X,*/ REF.*,6X,*/ REF.*,6X,*/ REF.*,6X,*/ REF.*,6X )
  800 FORMAT( //1H ,  11(A10,2X) / 13X, 11(A10,2X) )
  801 FORMAT(16X, 11(A3,9X) )
  802 FORMAT( 1X,O20, 2X, O20, 2X, O20, 2X, O20, 2X, O20, 2X, O20, / )
  850 FORMAT( 1H1, *TEST OF FILE COMPLETED * )
   90 FORMAT(E10.3)
  901 FORMAT(A10)
  902 FORMAT(O20)
  903 FORMAT( I3 )
      END
      SUBROUTINE OCTINTG ( IREF, IR )
C
C     THIS SUBROUTINE CONVERTS AN OCTAL INTEGER CONSTANT TO A DECIMAL INTEGER
C     WHICH WILL PRINT OUT UNDER AN I FORMAT EXACTLY AS THE OCTAL NUMBER
C     WOULD PRINT UNDER AN O FORMAT........THIS IS A NECESSARY INTERMEDIATE
C     STEP IN GOING FROM O FORMAT TO A FORMAT
C
C     THE ROUTINE IS CURRENTLY LIMITED TO HANDLE ONLY 3 DIGIT NUMBERS
C
      DATA MS1, MS2, MS3 / 00000000000000000007B, 00000000000000000070B,
     1    00000000000000000700B /
      K1 = AND( IREF, MS1 )
      K2 = AND( IREF, MS2 )
      KK2 = SHIFT( K2, -3 )
      K3 = AND( IREF, MS3 )
      KK3 = SHIFT( K3, -6 )
      IR = 100 *KK3 + 10 *KK2 + K1
      RETURN
      END
      SUBROUTINE BLNKFIL( A, N )
      DIMENSION A(1)
      DATA B/40007777777777777000B /
      DO 10 I = 1,N
   10 A(I) = B
      RETURN
      END
      SUBROUTINE DECODLN( K, LNTHREC )
C   SUBROUTINE TO DECODE THE RECORD LENGTH FROM THE DATA IN NAMED INDEX ARRAY
C   THIS DATA IS APPARENTLY CONTAINED IN THE 12TH THRU 19TH OCTAL CHARACTERS OF
C   THE ODD NUMBERED WORDS OF THE NAMED KEY  INDEX, EXCLUSIVE OF THE FIRST WORD
      DATA MASK / 07777777700000000000B /
C   ZERO OUT ALL OCTAL CHARACTERS EXCEPT THE RECORD LENGTH
      KK = AND( K, MASK )
C   SHIFT RECORD LENGTH TO LOW ORDER BITS OF THE DATA WORD, IN INTEGER FORMAT
      LNTHREC = SHIFT( KK, -33 )
      RETURN
```

```
      END
      SUBROUTINE FINDREF( DTA, IREF )
C   SUBROUTINE TO DECODE A 3 OCTAL DIGIT REFERENCE STORED IN THE 3 LEAST
C   SIGNIFICANT OCTAL CHARACTERS OF A REAL VARIABLE DATA WORD
      DATA MASK2 / 00000000000000000777B /
      IREF = AND( DTA, MASK2 )
      RETURN
      END
W
V
```

```
EMCPR,CM64000,T100.
TASK,TNEM71603,PWPRMPT,TRTS. RUZIC
ATTACH,TAPE11,TRANS,ID=EM71603.
FTN.
LGO.
W
      PROGRAM TRNUPDT  (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE11)
      DIMENSION DTA(50), OUTPT(60), IOUTPT(60), KEY(525), IDTA(50)
      DIMENSION NCHAR(4)
      EQUIVALENCE( OUTPT,IOUTPT ), ( DTA, IDTA )
      DATA IBLANK / 10H          /
      DATA NCHAR/ 7, 8, 6, 8 /
C    PROGRAM TO READ + CHECKOUT THE UPDATED SEMICONDUCTOR DATA FILE
      REWIND 11
      CALL OPENMS( 11, KEY, 523, 1 )
      KOUNT = 0
C    PRINT OUTPUT LABEL
      CALL PRNTHD
C    FIND NUMBER OF ELEMENTS IN DATA FILE
      DO 20 I = 2, 522, 2
      IF( KEY(I) .NE. 0 ) GO TO 20
      LAST = (I/2) -1
      GO TO 25
   20 CONTINUE
      PRINT 200
  200 FORMAT( * END OF INDEX NOT FOUND, JOB TERMINATED * )
      STOP
   25 DO 50 J = 1, LAST
      CALL DECODLN( KEY(2*J+1), LNTHREC )
C    BLANK OUT DATA AREA
      CALL BLNKFIL( DTA, 50 )
      CALL READMS( 11, DTA, LNTHREC, KEY(2*J) )
C    CHECK STD. TRAC DATA WORDS FOR EXISTENCE OF DATA
      DO 30 I = 1,20
      ENCODE(10, 901, IOUTPT(I) )IBLANK
      IF(IDTA(I).LT.40007777777777777000B)ENCODE(10,90,IOUTPT(I))IDTA(I)
   30 CONTINUE
      IOUTPT(21) = IDTA(21)
C     IF"DTA" INPUT RECORD DID NOT CONTAIN TRAC REF. SET IT TO BLANK FIELD
      IF(LNTHREC .LT. 21 ) IOUTPT(21) = IBLANK
C    INITIALIZE MAX POSSIBLE OUTPUT FIELD AFTER STD TRAC PARAMETERS TO BLANKS
      DO 38 I = 22, 60
      ENCODE(10,901,IOUTPT(I) ) IBLANK
   38 CONTINUE
C    CHECK IF RECORD CONTAINS DAMAGE PARAMETERS
      DO 40 I = 22, 33
C    IS DATA PRESENT FOR THIS PARAMETER
      IF(IDTA(I) .GE. 40007777777777777000B ) GO TO 40
      CALL FINDREF ( DTA(I), IREF )
      ENCODE(10,  90, IOUTPT(I) ) IDTA(I)
      CALL OCTINTG( IREF, IR )
      ENCODE( 3, 903, IOUTPT(I+LNTHREC-21) ) IR
   40 CONTINUE
      DO  42 I = 34, 37
      IF( I .GT. LNTHREC ) GO TO 43
      IF(IDTA(I) .GE. 40007777777777777000B ) GO TO 42
      CALL FINDREF ( DTA(I), IREF )
```

APPENDIX B

```
      JK= I - 33
      N = NCHAR(JK)
      ENCODE( 10 , 1, FMT ) N
      ENCODE(N,FMT, IOUTPT(I) )  IDTA(I)
      CALL OCTINTG( IREF, IR )
      ENCODE( 3, 903, IOUTPT(I+LNTHREC-21) ) IR
   42 CONTINUE
   43 IF(LNTHREC .LT. 20 ) GO TO 50
      KOUNT = KOUNT + 1
      IF( KOUNT .LT. 12 ) GO TO 45
      CALL PRNTHD
      KOUNT = 1
   45 PRINT 800,KEY(2*J), (IOUTPT(I), I = 1, 10 ), IOUTPT(22),IOUTPT(23)
      J1 = LNTHREC + 1
      J2 = LNTHREC + 2
      PRINT 810,IOUTPT(21),(IOUTPT(I), I = 11,20 ),IOUTPT(J1),IOUTPT(J2)
  810 FORMAT( 1H , A10, 10A10, 4X,A3, 7X, A3 )
      PRINT 820,(IOUTPT(I), I = 24, LNTHREC )
  820 FORMAT( 1H , 10A10, A8, A9, A7, A8 )
      J1 = LNTHREC + 3
      L2 = 2* LNTHREC - 21
      PRINT 830, (IOUTPT(I), I= J1, L2 )
  830 FORMAT( 1H , 4X, 9(A3, 7X), 2( A3, 6X, A3, 5X )  )
   50 CONTINUE
      PRINT 850
      STOP
    1 FORMAT( 2H(E, I1, 3H.2) )
  800 FORMAT( 1H0, A10, 12A10 )
  850 FORMAT( 1H1, *TEST OF FILE COMPLETED * )
   90 FORMAT(E10.3)
  901 FORMAT(A10)
  902 FORMAT(O20)
  903 FORMAT( I3 )
      END
      SUBROUTINE OCTINTG ( IREF, IR )
C
C   THIS SUBROUTINE CONVERTS AN OCTAL INTEGER CONSTANT TO A DECIMAL INTEGER
C   WHICH WILL PRINT OUT UNDER AN I FORMAT EXACTLY AS THE OCTAL NUMBER
C   WOULD PRINT UNDER AN O FORMAT........THIS IS A NECESSARY INTERMEDIATE
C   STEP IN GOING FROM O FORMAT TO A FORMAT
C
C   THE ROUTINE IS CURRENTLY LIMITED TO HANDLE ONLY 3 DIGIT NUMBERS
C
      DATA MS1, MS2, MS3 / 00000000000000000007B, 00000000000000000070B,
     1   00000000000000000700B /
      K1 = AND( IREF, MS1 )
      K2 = AND( IREF, MS2 )
      KK2 = SHIFT( K2, -3 )
      K3 = AND( IREF, MS3 )
      KK3 = SHIFT( K3, -6 )
      IR = 100 *KK3 + 10 *KK2 + K1
      RETURN
      END
      SUBROUTINE BLNKFIL( A, N )
      DIMENSION A(1)
      DATA B/40007777777777777000B /
      DO 10 I = 1,N
```

```
   10 A(I) = B
      RETURN
      END
      SUBROUTINE DECODLN( K, LNTHREC )
C   SUBROUTINE TO DECODE THE RECORD LENGTH FROM THE DATA IN NAMED INDEX ARRAY
C   THIS DATA IS APPARENTLY CONTAINED IN THE 12TH THRU 19TH OCTAL CHARACTERS OF
C   THE ODD NUMBERED WORDS OF THE NAMED KEY  INDEX, EXCLUSIVE OF THE FIRST WORD
      DATA MASK / 07777777770000000000B /
C   ZERO OUT ALL OCTAL CHARACTERS EXCEPT THE RECORD LENGTH
      KK = AND( K, MASK )
C   SHIFT RECORD LENGTH TO LOW ORDER BITS OF THE DATA WORD, IN INTEGER FORMAT
      LNTHREC = SHIFT( KK, -33 )
      RETURN
      END
      SUBROUTINE FINDREF( DTA, IREF )
C   SUBROUTINE TO DECODE A 3 OCTAL DIGIT REFERENCE STORED IN THE 3 LEAST
C   SIGNIFICANT OCTAL CHARACTERS OF A REAL VARIABLE DATA WORD
      DATA MASK2 / 00000000000000000777B /
      IREF = AND(  DTA, MASK2 )
      RETURN
      END
      SUBROUTINE PRNTHD
      DIMENSION HEADS(10), HEADS1(10)
      DATA HEADS/10H   HFEN   ,10H   HFEI   ,10H    TN    ,10H    TI
     1,10H    ICS   ,10H    MC    ,10H    CCO   ,10H   VCBI   ,10H    RC
     2L   ,10H    IES   /
      DATA HEADS1/10H    ME    ,10H    CEO   ,10H   VEBI   ,10H    REL
     1 ,10H   IPPC   ,10H   IPPE   ,10H  C-BBDV  ,10H  E-BBDV  ,10H SURG
     2E ZC ,10H SURGE ZE /
      PRINT 96,HEADS
      PRINT 144,HEADS1
      PRINT 100
      PRINT 101
      RETURN
   96 FORMAT(1H1, *DEVICE*, 5X,10A10, 3X, *-----BULK R-----* )
  100 FORMAT(1H0,4X, *ZSURGE C-B*, 10X,*ZSURGE E-B*, 9X,*DAMGE K(<50NS)*
     1, 6X, *DAMGE K(>50NS)*, 8X, *K FORWARD*, 9X, *K*, 7X, *K*, 8X, *K*
     2, *K* )
  101 FORMAT(1H , 2X, 2(*1 US*, 5X, *10 US*, 7X ), 2(*C-B*, 6X, *E-B*,8X
     1 ), *C-B*, 7X, *E-B*, 6X, *C-B*, 5X, *E-B*, 6X,*C-B*,4X,*E-B*/1H )
  144 FORMAT( 1H , *TRAC REF*, 3X, 10A10, 3X, *C-B*, 7X, *E-B* )
      END
V
```

APPENDIX C.--SUBROUTINE DESCRIPTIONS

C-1. INTRODUCTION

Appendix C presents a detailed description of each of the subroutines required for the main programs described in the main body of the report. The variables used in the subroutines are defined, the purpose and use of the subroutines are explained, and the interrelations of the subroutines are specified. A detailed logical flowchart of each subroutine is included with each description.

C-2. SUBROUTINE VARIABLE DEFINITIONS AND PROGRAM EXPLANATIONS

C-2.1 <u>REVTRAC(A,RECORD,IFILE)</u>

Additional entries to REVTRAC are REVBOTH and REVDAMG.

C-2.1.1 <u>Variable Definitions</u>

A	Real array containing new data values to be put on mass storage record, of the specified ID
I	Subscript used for elements of the A array and IREFS array, as well as standard TRAC parameter elements of the RECORD array
IBOTH	Hollerith constant used to check if the BOTH option was selected
ID	Named index value for the record currently being processed
IFILE	Logical unit for the mass storage file from which the existing data record is to be read
IOPTS	Integer array containing the options selected by the user's input in the main program
IR	Integer variable used to alter the indexing of the IREFS array
IREFS	Array containing new references for the data base or, alternatively, "-0" to indicate that an existing reference is to be deleted
IRF	Variable used to save an old reference which is to be combined with a new data value
IT1	Lower index limit for damage parameters to be processed

Preceding Page Blank

IT2	Upper index limit for damage parameters to be processed
IZERO	Integer zero constant
J	Subscript used for indexing the damage parameters of the RECORD array
KOPT	Number of optional damage parameters to be included in the revised record
LNTHDIO	Number of data records contained in the diode mass storage file
LNTHREC	Number of data words in the data record being processed
RECORD	Real array containing the revised or corrected record to be written to the mass storage file
TRACREF	Variable containing the reference source for the TRAC data in the record being considered
XNEGIND	Variable containing a negative indefinite, which is stored in output words for which there are no data

C-2.1.2 Subroutine Explanation

Subroutine REVTRAC (fig. C-1) checks for additions or deletions to the existing diode data file. REVTRAC is the entry used when only TRAC data are to be added or altered in the existing record. The entry REVDAMG is used when only damage data are to be revised, and the entry REVBOTH is used when both types of data are to be revised.

This subroutine assumes that the named index was previously searched for the record ID specified. This function is performed by DTABSE.

It checks each input data field for the presence of (1) a minus zero, (2) a blank or plus zero, and (3) a nonblank, nonzero field. A minus zero is used to indicate that the corresponding word on the existing data file is to be deleted, and there is no datum with which to replace it. A blank input data field indicates that the corresponding existing data field is to be left intact. A nonzero, nonblank field must be actual data to replace existing data in the corresponding word of the data file. A notable limitation is that a new data value of precisely zero cannot be read because the program necessarily checks for blank fields, which on the CDC 6000 series computers cannot be distinguished from zeros. It is suggested that if a zero is actually desired, a data value such as 1.0E-70 should be used instead.

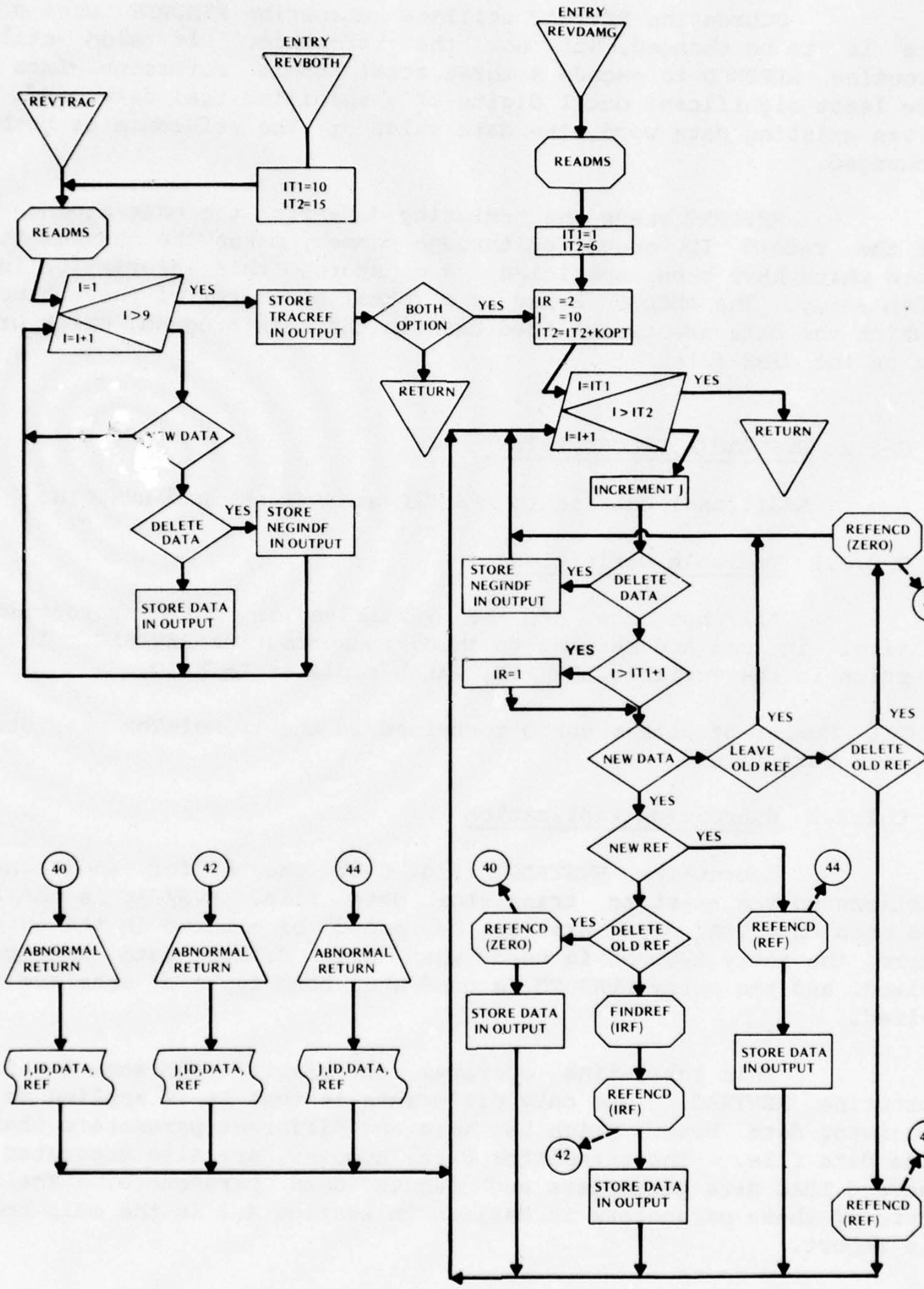

Figure C-1. Subroutine REVTRAC.

APPENDIX C

Subroutine REVTRAC utilizes subroutine FINDREF when a data value is to be changed, but not the reference. It also utilizes subroutine REFENCD to encode a three octal digit reference into the three least significant octal digits of a specified real data word. For a given existing data word, the data value or the reference or both can be changed.

REVTRAC reads the existing data from the mass storage file with the record ID specified through common, makes the changes to the record which have been specified, and stores this information in the RECORD array. The RECORD array is a formal parameter of the subroutine by which the data are transmitted back to the main program, which writes them on the disk file.

C-2.2 RVSTRAC(A,RECORD,IFILE)

Additional entries to RVSTRAC are RVSDAMG and RVSBOTH.

C-2.2.1 Variable Definitions

All but one of the variables in this subroutine are identical in use and meaning to those in subroutine REVTRAC. The one exception is the variable LNTHFIL, which replaces LNTHDIO.

LNTHFIL Number of data records contained in the transistor mass storage data file

C-2.2.2 Subroutine Explanation

Subroutine RVSTRAC (fig. C-2) checks for additions or deletions to the existing transistor data file. RVSTRAC is the entry used when only TRAC data are to be added or altered in the existing record, the entry RVSDAMG is used when only damage data are to be revised, and the entry RVSBOTH is used when both types of data are to be revised.

This subroutine operates in exactly the same fashion as subroutine REVTRAC. The only difference is that it is applied to the transistor data base, which has more and different parameters than the diode data file. The transistor data, however, are also separated into standard TRAC data parameters and damage data parameters. The exact nature of these parameters is defined in section 4.2 in the main body of this report.

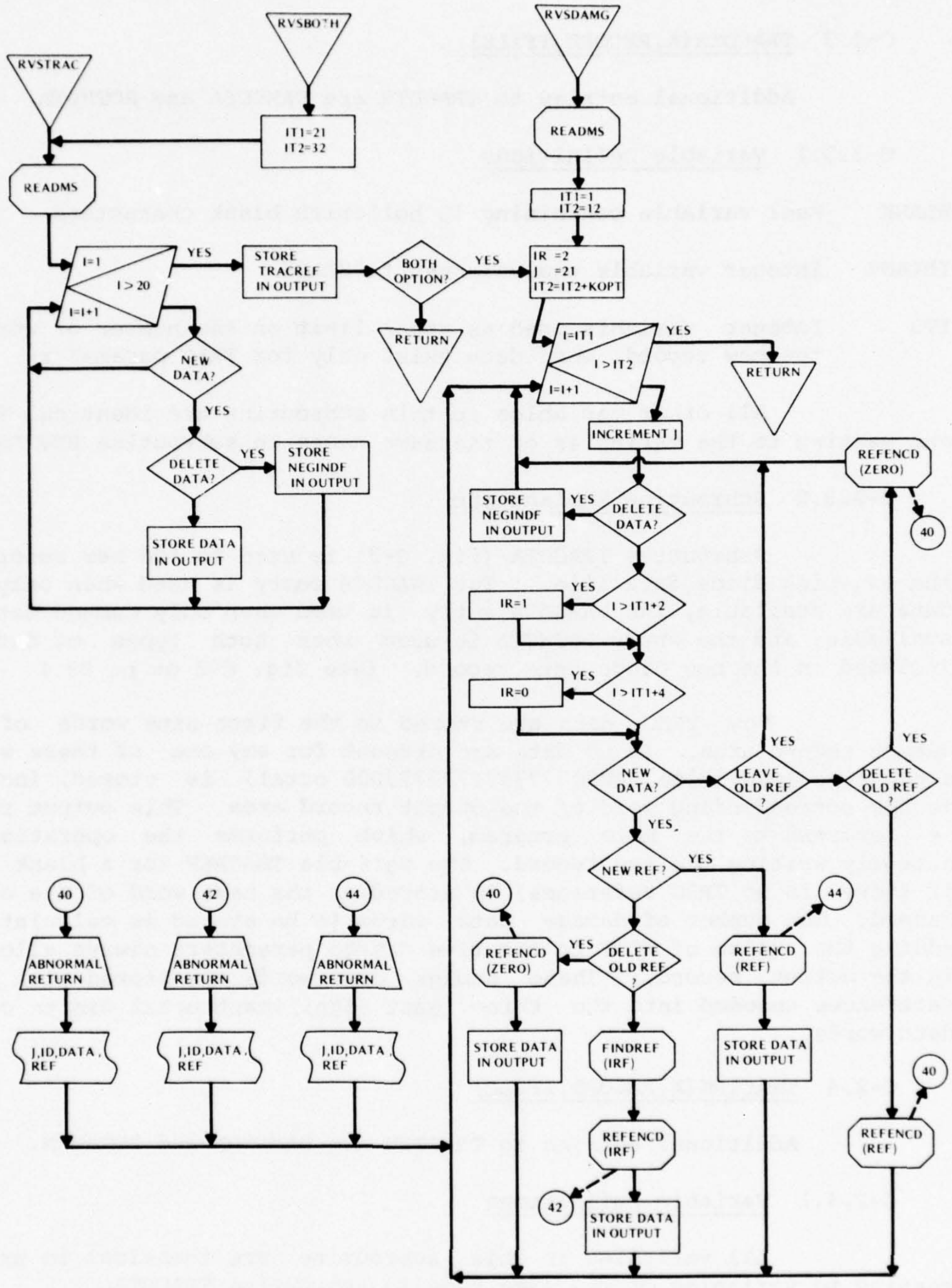

Figure C-2. Subroutine RVSTRAC.

C-2.3 TRACDTA(A,RECORD,IFILE)

Additional entries to TRACDTA are DAMGDTA and BOTHDTA.

C-2.3.1 Variable Definitions

BLANK Real variable containing 10 hollerith blank characters

IBLANK Integer variable equivalenced to BLANK

IT3 Integer variable used as upper limit on the number of words in the new record, when data exist only for TRAC parameters

All other variables in this subroutine are identical in use and meaning to the variables of the same names in subroutine REVTRAC.

C-2.3.2 Subroutine Explanation

Subroutine TRACDTA (fig. C-3) is used to add new records to the existing diode data file. The TRACDTA entry is used when only TRAC data are available, the DAMGDTA entry is used when only damage data are available, and the entry BOTHDTA is used when both types of data are included in the new diode data record. (See fig. C-3 on p. 83.)

New TRAC data are stored in the first nine words of the output record area. If no data are present for any one of these words, a negative indefinite (40007777777777777000 octal) is stored, instead, in the corresponding word of the output record area. This output record is returned to the main program, which performs the operation of actually writing the new record. The variable TRACREF (or a blank field if there is no TRAC reference) is stored as the next word of the output record. The number of damage data words to be stored is calculated by adding the value of KOPT to the five damage parameters always allocated in the output record. These damage data words are stored with their references encoded into the three least significant octal digits of the data words.

C-2.4 TRACTRN(A,RECORD,IFILE)

Additional entries to TRACTRN are DAMGTRN and BOTHTRN.

C-2.4.1 Variable Definitions

All variables in this subroutine are identical in use and meaning to variables of the same name in subroutine TRACDTA.

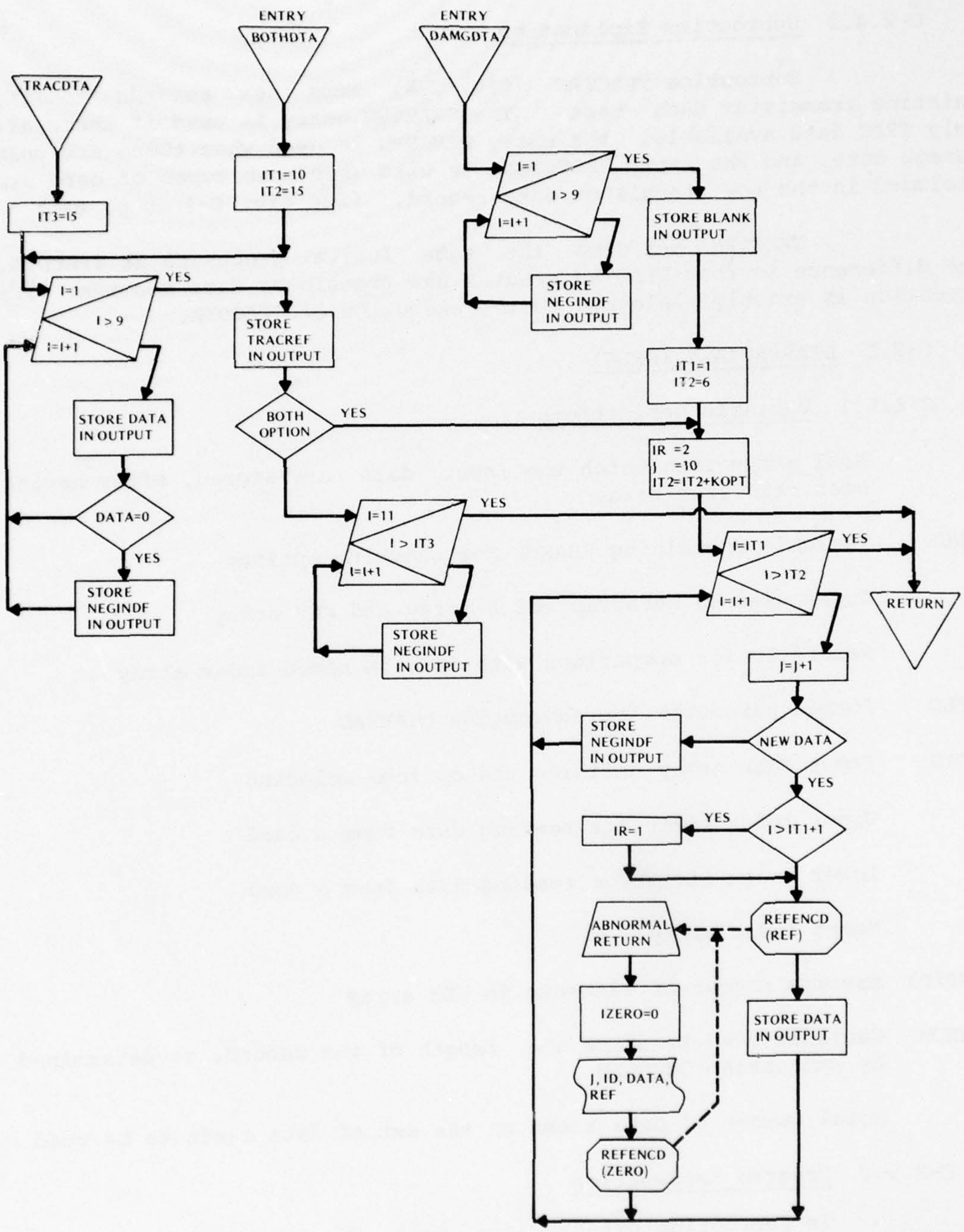

Figure C-3. Subroutine TRACDTA.

C-2.4.2 Subroutine Explanation

Subroutine TRACTRN (fig. C-4) adds new records to the existing transistor data base. The TRACTRN entry is used if there are only TRAC data available, the entry DAMGTRN is used when there are only damage data, and the entry BOTHTRN is used when both types of data are included in the new transistor data record. (See fig. C-4 on p. 85.)

TRACTRN performs the same logical processes as TRACDTA. The difference is that TRACTRN creates new transistor data records. Its execution is exactly analogous to the execution of TRACDTA.

C-2.5 DTAREAD(A,N,IDFLG)

C-2.5.1 Variable Definitions

A	Real array into which the input data are stored, after having been read from cards
CHNGE	Variable containing CHANGE for checking options
I	Index used as subscript of A array and KEY array
ID	Record ID for comparison with ID's in named index array
IDFLG	Formal parameter for subroutine DTAREAD
IOPTS	Input data array defining the options selected
J	Upper index limit for reading data from a card
J1	Lower index limit for reading data from a card
KEY	Named index array
LNTHDIO	Maximum number of elements in KEY array
LNTHREC	Variable used to store the length of the record, as determined by subroutine DECODLN
N	Total number of data items on the set of data cards to be read

C-2.5.2 Program Explanation

In subroutine DTAREAD (fig. C-5), the record ID, to be used as the named index, and the first seven data fields are read from the first data card. Succeeding data cards, each containing eight fields, are then read until the number of data fields read is greater than or

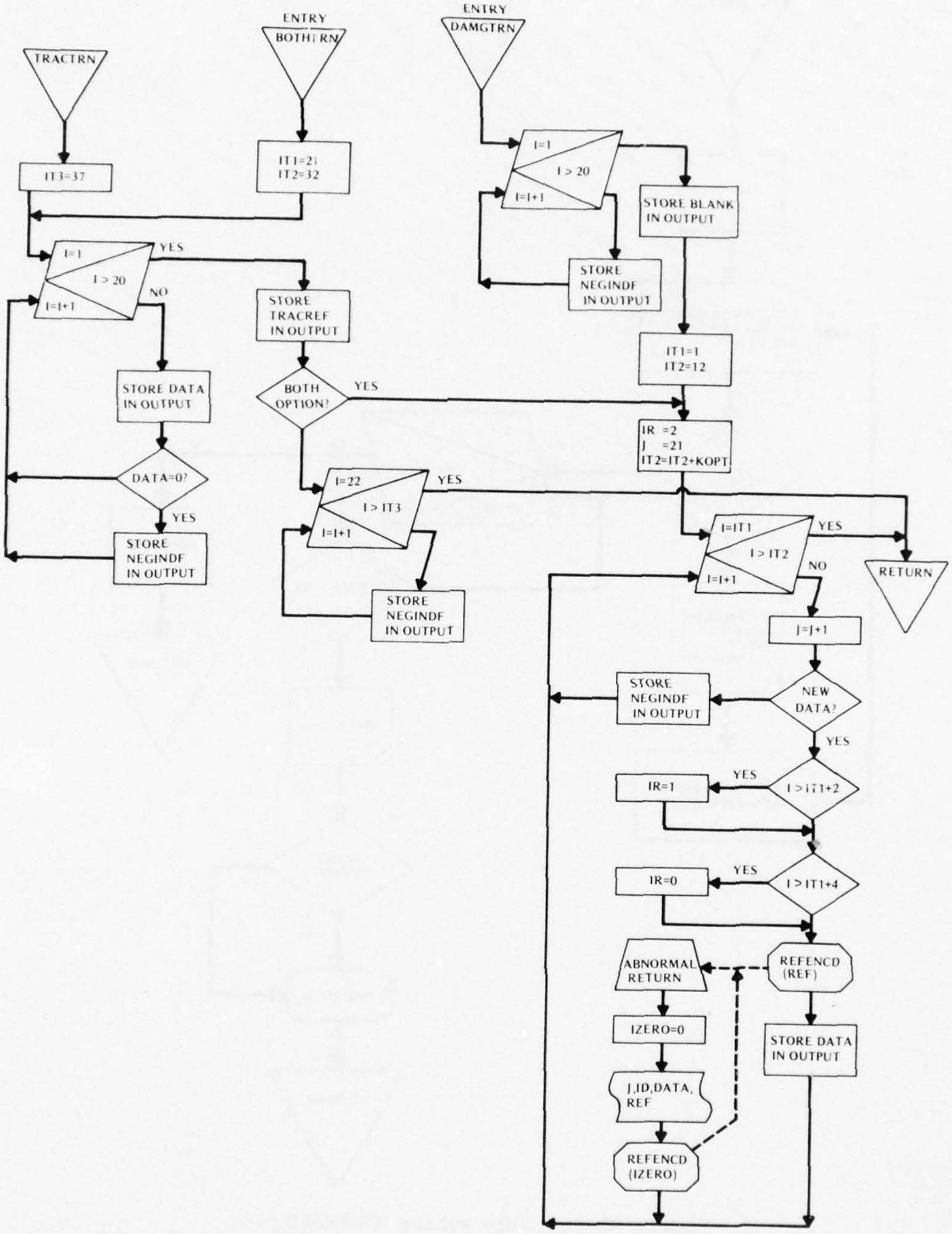

Figure C-4. Subroutine TRACTRN.

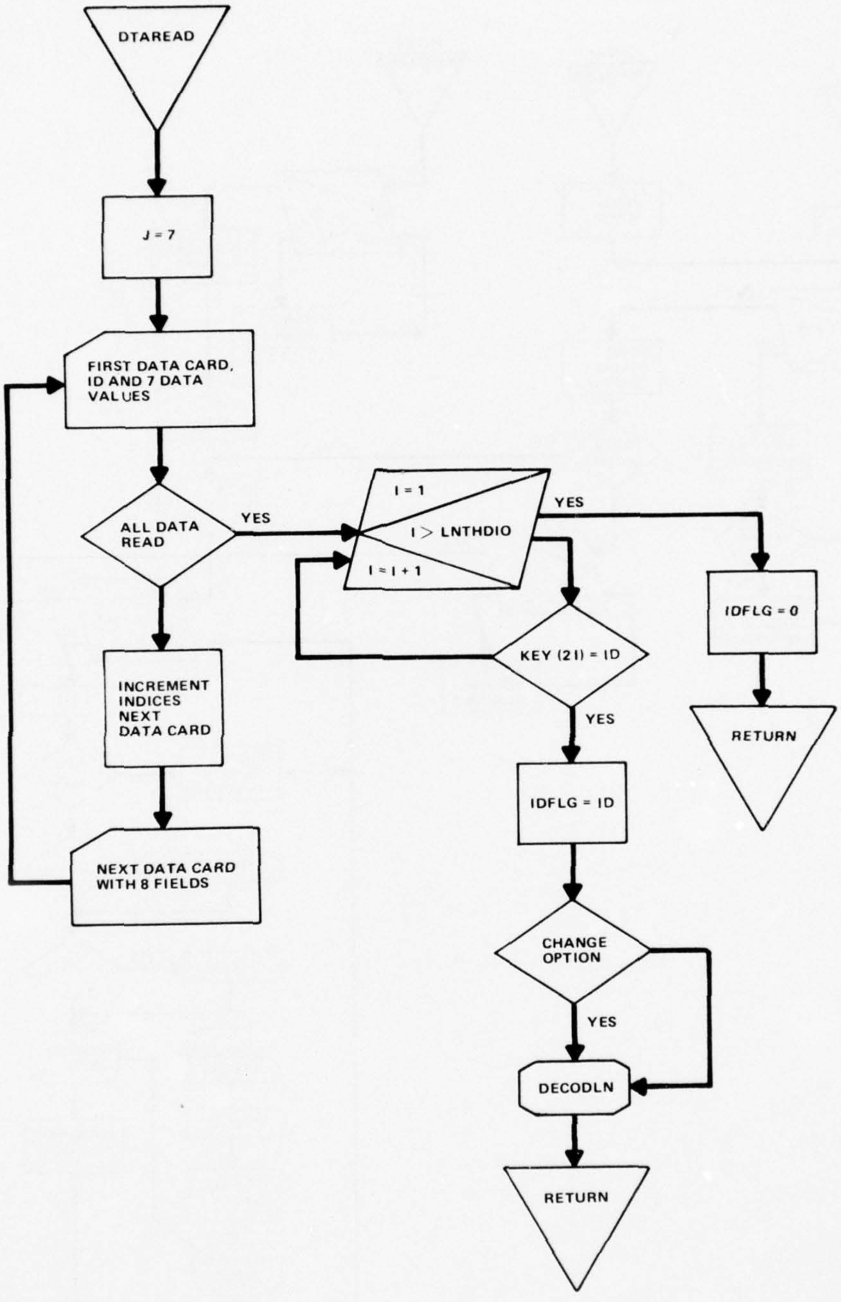

Figure C-5. Subroutine DTAREAD.

equal to N, the maximum number of such fields. The named index elements of the existing data file are then compared with the record ID specified. If a match is found, this record ID is stored in IDFLG; otherwise, a zero is stored in IDFLG.

Then, if the CHANGE option had been selected on the record currently being processed, subroutine DECODLN is called to determine the length of the existing record before returning to the main program.

C-2.6 REFENCD(DTA,IREF), RETURNS(I)

C-2.6.1 Variable Definitions

DTA — Variable into whose three least significant octal digits a coded three octal digit reference is stored

I — Statement number in calling program to which control returns under an abnormal return condition

IREF — Integer variable containing coded three octal digit reference

MASK1 — Logical mask used to put zeros in the three least significant octal digits of DTA before storing a reference there

C-2.6.2 Program Explanation

Subroutine REFENCD (fig. C-6) stores the three least significant octal digits of a specified integer data word into the three least significant octal digits of a real data word. It does this without altering the remaining, more significant portion of this real data word. (See fig. C-6 on p. 88.)

If this integer data word contains a value greather than 777 octal, then a nonstandard return is taken to the calling program.

C-2.7 FINDREF(DTA,IREF)

C-2.7.1 Variable Definitions

DTA — Real data word from which the three least significant octal digits are to be extracted

IREF — Integer variable into which a three octal digit reference is to be stored

MASK1 — Logical mask used to eliminate from consideration all but the three least significant octal digits of a CDC 60-bit word

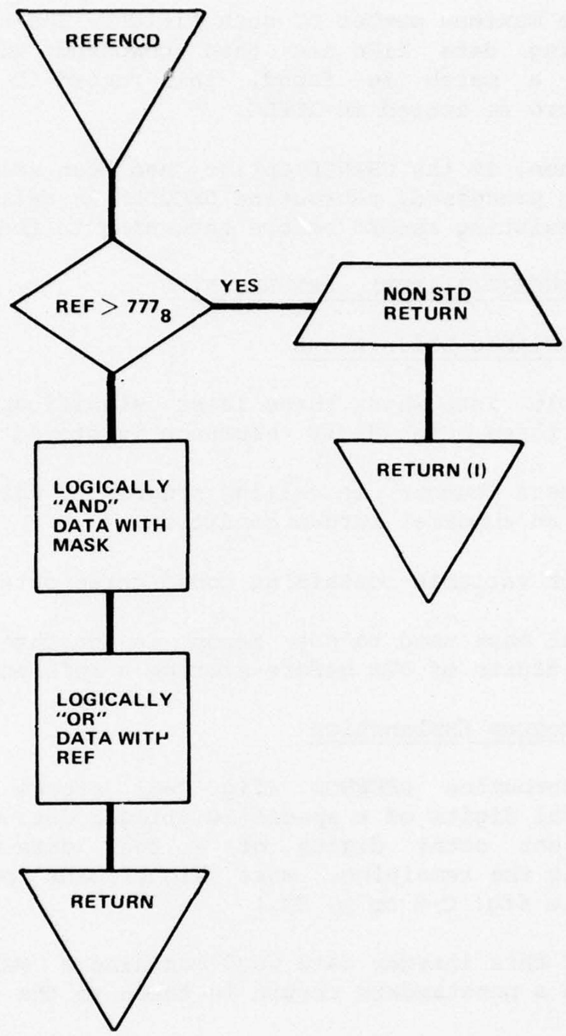

Figure C-6. Subroutine REFENCD.

C-2.7.2 Program Explanation

In subroutine FINDREF (fig. C-7), a mask and a logical AND are used to extract the three least significant octal digits from the real data word DTA. This information is stored into the low order portion of IREF. DTA is unaltered by this process.

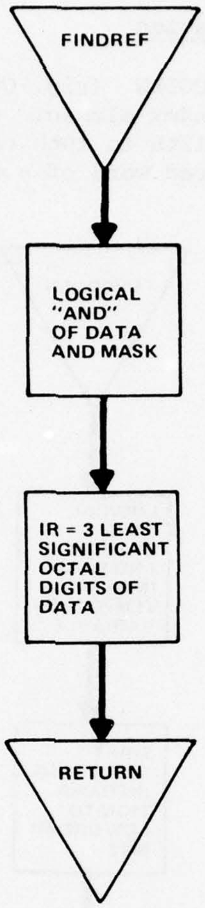

Figure C-7. Subroutine FINDREF.

C-2.8 DECODLN(K,LNTHREC)

C-2.8.1 Variable Definitions

K — Data word containing several pieces of information, one of which is extracted by this subroutine

KK — Variable to temporarily hold information extracted from K

LNTHREC — Record length obtained by operating on the data in KK

MASK — Logical mask used to remove from consideration all the octal digits of K which do not contain the desired information

C-2.8.2 Program Explanation

Subroutine DECODLN (fig. C-8) decodes the record length from the data in a named index element specified by K. The record length is contained in the 12th to 19th octal digits of K (where K is the appropriate odd numbered word of a named index, exclusive of the first word of that index).

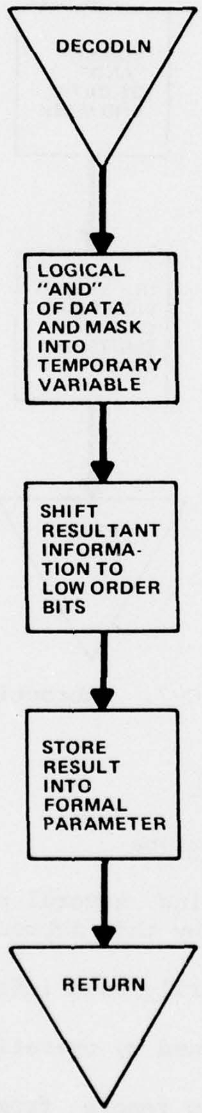

Figure C-8. Subroutine DECODLN.

A logical AND is performed to obtain the data in the 12th to 19th octal digits of the word K. This information is then shifted to the low order octal digits of a data word and stored in LNTHREC. The variable K is unaltered by this procedure. Control is then returned to the calling program.

C-2.9 OCTINTG(IREF,IR)

C-2.9.1 Variable Definitions

IR	Resultant three digit decimal integer variable
IREF	Variable containing three octal digit reference
KK2	Variable used to store second least significant octal digit of IREF in low order digit
KK3	Variable used to store third least significant octal digit of IREF in low order digit
K1	Variable used to store least significant octal digit of IREF
K2	Variable used to store in its second least significant octal digit the data extracted from the second least significant octal digit of IREF
K3	Variable used to store in its third least significant octal digit the data extracted from the third least significant octal digit of IREF
MS1	Logical mask used in extracting least significant octal digit from IREF
MS2	Logical mask used in extracting second least significant octal digit from IREF
MS3	Logical mask used in extracting third least significant octal digit from IREF

C-2.9.2 Subroutine Explanation

Subroutine OCTINTG (fig. C-9) converts an octal integer constant to a decimal integer constant which prints under an I format exactly as the octal number would print under an O format. This is a necessary intermediate step in going from an O format to an A format.

APPENDIX C

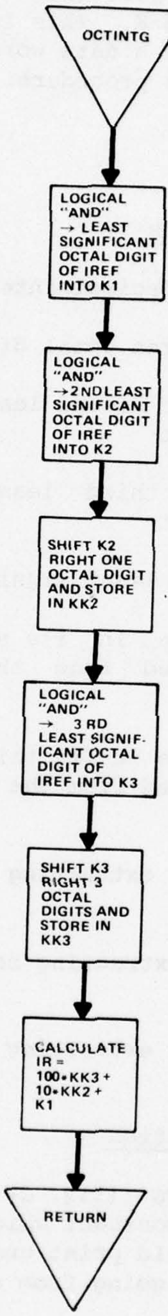

Figure C-9. Subroutine OCTINTG.

Each of the three octal digits of IREF is individually extracted and stored in separate words. The data words containing the extracted second and third digits are each shifted within their respective locations to the low order digit of the same word. These three digits are then multiplied by 1, 10, and 100, respectively, to produce a decimal integer constant which, when printed in I format, is identical to the printing of IREF in octal format. The result is stored in IR, and the subroutine ends, returning to the calling program.

C-2.10 SORTKEY(KEY,LENGTH)

C-2.10.1 Variable Definitions

I	Numerical index incremented for each record of the file to be sorted
II	Temporary lower limit of Do loop for checking the KEY array
J	Index of Do loop for checking the KEY array
KEY	Array containing the NAMED INDICES of the records of an associated disk storage file in the even numbered elements of the array (The odd numbered elements of the array contain information on other characteristics of the records within the associated file.)
K2	Temporary storage for an even element of the KEY array
K3	Temporary storage for an odd element of the KEY array
LENGTH	Number of data records in the disk file
L1	Upper index of Do loop, defined as "LENGTH - 1"

C-2.10.2 Subroutine Explanation

Subroutine SORTKEY (fig. C-10) sorts the elements of the KEY array in pairs, with the sort performed on the even numbered element of the pair, according to the standard CDC FORTRAN collating sequence. The first element of the KEY array is left intact, and the second and third elements constitute the first pair within KEY. The number of pairs, in the KEY array, to be sorted is defined by the parameter LENGTH.

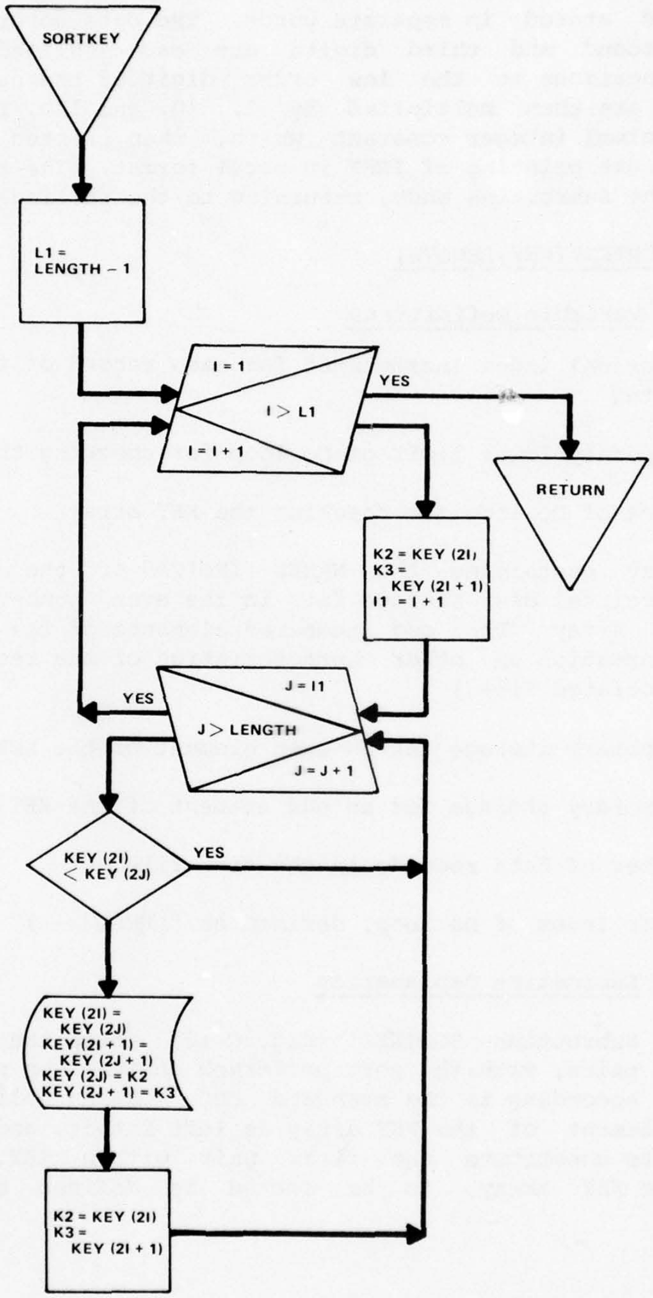

Figure C-10. Subroutine SORTKEY.

C-2.11 PRNTHD

C-2.11.1 Variable Definitions

HEADS — Array containing hollerith data to be used for first line of header in each printed output page

HEADS1 — Array containing hollerith data to be used for second line of header in each printed output page

C-2.11.2 Program Explanation

Subroutine PRNTHD (fig. C-11) is utilized to print out descriptive headers at the top of each computer printout page on which data from the transistor data base will be listed. The array of hollerith fields stored in HEADS is printed as the first line of the label. The array of hollerith fields stored in HEADS1 is then printed on the second line of output. Format 100 is then used to skip a line and print a third header line. Format 101 is then used to print the last line of the header label. The subroutine then returns control to the calling program. (See fig. C-11 on p. 96.)

C-2.12 BLNKFIL

C-2.12.1 Variable Definitions

A — Array which is to be filled with the constant B

B — An octal 40007777777777777000 constant which is a negative indefinite on CDC 6000 series computers

N — Number of elements of the array A which are to be filled with B

C-2.12.2 Program Explanation

Subroutine BLNKFIL (fig. C-12) stores the negative indefinite constant defined as B into the first N locations of the array A. Control is then returned to the calling program.

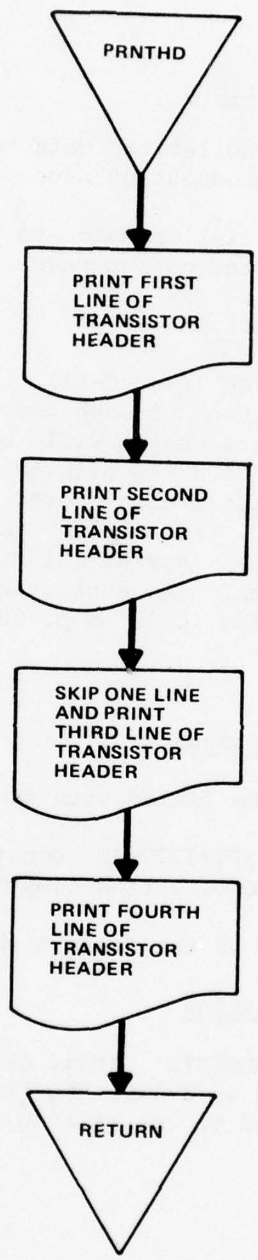

Figure C-11. Subroutine PRNTHD.

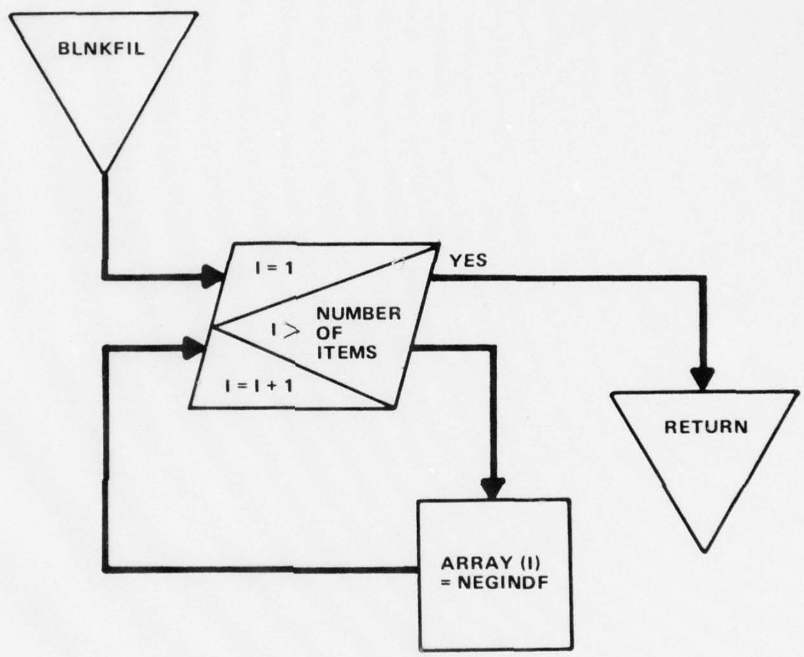

Figure C-12. Subroutine BLNKFIL.

AD-A050 845 HARRY DIAMOND LABS ADELPHI MD F/G 9/2
EXTENDING AND INTERFACING THE MSEP SEMICONDUCTOR DAMAGE DATA BA--ETC(U)
DEC 77 C P RUZIC
UNCLASSIFIED HDL-TR-1821 NL

2 OF 2
AD
A050 845

END
DATE
FILMED
4-78
DDC

DISTRIBUTION

DEFENSE DOCUMENTATION CENTER
CAMERON STATION, BUILDING 5
ALEXANDRIA, VA 22314
ATTN DDC-TCA (12 COPIES)

COMMANDER
US ARMY MATERIEL DEVELOPMENT
& READINESS COMMAND
5001 EISENHOWER AVENUE
ALEXANDRIA, VA 22333
ATTN DRXAM-TL, HQ TECH LIBRARY
ATTN DRCRP/MG C. M. MCKEEN, JR.
ATTN DRCRP-M/COL R. W. SPECKER
ATTN DRCPM-SCM-WF
ATTN DRCDE-D/MR. HUNT
ATTN DRCDE-D/COL J. F. BLEECKER
ATTN DRCDE, DIR FOR DEV & ENGR
ATTN DRCDE-DE/H. DARRACOTT
ATTN DRCMS-I/DR. R. P. UHLIG
ATTN DRCMS-I/MR. E. O'DONNEL
ATTN DRCDMD-ST/N. L. KLEIN

COMMANDER
US ARMY ARMAMENT MATERIEL
READINESS COMMAND
ROCK ISLAND ARSENAL
ROCK ISLAND, IL 61201
ATTN DRSAR-ASF, FUZE & MUNITION
SPT DIV
ATTN DRSAR-PDM/J. A. BRINKMAN
ATTN DRCPM-VFF

COMMANDER
USA MISSILE & MUNITIONS CENTER
& SCHOOL
REDSTONE ARSENAL, AL 35809
ATTN ATSK-CTD-F

DEFENSE ADVANCED RESEARCH
PROJECTS AGENCY
1400 WILSON BLVD
ARLINGTON, VA 22209
ATTN TECH INFORMATION OFFICE
ATTN DIR, STRATEGIC TECHNOLOGY
ATTN DIR, TACTICAL TECHNOLOGY

DIRECTOR
DEFENSE COMMUNICATION ENG CENTER
1860 WIEHLE AVENUE
RESTON, VA 22090
ATTN R104, M. J. RAFFENSPERGER
ATTN R800, R. E. LYONS
ATTN R320, A. IZZO

DIRECTOR
DEFENSE INTELLIGENCE AGENCY
WASHINGTON, DC 20301
ATTN DI-2, WEAPONS & SYSTEMS DIV

DIRECTOR
DEFENSE NUCLEAR AGENCY
WASHINGTON, DC 20305
ATTN PETER HAAS, DEP DIR,
SCIENTIFIC TECHNOLOGY
ATTN RAEV, MAJ S. O. KENNEDY, SR.
ATTN VLIS, LTC ADAMS

DEPARTMENT OF DEFENSE
DIRECTOR OF DEFENSE RESEARCH & ENGINEERING
WASHINGTON, DC 20301
ATTN DEP DIR (TACTICAL WARFARE PROGRAMS)
ATTN DEP DIR (TEST & EVALUATION)
ATTN DEFENSE SCIENCE BOARD
ATTN ASST DIR SALT SUPPORT GP/
MR. J. BLAYLOCK

CHAIRMAN
JOINT CHIEFS OF STAFF
WASHINGTON, DC 20301
ATTN J-3, NUCLEAR WEAPONS BR
ATTN J-3, EXER PLANS & ANALYSIS DIV
ATTN J-5, NUCLEAR DIR NUCLEAR POLICY BR
ATTN J-5, REQUIREMENT & DEV BR
ATTN J-6, COMMUNICATIONS-ELECTRONICS

DEPARTMENT OF DEFENSE
JOINT CHIEFS OF STAFF
STUDIES ANALYSIS & GAMING AGENCY
WASHINGTON, DC 20301
ATTN STRATEGIC FORCES DIV
ATTN GEN PURPOSE FORCES DIV
ATTN TAC NUC BR
ATTN SYS SUPPORT BR

ASSISTANT SECRETARY OF DEFENSE
PROGRAM ANALYSIS AND EVALUATION
WASHINGTON, DC 20301
ATTN DEP ASST SECY (GEN PURPOSE PROG)
ATTN DEP ASST SECY (REGIONAL PROGRAMS)
ATTN DEP ASST SECY (RESOURCE ANALYSIS)

DEPARTMENT OF THE ARMY
OFFICE, SECRETARY OF THE ARMY
WASHINGTON, DC 20301
ATTN ASST SECRETARY OF THE ARMY (I&L)
ATTN DEP FOR MATERIEL ACQUISITION
ATTN ASST SECRETARY OF THE ARMY (R&D)

DEPARTMENT OF THE ARMY
ASSISTANT CHIEF OF STAFF FOR INTELLIGENCE
WASHINGTON, DC 20301
ATTN DAMI-OC/COL J. A. DODDS
ATTN DAMI-TA/COL F. M. GILBERT

US ARMY SECURITY AGENCY
ARLINGTON HALL STATION
4000 ARLINGTON BLVD
ARLINGTON, VA 22212
ATTN DEP CH OF STAFF RESEARCH
& DEVELOPMENT

Preceding Page Blank

DISTRIBUTION (Cont'd)

DEPARTMENT OF THE ARMY
US ARMY CONCEPTS ANALYSIS AGENCY
8120 WOODMONT AVENUE
BETHESDA, MD 20014
ATTN COMPUTER SUPPORT DIV
ATTN WAR GAMING DIRECTORATE
ATTN METHODOLOGY AND RESOURCES DIR
ATTN SYS INTEGRATION ANALYSIS DIR
ATTN JOINT AND STRATEGIC FORCES DIR
ATTN FORCE CONCEPTS AND DESIGN DIR
ATTN OPERATIONAL TEST AND
EVALUATION AGENCY

DIRECTOR
NATIONAL SECURITY AGENCY
FORT GEORGE G. MEADE, MD 20755

COMMANDER-IN-CHIEF
EUROPEAN COMMAND
APO NEW YORK, NY 09128

HEADQUARTERS
US EUROPEAN COMMAND
APO NEW YORK, NY 09055

DIRECTOR
WEAPONS SYSTEMS EVALUATION GROUP
OFFICE, SECRETARY OF DEFENSE
400 ARMY-NAVY DRIVE
WASHINGTON, DC 20305
ATTN DIR, LT GEN GLENN A. KENT

DEPARTMENT OF THE ARMY
DEPUTY CHIEF OF STAFF
FOR OPERATIONS & PLANS
WASHINGTON, DC 20301
ATTN DAMO-RQD/COL E. W. SHARP
ATTN DAMO-SSP/COL D. K. LYON
ATTN DAMO-SSN/LTC R. E. LEARD
ATTN DAMO-SSN/LTC B. C. ROBINSON
ATTN DAMO-RQZ/COL G. A. POLLIN, JR.
ATTN DAMO-TCZ/MG T. M. RIENZI
ATTN DAMO-ZD/A. GOLUB
ATTN DAMO-RQA/COL M. T. SPEIR

DEPARTMENT OF THE ARMY
CHIEF OF RESEARCH, DEVELOPMENT
AND ACQUISITION OFFICE
WASHINGTON, DC 20301
ATTN DAMA-RAZ-A/R. J. TRAINOR
ATTN DAMA-CSM-N/LTC OGDEN
ATTN DAMA-WSA/COL W. E. CROUCH, JR.
ATTN DAMA-WSW/COL L. R. BAUMANN
ATTN DAMA-CSC/COL H. C. JELINEK
ATTN DAMA-CSM/COL H. R. BAILEY
ATTN DAMA-WSZ-A/MG D. R. KEITH
ATTN DAMA-WSM/COL J. B. OBLINGER, JR.
ATTN DAMA-PPR/COL D. E. KENNEY

COMMANDER
BALLISTIC MISSILE DEFENSE SYSTEMS
P.O. BOX 1500
HUNTSVILLE, AL 35807
ATTN BMDSC-TEN/MR. JOHN VEFNEMAN

COMMANDER
US ARMY FOREIGN SCIENCE
AND TECHNOLOGY CENTER
220 SEVENTH ST., NE
CHARLOTTESVILLE, VA 22901

DIRECTOR
US ARMY MATERIEL SYSTEMS ANALYSES ACTIVITY
ABERDEEN PROVING GROUND, MD 21005
ATTN DRXSY-C/DON R. BARTHEL
ATTN DPXSY-T/P. REID

COMMANDER
US ARMY SATELLITE COMMUNICATIONS AGENCY
FT. MONMOUTH, NJ 07703
ATTN LTC HOSMER

DIRECTOR
BALLISTIC RESEARCH LABORATORIES
ABERDEEN PROVING GROUND, MD 21005
ATTN DRXBR-XA/MR. J. MESZAROS

COMMANDER
US ARMY AVIATION SYSTEMS COMMAND
12TH AND SPRUCE STREETS
ST. LOUIS, MO 63160
ATTN DRCPM-AAH/ROBERT HUBBARD

DIRECTOR
EUSTIS DIRECTORATE
US ARMY AIR MOBILITY R&D LABORATORY
FORT EUSTIS, VA 23604
ATTN SAVDL-EU-MOS/MR. S. POCILUYKO
ATTN SAVDL-EU-TAS(TETRACORE)

COMMANDER
2D BDE, 101ST ABN DIV (AASLT)
FORT CAMPBELL, KY 42223
ATTN AFZB-KB-SO
ATTN DIV SIGNAL OFFICER,
AFBZ-SO/MAJ MASON

COMMANDER
US ARMY ELECTRONICS COMMAND
FT. MONMOUTH, NJ 07703
ATTN PM, ATACS/DRCPM-ATC/LTC DOBBINS
ATTN DRCPM-ATC-TM
ATTN PM, ARTADS/DRCPM-TDS/BG A. CRAWFORD
ATTN DRCPM-TDS-TF/COL D. EMERSON
ATTN DRCPM-TDS-TO
ATTN DRCPM-TDS-FB/LTC A. KIRKPATRICK
ATTN PM, MALOR/DRCPM-MALR/COL W. HARRISON

US ARMY ELECTRONICS COMMAND (Cont'd)
ATTN PM, NAVCOM/DRCPM-NC/COL C. MCDOWELL, JR.
ATTN PM, REMBASS/DRCPM-RBS/COL R. COTTEY, SR.
ATTN DRSEL-TL-IR/MR. R. FREIBERG
ATTN DRSEL-SA/NORMAN MILLSTEIN
ATTN DRSEL-MA-C/J. REAVIS
ATTN DRSEL-CE-ES/J. A. ALLEN

COMMANDER
US ARMY MISSILE MATERIEL READINESS COMMAND
REDSTONE ARSENAL, AL 35809
ATTN DRSMI-FRR/DR. F. GIPSON
ATTN DRCPM-HA/COL P. RODDY
ATTN DRCPM-LCCX/L. B. SEGGEL (LANCE)
ATTN DRCPM-MD/GENE ASHLEY (PATRIOT)
ATTN DRCPM-MP
ATTN DRCPM-PE/COL SKEMP (PERSHING)
ATTN DRCPM-SHO
ATTN DRCPM-TO
ATTN DRSMI-R, RDE & MSL DIRECTORATE

COMMANDER
US ARMY ARMAMENT RESEARCH & DEVELOPMENT COMMAND
DOVER, NJ 07801
ATTN DRDAR-ND-V/DANIEL WAXLER

COMMANDER
US ARMY TANK/AUTOMOTIVE MATERIEL READINESS COMMAND
WARREN, MI 48090
ATTN DRSI-RHT/MR. P. HASEK
ATTN DRCPM(XM-L)/MR. L. WOOLCOT
ATTN DRCPM-GCM-SW/MR. R. SLAUGHTER

PRESIDENT
DA, HA, US ARMY ARMOR AND ENGINEER BOARD
FORT KNOX, KY 40121
ATTN STEBB-MO/MAJ SANZOTERRA

COMMANDER
WHITE SANDS MISSILE RANGE
WHITE SANDS MISSILE RANGE, NM 88002
ATTN STEWS-TE-NT/MARVIN SQUIRES

COMMANDER
TRASANA
SYSTEM ANALYSIS ACTIVITY
WHITE SANDS, NM 88002
ATTN ATAA-TDO/DR. D. COLLIER

COMMANDER
197TH INFANTRY BRIGADE
FORT BENNING, GA 31905
ATTN COL WASIAK

COMMANDER
US ARMY COMMUNICATIONS COMMAND
FORT HUACHUCA, AZ 85613
ATTN ACC-AD-C/H. LASITTER (EMP STUDY GP)

COMMANDER
USA COMBINED ARMS COMBAT DEVELOPMENTS ACTIVITY
FT. LEAVENWORTH, KS 66027
ATTN ATCAC
ATTN ATCACO-SD/LTC L. PACHA
ATTN ATCA/COC/COL HUBBERT
ATTN ATCA-CCM-F/LTC BECKER
ATTN ATSW-TD-3 NUCLEAR STUDY TEAM/LT D. WILKINS

PROJECT MANAGER
MOBILE ELECTRIC POWER
7500 BACKLICK ROAD
SPRINGFIELD, VA 22150
ATTN DRCPM-MEP

DEPUTY COMMANDER
US ARMY NUCLEAR AGENCY
7500 BACKLICK RD
BUILDING 2073
SPRINGFIELD, VA 22150
ATTN MONA-WE/COL A. DEVERILL

OMMANDER
S ARMY SIGNAL SCHOOL
T. GORDON, GA 30905
ATTN AISO-CID/BILL MANNELL
ATTN ATST-CTD-CS/ CAPT G. ALEXANDER (INTACS)
ATTN ATSO-CID-CS/MR. TAYLOR
ATTN ATSN-CD-OR/MAJ CARR

DIRECTOR
JOINT TACTICAL COMMUNICATIONS OFFICE
FT. MONMOUTH, NJ 07703
ATTN TRI-TAC/NORM BECHTOLD

COMMANDER
US ARMY COMMAND AND GENERAL STAFF COLLEGE
FORT LEAVENWORTH, KS 66027

COMMANDER
US ARMY COMBAT DEVELOPMENTS EXPERIMENTATION COMMAND
FORT ORD, CA 93941

COMMANDER
HQ MASSTER
FORT HOOD, TX 76544

COMMANDER
US ARMY AIR DEFENSE SCHOOL
FORT BLISS, TX 79916
ATTN ATSA-CD

DISTRIBUTION (Cont'd)

COMMANDER
US ARMY ARMOR SCHOOL
FORT KNOX, KY 40121
ATTN ATSB-CTD (2 COPIES)

COMMANDER
US ARMY AVIATION CENTER
FORT RUCKER, AL 36360
ATTN ATST-D-MS (2 COPIES)

COMMANDER
US ARMY ORDNANCE CENTER AND SCHOOL
ABERDEEN PROVING GROUND, MD 21005
ATTN USAOC&S
ATTN ATSL-CTD

COMMANDER
US ARMY SIGNAL SCHOOL
FORT GORDON, GA 30905
ATTN ATSS-CTD (2 COPIES)

COMMANDER
US ARMY ENGINEER SCHOOL
FORT BELVOIR, VA 22060
ATTN ATSE-CTD (2 COPIES)

COMMANDER
US ARMY INFANTRY SCHOOL
FORT BENNING, GA 31905
ATTN ATSH-CTD (2 COPIES)

COMMANDER
US ARMY INTELLIGENCE CENTER
AND SCHOOL
FORT HUACHUCA, AZ 85613
ATTN ATSI-CTD (2 COPIES)

COMMANDER
US ARMY FIELD ARTILLERY SCHOOL
FORT SILL, OK 73503
ATTN ATSF-CTD (2 COPIES)

CHIEF OF NAVAL OPERATIONS
NAVY DEPARTMENT
WASHINGTON, DC 20350
ATTN NOP-932, SYS EFFECTIVENESS DIV
CAPT F. V. LANEY
ATTN NOP-9860, COMMUNICATIONS BR
COR L. LAYMAN
ATTN NOP-351, SURFACE WEAPONS BR
CAPT G. A. MITCHELL
ATTN NOP-622C, ASST FOR NUCLEAR
VULNERABILITY, R. PIACESI

COMMANDER
NAVAL ELECTRONICS SYSTEMS COMMAND, HQ
2511 JEFFERSON DAVIS HIGHWAY
ARLINGTON, VA 20360
ATTN PME-117-21, SANGUINE DIV

HEADQUARTERS, NAVAL MATERIEL COMMAND
STRATEGIC SYSTEMS PROJECTS OFFICE
1931 JEFFERSON DAVIS HIGHWAY
ARLINGTON, VA 20390
ATTN NSP-2201, LAUNCHING & HANDLING
BRANCH, BR ENGINEER, P. R. FAUROT
ATTN NSP-230, FIRE CONTROL & GUIDANCE
BRANCH, BR ENGINEER, D. GOLD
ATTN NSP-2701, MISSILE BRANCH,
BR ENGINEER, J. W. PITSENBERGER

COMMANDER
NAVAL SURFACE WEAPONS CENTER
WHITE OAK, MD 20910
ATTN CODE 222, ELECTRONICS & ELECTRO-
MAGNETICS DIV
ATTN CODE 431, ADVANCED ENGR DIV

US AIR FORCE, HEADQUARTERS
DCS, RESEARCH & DEVELOPMENT
WASHINGTON, DC 20330
ATTN DIR OF OPERATIONAL REQUIREMENTS
AND DEVELOPMENT PLANS, S/V &
LTC P. T. DUESBERRY

COMMANDER
AF WEAPONS LABORATORY, AFSC
KIRTLAND AFB, NM 87117
ATTN ES, ELECTRONICS DIVISION
ATTN EL, J. DARRAH
ATTN TECHNICAL LIBARY
ATTN D. I. LAWRY

COMMANDER
AERONAUTICAL SYSTEMS DIVISION, AFSC
WRIGHT-PATTERSON AFB, OH 45433
ATTN ASD/YH, DEPUTY FOR B-1

COMMANDER
HQ SPACE AND MISSILE SYSTEMS ORGANIZATION
P.O. 96960 WORLDWAYS POSTAL CENTER
LOS ANGELES, CA 90009
ATTN S7H, DEFENSE SYSTEMS APL SPO
ATTN XRT, STRATEGIC SYSTEMS DIV
ATTN SYS, SURVIVABILITY OFC

SPACE AND MISSILE SYSTEMS ORGANIZATION
NORTON AFB, CA 92409
ATTN MMH, HARD ROCK SILO DEVELOPMENT

COMMANDER
AF SPECIAL WEAPONS CENTER, AFSC
KIRTLAND AFB, NM 87117

ASSISTANT CHIEF OF STAFF FOR COMMUNICATIONS
ELECTRONICS
XVIII AIRBORNE CORPS
FORT BRAGG, NC 28307
ATTN AFZA-CE/LTC K. KILLINGSTEAD

DISTRIBUTION (Cont'd)

HARRY DIAMOND LABORATORIES
ATTN RAMSDEN, JOHN J., COL, COMMANDER/
FLYER, I.N./LANDIS, P.E./
SOMMER, H./OSWALD, R. B.
ATTN CARTER, W.W., DR., TECHNICAL
DIRECTOR/MARCUS, S.M.
ATTN WISEMAN, ROBERT S., DR., DRDEL-CT
ATTN KIMMEL, S., PAO
ATTN CHIEF, 0021
ATTN CHIEF, 0022
ATTN CHIEF, LAB 100
ATTN CHIEF, LAB 200
ATTN CHIEF, LAB 300
ATTN CHIEF, LAB 400
ATTN CHIEF, LAB 500
ATTN CHIEF, LAB 600
ATTN CHIEF, DIV 700
ATTN CHIEF, DIV 800
ATTN CHIEF, LAB 900
ATTN CHIEF, LAB 1000
ATTN RECORD COPY, BR 041
ATTN HDL LIBRARY (5 COPIES)
ATTN CHAIRMAN, EDITORIAL COMMITTEE
ATTN CHIEF, 047
ATTN TECH REPORTS, 013
ATTN PATENT LAW BRANCH, 071
ATTN GIDEP OFFICE, 741
ATTN LANHAM, C., 0021
ATTN CHIEF, 0024
ATTN CHIEF, 1010
ATTN CHIEF, 1020 (10 COPIES)
ATTN CHIEF, 1030
ATTN CHIEF, 1040
ATTN CHIEF, 1050
ATTN RUZIC, C. P., (10 COPIES)