AD-A067 217 COMPUTER SCIENCES CORP FALLS CHURCH VA F/G 17/2
INVESTIGATION OF THE VULNERABILITY/SURVIVABILITY OF SYSTEMS SUP--ETC(U)
JUN 78 H BLANK, G KINAL, M BASSMAN, D GAN DNA001-77-C-0015
UNCLASSIFIED DNA-4354F-1C-1 NL

1 OF 4
AD A067217

DNA 4354F-1C-1

INVESTIGATION OF THE VULNERABILITY/SURVIVABILITY OF SYSTEMS SUPPORTING THE NCA DECISION PROCESS

Task 7 Computer Program Documentation

Computer Sciences Corporation
6565 Arlington Boulevard
Falls Church, Virginia 22046

June 1978

Final Report for Period February 1977—October 1977

CONTRACT No. DNA 001-77-C-0015

THIS WORK SPONSORED BY THE DEFENSE NUCLEAR AGENCY
UNDER RDT&E RMSS CODE B325077464 V99QAXNA01106 H2590D.

Prepared for
Director
DEFENSE NUCLEAR AGENCY
Washington, D. C. 20305

Destroy this report when it is no longer needed. Do not return to sender.

PLEASE NOTIFY THE DEFENSE NUCLEAR AGENCY, ATTN: TISI, WASHINGTON, D.C. 20305, IF YOUR ADDRESS IS INCORRECT, IF YOU WISH TO BE DELETED FROM THE DISTRIBUTION LIST, OR IF THE ADDRESSEE IS NO LONGER EMPLOYED BY YOUR ORGANIZATION.

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DDC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER DNA 4354F-1C-1	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) INVESTIGATION OF THE VULNERABILITY/SURVIVABILITY OF SYSTEMS SUPPORTING THE NCA DECISION PROCESS Task 7 Computer Program Documentation		5. TYPE OF REPORT & PERIOD COVERED Final Report for Period February 1977—October 1977 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) H. Blank M. Bassman G. Kinal D. Gan		8. CONTRACT OR GRANT NUMBER(s) DNA 001-77-C-0015
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Sciences Corporation 6565 Arlington Boulevard Falls Church, Virginia 22046		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Subtask V99QAXNA011-06
11. CONTROLLING OFFICE NAME AND ADDRESS Director Defense Nuclear Agency Washington, D.C. 20305		12. REPORT DATE June 1978 13. NUMBER OF PAGES 208
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

This work sponsored by the Defense Nuclear Agency under RDT&E RMSS Code B325077464 V99QAXNA01106 H2590D.

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

Data Base	TOTAL R	Input
Primitive Matrix	Files	Output
Utility Program	Master Files	Parameter
Flow Chart	Variable Files	

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

This report contains the Computer Program Documentation for the Integrated Nuclear Communications Assessment (INCA) Transatlantic Trunk Utilization Study. It explains how the data base was created and defines each of the master files and variable files created. The individual programs utilized are explained in detail.

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

SECTION 1 - INTRODUCTION

1.1 BACKGROUND

Computer Sciences Corporation (CSC) and a number of agencies and other companies are engaged in a multiyear effort entitled the Integrated Nuclear Communications Assessment (INCA) Program. At a review briefing of this program, a question related to the general INCA problem was posed which required a rapid answer.

The question was whether the approximately equal apportionment of transoceanic trunking for U.S. command and control circuits among submarine cable, commercial satellites, and military satellites is the best apportionment from the standpoint of survivability. In short, what is the "best" transoceanic trunking medium "mix"? It was deemed of sufficient urgency to merit a separate concentrated effort.

Integrated Nuclear Communications Assessment (INCA), Transatlantic Trunk Utilization, Final Report (Top Secret), Computer Sciences Corporation (CSC), June 1978 is the result of this effort.

SECTION 2 - DATA BASE STRUCTURE AND ANALYSIS PROGRAMS

2.1 INTRODUCTION

In this section, the various software components which were used in the study analysis are described. Basically, there is a data base, or assets file, and the associated utility programs to expedite modification of the data. There are also network analysis programs which examine the networks described by the given file (which may represent a degraded situation) and provide connectivity and routing information.

2.2 TOTAL® DATA BASE

2.2.1 Data Base

A data base describing transatlantic communications was created to aid in the performance of the task analyses. This data base is on the IBM 370/155 computer at DCEC in Reston, Virginia, and is maintained by the TOTAL® data base management system. In accordance with DCA data base naming conventions, it is called RFBTA 1 (Reston Batch Transatlantic base number 1). The actual accurate data base exists only on the classified system. An unclassified facsimile exists on the unclassified system to aid in program development.

This data base was designed as a model of the communications network. The network consists of four types of components: circuits, trunks, links, and sites. Four master files have been created to contain the elements of these components. Two variable-entry files provide cross-reference among the elements of the master files. This enables analysis programs to determine such things as the trunks that a circuit traverses, the trunks which traverse a given link, or the links which emanate from a given site. The relationship among the files are shown in Figure 2-1.

A description of each file is given below. The DCA conventions for naming circuits, trunks, links, and sites are given in DCA circular 310-65-1.

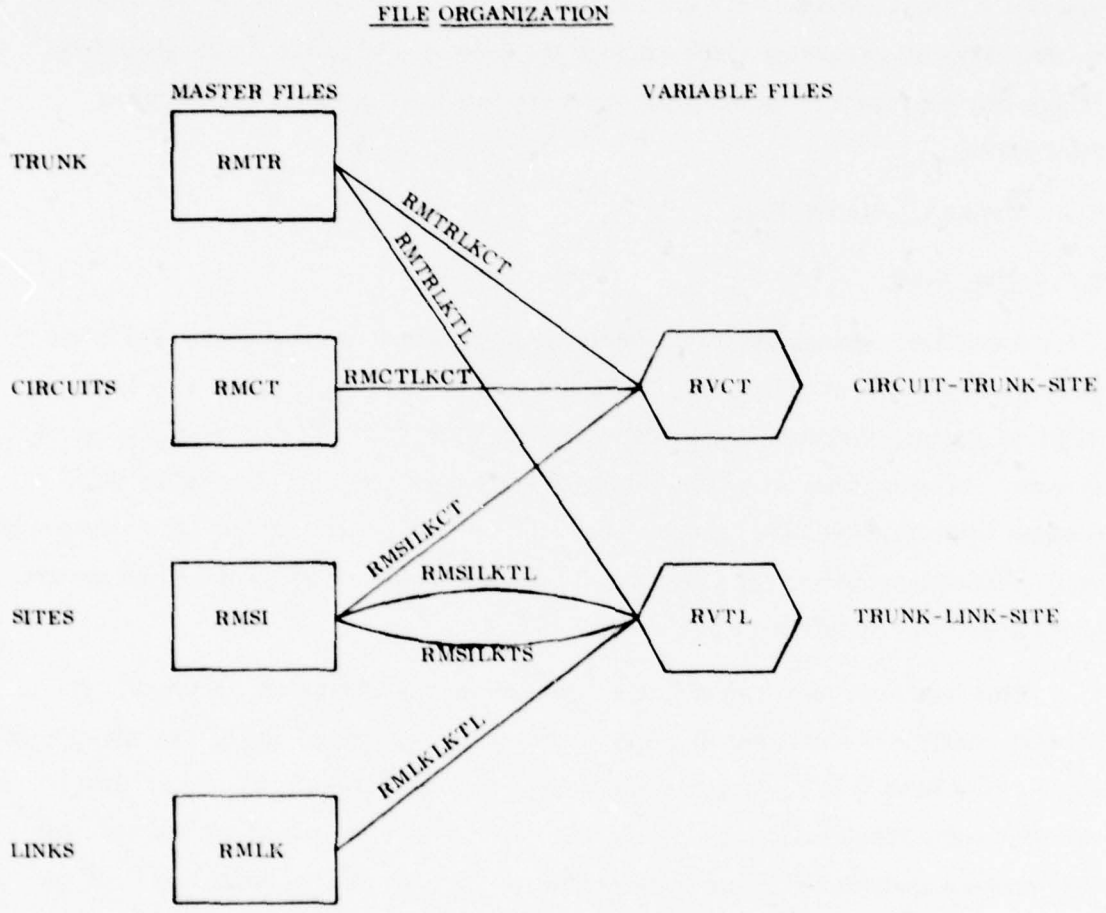


Figure 2-1. File Relationships

2.2.2 Circuit Data File (RMCT)

Records in this file describe individual circuits in the communications network. Each record contains the Command Communications Service Designator (CCSD), a status flag, restoration priority, type of routing, circuit multiplexing indicator, originating site and facility, and terminating site and facility.

These records contain a linkage to the circuit-trunk-site variable file (RVCT), which indicates all trunks the circuit traverses, and the end sites of these trunks. Users can obtain the more detailed data about all links and sites traversed by following the RMTR linkages to the RVTL file, if needed.

2.2.3 Trunk Data File (RMTR)

Records in this file describe individual trunks. Each record contains the DCA trunk identifier, a status flag, restoration category, capacity, bandwidth, number of channels available, originating site and facility, and terminating site and facility.

These records also contain linkages to the circuit-trunk-site variable file (RVCT) and the trunk-link-site variable file (RVTL). This facilitates retrieval of all circuits using a given trunk, and all links and sites which a trunk traverses.

2.2.4 Link Data File (RMLK)

These records describe each link in the communication network. Each record contains a DCA link identifier, status flag, transmission medium code, and site and facility designation for the link end points.

These records are linked to the trunk-link-site variable file (RVTL). This enables users to determine all trunks which use a given link.

2.2.5 Site Data File (RMSI)

These records describe each node in the network. Each record contains a DCA site abbreviation, a status flag, state or country of location, and coordinates of the site.

These records are linked to the RVTL and to the RVTC. There are two linkages to RVTL, so a trunk may be traced either forward or backward.

2.2.6 Circuit - Site - Trunk File (RVCT)

These records cross-reference the circuit, site, and trunk files. Each record contains a circuit identifier (CCSD), a trunk identifier, the abbreviation of the site at which the trunk originates, and the site at which the trunk terminates. Thus, there is a separate record on RVCT for each trunk which a circuit traverses. Each record is linked to the corresponding records in the three master files by a chain of RVCT records. Thus, each RMCT record is linked to a chain of RVCT records which describe all trunks and sites the circuit traverses. Each RMTR record is linked to a chain of RVCT records which describes which circuits traverse that trunk, and each site is linked to a chain which describes which circuits and trunks emanate from that site.

2.2.7 Trunk - Site - Link File (RVTL)

RVTL records cross-reference the trunk, site, and link files. Each record contains a trunk identifier, a link identifier, the location at which the link originates, and the location at which the link terminates. Each trunk is linked to a chain giving all the sites and links which the trunk traverses. Each RMLK record is linked to a chain listing all trunks which use that link. Each site is linked to two chains, one giving all links which originate at that site, and another giving all links which terminate at that site.

Implicit in the foregoing is the fact that there is a two-level hierarchical description of all circuits. Each circuit is described in terms of the trunks it traverses. Each trunk is described in terms of the link it traverses. However, circuits are not directly described in terms of links traversed.

SECTION 3 - ANALYSIS PROGRAMS

3.1 EMP INTERPOLATION AND LOOKUP PROGRAM

This paragraph contains a description of the EMP interpolation and lookup program.

Paragraph 3.1.1 provides a summary of the main FORTRAN program, including requirements, constraints, and general design.

Paragraph 3.1.2 provides a description of the two FORTRAN subroutines used by the main program.

3.1.1 Program Description

3.1.1.1 Requirements

Given a list of nuclear events and a list of targets, this program will calculate various EMP field strength parameters for each target-event combination.

3.1.1.2 Constraints

1. Height of burst may only be 50, 100, 250, or 400 kilometers.
2. Yield may only be 1.25 or 5.00 megatons.
3. Latitude of each event should be between 30 and 60 degrees. Values outside this range may be used, but the results will be distorted.
4. Program dimension statements are set to handle a maximum of 25 events and a maximum of 200 targets.

3.1.1.3 Inputs

There are two data sets read by the main routine: the event data and the target data.

3.1.1.3.1 Event Data Input

The event data set consists of one record containing the number of events, followed by a record for each event.

For the first record, the number appears in positions 1 through 3. Figure 3-1 shows the format for each event record.

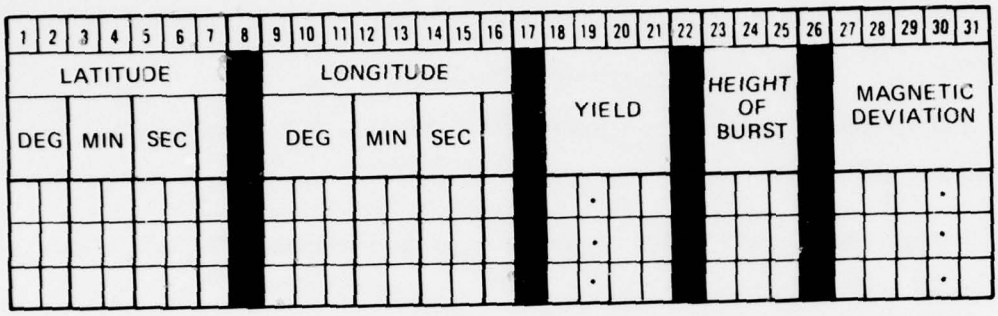


Figure 3-1. Event Data Input Format

Magnetic deviation is the angle between true north and magnetic north in degrees at the event location. (Minus if magnetic north is east of true north.)

3.1.1.3.2 Target Data Input

The target data set consists of a record containing the number of targets, followed by a record for each target.

For the first record, the number appears in positions 1 through 3. Figure 3-2 shows the format for each target record.

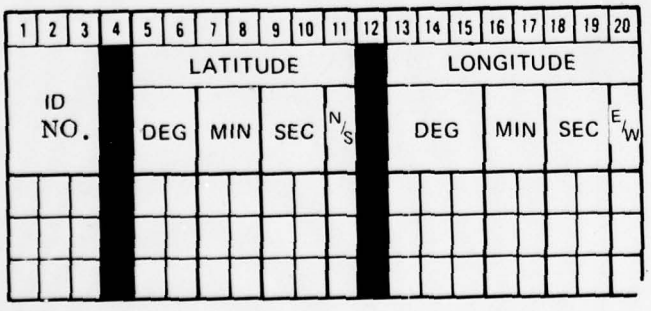


Figure 3-2. Target Data Input Format

ID No. is a three-digit identification number for each target.

Latitude is entered in degrees, minutes, and seconds, followed by a letter N or S denoting north or south.

Longitude is entered in degrees, minutes, and seconds, followed by a letter E or W denoting east or west.

In the batch version of this program, event and target data are entered on cards and read from FORTRAN unit 5.

In the time share operation (TSO) version, event and target data are first entered into separate data files and then allocated to FORTRAN units 11 and 10, respectively.

3.1.1.4 Output

Output from the program consists of a report written to FORTRAN unit 6. In the batch version of the program, this goes to the line printer. In the TSO version, this goes to the terminal used. Figure 3-3 shows a sample output report.

TG	EV	YIELD	HEIGHT	EVENT LAT.	EVENT LONG.	TARGET LAT.	TARGET LONG.	TARGET DIST.	TARGET ANG(T)	TARGET ANG(M)	TOTAL F.S.
1	3	1.25	400.	45.00	2.00	50.39	-7.66	914.71	45.25	52.25	16.28
2	3	1.25	400.	45.00	2.00	52.37	0.22	830.79	8.39	15.39	16.53
3	1	1.25	400.	44.00	61.00	44.65	70.78	781.68	8.71	31.71	17.43
3	2	1.25	400.	41.00	70.00	44.65	70.78	411.15	81.32	96.32	20.13
4	1	1.25	400.	44.00	61.00	38.81	76.87	1441.63	251.85	274.85	9.58
4	2	1.25	400.	41.00	70.00	38.81	76.87	634.56	249.66	264.66	22.12
5	1	1.25	400.	44.00	61.00	38.98	76.50	1414.52	251.92	274.92	9.82
5	2	1.25	400.	41.00	70.00	38.98	76.50	597.66	250.17	265.17	22.88

Figure 3-3. Sample Output

EV is the event sequence number.

TG is the target identification number.

Target angle (T) is the bearing angle from event to target clockwise with respect to true north.

Target angle (M) is the bearing angle from event to target clockwise with respect to magnetic north.

Total F.S. is the total EMP field strength in kilovolts per meter.

3.1.1.5 Information Processing

The program can be divided into four major sections:

1. Reading the event data
2. Reading the target data
3. Computing the EMP effects
4. Printing the results.

3.1.1.5.1 Reading the Event Data

Coordinates given in degrees, minutes, seconds, and E/W or N/S are converted to positive or negative degrees and fractional degrees. These are converted to radians as required by FORTRAN trigonometric functions.

An angular range to the horizon from each event is computed to determine the maximum range of effect. (See Figure 3-4.)

Finally, the deviation angle from true north to magnetic north is calculated.

3.1.1.5.2 Reading the Target Data

Coordinates given in positive or negative degrees and fraction of degrees are converted to radians and stored.

3.1.1.5.3 Computing the EMP Effects

Major computations in this appendix pertain to determining the range and bearing from each event to target. If the target is over the horizon from an event, no effect is assumed. (See Figure 3-5 for an explanation of range and bearing calculations.

The great circle distance P P_0 is given by considering the spherical triangle made by P, P_0, and the pole N.

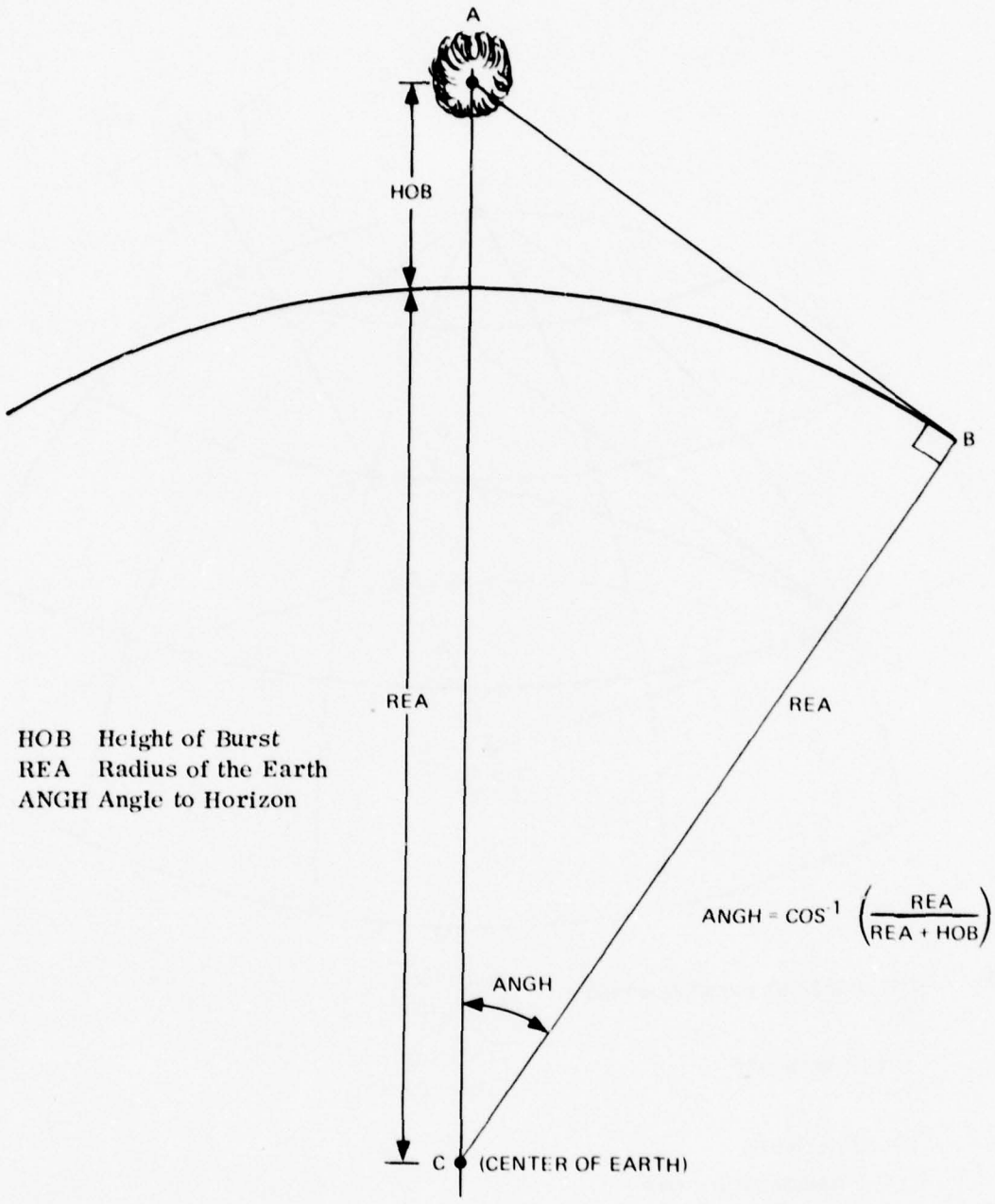
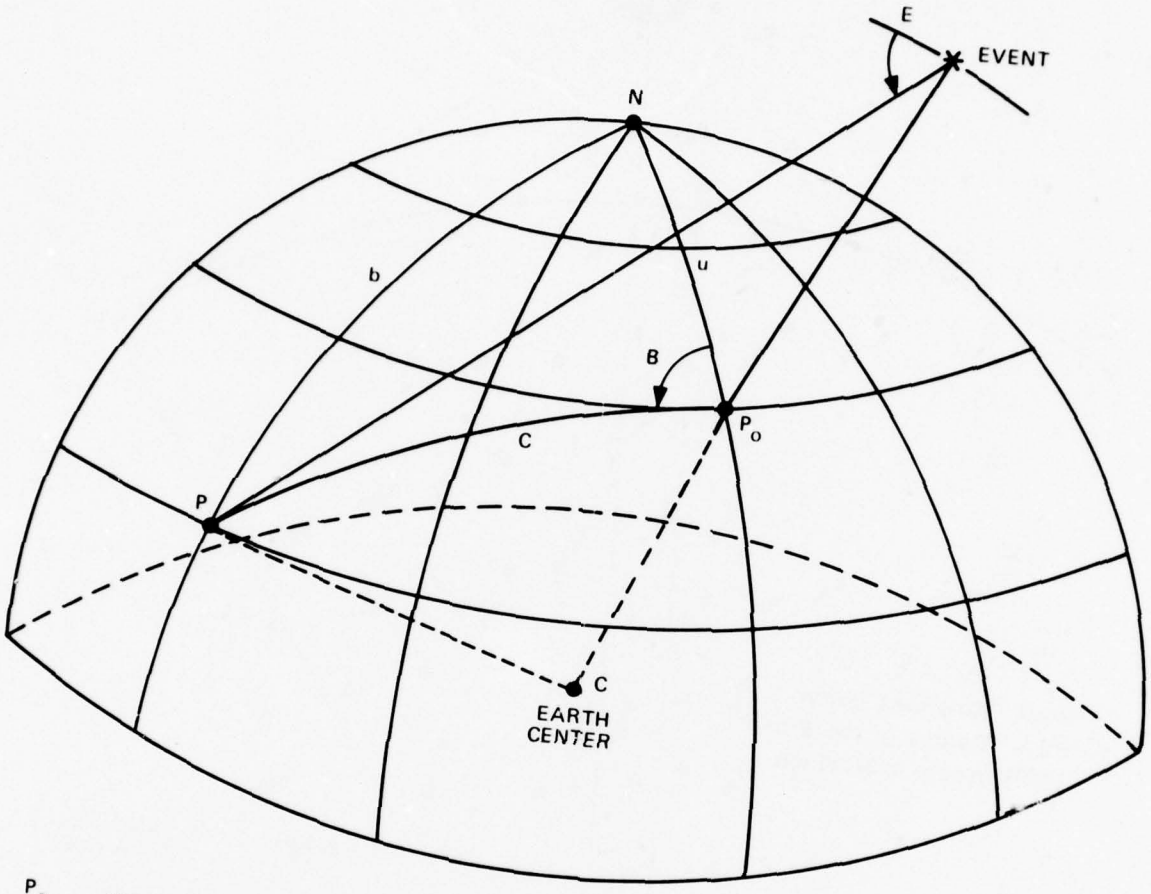


Figure 3-4. Angular Range Computation



P_o = GROUND ZERO, EVENT SUBPOINT { LATITUDE lo, LONGITUDE mo

P = SITE OF INTEREST { LATITUDE l, LONGITUDE m

h = EVENT ALTITUDE

s = SLANT RANGE FROM E TO P

Figure 3-5. Range and Bearing Calculations

Let the radius of the earth = 1.

Then the length PN along the meridian is

$$PN = \frac{\pi}{2}\left(\frac{90 - \ell}{90}\right) = b$$

and

$$P_oN = \frac{\pi}{2}\left(\frac{90 - \ell_o}{90}\right) = a$$

and the angle

$$P_oN\,P = m_o - m$$

Then the length PoP along the great circle is given by c, where

$$\cos c = \cos b \cos a + \sin b \sin a \cos (m_o - m) \qquad (3\text{-}1)$$

and the angle subtended at the earth's center by P_o and P is c radians.

The great circle through P and P_o cuts the earth into two hemispheres. EP_o is perpendicular to PP_o and is in the plane formed by the great circle. The azimuthal bearing of the line of sight EP to the geographic north is the angle the great circle makes to the geographic north at P_o. If this bearing angle is B

$$\frac{\sin B}{\sin b} = \frac{\sin C}{\sin c} \qquad (3\text{-}2)$$

where

$$C = m_o - m$$

and c was given previously.

$$\sin B = \frac{\sin (m_o - m)\ \sin \frac{\pi}{2}\left(\frac{90 - \ell}{90}\right)}{\sin c} \qquad (3\text{-}3)$$

Thus, the procedure is to find c using Equation 3-1; and then use the value of c to find B, the azimuthal bearing angle from Equation 3-3. The range is simple angle C in radians times the radius of the earth.

Next, the bearing angle from true north is calculated by determining the quadrant of the bearing angle. (See Figure 3-6.)

Quadrant	Correction
I	ANGR = ANGR
II	ANGR = 2π - ANGR
III	ANGR = π + ANGR
IV	ANGR = π - ANGR

(II)	(I)
TLON > ELON	TLON ≤ ELON
TLAT ≥ ELAT	TLAT > ELAT
(III)	**(IV)**
TLON > ELON	TLON ≤ ELON
TLAT < ELAT	TLAT ≤ ELAT

ELAT = Event latitude
ELON = Event longitude
TLAT = Target Latitude
TLON = Target Longitude

Figure 3-6. Defining Bearing Angle Quadrant

The bearing angle THETA is corrected for measurements relative to 10° E and 45° N. Subroutine INTERP is called to retrieve and interpolate the EMP values.

3.1.1.5.4 Printing the Results

Figure 3-3 shows a sample printout. A page restore and headings are printed every 35 lines.

The only EMP value printed is total field strength. The other values are available in array V if they should be required in the future.

3.1.1.6 Data Organization

Variable Name	Usage
A	Angular distance from event to North Pole
AD	Portion of event latitude in degrees
AM	Portion of event latitude in minutes
ANG	Bearing angle in degrees
ANG1	Bearing angle in degrees to magnetic north
ANGH (25)	Angle to horizon from each event
ANGM (25)	Magnetic correction angle from event to magnetic north
AS	Portion of event latitude in seconds
B	Angular distance from target to North Pole
D	Angle at center of earth to event and target locations
DIS	Surface distance in kilometers between event and target
ELAT (25)	Event latitudes in degrees
ELATR (25)	Event latitudes in radians
ELON (25)	Event longitudes in degrees
ELONR (25)	Event longitudes in radians
HOB (25)	Event heights of burst in kilometers
ID (200)	Target identification codes
IE	Constant of 1HE
IEVU	Event data set unit number (5 for batch version, 10 for TSO version)
IS	Constant of 1HS
ITARU	Targets data set unit number (5 for batch version, 11 for TSO version)
J	Loop index used for events
K	Loop index used for targets
KK	Loop index
LA	Event latitude code "N" or "S"
LO	Event longitude code "E" or "W"

Variable Name	Usage
NEV	Number of events
NTAR	Number of targets
OD	Portion of event longitude in degrees
OM	Portion of event longitude in minutes
OS	Portion of event longitude in seconds
PONP	Angle at North Pole between event and target
PI	π 3.14159265
PI2	$\pi/2$
RAD	Factor to convert degrees to radians
REA	Radius of the earth in kilometers
THETA	Bearing angle used in data retrieval call
TLAT (200)	Target latitude in degrees
TLATR (200)	Target latitude in radians
TLON (200)	Target longitude in degrees
TLONR (200)	Target longitude in radians
V (10)	Retrieved EMP data values
W	Factor to convert radians to degrees
YIELD (25)	Event yield in megatons

3.1.1.7 Routines Used

Subroutine INTERP - See Section 3.1.2.1.

The following FORTRAN library functions are used:

SIN - Sine

COS - Cosine

ARSIN - Inverse sine

ARCOS - Inverse cosine

MOD - Modulus remainder function

3.1.1.8 Order of Program Deck Including JCL Cards (for DCEC Computer Facility, Reston, Virginia)

```
//EMP2 Job (1763, R720, 60, 120), 'RXN50, U, SAUNDERS'
//STEP 1 EXEC FORTGCLG, PARM/ FORT = 'SOURCE'
//FORT. SYSIN DD*
- PROGRAM CARDS -
/*
//GO. FT09F001 DD DSNAME = M1763. EMPDAT, DISP = SHR
//GO. SYSIN DD *
- EVENT DATA CARDS -
- TARGET DATA CARDS -
/*
//
```

3.1.1.9 Detailed Flow Chart

See Figure 3-7.

3.1.2 Subroutine Description

3.1.2.1 Subroutine INTERP

3.1.2.1.1 Purpose

To permit retrieval of EMP values for a latitude other than 30, 45, or 60 degrees, by a linear interpolation between known values for 30, 45, and 60 degrees.

3.1.2.1.2 Calling Sequence

```
DIMENSION V (10)
.
.
.
CALL INTERP (SSL, SY, SH, SR, ST, V)
```

3.1.2.1.3 Detailed Description

Subroutine INTERP is capable of retrieving EMP values for ETOTAL, EVERT, ENORTH, EEAST, POYNTING, POLAR, ANGARR, RISETIME, PEAKWIDTH, and

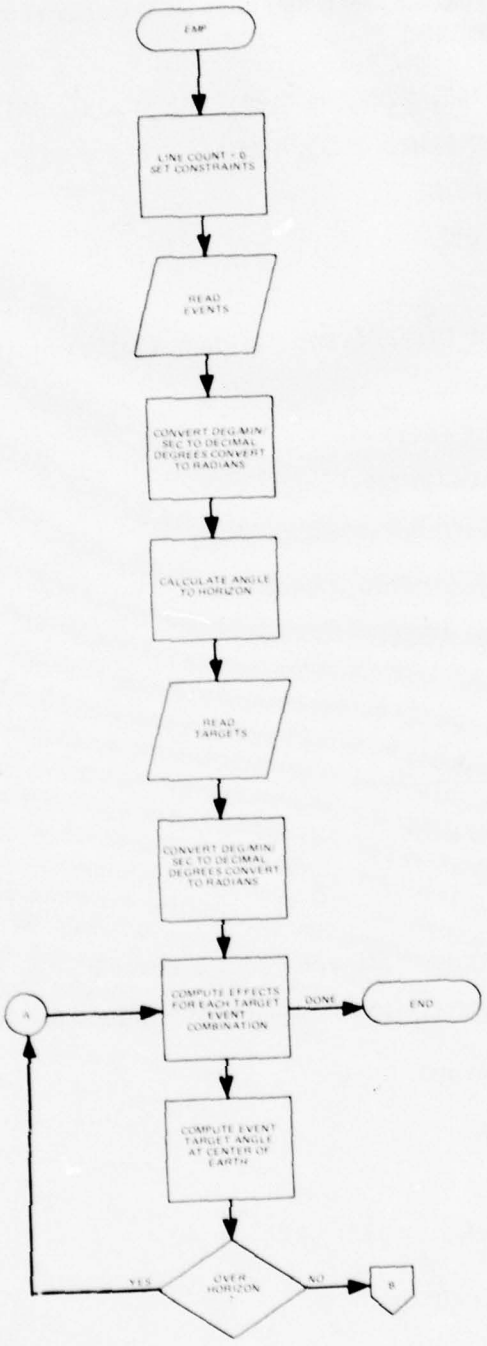


Figure 3-7. Main Program -- Detailed Flow Chart

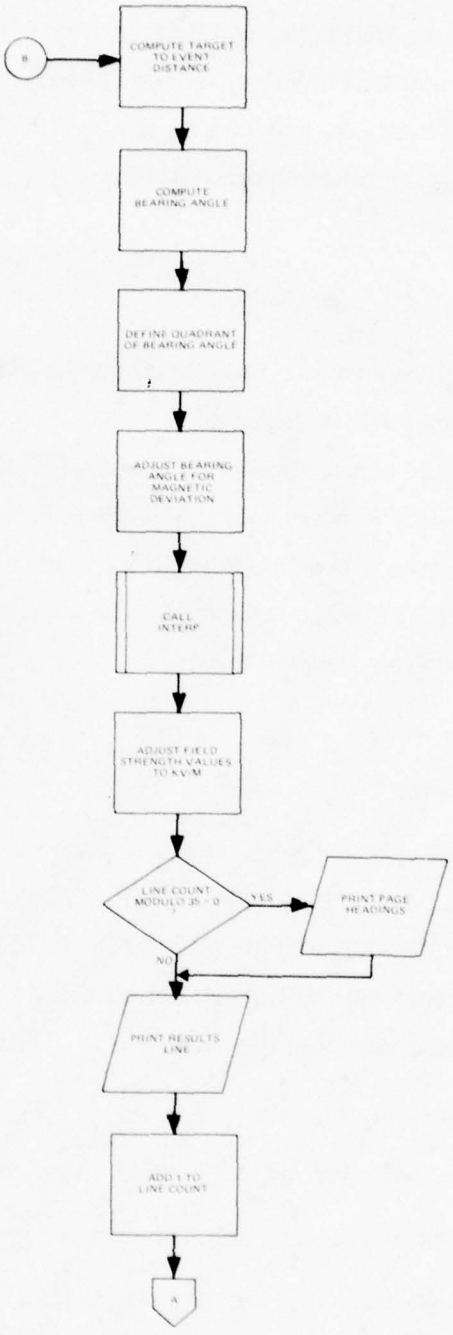


Figure 3-7. Main Program -- Detailed Flow Chart (cont'd)

TAIL. Search parameters YIELD and HOB must be exact values present in the data base. Search parameters BLAT, R, and THETA need not be exact values present in the data base. Retrieved EMP values will be linearly interpolated from the surrounding values of BLAT, R, and THETA that are in the data base file. See Paragraph 3.1.2.1.3.3 for interpolation details.

3.1.2.1.3.1 Inputs

Calling Parameters

SSL Search latitude value. Should be between 30 and 60 degrees. If <30, then 30 will be assumed. If >60, then 60 will be assumed.

SY Search yield value. Must be 1.25 or 5.00 megatons.

SH Search height of burst value. Must be 50, 100, 250, or 400 kilometers.

SR Search distance in kilometers from event to target

ST Search THETA angle in degrees. Bearing from event to target clockwise from true north.

3.1.2.1.3.2 Outputs

Returned Parameter

V A 10-word real array which will contain the EMP values for ETOTAL, EVERT, ENORTH, EEAST, POYNTING, POLAR, ANGARR, RISETIME, PEAKWIDTH, and TAIL, in that order. These values will be linearly interpolated for BLAT, R, and THETA from actual values in the data base.

Printed Messages:

Certain warning messages may be printed by subroutine INTERP.

"Latitude below 30 degrees. Values computed for 30 degrees."

"Latitude above 60 degrees. Values computed for 60 degrees."

Note: R and THETA have already been calculated using actual latitude and longitude coordinates. These changes affect only the retrieved data base values.

3.1.2.1.3.3 Processing

Retrieval.

Retrieval is done by calling subroutine SEARCH. See Paragraph 3.1.2.2.

Interpolation.

If the calling parameter SSL (SEARCH LATITUDE) takes a value of 30, 45, or 60 (or < 30 or > 60), only one call to subroutine SEARCH is made and no interpolation is necessary. If SSL is > 30 and < 45 or > 45 and < 60, two calls to subroutine SEARCH are made. The ten EMP data values are linearly interpolated from the two sets of values returned by subroutine SEARCH.

3.1.2.1.4 Data Organization

Variable Name	Usage
I	Do loop index.
K	Do loop index.
SH, SR, SSL, ST, SY	Input parameters.
SL (2)	The actual two latitudes used to call subroutine SEARCH.
V (10)	Output parameter. Also used for the first set of EMP values from SEARCH.
VY (10)	Second set of EMP values from SEARCH.
XLINT	Interpolation ratio.

3.1.2.1.5 Limitations

Search values for yield and height of burst may only take on certain values as described in Paragraph 3.1.2.1.3.1.

3.1.2.1.6 Routines Used

SEARCH subroutine to retrieve data base values.

3.1.2.1.7 Detailed Flow Chart

See Figure 3-8.

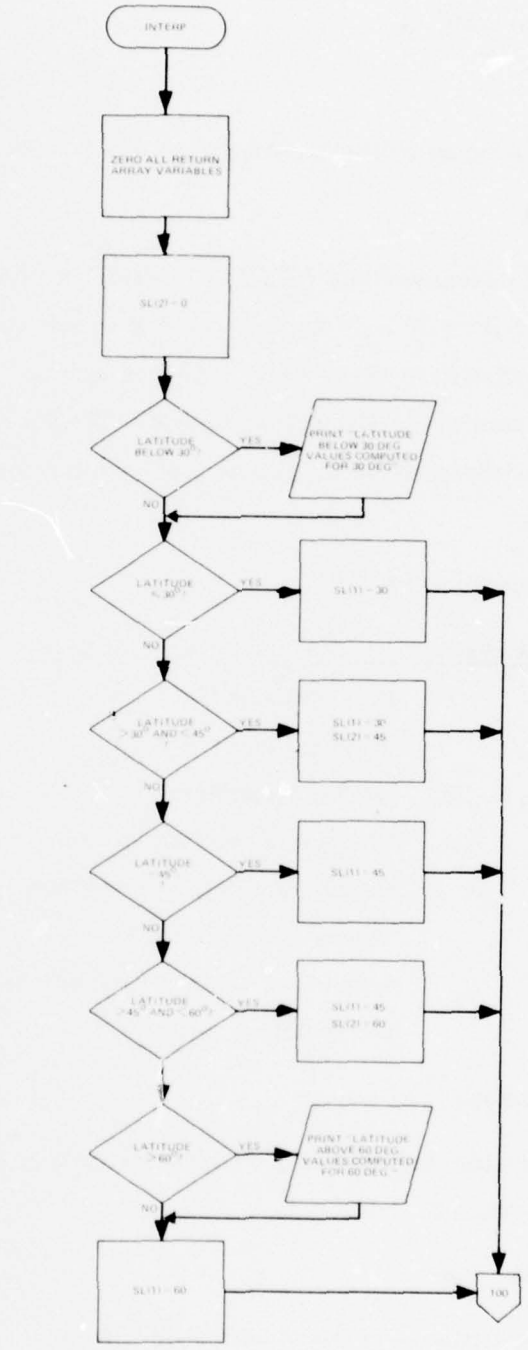


Figure 3-8. Subroutine INTERP -- Detailed Flow Chart

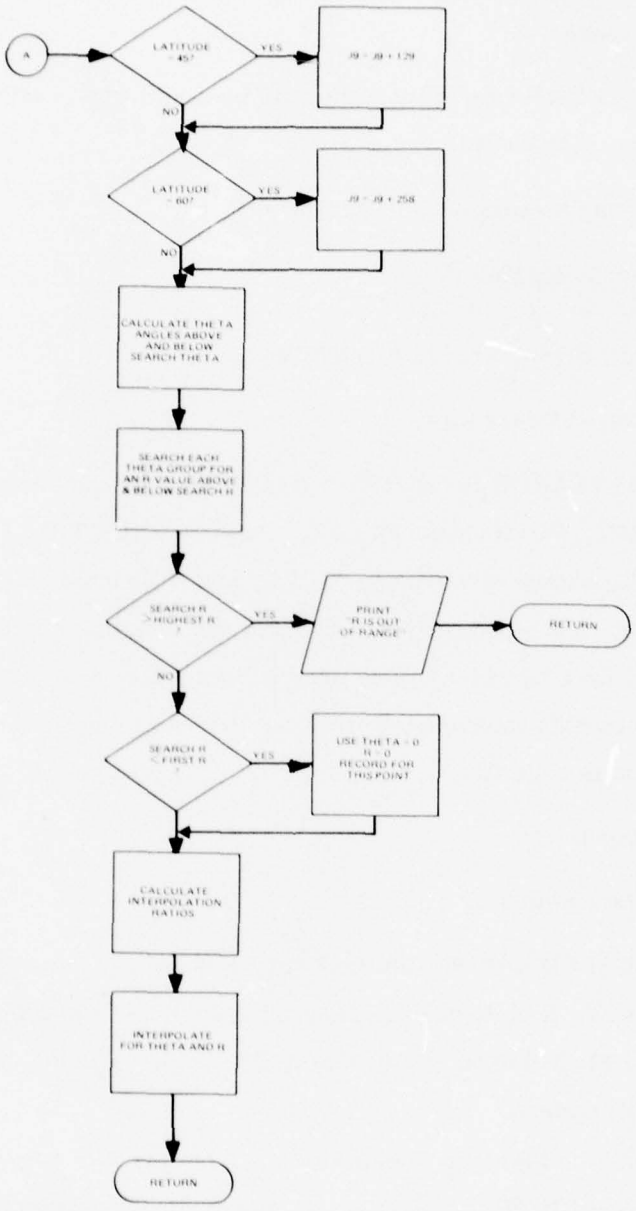


Figure 3-8. Subroutine INTERP -- Detailed Flow Chart (cont'd)

3.1.2.2 Subroutine Search

3.1.2.2.1 Purpose

To retrieve EMP values from the data base file when certain search parameter values are known.

3.1.2.2.2 Calling Sequence

```
Dimension VAL (10)
:
:
Call SEARCH (SL, SY, SH, SR, ST, VAL)
```

3.1.2.2.3 Detailed Description

Subroutine SEARCH is capable of retrieving EMP values for ETOTAL, EVERT, ENORTH, EEAST, POYNTING, POLAR, ANGARR, RISETIME, PEAKWIDTH, and TAIL. SEARCH parameters BLAT, YIELD, and HOB must be exact values present in the data base. See paragraph 3.1.2.2.3.1 for details. SEARCH parameters R and THETA need not be exact values present in the data base. Retrieved EMP values will be linearly interpolated from the four surrounding values of R and THETA that are in the data base file.

3.1.2.2.3.1 Inputs

Calling Parameters

SL SEARCH latitude value. Must be 30, 45, or 60 degrees.

SY SEARCH yield value. Must be 1.25 or 5.00 megatons.

SH SEARCH height of burst value. Must be 50, 100, 250, or 500 kilometers.

SR SEARCH distance in kilometers from event to target.

ST SEARCH THETA angle in degrees. Bearing from event to target clockwise from true north.

Data Base File:

The data base file is read from FORTRAN unit 9. Each 80-character record consists of the following fields:

NAME	FORMAT	DESCRIPION
IRUN	(I4)	Sequence number from 1 to 129
BLAT	(F5.1)	Burst latitude (degrees)
YIELD	(F5.2)	Burst yield (MT)
HOB	(F5.1)	Height of burst (km)
R	(F6.1)	Ground range (km) of line of sight at surface, measured from burst epicenter on surface
THETA	(F5.1)	Azimuthal bearing (degrees) of line of sight, clockwise from geographic north
ETOTAL	(F5.2)	Peak total electric field (as a fraction of 60 kV/m)
EVERT	(F5.2)	Peak vertical electric field (as a fraction of 60 kV/m) positive upward
ENORTH	(F5.2)	Peak north electric field (as a fraction of 60 kV/m) positive northward
EEAST	(F5.2)	Peak east electric field (as a fraction of 60 kV/m) positive eastward
POYNTING	(F5.4)	Time-integrated poynting vector of the free-field electric field (i.e., no ground reflection considered) (joules per square meter)
POLAR	(F6.1)	Polarization angle (degrees) of the electric field, defined as the angle the electric field is rotated counterclockwise from the horizontal. (This is not the angle between the electric field vector and its normal projection of the ground.)

NAME	FORMAT	DESCRIPTION
ANGARR	(F4.1)	Angle of arrival (degrees) of the line of sight measured at its intersection with the ground (i.e., angle between the line of sight and the horizontal)
RISETIME	(F4.1)	10% to 90% rise time of the amplitude of the total electric field (ns)
PEAKWIDTH	(F5.1)	90% to 90% shoulder width of the amplitude of the total electric field (ns)
TAIL	(F6.1)	90% to 10% delay time of the amplitude of the total electric field (ns)

3.1.2.2.3.2 Outputs

Returned Parameter:

VAL A 10-word real array which will contain the EMP values for ETOTAL, EVERT, ENORTH, EEAST, POYNTING, POLAR, ANGARR, RISETIME, PEAKWIDTH, and TAIL in that order. These values will be linearly interpolated for R and THETA from actual values in the data base.

Printed Messages:

Certain error messages may be printed by subroutine SEARCH.

"SEARCH DATA OUT OF RANGE"

See Paragraph 3.1.2.2.3.1 for the requirements placed on search data values.

"R is out of Range"

Value given for R is greater than the largest R for which a reading is present in the data base.

B.1.2.2.3.3 Processing

Retrieval

The data base file is a random access file. Each record may be read directly without having to sequentially read all previous records. The record number where appropriate data can be found can be calculated by knowing the layout of the file.

The file consists of 3096 records which can be thought of as 24 groups of 129 records. Table 3-1 shows the values for YIELD, BLAT, and HOB for each group. The 129 records in each group consist of a single record of measurement data for THETA = 0 and R = 0, followed by eight measurements taken at each of 16 THETA angles (0°, 22.5°, 45°, 67.5°, 90°, 112.5°, 135°, 157.5°, 180°, 202.5°, 225°, 247.5°, 270°, 292.5°, 315°, and 337.5°).

A record number (variable name J9) is calculated based on the search data values. First, YIELD value determines which half of the file the desired record will be in. So, if YIELD = 1.25, J9 is set to 1. If YIELD = 5.0, J9 is set of 1549 (the first record in the second half).

Next, the HOB value determines the quarter of the selected half of the file

IF HOB =	THEN J9 = J9 +
50	0
100	387
250	774
400	1161

Next, the BLAT value determines the third of the selected eighth of the file

IF BLAT =	THEN J9 = J9 +
30	0
45	129
60	258

At this point, the group of 129 records which contain the appropriate data has been determined.

Table 3-1. Values for YIELD, HOB, and BLAT for Each Group

Group	YIELD	HOB	BLAT
1	1.25	50	30
2	1.25	50	45
3	1.25	50	60
4	1.25	100	30
5	1.25	100	45
6	1.25	100	60
7	1.25	250	30
8	1.25	250	45
9	1.25	250	60
10	1.25	400	30
11	1.25	400	45
12	1.25	400	60
13	5.00	50	30
14	5.00	50	45
15	5.00	50	60
16	5.00	100	30
17	5.00	100	45
18	5.00	100	60
19	5.00	250	30
20	5.00	250	45
21	5.00	250	60
22	5.00	400	30
23	5.00	400	45
24	5.00	400	60

The specified THETA angle value divides this group into 16 subgroups of 8 or 9 records each. Since the R values stored in the data base follow no fixed sequence, these 8 or 9 records must be sequentially scanned to find two records--one with an R value less than or equal to the search R value, and another with an R value greater than the search R value.

As a special case, a search R value between zero and the first measured R value for the given THETA value must use the R = 0, THETA = record as the lower record regardless of the given THETA value.

The sequential search is done twice: once for a THETA value lower than or equal to the search THETA value, and again for the next higher THETA value. Note that THETA = 0° is considered to be the next higher value from THETA = 337.5°.

Four records have thus been read, and the final results can be interpolated from them.

Interpolation:

Figure 3-9 shows the interpolation procedure. Point S is defined by the search R and THETA values. Points 1, 2, 3, and 4 are the points about point S that actually contain values in the data base.

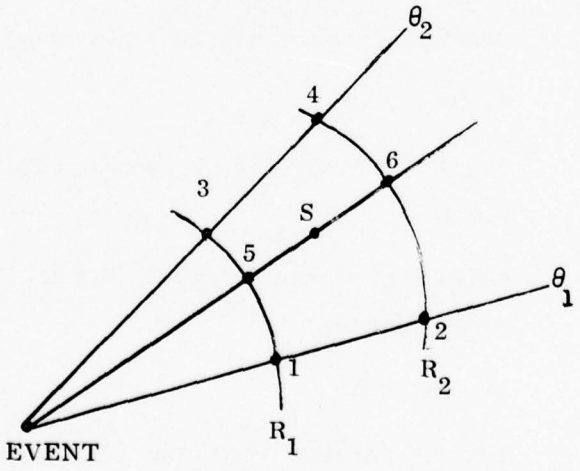


Figure 3-9. Interpolation Procedure

Values for points 1 and 3 are linearly interpolated on THETA to produce values for point 5. Values at points 2 and 4 are also linearly interpolated on THETA to produce values for point 6. Finally, the values for points 5 and 6 are linearly interpolated on R to produce the values for point S.

3.1.2.2.4 Data Organization

Variable Name	Usage
BLAT(5)	Array of latitude values read from data base records
HOB(5)	Array of height of burst values read from data base records
I	Do loop index
IRUN(5)	Array of run number values read from data base records
IT(5)	Array of THETA angle value indices in range 0 to 15 which define THETA angle values around the search THETA value.
J	= 0 if current group has 8 records = 1 if current group has 9 records (THETA = 0^{o})
J9	Number of current data base record being read
K	Do loop index
L9	Number of first data base record in 129 record group
M	Upper limit of search LOOP. Number of records in this group (8 or 9)
N	Do loop index
R(5)	Array of R values read from data base records
RINT	Interpolation ratio for R values

SH, SL, SR, ST, SY	Input parameters. See Paragraph 3.1.2.2.3.1.
T1, T2	Temporary variables corresponding to values at points 5 and 6 in Figure 3-9
THETA(5)	Array of THETA values read from data base records
TINT	Interpolation ratio for THETA values
V(10,5)	Array of EMP measurement values read from data base
VAL(10)	Output array. See Paragraph 3.1.2.2.3.2.
YIELD (5)	Array of yield values read from data base.

3.1.2.2.5 Limitations

This subroutine will run only in a FORTRAN which supports a random or direct file access procedure. It could be modified to use sequential access, however, run time and CPU time would be greatly increased.

3.1.2.2.6 Routines Used

IFIX FORTRAN function to convert real to integer.

3.1.2.2.7 Detailed Flow Chart

See Figure 3-10

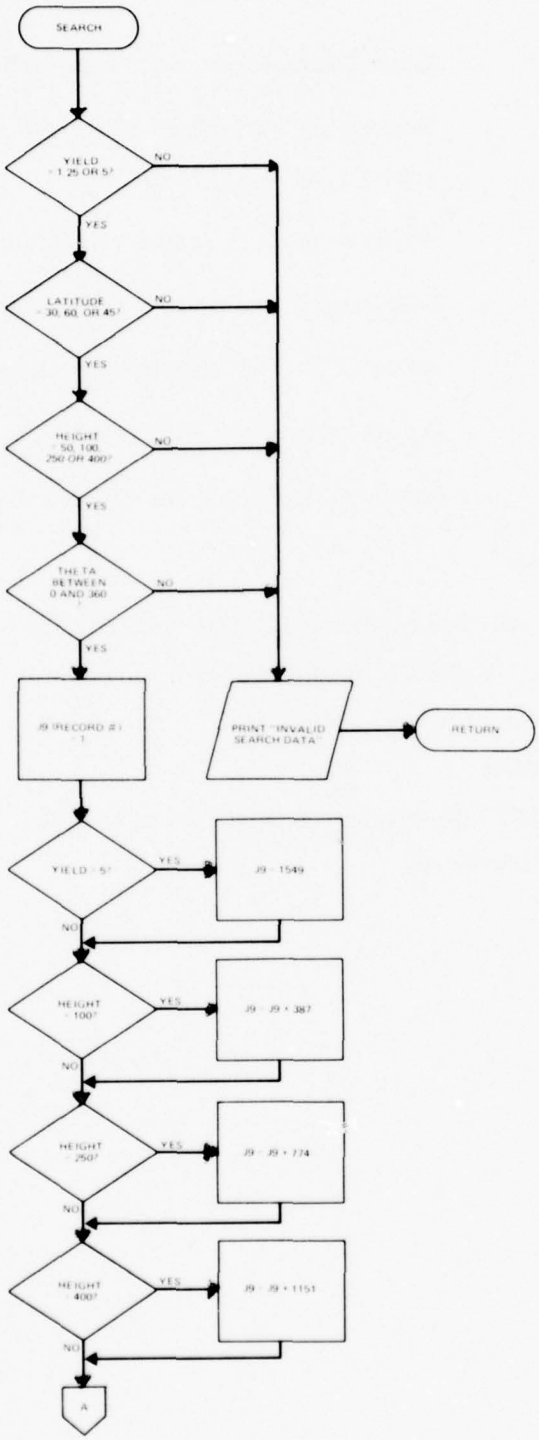


Figure 3-10. Subroutine SEARCH -- Detailed Flow Chart

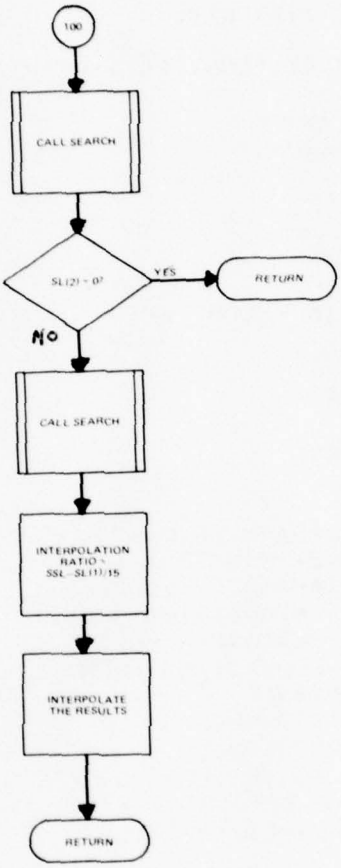


Figure 3-10. Subroutine SEARCH -- Detailed Flow Chart (Cont'd)

3.1.2.2.8 Program Listing

```
C       EMP BLAST EFFECTS PROGRAM
C       EVENT DATA
        DIMENSION ELAT(25),ELON(25),ELATR(25),ELONR(25),YIELD(25),HOB(25)
        DIMENSION ANGH(25),ANGM(25)
C       TARGET DATA
        DIMENSION TLAT(200),TLON(200),TLATR(200),TLONR(200),ID(200)
        DIMENSION V(10)
        DATA IS/1HS/, IE/1HE/
        DATA IEVU/5/, ITARU/5/
        LINCNT = 0
        PI=3.14159265
        PI2=PI/2.
        RAD=.017453
        W=57.295779
C       RADIUS OF EARTH IN KILOMETERS
        REA=6375.6625
C
C       READ THE EVENT DATA
C
        READ(IEVU,100) NEV
  100   FORMAT(I3)
        DO 20 J = 1,NEV
           READ(IEVU,101)AD,AM,AS,LA,OD,OM,OS,LO,YIELD(J),HOB(J),ANGM(J)
C          CONVERT DEGREES, MINUTES, SECONDS TO FRACTIONAL DEGREES
           ELAT(J) = AD+(AM/60.)+(AS/3600.)
           ELON(J) = OD+(OM/60.)+(OS/3600.)
           IF (LA.EQ.IS) ELAT(J) = -ELAT(J)
           IF (LO.EQ.IE) ELON(J) = -ELON(J)
C          CONVERT TO RADIANS
           ELATR(J) = ELAT(J)*RAD
           ELONR(J) = ELON(J)*RAD
C          COMPUTE ANGLE TO THE HORIZON FOR THIS EVENT
C          ESTABLISHES THE EFFECTIVE RANGE OF THIS EVENT
           ANGH(J) = ARCOS(REA/(REA+HOB(J)))
   20   CONTINUE
  101   FORMAT(3F2.0,A1,F4.0,2F2.0,A1,F5.2,F4.0,F5.1)
C
C       READ THE TARGET DATA
C
        READ(ITARU,100) NTAR
        DO 30 K = 1,NTAR
           READ(ITARU,102) ID(K),AD,AM,AS,LA,OD,OM,OS,LO
           TLAT(K) = AD+(AM/60.)+(AS/3600.)
           TLON(K) = OD+(OM/60.)+(OS/3600.)
           IF (LA.EQ.IS) TLAT(K) = -TLAT(K)
           IF (LO.EQ.IE) TLON(K) = -TLON(K)
C          CONVERT TO RADIANS
           TLATR(K) = TLAT(K)*RAD
           TLONR(K) = TLON(K)*RAD
   30   CONTINUE
  102   FORMAT(I3,1X,3F2.0,A1,F4.0,2F2.0,A1)
```

```
C
C       COMPUTE EMP EFFECTS FOR EACH TARGET
C
        DO 90 K = 1,NTAR
C          COMPUTE EFFECT OF EACH EVENT ON THIS TARGET
           DO 80 J = 1,NEV
C             TARGET ANGLE
              B = PI2-TLATR(K)
              A = PI2-ELATR(J)
              PONP = ABS(TLONR(K)-ELONR(J))
              D = ARCOS(COS(B)*COS(A)+SIN(B)*SIN(A)*COS(PONP))
C             IF OVER HORIZON, THEN NO EFFECT
              IF (D.GE.ANGH(J)) GO TO 80
              DIS = REA*D
              ANGR = ARSIN(SIN(PONP)*SIN(B)/SIN(D))
C
C             DEFINE QUADRANT OF TARGET
C
              IF (TLONR(K).GT.ELONR(J)) GO TO 40
              IF (TLATR(K).GT.ELATR(J)) GO TO 60
C
C             FOURTH QUADRANT
              ANGR = PI-ANGR
              GO TO 60
C
   40         IF (TLATR(K).GE.ELATR(J)) GO TO 50
C
C             THIRD QUADRANT
              ANGR = PI+ANGR
              GO TO 60
C
C             SECOND QUADRANT
   50         ANGR = PI2-ANGR
C
C             FIRST QUADRANT
   60         CONTINUE
C
C             CONVERT ANGLE TO DEGREES
              ANG = ANGR*W
C
C             RETRIEVE AND INTERPOLATE THE EMP VALUES
C
              THETA = ANG+ANGM(J)-2.
              CALL INTERP(ELAT(J),YIELD(J),HOB(J),DIS,THETA,V)
C
C             ADJUST TOTAL, VERT, NORTH, AND EAST TO KV/M
C
              DO 70 KK = 1,4
                 V(KK) = V(KK)*60.
   70         CONTINUE
```

```
C
C      PRINT THE RESULTS
C
              IF (MOD(LINCNT,35).NE.0) GO TO 75
              WRITE(6,103)
              WRITE(6,104)
              WRITE(6,105)
   75         CONTINUE
              ANG1 = ANGM(J)+ANG
              WRITE(6,106) ID(K),J,YIELD(J),HOB(J),ELAT(J),ELON(J),
     1        TLAT(K),TLON(K),DIS,ANG,ANG1,V(1)
              LINCNT = LINCNT+1
C
   80     CONTINUE
   90 CONTINUE
  103 FORMAT  ('1 TG EV YIELD HEI-  EVENT   EVENT TARGET  TARGET  TARGET
     1 TARGET TARGET TOTAL')
  104 FORMAT(1X,'                  GHT   LAT.   LONG.   LAT.   LONG.   DIST.
     1 ANG(T) ANG(M)  F.S.')
  105 FORMAT(1X,'--- -- ----- ---- ------ ------- ------ ------- -------
     1 ------ ------ -----',/)
  106 FORMAT(1X,2I3,F6.2,F5.0,F7.2,F8.2,F7.2,2F8.2,2F7.2,F6.2,/)
      STOP
      END
      SUBROUTINE INTERP(SSL,SY,SH,SR,ST,V)
C     THIS SUBROUTINE PERFORMS A LINEAR INTERPOLATION FOR A GIVEN
C     LATITUDE (SSL) BETWEEN 30 AND 60 DEGREES
      DIMENSION V(10),SL(2),VY(10)
      DO 10 I=1,10
         V(I) = 0.
         VY(I) = 0.
   10 CONTINUE
      SL(2) = 0.
      IF (SSL.LT.30.) WRITE(6,1)
    1 FORMAT(' LATITUDE BELOW 30 DEGREES. VALUES COMPUTED FOR 30 DEGREES
     2.')
      IF (SSL.LE.30.) GO TO 30
      IF (SSL.GT.30..AND.SSL.LT.45.) GO TO 3045
      IF (SSL.EQ.45.) GO TO 45
      IF (SSL.GT.45..AND.SSL.LT.60.) GO TO 4560
      IF (SSL.GT.60.) WRITE(6,2)
    2 FORMAT(' LATITUDE ABOVE 60 DEGREES. VALUES COMPUTED FOR 60 DEGREES
     2.')
      SL(1) = 60.
      GO TO 100
 3045 SL(1) = 30.
      SL(2) = 45.
      GO TO 100
   45 SL(1) = 45.
      GO TO 100
 4560 SL(1) = 45.
      SL(2) = 60.
      GO TO 100
```

```
   30 SL(1) = 30.
  100 CONTINUE
      CALL SEARCH(SL(1),SY,SH,SR,ST,V)
C     IF EXACT LATITUDE, NO NEED FOR SECOND CALL AND INTERPOLATION
      IF(SL(2).EQ.0.) RETURN
C     RETRIEVE THE SECOND LATITUDE
      CALL SEARCH (SL(2),SY,SH,SR,ST,VY)
C     CALCULATE LATITUDE INTERPOLATION RATIO
      XLINT = (SSL-SL(1))/15.
C     INTERPOLATE THE RETURN VALUES
      DO 110 K = 1,10
         V(K) = V(K)+(VY(K)-V(K))*XLINT
  110 CONTINUE
      RETURN
      END
      SUBROUTINE SEARCH (SL,SY,SH,SR,ST,VAL)
C     THIS SUBROUTINE SEARCHES THE EMP DATA BASE FILE. LATITUDE, HOB,
C     AND YIELD MUST BE EXACT VALUES PRESENT IN THE DATA BASE.
C     R AND THETA WILL BE LINEARLY INTERPOLATED FROM VALUES THAT ARE IN
C     THE DATA BASE.
C
      DEFINE FILE 9(3096,80,E,J9)
      DIMENSION IRUN(5),BLAT(5),YIELD(5),HOB(5),R(5),THETA(5)
      DIMENSION V(10,5),VAL(10),IT(5)
C
C     VERIFY THE VALIDITY OF SEARCH DATA
C
C     YIELD MUST BE 5.00 OR 1.25
C
      IF (SY.NE.5..AND.SY.NE.1.25) GO TO 800
C
C     LATITUDE MUST BE 30, 45, OR 60
C
      IF (SL.NE.30..AND.SL.NE.45..AND.SL.NE.60.) GO TO 800
C
C     HEIGHT OF BURST MUST BE 50, 100, 250, OR 400.
C
      IF (SH.NE.50..AND.SH.NE.100..AND.SH.NE.250..AND.SH.NE.400.) GO TO
     2   800
C
C     THETA ANGLE MUST BE BETWEEN 0 AND 360
C
      IF (SL.LT.0) GO TO 800
   15 CONTINUE
      IF (ST.LT.360) GO TO 20
      ST = ST - 360
      GO TO 15
   20 CONTINUE
C
C     START BY POINTING TO FIRST RECORD
C
```

```
      J9=1
C     IF YIELD IS 5 SET TO START OF SECOND HALF
      IF(SY.EQ.5.) J9 =  549
C
C     HEIGHT OF BURST DETERMINES WHICH QUARTER OF THIS HALF
C
      IF (SH.EQ.100.) J9=J9+ 387
      IF (SH.EQ.250.) J9=J9+ 774
      IF (SH.EQ.400.) J9=J9+1161
C
C     LATITUDE DETERMINES WHICH THIRD OF THIS QUARTER
C
      IF (SL.EQ.45.) J9=J9+129
      IF (SL.EQ.60.) J9=J9+258
C
C     NEXT, THETA ANGLE DIVIDES THIS GROUP INTO SIXTEENTHS
C     ANGLE NEED NOT MATCH EXACTLY, SO WILL FIND WHICH TWO GROUPS ITS BETWEEN
C     IT(2) IS LOWER OR EQUAL ANGLE THAT IS IN THE DATA BASE, AND IT(4)
C     IS NEXT HIGHER ANGLE THAT IS IN THE DATA BASE
C
      L9 = J9
      IT(2) = IFIX(ST/22.5)
      IT(4) = IT(2) + 1
      IF (IT(4).EQ.16) IT(4) = 0
C
C     EACH THETA ABOVE AND THEATA BELOW DEFINE 8 OR 9 RECORDS WHICH
C     MUST BE SEQUENTIALLY SEARCHED TO FIND THE TWO RECORDS WITH AN R
C     VALUE ABOVE AND BELOW THE SEARCH R VALUE.
C     IT(1) AND IT(3) ARE BELOW OR EQUAL TO THE SEARCH R AND IT(2) AND
C     IT(4) ARE ABOVE THE SEARCH R
C
      DO 30 K=1,5,2
         IT(K)     =0
         IRUN(K)   =0
         BLAT(K)   =0.
         YIELD(K)  =0.
         HOB(K)    =0.
         R(K)      =0.
         THETA(K)  =0.
         DO 29 I = 1,10
            V(I,K) = 0.
   29    CONTINUE
   30 CONTINUE
C
C     DO FOR THETA ABOVE AND THETA BELOW
C
      DO 60 K=2,4,2
C        SET IT(K) TO FIRST RECORD IN THIS THETA GROUP
C        EIGHT RECORDS IN EACH GROUP
         IT(K) = IT(K)*8
```

```
C         EXCEPT THE FIRST WHICH HAS 9
          J = 0
          IF (IT(K).NE.0) J = 1
          IT(K) = IT(K)+L9*J
C         SET RECORD NUMBER TO READ
          J9 = IT(K)
C
C         DO FOR 8 OR 9 RECORDS IN THIS GROUP
C
          M = 9-J
          DO 40 I = 1, M
             READ (9'J9,11) IRUN(K),BLAT(K),YIELD(K),HOB(K),R(K),
     2       THETA(K),(V(N,K),N=1,10)
    11 FORMAT(I4,F5.1,F5.2,F5.1,F6.1,F5.1,4F5.2,F5.4,F6.1,2F4.1,F5.1,F6.1
     2)
C            IF THIS R IS GREATER THAN SEARCH R, SEARCH R HAS BEEN
C            BRACKETED
             IF (R(K).GT.SR) GO TO 50
C
C            IF NOT, SAVE THIS RECORD AS POSSIBLE LOWER OR EQUAL VALUE
C
C            SAVE RECORD NUMBER
             IT(K-1) = J9-1
             IRUN(K-1) = IRUN(K)
             BLAT(K-1) = BLAT(K)
             YIELD(K-1) = YIELD(K)
             HOB(K-1) = HOB(K)
             R(K-1) = R(K)
             THETA(K-1) = THETA(K)
             DO 39 N = 1,10
                V(N,K-1) = V(N,K)
    39       CONTINUE
    40    CONTINUE
C         IF NO FIND, MUST BE OVER RANGE
          WRITE(6,14)
    14    FORMAT(18H R IS OUT OF RANGE)
          RETURN
C         IF FIRST ENTRY IS >SR, READ THE FIRST THETA=0 RECORD TO GET R=0 VALUE
    50    IT(K) = J9-1
          IF (I.NE.1) GO TO 55
          J9 = L9
          READ (9'J9,11) IRUN(K-1),BLAT(K-1),YIELD(K-1),HOB(K-1),R(K-1),
     2    THETA(K-1),(V(N,K-1),N=1,10)
    55    IT(K-1) = J9-1
    60 CONTINUE
C
C     COMPUTE INTERPOLATION RATIOS
C
      TINT = (ST-THETA(2))/22.5
      RINT = (SR-R(1))/(R(2)-R(1))
```

```
C
C     INTERPOLATE THE RESULTS
C
      DO 70 N = 1 , 10
         T2 = V(N, 2) + TINT*(V(N, 4) - V(N, 2))
         T1 = V(N, 1) + TINT*(V(N, 3) - V(N, 1))
         VAL (N) = T1 + RINT*(T2-T1)
   70 CONTINUE
      RETURN
C
C     SEARCH DATA OUT OF RANGE
C
  800 WRITE (6, 13)
   13 FORMAT  (20H INVALID SEARCH DATA)
      RETURN
      END
```

3.2 NETWORK ANALYSIS PROGRAMS

Network analysis computer programs are needed to help in the automated evaluation of the transatlantic communications assets before and after the introduction of nuclear threats in the environment. The programs have been developed in the PL/I language, basically for its character manipulation capability, powerful built-in functions, and its structured command format. The Defense Communications Engineering Center's Hybrid Simulation Facility (containing an IBM 370/155 computer) was used to develop the following three "stand-alone" programs:

1. Circuit Tracer Program
2. Primitive Connection Matrix Program
3. Connectivity (Routing) Search Program.

The first program, acting on the transatlantic-communications-assets-data-base, determines the sites and the links traversed by specified circuits. The second program, again acting on the data base, computes the primitive connection matrices of the switched networks --AUTOVON, AUTODIN, and AUTOSEVOCOM. A primitive connection matrix provides the "next neighbor" information about the sites in the network. Using this information, the third program determines all unique "simple" paths from one specified site to another. A "simple" path is defined as an alternating sequence of sites and links in which no site appears more than once.

3.2.1 *Existing Programs*

3.2.1.1 Circuit Tracer Program

This program traces a circuit path in terms of alternating sequence of sites and links traversed by it, along with the In/Out flags of the sites and the links. The program is based on a simple search technique in which the trunks traversed by the circuit, then the links (and the sites) traversed by the trunks carrying the circuit, are determined as they exist in the data base using TOTAL® commands.

The program does not use any external procedures. It contains traps to detect events such as:

1. The input circuit does not exist in Circuit Master File
2. The trunk segment end sites are not properly specified in Circuit-Trunk-Site and Trunk-Link-Site Variable Files.

A "normal" exit is taken from the program if on reading a record in any file of the data base, the TOTAL returned status is "abnormal."

Figure 3-11 shows the functional input/output block diagram of this program. The inputs are: 1) CKT and LCKTNO contained in file, DECAIN and, 2) (FILE) where:

```
/*          (FILE): THIS REFERS TO ALL USER DEFINED TOTAL
                    FILES INCLUDING:
                      MASTER FILES: RMCT, RMTR, RMLK, RMSI
                                    AND
                      VARIABLE FILES: RVCT, RVTL
                CKT: CIRCUIT CCSD# (9 CHARACTERS)
             LCKTNO: NUMBER OF CIRCUITS TO BE TRACED        */
```

The outputs LKCKT, SICKT and SERIES are contained in file, DECAOUT where:

```
/*          LKCKT: A STRING OF CHARACTERS CONTAINING CCSD#
                   AND CIRCUIT IN/OUT FLAG (F) AS HEADER
                   FOLLOWED BY THE NAMES OF THE LINKS
                   TRAVERSED ALONG WITH THEIR IN/OUT FLAGS
                   AND A SPECIAL CHARACTER, @ AS TRAILER
                    (605 CHARACTERS FOR MAXIMUM ALLOWED
                     99 LINKS ON THE CIRCUIT PATH)
```

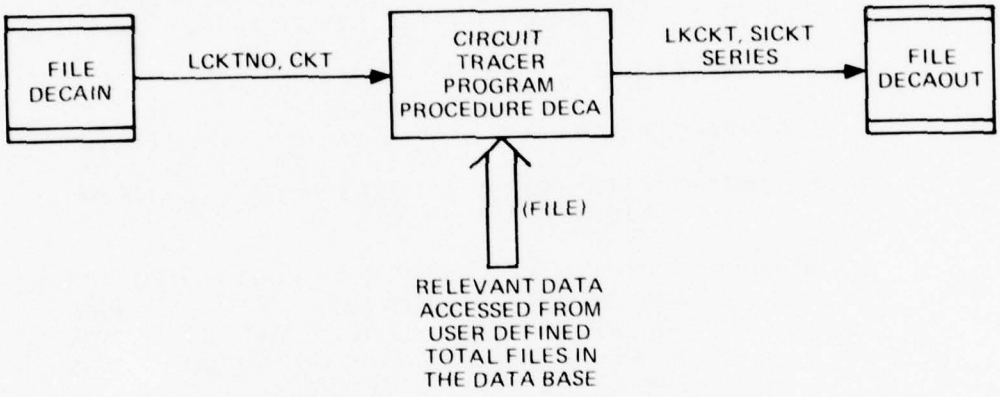
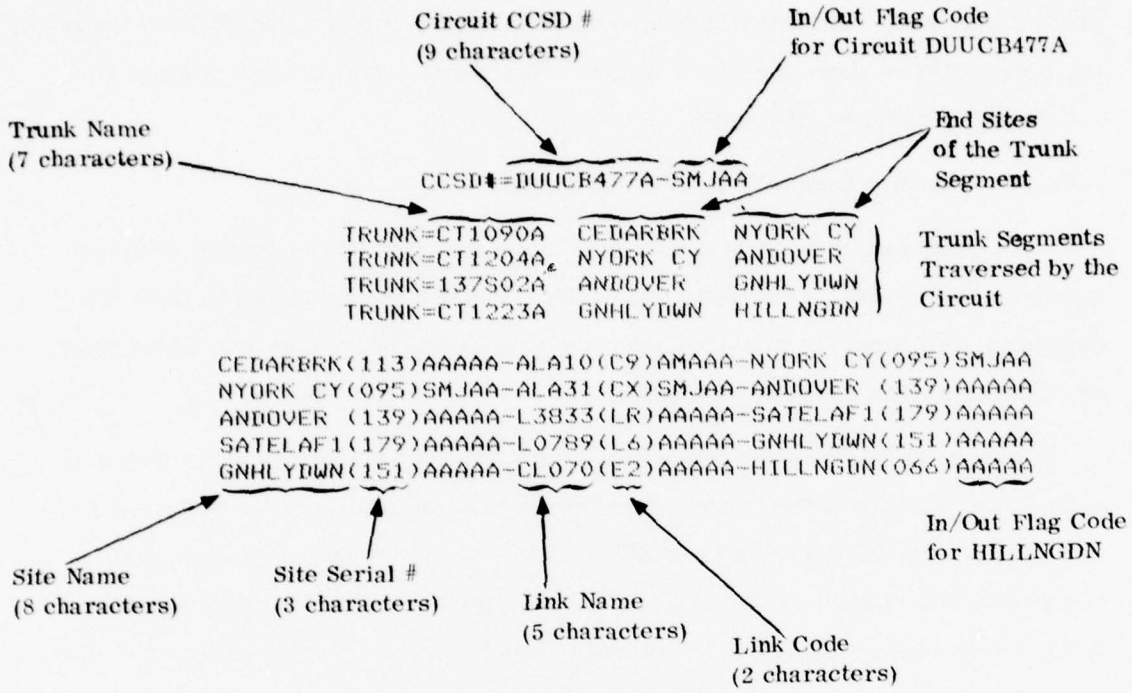


Figure 3-11. Functional Input/Output Block Diagram for Circuit Tracer Program

```
EXAMPLE:1:

CCSD#FLINK1FLINK2F----------LINK99F@

  (NOTE- FLAG,F IS 1 CHARACTER AND LINK1
         THROUGH LINK99 EACH CONTAINS 5
         CHARACTERS)

SICKT: A STRING OF CHARACTERS CONTAINING CCSD#
       AND CIRCUIT IN/OUT FLAG (F) AS HEADER
       FOLLOWED BY THE NAMES OF THE SITES
       TRAVERSED ALONG WITH THEIR IN/OUT FLAGS
       AND A SPECIAL CHARACTER, @ AS TRAILER
       ( 911 CHARACTERS FOR MAXIMUM ALLOWED
         100 SITES ON THE CIRCUIT PATH )

EXAMPLE:2:

CCSD#FSITE1FSITE2F----------SITE100F@

  (NOTE- SITE1 THROUGH SITE100 EACH CONTAINS
         8 CHARACTER SITE NAME)

SERIES: A STRING OF CHARACTERS CONTAINING CCSD#
       AND CIRCUIT IN/OUT FLAG (F) AS HEADER
       FOLLOWED BY THE SITE SERIAL #S OF THE SITES
       , THE LINK CODES AND THE IN/OUT FLAGS OF
       THE LINKS TRAVERSED AND A SPECIAL
       CHARACTER, @ AS TRAILER ( 608 CHARACTERS
       FOR MAXIMUM ALLOWED 99 LINKS AND 100
       SITES )

EXAMPLE 3:

CCSD#FSITENO1LINKCODE1F------LINKCODE99FSITENO100@

  (NOTE- SITENO1 THROUGH SITENO100 EACH CONTAINS
         3 CHARACTER SITE SERIAL #
                   AND
         LINKCODE1 THROUGH LINKCODE99 EACH
         CONTAINS 2 CHARACTER LINK CODE)
```

The above outputs, sorted to print the circuit information in an easily readable form on file DECAOUT, are as follows:



The characters in the flag code (referred from left to right) are to be decoded as follows:

Character # 1
- S ≙ Satellite Link Outage
- A ≙ No Outage for above Reason

Character # 2
- M ≙ MHD Outage
- A ≙ No Outage for above Reason

Character # 3
- J ≙ Jamming Outage
- A ≙ No Outage for above Reason

Character # 4
- B ≙ Blast Outage
- A ≙ No Outage for above Reason

Character # 5
- E ≙ EMP Outage
- A ≙ No Outage for above Reason

There are constraints on the size of some identifiers such as LKCKT, SICKT, and SERIES mainly to save the core size of the program. The experience in the present data base indicates that no circuit will ever span more than 99 links justifying the constraints on these three identifiers. Such constraints are user-defined and can be easily changed, if needed.

3.2.1.2 Primitive Connection Matrix Program

This program, upon gathering information about the three specified switched networks (AUTOVON, AUTODIN, and AUTOSEVOCOM in this instance) from the data base, computes the primitive connection matrices of the networks in terms of two-character link codes as elements.

The core of this program is two external procedures (Figure 3-12). The program "serially" reads the Circuit Master File to access all circuits in the networks one by one. The first external procedure is the Circuit Tracer Program, which traces the path of each circuit in the network in terms of a string of characters, SERIES (see Paragraph 3.2.1.1.). SERIES contains an alternating sequence of site serial #s and link codes, along with the In/Out flags of the links traversed by a circuit. This information is used by the second external procedure, called the Network Adder Program to build up the primitive connection matrix of a network. The Network Adder Program will examine all links in the circuit path. A link will be added to the "pertinent" network to which the circuit under consideration belongs only if the link is "In", as indicated by its In/Out flag. If multiple links are found between two sites in a network during the build up procedure, only the "first found" link is kept in the network and an information message is printed if the multiple link is not the same as the one that already exists between the sites in the network.

The program also contains internal procedures provided for such utility functions as:

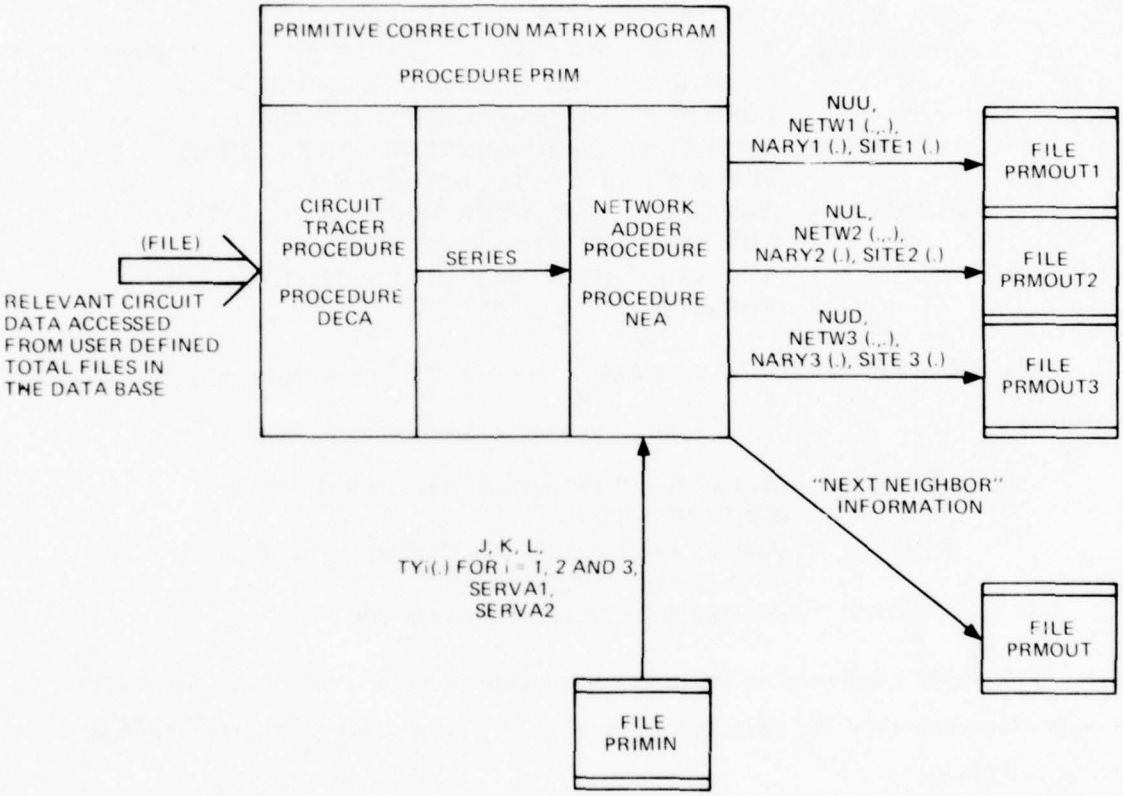


Figure 3-12. Functional Input/Output Block Diagram for Primitive Connection Matrix Program

N:	NUMBER OF SITES IN THE NETWORK
PCM(. . .):	PRIMITIVE CONNECTION MATRIX OF THE NETWORK WITH LINK CODES (2 CHARACTERS EACH) AS ELEMENTS
SITES(.):	ARRAY OF SITE SERIAL #S IN THE NETWORK
SITEN(.):	ARRAY OF SITE NAMES (8 CHARACTERS EACH) IN THE NETWORK
NFF:	"FROM" SITE SERIAL # (3 CHARACTERS)
NTT:	"TO" SITE SERIAL # (3 CHARACTERS)
PATHLTH:	MAXIMUM LENGTH OF A SIMPLE PATH (IN TERMS OF THE NUMBER OF LINKS TRAVERSED) TO BE USED IN PRINTING OUT SIMPLE PATHS
NUMC:	NUMBER OF SITES WHICH ARE "SINK" SITES FOR SOME (SAY N TOTAL) LINKS
FROMTOS(.):	ARRAY OF CHARACTER STRING CONTAINING THE "SINK" SITE NAME (IN THE ABOVE SENSE) AND SITE NAMES (MAXIMUM N=20 ALLOWED) FOR WHICH IT IS A SINK

The outputs SITEN (PDL(.)), PTAG(•) and NPATH are contained in file CNSEOTi where:

SITEN(PDL(.)):	ARRAY OF SITE NAMES TRAVERSED IN A SIMPLE PATH
PTAG (.):	ARRAY OF LINKS (LINK CODES) TRAVERSED IN A SIMPLE PATH
NPATH:	NUMBER OF SIMPLE PATHS FOUND

The major constraint in the program is again the size of the largest primitive connection matrix of the switched network and the comments made in Paragraph 3.2.1.2 apply.

1. Correlation of site names with site serial numbers
2. Correlation of link names with link codes
3. Alphanumeric sorting of arrays.

The program contains traps in addition to some of those already mentioned in the Circuit Tracer Program. Some examples are:

1. Omit the information about a circuit and go to the next circuit in the Circuit Master File if there is a file error, as indicated by "abnormal" TOTAL returned status in the Circuit Tracer Program. This will ensure that the program will not stop as a result of file error(s) in a few circuits.
2. The trap regarding the multiple links between sites as mentioned earlier in the paragraph.

Figure 3-12 shows the functional input/output block diagram of this program. The inputs are 1) J,K,L. (TYi(·) for i = 1, 2 and 3), SERVA1 and SERVA2 contained in file PRIMIN and 2) (FILE), where:

```
(FILE): THIS REFERS TO ALL USER DEFINED TOTAL
        FILES INCLUDING;
         MASTER FILES: RMCT,RMTR,RMLK,RMSI
                       AND
         VARIABLE FILES: RVCT,RVTL
     J: NUMBER OF IDENTIFIERS FOR FIRST
        SWITCHED NETWORK
     K: NUMBER OF IDENTIFIERS FOR SECOND
        SWITCHED NETWORK
     L: NUMBER OF IDENTIFIERS FOR THIRD
        SWITCHED NETWORK
TY1(.): ARRAY OF 3 CHARACTER IDENTIFIERS FOR
        FIRST SWITCHED NETWORK
TY2(.): ARRAY OF 3 CHARACTER IDENTIFIERS FOR
        SECOND SWITCHED NETWORK
TY3(.): ARRAY OF 3 CHARACTER IDENTIFIERS FOR
        THIRD SWITCHED NETWORK
SERVA1: 8 BIT (BYTE) REPRESENTATION OF USER
        DEFINED 'IN' STATE
SERVA2: 8 BIT (BYTE) REPRESENTATION OF USER
        DEFINED 'IN' STATE
```

The principal outputs NUU, NUL, NUD, (NETWi(...), NARYi, and SITEi(.) for i = 1, 2 and 3) are contained in files PRMOUT1, PRMOUT2 and PRMOUT3 (see Figure 5-2-2) where:

```
          NUU: NUMBER OF SITES IN THE FIRST SWITCHED
               NETWORK
          NUL: NUMBER OF SITES IN THE SECOND SWITCHED
               NETWORK
          NUD: NUMBER OF SITES IN THE THIRD SWITCHED
               NETWORK
    NETW1(.,.): PRIMITIVE CONNECTION MATRIX OF THE
               FIRST SWITCHED NETWORK WITH 2
               CHARACTER LINK CODES AS ELEMENTS
    NETW2(.,.): PRIMITIVE CONNECTION MATRIX OF THE
               SECOND SWITCHED NETWORK WITH 2
               CHARACTER LINK CODES AS ELEMENTS
    NETW3(.,.): PRIMITIVE CONNECTION MATRIX OF THE
               THIRD SWITCHED NETWORK WITH 2
               CHARACTER LINK CODES AS ELEMENTS
      NARY1(.): ARRAY OF 3 CHARACTER SITE SERIAL #S
               FOR FIRST SWITCHED NETWORK
      NARY2(.): ARRAY OF 3 CHARACTER SITE SERIAL #S
               FOR SECOND SWITCHED NETWORK
      NARY3(.): ARRAY OF 3 CHARACTER SITE SERIAL #S
               FOR THIRD SWITCHED NETWORK
      SITE1(.): ARRAY OF 8 CHARACTER SITE NAMES IN
               FIRST SWITCHED NETWORK
      SITE2(.): ARRAY OF 8 CHARACTER SITE NAMES IN
               SECOND SWITCHED NETWORK
      SITE3(.): ARRAY OF 8 CHARACTER SITE NAMES IN
               THIRD SWITCHED NETWORK
```

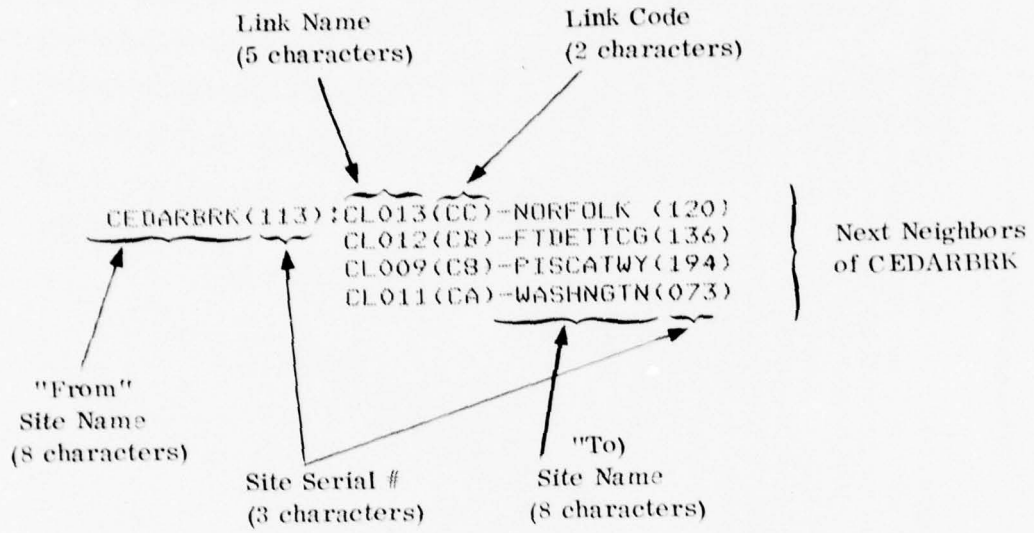
An example of a primitive connection matrix containing 32 sites is shown in Figure 3-13. The matrix is symmetrical about the diagonal, and its two-character elements are:

$$(\text{element})_{ij} = \begin{cases} \text{"01"} \ ; \quad i = j \\ \text{"00"; if no link exists between} \\ \text{"From" site i to "To" site j} \\ \text{"XY"; linkcode XY if a link exists between} \\ \text{"From" site i to "To" site j} \end{cases}$$

Row i of the matrix corresponds to "From" site i and the columns containing the link codes indicate its "To" sites. For example, "From" site 1 has sites 2, 11, 25, 27 as

Figure 3-13. An Example of Primitive Connection Matrix

"To sites. It is clear that the "next neighbor" information contained in the primitive connection matrix is not easily readable. Hence, the outputs of the program are sorted and made available in an easily readable form in file PRMOUT as follows:



In addition to the easily removable user constraints of the Circuit Tracer Program, this program has one important constraint; the size of the largest primitive connection matrix. This matrix is restricted to (150 x 150) elements. The restriction is again to save the core size of the program. Experience with the data base indicates that the largest network considered, AUTOVON, does not contain more than 100 sites.

3.2.1.3 Connectivity Search Program

This program, upon gathering the "next neighbor" information contained in a primitive connection matrix of the network, determines all "simple" paths from one specified site of the network to another in terms of alternating sequence of sites and links.

The program calls one external procedure containing the algorithm to determine all "simple" paths. This algorithm is based on a powerful, yet simple, algorithm.* The original algorithm was written in SNOBOL. The core of the algorithm is two loops which add or delete sites from the path based on: 1) the 'next neighbor" information, 2) the sites already contained on the built up path and, 3) the information regarding addition/deletion from the path to obtain all unique "simple" paths. The "modified" algorithm is a PL/I coded version of the original algorithm, with following modifications:

1. "Next neighbor" information is derived from the modified primitive connection matrix (see Figure 3-13) which can be nonsymmetric. This allows for "directional" links between the sites. This issue is of great importance in large networks containing a large number of sites, such as AUTOVON. It also permits a sense of direction to be introduced in the links connected with "gateway" sites, such as cableheads.
2. A trap is introduced at a proper location in the algorithm to take care of "sink" sites having no "next neighbors" (the row corresponding to this site in the primitive connection matrix contains no link codes). This trap enables the algorithm to recover from encounters with such sites and continue on its way. The program will print out the name of the "sink" site whenever it is encountered during the path buildup.

Figure 3-14 shows the functional input/output block diagram of this program. The inputs N, PCM(.,.), SITES(.), and SITEN(.) are contained in file PRMOUTi and NFF, NTT, PATHLTH, NUMC, and FROMTOS(.) are contained in file CNSEINi (i in PRMOUTi and CNSEINi refers to i^{th} the switched network) where:

---

*Reference: D. Kroft, "All Paths Through A Maze," Proc. of IEEE, January 1967, p.p. 88-90, (Unclassified).

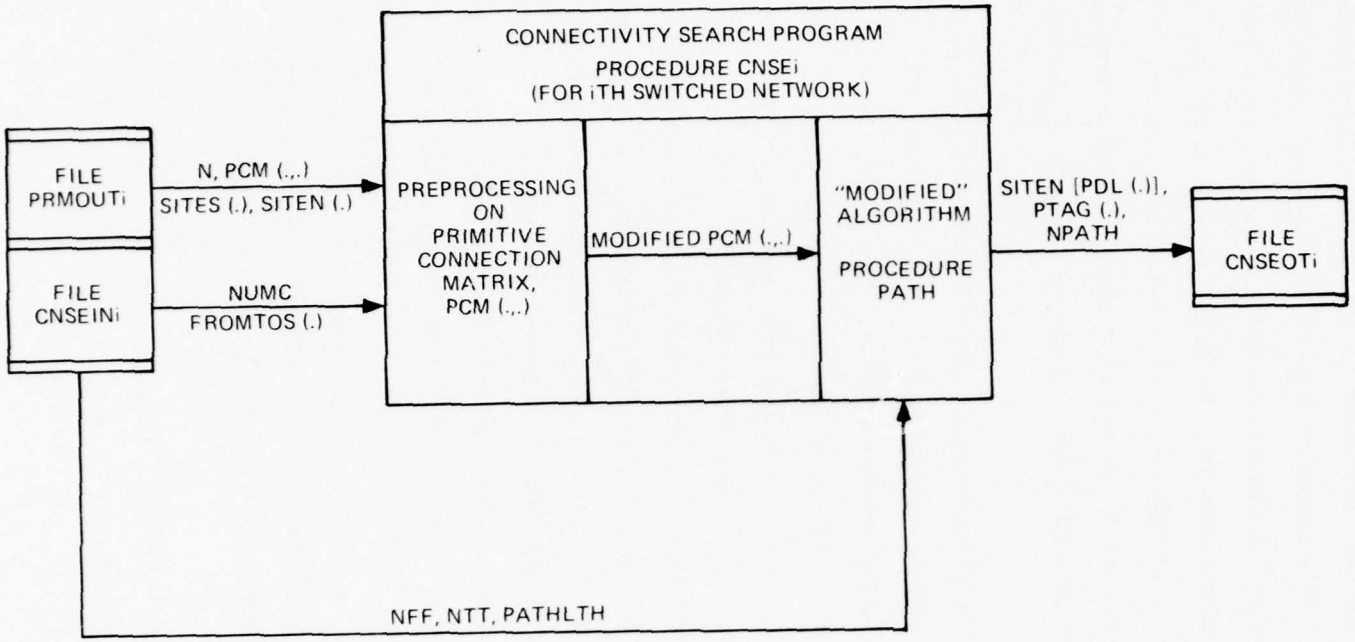


Figure 3-14. Functional Input/Output Block Diagram for Connectivity Search Program

3.2.2 Circuit Tracer

```
/*****************************************************************************/
/*                        MAIN PROCEDURE DECA                                */
/*****************************************************************************/

      /*  MULTIPLE CIRCUIT ANALYSER PROGRAM */
      /*                                    */
      /*  THIS PROCEDURE DETERMINES THE LINKS AND THE SITES WHICH
          CONSTITUTE A GIVEN CIRCUIT ALONG WITH THE IN/OUT
          FLAGS FOR CIRCUIT,LINK(S) AND SITES FOR ALL TEH
          ALPHANUMERICALLY SORTED CIRCUITS IN THE CIRCUIT
          MASTER FILE */
      /*                                    */
      /*  INPUTS-  (FILE),CKT(.),LCKTNO */

      /*           (FILE): THIS REFERS TO ALL USER DEFINED TOTAL
                           FILES INCLUDING;
                            MASTER FILES: RMCT,RMTR,RMLK,RMSI
                                          AND
                            VARIABLE FILES: RVCT,RVTI
                       CKT(.): ARRAY OF 9 CHARACTER CIRCUIT CCSD#S
                       LCKTNO: NUMBER OF CIRCUITS TO BE TRACED     */
      /*                                    */
      /*  OUTPUTS- LKCKT,SICKT,SERIES */

      /*           LKCKT: A STRING OF CHARACTERS CONTAINING CCSD#
                          AND CIRCUIT IN/OUT FLAG (F) AS HEADER
                          FOLLOWED BY THE NAMES OF THE LINKS
                          TRAVERSED ALONG WITH THEIR IN/OUT FLAGS
                          AND A SPECIAL CHARACTER, @ AS TRAILER
                          ( 605 CHARACTERS FOR MAXIMUM ALLOWED
                            99 LINKS ON THE CIRCUIT PATH )

                           EXAMPLE:

                           CCSD#FLINK1FLINK2F-----------------LINK99F@

                            (NOTE- FLAG,F IS ONE CHARACTER AND LINK1
                                   THROUGH LINK99 EACH CONTAINS 5
                                   CHARACTERS)

                          SICKT: A STRING OF CHARACTERS CONTAINING CCSD#
                                 AND CIRCUIT IN/OUT FLAG (F) AS HEADER
```

```
                    FOLLOWED BY THE NAMES OF THE SITES
                    TRAVERSED ALONG WITH THEIR IN/OUT FLAGS
                    AND A SPECIAL CHARACTER, @ AS TRAILER
                    ( 911 CHARACTERS FOR MAXIMUM ALLOWED
                      100 SITES ON THE CIRCUIT PATH )

                  EXAMPLE:

                  CCSD#FSITE1FSITE2F------------------SITE100F@

                    (NOTE- SITE1 THROUGH SITE100 EACH CONTAINS
                           8 CHARACTER SITE NAME)

                SERIES: A STRING OF CHARACTERS CONTAINING CCSD#
                        AND CIRCUIT IN/OUT FLAG (F) AS HEADER
                        FOLLOWED BY THE SITE SERIAL #S OF THE SITES
                        , THE LINK CODES AND THE IN/OUT FLAGS OF
                        THE LINKS TRAVERSED AND A SPECIAL
                        CHARACTER, @ AS TRAILER ( 608 CHARACTERS
                        FOR MAXIMUM ALLOWED 99 LINKS AND 100
                        SITES )

                  EXAMPLE:

                  CCSD#FSITENO1LINKCODE1F-----LINKCODE99FSITENO100@

                    (NOTE- SITENO1 THRCUGH SITENO100 EACH CONTAINS
                           3 CHARACTER SITE SERIAL #
                                  AND
                           LINKCODE1 THROUGH LINKCODE99 EACH
                           CONTAINS 2 CHARACTER LINK CODE)       */

/************************************************************/

                /*** NOTES ***/

   /*  AN EXIT IS TAKEN OUT OF THE PROCEDURE WITHOUT
       ERROR MESSAGES FOR TOTAL RETURNED STATUS¬=OK

       EXIT IS ALSO TAKEN OUT OF THE PROCEDURE WITH
       ERROR MESSAGES FOR ANY ONE OF THE FOLLOWING:

        1. INPUT CIRCUIT NOT FOUND IN MASTER FILE,
           RMCI

        2. IF THE TRUNK SEGMENT END SITES, AS GIVEN
           IN VARIABLE FILE,RVCT, ARE NOT FOUND IN
           VARIABLE FILE,RVTL ( THIS COULD BE CAUSED
           BY IMPROPER ORDER OF LINKS IN RVTL )

        3. IF THE TRUNK SEGMENT END SITES ARE THE SAME
```

```
     (SSFR,SSTO) CHAR(8),
     FISITE CHAR(8),

     (SRFR,SRTO) CHAR(3),
     SLSS CHAR(52);
DCL 1 MAST_CIRCUIT,
       2 MCTFLAG CHAR(1),
       2 MCTFRLC CHAR(8),
       2 MCTFFAC CHAR(3),
       2 MCTTOLC CHAR(8),
       2 MCTTFAC CHAR(3);
DCL MASTER_CIRCUIT CHAR(23) DEFINED MAST_CIRCUIT;
DCL 1 VARI_CTS,
       2 VCTSCONTROL CHAR(7),
       2 VCISSITE CHAR(11),
       2 VCTSSITT CHAR(8);
DCL VARIABLE_CTS CHAR(26) DEFINED VARI_CTS;
DCL 1 VARI_TLS,
       2 VTLSLINK CHAR(5),
       2 VTLSSITE CHAR(11),
       2 VTISSITT CHAR(8);
DCL VARIABLE_TLS CHAR(24) DEFINED VARI_TLS;
DCL 1 MAST_LINK,
       2 MLKFLAG CHAR(1),
       2 MLKCODE CHAR(2);
DCL MASTER_LINK CHAR(3) DEFINED MAST_LINK;
DCL 1 MAST_SITE,
       2 MSIFLAG CHAR(1),
       2 MSINUMB CHAR(3);
DCL MASTER_SITE CHAR(4) DEFINED MAST_SITE;
     /*                          */
     /* OPEN TOTAL FILES FOR READ ONLY ACCESS */
     /*                          */
FUNCT=QUESI;
TASKID='DECAM';
DBMOD='RFBTA1';
CALL DBIO(5,0);
     /*                          */
     /* OPEN FILES */
     /*                          */
FUNCT=OPENM;
STATUS=OK;
TFILES(1)='RMCT';
TFILES(2)='RVCT';
TFILES(3)='RMTR';
TFILES(4)='RVTL';
TFILES(5)='RMLK';
TFILES(6)='RMSI';
CALL DBIO(4,1);
IF (STATUS¬=OK) THEN GO TO EXIT;
     /*                          */
     /* RESET CIRCUIT MASTER FILE */
```

```
      /*                          */
FUNCT=RESTM;
CALL DBIO(3,0);
      /*                          */
      /* READ CIRCUIT MASTER FILE SERIALLY TO
         PULL OUT CCSD#S  */
      /*                          */
STATUS=OK;
LCKTNO=0;
CIRCUIT_READ:DO WHILE(STATUS¬=ENDF);
             FUNCT=SEQRM;
             STATUS=OK;
             FILEID='RMCT';
             ELMLIST='RMCTCTRLEND.';
             CALL DBIO(6,1);
             IF (STATUS¬=OK) THEN GO TO EXIT1;
             LCKTNO=LCKTNO+1;
             CKT(LCKTNO)=IOAREA;
             END CIRCUIT_READ;
EXIT1:;
      /*                          */
      /* SORT CCSD#S ALPHANUMERICALLY */
      /*                          */

CALL SORT(LCKTNO,CKT,CKT,TMP9);

PUT FILE(DECAOUT) LIST('TOTAL CIRCUITS TRACED=',LCKTNO);
DO I=1 TO LCKTNO;
  PUT FILE(DECAOUT) EDIT(CKT(I)) (COL(10),A(9));
END;
      /*                          */
      /* READ A RECORD CORRESPONDING TO THE SPECIFIED
         CIRCUIT FROM THE CIRCUIT MASTER FILE */
      /*                          */

   /*******     BEGIN MAIN LOOP      *******/

LPP:DO LOOP=1 TO LCKTNO;
    IJK=0;
PUT SKIP(1) FILE(DECAOUT);
     FUNCT=READM;
     STATUS=OK;
     FILEID='RMCT';
     CONTROL=CKT(LOOP);
     ELMLIST='RMCTFLAGRMCTFRLCRMCTFFACRMCTTCLCRMCTTFACEND.';
     CALL DBIO(7,1);
     IF (STATUS='MRNF') THEN
       DO;
         PUT SKIP FILE(DECAOUT)
             EDIT('** CIRCUIT=',CKT(LOOP),' NOT FOUND')
             (COL(2),A(11),A(9),A(10));
         GO TO DE9;
```

```
      END;
    ELSE;
    IF (STATUS¬=OK) THEN GO TO EXIT;
    MASTER_CIRCUIT=IOAREA;
    LKCKT,SICKT,SERIES=CKT(LOOP)||MCTFLAG; /* CCSD# AND I/O FLAG
                                              ADDED */
        /*                          */
        /* LINKAGE PATH TO CIRCUIT-TRUNK-SITE VARIABLE FILE-
           TO DETERMINE ALL TRUNKS  CONTAINING THE
           CHOSEN CIRCUIT */
        /*                          */
    FLGG=0;
    TMPSFXX,TMPSTXX=(8)'0';
    RR(1)='LKCT';

       /****    BEGIN LOOP FOR A CIRCUIT    ****/

CIRCT:DO WHILE(RR(1)¬=ENDP); /* LOOP FOR FINDING ALL TRUNKS
                                IN THE GIVEN CIRCUIT */
    LL,LLL=0;
    DO I=1 TO 50;
      TMPFRLC(I),TMPTOLC(I),TMPSXX(I)=(8)'0';
      IF (I<=49) THEN TMPLINK(I),TMPLXX(I)=(5)'0';
    END;
    TMPRF=TMPSFXX;
    TMPRT=TMPSTXX;
    REFER=RR(1);
    FUNCT=READV;
    STATUS=OK;
    FILEID='RVCT';
    LKPATH='RMCTLKCT';
    CONTROL=CKT(LOOP);       /* SAME AS BEFORE */
    ELMLIST='RVCTRMTRRVCTRMSIRVCTTOICEND.';
    CALL DBIO(9,1);
    IF (STATUS¬=OK) THEN GO TO EXIT;
    VARIABLE_CTS=IOAREA;
    TMPSFXX=SUBSTR(VCTSSITE,1,8);
    TMPSTXX=VCTSSITT;
        /*                          */
        /* SWITCH THE END SITES OF THE TRUNK
           SEGMENT (IF NEEDED) */
        /*                          */
    IF (FLGG=0) THEN
      DO;
        FLGG=1;
        GO TO NO_CHECK;
      END;
    ELSE;
    IF (TMPRT=TMPSFXX) THEN GO TO CHANGE_EXIT;
    IF (TMPRT=TMPSTXX) THEN
      DO;
        TMPR8=TMPSFXX;
```

```
            TMPSPXX=TMPSTXX;
            TMPSIXX=TMPRR;
            GO TO CHANGE_EXIT;
          END;
        ELSE;
        CHANGE_EXIT:;
        NO_CHECK:;
        RR(1)=REFER;
        IF (RR(1)=ENDP) THEN GO TO DES;  /* ALL TRUNKS FINISHED */
        IF (IJK=0) THEN
          DO;

            CALL FLAGS(MCTFIAG,TMP5);

            CKTX=CKT(LOCP)||'-'||TMP5;
            PUT SKIP(2) FILE(DECAOUT) EDIT('CCSD#=',CKTX)
                                      (COL(23),A(6),A(15));
            PUT SKIP(1) FILE(DECAOUT);
            IJK=1;
          END;
        ELSE;
PUT SKIP FILE(DECAOUT) EDIT('TRUNK=',VCTSCONTROL,'  ',
                        TMPSPXX,'  ',TMPSTXX)
                        (COL(18),A(6),A(7),(2)(A(2),A(8)));
            /*                          */
            /* LINKAGE PATH TO TRUNK-LINK-SITE VARIABLE FILE-
               TO DETERMINE THE LINK(S) IN THE CHOSEN TRUNK
               TO WHICH THE SPECIFIED DEDICATED CIRCUIT BELONGS
               AND THE SITES TRAVERSED */
            /*                          */
        RR(2)='LKTL';

            /****   BEGIN LOOP FOR A TRUNK   ****/

TRUNK:DO WHILE(RR(2)¬=ENDP);        /* LOOP FOR FINDING ALL SITES AND
                                       LINKS IN A GIVEN TRUNK */
        REFER=RR(2);
        FUNCT=READV;
        STATUS=OK;
        FILEID='RVTL';
        LKPATH='RMTRLKTL';
        CONTROL=VCTSCONTROL;
        ELMLIST='RVTLRMLKRVTLRMSIRVTLTOLCEND.';
        CALL DBIO(9,1);
        IF (STATUS¬=OK) THEN GO TO EXIT;
        VARIABLE_TLS=IOAREA;
            /*                          */
            /* CREATE ARRAYS OF FROM AND TO SITES AND LINKS
               IN THE CHOSEN TRUNK */
            /*                          */
        LL=LL+1;
        TMPFRLC(LL)=SUBSTR(VTLSSITE,1,8);
```

```
TMPTOLC(LL)=VTLSSITT;
TMPLINK(LL)=VTLSLINK;
IF (LL>=2) THEN
  DO;
    IF((TMPFRLC(LL)=TMPFRLC(LL-1)) &
       (TMPTOLC(LL)=TMPTOLC(LL-1))) THEN
      DO;
        TMPFRLC(LL),TMPTOLC(LL)=(8)'0';
        TMPLINK(LL)=(5)'0';
        LL=LL-1;
      END;
    ELSE;
  END;
ELSE;
RE(2)=REFER;
END TRUNK;

   /****    END LOOP FOR A TRUNK    ****/

     /*                              */
     /* SORT ABOVE ARRAYS TO DETERMINE THE LINKS AND
        SITES TRAVERSED IN THE CHOSEN TRUNK BY THE
        GIVEN CIRCUIT */
     /*                              */
  DO;
    II,JJ,KK,K=0;
    LLL=LL;
     /*                              */
     /* CREATE ARRAYS OF SITES AND LINKS IN THE TRUNK
        WHICH ARE COMMON TO BOTH THE CIRCUIT AND THE TRUNK */
     /*                              */
    TMPFRLC(LLL+1)=TMPTOLC(LLL);
    DE1:DO I=1 TO LLL+1;
        IF (TMPFRLC(I)=TMPSFXX) THEN JJ=I;
        IF (TMPFRLC(I)=TMPSTXX) THEN KK=I;
        END DE1;
    IF ((JJ=0)|(KK=0)) THEN
      DO;
        PUT SKIP FILE(DECAOUT)
            EDIT('** ERROR TRUNK=',VCTSCONTROL,' CIRCUIT=',
                 CKT(LOOP))
            (COL(2),A(15),A(7),(2)A(9));
        GO TO DE9;
      END;
    ELSE;
    IF (JJ=KK) THEN
      DO;
        PUT SKIP FILE(DECAOUT) EDIT('** ERROR=',
                VCTSCONTROL) (COL(2),A(9),A(7));
        GO TO DE7;
      END;
    ELSE;
```

```
    IF (KK>JJ)THEN
     /*                          */
     /* TRUNK AND CIRCUIT DIRECTION IS IDENTICAL */
     /*                          */
      DO;
        II=1;
        LLL=KK-JJ;
        DE2:DO I=JJ TO KK;
            K=K+1;
            TMPSXX(K)=TMPFRLC(I);
            IF (I<KK) THEN TMPLXX(K)=TMPLINK(I);
            END DE2;
        GO TO DE4;
      END;
    ELSE;
    IF (JJ>KK) THEN
     /*                          */
     /* TRUNK AND CIRCUIT DIRECTIONS ARE NOT IDENTICAL */
     /*                          */
      DO;
        II=2;
        LLL=JJ-KK;
        DE3:DO I=JJ TO KK BY -1;
            K=K+1;
            TMPSXX(K)=TMPFRLC(I);
            IF (I<JJ) THEN TMPLXX(K-1)=TMPLINK(I);
            END DE3;
        GO TO DE4;
      END;
    ELSE;
    DE4:;
    FISITE=TMPSXX(K);  /* FINAL SITE SAVED */
     /*                          */
     /* READ RECORDS IN THE SITE AND LINK MASTER FILES
        TO DETERMINE IN/OUT FLAGS, LINK CODES AND SITE
        SERIAL #S */
     /*                          */
    DE5:DO J=1 TO LLL;
        CALL FIND(TMPSXX(J),TMPIXX(J));
        END DE5;
  END;
  DE7:;
END CIRCT;

   /*****     END LOOP FOR A CIRCUIT     *****/

DE8:;
     /*                          */
     /* ADD FINAL SITE */
     /*                          */
FUNCT=READM;
STATUS=OK;
```

```
      FILEID='RMSI';
      CONTROL=FISITE;
      IF (CONTROL=(8)'0') THEN
        DO;
          PUT SKIP FILE(DECAOUT)
              EDIT('** ERROR IN FINAL SITE FOR CIRCUIT=',CKT(LOOP))
              (COL(2),A(35),A(9));
          GO TO DE9;
        END;
      ELSE;
      ELMLIST='RMSIFLAGRMSINUMBEND.';
      CALL DBIO(7,1);
      IF (STATUS¬=OK) THEN GO TO EXIT;
      MASTER_SITE=IOAREA;
      SICKT=SICKT||FISITE||MSIFLAG;
      SERIES=SERIES||MSINUMB;
PUT SKIP(1) FILE(DECAOUT);
      LKCKT=LKCKT||'@';
      SICKT=SICKT||'@';
      SERIES=SERIES||'@';
           /*                          */
           /* SORT CIRCUIT DATA FOR OUTPUT FORMAT */
           /*                          */
       II=LENGTH(LKCKT)-11;
       II=BIN(II/6,15,0);
       DO KK=1 TO II;
        I=11+6*(KK-1);
        J=11+9*(KK-1);
        K=11+6*(KK-1);
        L=14+6*(KK-1);
        LFR=SUBSTR(LKCKT,I,6);
        SFR=SUBSTR(SICKT,J,9);
        LLFR=SUBSTR(LFR,1,5);
        LKFL=SUBSTR(LFR,6,1);

        CALL FLAGS(LKFL,LKFL5);

        SSFR=SUBSTR(SFR,1,8);
        FRFL=SUBSTR(SFR,9,1);

        CALL FLAGS(FRFL,FRFL5);

        SRFR=SUBSTR(SERIES,K,3);
        LXT=SUBSTR(SERIES,L,2);
        J=11+9*KK;
        K=11+6*KK;
        STO=SUBSTR(SICKT,J,9);
        SSTO=SUBSTR(STO,1,8);
        TOFL=SUBSTR(STO,9,1);

        CALL FLAGS(TOFL,TOFL5);
```

```
      SRTO=SUBSTR(SERIES,K,3);
      SLSS=SSFR||'('||SRFR||')'||FRFL5||'-'||LLFR||'('||LXT||')'
           ||LKFL5||'-'||SSTO||'('||SFTO||')'||TOFL5;
      PUT SKIP FILE(DECAOUT) EDIT(SLSS) (COL(10),A(52));
     END;
   DE9:;
   END LFP;

      /*******      END MAIN LOOP      *******/

EXIT:;
         /*                          */
         /* CLOSE FILES */
         /*                          */
    CLOSE FILE(DECAOUT);
    FUNCT=CLOSM;
    STATUS=OK;
    CALL DBIO(4,0);
         /*                          */
         /* SIGN OFF THE TASK */
         /*                          */
    FUNCT=DEQUE;
    TASKID='DECAM';
    CALL DBIO(4,0);


/*****************************************************************/
/*     BEGIN INTERNAL PROCEDURES OF PROCEDURE DECA               */
/*****************************************************************/


         /* PROCEDURE FIND */


   FIND:PROC(FISITE,FILINK);
        DCL FISITE CHAR(8),FILINK CHAR(5);
        FUNCT=READM;
        STATUS=OK;
        FILEID='RMSI';
        CONTROL=FISITE;
        IF (CONTROL=(8)'0') THEN
          DO;
            PUT SKIP FILE(DECAOUT) EDIT('** ERROR TRUNK=',
                VCTSCONTROL) (COL(2),A(15),A(7));
            GO TO EXIT;
          END;
        ELSE;
        ELMLIST='RMSIFLAGRMSINUMBEND.';
        CALL DBIO(7,1);
        IF (STATUS^=OK) THEN GO TO EXIT;
        MASTER_SITE=IOAREA;
        SICKT=SICKT||FISITE||MSIFLAG; /* SITE ADDED */
```

```
        STATUS=OK;
        FILEID='RMLK';
        CONTROL=FILINK;
        IF (CONTROL=(5)'0') THEN
          DO;
            PUT SKIP FILE(DECAOUT) EDIT('** ERROR TRUNK=',
                VCTSCONTROL) (COL(2),A(15),A(7));
            GO TO EXIT;
          END;
        ELSE;
        ELMLIST='RMLKFLAGRMLKCODEEND.';
        CALL DBIO(7,1);
        IF (STATUS¬=OK) THEN GO TO EXIT;
        MASTER_LINK=IOAREA;
        LKCKT=LKCKT||FILINK||MLKFLAG;  /* LINK ADDED */
        SERIES=SERIES||MSINUMB||MLKCODE||MLKFLAG; /* LINK,SITE AND
                                                     LINK FLAG ADDED */
        END FIND;


         /* PROCEDURE FLAGS */


/* NOTE- ALL THE CHARACTERS USED IN CHAR5 EXCEPT CHARACTER "A" HAVE
         THE FOLLOWING MEANINGS IN THEIR RESPECTIVE CHARACTER
         POSITIONS

          A='00000000'B: "IN" (AVAILABLE) STATE
          S='00010000'B: SATELLITE LINK OUTAGE
          M='00001000'B: MHD
          J='00000100'B: JAMMING
          B='00000010'B: BLAST
          E='00000001'B: EMP                              */


  FLAGS:PROC(CHAR1,CHAR5);
        DCL CHAR1 CHAR(1),CHAR5 CHAR(5),
            BIT8 BIT(8);
        CHAR5='SMJBE';
        BIT8=UNSPEC(CHAR1);
        IF (BIT8=('00000000'B)) THEN
          DO;
            CHAR5='AAAAA';
            GO TO XIT;
          END;
        ELSE;
        DO I=4 TO 8;
         IF (SUBSTR(BIT8,I,1)='0'B) THEN SUBSTR(CHAR5,I-3,1)='A';
        END;
        XIT:;
        END FLAGS;
```

```
          /* PROCEDURE SORT */

SORT:PROC(N,XIN,XOUT,TMP);
     DCL (XIN(*),XOUT(*),TMP) CHAR(*),
         (M,N,I,J) BIN FIXED(15,0);
     DO I=1 TO N;
       XOUT(I)=XIN(I);
     END;
        /*                          */
        /* MODIFIED SHELL SORT LOGIC */
        /*                          */
     M=1;
     S1:DO I=1 TO N WHILE(M<N);
        M=M+M;
        END S1;
     M=M-1;
     BACK:;
          M=M/2;
     S2:DO J=1 TO N-M;
        TMP=XOUT(J+M);
        S3:DO I=J TO 1 BY -M;
           IF (XOUT(I)<=TMP) THEN GO TO NEXT;
           XOUT(I+M)=XOUT(I);
           END S3;
        NEXT:;
             XOUT(I+M)=TMP;
        END S2;
     IF (M>1) THEN GO TO BACK;
     END SORT;


/*************************************************************/
/*      END INTERNAL PROCEDURES OF PROCEDURE DECA            */
/*************************************************************/


    END DECA;


/*
//LKED.SYSLMOD DD DSN=TOTAL.DBLIB(DECAM),DISP=SHR
//LKED.DBIC DD DSN=SYS9.TOTAL.LINKLIB4,DISP=SHR
//LKED.SYSIN DD *
     INCLUDE DBIC(DBIC)
/*
//  EXEC TOTAL4,PGNAME=DECAM,REGION=349K
//DECAOUT DD DSN=M1765.DECAOUT.DATA,DISP=OLD
//RMCT DD DSN=UH01C.PI119.RMCT,DISP=SHR
```

```
//RMSI DD DSN=UH010.PI119.RMSI,DISP=SHR
//RMTR DD DSN=UH010.PI119.RMTR,DISP=SHR
//RMLK DD DSN=UH010.PI119.RMLK,DISP=SHR
//RVTL DD DSN=UH010.PI119.RVTL,DISP=SHR
//RVCT DD DSN=UH010.PI119.RVCT,DISP=SHR
//SYSPRINT DD SYSOUT=(A,U)
//
```

3.2.3 Primitive Matrix Generation Program

```
/*****************************************************************/
/*                    MAIN PROCEDURE PRIM                        */
/*****************************************************************/


      /* PRIMITIVE CONNECTION MATRIX PROGRAM */
      /*                                     */
      /* THIS PROCEDURE, UPON GATHERING INFORMATION ABOUT
         THE THREE SPECIFIED SWITCHED NETWORKS FROM THE
         FILE, COMPUTES THE PRIMITIVE CONNECTION MATRICES
         FOR THE NETWORKS IN TERMS OF LINK CODES AS ELEMENTS */
      /*                                     */
      /* INPUTS- (FILE),J,K,L,TY1(.),TY2(.),TY3(.),
                 SERVA1,SERVA2 */

      /*         (FILE): THIS REFERS TO ALL USER DEFINED TOTAL
                         FILES INCLUDING;
                          MASTER FILES: RMCT,RMTR,RMLK,RMSI
                                        AND
                          VARIABLE FILES: RVCT,RVTL
                      J: NUMBER OF IDENTIFIERS FOR FIRST
                         SWITCHED NETWORK
                      K: NUMBER OF IDENTIFIERS FOR SECOND
                         SWITCHED NETWORK
                      L: NUMBER OF IDENTIFIERS FOR THIRD
                         SWITCHED NETWORK
                 TY1(.): ARRAY OF 3 CHARACTER IDENTIFIERS FOR
                         FIRST SWITCHED NETWORK
                 TY2(.): ARRAY OF 3 CHARACTER IDENTIFIERS FOR
                         SECOND SWITCHED NETWORK
                 TY3(.): ARRAY OF 3 CHARACTER IDENTIFIERS FOR
                         THIRD SWITCHED NETWORK
                 SERVA1: 8 BIT (BYTE) REPRESENTATION OF USER
                         DEFINED "IN" STATE
                 SERVA2: 8 BIT (BYTE) REPRESENTATION OF USER
                         DEFINED "IN" STATE   */
      /*                                     */
      /* OUTPUTS- NUU,NUL,NUD
                  NETW1(.,.),NETW2(.,.),NETW3(.,.)
                  NARY1(.),NARY2(.),NARY3(.)
                  SITE1(.),SITE2(.),SITE3(.)        */
```

```
/*                  NUU: NUMBER OF SITES IN THE FIRST SWITCHED
                         NETWORK
                    NUL: NUMBER OF SITES IN THE SECOND SWITCHED
                         NETWORK
                    NUD: NUMBER OF SITES IN THE THIRD SWITCHED
                         NETWORK
               NETW1(.,.): PRIMITIVE CONNECTION MATRIX OF THE
                         FIRST SWITCHED NETWORK WITH 2
                         CHARACTER LINK CODES AS ELEMENTS
               NETW2(.,.): PRIMITIVE CONNECTION MATRIX OF THE
                         SECOND SWITCHED NETWORK WITH 2
                         CHARACTER LINK CODES AS ELEMENTS
               NETW3(.,.): PRIMITIVE CONNECTION MATRIX OF THE
                         THIRD SWITCHED NETWORK WITH 2
                         CHARACTER LINK CODES AS ELEMENTS
                 NARY1(.): ARRAY OF 3 CHARACTER SITE SERIAL #S
                         FOR FIRST SWITCHED NETWORK
                 NARY2(.): ARRAY OF 3 CHARACTER SITE SERIAL #S
                         FOR SECOND SWITCHED NETWORK
                 NARY3(.): ARRAY OF 3 CHARACTER SITE SERIAL #S
                         FOR THIRD SWITCHED NETWORK
                  SITE1(.): ARRAY OF 8 CHARACTER SITE NAMES IN
                         FIRST SWITCHED NETWORK
                  SITE2(.): ARRAY OF 8 CHARACTER SITE NAMES IN
                         SECOND SWITCHED NETWORK
                  SITE3(.): ARRAY OF 8 CHARACTER SITE NAMES IN
                         THIRD SWITCHED NETWORK                */
/*                      */
/* AUXULIARY OUTPUTS- SITE11(.),SITE22(.),SITE33(.)
                      SITENMI(.),SITENMS(.),SITESNI(.),NUSI
                      LINKNM(.),LINKCO(.),LINKCOS(.),NULK   */

/*              SITE11(.): SAME AS SITE1(.) BUT ALPHANUMERICALLY
                         SORTED
                SITE22(.): SAME AS SITE2(.) BUT ALPHANUMERICALLY
                         SORTED
                SITE33(.): SAME AS SITE3(.) BUT ALPHANUMERICALLY
                         SORTED
               SITENMI(.): ARRAY OF 8 CHARACTER SITE NAMES OF ALL
                         THE SITES AS THEY EXIST IN SITE MASTER
                         FILE,RMSI
               SITENMS(.): SAME AS ABOVE BUT ALPHANUMERICALLY
                         SORTED
               SITESNI(.): ARRAY OF 3 CHARACTER SITE SERIAL #S
                         AS THEY EXIST IN THE SITE MASTER
                         FILE,RMSI
                     NUSI: NUMBER OF SITES IN SITE MASTER FILE
                LINKNM(.): ARRAY OF 5 CHARACTER LINK NAMES OF ALL
                         THE LINKS IN THE LINK MASTER FILE,RMLK
                LINKCO(.): ARRAY OF 2 CHARACTER LINK CODES OF ALL
                         THE LINKS IN THE LINK MASTER FILE,RMLK
```

```
                    LINKCOS(.): SAME AS ABOVE BUT ALPHANUMERICALLY
                                SORTED
                          NULK: NUMBER OF LINKS IN THE LINK MASTER
                                FILE                    */

PRIM:PROC OPTIONS(MAIN);

 /*************************************************************/
       /* PROCEDURES CALLED BY PROCEDURE PRIM */
 /*************************************************************/


       /* EXTERNAL PROCEDURES */


     DCL DECA EXT ENTRY(CHAR(9),CHAR(605) VAR,CHAR(911) VAR,
                        CHAR(608) VAR,BIN FIXED(15,0));
     DCL NEA EXT ENTRY(BIN FIXED(15,0),CHAR(608) VAR,BIN
                       FIXED(15,0),,);


       /* INTERNAL PROCEDURES */


 /*  DCL CORELTSI ENTRY(BIN FIXED(15,0));   */
 /*  DCL COREITSI ENTRY(BIN FIXED(15,0));   */
 /*  DCL CORELTSIM ENTRY(BIN FIXED(15,0),,);   */
 /*  DCL SORT ENTRY(BIN FIXED(15,0),,,);   */
 /*  DCL PRNT ENTRY(BIN FIXED(15,0),,,,);   */


       /* DECLARATIONS FOR THE FILES AND THE
          IDENTIFIERS OF PROCEDURE PRIM */


     DCL PRIMIN INPUT FILE STREAM,
         PRMOUT OUTPUT FILE PRINT STREAM,
         PRMOUT1 OUTPUT FILE STREAM,
         PRMOUT2 OUTPUT FILE STREAM,
         PRMOUT3 OUTPUT FILE STREAM;

 /*      FILE PRIMIN CONTAINS 1. THE NUMBER AND THE DISCRIPTIONS
         OF IDENTIFIERS AND 2. THE "IN" STATE BYTES
         FILE PRMOUT CONTAINS 1. ALL THE CIRCUITS TRACED 2. THE
         ALPHANUMERICALLY SORTED SITE NAMES VS SITE SERIAL NO
         AND LINK CODE VS LINK NAMES TABLES AND 3. THE NEXT
         NEIGHBOR INFORMATION FOR THE THREE SWITCHED NETWORKS
         UNDER CONSIDERATION
         FILE PRMOUT1 CONTAINS THE PRIMITIVE CONNECTION MATRIX OF
         THE FIRST SWITCHED NETWORK AND A TABLE OF SITE NAMES VS
         SITE SERIAL NOS IN THE NETWORK
```

```
     FILE PRMOUT2 CONTAINS INFORMATION IDENTICAL AS IN FILE
     PRMOUT1 BUT FOR SECOND SWITCHED NETWORK
     FILE PRMOUT3 CONTAINS INFORMATION IDENTICAL AS IN FILE
     PRMOUT1 BUT FOR THIRD SWITCHED NETWORK                    */

%INCLUDE EXTDCL;
DCL (NUU,NUD,NUL,LINKS) BIN FIXED(15,0),
    (I,J,K,L,II,III,IFL,NULK,NUSI,RFLAG) BIN FIXED(15,0),
    SERVY CHAR(1) INIT('0'),
    (SERVA1,SERVA2,SERVC) BIT(8) INIT('00000000'B) EXT;
DCL CKT CHAR(9),
    LKCKT CHAR(605) VAR,
    SICKT CHAR(911) VAR,
    SERIES CHAR(608) VAR;
DCL YY CHAR(3);
DCL NETW1(150,150) CHAR(2) INIT((22500)(1)'00'),
    (NETW2(50,50),NETW3(50,50)) CHAR(2) INIT((2500)(1)'00'),
    NARY1(150) CHAR(3) INIT((150)(1)'000'),
    (NARY2(50),NARY3(50)) CHAR(3) INIT((50)(1)'000');
DCL (SITE1(150),SITE11(150)) CHAR(8) INIT((150)(1)'00000000'),
    (SITE2(50),SITE22(50),SITE3(50),SITE33(50)) CHAR(8)
                                    INIT((50)(1)'00000000'),
    LINKNM(400) CHAR(5) INIT((400)(1)'00000'),
    TMP2 CHAR(2) INIT('00'),
    (LINKCO(400),LINKCOS(400)) CHAR(2) INIT((400)(1)'00');
                 /* MAX 400 LINKS ALLOWED IN
                    ABOVE TWO ARRAYS */
DCL (SITENMI(400),SITENMS(400)) CHAR(8) INIT((400)(1)'00000000'),
    TMP8 CHAR(8) INIT('00000000'),
    SITESNI(400) CHAR(3) INIT((400)(1)'000');
                 /* MAX 400 SITES ALLOWED IN
                    ABOVE TWO ARRAYS */
     /*                          */
     /* OPEN TOTAL FILES FOR READ ONLY ACCESS */
     /*                          */
FUNCT=QUESI;
TASKID='PRIMS';
DBMOD='REBTA1';
CALL DBIC(5,);
     /*                          */
     /* READ INPUT INFORMATION ABOUT THE
        THREE SWITCHED NETWORKS */
     /*                          */
    GET FILE(PRIMIN) LIST(J,K,L) COPY(PRMOUT);

/*********************************************************/
     /* BEGIN BLOCK WITHIN PROCEDURE PRIM */
/*********************************************************/

     SCOPE_JKL_FOR_TYS:BEGIN;
```

```
     DCL TY1(J) CHAR(3),
         TY2(K) CHAR(3),
         TY3(L) CHAR(3);
     GET FILE(PRIMIN) EDIT((TY1(I) DO I=1 TO J))
                           (COL(1),(J)A(3)) COPY(PRMOUT);
     GET FILE(PRIMIN) EDIT((TY2(I) DO I=1 TO K))
                           (COL(1),(K)A(3)) COPY(PRMOUT);
     GET FILE(PRIMIN) EDIT((TY3(I) DO I=1 TO L))
                           (COL(1),(L)A(3)) COPY(PRMOUT);
     GET FILE(PRIMIN) EDIT(SERVA1) (COL(1),B(8))
                                    COPY(PRMOUT);
     GET FILE(PRIMIN) EDIT(SERVA2) (COL(1),B(8))
                                    COPY(PRMOUT);
      /*                           */
FUNCT=OPENM;
STATUS=OK;
TFILES(1)='RMCT';
TFILES(2)='RMTR';
TFILES(3)='RMLK';
TFILES(4)='RMSI';
TFILES(5)='RVCT';
TFILES(6)='RVTL';
CALL DBIO(4,1);
IF (STATUS¬=OK) THEN GO TO EXIT;
      /*                           */
      /* RESET CIRCUIT-MASTER FILE */
      /*                           */
FUNCT=RESTM;
CALL DBIO(3,0);
      /*                           */
      /* INITIALISATION */
      /*                           */
II,III=0;
NUU,NUD,NUL=0;
DO I=1 TO 150;
 NETW1(I,I)='01';
 IF (I<=50) THEN NETW2(I,I),NETW3(I,I)='01';
END;
      /*                           */
      /* SERIAL READ OF CIRCUIT MASTER FILE TO DETERMINE
         ALL CIRCUITS WITHIN IT */
      /*                           */
STATUS=OK;


CNTL:DO WHILE(STATUS¬=ENDP);
      FUNCT=SEQRM;
      STATUS=OK;
      FILEID='RMCT';
      ELMLIST='RMCTCTRLEND.';
      CALL DBIO(6,1);
```

```
          IF (STATUS¬=OK) THEN GO TO LEXIT;
          CKT=IOAREA;
          /*                              */
          /* SET FLAG TO 1 OR 2 OR 3 TO DETECT IF THE CIRCUIT
             BELONGS TO ANY ONE OF THE THREE SWITCHED NETWORKS */
          /*                              */
          YY=SUBSTR(CKT,2,3);
          IFL=0;
          DO I=1 TO J;
           IF (YY=TY1(I)) THEN
             DO;
               IFL=1;
               GO TO FLAG_IS_SET;
             END;
           ELSE;
          END;
          DO I=1 TO K;
           IF (YY=TY2(I)) THEN
             DO;
               IFL=2;
               GO TO FLAG_IS_SET;
             END;
           ELSE;
          END;
          DO I=1 TO L;
           IF (YY=TY3(I)) THEN
             DO;
               IFL=3;
               GO TO FLAG_IS_SET;
             END;
           ELSE;
          END;
          FLAG_IS_SET:;
          IF (IFL=1|IFL=2|IFL=3) THEN
          /*                              */
          /* CIRCUIT ANALYSER IS NEEDED TO TRACE THE PATH OF
             THE CIRCUIT THAT BELONGS TO ONE OF THE SPECIFIED
             SWITCHED NETWORKS */
          /*                              */
           DO;

            CALL DECA(CKT,LKCKT,SICKT,SERIES,RFLAG);

            IF (RFLAG=1) THEN GO TO CMIT;  /* OMIT CIRCUIT FOR
                                              FILE ERROR */
            SERVX=SUBSTR(SERIES,10,1);
            SERVC=UNSPEC(SERVX);
PUT SKIP FILE(PRMOUT) LIST('SERVC=',SERVC);
             II=LENGTH(SERIES)-14;
             LINKS=BIN(II/6,15,0);
          /*                              */
          /* FILL IN PRIMITIVE CONNECTION MATRICES
```

```
   WITH LINK CODES AS ELEMENTS */
/*                          */

    IF(IFL=1) THEN CALL NEA(LINKS,SERIES,NUU,NARY1,NETW1);

    IF(IFL=2) THEN CALL NEA(LINKS,SERIES,NUL,NARY2,NETW2);

    IF(IFL=3) THEN CALL NEA(LINKS,SERIES,NUD,NARY3,NETW3);

 END;
ELSE;
OMIT:;
END CNTL;


LEXIT:;

/*************************************************************/
  /* END OF BEGIN BLOCK WITHIN PROCEDURE PRIM */
/*************************************************************/


END SCOPE_JKL_FOR_TYS;

/*                          */
/* PRINT PRIMITIVE CONNECTION MATRICES OF
   THREE SWITCHED NETWORKS */
/*                          */
   PUT FILE(PRMOUT1) EDIT(NUU) (COL(1),F(3,0));
   DO I=1 TO NUU;
     PUT FILE(PRMOUT1) EDIT((NETW1(I,J) DO J=1 TO NUU))
                       (COL(1),(NUU)A(2));
   END;
   PUT FILE(PRMOUT2) EDIT(NUL) (COL(1),F(3,0));
   DO I=1 TO NUL;
     PUT FILE(PRMOUT2) EDIT((NETW2(I,J) DO J=1 TO NUL))
                       (COL(1),(NUL)A(2));
   END;
   PUT FILE(PRMOUT3) EDIT(NUD) (COL(1),F(3,0));
   DO I=1 TO NUD;
     PUT FILE(PRMOUT3) EDIT((NETW3(I,J) DO J=1 TO NUD))
                       (COL(1),(NUD)A(2));
   END;
/*                          */
/* CORRELATE SITE NAMES WITH SITE SERIAL #S */
/* IN MASTER FILE */
/*                          */

   CALL CORELTSI(NUSI);

/*                          */
```

```
/* CORRELATE LINK NAMES WITH LINK CODES */
/* IN MASTER FILE */
/*                    */

   CALL CORELTIK(NULK);

/*                    */
/* PRINT SITE NAMES VS SITE SERIAL NOS. AND
      LINK CODES VS LINK NAMES TABLES
      AFTER SORTINGS */
/*                    */
   PUT PAGE FILE(PRMOUT);
   PUT SKIP(2) FILE(PRMOUT);
   PUT SKIP FILE(PRMOUT)
       EDIT('SITE NAME--SITE SERIAL NO')
       (COL(40),A(25));
   PUT SKIP(1) FILE(PRMOUT);

   CALL SORT(NUSI,SITENMI,SITENMS,TMP8);

     DO I=1 TO NUSI;
       DO J=1 TO NUSI;
         IF (SITENMS(I)=SITENMI(J)) THEN
         PUT FILE(PRMOUT) EDIT(SITENMS(I),
         SITESNI(J)) (COL(41),A(8),X(8),A(3));
       END;
     END;

   PUT PAGE FILE(PRMOUT);
   PUT SKIP(2) FILE(PRMOUT);
   PUT SKIP FILE(PRMOUT)
       EDIT('LINK CODE--LINK NAME')
       (COL(40),A(20));
   PUT SKIP(1) FILE(PRMOUT);

   CALL SORT(NULK,LINKCO,LINKCOS,TMP2);

     DO I=1 TO NULK;
       DO J=1 TO NULK;
         IF (LINKCOS(I)=LINKCO(J)) THEN
         PUT FILE(PRMOUT) EDIT(LINKCOS(I),
         LINKNM(J)) (COL(43),A(2),X(8),A(5));
       END;
     END;

/*                    */
/* CORRELATE SITE NAMES WITH SITE SERIAL #S IN
   THE THREE SWITCHED NETWORKS */
/*                    */

   CALL CORELTSIM(NUU,NARY1,SITE1);
```

```
FUNCT=READM;
STATUS=OK;
```



```
      CALL CORELTSIM(NUL,NARY2,SITE2);

      CALL CORELTSIM(NUD,NARY3,SITE3);

   /*                          */
   /* PRINT SITE NAMES VS SITE SERIAL #S IN THE
      THREE SWITCHED NETWORKS */
   /*                          */
      DO I=1 TO NUU;
        PUT FILE(PRMOUT1) EDIT(SITE1(I),NARY1(I))
            (COL(5),A(8),X(2),A(3));
      END;
      DO I=1 TO NUL;
        PUT FILE(PRMOUT2) EDIT(SITE2(I),NARY2(I))
            (COL(5),A(8),X(2),A(3));
      END;
      DO I=1 TO NUD;
        PUT FILE(PRMOUT3) EDIT(SITE3(I),NARY3(I))
            (COL(5),A(8),X(2),A(3));
      END;
   /*                          */
   /* CREATE ALPHANUMERICALLY SORTED ARRAYS OF SITE NAMES
      IN THE THREE SWITCHED NETWORKS */
   /*                          */

      CALL SORT(NUU,SITE1,SITE11,TMP8);

      CALL SORT(NUL,SITE2,SITE22,TMP8);

      CALL SORT(NUD,SITE3,SITE33,TMP8);

   /*                          */
   /* PRINT NEXT NEIGHBOUR INFORMATION CONTAINED IN
      THE THREE PRIMITIVE CONNECTION MATRICES */
   /*                          */
      PUT PAGE FILE(PRMOUT);
      PUT SKIP(3) FILE(PRMOUT);
      PUT SKIP FILE(PRMOUT) EDIT('AUTOVON') (COL(47),A(7));
      PUT SKIP(2) FILE(PRMOUT);

CALL PRNT(NUU,NETW1,SITE1,SITE11,NARY1);

      PUT PAGE FILE(PRMOUT);
      PUT SKIP(3) FILE(PRMOUT);
      PUT SKIP FILE(PRMOUT) EDIT('AUTODIN') (COL(47),A(7));
      PUT SKIP(2) FILE(PRMOUT);

CALL PRNT(NUL,NETW2,SITE2,SITE22,NARY2);

      PUT SKIP(3) FILE(PRMOUT);
      PUT PAGE FILE(PRMOUT);
      PUT SKIP FILE(PRMOUT) EDIT('AUTOSECOCOM')
```

```
                                                (COL(45),A(11));
            PUT SKIP(2) FILE(PRMOUT);

        CALL PRNT(NUD,NETW3,SITE3,SITE33,NARY3);


/*****************************************************/
   /* BEGIN INTERNAL PROCEDURES OF PROCEDURE PRIM */
/*****************************************************/


      /* PROCEDURE CORELTSI */


    CORELTSI:PROC(NSITE);
             DCL NSITE BIN FIXED(15,0);
             DCL 1 MAST_SITE,
                   2 MSICTRL CHAR(11),
                   2 MSINUMB CHAR(3);
             DCL MASTER_SITE CHAR(14) DEF MAST_SITE;

          /* RESET SITE MASTER FILE */

             FUNCT=RESTM;
             CALL DBIO(3,0);

          /* SERIAL READ OF CIRCUIT MASTER FILE TO PULL
             OUT SITE NAMES AND CORRESPONDING SITE #S */

             STATUS=OK;
             NSITE=0;
             SITE_READ:DO WHILE(STATUS¬=ENDF);
                       FUNCT=SEQRM;
                       STATUS=OK;
                       FILEID='RMSI';
                       ELMLIST='RMSICTRLRMSINUMBEND.';
                       CALL DBIO(6,1);
                       IF (STATUS¬=OK) THEN GO TO EXIT1;
                       MASTER_SITE=IOAREA;
                       NSITE=NSITE+1;
                       SITENMI(NSITE)=SUBSTR(MSICTRL,1,8);
                       SITESNI(NSITE)=MSINUMB;
                       END SITE_READ;
             EXIT1:;
             END CORELTSI;


      /* PROCEDURE CORELTLK */


    CORELTLK:PROC(NLINK);
```

```
        DCL NLINK BIN FIXED(15,0);
        DCL 1 MAST_LINK,
              2 MLKCTRL CHAR(5),
              2 MLKCODE CHAR(2);
        DCL MASTER_LINK CHAR(7) DEF MAST_LINK;

      /* RESET LINK MASTER FILE */

         FUNCT=RESTM;
         CALL DBIO(3,0);

      /* SERIAL READ OF LINK MASTER FILE TO PULL OUT
         LINK NAMES CORRESPONDING TO LINK CODES */

        STATUS=OK;
        NLINK=0;
        LINK_READ:DO WHILE (STATUS¬=ENDF);
                  FUNCT=SEQRM;
                  STATUS=OK;
                  FILEID='RMLK';
                  ELMLIST='RMLKCTRLRMLKCODEEND.';
                  CALL DBIO(6,1);
                  IF (STATUS¬=OK) THEN GO TO EXIT2;
                  MASTER_LINK=ICAREA;
                  NLINK=NLINK+1;
                  LINKNM(NLINK)=MLKCTRL;
                  LINKCC(NLINK)=MLKCODE;
                  END LINK_READ;
        EXIT2:;
        END CORELTLK;


 /* PROCEDURE CORELTSIM */


CORELTSIM:PROC(NU,TNARY,TSITE);
          DCL NU BIN FIXED(15,0),
              TNARY(*) CHAR(3),TSITE(*) CHAR(8);
          DO I=1 TO NU;
            DO J=1 TO NUSI;
            IF (TNARY(I)=SITESNI(J)) THEN
              DO;
                TSITE(I)=SITENMI(J);
                GO TO EXTLOO;
              END;
            ELSE;
            END;
          EXTLOO:;
          END;
          END CORELTSIM;
```

```
      /* PROCEDURE PRNT */

PRNT:PROC(NENODE,TNET,TSITE,TTSITE,TNARY);
     DCL  ILK BIN FIXED(15,0),
          (INDXS(150),INDXO(150),INDXL(150)) BIN FIXED(15,0)
                                             INIT((150)0),
          XTMP CHAR(2);
     DCL NENODE BIN FIXED(15,0),
          TNET(*,*) CHAR(2),(TSITE(*),TTSITE(*)) CHAR(8),
          TNARY(*) CHAR(3);
     DO I=1 TO NENODE;
       DO J=1 TO NENODE;
       IF (TTSITE(I)=TSITE(J)) THEN
         DO;
           INDXO(I)=J;
           GO TO EXTL0;
         END;
       ELSE;
       END;
     EXTL0:;
     END;
PUT SKIP(2) FILE(PRMOUT);
PUT SKIP FILE(PRMOUT) LIST(TSITE);
PUT SKIP FILE(PRMOUT) LIST(TTSITE);
PUT SKIP FILE(PRMOUT) LIST(TNARY);
PUT SKIP(2) FILE(PRMOUT);
     DO L=1 TO NENODE;
     ILK=0;
     I=INDXO(L);
       DO J=1 TO NENODE;
       IF ((TNET(I,J)='00')|(TNET(I,J)='01')) THEN GO TO EXTL1;
       XTMP=TNET(I,J);
       ILK=ILK+1;
       INDXS(ILK)=J;
         DO K=1 TO NULK;
         IF (XTMP=LINKCO(K)) THEN
           DO;
             INDXL(ILK)=K;
             GO TO EXTL1;
           END;
         ELSE;
         END;
       EXTL1:;
       END;
       II=INDXL(1);
       III=INDXS(1);
       PUT SKIP FILE(PRMOUT)
           EDIT(TSITE(I),' (',TNARY(I),') :',LINKNM(II),' (',LINKCO(II)
                ,') -',TSITE(III),' (',TNARY(III),') ')
           (COL(31),A(8),A(1),A(3),A(2),A(5),A(1),A(2),A(2),A(8),
           A(1),A(3),A(1));
```

```
        DO K=2 TO ILK;
        II=INDXI(K);
        III=INDXS(K);
        PUT SKIP FILE(PRMOUT)
            EDIT(LINKNM(II),' (',LINKCO(II),') -',TSITE(III),
            '(',TNARY(III),')')
            (COL(45),A(5),A(1),A(2),A(2),A(8),A(1),A(3),A(1));
        END;
PUT SKIP FILE(PRMOUT);
      END;
      END PRNT;


       /* PROCEDURE SORT */


SORT:PROC(N,XIN,XOUT,TMP);
      DCL  (XIN(*),XOUT(*),TMP) CHAR(*),
           (M,N,I,J) BIN FIXED(15,0);
      DO I=1 TO N;
        XOUT(I)=XIN(I);
      END;
         /*                          */
         /* MODIFIED SHELL SORT LOGIC */
         /*                          */
      M=1;
      S1:DO  I=1 TO N WHILE(M<N);
         M=M+M;
         END S1;
      M=M-1;
      BACK:;
           M=M/2;
      S2:DO J=1 TO N-M;
         TMP=XOUT(J+M);
         S3:DO  I=J TO 1 BY -M;
            IF (XOUT(I)<=TMP) THEN GO TO NEXT;
            XOUT(I+M)=XOUT(I);
            END S3;
         NEXT:;
              XOUT(I+M)=TMP;
         END S2;
      IF  (M>1) THEN GO TO BACK;
      END SORT;


/*****************************************************************/
     /* END INTERNAL PROCEDURES OF PROCEDURE PRIM */
/*****************************************************************/


    EXIT:;
```

```
        /*                          */
        /* CLOSE FILES */
        /*                          */
         CLOSE FILE(PRIMIN);
         CLOSE FILE(PRMOUT);
         CLOSE FILE(PRMOUT1);
         CLOSE FILE(PRMOUT2);
         CLOSE FILE(PRMOUT3);
         FUNCT=CLOSM;
         STATUS=OK;
         CALL DBIO(4,0);
        /*                          */
        /* SIGN OFF THE TASK */
        /*                          */
         FUNCT=DEQUE;
         TASKID='PRIMS';
         CALL DBIO(4,0);
    END PRIM;

*PROCESS;

  /*************************************************************/
   /* BEGIN FIRST EXTERNAL PROCEDURE OF PROCEDURE PRIM */
  /*************************************************************/


        /* CIRCUIT ANALYSER PROGRAM */
        /*                          */
        /* THIS PROCEDURE, UPON GATHERING INFORMATION ABOUT A GIVEN
           CIRCUIT FROM THE FILE, DETERMINES THE LINK(S) AND SITES
           (ALONG WITH THEIR IN/OUT FLAGS) WHICH CONSTITUTE IT */
        /*                          */
        /* INPUTS- (FILE),CKT */

        /*          (FILE): THIS REFERS TO ALL USER DEFINED TOTAL
                             FILES INCLUDING;
                              MASTER FILES: RMCT,RMTR,RMLK,RMSI
                                        AND
                              VARIABLE FILES: RVCT,RVTL
                        CKT: CIRCUIT CCSD# (9 CHARACTERS)       */

        /* OUTPUTS- IKCKT,SICKI,SERIES,RFLAG */

        /*          IKCKT: A STRING OF CHARACTERS CONTAINING CCSD#
                           AND CIRCUIT IN/OUT FLAG (F) AS HEADER
                           FOLLOWED BY THE NAMES OF THE LINKS
                           TRAVERSED ALONG WITH THEIR IN/OUT FLAGS
                           AND A SPECIAL CHARACTER, @ AS TRAILER
                           ( 6[illegible]5 CHARACTERS FOR MAXIMUM ALLOWED
```

```
                    99 LINKS ON THE CIRCUIT PATH )

                EXAMPLE:

                CCSD#FLINK1FLINK2F----------LINK99F@

              SICKT: A STRING OF CHARACTERS CONTAINING CCSD#
                     AND CIRCUIT IN/OUT FLAG (F) AS HEADER
                     FOLLOWED BY THE NAMES OF THE SITES
                     TRAVERSED ALONG WITH THEIR IN/OUT FLAGS
                     AND A SPECIAL CHARACTER, @ AS TRAILER
                     ( 911 CHARACTERS FOR MAXIMUM ALLOWED
                       100 SITES ON THE CIRCUIT PATH )

                EXAMPLE:

                CCSD#FSITE1FSITE2F----------SITE100F@

             SERIES: A STRING OF CHARACTERS CONTAINING CCSD#
                     AND CIRCUIT IN/OUT FLAG (F) AS HEADER
                     FOLLOWED BY THE SITE SERIAL #S OF THE SITES
                     , THE LINK CODES AND THE IN/OUT FLAGS OF
                     THE LINKS TRAVERSED AND A SPECIAL
                     CHARACTER, @ AS TRAILER ( 608 CHARACTERS
                     FOR MAXIMUM ALLOWED 99 LINKS AND 100
                     SITES )

                EXAMPLE:

                CCSD#FSITENO1LINKCODE1F-----LINKCODE99FSITENO100@

              RFLAG: A FLAG (BIN FIXED(15,0)) WHICH IS SET TO
                     1 FROM ITS INITIAL VALUE 0 FOR FILE ERROR
                                                              */

/*****************************************************************/

                    /*** NOTES ***/

    /*  AN EXIT IS TAKEN OUT OF THE PROCEDURE WITHOUT
        ERROR MESSAGES FOR TOTAL RETURNED STATUS¬=OK

        EXIT IS ALSO TAKEN OUT OF THE PROCEDURE WITH
        ERROR MESSAGES FOR ANY ONE OF THE FOLLOWING:

          1. IF THE TRUNK SEGMENT END SITES, AS GIVEN
             IN VARIABLE FILE,RVCT, ARE NOT FOUND IN
             VARIABLE FILE,RVTL ( THIS COULD BE CAUSED
             BY IMPROPER ORDER OF LINKS IN RVTL )

          2. IF THE TRUNK SEGMENT END SITES ARE THE SAME
```

```
          3. IF CONTROL KEY FOR EITHER THE SITE OR THE
             LINK ON THE CIRCUIT PATH IS IN ERROR  */

/*****************************************************************/


DECA:PROC(CKT,LKCKT,SICKT,SERIES,RFLAG);

       /*****************************************/
       /* PROCEDURE CALLED BY PROCEDURE DECA */
       /*****************************************/


       /* INTERNAL PROCEDURE */


/*   DCL FIND ENTRY(CHAR(8),CHAR(5)) ;  */
/*   DCL FLAGS ENTRY(CHAR(1),CHAR(5)) ;  */


       /* DECLARATIONS FOR THE FILE AND THE
          IDENTIFIERS OF PROCEDURE DECA */


     DCL PRMOUT OUTPUT FILE PRINT STREAM;

     %INCLUDE EXTDCL;
     DCL (TMPFRLC(5),TMPTOLC(50),TMPSXX(50)) CHAR(8),
         (TMPLINK(49),TMPLXX(49)) CHAR(5);
     DCL CKT CHAR(9),
         LKCKT CHAR(675) VAR,
         SICKT CHAR(9911) VAR,          /* MAX 99 LINKS & 100 SITES */
         SERIES CHAR(678) VAR,
         RR(2) CHAR(4),
         (I,J,K,L,II,JJ,KK,LL,LLL,FLGG,RFLAG) BIN FIXED(15,0),
         (TMPSFXX,TMPSTXX,TMPRF,TMPRT,TMPRS) CHAR(8),
         (SFR,STO) CHAR(9),
         LFR CHAR(6),
         LXT CHAR(2),
         (FRFL,TOFL,LKFL) CHAR(1),
         (FRFL5,TOFL5,LKFL5) CHAR(5),
         (LLFR,TMPS) CHAR(5),
         (SSFR,SSTO) CHAR(8),
         FISITE CHAR(8),
         (SRFR,SRTO) CHAR(3),
         SLSS CHAR(52);
     DCL 1 MAST_CIRCUIT,
           2 MCTFLAG CHAR(1),
           2 MCTFRLC CHAR(8),
           2 MCTFFAC CHAR(3),
           2 MCTTOLC CHAR(8),
           2 MCTTFAC CHAR(3);
```

```
      DCL MASTER_CIRCUIT CHAR(23) DEFINED MAST_CIRCUIT;
      DCL 1 VARI_CTS,
            2 VCTSCONTROL CHAR(7),
            2 VCTSSITE CHAR(11),
            2 VCTSSITT CHAR(8);
      DCL VARIABLE_CTS CHAR(26) DEFINED VARI_CTS;
      DCL 1 VARI_TLS,
            2 VTLSLINK CHAR(5),
            2 VTLSSITE CHAR(11),
            2 VTLSSITT CHAR(8);
      DCL VARIABLE_TLS CHAR(24) DEFINED VARI_TLS;
      DCL 1 MAST_LINK,
            2 MLKFLAG CHAR(1),
            2 MLKCODE CHAR(2);
      DCL MASTER_LINK CHAR(3) DEFINED MAST_LINK;
      DCL 1 MAST_SITE,
            2 MSIFLAG CHAR(1),
            2 MSINUMB CHAR(3);
      DCL MASTER_SITE CHAR(4) DEFINED MAST_SITE;

      FFLAG=0;      /* INITIALISE FILE ERROR FLAG */

           /*                          */
           /* READ A RECORD CORRESPONDING TO THE SPECIFIED
              CIRCUIT FROM THE CIRCUIT MASTER FILE */
           /*                          */
      FUNCT=READM;
      STATUS=OK;
      FILEID='RMCT';
      CONTROL=CKT;
      ELMLIST='RMCTFLAGRMCTFBLCRMCTFFACRMCTTOLCRMCTTFACEND.';
      CALL DBIO(7,1);
      IF (STATUS¬=OK) THEN GO TO EXIT;
      MASTER_CIRCUIT=IOAREA;
      LKCKT,SICKT,SERIES=CKT||MCTFLAG; /* CCSD# AND I/C FLAG ADDED */
           /*                          */
           /* LINKAGE PATH TO CIRCUIT-TRUNK-SITE VARIABLE FILE-
              TO DETERMINE ALL TRUNKS  CONTAINING THE
              CHOSEN CIRCUIT */
           /*                          */
      RR(1)='LKCT';
      FLGG=' ';
      TMPSFXX=TMPSTXX=(8)'0';

   /****   BEGIN LOOP FOR THE CIRCUIT   ****/

CIRCT:DO WHILE(RR(1)¬=ENDP);  /* LOOP FOR FINDING ALL TRUNKS
                                 IN THE GIVEN CIRCUIT */
      LL,LLL=0;
      DO I=1 TO 50;
        TMPFRLC(I),TMPTOLC(I),TMPSXX(I)=(8)'0';
        IF (I<=49) THEN TMPLINK(I),TMPLXX(I)=(5)'0';
```

```
      END;
      TMPRF=TMPSFXX;
      TMPRT=TMPSTXX;
      REFER=RR(1);
      FUNCT=READV;
      STATUS=OK;
      FILEID='RVCT';
      LKPATH='RMCTLKCT';
      CONTROL=CKT;        /* SAME AS BEFORE */
      ELMLIST='RVCTRMTRRVCTRMSIRVGTTOLCEND.';
      CALL DBIO(9,1);
      IF (STATUS¬=OK) THEN GO TO EXIT;
      VARIABLE_CTS=IOAREA;
      TMPSFXX=SUBSTR(VCTSSITE,1,8);
      TMPSTXX=VCTSSITT;
           /*                          */
           /* SWITCH THE END SITES OF THE TRUNK
              SEGMENT (IF NEEDED)                  */
           /*                          */
      IF (FLGG=') THEN
        DO;
          FLGG=1;
          GO TO NO_CHECK;
        END;
      ELSE;
      IF (TMPRT=TMPSFXX) THEN GO TO CHANGE_EXIT;
      IF (TMPRT=TMPSTXX) THEN
        DO;
          TMPR8=TMPSFXX;
          TMPSFXX=TMPSTXX;
          TMPSTXX=TMPR8;
          GO TO CHANGE_EXIT;
        END;
      ELSE;
      CHANGE_EXIT:;
      NO_CHECK:;
      IF (REFER=ENDP) THEN GO TO DE8;     /* ALL TRUNKS FINISHED */
      RR(1)=REFER;
           /*                          */
           /* LINKAGE PATH TO TRUNK-LINK-SITE VARIABLE FILE-
              TO DETERMINE THE LINK(S) IN THE CHOSEN TRUNK
              TO WHICH THE SPECIFIED CIRCUIT BELONGS
              AND THE SITES TRAVERSED */
           /*                          */
      RR(2)='LKTL';

     /****   BEGIN LOOP FOR A TRUNK   ****/

TRUNK:DO WHILE(RR(2)¬=ENDP);  /* LOOP FOR FINDING ALL SITES AND
                                 LINKS IN A GIVEN TRUNK */
      REFER=RR(2);
      FUNCT=READV;
```

```
STATUS=OK;
FILEID='RVTL';
LKPATH='RMTBLKTI';
CONTROL=VCTSCONTROL;
DIMLIST='RVTLRMLKRVTLRMSIRVTLTOLCEND.';
CALL DBIO(0,1);
IF (STATUS¬=OK) THEN GO TO EXIT;
VARIABLE_FLS=ICAREA;
     /*                              */
     /* CREATE ARRAYS OF FROM AND TO SITES AND LINKS
        IN THE CHOSEN TRUNK */
     /*                              */
LL=LL+1;
TMPFRLC(LL)=SUBSTR(VTLSSITF,1,8);
TMPTOLC(LL)=VTLSSITT;
TMPLINK(LL)=VTLSLINK;
IF (LL>=2) THEN
  DO;
    IF ((TMPFRLC(LL)=TMPFRLC(LL-1)) &
        (TMPTOLC(LL)=TMPTOLC(LL-1))) THEN
      DO;
        TMPFRLC(LL),TMPTOLC(LL)=(8)'0';
        TMPLINK(LL)=(5)'0';
        LL=LL-1;
      END;
    ELSE;
  END;
ELSE;
PR(2)=RPFBR;
END TRUNK;

/****   END LOOP FOR A TRUNK   ****/

     /*                              */
     /* SORT ABOVE ARRAYS TO DETERMINE THE LINKS AND
        SITES TRAVERSED IN THE CHOSEN TRUNK BY THE
        GIVEN CIRCUIT. */
     /*                              */
  DO;
    II,JJ,KK,K=0;
    LLL=LL;
     /*                              */
     /* CREATE ARRAYS OF SITES AND LINKS IN THE TRUNK
        WHICH ARE COMMON TO BOTH THE CIRCUIT AND THE TRUNK */
     /*                              */
    TMPFRLC(LLL+1)=TMPTOLC(LLL);
    DO1:DO I=1 TO LLL+1;
        IF (TMPFRLC(I)=TMPSIXX) THEN JJ=I;
        IF (TMPFRLC(I)=TMPSTXX) THEN KK=I;
        END DO1;
    IF ((JJ=0)|(KK=0)) THEN
      DO;
```

```
      RFLAG=1;
      PUT SKIP FILE(PRMOUT)
          EDIT('** ERROR TRUNK=',VCTSCONTROL,' CIRCUIT=',CKT)
              (COL(2),A(15),A(7),(2)A(9));
            GO TO EXIT;
    END;
  ELSE;
  IF (JJ=KK) THEN
    DO;
       RFLAG=1;
       PUT SKIP FILE(PRMOUT) EDIT('** ERROR=',
                 VCTSCONTROL) (COL(2),A(9),A(7));
       GO TO DE7;          /* NO ADD ACTION ON DEFAULT */
    END;
  ELSE;
  IF (KK>JJ) THEN
   /*                          */
   /* TRUNK AND CIRCUIT DIRECTION IS IDENTICAL */
   /*                          */
    DO;
      II=1;
      LLL=KK-JJ;
      DE2:DO I=JJ TO KK;
          K=K+1;
          TMPSXX(K)=TMPFRLC(I);
          IF (I<KK) THEN TMPLXX(K)=TMPLINK(I);
          END DE2;
      GO TO DE4;
    END;
  ELSE;
  IF (JJ>KK) THEN
   /*                          */
   /* TRUNK AND CIRCUIT DIRECTIONS ARE NOT IDENTICAL */
   /*                          */
    DO;
      II=2;
      LLL=JJ-KK;
      DE3:DO I=JJ TO KK BY -1;
          K=K+1;
          TMPSXX(K)=TMPFRLC(I);
          IF (I<JJ) THEN TMPLXX(K-1)=TMPLINK(I);
          END DE3;
      GO TO DE4;
    END;
  ELSE;
  DE4:;
  FISITE=TMPSXX(K);  /* SAVE FINAL SITE */
   /*                          */
   /* READ RECORDS IN THE SITE AND LINK MASTER FILES
      TO DETERMINE IN/OUT FLAGS, LINK CODES AND SITE
      SERIAL #S */
   /*                          */
```

```
      DE5:DO J=1 TO LLL;

          CALL FIND(TMPSXX(J),TMPIXX(J));

          END DE5;
        END;
        DE7:;
      END CIRCT;

     /****    END LOOP FOR THE CIRCUIT    ****/

      DE8:;
          /*                          */
          /* ADD FINAL SITE  */
          /*                          */
      FUNCT=READM;
      STATUS=OK;
      FILEID='RMSI';
      CONTROL=FISITE;
      IF (CONTROL=(8)'0') THEN
        DO;
          RFLAG=1;
          PUT SKIP FILE(PRMOUT)
              EDIT('** ERROR IN FINAL SITE FOR CIRCUIT',CKT)
              (COL(2),A(35),A(9));
          GO TO EXIT;
        END;
      ELSE;
      FLMLIST='RMSIFLAGRMSINUMBEND.';
      CALL DBIC(7,1);
      IF (STATUS¬=OK) THEN GO TO EXIT;
      MASTER_SITE=IOAREA;
      SICKT=SICKT||FISITE||MSIFLAG;  /* SITE & I/O FLAG ADDED */
      SERIES=SERIES||MSINUMB;
      LKCKT=LKCKT||'@';
      SICKT=SICKT||'@';
      SERIES=SERIES||'@';
PUT SKIP FILE(PRMOUT) LIST(SERIES);
          /*                          */
          /*  SORT CIRCUIT DATA FOR OUTPUT FORMAT
              NOTE: THIS PORTION OF THE CODE THAT
                    FOLLOWS FROM HERE TILL THE BEGIN
                    INTERNAL PROCEDURE OF PROCEDURE
                    DECA CAN BE OMITTED.  IT IS THERE
                    ONLY FOR THE DEBUGGING AID IN ANY
                    FUTURE TROUBLE SO THAT A COMPLETE
                    CIRCUIT PATH CAN BE TRACED IN TERMS
                    OF LINKS TRAVERSED              */
          /*                          */
       II=LENGTH(LKCKT)-11;
       II=BIN(II/6,15,0);
       DO KK=1 TO II;
```

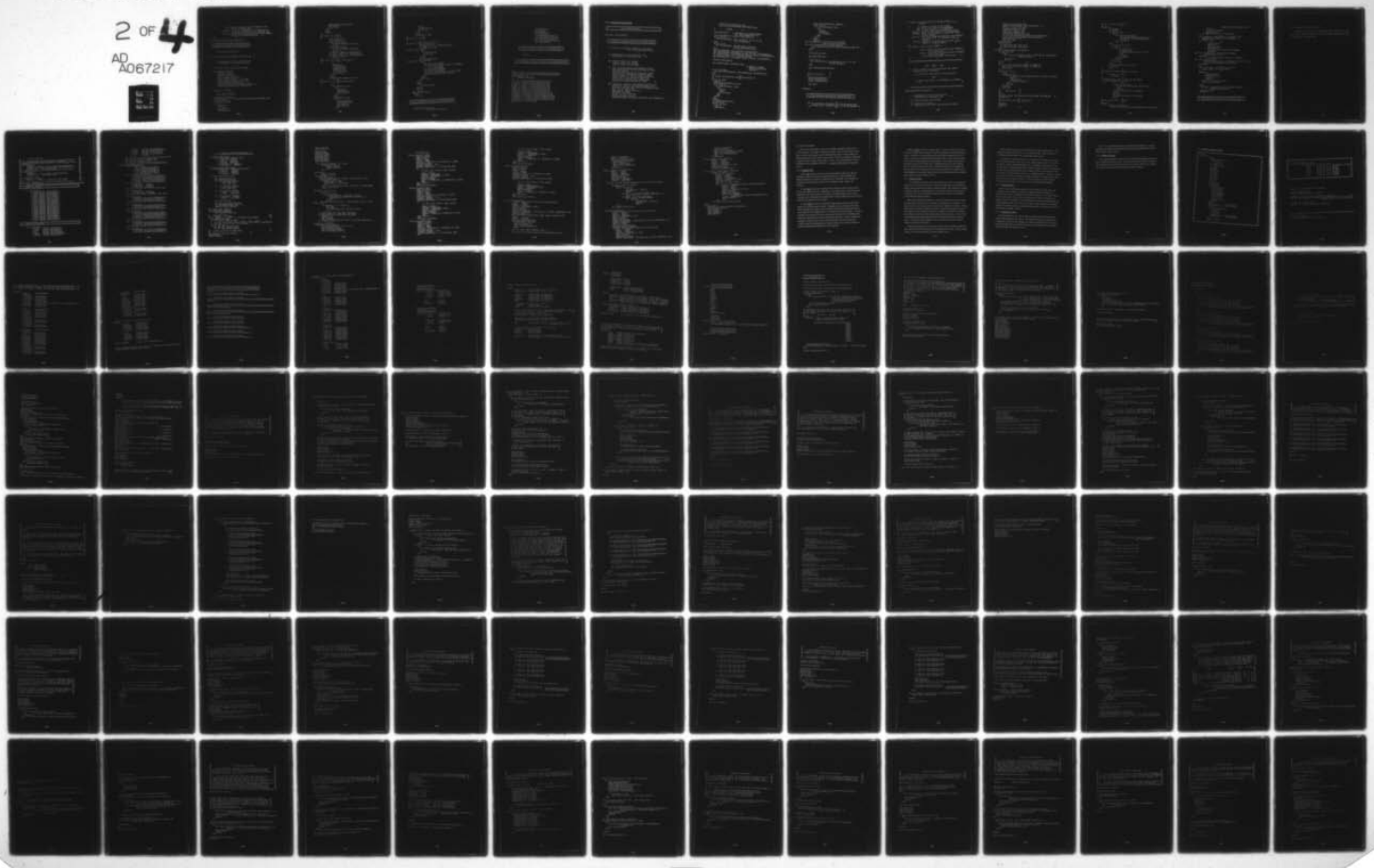
```
      I=11+6*(KK-1);
      J=11+9*(KK-1);
      K=11+6*(KK-1);
      L=14+6*(KK-1);
      LFR=SUBSTR(LKCKT,I,6);
      SFR=SUBSTR(SICKT,J,9);
      LLFR=SUBSTR(LFR,1,5);
      LKFL=SUBSTR(LFR,6,1);

      CALL FLAGS(LKFL,LKFL5);

      SSFR=SUBSTR(SFR,1,8);
      FRFL=SUBSTR(SFR,9,1);

      CALL FLAGS(FRFL,FRFL5);

      SRFR=SUBSTR(SERIES,K,3);
      LXT=SUBSTR(SERIES,L,2);
      J=11+9*KK;
      K=11+6*KK;
      STO=SUBSTR(SICKT,J,9);
      SSTO=SUBSTR(STO,1,8);
      TOFL=SUBSTR(STO,9,1);

      CALL FLAGS(TOFL,TOFL5);

      SRTO=SUBSTR(SERIES,K,3);
      SLSS=SSFR||'('||SRFR||')'||FRFL5||'-'||LLFR||'('||LXT||')'
           ||LKFL5||'-'||SSTO||'('||SRTO||')'||TOFL5;
    END;


/*****************************************************/
/* BEGIN INTERNAL PROCEDURES OF PROCEDURE DECA */
/*****************************************************/


FIND:PROC(FISITE,FILINK);
      DCL FISITE CHAR(8),FILINK CHAR(5);
      FUNCT='READM';
      STATUS='OK';
      FILEID='RMSI';
      CONTROL=FISITE;
      IF (CONTROL=(8)' ') THEN
        DO;
          RFLAG=1;
          PUT SKIP FILE(PRMOUT) EDIT('** ERROR TRUNK=',
              VCTSCONTROL) (COL(2),A(15),A(7));
          GO TO EXIT;
        END;
      ELSE;
      ELMLIST='RMSIFLAGRMSINUMEND.';
```

```
        CALL DBIO(7,1);
        IF (STATUS¬=OK) THEN GO TO EXIT;
        MASTER_SITE=ICAREA;
        SICKT=SICKT||FISITE||MSIFLAG; /* SITE ADDED */
        STATUS=OK;
        FILEID='FMLK';
        CONTROL=FILINK;
        IF (CONTROL=(5)'J') THEN
          DO;
            RFLAG=1;
            PUT SKIP FILE(PRMOUT) EDIT('** ERROR TRUNK=',
                VCISCONTROL) (COL(2),A(15),A(7));
            GO TO EXIT;
          END;
        ELSE;
        FLMLIST='FMLKFLAGFMLKCODEEND.';
        CALL DBIO(7,1);
        IF (STATUS¬=OK) THEN GO TO EXIT;
        MASTER_LINK=ICAREA;
        LKCKT=LKCKT||FILINK||MLKFLAG;  /* LINK ADDED */
        SERIES=SERIES||MSINUMB||MLKCODE||MLKFLAG; /* SITE,LINK CODE
                                                     LINK FLAG ADDED */
        END FIND;


         /* PROCEDURE FLAGS */


/* NOTE- ALL THE CHARACTERS USED IN CHAR5 EXCEPT CHARACTER "A" HAVE
         THE FOLLOWING MEANINGS IN THEIR RESPECTIVE CHARACTER
         POSITIONS

           A='00000000'B: "IN" (AVAILABLE) STATE
           S='00001000'B: SATELLITE LINK OUTAGE
           M='00001000'B: MHD
           J='00000100'B: JAMMING
           B='00000010'B: BLAST
           E='00000001'B: EMP                          */


  FLAGS:PROC(CHAR1,CHAR5);
        DCL CHAR1 CHAR(1),CHAR5 CHAR(5),
            BIT8 BIT(8);
        CHAR5='SMJBE';
        BIT8=UNSPEC(CHAR1);
        IF (BIT8=('00000000'B)) THEN
          DO;
            CHAR5='AAAAA';
            GO TO XIT;
          END;
        ELSE;
        DO I=4 TO 8;
```

```
          IF (SUBSTR(BIT8,I,1)='0'B) THEN SUBSTR(CHAR5,I-3,1)='A';
        END;
        XIT:;
        END FLAGS;


      /*************************************************/
      /* END INTERNAL PROCEDURES OF PROCEDURE DECA */
      /*************************************************/


      EXIT:;
      END DECA;

  /*****************************************************************/
   /*  END FIRST EXTERNAL PROCEDURE OF PROCEDURE PRIM  */
  /*****************************************************************/


*PROCESS;

  /*****************************************************************/
   /* BEGIN SECOND EXTERNAL PROCEDURE OF PROCEDURE PRIM */
  /*****************************************************************/


        /* NETWORK ADDER PROGRAM */
        /*                       */
        /* THIS PROCEDURE UPON RECEIVING INFORMATION
           ABOUT A CIRCUIT FROM PROCEDURE DECA
           ADDS SITE(S) AND LINK(S) TO THE SPECIFIED
           INPUT NETWORK PRIMITIVE CONNECTION MATRIX */
        /*                       */
        /* INPUTS- LINKS,SERIES,MNDS,LNDS(.),NET(.,.),
                   SERVA1,SERVA2                       */

        /*          LINKS: NUMBER OF LINKS TRAVERSED BY THE CIRCUIT
                   SERIES: SAME AS DEFINED IN PROCEDURE DECA
                     MNDS: NUMBER OF SITES IN THE NETWORK AT THE
                           START OF PROCEDURE NEA
                 LNDS(.): ARRAY OF 3 CHARACTER SITE SERIAL #S IN
                           THE NETWORK AT THE START OF PROCEDURE NEA
                NET(.,.): PRIMITIVE CONNECTION MATRIX OF THE NETWORK
                           AT THE START OF PROCEDURE NEA
                  SERVA1: SAME AS DEFINED IN MAIN PROCEDURE PRIM
                  SERVA2: SAME AS DEFINED IN MAIN PROCEDURE PRIM */
        /*                       */
        /* OUTPUTS- MNDS,LNDS(.),NET(.,.)  */
```

AD-A067 217 COMPUTER SCIENCES CORP FALLS CHURCH VA F/G 17/2
INVESTIGATION OF THE VULNERABILITY/SURVIVABILITY OF SYSTEMS SUP--ETC(U)
JUN 78 H BLANK, G KINAL, M BASSMAN, D GAN DNA001-77-C-0015
UNCLASSIFIED DNA-4354F-1C-1 NL

2 OF 4
AD A067217



```
      /*          MNDS: NUMBER OF SITES IN THE NETWORK AT THE
                        END OF PROCEDURE NEA
                LNDS(.): ARRAY OF 3 CHARACTER SITE SERIAL #S IN
                        THE NETWORK AT THE END OF PROCEDURE NEA
               NET(.,.): PRIMITIVE CONNECTION MATRIX OF THE NETWORK
                        AT THE END OF PROCEDURE NEA    */
      /*                    */

NEA:PROC(LINKS,SERIES,MNDS,LNDS,NET);

   /*************************************/
   /* PROCEDURE CALLED BY PROCEDURE NEA */
   /*************************************/


      /* INTERNAL PROCEDURE */


/*  DCL INDX_INFORM ENTRY(BIN FIXED(15,0));  */


      /* DECLARATIONS FOR THE FILE AND THE
         IDENTIFIERS OF PROCEDURE NEA */


   DCL PRMOUT OUTPUT FILE PRINT STREAM;

   DCL NET(*,*) CHAR(2),
       LNDS(*) CHAR(3),
       (XX,XXN) CHAR(2),
       SERIES CHAR(608) VAR,
       CHARF CHAR(1) INIT('0'),
       CHARFB BIT(8) INIT('00000000'B),
       (SERVC,SERVA1,SERVA2) BIT(8) EXT,
       (ZZ1,ZZ2,ZZF,ZZT) CHAR(3),
       (I,J,II,JJ,KK,LL,LINKS) BIN FIXED(15,0),
       (MNDS,NF,NT,MFLAG) BIN FIXED(15,0);

/*                  */
   R2:DO I=1 TO LINKS;

      CALL INDX_INFORM(I);

   CHARFB=UNSPEC(CHARF);
   IF (CHARFB=('00000000'B)|CHARFB=SERVA1|CHARFB=SERVA2) THEN
   LINK_ADD_LOOP:DO;
    IF (MNDS=0) THEN
     DO;
       NF=MNDS+1;
       NT=NF+1;
       LNDS(NF)=ZZ1;
       LNDS(NT)=ZZ2;
```

```
        NET(NF,NT),NET(NT,NF)=XX;
        MNDS=MNDS+2;
        GO TO RR3;
      END;
     ELSE;
     R1:DO;
/*                  */
/* SEARCH FOR SITES */
/*                  */
        ZZF=ZZ1;ZZT=ZZ2;
        NF,NT,MFLAG=0;
        R3:DO J=1 TO MNDS;
           IF (ZZF=LNDS(J)) THEN NF=J;
           END R3;
        R4:DO J=1 TO MNDS;
           IF (ZZT=LNDS(J)) THEN NT=J;
           END R4;
        IF ((NF¬=0)&(NT=0)) THEN MFLAG=1;
        IF ((NF=0)&(NT¬=0)) THEN MFLAG=2;
        IF ((NF¬=0)&(NT¬=0)) THEN MFLAG=3;
        IF (MFLAG=0) THEN
/*                  */
/* NO OLD SITE (ADD A NEW LINK) */
/*                  */
          DO;
            NF=MNDS+1;
            NT,MNDS=NF+1;
            LNDS(NF)=ZZF;
            LNDS(NT)=ZZT;
            NET(NF,NT)=XX;
            GO TO RR1;
          END;
        ELSE;
        IF ((MFLAG=1)|(MFLAG=2)) THEN
/*                  */
/* ONE OLD SITE (ADD A NEW LINK) */
/*                  */
          DO;
            IF (MFLAG=1) THEN
              DO;
                NT=MNDS+1;
                MNDS=MNDS+1;
                LNDS(NT)=ZZT;
              END;
            ELSE;
            IF (MFLAG=2) THEN
              DO;
                NF=NT;NT=MNDS+1;
                ZZF=ZZ2;ZZT=ZZ1;
                MNDS=MNDS+1;
                LNDS(NT)=ZZT;
              END;
```

```
            ELSE;
            NET(NF,NT)=XX;
            GO TO RR1;
          END;
        ELSE;
        IF (MFLAG=3) THEN
/*              */
/* BOTH OLD SITES */
/*              */
          DO;
/* CHECK IF LINK ALREADY EXISTS */
            XXN=NET(NF,NT);
            IF (XXN='00') THEN XXN=NET(NT,NF);
            IF (XXN='00') THEN
/* ADD A NEW LINK  */
              DO;
                NET(NF,NT)=XX;
                GO TO RR1;
              END;
            ELSE;
            IF (XXN¬=XX) THEN
/* MULTIPLE LINK FOUND */
              DO;
                PUT SKIP FILE(PRMOUT)
                EDIT('MULTIPLE LINKS BETWEEN ',LNDS(NF),
                     ' & ',LNDS(NT))
                (COL(1),A(23),(3)A(3));
                PUT SKIP FILE(PRMOUT)
                EDIT('OLD LINK=',XXN,' TST LINK=',XX)
                (COL(1),A(9),A(2),A(10),A(2));
              END;
            ELSE;
            GO TO RR2;
          END;
        ELSE;
        RR1:;
            NET(NT,NF)=XX;
        RR2:;
       END R1;
        RR3:;
    END LINK_ADD_LOOP;
     ELSE;
       END R2;


/*****************************************************/
/* BEGIN INTERNAL PROCEDURE OF PROCEDURE NEA */
/*****************************************************/


     INDX_INFORM:PROC(N);
                 DCL N BIN FIXED(16,0);
```

```
                    II=11+6*(N-1);
                    JJ=11+6*N;
                    KK=14+6*(N-1);
                    LL=16+6*(N-1);
                    ZZ1=SUBSTR(SERIES,II,3);
                    ZZ2=SUBSTR(SERIES,JJ,3);
                    XX=SUBSTR(SERIES,KK,2);
                    CHARF=SUBSTR(SERIES,LL,1);
                    END INDX_INFORM;


      /*************************************************/
      /*  END INTERNAL PROCEDURE OF PROCEDURE NEA  */
      /*************************************************/


          END NEA;


  /*****************************************************************/
    /* END SECOND EXTERNAL PROCEDURE OF PROCEDURE PRIM */
  /*****************************************************************/



/*
//LKED.SYSLMOD DD DSN=TOTAL.DBLIB(PRIMS),DISP=SHR
//LKED.DBIC DD DSN=SYS9.TOTAL.LINKLIB4,DISP=SHR
//LKED.SYSIN DD *
      INCLUDE DBIC(DBIC)
/*
//  EXEC TOTAL4,PGNAME=PRIMS,REGION=350K
//RMCT DD DSN=UH010.PI119.RMCT,DISP=SHR
//RMTR DD DSN=UH010.PI119.RMTR,DISP=SHR
//RMLK DD DSN=UH010.PI119.RMLK,DISP=SHR
//RMSI DD DSN=UH010.PI119.RMSI,DISP=SHR
//RVCT DD DSN=UH010.PI119.RVCT,DISP=SHR
//RVTL DD DSN=UH010.PI119.RVTL,DISP=SHR
//PRIMIN DD DSN=M1765.PRIMIN.DATA,DISP=SHR
//PRMOUT DD DSN=M1765.PRMOUT.DATA,DISP=OLD
//PRMOUT1 DD DSN=M1765.PRMOUT1.DATA,DISP=SHR
//PRMOUT2 DD DSN=M1765.PRMOUT2.DATA,DISP=SHR
//PRMOUT3 DD DSN=M1765.PRMOUT3.DATA,DISP=SHR
//SYSPRINT DD SYSOUT=(A,U)
//
```

3.2.4 Connectivity Search Program

```
/*************************************************************/
/*                MAIN PROCEDURE CNSE                        */
/*************************************************************/

CNSE:PROC OPTIONS(MAIN);

   /*****************************************************/
   /*  EXTERNAL PROCEDURE CALLED BY PROCEDURE CNSE      */
   /*****************************************************/

     DCL PATH EXT ENTRY(BIN FIXED(15,0),PIC '999',
                        PIC '999',BIN FIXED(15,0));

     /* DECLARATIONS FOR THE FILES AND THE
        IDENTIFIERS OF PROCEDURE CNSE     */

     DCL PRMCUT1 INPUT FILE STREAM,
         CNSEIN1 INPUT FILE STREAM,
         CNSEOT1 OUTPUT FILE STREAM;

     /*   FILE PRMOUT1 CONTAINS THE NUMBER OF SITES
          AND THE PRIMITIVE CONNECTION MATRIX FOR THE
          SWITCHED NETWORK
          FILE CNSEIN1 CONTAINS THE FROM AND TO SITE
          SERIAL #S AND INFORMATION REGARDING WHAT
          NEXT NEIGHBORS IN THE PRIMITIVE CONNECTION
          MATRIX ARE TO BE DELETED TO MODIFY IT
          FILE CNSEOT1 CONTAINS THE SIMPLE PATHS
          BETWEEN THE GIVEN FROM AND TO SITES        */

     DCL PCM(150,150) CHAR(2) INIT((22500)(1)'00') EXT,
         SITES(150) CHAR(3 ) INIT((150)(1)'000') EXT,
         SITEN(150) CHAR(8) INIT((150)(1)'00000000') EXT,
         TMP1 PIC '9' INIT('0'),
         TMP8 CHAR(8) INIT('00000000'),
         (NFF,NTT) PIC '999',
         PDL(*) PIC '999' CTL EXT,
         PTAG(*) CHAR(2) CTL EXT,
         (NUMC,NTMP,ROW,COLUMN,FLAG,FLGR,FLGC) BIN FIXED(15,0),
```

```
     PATHLTH BIN FIXED(15,0) EXT,
     (I,J,K,N,IP,LOOP,NPATH) BIN FIXED(15,0);

/*           INPUTS           */

 FLAG,PATHLTH=0;   /* INITIALISE NO MODIFICATIONS
                      AND PATH LENGTH FLAGS */
 GET FILE(PRMOUT1) LIST(N) COPY(CNSEOT1);
 DO I=1 TO N;
   GET FILE(PRMOUT1) EDIT ((PCM(I,J) DO J=1 TO N))
                     (COL(1),(N)A(2));
 END;
 DO I=1 TO N;
   GET FILE(PRMOUT1) EDIT(SITEN(I),SITES(I))
                     (COL(5),A(8),X(2),A(3));
 END;
 GET FILE(CNSEIN1) LIST(NFF,NTT) COPY(CNSEOT1);
 GET FILE(CNSEIN1) EDIT(FLAG) (COL(1),F(1,0)) COPY(CNSEOT1);
 GET FILE(CNSEIN1) EDIT(PATHLTH) (COL(1),F(2,0)) COPY(CNSEOT1);
 IF (FLAG=0) THEN GO TO NO_MODIFICATIONS;
 GET FILE(CNSEIN1) EDIT(NUMC) (COL(1),F(2,0)) COPY(CNSEOT1);

 SCOPE_OF_NUMC:BEGIN;

 DCL FROMTOS(NUMC) CHAR(169) VAR;
                                   /* MAXIMUM 20 NEXT
                                      NEIGHBOR DEIETICNS
                                      ALLOWED */
 DO I=1 TO NUMC;
   GET SKIP FILE(CNSEIN1) LIST(FROMTOS(I)) COPY(CNSEOT1);
 END;
 /*                               */
 /* MODIFY THE PRIMITIVE CONNECTION MATRIX */
 /*                               */
 DO I=1 TC NUMC;
   FLGR=0;
   TMP8=SUBSTR(FROMTOS(I),1,8);
   DO J=1 TO N;
     IF (TMP8=SITEN(J)) THEN
       DO;
         FLGR=1;
         ROW=J;
         GO TO DONE1;
       END;
     ELSE;
   END;
   DONE1:;
   IP=LENGTH(FROMTOS(I));
   IP=(IP-1)/8;
   DO J=1 TC IP-1;
     FLGC=0;
     NTMP=J*8+1;
```

```
            TMP8=SUBSTR(FROMTOS(I),NTMP,8);
            DO K=1 TO N;
              IF (TMP8=SITEN(K)) THEN
                DO;
                  FLGC=1;
                  COLUMN=K;
                  GO TO DONE2;
                END;
              ELSE;
            END;
            DONE2:;
    PUT FILE(CNSEOT1) EDIT('ROW=',ROW,COLUMN)
                      (COL(1),A(4),(2)F(3,0));
    PUT SKIP FILE(CNSECT1) LIST(FLGR,FLGC);
          IF ((FLGR=1)&(FLGC=1)) THEN PCM(ROW,COLUMN)='00';
        END;
      END;

      END SCOPE_OF_NUMC;

      NO_MODIFICATIONS:;

      DO I=1 TO N;
        PUT FILE(CNSEOT1) EDIT((PCM(I,J) DO J=1 TO N))
                          (COL(1),(N)A(2));
      END;

      CALL PATH(N,NFF,NTT,NPATH);


     /*                          */
     /* CLOSE FILES */
     /*                          */
      CLOSE FILE(FRMCUT1);
      CLOSE FILE(CNSEIN1);
      CLOSE FILE(CNSEOT1);

      END CNSE;


*PROCESS;


    /*****************************************************/
    /*  BEGIN EXTERNAL PROCEDURE OF PROCEDURE CNSE  */
    /*****************************************************/


      /*                                  */
      /* THIS PROCEDURE DETERMINES ALL SIMPLE PATHS FROM
         A GIVEN SITE TO ANOTHER SITE IN A GIVEN NETWORK */
      /*                                  */
```

```
/* INPUTS- N,NFF,NTT,SITES(.),SITEN(.),PCM(.,.),
           PATHLTH                                        */

/*          N: NUMBER OF SITES IN THE NETWORK
          NFF: FROM SITE SERIAL # (3 CHARACTERS)
          NTT: TO SITE SERIAL # (3 CHARACTERS)
     SITES(.): ARRAY OF SITE SERIAL #S IN THE NETWORK
     SITEN(.): ARRAY OF SITE NAMES (8 CHARACTERS EACH)
               IN THE NETWORK
     PCM(.,.): PRIMITIVE CONNECTION MATRIX OF THE NETWORK
               WITH LINK CODES (2 CHARACTERS EACH) AS
               ELEMENTS
      PATHLTH: LENGTH OF A SIMPLE PATH IN TERMS OF THE
               NUMBER OF LINKS TRAVERSED                */
/*                                                      */
/* OUTPUTS- PDL(.),PTAG(.),NPATH */

/*          PDL(.): ARRAY OF SITES (SERIAL #S) TRAVERSED
                    IN A SIMPLE PATH
                PTAG(.): ARRAY OF LINKS (LINK CODES) TRAVERSED
                    IN A SIMPLE PATH
                  NPATH: NUMBER OF SIMPLE PATHS FOUND     */
/*                                                      */

/*********************************************************/


                    /***   NOTES   ***/

/*  EXIT IS TAKEN OUT OF THE PROCEDURE WITH ERROR
    MESSAGE FOR THE FOLLOWING:

     1. FROM AND TO SITES ARE THE SAME (NFF=NTT)

                         ALSO

     2. "SINK" SITE NAME, IF ENCOUNTERED, IS PRINTED
        OUT                                             */


/*********************************************************/

PATH:PROC(N,NFF,NTT,NPATH);

/*  DECLARATIONS FOR THE FILE AND THE
    IDENTIFIERS OF PROCEDURE PATH        */

DCL CNSOUT1 OUTPUT FILE STREAM;

DCL NN(150) INIT((150)0),
    INN(150,150) PIC '999' INIT((22500)(1)'000'),
```

```
        PCM(150,150) CHAR(2) EXT,
        PPCM(150,149) CHAR(2) INIT((22350)(1)'  '),
        PDL(*) PIC '999' CTL EXT,
        PTAG(*) CHAR(2) CTL EXT,
        SITES(150) CHAR(3) EXT,
        SITEN(150) CHAR(8) EXT,
        PATHLTH BIN FIXED(15,0) EXT,
        (I,J,K,L,N,NF,NT,NX,JL,IP,NXT) BIN FIXED(15,0),
        (IFLG,NPATH,COUNT,LASTNN) BIN FIXED(15,0),
        (NFF,NTT) PIC '999',
        DECI PIC '99';
    NPATH,COUNT=0;
    DO I=1 TO N;
      IF (NFF=SITES(I)) THEN NF=I;
      IF (NTT=SITES(I)) THEN NT=I;
    END;
PUT SKIP FILE(CNSEOT1) LIST(NF,NT);
    IF (NF=NT) THEN
       S5:DO;
          PUT SKIP FILE(CNSEOT1) EDIT ('FROM AND TO SITES SAME')
                                 (COL(5),A(22));
          GO TO EXIT;
          END S5;
    ELSE;
    /*                                  */
    /* SET UP NEXT NEIGHBOUR MATRIX IN TERMS OF
       SITE SERIAL # AND TAG MATRIX IN TERMS OF
       LINK CODE */
    /*                                  */
    S10:DO I=1 TO N;
        K=0;
        S20:DO J=1 TO N;
            IF ((PCM(I,J)='00')|(PCM(I,J)='01')) THEN GO TO S30;
                K=K+1;
                LNN(I,K)=J;
                PPCM(I,K)=PCM(I,J);
            S30:;
            END S20;
        NN(I)=K;
        END S10;
    /*                      */
    /*     MAIN LOGIC       */
    /*                      */
    IP=1;
    ALLOCATE PDL(N) INIT((N)(1)'000'),PTAG(N) INIT((N)(1)'  ');
    PDL(IP)=NF;
    /*                      */
    /* END SITE IS NOT YET OBTAINED */
    /*                      */
    ENDNODE:;
    LASTNN=0;
    /*                      */
```

```
/* PDL IS NOT YET EMPTY */
/*                      */
PDLEMTY:;
         NF=PDL(IP);
         JL=NN(NF);
         IF (JL=0) THEN
            S35:DO;
                PUT SKIP FILE(CNSEOT1)
                    EDIT('***NO NEXT NEIGHBOURS OF ',
                    SITEN(NF)) (COL(5),A(25),A(8));
                GO TO POPNODE;
                END S35;
         ELSE;
            IF (LASTNN~=0) THEN
            S40:DO;
                S50:DO J=1 TO JL;
                    NX=LNN(NF,J);
                    IF (NX=LASTNN) THEN GO TO S60;
                    ELSE;
                    END S50;
                GO TO POPNODE;
                END S40;
            ELSE J=1;
            GO TO S70;
         S60:;
         IF (J=JL) THEN GO TO POPNODE;
         ELSE J=J+1;
         S70:;
/*                      */
/* CHECK ALL NEXT NEIGHBOURS IN LNN */
/*                      */
         S80:DO L=J TO JL;
             IFLG=0;
/*                      */
/* CONSIDER ALL SITES ON THE PDL FOR CHECK
   BEFORE ADDING TO THE PDL */
/*                      */
             S90:DO K=1 TO IP;
                 IF (PDL(K)=LNN(NF,L)) THEN IFLG=1;
                 ELSE;
                 END S90;
             IF (IFLG=0) THEN GO TO NEWNODE;
             ELSE;
             END S80;
             IF (IFLG=1) THEN GO TO POPNODE;
             ELSE;
/*                      */
/* ADD A NEW SITE TO PDL */
/*                      */
NEWNODE:;
         PTAG(IP)=PPCM(NF,L);
          IF(IP=1) THEN PUT SKIP FILE(CNSEOT1) EDIT('PTAG1='
```

```
                                ,PTAG(IP)) (COL(5),A(6),A(2));
            IP=IP+1;
            PDL(IP)=LNN(NF,L);
            NXT=PDL(IP);
        IF (NXT¬=NT) THEN GO TO ENDNODE;
    /*                          */
    /* A CANDIDATE SIMPLE PATH IS OBTAINED. PATH
       INFORMATION IN ARRAYS PDL AND PTAG IS
       PRINTED OUT */
    /*                          */
            COUNT=COUNT+1;
            DECI=IP-1;
            PTAG(IP)=DECI;
            NPATH=NPATH+1;
            IF (COUNT¬=20) THEN GO TO COPRINT;
            COUNT=0;
PUT SKIP FILE(CNSECT1) EDIT('PATHS=',NPATH)
        (COL(1),A(6),F(4,0));
    COPRINT:;
            IF (IP>PATHLTH+1) THEN GO TO NOPRINT;
PUT SKIP FILE(CNSEOT1)
        EDIT((SITEN(PDL(J)),'(',PTAG(J),')' DO J=1 TO IP))
        (COL(5),(10)(A(8),A(1),A(2),A(1)));
    NOPRINT:;
    /*                          */
    /* POP OFF LAST SITE FROM PDL AND SET
       LASTNN EQUAL TO POPPED SITE */
    /*                          */
    POPNODE:;
            LASTNN=PDL(IP);
            PDL(IP)='000';
            PTAG(IP)='  ';
            IP=IP-1;
        IF (IP=0) THEN GO TO EXIT;
        ELSE GO TO POPNEMTY;

    EXIT:;
      PUT SKIP FILE(CNSEOT1) EDIT('NPATH=',NPATH)
              (COL(40),A(6),F(4,0));
      FREE PDL,PTAG;
    END PATH;


  /*****************************************************/
  /*    END EXTERNAL PROCEDURE OF PROCEDURE CNSE       */
  /*****************************************************/
```

3.3 DATA BASE UPDATING PROGRAM

The purpose of this program is to update status flags in the data base (file). These changes in the flags represent node and link outages and degradations which occur from nuclear weapons effects.

```
/*        UPDATE DATA BASE                                              */
/*                                                                      */
/*  THIS PROGRAM SERVES TO UPDATE THE DATA BASE RFBTA1 WHEN             */
/*  THE COMMUNICATIONS NETWORK UNDERGOES A CHANGE.                      */
/*  CHANGES ARE PROVIDED ON CARD INPUT IN THE FOLLOWING FORMS:          */
/*  NODE CHANGES:                                                       */
/*       COLUMN 1      S                                              */
/*       COLUMNS 3-10 (NAME OF SITE, DCA CODE CONVENTIONS)              */
/*       COLUMNS 11-13   (TYPE OF FACILITY, THREE LETTER CODE)          */
/*       COLUMN  15-19     (NEW FLAG VALUE)                             */
/*  IF COL 20 CONTAINS 'R', CONDITIONS SPECIFIED ARE RESET.             */
/*  LINK CHANGES:                                                       */
/*       COLUMNS 1      L                                              */
/*       COLUMNS 3-7     (LINK NUMBER, ICA CONVENTIONS)                 */
/*       COLUMNS 15-19      (NEW FLAG VALUE)                            */

/*    FLAG VALUES ARE:                                                  */
/*  E = EMP,      B = BLAST,                                            */
/*  J = JAMMING, M = MHD,  S = SATTELITE LINK ATTACK.                   */
/*                                                                      */
/*  IF COL 20 CONTAINS 'R', CONDITIONS SPECIFIED ARE RESET.             */
UPDATE: PROC OPTIONS(MAIN);
        %INCLUDE EXTDCL;*************************************************
/**********************************************************************/
/*      TOTAL 5/6 COMMANDS                                            */
/**********************************************************************/

       DCL 1 CMDS EXT,
             2 OPENM CHAR(5)  INIT('OPENM'),
             2 CLOSM CHAR(5)  INIT('CLOSM'),
             2 OPENV CHAR(5)  INIT('OPENV'),
             2 CLOSV CHAR(5)  INIT('CLOSV'),
             2 SEQRM CHAR(5)  INIT('SEQRM'),
             2 RESTM CHAR(5)  INIT('RESTM'),
             2 SEQRV CHAR(5)  INIT('SEQRV'),
             2 SERLV CHAR(5)  INIT('SERLV'),
             2 RESTV CHAR(5)  INIT('RESTV'),
             2 READM CHAR(5)  INIT('READM'),
             2 WRITM CHAR(5)  INIT('WRITM'),
             2 ADD_M CHAR(5)  INIT('ADD-M'),
             2 DEL_M CHAR(5)  INIT('DEL-M'),
             2 READV CHAR(5)  INIT('READV'),
             2 WRITV CHAR(5)  INIT('WRITV'),
             2 ADDVC CHAR(5)  INIT('ADDVC'),
             2 ADDVB CHAR(5)  INIT('ADDVB'),
             2 ADDVA CHAR(5)  INIT('ADDVA'),
             2 ADDVR CHAR(5)  INIT('ADDVR'),
             2 DELVD CHAR(5)  INIT('DELVD'),
             2 READR CHAR(5)  INIT('READR'),
             2 READD CHAR(5)  INIT('READD'),
             2 SEQWV CHAR(5)  INIT('SEQWV'),
             2 RQLOC CHAR(5)  INIT('RQLOC'),
             2 DEQUE CHAR(5)  INIT('DEQUE');

/**********************************************************************/
/*      TOTAL 5/6 CONSTANTS                                           */
/**********************************************************************/

       DCL 1 CONS EXT,
             2 STATUS   CHAR(4)  INIT('****'),
             2 REFER    CHAR(4)  INIT('LKXX'),
             2 ENDP     CHAR(4)  INIT('END.'),
             2 CK       CHAR(4)  INIT('****'),
             2 DBMOD    CHAR(6)  INIT('DBNAME'),
             2 TASKID   CHAR(8)  INIT('TASKNAME'),
             2 FILEID   CHAR(4)  INIT('FILE'),
```

```
            2 FUNCT     CHAR(5) INIT('TOTAL'),
            2 LKPATH    CHAR(8) INIT('LINKLKXX'),
            2 TOTAL     CHAR(5) INIT('TOTAL'),
            2 QUEST     CHAR(5) INIT('QUEST'),
            2 MPTOT     CHAR(5) INIT('MPTOT'),
            2 RSA       CHAR(4);

      DCL TFILES(12) CHAR(4) INIT((11)(1)'END.') EXT;
      DCL CONTROL CHAR(30) ALIGNED EXT;
      DCL ICAREA CHAR(256) ALIGNED EXT;
      DCL RLMLIST CHAR(228) EXT;
      DCL DBIO EXT ENTRY (FIXED BIN,FIXED BIN);
      DCL SIATANL EXT ENTRY (FIXED BIN,FIXED BIN);
**************************
        DCL 1 TOTAL_FILES,
            2 RMTR CHAR(4)INIT('RMTR'),
            2 RMLK CHAR(4)INIT('RMLK'),
            2 RMSI CHAR(4)INIT('RMSI'),
            2 RMCT CHAR(4)INIT('RMCT'),
            2 RVCT CHAR(4)INIT('RVCT'),
            2 RVTL CHAR(4)INIT('RVTL');
      DCL 1 TOTAL_LINKAGE_PATHS,
            2 RMTRLKCT CHAR(8)INIT('RMTRLKCT'),
            2 RMTRLKTL CHAR(8)INIT('RMTRLKTL'),
            2 RMCTLKCT CHAR(8)INIT('RMCTLKCT'),
            2 RMSILKCT CHAR(8)INIT('RMSILKCT'),
            2 RMSILKTL CHAR(8)INIT('RMSILKTL'),
            2 RMLKLKTL CHAR(8)INIT('RMLKLKTL');
      DCL 1 INFO_LINK,
            2 LINK_SITE   CHAR(11),
            2 LINK_TRUNK  CHAR(7),
            2 LINK_LINK   CHAR(5);
      DCL LINK_INFO CHAR(23) DEFINED INFO_LINK;

      DCL 1 INFO_TRUNK,
            2 TRUNK_SITE  CHAR(11),
            2 TRUNK_CIRCUIT CHAR(9);
      DCL TRUNK_INFO CHAR(20) DEFINED INFO_TRUNK;

      DCL 1 ELEM_RMLK,
            2 RMLKCTRL CHAR(8)INIT('RMLKCTRL'),
            2 RMLKFLAG CHAR(8)INIT('RMLKFLAG'),
            2 ENDPARM  CHAR(4)INIT('END.');
      DCL RMLK_ELEM CHAR(20) DEFINED ELEM_RMLK;

      DCL 1 ELEM_RMSI,
            2 RMSICTRL CHAR(8)INIT('RMSICTRL'),
            2 RMSIFLAG CHAR(8)INIT('RMSIFLAG'),
            2 ENDPARM  CHAR(4)INIT('END.');
      DCL RMSI_ELEM CHAR(20) DEFINED ELEM_RMSI;

      DCL 1 ELEM_RMCT,
            2 RMCTCTRL CHAR(8)INIT('RMCTCTRL'),
            2 RMCTFLAG CHAR(8)INIT('RMCTFLAG'),
            2 RMCTIDEN CHAR(8)INIT('RMCTIDEN'),
            2 RMCTXREF CHAR(8)INIT('RMCTXREF'),
            2 ENDPARM  CHAR(4)INIT('END.');
      DCL RMCT_ELEM CHAR(36) DEFINED ELEM_RMCT;

      DCL 1 ELEM_RVCT,
            2 RVCTRMSI CHAR(8)INIT('RVCTRMSI'),
            2 RVCTRMCT CHAR(8)INIT('RVCTRMCT'),
            2 ENDPARM  CHAR(4)INIT('END.');
      DCL RVCT_ELEM CHAR(20) DEFINED ELEM_RVCT;

      DCL 1 ELEM_RVTL,
            2 RVTLRMSI CHAR(8)INIT('RVTLRMSI'),
            2 RVTLRMTR CHAR(8)INIT('RVTLRMTR'),
            2 RVTLRMLK CHAR(8)INIT('RVTLRMLK'),
```

```
          2 ENDPARM  CHAR(4)INIT('END.');
      DCL RVTL_ELEM CHAR(28) DEFINED ELEM_RVTL;

    DCL CARD CHAR (50);
       DCL 1 SITE_CHANGE DEFINED CARD,
             2 CARD_TYPE CHAR(1),
             2 FILLERA   CHAR(1),
             2 SITE_ID   CHAR(11),
             2 FILLERB   CHAR(1),
             2 SITE_FLAG(5)  CHAR(1),
             2 INDICATOR     CHAR(1);

   DCL CHANGE_LINK CHAR (20);
      DCL 1 LINK_CHANGE DEFINED CHANGE_LINK,
             2 FILLERA     CHAR(2),
             2 LINK_ID     CHAR(5),
             2 FILLERB     CHAR(7),
             2 LINK_FLAG   CHAR(1);

    DECLARE NEW_FLAG BIT (8);

         DCL TRUNK_ID CHAR (7);
         DCL BUFFER CHAR (50);
         DCL 1 MLK DEFINED BUFFER,
               2 IKID CHAR (5),
               2 FLAG CHAR (1);

         DCL 1 MSI DEFINED BUFFER,
               2 SITE CHAR (11),
               2 FLAG CHAR (1);

         DCL 1 MTR DEFINED BUFFER,
               2 TRUNK     CHAR(7),
               2 FLAG      CHAR (1);

         DCL 1 MCT DEFINED BUFFER,
               2 CIRCUIT     CHAR(9),
               2 FLAG      CHAR (1);

         DCL CODE BIT (8);

         DCL OLD_LINKS(10) CHAR (5),
         N FIXED BINARY (15) INIT (0),
         (I, J) FIXED BINARY (15),
         (UNSPEC, BOOL) BUILTIN;

DCL TRUNK_REFER CHAR(4);
DCL LINK_REFER CHAR(4);
  DCL SITE_REFER CHAR(4);
       DCL CHARACTER CHAR (1);

/*    FLAG VALUES ARE:                                                */
/*   E = EMP,      B = BLAST,                                         */
/*  J = JAMMING, M = MHD,  S = SATTELITE LINK ATTACK.                 */
  DCL LNKPTH CHAR (8);
   DCL FLAGS (7) CHAR (1) INIT
             (' ', 'E', 'B', 'J', 'M', 'S');
      BITS (7) BIT(8) INIT ('00000000'B, '00000001'B, '00000010'B,
  '00000100'B, '00001000'B, '00010000'B);
   DCL W   BIT (4),
       OR  BIT (4) INIT ('0111'B),
       AND BIT (4) INIT ('0001'B),
       XOR BIT (4) INIT ('0110'B);
/*                                                                   */
/*  PROGRAM BEGINS BY OPENING FILES                                  */
/*                                                                   */
DCL CHANGIN INPUT FILE STREAM;
FUNCT=TOTAL;
TASKID='UPDATE';
```

```
 DBMOD='RFBTA1';
 CALL DBIO(5,0);

 FUNCT=OPENM;
 TFILES(1)=RMTR;
 TFILES(2)=RMIK;
 TFILES(3)=RMSI;
 TFILES(4)=RMCT;
 TFILES(5)=RVCT;
 TFILES(6)=RVTL;
 TFILES(7)=ENDP;
 CALL DBIO(4,0);

   ON ENDFILE (CHANGIN) GO TO STOP;
        ON ERROR BEGIN;
             FUNCT = DEQUE;
             CALL DBIO(4,0);
             STOP;
        END;

NEXT_CARD:
        FUNCT = RESTM;
        CALL DBIO(3,0);
 FUNCT=OPENM;
      CALL DBIO(4,0);
     GET FILE (CHANGIN) EDIT (CARD) (COL(1),A(50)) COPY;
      CHANGE_LINK = CARD;
        IF((CARD_TYPE ¬= 'A') &
           (CARD_TYPE ¬= 'S') &
           (CARD_TYPE ¬= 'L')) THEN
         DO;
         PUT SKIP LIST('INPUT ERROR, CARD TYPE =',CARD_TYPE);
         GO TO NEXT_CARD;
         END;

NEW_FLAG = '00000000'B;
        W = OR;
        DO J = 1 TO 5;
          DO I = 1 TO 7;
             IF(SITE_FLAG(J) = FLAGS(I)) THEN DO;
                NEW_FLAG = BOOL(NEW_FLAG, BITS(I), OR);
                GO TO END_I;
                END;
             END;
       PUT EDIT ('ILLEGAL FLAG ', SITE_FLAG(J)) (SKIP, A, A);
END_I: END;

        IF(INDICATOR = 'R') THEN DO;
             W = AND;
             NEW_FLAG =
                BOOL(NEW_FLAG, '11111'B, XOR);
             END;

     IF (CARD_TYPE='A') THEN CALL RESET_ALL;
     IF (CARD_TYPE='S') THEN CALL SET_SITE;
     IF (CARD_TYPE='L') THEN CALL SET_LINK;
        STATUS = OK;
        FUNCT=CLOSM;
        CALL DBIO(4,0);
        PUT SKIP LIST ('PREVIOUS CARD PROCESSING COMPLETED');
     GO TO NEXT_CARD;

RESET_ALL: PROCEDURE;
   NEW_FLAG = '00000000'B;
        UNSPEC(CHARACTER) = '00000000'B;
        CALL RESET_ALL_SITES;
        CALL RESET_ALL_LINKS;
        CALL RESET_ALL_TRUNKS;
        CALL RESET_ALL_CIRCUITS;
```

```
        END RESET_ALL;

RESET_ALL_SITES: PROC;
        FUNCT = RESTM;
        CALL DBIO(3,0);
        STATUS = OK;
        FUNCT = SEQRM;
        FILEID = RMSI;
        ELMLIST = 'RMSICTRL' || 'RMSIFLAG' || ENDP;
        CALL DBIO(6,0);
        DO WHILE (STATUS = OK);
        BUFFER = IOAREA;
        IF(UNSPEC(MSI.FLAG) ¬= '00000000'B) THEN
        DO;

             PUT SKIP EDIT ('MSI=', MSI) ((5)A);

             FUNCT = WRITM;
             CONTROL = MSI.SITE;
             ELMLIST = 'RMSIFLAG' || ENDP;
             IOAREA = CHARACTER;
             CALL DBIO(7,0);
             FUNCT = SEQRM;
             ELMLIST = 'RMSICTRL' || 'RMSIFLAG' || ENDP;
             END;
        CALL DBIO(6,0);
        END;
        END RESET_ALL_SITES;

RESET_ALL_LINKS: PROC;
        FUNCT = RESTM;
        CALL DBIO(3,0);
        STATUS = OK;
        FUNCT = SEQRM;
        FILEID = RMLK;
        ELMLIST = RMLKCTRL || RMLKFLAG || ENDP;
        CALL DBIO(6,0);
        DO WHILE (STATUS = OK);
        BUFFER = IOAREA;
        IF(UNSPEC(MLK.FLAG) ¬= '00000000'B) THEN
        DO;

             PUT SKIP EDIT ('MLK=', MLK) ((5)A);

             FUNCT = WRITM;
             CONTROL = MLK.LKID;
             ELMLIST = RMLKFLAG || ENDP;
             IOAREA = CHARACTER;
             CALL DBIO(7,0);
             FUNCT = SEQRM;
             ELMLIST = RMLKCTRL || RMLKFLAG || ENDP;
             END;
        CALL DBIO(6,0);
        END;
        END RESET_ALL_LINKS;

RESET_ALL_TRUNKS: PROC;
        FUNCT = RESTM;
        CALL DBIO(3,0);
        STATUS = OK;
        FUNCT = SEQRM;
        FILEID = RMTR;
        ELMLIST = 'RMTRCTRL' || 'RMTRFLAG' || ENDP;
        CALL DBIO(6,0);
        DO WHILE (STATUS = OK);
        BUFFER = IOAREA;
        IF(UNSPEC(MTR.FLAG) ¬= '00000000'B) THEN
        DO;
```

```
            PUT SKIP EDIT ('MTR=', MTR) ((5)A);

            FUNCT = WRITM;
            CONTROL = MTR.TRUNK;
            ELMLIST = 'RMTRFLAG' || ENDP;
            IOAREA = CHARACTER;
            CALL DBIO(7,0);
            FUNCT = SEQRM;
            ELMLIST = 'RMTRCTRL' || 'RMTRFLAG' || ENDP;
            END;
         CALL DBIO(6,0);
         END;
         END RESET_ALL_TRUNKS;

RESET_ALL_CIRCUITS: PROC;
         FUNCT = RESTM;
         CALL DBIO(3,0);
         STATUS = OK;
         FUNCT = SEQRM;
         FILEID = RMCT;
         ELMLIST = RMCTCTRL || RMCTFLAG || ENDP;
         CALL DBIO(6,0);
         DO WHILE (STATUS = OK);
         BUFFER = IOAREA;
         IF(UNSPEC(MCT.FLAG) ¬= '00000000'B) THEN
         DO;

            PUT SKIP EDIT ('MCT=', MCT) ((5)A);

            FUNCT = WRITM;
            CONTROL = MCT.CIRCUIT;
            ELMLIST = RMCTFLAG || ENDP;
            IOAREA = CHARACTER;
            CALL DBIO(7,0);
            FUNCT = SEQRM;
            ELMLIST = RMCTCTRL || RMCTFLAG || ENDP;
            END;
         CALL DBIO(6,0);
         END;
         END RESET_ALL_CIRCUITS;


SET_SITE: PROCEDURE;
         PUT SKIP EDIT ('SITE=',SITE_ID) (COL(10),A,A);
         FUNCT   = READM;
         FILEID  = RMSI;
         CONTROL = SITE_ID;
         ELMLIST = RMSIFLAG || ENDP;
         CALL DBIO(7,0);
         FUNCT = WRITM;
         CHARACTER = IOAREA;
         UNSPEC (CHARACTER)  = BOOL(NEW_FLAG, UNSPEC (CHARACTER), W);
         IOAREA = CHARACTER;
         CALL DBIO(7,0);
         PUT SKIP EDIT ('LINKS WHICH START HERE') (COL(12), A);
         N=0;
         LNKPTH = RMSILKTL;
         REFER  = 'LKTL';
         CALL READ_TL_GIVEN_SITE;
         DO WHILE (REFER ¬= ENDP);
           SITE_REFER = REFER;
           CALL SET_LINK;
           REFER = SITE_REFER;
           CALL READ_TL_GIVEN_SITE;
           END;
/*
         NOW GET LINKS THAT TERMINATE HERE.
                                                    */
         PUT SKIP EDIT ('LINKS WHICH END HERE') (COL(12), A);
```

```
          LNKPTH = 'RMSILKTS';
          REFER  = 'LKTS';
          CALL READ_TL_GIVEN_SITE;
          DO WHILE (REFER ¬= ENDP);
            SITE_REFER = REFER;
            CALL SET_LINK;
            REFER = SITE_REFER;
            CALL READ_TL_GIVEN_SITE;
            END;
          END SET_SITE;

READ_TL_GIVEN_SITE: PROCEDURE;
          FUNCT   = READV;
          FILEID  = RVTL;
          LKPATH  = LNKPTH;
          ELMLIST = RVTL_ELEM;
          CONTROL = SITE_ID;
          CALL DBIO(9,0);
          LINK_INFO = IOAREA;
          TRUNK_ID = LINK_TRUNK;
          LINK_ID = LINK_LINK;
   /*
      CHECK IF WE'VE ALREADY PROCESSED THIS LINK FOR THIS SITE.
                                                    */
          DO WHILE (REFER ¬= ENDP);
               NTFD = 0;
               IF N < 1 THEN DO;
                         N = 1;
                         OLD_LINKS(N) = LINK_ID;
                         RETURN;
                         END;
                    ELSE DO I = 1 TO N WHILE (NTFD = 0);
                         IF OLD_LINKS(I) = LINK_ID THEN NTFD = 1;
                         END;
                    IF NTFD = 1 THEN CALL DBIO(9,0);
                         ELSE IF N >= 10 THEN RETURN;
                              ELSE DO;
                                   N = N + 1;
                                   OLD_LINKS(N) = LINK_ID;
                                   RETURN;
                                   END;
               END;
          END READ_TL_GIVEN_SITE;


SET_LINK: PROCEDURE;
          PUT SKIP EDIT ('LINK=',LINK_ID) (COL(15),A,A);
          FUNCT   = READM;
          FILEID  = RMLK;
          CONTROL = LINK_ID;
          ELMLIST = RMLKFLAG || ENDP;
          CALL DBIO(7,0);
          FUNCT = WRITM;
          CHARACTER = IOAREA;
          UNSPEC (CHARACTER)  = BOOL(NEW_FLAG, UNSPEC (CHARACTER), W);
          IOAREA = CHARACTER;
          CALL DBIO(7,0);
          REFER  = 'LKTL';
          CALL READ_TL_GIVEN_LINK;
          DO WHILE (REFER ¬= ENDP);
          PUT SKIP EDIT ('TRUNK=',TRUNK_ID) (COL(20),A,A);
             FUNCT   = READM;
             FILEID  = RMTR;
             CONTROL = TRUNK_ID;
             ELMLIST = 'RMTRFLAG' || ENDP;
             CALL DBIO(7,0);
             FUNCT = WRITM;
             CHARACTER = IOAREA;
             UNSPEC (CHARACTER) = BOOL(NEW_FLAG, UNSPEC (CHARACTER), W);
```

```
            IOAREA = CHARACTER;
            CALL DBIO(7,0);
            LINK_REFER = REFER;
            CALL SET_CIRCUITS_GIVEN_TRUNK;
            REFER    = LINK_REFER;
            CALL READ_TL_GIVEN_LINK;
            END;
        END SET_LINK;

READ_TL_GIVEN_LINK: PROCEDURE;
        FUNCT   = READV;
        FILEID  = RVTL;
        LKPATH  = RMLKLKTL;
        CONTROL = LINK_ID;
        ELMLIST = RVTLRMTR || ENDP;
        CALL DBIO(9,0);
        TRUNK_ID = IOAREA;
        END READ_TL_GIVEN_LINK;

SET_CIRCUITS_GIVEN_TRUNK: PROCEDURE;
        REFER   = 'LKCT';
        DO WHILE(REFER ¬= ENDP);
            FUNCT   = READV;
            FILEID  = RVCT;
            LKPATH  = RMTRLKCT;
            CONTROL = TRUNK_ID;
            ELMLIST = RVCTRMCT || ENDP;
            CALL DBIO(9,0);
            IF (REFER ¬= ENDP) THEN
            DO;
            PUT SKIP EDIT ('CIRCUIT=',IOAREA) (COL(25),A,A(9));
                CONTROL = IOAREA;
                FUNCT = READM;
                FILEID = RMCT;
                ELMLIST = RMCTFLAG || ENDP;
                CALL DBIO(7,0);
                FUNCT = WRITM;
                CHARACTER = IOAREA;
                UNSPEC (CHARACTER) =
                        BOOL(NEW_FLAG, UNSPEC (CHARACTER), W);
                IOAREA = CHARACTER;
                CALL DBIO(7,0);
                END;
            END;
        END SET_CIRCUITS_GIVEN_TRUNK;

STOP:
EXIT_ROUTINE:
   PUT SKIP LIST ('ALL CHANGES COMPLETED');
   CLOSE FILE (CHANGIN);
   FUNCT=CLOSM;
   CALL DBIO(4,0);
   FUNCT=DEQUE;
   CALL DBIO(4,0);
   END UPDATE;
```

3.4 UTILITY PROGRAM

DBUTIL (Data Base Utility Program) is a general maintenance facility for the TOTAL data base. It provides a mechanism by which a user can add circuits, trunks, links, or sites to the data base; delete circuits, trunks, links, or sites; change any data fields associated with circuits, trunks, links, or sites; or change the relationships among the various circuits, trunks, links, or sites. This section contains a general description of the capabilities of DBUTIL. A program listing appears in Paragraph 3.4.6.

3.4.1 Adding Records

To add a site master record, the user must specify the control key, and may optionally specify any data fields contained in the record. Any unspecified data fields are left blank, except for the flag field which is (initially) set to indicate that the site has not been knocked out; this rule also applies to links, trunks, and circuits.

To add a link, the user must specify the link control key and the control keys of the two sites joined by the link. Optionally, he may specify any of the data fields contained in the link record except for the location and facility fields, which are automatically set to correspond to the specified sites. If the specified sites are not already present in the data base, they are added.

To add a trunk, the user must specify the trunk control key and the ordered set of links over which the trunk is routed. Optionally, he may specify any of the data fields contained in the trunk record except for the location and facility fields, which are derived from the endpoints of the specified links and set automatically. The specified links must already be present in the data base and must trace a continuous path, in the order specified, from one end of the trunk to the other. If these conditions are satisfied, the trunk master record is added to the data base; an ordered set of variable records (corresponding to the specified links) is also added, providing the desired relationships among the trunk, links, and sites.

Adding a circuit is similar to adding a trunk. To add a circuit, the user must specify the circuit control key and the ordered set of trunks over which the circuit is routed. Optionally, he may specify any of the data fields contained in the circuit record, except for the location and facility fields, which are derived from the end-points of the specified trunks and set automatically. These trunks must already be present in the data base and must trace a continuous path, in the order specified, from one end of the circuit to the other. When these conditions are satisfied, the circuit master record is added to the data base; an ordered set of variable records (corresponding to the specified trunks) is also added, providing the desired relationships among the circuit, trunks, and sites.

3.4.2 Deleting Records

A user of DBUTIL can delete any circuit, trunk, link, or site by specifying the control key of the record to be deleted. Because sites and links are actual physical entities, whereas trunks and circuits are logical concepts, their deletions are handled in different manners. Although deletion of a site or link is likely to be permanent, a "deleted" circuit or trunk will often reappear with most of the same data fields but with different routing specified.

When a user specifies the deletion of a circuit, the variable records specifying the routing of the circuit over a set of trunks are deleted from the data base. The circuit master record remains in the data base but is "flagged for deletion." That is, unless the user specifies the addition of another circuit with the identical control key later during the same invocation of DBUTIL, the flagged record will be physically deleted from the data base during a final cleanup step in the program. If the circuit is subsequently added again, any data fields not respecified retain their previous values. The user thus has the ability to add a circuit twice and delete it without having to respecify the data fields.

Similar processing is done for deletion of a trunk with the additional restriction that a trunk cannot be deleted unless it is not being used by circuits. Before deleting a trunk, a user must first delete all circuits riding the trunk.

Before deleting a link, a user must first delete all trunks using the link. Unlike deletion of circuits and trunks, however, a request to delete a link results in the immediate physical deletion of the specified link master record.

As with links, a request to delete a site results in the immediate physical deletion of the specified site master record. Also, before deleting a site, a user must first delete all trunks passing through the site. In addition, all links that have the specified site as an endpoint should also be deleted. However, since it is inconvenient to verify that this last condition has been satisfied, DBUTIL instead will honor only other "delete" requests following the successful deletion of a site . Further, a successful site deletion enables a subsequent "garbage collection" step during which all links and sites used by no trunks are deleted. These precautions are taken to ensure the user does not attempt to route a trunk over a link with deleted endpoints.

3.4.3 Rerouting Trunks

Since a common maintenance operation is the rerouting of a trunk over a new set of links without changing its endpoints or affecting the trunk routing of all affected circuits, such a capability has been built into DBUTIL. To reroute a trunk, the user must specify only the trunk control key and the ordered set of links over which the trunk is to be routed. The specified links must trace a continuous path, and the new endpoints must be the same as the old ones. If the conditions are satisfied, then the old variable records associating the trunk and links are deleted and a new set, reflecting the rerouting, is added.

3.4.4 Changing Data Fields

DBUTIL provides the user with the ability to modify any of the data fields that can be specified during an add request. One operation is provided to allow the user to update any of the data fields contained in a master record except for the location and facility fields, which are derived from and must correspond to the routing information reflected by the variable records.

Since it is occasionally necessary to modify those fields directly, a separate operation is provided to permit updating the location and facility fields of circuit, trunk, and link master records.

3.4.5 Garbage Collection

The "garbage collection" step discussed in connection with deletion of sites can also be enabled directly by the user. If specified by the user, DBUTIL will delete from the data base any link and sites not used by circuits or trunks, thus reducing the amount of data stored and freeing space for future additions.

3.4.6 DBUTIL PROGRAM LISTING

```
/*************************************************************************/
/* STATIC                                                                */
/* STRUCTURE:                                                            */
/*            DBUTIL                                                     */
/*              CONDITION(TOTERR)                                        */
/*              AC_CIRCUIT                                               */
/*                 SET_UPDATE_C                                          */
/*              AC_TRUNK                                                 */
/*                 SET_UPDATE_T                                          */
/*              AC_LINK                                                  */
/*              AC_SITES                                                 */
/*              DEL_CIRCUIT                                              */
/*              DEL_TRUNK                                                */
/*              DEL_LINK                                                 */
/*              DEL_SITES                                                */
/*              RE_TRUNK                                                 */
/*              CL_CIRCUIT                                               */
/*              CL_TRUNK                                                 */
/*              CL_LINK                                                  */
/*              ROUTE_TRACE                                              */
/*              ADD_ROUTING                                              */
/*              READ_REQUEST                                             */
/*                 READ_ADDCHANGE                                        */
/*                 READ_DELETE                                           */
/*                 READ_REROUTE                                          */
/*                 READ_CHANGELOC                                        */
/*                 READ_SUBFIELDS                                        */
/*                 READ_ROUTING                                          */
/*                    NEXTKEY                                            */
/*                 READ_SITE_DATA                                        */
/*                 SETFLAG                                               */
/*              FLUSH                                                    */
/*               (FLUSH_ALL     -- ENTRY POINT)                          */
/*               (FLUSH_REQUEST -- ENTRY POINT)                          */
/*              INPUT                                                    */
/*                 CONDITION(ENDFILE)                                    */
/*                 NEXTCHAR                                              */
/*                    GETCHAR                                            */
/*                 PRINT_INPUT                                           */
/*                  (PRINT_INREC -- ENTRY POINT)                         */
/*                  (PRINT_REQ   -- ENTRY POINT)                         */
/*              MSG                                                      */
/*              GARBAGE_COLLECT                                          */
/*                                                                       */
/*************************************************************************/
```

```
/*********************************************************************/
/*                                                                   */
/* PROGRAM MAINTENANCE RECORD:                                       */
/*              VERSION.RELEASE --DATE---  NAME                      */
/*                    01.01      06 OCT 77  M. J. BASSMAN(1ST RELEASE)*/
/*                    02.01      17 OCT 77  M. J. BASSMAN            */
/*                    02.02      18 OCT 77  M. J. BASSMAN            */
/*                    02.03      19 OCT 77  M. J. BASSMAN            */
/*                    03.01      28 OCT 77  M. J. BASSMAN            */
/*                    03.02      29 OCT 77  M. J. BASSMAN            */
/*                    03.03      02 NOV 77  M. J. BASSMAN            */
/*                                                                   */
/*********************************************************************/



DBUTIL: PROCEDURE(PARM) OPTIONS(MAIN);

/* INPUT PARAMETER */
DECLARE PARM CHARACTER(100) VARYING;

/* CHANGE THIS FIELD WHEN NEW VERSION COMPILED:                      */
DECLARE VERSION_AND_DATE CHARACTER(24) /*VERSION XX.XX  DD MON YY*/
                                 INITIAL('VERSION 03.03  02 NOV 77');

/* JOB STEP RETURN CODE */
DECLARE RC   BINARY FIXED(31,0) INITIAL(0);



/* DECLARATIONS FROM THE TOTAL SOURCE LIBRARY (SYS9.TOTAL.PL1DBIO) */

% INCLUDE EXTDCL;
/* END OF %INCLUDE-D DECLARATIONS */
```

```
/*  IOAREA FORMATS FOR SITE, LINK, TRUNK, AND CIRCUIT MASTER FILES  */
/*  (THESE STRUCTURES ARE BASED SO THAT THEY CAN OCCUPY THE SAME    */
/*  STORAGE AS IOAREA.  IOAREAP WILL BE SET EQUAL TO ADDR(IOAREA).  */

DECLARE
        1 IOSITE        BASED(IOAREAP),
         2 RMSICTRL,
          3 GEOLOC      CHARACTER(8),
          3 FACILITY    CHARACTER(3),
         2 RMSISTCT     CHARACTER(2),
         2 RMSIFLAG     CHARACTER(1),
         2 RMSICORD     CHARACTER(15), /* LATITUDE (7), LONGITUDE(8) */
         2 RMSINUMB     CHARACTER(3),
         2 RMSIFILL     CHARACTER(7),

        1 IOLINK        BASED(IOAREAP),
         2 RMLKCTRL     CHARACTER(5),
         2 RMLKFLAG     CHARACTER(1),
         2 RMLKTRAN     CHARACTER(3),
         2 RMLKCODE     CHARACTER(2),
         2 RMLKFILL     CHARACTER(2),
         2 LINKSITE1,
          3 RMLKFLOC    CHARACTER(8),
          3 RMLKFFAC    CHARACTER(3),
         2 LINKSITE2,
          3 RMLKTOLC    CHARACTER(8),
          3 RMLKTFAC    CHARACTER(3),

        1 IOTRUNK       BASED(IOAREAP),
         2 RMTRCTRL,
          3 TRUNK_ID    CHARACTER(6),
          3 SA          CHARACTER(1),
         2 RMTRFLAG     CHARACTER(1),
         2 RMTRPCAT     CHARACTER(1),
         2 RMTRTCAP     CHARACTER(4),
         2 RMTRBAND     CHARACTER(5),
         2 RMTRAVAL     CHARACTER(3),
         2 RMTRFILL     CHARACTER(4),
         2 TRUNKSITE1,
          3 RMTRFLOC    CHARACTER(8),
          3 RMTRFFAC    CHARACTER(3),
         2 TRUNKSITE2,
          3 RMTRTOLC    CHARACTER(8),
          3 RMTRTFAC    CHARACTER(3),
```

```
        1 IOCIRCUIT     BASED(IOAREAP),
          2 RMCTCTRL,
            3 CCSD      CHARACTER(8),
            3 SA        CHARACTER(1),
          2 RMCTFLAG    CHARACTER(1),
          2 RMCTRSTP    CHARACTER(2),
          2 RMCTRAPL    CHARACTER(1),
          2 RMCTIDEN    CHARACTER(1),
          2 RMCTXREF    CHARACTER(8),
          2 RMCTFILL    CHARACTER(4),
          2 CIRCUITSITE1,
            3 RMCTFRLC  CHARACTER(8),
            3 RMCTFFAC  CHARACTER(3),
          2 CIRCUITSITE2,
            3 RMCTTOLC  CHARACTER(8),
            3 RMCTTFAC  CHARACTER(3);

/*  IOAREA FORMATS FOR VARIABLE FILES */
DECLARE
        1 IORVTL        BASED(IOAREAP),
          2 RVTLRMTR    CHARACTER(7),
          2 RVTLRMLK    CHARACTER(5),
          2 RVTLRMSI    CHARACTER(11),
          2 RVTLTOLC    CHARACTER(11),

        1 IORVCT        BASED(IOAREAP),
          2 RVCTRMCT    CHARACTER(9),
          2 RVCTRMTR    CHARACTER(7),
          2 RVCTRMSI    CHARACTER(11),
          2 RVCTTOLC    CHARACTER(8);

DECLARE IOAREAP         POINTER; /* =ADDR(IOAREA) */

/* FOR TEMPORARY STORAGE WHEN IOAREA IS REQUIRED FOR ANOTHER ACCESS */
DECLARE IOSAVE CHARACTER(200) ALIGNED;
```

```
/*************************************************************/
/* ELEMENT LIST DESCRIPTORS FOR BASE OF SPECIFICATION */
/*************************************************************/
DECLARE
        RMSI_ELEM CHARACTER(52) INITIAL(
'RMSICTRLRMSISTCTRMSIFLAGRMSICORDRMSINUMBRMSIFILLEND.'),

        RMLK_ELEM CHARACTER(76) INITIAL(
'RMLKCTRLRMLKFLAGRMLKTRANRMLKCODERMIKFILLRMLKFRLCRMLKFFACRMLKTCLCRMLKTF
ACEND.'),

        RMTR_ELEM CHARACTER(92) INITIAL(
'RMTRCTRLRMTRFLAGRMTRRCATRMTRTCAPRMTRBANDRMTRAVALRMTRFILLRMTRFRLCRMTRFF
ACRMTRTCLCRMTRTFACEND.'),

        RMCT_ELEM CHARACTER(92) INITIAL(
'RMCTCTRLRMCTFLAGRMCTRSTPRMCTRAPIRMCTIDENRMCTXREFRMCTFILLRMCTFRLCRMCTFF
ACRMCTTCLCRMCTTFACEND.'),

        RVTL_ELEM CHARACTER(36) INITIAL(
'RVTLRMTRRVTLRMLKRVTLRMSIRVTLTCLCEND.'),

        RVCT_ELEM CHARACTER(36) INITIAL(
'RVCTRMCTRVCTRMTRRVCTRMSIRVCTTCLCEND.'),

        RMLK_LOCS CHARACTER(44) INITIAL(
'RMLKCTRLRMLKFRLCRMLKFFACRMLKTCICRMLKIFACEND.'),

        RMTR_LOCS CHARACTER(44) INITIAL(
'RMTRCTRLRMTRFRLCRMTRFFACRMTRTCLCRMTRTFACEND.'),

        RMCT_LOCS CHARACTER(44) INITIAL(
'RMCTCTRLRMCTFRLCRMCTFFACRMCTTCLCRMCTTFACEND.');
```

```
/* FIELDS FOR USER'S INPUT SPECIFICATIONS */
DECLARE
        1 INDATA,
         2 SITE(1:2),
          3 RMSICTRL,
           4 GEOLOC       CHARACTER(8),
           4 FACILITY     CHARACTER(3),
          3 RMSISTCT      CHARACTER(2),
          3 RMSIFLAG      CHARACTER(1),
          3 RMSICORD      CHARACTER(15), /* LATITUDE (7), LONGITUDE (8) */
          3 RMSINUMB      CHARACTER(3),
          3 RMSIFILL      CHARACTER(7),

         2 LINK,
          3 RMLKCTRL      CHARACTER(5),
          3 RMLKFLAG      CHARACTER(1),
          3 RMLKTRAN      CHARACTER(3),
          3 RMLKCODE      CHARACTER(2),
          3 RMLKFILL      CHARACTER(2),

         2 TRUNK,
          3 RMTRCTRL,
           4 TRUNK_ID     CHARACTER(6),
           4 SA           CHARACTER(1),
          3 RMTRFLAG      CHARACTER(1),
          3 RMTRECAT      CHARACTER(1),
          3 RMTRTCAP      CHARACTER(4),
          3 RMTRBAND      CHARACTER(5),
          3 RMTRAVAL      CHARACTER(3),
          3 RMTRFILL      CHARACTER(4),

         2 CIRCUIT,
          3 RMCTCTRL,
           4 CCSD         CHARACTER(8),
           4 SA           CHARACTER(1),
          3 RMCTFLAG      CHARACTER(1),
          3 RMCTRSTP      CHARACTER(2),
          3 RMCTRAPL      CHARACTER(1),
          3 RMCTIDEN      CHARACTER(1),
          3 RMCTXREF      CHARACTER(8),
          3 RMCTFILL      CHARACTER(4),

         2 CHANGE_LOCS,
          3 FSLC          CHARACTER(8),
          3 FFAC          CHARACTER(3),
          3 TOLC          CHARACTER(8),
          3 TFAC          CHARACTER(3);
```

```
/******************/
/* ROUTING LIST */
/******************/
DECLARE 1 ROUTE_CHAIN  BASED(RPTR),
          2 RKEY       CHARACTER(7),
          2 RSITE1     CHARACTER(11),
          2 RSITE2     CHARACTER(11),
          2 NEXTR      POINTER,

        RPTR           POINTER,
        LASTR          POINTER,
        ROUTESPEC      POINTER;


/********************/
/* DELETION LISTS */
/********************/
DECLARE 1 DTRUNK       BASED(DTPTR),
          2 DTKEY      CHARACTER(7),
          2 NEXTDT     POINTER,

        1 DCIRCUIT     BASED(DCPTR),
          2 DCKEY      CHARACTER(9),
          2 NEXTDC     POINTER,

        DTPTR          POINTER,
        DCPTR          POINTER,
        DEL_TR_PTR     POINTER,
        DEL_CT_PTR     POINTER;
```

```
/* MAIN PROCESSING VARIABLES */

DECLARE
        REQUEST         CHARACTER(1), /* @, -, #, % */
        REQUEST_LEVEL   CHARACTER(1), /* C, T, L, S */

        ADDCHANGE       CHARACTER(1)  INITIAL('@'),
        DELETE          CHARACTER(1)  INITIAL('-'),
        REROUTE         CHARACTER(1)  INITIAL('#'),
        CHANGELOC       CHARACTER(1)  INITIAL('%'),
        NEXT            CHARACTER(1)  INITIAL(' '),

        /* ARGUMENTS TO SUBROUTINE INPUT */
        SEPARATOR       CHARACTER(1),
        ITEM            CHARACTER(100) VARYING,

        /* THIS IDENTIFIER IS USED BY SUBROUTINE INPUT ONLY.  IT MUST*/
        /* BE GLOBAL BECAUSE IT PASSES INFORMATION FROM ONE          */
        /* INVOCATION OF INPUT TO THE NEXT.                          */
        NEXT_SEPARATOR CHARACTER(1) INITIAL(' '),

        VALID_REQ_SEP CHARACTER(4)  INITIAL('@-#%'),
        VALID_END_SEP CHARACTER(6)  INITIAL('@-#% ;'),

        KEY             CHARACTER(9) VARYING, /* CONTROL FOR      */
                                              /* CURRENT REQUEST  */

        /* WORK FIELDS FOR ENDPOINTS OF CURRENT */
        /* CIRCUIT, TRUNK, OR LINK              */
        WKSITE1         CHARACTER(11),
        WKSITE2         CHARACTER(11),

        DFIELD(6)       CHARACTER(15), /* DATA SUBFIELDS */
        ZS              CHARACTER(1); /* '000000000'B (DEFAULT FLAG) */
```

```
        /**********/
DECLARE /* FLAGS */
        /**********/

        UPDATESITE(2) BIT(1),
        UPDATELINK    BIT(1),
        UPDATETRUNK   BIT(1),
        UPDATECIRCUIT BIT(1),

        DELETE_ONLY   BIT(1) INITIAL('0'B),
        GARBCCL       BIT(1) INITIAL('0'B),
        EOF           BIT(1) INITIAL('0'B);

DECLARE
        DATE_AND_TIME CHARACTER(17), /* PRINTED ON EACH PAGE */
        PAGE_NUM  BINARY FIXED(15,0) INITIAL(1), /* FOR PAGE HEADING*/
        MSGC      BINARY FIXED(15,0) INITIAL(1), /*STRT CCL FOR MSGS*/
        MSG_COUNT BINARY FIXED(15,0) INITIAL(0), /* # MSGS >= WARNING*/
        #_DELS    BINARY FIXED(15,0); /* CIRCUIT AND TRUNK CLEANUP */

DECLARE /* REQUEST ATTEMPT COUNTERS */
        #_ADDCHANGES BINARY FIXED(15,0) INITIAL(0),
        #_DELETES    BINARY FIXED(15,0) INITIAL(0),
        #_REROUTES   BINARY FIXED(15,0) INITIAL(0),
        #_CHANGELCCS BINARY FIXED(15,0) INITIAL(0);

DECLARE OUTPIC       PICTURE'ZZZ9'; /* FOR FORMATTING COUNTERS */



/* THESE ARE "PARAMETERS" THAT CONTROL INPUT RECORD PROCESSING */
/* AND OUTPUT LISTING.  IDEALLY, THEY SHOULD BE SPECIFIABLE AS */
/* EXECUTION-TIME OPTIONS.  SEE SUBROUTINES PRINT_INPUT AND    */
/* GETCHAR.                                                    */
DECLARE
        LINES     BINARY FIXED(15,0) INITIAL(75),
        BUFPOS    BINARY FIXED(15,0),
        BUFBEGIN  BINARY FIXED(15,0),
        BUFEND    BINARY FIXED(15,0),
        INRECLEN  BINARY FIXED(15,0),
        INREC#    BINARY FIXED(15,0),
        COL#      BINARY FIXED(15,0), /* STRT COL FOR INREC# */
        COLREC    BINARY FIXED(15,0); /* STRT CCL FOR INREC */

DECLARE INREC(1:11) CHAR(80);/*TEMP DCL...CHANGE BLOCK STRUCTURE*/
DECLARE INREC_LEVEL BINARY FIXED(15,0) INITIAL(0);
```

```
        /*************************/
DECLARE /* BUILT-IN FUNCTIONS */
        /*************************/
       (ADDR,
        CHAR,
        DATE,
        DIM,
        DIVIDE,
        INDEX,
        LENGTH,
        LINENO,
        MAX,
        NULL,
        PLIRETC,
        STRING,
        SUBSTR,
        TIME,
        UNSPEC,
        VERIFY) BUILTIN;

        /**********/
        /* FILES */
        /**********/
DECLARE SYSIN    FILE STREAM INPUT,
        SYSPRINT FILE STREAM PRINT;
        /* FILE(GCWORK) APPEARS ONLY IN SUBROUTINE GARBAGE_COLLECT */


        /*****************************/
        /* USER-DEFINED CONDITION */
        /*****************************/
DECLARE TOPERR   CONDITION;
```

```
/*********************/
/* I. PRELIMINARIES */
/*********************/

/* (1) PREPARE PRINT FILE */

/* (LET SYSTEM HANDLE UNDEFINEDFILE CONDITION.) */

OPEN FILE(SYSPRINT) LINESIZE(132) PAGESIZE(LINES);

/* PAGE TURNER AND HEADING PRINTER */
ON ENDPAGE(SYSPRINT)
   BEGIN;
      PUT FILE(SYSPRINT) EDIT('DBUTIL '||VERSION_AND_DATE,
                              DATE_AND_TIME,'PAGE ',PAGE_NUM)
                             (PAGE,A(32),COL(61),A(17),
                              COL(112),A(5),F(4));
      PUT FILE(SYSPRINT) SKIP(2);
      PAGE_NUM=PAGE_NUM+1;
   END;

/* INITIALIZE DATE AND TIME AND PREPARE FIRST PAGE */
/* THIS BLOCK INITIALIZES THE DATE/TIME STAMP IN   */
/* THE FORMAT                                      */
/*            'DD MON YY   HH:MM'                  */
BEGIN;
   DECLARE 1 STAMP, /* DATE AND TIME STAMP */
             2 (Y,M,D,HR,MIN) CHARACTER(2),

           MONTHS(12) CHARACTER(3) INITIAL('JAN',
                                           'FEB',
                                           'MAR',
                                           'APR',
                                           'MAY',
                                           'JUN',
                                           'JUL',
                                           'AUG',
                                           'SEP',
                                           'OCT',
                                           'NOV',
                                           'DEC');

   STRING(STAMP)=DATE||TIME;
   DATE_AND_TIME=D||' '||MONTHS(M)||' '||Y||'   '||HR||':'||MIN;
END;

SIGNAL ENDPAGE(SYSPRINT);
```

```
/* I. (2) MISCELLANEOUS INITIALIZATIONS */

/* INPUT PROCESSING AND OUTPUT LISTING: */
/* (IDEALLY, THIS SHOULD BE SPECIFIABLE AS AN EXECUTION-TIME */
/* OPTION.  THE VALUES SPECIFIED HERE SHOULD BE THE DEFAULTS.*/
/* CURRENTLY, THE INPUT MUST BE IN FIXED LENGTH EIGHTY-BYTE  */
/* RECORDS.  ONLY COLUMNS 1 THROUGH 72 ARE SIGNIFICANT.      */
/* COLUMNS 73 THROUGH 80 ARE IGNORED AND CAN BE USED FOR     */
/* SEQUENCE NUMBERS.)                                        */
BUFBEGIN=1;
BUFEND  =72;
BUFPOS  =BUFEND;
INRECLEN=80;
INREC#  =1;
COL#    =1;
COLREC  =7;

MSGC    =COLREC;  /* <-- */

/* ALL ZERO BYTE */
UNSPEC(ZB)='00000000'B;

/* LIST PROCESSING HEAD POINTERS */
ROUTESPEC=NULL;
DEL_TR_PTR=NULL;
DEL_CT_PTR=NULL;

/* PROCESS PARAMETERS */
IF LENGTH(PARM)>0 THEN
   DO;
      CALL MSG(0,'PARAMETERS SPECIFIED: '||PARM);
      IF INDEX(PARM,'GARBCOL')^=0 THEN GARBCOL='1'B;
   END;


/* ESTABLISH ADDRESSABILITY FOR IOAREA FORMATS */
IOAREAP=ADDR(IOAREA);
```

```
/* I. (3) ESTABLISH COMMUNICATIONS WITH TOTAL */

/* TOTERR --                                                          */
/*       AN UNEXPECTED STATUS WAS RETURNED BY TOTAL.  DISPLAY A       */
/* MESSAGE ABOUT THE STATUS RETURNED AND ABOUT WHAT WAS BEING         */
/* ATTEMPTED.  FLUSH THE INPUT STREAM, PRINTING THE REMAINDER OF      */
/* THE USER'S INPUT.  THEN, WRAP UP THE PROGRAM.                      */

ON CONDITION(TOTERR)
   BEGIN;
      CALL MSG(0,'**** TOTAL: FUNCT='||FUNCT||',  STATUS='||STATUS||
                        ',  FILEID='||FILEID||',  REFER='||REFER||
                        ',  LKPATH='||LKPATH||',  CONTROL='||CONTROL);
      CALL MSG(16,'UNEXPECTED STATUS RETURN FROM TOTAL.');
      /* SUBROUTINE MSG FLUSHES INPUT STREAM AND TRANSFERS TO */
      /* TERMERR AS A RESULT OF THE TERMINAL ERROR CODE 16.   */

      /*<DEBUG CODE> THIS SHOULD NOT HAPPEN */
      CALL MSG(12,'UNEXPECTED RETURN FROM MSG TO TOTERR.');
      GO TO TERMERR;
   END;

/* PREPARE TOTAL.  LET STATUS_ANALYSIS ROUTINE (VIA DBIO) HANDLE */
/* TERMINATION IF "TOTAL" OR "OPENM" FAILS.                      */
FUNCT='TOTAL';
TASKID='DBUTIL';
DBMOD='DBETA1';
CALL DBIO(5,0);
FUNCT=OPENM;
TFILES(1)='RMSI';
TFILES(2)='RMLK';
TFILES(3)='RMTR';
TFILES(4)='RMCT';
TFILES(5)='RVTL';
TFILES(6)='RVCT';
CALL DBIO(4,0);
```

```
/* I. (4) PREPARE INPUT FILE */
ON UNDEFINEDFILE(SYSIN)
   BEGIN;
      EOF='1'B;
      SEPARATOR=' ';
      NEXT_SEPARATOR=' ';
      CALL MSG(4,'INCOMPLETE OR NO DD STATEMENT FOR FILE SYSIN.');
      /* PROCEED WITH POSSIBLE GARBAGE COLECTION */
   END;

/* ENDFILE(SYSIN) IS HANDLED IN SUBROUTINE INPUT */

OPEN FILE(SYSIN);



/* NOW START PROCESSING.  FIRST INPUT DATA ITEM MUST BE READ */
/* BEFORE ENTERING THE MAIN PROCESSING LOOP.                 */

CALL MSG(^,' ');
CALL INPUT(SEPARATOR,ITEM);
```

```
/*************************/
/* II. PROCESS REQUESTS */
/*************************/

DO WHILE(¬EOF);

   CALL READ_REQUEST;

   IF REQUEST=DELETE THEN
      DO;
         #_DELETES=#_DELETES+1;
         IF       REQUEST_LEVEL='C' THEN CALL DEL_CIRCUIT;
         ELSE IF REQUEST_LEVEL='T' THEN CALL DEL_TRUNK;
         ELSE IF REQUEST_LEVEL='L' THEN CALL DEL_LINK;
         ELSE IF REQUEST_LEVEL='S' THEN CALL DEL_SITES;
         ELSE CALL MSG(12,'INVALID RETN FROM READ_REQUEST: '''||REQUEST
                  ||REQUEST_LEVEL||''.'' PROCESSING CONTINUES.');
      END;

   ELSE IF REQUEST=ADDCHANGE THEN
      DO;
         #_ADDCHANGES=#_ADDCHANGES+1;
         IF       REQUEST_LEVEL='C' THEN CALL AC_CIRCUIT;
         ELSE IF REQUEST_LEVEL='T' THEN CALL AC_TRUNK;
         ELSE IF REQUEST_LEVEL='L' THEN CALL AC_LINK;
         ELSE IF REQUEST_LEVEL='S' THEN CALL AC_SITES;
         ELSE CALL MSG(12,'INVALID RETN FROM READ_REQUEST: '''||REQUEST
                  ||REQUEST_LEVEL||''.'' PROCESSING CONTINUES.');
      END;

   ELSE IF REQUEST=REROUTE THEN
      DO;
         #_REROUTES=#_REROUTES+1;
         IF       REQUEST_LEVEL='T' THEN CALL RE_TRUNK;
         ELSE CALL MSG(12,'INVALID RETN FROM READ_REQUEST: '''||REQUEST
                  ||REQUEST_LEVEL||''.'' PROCESSING CONTINUES.');
      END;

   ELSE IF REQUEST=CHANGELOC THEN
      DO;
         #_CHANGELOCS=#_CHANGELOCS+1;
         IF       REQUEST_LEVEL='C' THEN CALL CL_CIRCUIT;
         ELSE IF REQUEST_LEVEL='T' THEN CALL CL_TRUNK;
         ELSE IF REQUEST_LEVEL='L' THEN CALL CL_LINK;
         ELSE CALL MSG(12,'INVALID RETN FROM READ_REQUEST: '''||REQUEST
                  ||REQUEST_LEVEL||''.'' PROCESSING CONTINUES.');
      END;
```

```
   ELSE IF REQUEST¬=NEXT THEN
        /* IT SHOULD NOT BE POSSIBLE FOR THIS TO HAPPEN BUT WATCH */
        /* FOR IT ANYWAY.                                         */
      CALL MSG(12,'INVALID REQUEST '''||REQUEST||
                  ''' DETECTED.  FURTHER PROCESSING WILL BE ATTEMPTED.');

   /* "NEXT" MEANS READ_REQUEST DETECTED AN ERROR AND THE CURRENT */
   /* REQUEST SHOULD NOT BE PROCESSED.  GO DO THE NEXT ONE.       */


   IF SEPARATOR=';' THEN
      DO;
         SEPARATOR=' ';  /* CLEAR STOP SIGNAL */
         CALL FLUSH_REQUEST;
      END;

END /* OF MAIN PROCESSING LOOP */ ;
```

```
/*****************/
/* III. CLEANUP */
/*****************/

CALL MSG(0,' ');
CALL MSG(0,(111)'*');
CALL MSG(0,' ');

/* DELETE FLAGGED CIRCUITS AND TRUNKS */
FUNCT=DEL_M;
FILEID='RMCT';
ELMLIST='RMCTCTRLEND.';
CALL MSG(0,'STATUS CHECK FOR DELETED CIRCUITS:');
#_DELS=0;
DO DCPTR=DEL_CT_PTR REPEAT NEXTDC WHILE(DCPTR¬=NULL);
   CONTROL=DCKEY;
   CALL DBIO(7,2);
   IF STATUS='IMDL' THEN
      CALL MSG(0,'  '||DCKEY||' RE-ADDED');
   ELSE IF STATUS=OK THEN
      DO;
         CALL MSG(0,'  '||DCKEY||' DELETED <--');
         #_DELS=#_DELS+1;
      END;
   ELSE IF STATUS¬='MRNF' THEN
      SIGNAL CONDITION(TOTERR);
END;
CALL MSG(0,CHAR(#_DELS)||' CIRCUITS DELETED');
FILEID='RMTR';
ELMLIST='RMCTCTRLEND.';
CALL MSG(0,'STATUS CHECK FOR DELETED TRUNKS:');
#_DELS=0;
DO DTPTR=DEL_TR_PTR REPEAT NEXTDT WHILE(DTPTR¬=NULL);
   CONTROL=DTKEY;
   CALL DBIO(7,2);
   IF STATUS='IMDL' THEN
      CALL MSG(0,'  '||DTKEY||' RE-ADDED');
   ELSE IF STATUS=OK THEN
      DO;
         CALL MSG(0,'  '||DTKEY||' DELETED <--');
         #_DELS=#_DELS+1;
      END;
   ELSE IF STATUS¬='MRNF' THEN
      SIGNAL CONDITION(TOTERR);
END;
CALL MSG(0,CHAR(#_DELS)||' TRUNKS DELETED');
CALL MSG(0,' ');

IF GARBCOL THEN
   CALL GARBAGE_COLLECT; /* DELETE UNUSED LINKS AND SITES */
ELSE
   CALL MSG(0,'GARBAGE COLLECTION NOT PERFORMED ON LINKS AND SITES.');
```

```
/******/
TERMERR:
/******/

        /* IF CONTROL REACHES THIS POINT BECAUSE OF A TERMINAL ERROR, */
        /* THEN CIRCUITS AND TRUNKS FLAGGED FOR DELETION HAVE NOT BEEN*/

        /* DELETED AND GARBAGE COLLECTION HAS NOT BEEN PERFORMED ON   */
        /* SITES AND LINKS.  CONDITION(TOTERR) HAS INFORMED THE USER. */

/* SET JCB STEP RETURN CODE */
CALL FLIPETC(RC);

/* REPORT STATISTICS */
/* MAKE SURE THERE IS ENOUGH ROOM ON CURRENT PAGE */
IF LINENO(SYSPRINT) > LINES-14 THEN SIGNAL ENDPAGE(SYSPRINT);
CALL MSG(0,'');
CALL MSG(0,'');
CALL MSG(0,'NUMBER OF OPERATIONS ATTEMPTED:');
OUTPIC=#_ADDCHANGES;
CALL MSG(0,'                                        @ = '||OUTPIC);
OUTPIC=#_DELETES;
CALL MSG(0,'                                        - = '||OUTPIC);
OUTPIC=#_REROUTES;
CALL MSG(0,'                                        # = '||OUTPIC);
OUTPIC=#_CHANGELOCS;
CALL MSG(0,'                                        % = '||OUTPIC);
OUTPIC=#_ADDCHANGES+#_DELETES+#_REROUTES+#_CHANGELOCS;
CALL MSG(0,'                                    TOTAL = '||OUTPIC);
CALL MSG(0,'');
OUTPIC=MSG_COUNT;
CALL MSG(0,'NUMBER OF ERROR OR WARNING MESSAGES = '||OUTPIC);
CALL MSG(0,'');
OUTPIC=RC;
CALL MSG(0,'PROCESSING TERMINATED.  RETURN CODE = '||OUTPIC);
CALL MSG(0,' ');

/* CLOSE TOTAL FILES AND RELEASE TOTAL */
FUNCT=CLOSM;
CALL DBIO(4,0);
FUNCT=DEQUE;
CALL DBIO(4,0);

/* CLOSE OTHER FILES */
CLOSE FILE(SYSIN),
      FILE(SYSPRINT);


/*** LOGICAL END OF PROGRAM.  INTERNAL PROCEDURES FOLLOW.  ***/
/*** SEE STATIC STRUCTURE OUTLINE ABOVE.                   ***/
```

```
/*                  SUBROUTINE AC_CIRCUIT                                  */
/*                                                                         */
/*      THIS SUBROUTINE PERFORMS THE ACTION SPECIFIED BY A @C REQUEST.     */
/* IF NO RECORD IS FOUND WITH THE SPECIFIED CONTROL KEY OR IF A            */
/* RECORD IS FOUND BUT CONTAINS NO LINKAGE TO VARIABLE RECORDS,            */
/* THEN IT IS ASSUMED THAT THE USER IS DOING AN "ADD."  ROUTING            */
/* INFORMATION MUST BE SPECIFIED.  IF THE MASTER RECORD IS PRE-            */
/* SENT WITH LINKAGE, THEN ONLY A "CHANGE DATA" IS PERFORMED AND           */
/* NO ROUTING MAY BE SPECIFIED.                                            */
/*                                                                         */
/*       AC_CIRCUIT HAS NO PARAMETERS.  ALL DATA ARE OBTAINED FROM         */
/* GLOBAL VARIABLES DECLARED IN THE MAIN PROCEDURE.                        */

AC_CIRCUIT: PROCEDURE;


DECLARE KEYC CHARACTER(9);
KEYC=STRING(INDATA.RMCTCTRL);

/* CHECK RMCT FOR PRESENCE OF CIRCUIT RECORD */

FUNCT=READM;
FILEID='RMCT';
CONTROL=KEYC;
FLMLIST=RMCT_ELEM; /* ALL NON-LINKPATH INFO IN CIRCUIT RECORD */

CALL DBIO(7,2);  /* READ THE RECORD */
```

```
IF STATUS='MRNF' THEN /* MASTER CIRCUIT RECORD NOT FOUND */
   DO;

      STATUS=OK;

      /* FOR AN ADD, USER MUST SPECIFY TRUNKS USED BY NEW CIRCUIT */
      IF ROUTESPEC=NULL THEN
         DO;
            CALL MSG(8,'CIRCUIT '||KEYC||
                     ' NOT ADDED.  NO TRUNK(S) SPECIFIED.');
            RETURN;
         END;

      /* THE SPECIFIED TRUNKS MUST TRACE A CONTINUOUS PATH.*/
      /* FUNCTION ROUTE_TRACE RETURNS THE VALUE  TRUE  IF  */
      /* THE PATH IS TRACEABLE AND RETURNS THE ENDSITES AS */
      /* ARGUMENTS.                                        */

      IF ¬ROUTE_TRACE(WKSITE1,WKSITE2) THEN
         DO;  /* USER HAS SPECIFIED BAD SET OF TRUNKS */
            CALL MSG(8,'CIRCUIT '||KEYC||
                     ' NOT ADDED.  SPECIFIED TRUNKS DO NOT TRACE A '||
                     'CONTINUOUS PATH.');
            RETURN;
         END;

      /*TRUNKS CHECKED OUT. MOVE DATA TO IOAREA, ADD CIRCUIT TO RMCT.*/
      IOCIRCUIT=INDATA.CIRCUIT, BY NAME;
      IF IOCIRCUIT.RMCTFLAG=' ' THEN IOCIRCUIT.RMCTFLAG='8';/*DEFAULT*/
      STRING(IOCIRCUIT.CIRCUITSITE1)=WKSITE1;
      STRING(IOCIRCUIT.CIRCUITSITE2)=WKSITE2;

      FUNCT=ADD_M;
      FILEID='RMCT';
      CONTROL=KEYC;
      ELMLIST=RMCT_ELEM;

      CALL DBIO(7,2);  /* WRITE THE NEW CIRCUIT MASTER RECORD */
      IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);

      /* DISPLAY MESSAGES ABOUT ADDITION */
      CALL MSG(0,'CIRCUIT '||KEYC||' ADDED');

      /* NOW ADD RECORDS TO RVCT TO PROVIDE LINKAGE TO RMTR. */
      CALL ADD_ROUTING;

      /* FALL THROUGH AND RETURN. */

   END /* OF PROCESSING FOR MASTER CIRCUIT RECORD NOT FOUND. */ ;
```

```
ELSE IF STATUS=OK THEN /* MASTER CIRCUIT RECORD FOUND */
   DO;
      /* SEE IF LINKAGE TO RVCT IS ALREADY PRESENT BY DOING A READV. */
      FUNCT='READV';
      FILEID='RVCT';
      REFER='LKCT';
      LKPATH='RMCTLKCT';
      FLMLIST='RVCTRMCTEND.';
      ENDP='ELSE'; /* WILL NOT UPDATE THIS RECORD */
      /* CONTROL ALREADY SET */

      IOSAVE=IOAREA; /* SAVE CIRCUIT DATA FOR UPDATE */

      CALL DBIO(9,2); /* READ THE VARIABLE RECORD */

      ENDP='END.'; /* DON'T FORGET TO RESET ENDP. */
      IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);

      /* REFER=ENDP MEANS: NO LINKAGE PRESENT,                    */
      /*                   RECORD IS FLAGGED FOR DELETION, AND    */
      /*                   USER IS DOING AN "ADD."                */
      /* OTHERWISE, THIS IS A "CHANGE" REQUEST.                   */
```

```
/* IF NO LINKAGE TO RVCT PRESENT, THEN ASSUME USER IS DOING ADD*/
IF REFER=ENDF THEN
   /* NO LINKAGE, SO TREAT AS ADD. */
   DO;
      /* FOR AN ADD, USER MUST SPECIFY TRUNKS USED BY NEW CRCT */
      IF ROUTESPEC=NULL THEN
         DO;
            CALL MSG(8,'CIRCUIT '||KEYC||
                     ' NOT ADDED.  NO TRUNK(S) SPECIFIED.');
            RETURN;
         END;

      /* THE SPECIFIED TRUNKS MUST TRACE A CONTINUOUS PATH.*/
      /* FUNCTION ROUTE_TRACE RETURNS THE VALUE  TRUE  IF  */
      /* THE PATH IS TRACEABLE AND RETURNS THE ENDSITES AS */
      /* ARGUMENTS.                                        */
      IF ¬ROUTE_TRACE(WKSITE1,WKSITE2) THEN
         DO;  /* USER HAS SPECIFIED BAD SET OF TRUNKS */
            CALL MSG(8,'CIRCUIT '||KEYC||' NOT ADDED.  SPECIFIED'
                     ||' TRUNKS DO NOT TRACE A CONTINUOUS PATH.');
            RETURN;
         END;

      /* RESTORE SAVED IOAREA AND SET   */
      /* ENDSITES TO PREPARE FOR UPDATE */
      IOAREA=IOSAVE;
      STRING(IOCIRCUIT.CIRCUITSITE1)=WKSITE1;
      STRING(IOCIRCUIT.CIRCUITSITE2)=WKSITE2;
      IOCIRCUIT.RMCIFLAG=28; /* SET DEFAULT VALUE FOR ADD */

      IF UPDATECIRCUIT THEN /* REPLACEMENT DATA SPECIFIED */
         CALL SET_UPDATE_C;

      /* EVEN IF NO UPDATES ARE SPECIFIED, THE RECORD */
      /* MUST STILL BE REWRITTEN TO REFLECT THE NEW   */
      /* END SITES.                                   */

      FUNCT=WRITM;
      FILEID='RMCT';
      CONTROL=KEYC;
      ELMLIST=RMCT_ELEM;

      CALL DBIO(7,2);
      IF STATUS¬=OK THEN SIGNAL CONDITION(IOERR);

      /* DISPLAY MESSAGE ABOUT ADDITION */
      CALL MSG(0,'CIRCUIT '||KEYC||' ADDED');

      /* NOW ADD RECORDS TO RVCT TO PROVIDE LINKAGE TO RMTR. */
      CALL ADD_ROUTING;
      /* FALL THROUGH AND RETURN */
   END;
```

```
      ELSE /* LINKAGE ALREADY PRESENT.  UPDATE ONLY. */
         DO;

            /* RESTORE SAVED IOAREA */
            IOAREA=IOSAVE;

            /* USER NOT PERMITTED TO SPECIFY TRUNKS FOR AN UPDATE */
            IF ROUTESPEC¬=NULL THEN /* IF HE DID, DON'T. */
               DO;
                  CALL MSG(8,'CIRCUIT '||KEYC||
                           ' ALREADY PRESENT.  SPECIFIED TRUNKS NOT'||
                           ' CHECKED FOR CONTINUITY.');
                  IF UPDATECIRCUIT THEN
                     CALL MSG(4,
                              'SPECIFIED DATA UPDATES NOT PERFORMED.');

                  RETURN;
               END;

            /* NO TRUNKS SPECIFIED.  CHECK FOR UPDATES. */
            IF UPDATECIRCUIT THEN
               DO;
                  /* DISPLAY "BEFORE" VALUES */
                  /*<TEMP:>*/ CALL MSG(0,'BEFORE:'||STRING(IOCIRCUIT));

                  CALL SET_UPDATE_C;
                  FUNCT=WRITM;
                  FILEID='RMCT';
                  ELMLIST=RMCT_ELEM;
                  /* CONTROL ALREADY SET */

                  CALL DBIO(7,2);
                  IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);

                  /* DISPLAY "AFTER" VALUES */
                  /*<TEMP:>*/ CALL MSG(0,'AFTER: '||STRING(IOCIRCUIT));
               END;


            ELSE /* CIRCUIT SPECIFIED WAS ALREADY PRESENT, AND USER */
                 /* SPECIFIED NO UPDATES: WARN THE USER.           */
               CALL MSG(4,'CIRCUIT SPECIFIED ALREADY PRESENT IN DATA'||
                          ' BASE AND NO UPDATES SPECIFIED.');
         END;

   END /* OF PROCESSING FOR MASTER RECORD FOUND IN RMCT */ ;

FLSE /* INVALID STATUS TRYING TO LOCATE SPECIFIED CIRCUIT RECORD */
   SIGNAL CONDITION(TOTERR);

RETURN;
```

```
/*                    SUBROUTINE SET_UPDATE_C                          */
/*                                                                     */
/*       THIS SUBROUTINE IS CALLED BY AC_CIRCUIT TO SET NEW VALUES     */
/* FOR THE DATA FIELDS IN A CIRCUIT MASTER RECORD.  IT IS CALLED FOR   */
/* A RE-ADD IF SOME DATA FIELDS ARE RESPECIFIED AND FOR A CHANGE.      */
/* THE ASTERISK (*) INDICATES THAT THE DEFAULT VALUE IS TO BE RESET.   */

SET_UPDATE_C: PROCEDURE;

/* THE FIRST IN EACH PAIR OF ASSIGNMENT STATEMENTS IS UNNECESSARY IF   */
/* "*" WAS SPECIFIED, BUT IT MAKES THE PROGRAM CLEARER; AND "*" WON'T  */
/* ACTUALLY BE USED MUCH, SO IT ISN'T REALLY INEFFICIENT.              */

IF INDATA.RMCTFLAG¬='' THEN IOCIRCUIT.RMCTFLAG=INDATA.RMCTFLAG;
IF INDATA.RMCTFLAG='*' THEN IOCIRCUIT.RMCTFLAG=28;

IF INDATA.RMCTRSTP¬='' THEN IOCIRCUIT.RMCTRSTP=INDATA.RMCTRSTP;
IF INDATA.RMCTRSTP='*' THEN IOCIRCUIT.RMCTRSTP=' ';

IF INDATA.RMCTRAPL¬='' THEN IOCIRCUIT.RMCTRAPL=INDATA.RMCTRAPL;
IF INDATA.RMCTRAPL='*' THEN IOCIRCUIT.RMCTRAPL=' ';

IF INDATA.RMCTIDEN¬='' THEN IOCIRCUIT.RMCTIDEN=INDATA.RMCTIDEN;
IF INDATA.RMCTIDEN='*' THEN IOCIRCUIT.RMCTIDEN=' ';

IF INDATA.RMCTXREF¬='' THEN IOCIRCUIT.RMCTXREF=INDATA.RMCTXREF;
IF INDATA.RMCTXREF='*' THEN IOCIRCUIT.RMCTXREF=' ';

IF INDATA.RMCTFILL¬='' THEN IOCIRCUIT.RMCTFILL=INDATA.RMCTFILL;
IF INDATA.RMCTFILL='*' THEN IOCIRCUIT.RMCTFILL=' ';

RETURN;

END /* SET_UPDATE_C */ ;


END /* AC_CIRCUIT */ ;
```

```
/*                SUBROUTINE AC_TRUNK                                    */
/*                                                                       */
/*      THIS SUBROUTINE PERFORMS THE ACTION SPECIFIED BY A @T REQUEST.*/
/* IF NO RECORD IS FOUND WITH THE SPECIFIED CONTROL KEY OR IF A          */
/* RECORD IS FOUND BUT CONTAINS NO LINKAGE TO VARIABLE RECORDS,          */
/* THEN IT IS ASSUMED THAT THE USER IS DOING AN "ADD."  ROUTING          */
/* INFORMATION MUST BE SPECIFIED.  IF THE MASTER RECORD IS PRESENT       */
/* WITH LINKAGE, THEN ONLY A "CHANGE DATA" IS PERFORMED AND NO           */
/* ROUTING MAY BE SPECIFIED.                                             */
/*                                                                       */
/*      AC_TRUNK HAS NO PARAMETERS.  ALL DATA ARE OBTAINED FROM          */
/* GLOBAL VARIABLES DECLARED IN THE MAIN PROCEDURE.                      */


AC_TRUNK: PROCEDURE;


DECLARE KEYT CHARACTER(7);
KEYT=STRING(INDATA.RMTRCTRL);

/* CHECK RMTR FOR PRESENCE OF TRUNK RECORD */

FUNCT='READM';
FILEID='RMTR';
CONTROL=KEYT;
ELMLIST=RMTR_ELEM; /* ALL NON-LINKPATH INFO IN TRUNK RECORD */

CALL DBIO(7,2);  /* READ THE RECORD */
```

```
IF STATUS='MRNF' THEN /* MASTER TRUNK RECORD NOT FOUND */
   DO;

      STATUS=OK;

      /* FOR AN ADD, USER MUST SPECIFY LINKS USED BY NEW TRUNK */
      IF ROUTESPEC=NULL THEN
         DO;
            CALL MSG(8,'TRUNK '||KEYT||
                     ' NOT ADDED.  NO LINK(S) SPECIFIED.');
            RETURN;
         END;

      /* THE SPECIFIED LINKS MUST TRACE A CONTINUOUS PATH. */
      /* FUNCTION ROUTE_TRACE RETURNS THE VALUE  TRUE  IF  */
      /* THE PATH IS TRACEABLE AND RETURNS THE ENDSITES AS */
      /* ARGUMENTS.                                        */
      IF ¬ROUTE_TRACE(WKSITE1,WKSITE2) THEN
         DO;  /* USER HAS SPECIFIED BAD SET OF LINKS */
            CALL MSG(8,'TRUNK '||KEYT||
                     ' NOT ADDED.  SPECIFIED LINKS DO NOT TRACE A '||
                     'CONTINUOUS PATH.');
            RETURN;
         END;

      /* LINKS CHECKED OUT.  MOVE DATA TO IOAREA & ADD TRUNK TO RMTR.*/
      IOTRUNK=INDATA.TRUNK, BY NAME;
      IF IOTRUNK.RMTRFLAG=' ' THEN IOTRUNK.RMTRFLAG=Z8; /* DEFAULT */
      STRING(IOTRUNK.TRUNKSITE1)=WKSITE1;
      STRING(IOTRUNK.TRUNKSITE2)=WKSITE2;

      FUNCT=ADD_M;
      FILEID='RMTR';
      CONTROL=KEYT;
      ELMLIST=RMTR_ELEM;

      CALL DBIO(7,2);  /* WRITE THE NEW TRUNK MASTER RECORD */
      IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);

      /* DISPLAY MESSAGES ABOUT ADDITION */
      CALL MSG(0,'TRUNK '||KEYT||' ADDED');

      /* NOW ADD RECORDS TO RVTL TO PROVIDE LINKAGE TO RMLK. */
      CALL ADD_ROUTING;

      /* FALL THROUGH AND RETURN. */

   END /* OF PROCESSING FOR MASTER TRUNK RECORD NOT FOUND. */ ;
```

```
ELSE IF STATUS=OK THEN /* MASTER TRUNK RECORD FOUND */
   DO;
      /* SEE IF LINKAGE TO RVTL IS ALREADY PRESENT BY DOING A READV. */
      FUNCT='READV';
      FILEID='RVTL';
      REFER='LKTL';
      LKPATH='RMTFLKTI';
      FLMLIST='RVTLRMTREND.';
      ENDP='ELSE'; /* WILL NOT UPDATE THIS RECORD */
      /* CONTROL ALREADY SET */

      IOSAVE=IOAREA; /* SAVE TRUNK DATA FOR UPDATE */

      CALL DBIO(9,2); /* READ THE VARIABLE RECORD */

      ENDP='END.';  /* DON'T FORGET TO RESET ENDP. */
      IF STATUS¬=OK THEN SIGNAL CONDITION(TCTERR);
```

```
/* IF NO LINKAGE TO RVTL PRESENT (WHICH IMPLIES THAT THERE IS  */
/* NONE TO RVCT), THEN ASSUME USER IS DOING AN ADD.            */
IF REFER=ENDP THEN
   /* NO LINKAGE, SO TREAT AS ADD. */
   DO;
      /* FOR AN ADD, USER MUST SPECIFY LINKS USED BY NEW TRUNK */
      IF ROUTESPEC=NULL THEN
         DO;
            CALL MSG(8,'TRUNK '||KEYT||
                     ' NOT ADDED.  NO LINK(S) SPECIFIED.');
            RETURN;
         END;

      /* THE SPECIFIED LINKS MUST TRACE A CONTINUOUS PATH.  */
      /* FUNCTION ROUTE_TRACE RETURNS THE VALUE  TRUE  IF   */
      /* THE PATH IS TRACEABLE AND RETURNS THE ENDSITES AS  */
      /* ARGUMENTS.                                         */

      IF ¬ROUTE_TRACE(WKSITE1,WKSITE2) THEN
         DO;  /* USER HAS SPECIFIED BAD SET OF LINKS */
            CALL MSG(8,'TRUNK '||KEYT||
                     ' NOT ADDED.  SPECIFIED LINKS DO NOT TRACE '
                     ||'A CONTINUOUS PATH.');
            RETURN;
         END;

      /* RESTORE SAVED IOAREA AND SET    */
      /* ENDSITES TO PREPARE FOR UPDATE */
      IOAREA=IOSAVE;
      STRING(IOTRUNK.TRUNKSITE1)=WKSITE1;
      STRING(IOTRUNK.TRUNKSITE2)=WKSITE2;
      IOTRUNK.RMTRFLAG=Z8; /* SET DEFAULT VALUE FOR ADD */
      IF UPDATETRUNK THEN /* REPLACEMENT DATA SPECIFIED */
         CALL SET_UPDATE_T;

      /* EVEN IF NO UPDATES ARE SPECIFIED, THE RECORD        */
      /* MUST STILL BE REWRITTEN TO REFLECT THE NEW END SITES */
      FUNCT=WRITM;
      FILEID='RMTR';
      CONTROL=KEYT;
      ELMLIST=RMTR_ELEM;
      CALL DBIO(7,2);
      IF STATUS¬=OK THEN SIGNAL CONDITION(TCTERR);

      /* DISPLAY MESSAGE ABOUT ADDITION */
      CALL MSG(0,'TRUNK '||KEYT||' ADDED');

      /* NOW ADD RECORDS TO RVTL TO PROVIDE LINKAGE TO RMLK. */
      CALL ADD_ROUTING;

      /* FALL THROUGH AND RETURN */
   END;
```

```
      ELSE /* LINKAGE ALREADY PRESENT.  UPDATE ONLY. */
         DO;

            /* RESTORE SAVED IOAREA */
            ICAREA=ICSAVE;

            /* USER NOT PERMITTED TO SPECIFY LINKS FOR AN UPDATE */
            IF ROUTESPEC¬=NULL THEN /* IF HE DID, DON'T. */
               DO;
                  CALL MSG(8,'TRUNK '||KEYT||
                           ' ALREADY PRESENT.  SPECIFIED LINKS NOT'||
                           ' CHECKED FOR CONTINUITY.');
                  IF UPDATETRUNK THEN
                     CALL MSG(4,
                              'SPECIFIED DATA UPDATES NOT PERFORMED.');

                  RETURN;
               END;

            /* NO LINKS SPECIFIED.  CHECK FOR UPDATES. */
            IF UPDATETRUNK THEN
               DO;
                  /* DISPLAY "BEFORE" VALUES */
                  /*<TEMP:>*/ CALL MSG(0,'BEFORE:'||STRING(IOTRUNK));

                  CALL SET_UPDATE_1;
                  FUNCT=WRITM;
                  FILEID='RMTR';
                  ELMLIST=RMTR_ELEM;
                  /* CONTROL ALREADY SET */

                  CALL DBIO(7,2);
                  IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);

                  /* DISPLAY "AFTER" VALUES */
                  /*<TEMP:>*/ CALL MSG(0,'AFTER: '||STRING(IOTRUNK));

               END;


            ELSE /* TRUNK SPECIFIED WAS ALREADY PRESENT, AND USER */
                 /* SPECIFIED NO UPDATES: WARN THE USER.          */
               CALL MSG(4,'TRUNK SPECIFIED ALREADY PRESENT IN DATA '||
                          'BASE AND NO UPDATES SPECIFIED.');
         END;

   END /* OF PROCESSING FOR MASTER RECORD FOUND IN RMTR */ ;

ELSE /* INVALID STATUS TRYING TO LOCATE SPECIFIED TRUNK RECORD */
   SIGNAL CONDITION(TOTERR);

RETURN;
```

```
/*                    SUBROUTINE SET_UPDATE_T                          */
/*                                                                     */
/*      THIS SUBROUTINE IS CALLED BY AC_TRUNK TO SET NEW VALUES        */
/* FOR THE DATA FIELDS IN A TRUNK MASTER RECORD.  IT IS CALLED FOR     */
/* A RE-ADD IF SOME DATA FIELDS ARE RESPECIFIED AND FOR A CHANGE.      */
/* THE ASTERISK (*) INDICATES THAT THE DEFAULT VALUE IS TO BE RESET.   */

SET_UPDATE_T: PROCEDURE;

/* THE FIRST IN EACH PAIR OF ASSIGNMENT STATEMENTS IS UNNECESSARY IF   */
/* "*" WAS SPECIFIED, BUT IT MAKES THE PROGRAM CLEARER; AND "*" WON'T  */
/* ACTUALLY BE USED MUCH, SO IT ISN'T REALLY INEFFICIENT.              */

IF INDATA.RMTRFLAG¬='' THEN IOTRUNK.RMTRFLAG=INDATA.RMTRFLAG;
IF INDATA.RMTRFLAG='*' THEN IOTRUNK.RMTRFLAG=ZB;

IF INDATA.RMTRRCAT¬='' THEN IOTRUNK.RMTRRCAT=INDATA.RMTRRCAT;
IF INDATA.RMTRRCAT='*' THEN IOTRUNK.RMTRRCAT=' ';

IF INDATA.RMTRTCAP¬='' THEN IOTRUNK.RMTRTCAP=INDATA.RMTRTCAP;
IF INDATA.RMTRTCAP='*' THEN IOTRUNK.RMTRTCAP=' ';

IF INDATA.RMTRBAND¬='' THEN IOTRUNK.RMTRBAND=INDATA.RMTRBAND;
IF INDATA.RMTRBAND='*' THEN IOTRUNK.RMTRBAND=' ';

IF INDATA.RMTRAVAL¬='' THEN IOTRUNK.RMTRAVAL=INDATA.RMTRAVAL;
IF INDATA.RMTRAVAL='*' THEN IOTRUNK.RMTRAVAL=' ';

IF INDATA.RMTRFILL¬='' THEN IOTRUNK.RMTRFILL=INDATA.RMTRFILL;
IF INDATA.RMTRFILL='*' THEN IOTRUNK.RMTRFILL=' ';

RETURN;

END /* SET_UPDATE_T */ ;


END /* AC_TRUNK */ ;
```

```
/*                SUBROUTINE AC_LINK, AC_SITES                          */
/*                                                                      */
/*     THIS SUBROUTINE PERFORMS THE ACTION SPECIFIED BY A @L OR @S      */
/* REQUEST.                                                             */
/*                                                                      */
/* @S:                                                                  */
/*     ONE OR TWO SITES MAY BE SPECIFIED.  EACH SITE IS PROCESSED       */
/* INDEPENDENTLY.  IF NO RECORD IS FOUND WITH THE SPECIFIED CONTROL      */
/* KEY, THE SITE IS ADDED; OTHERWISE DATA FIELDS ARE UPDATED AS         */
/* SPECIFIED.                                                           */
/*                                                                      */
/* @L:                                                                  */
/*     IF SITES ARE SPECIFIED, THEY ARE PROCESSED FIRST AS DESCRIBED    */
/* ABOVE.  IF NO MASTER LINK RECORD IS FOUND WITH THE SPECIFIED LINK     */
/* CONTROL KEY AND IF EXACTLY TWO VALID SITES WERE SPECIFIED, THEN      */
/* THE LINK IS ADDED.  IF THE LINK WAS PRESENT AND EITHER NO SITES OR   */
/* THE CORRECT TWO SITES WERE SPECIFIED, THEN DATA FIELDS ARE UPDATED   */
/* AS SPECIFIED.                                                        */
/*                                                                      */
/*     AC_LINK/AC_SITES HAS NO PARAMETERS.  ALL VALUES ARE OBTAINED     */
/* FROM GLOBAL VARIABLES DECLARED IN THE MAIN PROCEDURE.                */

AC_LINK:
AC_SITES: PROCEDURE;

DECLARE
          SITE1 CHARACTER(11),
          SITE2 CHARACTER(11),
          KEYS  CHARACTER(11),
          I     BINARY FIXED(15,0);


/* CHECK SITES AND ADD IF NECESSARY */
DO I=1 TO 2 WHILE(INDATA.GEOLOC(I)^='          ');

   KEYS=STRING(INDATA.RMSICTRL(I));

   /* CHECK SITE MASTER FILE FOR PRESENCE OF SPECIFIED SITE. */

   FUNCT=READM;
   FILEID='RMSI';
   CONTROL=KEYS;
   ELMLIST=RMSI_ELEM;
   CALL DBIO(7,2); /* READ A SITE RECORD */

   /* IF THE RECORD WAS NOT FOUND, ADD IT.  IF IT WAS FOUND, UPDATE    */
   /* THE DATA FIELDS IF NEW ONES WERE SPECIFIED.  IF ANY OTHER        */
   /* STATUS IS RETURNED, SIGNAL CONDITION(TOTERR) TO PRINT A MESSAGE  */
   /* AND TERMINATE.                                                   */
```

```
IF STATUS='MENF' THEN /* MASTER SITE RECORD NOT FOUND */
   DO;
      /* ADD A NEW SITE RECORD */
      IOSITE=INDATA.SITE(I); /* COPY DATA TO IOAREA */
      IF IOSITE.RMSIFLAG=' ' THEN IOSITE.RMSIFLAG=Z8; /* DEFAULT */
      FUNCT=ADD_M;
      CALL DBIO(7,2);
      IF STATUS^=OK THEN SIGNAL CONDITION(TOTERR);
      CALL MSG(0,'SITE '||KEYS||' ADDED.');
   END;
```

```
   ELSE IF STATUS=OK THEN /* RECORD FOUND */
      DO;
         /* UPDATE DATA FIELDS IF SPECIFIED */
         IF UPDATESITE(I) THEN /* REPLACEMENT DATA SPECIFIED */
            DO;

               /* DISPLAY OLD VALUES FOR USER */
               CALL MSG(0,'BEFORE:'||STRING(IOSITE));

               IF INDATA.RMSISTCT(I)¬='' THEN
                  IOSITE.RMSISTCT=INDATA.RMSISTCT(I);
               IF INDATA.RMSISTCT(I)='*' THEN
                  IOSITE.RMSISTCT=' ';

               IF INDATA.RMSIFLAG(I)¬='' THEN
                  IOSITE.RMSIFLAG=INDATA.RMSIFLAG(I);
               IF INDATA.RMSIFLAG(I)='*' THEN
                  IOSITE.RMSIFLAG=Z8;

               IF INDATA.RMSICORD(I)¬='' THEN
                  IOSITE.RMSICORD=INDATA.RMSICORD(I);
               IF INDATA.RMSICORD(I)='*' THEN
                  IOSITE.RMSICORD=' ';

               IF INDATA.RMSINUMB(I)¬='' THEN
                  IOSITE.RMSINUMB=INDATA.RMSINUMB(I);
               IF INDATA.RMSINUMB(I)='*' THEN
                  IOSITE.RMSINUMB=' ';

               IF INDATA.RMSIFILL(I)¬='' THEN
                  IOSITE.RMSIFILL=INDATA.RMSIFILL(I);
               IF INDATA.RMSIFILL(I)='*' THEN
                  IOSITE.RMSIFILL=' ';

               FUNCT=WRITM;
               CALL DBIO(7,2); /* UPDATE THE SITE RECORD */
               IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);

               /* DISPLAY NEW DATA FOR USER */
               CALL MSG(0,'AFTER: '||STRING(IOSITE));
            END;

         ELSE IF REQUEST_LEVEL='S' THEN /*    */
            CALL MSG(4,'SITE '||KEYS||' ALREADY PRESENT IN DATA'
                     ||' BASE AND NO UPDATES SPECIFIED.');
      END;

   ELSE /* INVALID STATUS TRYING TO READ SITE RECORD */
      SIGNAL CONDITION(TOTERR);

END; /* OF LOOP TO PROCESS SITES */
```

```
IF REQUEST_LEVEL='S' THEN RETURN;

/* ONLY THE FIRST EIGHT BYTES ARE CURRENTLY SIGNIFICANT . . .
SITE1=STRING(INDATA.RMSICTRL(1));
SITE2=STRING(INDATA.RMSICTRL(2));
*/
SITE1=INDATA.GEOLOC(1);
SITE2=INDATA.GEOLOC(2);
```

```
/*** PROCESS LINK ***/

/* CHECK RMLK FOR PRESENCE OF LINK RECORD */
FUNCT=READM;
FILEID='RMLK';
CONTROL=INDATA.RMLKCTRL;
ELMLIST=RMLK_ELEM;
CALL DBIO(7,2);

IF STATUS='MRNF' THEN /* MASTER LINK RECORD NOT FOUND */
   DO;
      /* BEFORE ADDING A NEW LINK, MAKE SURE TWO SITES SPECIFIED */
      IF SITE1=' ' THEN /* NO SITES GIVEN */
         DO;
            CALL MSG(8,'LINK '||INDATA.RMLKCTRL||
                       ' NOT ADDED.  NO SITES SPECIFIED.');
            RETURN;
         END;
      IF SITE2=' ' THEN /* SECOND SITE NOT GIVEN */
         DO;
            CALL MSG(8,'LINK '||INDATA.RMLKCTRL||
                       ' NOT ADDED.  ONLY ONE SITE SPECIFIED.');
            RETURN;
         END;

      /* MOVE DATA TO ICAREA */
      IOLINK=INDATA.LINK, BY NAME;
      IF IOLINK.RMLKFLAG=' ' THEN IOLINK.RMLKFLAG=Z8; /* DEFAULT */
      IOLINK.RMLKFBLC=INDATA.GEOLOC(1);
      IOLINK.RMLKFFAC=INDATA.FACILITY(1);
      IOLINK.RMLKTOLC=INDATA.GEOLOC(2);
      IOLINK.RMLKTFAC=INDATA.FACILITY(2);

      FUNCT=ADD_M;
      CALL DBIO(7,2);
      IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);

      CALL MSG(0,'LINK '||INDATA.RMLKCTRL||' ADDED.');

      /* FALL THROUGH AND RETURN */

   END;
```

```
ELSE IF STATUS=OK THEN /* LINK RECORD FOUND */
   DO;
      /* COLLECT CONTROL KEY OF SITES ASSOCIATED WITH LINK */
      /* IN FIELDS FOR WORKING SITES                       */
      WKSITE1= /* STRING(LINKSITE1) */ RMLKFRLC;
      WKSITE2= /* STRING(LINKSITE2) */ RMIKTCIC;

      /* IF SITES WERE SPECIFIED, THEY MUST BE THE SAME AS THE */
      /* ONES ASSOCIATED WITH THE LINK; OTHERWISE, THE USER HAS*/
      /* MADE AN ERROR.  HE CANNOT CHANGE THE SITES ASSOCIATED */
      /* WITH A LINK IN THIS MANNER BECAUSE ALL RELATED TRUNKS */
      /* AND CIRCUITS WOULD BE AFFECTED.  TO CHANGE THE SITES  */
      /* ASSOCIATED WITH A GIVEN LINK: PURGE ALL AFFECTED CIR- */
      /* CUITS AND TRUNKS; THEN USE %L TO CHANGE THE DATA IN   */
      /* THE LOCATION AND FACILITY FIELDS OF THE LINK MASTER   */
      /* RECORD. (THE LINK RECORD DOES NOT ACTUALLY CONTAIN    */
      /* LINKAGE TO THE SITE MASTER FILE.)  ALTERNATIVELY,     */
      /* AFTER DELETING ALL AFFECTED TRUNKS AND CIRCUITS, DE-  */
      /* LETE THE LINK AND THEN RE-ADD IT WITH THE CORRECT     */
      /* SITES.                                                */

      /* IF SITES WERE SPECIFIED, THEN IF THEY DO NOT MATCH THE */
      /* CURRENT LINK END SITES IN EITHER ORDER, THEN THE USER  */
      /* HAS MADE AN ERROR.                                     */
      IF SITE1¬='' THEN
         IF ¬((SITE1=WKSITE1 & SITE2=WKSITE2) |
              (SITE1=WKSITE2 & SITE2=WKSITE1)) THEN
            DO;
               CALL MSG(8,'SPECIFIED SITES: '||SITE1||' '||SITE2||
                          ' DO NOT MATCH END SITES OF SPECIFIED LINK: '
                          ||WKSITE1||' '||WKSITE2);
               RETURN;
            END;

      /* WE NOW HAVE APPROPRIATE LINK AND SITE INFORMATION. */
      /* IF REPLACEMENT DATA SPECIFIED, UPDATE RMLK.        */
```

```
      IF UPDATELINK THEN /* REPLACEMENT DATA SPECIFIED */
         DO;

            /* DISPLAY OLD VALUES */
            CALL MSG(0,'BEFORE:'||STRING(IOLINK));

           IF INDATA.RMLKFLAG¬='' THEN IOLINK.RMLKFLAG=INDATA.RMLKFLAG;
            IF INDATA.RMLKFLAG='*' THEN IOLINK.RMLKFLAG=ZS;

           IF INDATA.RMLKTRAN¬='' THEN IOLINK.RMLKTRAN=INDATA.RMLKTRAN;
            IF INDATA.RMLKTRAN='*' THEN IOLINK.RMLKTRAN=' ';

           IF INDATA.RMLKCODE¬='' THEN IOLINK.RMLKCODE=INDATA.RMLKCODE;
            IF INDATA.RMLKCODE='*' THEN IOLINK.RMLKCODE=' ';

           IF INDATA.RMLKFILL¬='' THEN IOLINK.RMLKFILL=INDATA.RMLKFILL;
            IF INDATA.RMLKFILL='*' THEN IOLINK.RMLKFILL=' ';

            FUNCT=WAITM;
            CALL DBIO(7,2); /* UPDATE THE RECORD */
            IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);

            /* DISPLAY NEW DATA */
            CALL MSG(0,'AFTER: '||STRING(IOLINK));

         END;
      ELSE
         CALL MSG(4,'LINK SPECIFIED ALREADY PRESENT IN DATA BASE AND '
                  ||'NO UPDATES SPECIFIED.');
   END;

ELSE /* INVALID STATUS TRYING TO READ LINK MASTER RECORD */
   SIGNAL CONDITION(TOTERR);

/* END PROCESSING FOR LINK */

RETURN;

END /* AC_LINK, AC_SITES */ ;
```

```
/*                SUBROUTINE DEL_CIRCUIT                                  */
/*                                                                        */
/*      THIS SUBROUTINE PERFORMS THE ACTION SPECIFIED BY A -C REQUEST.    */
/* IT FIRST DELETES ALL LINKAGE TO RVCT AND THEN FLAGS THE RECORD         */
/* FOR DELETION BY ADDING IT TO A LINKED LIST.  IF THE CIRCUIT IS         */
/* SUBSEQUENTLY RE-ADDED, THE DATA FIELDS WILL BE RETAINED.  IF THE       */
/* CIRCUIT IS NOT RE-ADDED IT IS PHYSICALLY DELETED IN A LATER            */
/* CLEANUP STEP.                                                          */
/*                                                                        */
/*      DEL_CIRCUIT HAS NO PARAMETERS.  ALL DATA ARE OBTAINED FROM        */
/* GLOBAL VARIABLES DECLARED IN THE MAIN PROCEDURE.                       */

DEL_CIRCUIT: PROCEDURE;


DECLARE #_TRUNKS BINARY FIXED(15,0);

DECLARE KEYC     CHARACTER(9);

KEYC=STRING(INDATA.RMCTCIRL);

/* PREPARE TO DELETE ALL LINKAGE IN RVCT TO TRUNKS.  THERE IS NO NEED*/
/* TO READ RMCT FIRST BECAUSE, IF MASTER CIRCUIT RECORD IS NOT       */
/* PRESENT, THE FIRST ATTEMPT TO READV WILL RESULT IN STATUS='MRNF'. */

FUNCT=READV;
FILEID='RVCT';
REFER='LKCT';
LKPATH='RMCTLKCT';
FIMLIST=RVCT_ELEM;
CONTROL=KEYC;

CALL DBIO(9,2); /* ACCESS FIRST VARIABLE RECORD */

IF STATUS='MRNF' THEN
   DO;
      CALL MSG(4,'SPECIFIED CIRCUIT '||KEYC||
               ' NOT IN DATA BASE.  NO RECORDS DELETED.');
      STATUS=OK;
      RETURN;
   END;

IF STATUS¬=OK THEN SIGNAL CONDITION(ICTERR);
/*<IF *PNR IS TO BE HANDLED, DO IT HERE.>*/

/* DISPLAY MSG ABOUT CIRCUIT TO BE DELETED */
CALL MSG(0,'CIRCUIT '||KEYC);

#_TRUNKS=0;
```

```
/* DELETE ALL ASSOCIATED VARIABLE RECORDS IN RVCT */
DO WHILE(REFER¬=ENDP);

   /* PRINT DATA ABOUT TRUNK */
   CALL MSG(0,'  USING TRUNK '||RVCTRMTR||' BETWEEN '||RVCTRMSI||
                                          ' AND '||RVCTTOLC);

   FUNCT=DELVD;
   CALL DBIO(9,2); /* DELETE THIS RECORD */
   #_TRUNKS=#_TRUNKS+1; /* COUNT IT */
   IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);
   FUNCT=READV;
   CALL DBIO(9,2); /* READ THE NEXT VARIABLE RECORD */
   IF STATUS¬=OK THEN SIGNAL CONDITION(TCTERR);
END;

/* REWRITE THE RECORD WITH MODIFIED SITE FIELDS */
FUNCT=WRITM;
FILEID='RMCT';
ELMLIST=RMCT_LOCS;
IOAREA=KEYC||'*DELETED****DELETED***';
/* CONTROL ALREADY SET */

CALL DBIO(7,2);
IF STATUS¬=OK THEN SIGNAL CONDITION(ICTERR);

 /* ADD CIRCUIT TO BEGINNING OF LIST OF THOSE FLAGGED FOR DELETION */
ALLOCATE DCIRCUIT;
DCKEY=KEYC;
NEXTDC=DEL_CT_PTR;
DEL_CT_PTR=DCPTR;

/* DISPLAY MESSAGES ABOUT RECORD 'DELETED' */
CALL MSG(0,'  FLAGGED FOR DELETION.');

CALL MSG(0,'NUMBER OF VARIABLE RECORDS DELETED:'||
          SUBSTR(CHAR(#_TRUNKS),6));
CALL MSG(0,'CIRCUIT MASTER RECORD RETAINED WITH BLANK LINKPATHS.');

RETURN;
END /* DEL_CIRCUIT */ ;
```

```
/*                SUBROUTINE DEL_TRUNK                                    */
/*                                                                        */
/*      THIS SUBROUTINE PERFORMS THE ACTION SPECIFIED BY A -T REQUEST.    */
/* IF THE TRUNK IS USED BY NO CIRCUITS, IT DELETES ALL LINKAGE TO         */
/* RVTL AND THEN FLAGS THE TRUNK RECORD FOR DELETION BY ADDING IT         */
/* TO A LINKED LIST.  IF THE TRUNK IS SUBSEQUENTLY RE-ADDED, THE DATA     */
/* FIELDS WILL BE RETAINED.  IF THE TRUNK IS NOT RE-ADDED, IT IS          */
/* PHYSICALLY DELETED IN A LATER CLEANUP STEP.                            */
/*                                                                        */
/*      DEL_TRUNK HAS NO PARAMETERS.  ALL DATA ARE OBTAINED FROM          */
/* GLOBAL VARIABLES DECLARED IN THE MAIN PROCEDURE.                       */

DEL_TRUNK: PROCEDURE;

DECLARE #_LINKS BINARY FIXED(15,0);

DECLARE KEYT    CHARACTER(7);

KEYT=STRING(INDATA.RMTRCTRL);

/* THERE MUST BE NO ENTRIES IN RVCT FOR THIS TRUNK; OTHERWISE, THE        */
/* TRUNK IS BEING USED BY AT LEAST ONE CIRCUIT AND CANNOT BE DELETED.     */

FUNCT=READV;
FILEID='RVCT';
REFER='LKCT';
LKPATH='RMTRLKCT'; /* ACCESS IT FROM THE TRUNK */
FLMLIST='RVCTRMTREND.';
CONTROL=KEYT;

CALL DBIO(0,2); /* LOCK FOR LINKAGE TO RVCT */

IF STATUS='MDNF' THEN
   DO;
      CALL MSG(0,'SPECIFIED TRUNK '||KEYT||
                 ' NOT IN DATA BASE.  NO RECORDS DELETED.');
      STATUS=OK;
      RETURN;
   END;

IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);

IF REFER¬=ENDP THEN
   DO;
      CALL MSG(8,'SPECIFIED TRUNK '||KEYT||
                 ' USED BY ONE OF MORE CIRCUITS.  NO RECORDS DELETED.');
      RETURN;
   END;
```

```
/* AT THIS POINT TRUNK RECORD HAS BEEN FOUND AND IS USED BY NO */
/* CIRCUITS. NOW PREPARE TO DELETE LINKAGE IN RVTL.            */

/* DISPLAY MESSAGE ABOUT TRUNK TO BE DELETED */
CALL MSG(0,'TRUNK '||KEYT);

/* NOW DELETE ALL ASSOCIATED VARIABLE RECORDS IN RVTL */
FUNCT=READV;
FILEID='RVTL';
REFER='LKTL';
LKPATH='RMTRLKTL';
```

```
ELMLIST=RVTL_ELEM;
/* CONTROL ALREADY SET */

CALL DBIO(9,2); /* ACCESS FIRST RECORD IN RVTL */
IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);

/*<IF *DNR IS TO BE HANDLED, DO IT HERE.>*/

#_LINKS=0;

/* DELETE THEM */
DO WHILE(RETERR¬=ENDF);

   /* PRINT DATA ABOUT LINK */
   CALL MSG(0,'  USING LINK '||RVTLRMLK||' BETWEEN '||RVTLRMSI||
                                          ' AND '||RVTLTOLC);

   FUNCT=DELVD;
   CALL DBIO(9,2);
   #_LINKS=#_LINKS+1;
   IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);

   FUNCT=READV; /* READ NEXT VARIABLE RECORD */
   CALL DBIO(9,2);
   IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);
END;

/*REWRITE RECORD WITH MODIFIED SITE FIELDS */
FUNCT=WRITM;
FILEID='RMTR';
ELMLIST=RMTR_LOCS;
IOAREA=KEYT||'*DELETED****DELETED***';
/* CONTROL ALREADY SET */

CALL DBIO(7,2);
IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);

/* ADD TRUNK TO BEGINNING OF LIST OF THOSE FLAGGED FOR DELETION */
ALLOCATE DTRUNK;
DTKEY=KEYT;
NEXTDT=DEL_TR_PTR;
DEL_TR_PTR=DTPTR;

/* DISPLAY MESSAGES */
CALL MSG(0,'  FLAGGED FOR DELETION.');

CALL MSG(0,'NUMBER OF VARIABLE RECORDS DELETED:'||
        SUBSTR(CHAR(#_LINKS),6));
CALL MSG(0,'TRUNK MASTER FILE RECORD RETAINED WITH BLANK LINKPATHS.');

RETURN;

END /* DEL_TRUNK */ ;
```

```
/*                  SUBROUTINE DEL_LINK                                    */
/*                                                                         */
/*      THIS SUBROUTINE PERFORMS THE ACTION INDICATED BY A -L REQUEST.     */
/*  THERE MUST BE NO ENTRIES IN BVTL FOR THE LINK; OTHERWISE, THE LINK     */
/*  IS BEING USED BY AT LEAST ONE TRUNK AND CANNOT BE DELETED.             */
/*  THERE IS NO NEED TO CHECK EXPLICITLY THAT NO LINKAGE IS PRESENT        */
/*  BECAUSE AN ATTEMPT TO DELETE A MASTER RECORD TO WHICH VARIABLE         */
/*  RECORDS ARE LINKED WILL RESULT IN A STATUS RETURN OF 'IMDL' FROM       */
/*  TOTAL.                                                                 */
/*                                                                         */
/*      LIKE -S AND UNLIKE -C AND -T, A -L RESULTS IN THE IMMEDIATE        */
/*  PHYSICAL DELETION OF THE SPECIFIED RECORD.                             */
/*                                                                         */
/*      DEL_LINK HAS NO PARAMETERS.  ALL VALUES ARE OBTAINED FROM          */
/*  GLOBAL VARIABLES DECLARED IN THE MAIN PROCEDURE.                       */

DEL_LINK: PROCEDURE;

/* FIRST READ THE LINK RECORD SO THAT CURRENT DATA CAN BE DIS-    */
/* PLAYED FOR THE USER.                                           */

FUNCT=READM;
FILEID='RMLK';
CONTROL=INDATA.RMLKCTRL;
ELMLIST=RMLK_ELEM;

CALL DBIO(7,2);

IF STATUS='MBNF' THEN
   DO;
      CALL MSG(4,'SPECIFIED LINK '||INDATA.RMLKCTRL||
                 ' NOT IN DATA BASE.  NO RECORDS DELETED.');
      STATUS=OK;
      RETURN;
   END;
```

```
/* WE KNOW LINK RECORD IS PRESENT AND IS USED BY NO TRUNKS. */
/* TRY TO DELETE IT AND TELL USER.                          */

FUNCT=DEL_M;
CALL DBIO(7,2);

IF STATUS='IMDL' THEN
   DO;
      CALL MSG(8,'SPECIFIED LINK '||INDATA.RMLKCTRL||
                 ' USED BY ONE OF MORE TRUNKS.  NO RECORDS DELETED.');
      STATUS=OK;
      RETURN;
   END;

IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);

CALL MSG(0,'LINK '||INDATA.RMLKCTRL||' BETWEEN '||STRING(LINKSITE1)||
           ' AND '||STRING(LINKSITE2)||' DELETED.');


RETURN;

END /* DEL_LINK */ ;
```

```
/*                SUBROUTINE DEL_SITES                                  */
/*                                                                      */
/*    THIS SUBROUTINE PERFORMS THE ACTION INDICATED BY A -S REQUEST.    */
/* EITHER ONE OR TWO SITES CAN BE SPECIFIED.  LIKE -L AND UNLIKE -C     */
/* AND -T, A -S RESULTS IN IMMEDIATE PHYSICAL DELETION OF THE SPECI-    */
/* FIED RECORD.  A SUCCESSFUL -S OPERATION ALSO RESULTS IN TURNING      */
/* ON THE DELETE_ONLY AND GARBCOL FLAGS.                                */
/*                                                                      */
/*    DEL_SITES HAS NO PARAMETERS.  ALL VALUES ARE OBTAINED FROM        */
/* GLOBAL VARIABLES DECLARED IN THE MAIN PROCEDURE.                     */

DEL_SITES: PROCEDURE;

DECLARE KEYS CHARACTER(11),
        I    BINARY FIXED(15,0);

/* USER MAY HAVE SPECIFIED ONE OR TWO SITES FOR DELETION */
DO I=1 TO 2 WHILE(INDATA.GEOLOC(I)¬='          ');

   KEYS=STRING(INDATA.RMSICTRL(I));

   /* EACH SITE MASTER FILE RECORD HAS THREE LINKPATHS: ONE TO*/
   /* RVCT AND TWO TO RVTL.  IF BOTH PATHS TO RVTL ARE BLANK, */
   /* THEN THE ONE TO RVCT MUST BE BLANK ALSO (BECAUSE A CIR- */
   /* CUIT CANNOT BEGIN AT A SITE UNLESS AT LEAST ONE TRUNK   */
   /* BEGINS OR ENDS THERE).                                  */
   /*                                                         */
   /* THERE IS NO NEED TO CHECK EXPLICITLY THAT NO LINKAGE IS */
   /* PRESENT BECAUSE AN ATTEMPT TO DELETE A MASTER RECORD TO */
   /* WHICH VARIABLE RECORDS ARE LINKED WILL RESULT IN A      */
   /* STATUS RETURN OF 'IMDL' FROM TOTAL.                     */

   /* FIRST READ THE SITE RECORD */

   FUNCT=READM;
   FILEID='RMSI';
   CONTROL=KEYS;
   ELMLIST=RMSI_ELEM;
   CALL DBIO(7,2);

   IF STATUS='MRNF' THEN
      DO;
         CALL MSG(4,'SPECIFIED SITE '||KEYS
                    ||' NOT IN DATA BASE.  RECORD NOT DELETED.');
         STATUS=OK;
         GO TO NEXTSITE; /* ESCAPE TO NEXT ITERATION OF DO LOOP */
      END;
```

```
/* NOW TRY TO DELETE IT AND TELL THE USER */

FUNCT=DEL_M;
CALL DBIO(7,2);

IF STATUS='IMDL' THEN
   DO;
      CALL MSG(8,'SPECIFIED SITE '||KEYS||
               ' USED BY ONE OR MORE TRUNKS.  RECORD NOT DELETED.');
      STATUS=OK;
      GO TO NEXTSITE; /* ESCAPE TO NEXT ITERATION OF DO LOOP */
   END;




   IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);

   CALL MSG(C,'SITE '||KEYS||' DELETED.');

   DELETE_ONLY='1'B; /* AFTER SUCCESSFUL -S, ONLY DELETES PERMITTED */
   GARBCOL='1'B; /* SUCCESSFUL -S FORCES GARBAGE COLLECTION */

/******/
NEXTSITE:
/******/

END;

RETURN;

END /* DEL_SITES */ ;
```

```
/*                    SUBROUTINE RE_TRUNK                                  */
/*                                                                         */
/*      THIS SUBROUTINE PERFORMS THE ACTION INDICATED BY A #T REQUEST.*/
/* THE SPECIFIED TRUNK MUST ALREADY BE PRESENT IN THE DATA BASE; THE */
/* SPECIFIED LINK(S) MUST ALL BE PRESENT; AND THE CURRENT TRUNK END  */
/* SITES MUST BE THE SAME AS THE END SITES OF THE NEWLY SPECIFIED    */
/* CHAIN OF LINKS.  IF ALL CONDITIONS ARE SATISFIED, THE CURRENT     */
/* LINKAGE TO RVTL IS DELETED AND THE NEW LINKAGE IS ADDED.          */
/*                                                                   */
/*      RE_TRUNK HAS NO PARAMETERS.  ALL VALUES ARE OBTAINED FROM    */
/* GLOBAL VARIABLES DECLARED IN THE MAIN PROCEDURE.                  */

RE_TRUNK: PROCEDURE;

DECLARE KEYT CHARACTER(7);
KEYT=STRING(INDATA.RMTRCTRL);


/* DELETE CURRENT LINKAGE TO RVTL AND ADD NEWLY SPECIFIED LINKAGE.  */
/* ENDPOINTS OF TRUNK MUST NOT CHANGE. <---                         */

/* CHECK FOR PRESENCE OF TRUNK RECORD */
FUNCT=READM;
FILEID='RMTR';
CONTROL=KEYT;
FLMLIST=RMTR_ELEM;
CALL DBIO(7,2);

IF STATUS='MRNF' THEN /* MASTER TRUNK RECORD NOT FOUND */
   DO;
      CALL MSG(8,'SPECIFIED TRUNK '||KEYT||
               ' NOT PRESENT IN DATA BASE.  REROUTE NOT PERFORMED.');
      STATUS=OK;
      RETURN;
   END;

IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);

/* THE SPECIFIED LINKS MUST TRACE A CONTINUOUS PATH.  */
/* FUNCTION ROUTE_TRACE RETURNS THE VALUE  TRUE  IF   */
/* THE PATH IS TRACEABLE AND RETURNS THE ENDSITES AS  */
/* ARGUMENTS.                                         */
IOSAVE=IOAREA; /* SAVE CURRENT TRUNK DATA */
IF ¬ROUTE_TRACE(WKSITE1,WKSITE2) THEN
   DO;
      CALL MSG(8,'SPECIFIED LINKS DO NOT TRACE A CONTINUOUS PATH.'
            ||'  REROUTE NOT PERFORMED.');
      RETURN;
   END;
```

```
/* NOW COMPARE NEW ENDPOINTS WITH CURRENT ONES */
IOAREA=IOSAVE; /* RESTORE TRUNK DATA */
IF ¬((RMTREMLC=WKSITE1 & RMTRTOLC=WKSITE2) |
     (RMTREMLC=WKSITE2 & RMTRTOLC=WKSITE1)) THEN
   DO;
      CALL MSG(8,'ENDSITES DO NOT MATCH.  NEW:'||WKSITE1||' '||WKSITE2
             ||' OLD:'||STRING(TRUNKSITE1)||' '||STRING(TRUNKSITE2)
             ||'.  REROUTE NOT PERFORMED.');
      RETURN;
   END;

/* NEW LINKS CHECKED OUT.  DELETE OLD AND ADD NEW */
CALL MSG(0,'TRUNK '||KEYT||' BETWEEN '||
         STRING(TRUNKSITE1)||' AND '||STRING(TRUNKSITE2));

/* DELETE OLD: */
FUNCT=READV;
FILFID='RVTL';
REFER='LKTL';
LKPATH='RMTRLKTL';
ELMLIST=RVTL_ELEM;
CONTROL=KEYT;

CALL DBIO(9,2); /* GET FIRST RECORD */
IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);

DO WHILE(REFER¬=ENDP);
   /* PRINT DATA ABOUT LINK */
   CALL MSG(0,'  USED LINK '||RVTLRMLK||' BETWEEN '||RVTLREMSI||
            ' AND '||RVTLTOLC);
   FUNCT=DELVD;  /* DELETE IT */
   CALL DBIO(9,2);
   IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);
   FUNCT=READV;  /* READ NEXT VARIABLE RECORD */
   CALL DBIO(9,2);
   IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);
END;

CALL MSG(0,'  AND IS NOW');

/* NOW ADD NEW LINKAGE: */
CALL ADD_ROUTING;

END /* RE_TRUNK */;
```

```
/*                    SUBROUTINE CL_CIRCUIT                                  */
/*                                                                           */
/*      THIS SUBROUTINE PERFORMS THE ACTION INDICATED BY A %C REQUEST.*/
/* IF THE SPECIFIED CIRCUIT IS PRESENT IN THE DATA BASE, THEN THE    */
/* RMCTFBLC, --FFAC, --TCLC, --TFAC FIELDS ARE MODIFIED AS INDICATED.*/
/*                                                                           */
/*      CL_CIRCUIT HAS NO PARAMETERS.  ALL VALUES ARE OBTAINED FROM   */
/* GLOBAL VARIBALES DECLARED IN THE MAIN PROCEDURE.                          */

CL_CIRCUIT: PROCEDURE;
DECLARE KEYC CHARACTER(9);

KEYC=STRING(INDATA.RMCTCTRL);

/* CHECK FOR PRESENCE OF CIRCUIT RECORD */
FUNCT=READM;
FILEID='RMCT';
CONTROL=KEYC;
ELMLIST=RMCT_ELEM;
CALL DBIO(7,2);

IF STATUS='MRNF' THEN /* MASTER CIRCUIT RECORD NOT FOUND */
   DO;
      STATUS=OK;
      CALL MSG(8,'SPECIFIED CIRCUIT NOT IN DATA BASE.');
      /* FALL THROUGH AND RETURN */
   END;
```

```
ELSE IF STATUS=OK THEN /* CIRCUIT PRESENT IN DATA BASE */
   DO;
      /* UPDATE LCC FIELDS */

      /* DISPLAY "BEFORE" VALUES */
      /*<TEMP>*/ CALL MSG(0,'BEFORE: '||STRING(CIRCUITSITE1)||' '
                                      ||STRING(CIRCUITSITE2));
      IF FRLC¬='' THEN RMCTFRLC=FRLC;
      IF FRLC='*' THEN RMCTFRLC='';

      IF FFAC¬='' THEN RMCTFFAC=FFAC;
      IF FFAC='*' THEN RMCTFFAC='';

      IF TOLC¬='' THEN RMCTTOLC=TOLC;
      IF TOLC='*' THEN RMCTTOLC='';

      IF TFAC¬='' THEN RMCTTFAC=TFAC;
      IF TFAC='*' THEN RMCTTFAC='';

      FUNCT=WRITM;
      CALL DBIO(7,2);
      IF STATUS¬=OK THEN SIGNAL CONDITION(TCTERR);

      /* DISPLAY "AFTER" VALUES */
      /*<TEMP>*/ CALL MSG(0,'AFTER:  '||STRING(CIRCUITSITE1)||' '
                                      ||STRING(CIRCUITSITE2));
   END;

ELSE /* INVALID STATUS TRYING TO LOCATE CIRCUIT RECORD */
   SIGNAL CONDITION(TCTERR);

RETURN;

END /* CL_CIRCUIT */ ;
```

```
/*                    SUBROUTINE CL_TRUNK                                  */
/*                                                                         */
/*      THIS SUBROUTINE PERFORMS THE ACTION INDICATED BY A XT REQUEST.     */
/* IF THE SPECIFIED TRUNK IS PRESENT IN THE DATA BASE, THEN THE            */
/* RMTRFRLC, --FFAC, --TCLC, --TFAC FIELDS ARE MODIFIED AS INDICATED.      */
/*                                                                         */
/*      CL_TRUNK HAS NO PARAMETERS.  ALL VALUES ARE OBTAINED FROM          */
/* GLOBAL VARIABLES DECLARED IN THE MAIN PROCEDURE.                        */

CL_TRUNK: PROCEDURE;
DECLARE KEYT CHARACTER(7);

KEYT=STRING(INDATA.RMTRCTRL);

/* CHECK FOR PRESENCE OF TRUNK RECORD */
FUNCT=READM;
FILEID='RMTR';
CONTROL=KEYT;
FLMLIST=RMTR_ELEM;
CALL DBIO(7,2);

IF STATUS='MRNF' THEN /* MASTER TRUNK RECORD NOT FOUND */
   DO;
      STATUS=OK;
      CALL MSG(8,'SPECIFIED TRUNK NOT IN DATA BASE.');
      /* FALL THROUGH AND RETURN */
   END;
```

```
ELSE IF STATUS=OK THEN /* TRUNK PRESENT IN DATA BASE */
   DO;
      /* UPDATE LOC FIELDS */

      /* DISPLAY "BEFORE" VALUES */
      /*<TEMP>*/ CALL MSG(0,'BEFORE: '||STRING(TRUNKSITE1)||' '
                                     ||STRING(TRUNKSITE2));
      IF FRLC¬='' THEN RMTRFRLC=FRLC;
      IF FRLC='*' THEN RMTRFRLC='';

      IF FFAC¬='' THEN RMTRFFAC=FFAC;
      IF FFAC='*' THEN RMTRFFAC='';

      IF TOLC¬='' THEN RMTRTOLC=TOLC;
      IF TOLC='*' THEN RMTRTOLC='';

      IF TFAC¬='' THEN RMTRTFAC=TFAC;
      IF TFAC='*' THEN RMTRTFAC='';

      FUNCT=WRITM;
      CALL DBIO(7,2);
      IF STATUS¬=OK THEN SIGNAL CONDITION(TCTERR);

      /* DISPLAY "AFTER" VALUES */
      /*<TEMP>*/ CALL MSG(0,'AFTER:  '||STRING(TRUNKSITE1)||' '
                                     ||STRING(TRUNKSITE2));
   END;

ELSE /* INVALID STATUS TRYING TO LOCATE TRUNK RECORD */
   SIGNAL CONDITION(TCTERR);

RETURN;

END /* CL_TRUNK */ ;
```

```
/*                    SUBROUTINE CL_LINK                                  */
/*                                                                        */
/*      THIS SUBROUTINE PERFORMS THE ACTION INDICATED BY A %L REQUEST.*/
/* IF THE SPECIFIED LINK IS PRESENT IN THE DATA BASE, THEN THE            */
/* RMLKFBLC, --FFAC, --TOLC, --TFAC FIELDS ARE MODIFIED AS INDICATED.*/
/*                                                                        */
/*      CL_LINK HAS NO PARAMETERS.  ALL VALUES ARE OBTAINED FROM          */
/* GLOBAL VARIABLES DECLARED IN THE MAIN PROCEDURE.                       */

CL_LINK: PROCEDURE;
DECLARE KEYL CHARACTER(5);

KEYL=INDATA.RMLKCTRL;

/* CHECK FOR PRESENCE OF LINK RECORD */
FUNCT=READM;
FILEID='RMLK';
CONTROL=KEYL;
ELMLIST=RMLK_ELEM;
CALL DBIO(7,2);

IF STATUS='MRNF' THEN /* MASTER LINK RECORD NOT FOUND */
   DO;
      STATUS=OK;
      CALL MSG(8,'SPECIFIED LINK NOT IN DATA BASE.');
      /* FALL THROUGH AND RETURN */
   END;
```

```
ELSE IF STATUS=OK THEN /* LINK PRESENT IN DATA BASE */
   DO;
      /* UPDATE LOC FIELDS */

      /* DISPLAY "BEFORE" VALUES */
      /*<TEMP>*/ CALL MSG(0,'BEFORE: '||STRING(LINKSITE1)||' '
                                     ||STRING(LINKSITE2));
      IF FRLC¬='' THEN RMLKFRLC=FRLC;
      IF FRLC='*' THEN RMLKFRLC='';

      IF FFAC¬='' THEN RMLKFFAC=FFAC;
      IF FFAC='*' THEN RMLKFFAC='';

      IF TOLC¬='' THEN RMLKTOLC=TOLC;
      IF TOLC='*' THEN RMLKTOLC='';

      IF TFAC¬='' THEN RMLKTFAC=TFAC;
      IF TFAC='*' THEN RMLKTFAC='';

      FUNCT=WRITM;
      CALL DBIO(7,2);
      IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);

      /* DISPLAY "AFTER" VALUES */
      /*<TEMP>*/ CALL MSG(0,'AFTER:  '||STRING(LINKSITE1)||' '
                                     ||STRING(LINKSITE2));
   END;

ELSE /* INVALID STATUS TRYING TO LOCATE LINK RECORD */
   SIGNAL CONDITION(IOTERR);

RETURN;

END /* CL_LINK */ ;
```

```
/*                    FUNCTION ROUTE_TRACE                                  */
/*                                                                          */
/* FUNCTION ROUTE_TRACE READS A MASTER FILE RECORD FOR EACH TRUNK/          */
/* LINK USED BY THE CIRCUIT/TRUNK TO BE ADDED AS SPECIFIED BY THE           */
/* USER.  THE SITES CONNECTED BY THE TRUNKS/LINKS ARE EXAMINED FOR          */
/* CONTINUITY AND A RECORD OF THE END POINTS IS MAINTAINED.                 */
/*                                                                          */
/* ROUTESPEC POINTS TO THE HEAD OF A LIST OF ROUTE_CHAIN NODES, EACH        */
/* OF WHICH CONTAINS A SPECIFIED CONTROL KEY. THE SITE FIELDS ARE           */
/* FILLED IN BY THIS FUNCTION.                                              */
/*                                                                          */
/* IF ROUTE_TRACE DETECTS A DISCONTINUITY, IT TERMINATES THE TRACE          */
/* AND RETURNS THE VALUE FALSE ('0'B).                                      */
/*                                                                          */
/* IF A CONTINUOUS PATH IS TRACED, THEN THE ENDSITES ARE                    */
/* RETURNED AS PARAMETERS, AND THE FUNCTIONAL VALUE IS TRUE ('1'B).         */
/*                                                                          */
/* THE CALLING PROGRAM CHECKS THAT ROUTESPEC IS NOT NULL.                   */

ROUTE_TRACE: PROCEDURE(FROMSITE,TOSITE) RETURNS(BIT(1));

DECLARE /* PARAMETERS */
        FROMSITE CHARACTER(11),
        TOSITE   CHARACTER(11);

DECLARE SEGMENT    CHARACTER(5) VARYING,
        SITE_INDEX BINARY FIXED(15,0),
        FIRST_TIME BIT(1);

/* PARAMETERS SHOULD BE INITIALLY BLANK */
FROMSITE=' ';
TOSITE=' ';
```

```
/* PREPARE TO READ MASTER FILE RECORDS */
FUNCT=READM;

IF REQUEST_LEVEL='C' THEN
   DO;
      FILEID='RMTR';
      ELMLIST=RMTR_LOCS;
      SEGMENT='TRUNK';
      SITE_INDEX=8;
   END;
ELSE IF REQUEST_LEVEL='T' THEN
   DO;
      FILEID='RMLK';
      ELMLIST=RMLK_LOCS;
      SEGMENT='LINK';
      SITE_INDEX=6;
   END;
ELSE  /* THIS "CANNOT" HAPPEN */
   DO;
      CALL MSG(12,'FUNCTION ROUTE_TRACE CALLED WITH REQUEST_LEVEL='||
               REQUEST_LEVEL);
      RETURN('0'B);
   END;



/* NOW PROCESS THE ROUTE CHAIN */
RPTR=ROUTESPEC;
DO WHILE(RPTR¬=NULL);

   CONTROL=RKEY;
   CALL DBIO(7,2);
   IF STATUS='MRNF' THEN
      DO;
         CALL MSG(8,'SPECIFIED '||SEGMENT||' '||RKEY||
                  ' NOT PRESENT IN DATA BASE.');
         IF NEXTR¬=NULL THEN
            CALL MSG(4,'SUBSEQUENT '||SEGMENT||'S NOT EXAMINED.');
         STATUS=OK;
         RETURN('0'B);
      END;

   IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);

   /* WE HAVE A GOOD RECORD.  NOW GET THE SITE DATA */
   /* AND CHECK IT OUT.                             */

   RSITE1=SUBSTR(IOAREA,SITE_INDEX,8);
   RSITE2=SUBSTR(IOAREA,SITE_INDEX+11,8);
   /***(FOR NOW, ONLY USE 8 BYTES INSTEAD OF 11 BECAUSE NOT ALL ***/
   /*** FACILITY FIELDS ARE SPECIFIED IN THE CURRENT DATA BASE.)***/
```

```
   /* IF THIS IS THE FIRST SEGMENT SPECIFIED, JUST COPY THE SITES */
   /* TO THE PARAMETERS; OTHERWISE, WE DO SOME COMPARISONS.     */
   IF RPTR=ROUTESPEC THEN
      DO;
         FROMSITE=RSITE1;
         TOSITE=RSITE2;
      END;
   ELSE
      DO;
         /* WE COMPARE THE CURRENT FROM- AND TOSITE WITH THE CURRENT */
         /* RSITE1 AND RSITE2.    THERE ARE SEVEN CASES THAT CAN BE  */
         /* COLLAPSED INTO FIVE.  THE TWO THAT WE DON'T WORRY ABOUT  */
         /* ARE FROMSITE AND TOSITE EQUAL TO RSITE1 AND RSITE2 IN    */
         /* EITHER ORDER.  THESE ARE HANDLED PROPERLY BY THE FIRST   */
         /* TWO CHECKS.  <NOTE TO SELF: THERE IS STILL A BUG.>       */

/*              GIVEN:                    FROMSITE  TOSITE    TL1 TL2 */
/*                                           A        B                */
/*                                                                     */
/*1:*/ IF       TOSITE  =RSITE1 THEN TOSITE  =RSITE2;   /*  B    C   */
/*2:*/ ELSE IF TOSITE  =RSITE2 THEN TOSITE  =RSITE1;   /*  C    B   */
/*3:*/ ELSE IF FROMSITE=RSITE1 THEN FROMSITE=RSITE2;   /*  A    C   */
/*4:*/ ELSE IF FROMSITE=RSITE2 THEN FROMSITE=RSITE1;   /*  C    A   */
/*5:*/                          /* (HANDLED BY CASE 2:)     A    B   */
/*6:*/                          /* (HANDLED BY CASE 1:)     B    A   */
/*7:*/ ELSE     /* THEY DON'T MATCH IN EITHER ORDER:       C1   C2  */
            DO;
               CALL MSG(8,SEGMENT||' '||RKEY||
                        ' BETWEEN SITES '||RSITE1||' AND '||RSITE2||
                        ' NOT REACHABLE FROM PREVIOUS '||SEGMENT||'.');
               RETURN('0'B);
            END;
      END;


   /* ADVANCE TO NEXT ROUTE_CHAIN NODE */
   RPTR=NEXTR;

END;

RETURN('1'B);

END /* ROUTE_TRACE */ ;
```

```
/*                    SUBROUTINE ADD_ROUTING                              */
/*                                                                        */
/*      THIS SUBROUTINE FOLLOWS THE ROUTE_CHAIN NODES AND ADDS A          */
/* VARIABLE RECORD FOR EACH TRUNK/LINK SPECIFIED.   THIS PROVIDES         */
/* THE LINKAGE SHOWING THE TRUNKS/LINKS USED BY THE CIRCUIT/TRUNK         */
/* BEING ADDED.                                                           */
/*                                                                        */
/*      ADD_ROUTING DOES NOT FREE THE BASED STRUCTURE VARIABLES.          */

ADD_ROUTING: PROCEDURE;

DECLARE KEY     CHARACTER(9) VARYING, /* CONTROL KEY */
        SEGMENT CHARACTER(5) VARYING, /* 'LINK' OR 'TRUNK' */
        TSITE   CHARACTER(11),        /* TEMPORARY SITE FIELD */
        RKEYLEN BINARY FIXED(15,0);

/* SET UP TOTAL FOR ADDING TO RVCT OR RVTL */
FUNCT=ADDVC;

IF REQUEST_LEVEL='C' THEN
   DO;
      FILEID='RVCT';
      REFER='LKCT';
      LKPATH='RMCTLKCT';
      ELMLIST=RVCT_ELEM;
      KEY=STRING(INDATA.RMCTCTRL);
      RKEYLEN=7;
      SEGMENT='TRUNK';
   END;
ELSE IF REQUEST_LEVEL='T' THEN
   DO;
      FILEID='RVTL';
      REFER='LKTL';
      LKPATH='RMTRLKTL';
      ELMLIST=RVTL_ELEM;
      KEY=STRING(INDATA.RMTRCTRL);
      RKEYLEN=5;
      SEGMENT='LINK';
   END;
ELSE /* THIS "CANNOT" HAPPEN */
   DO;
      CALL MSG(12,'SUBROUTINE ADD_ROUTING CALLED WITH REQUEST_LEVEL='||
                  REQUEST_LEVEL);
      RETURN;
   END;
```

```
/* NOW FOLLOW THE CHAIN AND ADD THE RECORDS */
RPTR=ROUTESPEC;


/* PREPARE TO ORDER FROM- AND TO-LOCATIONS IN VARIABLE RECORD */
IF RPTR=NULL THEN
   DO;
      CALL MSG(12,'ADD_ROUTING CALLED WITH NO ROUTE SPECIFICATION.');
      RETURN;
   END;

IF      RSITE1=WKSITE1 | RSITE1=WKSITE2 THEN TSITE=RSITE1;
ELSE IF RSITE2=WKSITE1 | RSITE2=WKSITE2 THEN TSITE=RSITE2;
ELSE
   DO;
      CALL MSG(12,KEY||' ENDPOINTS IMPROPERLY SET.  TO FIX IT,');
      CALL MSG(0,'       1. DELETE IT.');
      CALL MSG(0,'       2. REVERSE THE FROM- AND TO- LOCATION FIELDS'||
                 ' IN THE '||RKEY||' '||SEGMENT||' MASTER RECORD.');

      CALL MSG(0,'       3. RE-ADD '||KEY);
      RETURN;
   END;
```

```
DO WHILE(RPTR¬=NULL);

   /* SWITCH FROM- AND TO-LOCATION FIELDS IF NECESSARY */
   IF RSITE1=TSITE THEN
      TSITE=RSITE2;
   ELSE
      DO;
         TSITE=RSITE1;
         RSITE1=RSITE2;
         RSITE2=TSITE;
      END;

   IOAREA=KEY||SUBSTR(RKEY,1,RKEYLEN)||RSITE1||RSITE2;
   CALL DBIC(9,2);
   IF STATUS='NSMR' THEN /* NO SECONDARY MASTER RECORD */
      DO;
         STATUS=OK;
         CALL MSG(8,'SITE FIELD '||RSITE1||' OR '||RSITE2||' IN '||
                    RKEY||' MASTER RECORD MAY BE INVALID.');
         CALL MSG(4,'USER SHOULD DELETE '||KEY||' AND RE-ADD IT '||
                    'AFTER CORRECTING LOCATION FIELD. OR '||
                    'ADDING A NEW SITE.');
         RETURN;
      END;

   IF STATUS¬=OK THEN SIGNAL CONDITION(TCTERR);

   /* DISPLAY MESSAGE ABOUT LINKAGE */
   CALL MSG( ,'  USING '||SEGMENT||' '||RKEY||' BETWEEN '||
              RSITE1||' AND '||RSITE2);

   RPTR=NEXTR;
END;

END /* ADD_ROUTING */ ;
```

```
/*                      SUBROUTINE READ_REQUEST                          */
/*                                                                       */
/*      THIS SUBROUTINE IS INVOKED AT THE BEGGINING OF THE MAIN          */
/* PROCESSING LOOP TO PROCESS THE INPUT SPECIFICATIONS FOR THE           */
/* USER'S NEXT REQUEST.  WHEN READ_REQUEST IS CALLED, THE CURRENT        */
/* SEPARATOR MUST BE @, -, #, OR %.                                      */
/*                                                                       */
/*      THIS ROUTINE CLEARS THE INPUT DATA AREA, PERFORMS SOME           */
/* COMMON PROCESSING (INCLUDING EXTRACTING THE REQUEST AND               */
/* REQUEST_LEVEL FIELDS), AND THEN CALLS ONE OF FOUR SUBROUTINES TO      */
/* PROCESS THE INPUT SPECIFICATIONS FOR @, -, #, OR %.  WHEN             */
/* READ_REQUEST RETURNS TO THE POINT OF INVOCATION, INDATA CONTAINS      */
/* THE INPUT SPECIFICATIONS READY FOR ACTUAL PROCESSING.  IF A           */
/* SERIOUS SYNTACTIC ERROR WAS DETECTED, PREVENTING SUCCESSFUL           */
/* PROCESSING, REQUEST IS SET TO NEXT SO THAT THE MAIN LOOP WILL         */
/* PROCEED TO THE NEXT REQUEST INSTEAD OF PROCESSING THE CURRENT ONE.*/

READ_REQUEST: PROCEDURE;



/* THIS ERROR SHOULD NEVER OCCUR, BUT CHECK FOR IT ANYWAY:           */
/* WHEN READ_REQUEST IS CALLED, THE CURRENT SEPARATOR CHARACTER      */
/* MUST INDICATE THE BEGINNING OF A REQUEST.  IT IS EITHER THE       */
/* FIRST CALL (IN WHICH CASE THE SEPARATOR HAS BEEN CHECKED BY       */
/* THE CALLING PROCEDURE) OR THE PREVIOUS CALL OF THIS ROUTINE       */
/* HAS FORCED THAT TO BE TRUE.                                       */
IF VERIFY(SEPARATOR,'@-#%')¬=0 THEN
   DO;
      CALL MSG(12,'SUBROUTINE READ_REQUEST INVOKED WITH SEPARATOR = '''
                  ||SEPARATOR||'''');
      CALL FLUSH_REQUEST; /* FIND BEGINNING OF NEXT REQUEST */
      REQUEST=NEXT;       /* FORCES CALLING PROC TO READ NEXT REQUEST*/
      RETURN;
   END;

/* HAS THE DELETE_ONLY FLAG BEEN SET BY THE DEL_SITES SUBROUTINE? */
IF DELETE_ONLY & SEPARATOR¬='-' THEN
   DO;
      CALL MSG(8,'ONLY DELETES PERMITTED FOLLOWING A SUCCESSFUL -S '||
                 'REQUEST.  '''||SEPARATOR||''' NOT PROCESSED.');
      CALL FLUSH_REQUEST;
      REQUEST=NEXT;
      RETURN;
   END;

/* CLEAR USER INPUT AREA */
INDATA=' ';
```

```
/* COMMON PROCESSING:                                                    */
/*      ALL REQUESTS MUST BEGIN WITH "REQUEST" (@, -, #, OR %)           */
/* FOLLOWED BY "REQUEST_LEVEL" (C, T, I, OR S) FOLLOWED BY A CONTROL KEY.*/
/*  LOCATE THOSE COMMON ITEMS, THEN CALL SEPARATE ROUTINE TO PROCESS     */
/*  INPUT FOR INDIVIDUAL REQUESTS.                                       */

REQUEST=SEPARATOR;
IF LENGTH(ITEM) > 0 THEN
   REQUEST_LEVEL=SUBSTR(ITEM,1,1);
ELSE
   REQUEST_LEVEL=' ';

IF VERIFY(REQUEST_LEVEL,'CTIS')¬=0 THEN
   DO;
      CALL MSG(8,'UNRECOGNIZED REQUEST '''||REQUEST||REQUEST_LEVEL||
                 ''' NOT PROCESSED.');
      CALL FLUSH_REQUEST;
      REQUEST=NEXT;
      RETURN;
   END;

/* CONTROL KEY SHOULD START IN OR FOLLOWING THE SECOND BYTE OF ITEM */
IF LENGTH(ITEM) < 2 THEN
   ITEM='';
ELSE
   ITEM=SUBSTR(ITEM,2);

IF ITEM='' THEN /* ALL BLANKS OR NULL */
   DO;
      CALL MSG(8,'NO CONTROL KEY SPECIFIED.  REQUEST NOT PROCESSED.');
      CALL FLUSH_REQUEST;
      REQUEST=NEXT;
      RETURN;
   END;
KEY=SUBSTR(ITEM,VERIFY(ITEM,' ')); /* STRIP LEADING BLANKS */

/* RETRIEVE NEXT SEPARATOR AND ITEM FROM INPUT */
CALL INPUT(SEPARATOR,ITEM);
```

```
/* BEFORE PROCESSING REMAINDER OF REQUEST, WHICH MAY INCLUDE  */
/* ROUTING INFORMATION, FREE ALL PREVIOUSLY USED ROUTE_CHAIN  */
/* NODES.                                                     */
RPTR=ROUTESPEC;
ROUTESPEC=NULL;
DO WHILE(RPTR¬=NULL);
   LASTR=RPTR;
   RPTR=NEXTR;
   FREE LASTR->ROUTE_CHAIN;
END;

/* CLEAR UPDATE FLAGS */
UPDATESITE    ='0'B;
UPDATELINK    ='0'B;
UPDATETRUNK   ='0'B;
UPDATECIRCUIT='0'B;

/* NOW PROCESS REMAINDER OF INPUT FOR CURRENT REQUEST */
IF       REQUEST=ADDCHANGE THEN CALL READ_ADDCHANGE;
ELSE IF REQUEST=DELETE     THEN CALL READ_DELETE;
ELSE IF REQUEST=REROUTE    THEN CALL READ_REROUTE;
ELSE IF REQUEST=CHANGELOC THEN CALL READ_CHANGELOC;
ELSE
   DO; /* THIS "CANNOT" HAPPEN. */
      CALL MSG(12,'REQUEST FIELD IMPROPERLY SET WITHIN SUBROUTINE '||
                  'READ_REQUEST: '''||REQUEST||''''');
      CALL FLUSH_REQUEST;
      REQUEST=NEXT;
   END;

RETURN;

/* LOGICAL END OF PROCESSING FOR SUBROUTINE READ_REQUEST */

/* PROCEDURES INTERNAL TO READ_REQUEST FOLLOW: */
```

```
/*                        SUBROUTINE READ_ADDCHANGE                        */
/*                                                                         */
/*       THIS SUBROUTINE, CALLED BY READ_REQUEST, PROCESSES THE USER       */
/* INPUT FOR AN ADDCHANGE REQUEST.  WHEN READ_ADDCHANGE FINISHES, THE      */
/* USER'S INPUT SPECIFICATIONS ARE IN THE INDATA STRUCTURE, READY FOR      */
/* FURTHER PROCESSING.                                                     */

READ_ADDCHANGE: PROCEDURE;

IF REQUEST_LEVEL='C' THEN          /*** CIRCUIT ***/
   DO;
      CALL READ_SUBFIELDS(6);  /* UP TO 6 SUBFIELDS SEP BY "," */
      STRING(INDATA.RMCTCTRL)=KEY;
      INDATA.RMCTFLAG=SETFLAG(DFIELD(1));
      INDATA.RMCTRSTP=DFIELD(2);
      INDATA.RMCTRAPL=DFIELD(3);
      INDATA.RMCTIDEN=DFIELD(4);
      INDATA.RMCTXREF=DFIELD(5);
      INDATA.RMCTFILL=DFIELD(6);
      UPDATECIRCUIT = STRING(DFIELD)¬=' ';

      IF SEPARATOR=':' THEN
         CALL READ_ROUTING; /* TRUNK ROUTING PRESENT */
   END;

ELSE IF REQUEST_LEVEL='T' THEN    /*** TRUNK ***/
   DO;
      CALL READ_SUBFIELDS(6);
      STRING(INDATA.RMTRCTRL)=KEY;
      INDATA.RMTRFLAG=SETFLAG(DFIELD(1));
      INDATA.RMTRRCAT=DFIELD(2);
      INDATA.RMTRTCAP=DFIELD(3);
      INDATA.RMTRBAND=DFIELD(4);
      INDATA.RMTRAVAL=DFIELD(5);
      INDATA.RMTRFILL=DFIELD(6);
      UPDATETRUNK = STRING(DFIELD)¬=' ';

      IF SEPARATOR=':' THEN
         CALL READ_ROUTING; /* LINK ROUTING PRESENT */
   END;
```

```
ELSE IF REQUEST_LEVEL='L' THEN   /*** LINK ***/
   DO;
      CALL READ_SUBFIELDS(4);
      INDATA.RMLKCTRL=KEY;
      INDATA.RMLKFLAG=SETFLAG(DFIELD(1));
      INDATA.RMLKTRAN=DFIELD(2);
      INDATA.RMLKCODE=DFIELD(3);
      INDATA.RMLKFILL=DFIELD(4);
      UPDATELINK = STRING(DFIELD)¬=' ';

      IF SEPARATOR=':' THEN
         CALL READ_SITE_DATA;  /* SITE DATA PRESENT */
   END;

ELSE IF REQUEST_LEVEL='S' THEN   /*** SITE(S) ***/
   CALL READ_SITE_DATA;

ELSE
   DO; /* THIS "CANNOT" HAPPEN */
      CALL MSG(12,'REQUEST LEVEL IMPROPERLY SET WITHIN SUBROUTINE '||
                  'READ_ADDCHANGE: '''||REQUEST_LEVEL||'''');
      CALL FLUSH_REQUEST;
      REQUEST=NEXT;
      RETURN;
   END;

/* COMMON TEST FOR END OF REQUEST */
IF VERIFY(SEPARATOR,VALID_END_SEP)¬=0 THEN
   DO;
      CALL MSG(4,'EXTRANEOUS DATA BEGINNING '''||SEPARATOR||
                 ''' IGNORED.');
      CALL FLUSH_REQUEST;
   END;

RETURN;

END /* READ_ADDCHANGE */ ;
```

```
/*                          SUBROUTINE READ_DELETE                          */
/*                                                                          */
/*      THIS SUBROUTINE, CALLED BY READ_REQUEST, PROCESSES THE USER         */
/*  INPUT FOR A DELETE REQUEST.  WHEN READ_DELETE FINISHES, THE             */
/*  USER'S INPUT SPECIFICATIONS ARE IN THE INDATA STRUCTURE, READY          */
/*  FOR FURTHER PROCESSING.                                                 */

READ_DELETE: PROCEDURE;

/* THERE SHOULD BE ONLY CONTROL KEY(S) SPECIFIED IN DELETE REQUEST */

IF       REQUEST_LEVEL='C' THEN STRING(INDATA.RMCTCTRL)=KEY;
ELSE IF REQUEST_LEVEL='T' THEN STRING(INDATA.RMTRCTRL)=KEY;
ELSE IF REQUEST_LEVEL='L' THEN INDATA.RMLKCTRL=KEY;
ELSE IF REQUEST_LEVEL='S' THEN CALL READ_SITE_DATA;
ELSE
   DO; /* THIS "CANNOT" HAPPEN */
      CALL MSG(12,'REQUEST LEVEL IMPROPERLY SET WITHIN SUBROUTINE '||
                  'READ_DELETE: '''||REQUEST_LEVEL||'''');
      CALL FLUSH_REQUEST;
      REQUEST=NEXT;
      RETURN;
   END;


/* COMMON TEST FOR END OF REQUEST */
IF VERIFY(SEPARATOR,VALID_END_SEP)¬=0 THEN
   DO;
      CALL MSG(4,'EXTRANEOUS INPUT BEGINNING '''||SEPARATOR||
                 ''' IGNORED FOLLOWING CONTROL KEY IN DELETE REQUEST.');
      CALL FLUSH_REQUEST;
   END;

RETURN;

END /* READ_DELETE */ ;
```

```
/*                        SUBROUTINE READ_REROUTE                          */
/*                                                                         */
/*      THIS SUBROUTINE, CALLED BY READ_REQUEST, PROCESSES THE USER        */
/* INPUT FOR A REROUTE REQUEST.  WHEN READ_REROUTE FINISHES, THE           */
/* USER'S INPUT SPECIFICATIONS ARE IN THE INDATA STRUCTURE, READY          */
/* FOR FURTHER PROCESSING.                                                 */

READ_REROUTE: PROCEDURE;

IF REQUEST_LEVEL¬='T' THEN
   DO;
      CALL MSG(8,'INVALID REQUEST '''||REQUEST||REQUEST_LEVEL||
                 ''' NOT PROCESSED.');
      CALL FLUSH_REQUEST;
      REQUEST=NEXT;
      RETURN;
   END;

STRING(INDATA.RMTRCTRL)=KEY;

CALL READ_SUBFIELDS(6);

IF STRING(DFIELD)¬=' ' THEN
   CALL MSG(4,'SPECIFIED DATA SUBFIELDS IGNORED.  '||
              'USE @T TO CHANGE DATA FIELDS.');

IF SEPARATOR=':' THEN
   CALL READ_ROUTING;

IF VERIFY(SEPARATOR,VALID_END_SEP)¬=0 THEN
   DO;
      CALL MSG(4,'EXTRANEOUS INPUT BEGINNING '''||SEPARATOR||
                 ''' IGNORED.');
      CALL FLUSH_REQUEST;
   END;

RETURN;

END /* READ_REROUTE */ ;
```

```
/*                        SUBROUTINE READ_CHANGELOC                        */
/*                                                                         */
/*      THIS SUBROUTINE, CALLED BY READ_REQUEST, PROCESSES THE USER        */
/* INPUT FOR A CHANGELOC REQUEST.  WHEN READ_CHANGELOC FINISHES, THE       */
/* USER'S INPUT SPECIFICATIONS ARE IN THE INDATA STRUCTURE, READY          */
/* FOR FURTHER PROCESSING.                                                 */

READ_CHANGELOC: PROCEDURE;

IF       REQUEST_LEVEL='C' THEN STRING(INDATA.RMCTCTRL)=KEY;
ELSE IF REQUEST_LEVEL='T' THEN STRING(INDATA.RMTRCTRL)=KEY;
ELSE IF REQUEST_LEVEL='L' THEN INDATA.RMLKCTRL=KEY;
ELSE
   DO;
      CALL MSG(8,'INVALID REQUEST '''||REQUEST||REQUEST_LEVEL||
                 ''' NOT PROCESSED.');
      CALL FLUSH_REQUEST;
      REQUEST=NEXT;
      RETURN;
   END;

CALL READ_SUBFIELDS(4);

FRLC=DFIELD(1);
FRAC=DFIELD(2);
TOLC=DFIELD(3);
TRAC=DFIELD(4);

IF VERIFY(SEPARATOR,VALID_END_SEP)¬=0 THEN
   DO;
      CALL MSG(4,'EXTRANEOUS INPUT BEGINNING '''||SEPARATOR||
                 ''' IGNORED.');
      CALL FLUSH_REQUEST;
   END;

RETURN;

END /* READ_CHANGELOC */ ;
```

```
/*                        SUBROUTINE READ_SUBFIELDS                      */
/*                                                                       */
/*     THIS SUBROUTINE READ DATA SUBFIELDS SEPARATED BY COMMAS.          */
/* WHEN READ_SUBFIELDS IS CALLED WITH SEPARATOR=',', THE FIRST DATA      */
/* SUBFIELD IS ALREADY IN  ITEM .  THE PARAMETER FIELDCOUNT SPECIFIES    */
/* THE MAXIMUM NUMBER OF DATA SUBFIELDS TO BE READ BY THIS PROCEDURE.    */
/* READ_SUBFIELDS WILL READ FIELDCOUNT SUBFIELDS (UP TO 6).              */
/* IF MORE SUBFIELDS REMAIN IN THE INPUT STREAM (INDICATED BY            */
/* COMMAS), THIS ROUTINE WILL SKIP OVER THEM SO THAT SEPARATOR¬=','      */
/* WHEN READ_SUBFIELDS TERMINATES.                                       */

READ_SUBFIELDS:PROCEDURE(FIELDCOUNT);


DECLARE FIELDCOUNT BINARY FIXED(15,0); /* MAX # SUBFIELDS TO BE READ */

DECLARE I          BINARY FIXED(15,0);


/* CLEAR DATA FIELDS */
DFIELD=' ';

/* AS WE DO EVERYWHERE, WE CHECK UP ON OURSELVES HERE: */
IF FIELDCOUNT>6 THEN
   DO;
      CALL MSG(12,
               'SUBROUTINE READ_SUBFIELDS CALLED WITH FIELDCOUNT > 6:'
               ||SUBSTR(CHAR(FIELDCOUNT),4));
      CALL FLUSH_REQUEST;
      REQUEST=NEXT;
      RETURN;
   END;

/* NOW PROCESS ALL THE SUBFIELDS OF THE PRIMARY DATA FIELD.      */
/* THERE CAN BE AT MOST 6 DATA FIELDS FOLLOWING THE CONTROL KEY. */
DO I=1 TO FIELDCOUNT WHILE(SEPARATOR=',');
   DFIELD(I)=ITEM;
   CALL INPUT(SEPARATOR,ITEM);
END;

IF SEPARATOR=',' THEN  /* TOO MANY DATA SUBFIELDS */
   DO;
      CALL MSG(4,'TOO MANY DATA SUBFIELDS.  EXTRANEOUS ONES IGNORED.');
      DO WHILE(SEPARATOR=',');  /* SKIP EXTRAS */
         CALL INPUT(SEPARATOR,ITEM);
      END;
   END;

RETURN;

END /* READ_SUBFIELDS */ ;
```

```
/*                        SUBROUTINE READ_ROUTING                          */
/*                                                                         */
/*      THIS SUBROUTINE READS TRUNK OR LINK CONTROL KEYS FOLLOWING A       */
/* ":" AND SEPARATED BY "&".  THERE IS NO LOGICAL LIMIT TO THE NUMBER      */
/* OF KEYS;  THE LIMIT DEPENDS ON THE AMOUNT OF CORE AVAILABLE FOR         */
/* ALLOCATING ROUTE_CHAIN NODES.                                           */
/*                                                                         */
/*      INTERNAL SUBROUTINE NEXTKEY DOES THE ACTUAL WORK OF LISTING        */
/* THE ROUTE KEYS AND READING THE NEXT INPUT ITEM. READ_ROUTING            */
/* SIMPLY LOOPS UNITL THERE ARE NO MORE ROUTING KEYS TO BE PROCESSED.      */

READ_ROUTING: PROCEDURE;

DECLARE KEYOK BIT(1);

DO UNTIL(SEPARATOR¬='&' | ¬KEYOK);
   CALL NEXTKEY(KEYOK);
END;

IF VERIFY(SEPARATOR,VALID_END_SEP)¬=0 THEN
   DO;
      CALL MSG(4,'EXTRANEOUS DATA BEGINNING '''||SEPARATOR||
                 ''' IGNORED.');
      CALL FLUSH_REQUEST;
   END;

RETURN;
```

```
/*                          SUBROUTINE NEXTKEY                          */
/*                                                                      */
/*      THIS SUBROUTINE IS CALLED BY READ_ROUTING TO ADD A ROUTE_CHAIN*/
/* NODE TO THE END OF A LINKED LIST AND FILL THE RKEY FIELD WITH THE */
/* CURRENT ROUTING CONTROL KEY.                                         */
/*                                                                      */
/*      THE PARAMETER  OK  IS A RESULT PARAMETER.  IT IS SET TO '0'B  */
/* IF THE CURRENT CONTROL KEY IS BLANK; OTHERWISE, IT IS SET TO '1'B.*/
/*                                                                      */
/*      NEXTKEY CALLS INPUT TO READ THE NEXT DATA ITEM BEFORE          */
/* RETURNING TO READ_ROUTING.                                           */

NEXTKEY: PROCEDURE(OK);

DECLARE OK BIT(1);
DECLARE LASTR POINTER;

OK= ITEM¬='';
IF OK THEN /* ITEM NOT NULL => CONTROL KEY PRESENT */
   DO;
      IF ROUTESPEC¬=NULL THEN LASTR=RPTR;

      ALLOCATE ROUTE_CHAIN;
      RKEY=ITEM;
      RSITE1=' ';
      RSITE2=' ';
      NEXTR=NULL;

      IF ROUTESPEC¬=NULL THEN
         LASTR->NEXTR=RPTR;
      ELSE
         ROUTESPEC=RPTR;

   END;


ELSE /* (NOT OK) NO CONTROL KEY SPECIFIED */
   CALL MSG(8,'MISSING CONTROL KEY FOLLOWING &.');

CALL INPUT(SEPARATOR,ITEM);

RETURN;

END /* NEXTKEY */ ;

END /* READ_ROUTING */ ;
```

```
/*                    SUBROUTINE READ_SITE_DATA                          */
/*                                                                       */
/*       THIS SUBROUTINE IS CALLED BY READ_ADDCHANGE OR READ_DELETE      */
/* TO PROCESS THE USER'S SITE SPECIFICATIONS FOR @S, @L, OR -S           */
/* REQUESTS.  IT FILLS THE SITE(1:2) SUBSTRUCTURE OF INDATA.             */

READ_SITE_DATA: PROCEDURE;

DECLARE SITEK CHARACTER(11),
        I     BINARY FIXED(15,0);

IF REQUEST_LEVEL='L' THEN
   DO;
      SITEK=ITEM;
      CALL INPUT(SEPARATOR,ITEM);
   END;
ELSE IF REQUEST_LEVEL='S' THEN
   SITEK=KEY;
ELSE /* THIS "CANNOT" HAPPEN <BUT HANDLE IT ANYWAY> */
   SITEK='';

/* TWO SITES MAY BE SPECIFIED */
DO I=1 TO 2 WHILE(SITEK¬='');

   STRING(INDATA.RMSICTRL(I))=SITEK;
   SITEK='';
   CALL READ_SUBFIELDS(5);
   INDATA.RMSISTCT(I)=DFIELD(1);
   INDATA.RMSIFLAG(I)=SETFLAG(DFIELD(2));
   INDATA.RMSICODE(I)=DFIELD(3);
   INDATA.RMSINUME(I)=DFIELD(4);
   INDATA.RMSIFILL(I)=DFIELD(5);
   UPDATESITE(I)     =STRING(DFIELD)¬=' ';

   IF I=1 & SEPARATOR='&' THEN /* ANOTHER SITE PRESENT */
      DO;
         SITEK=ITEM;
         CALL INPUT(SEPARATOR,ITEM);
      END;

END;

IF VERIFY(SEPARATOR,VALID_END_SEP)¬=0 THEN
   DO;
      CALL MSG(4,'EXTRANEOUS INPUT BEGINNING '''||SEPARATOR||
                 ''' IGNORED.');
      CALL FLUSH_REQUEST;
   END;

RETURN;

END /* READ_SITE_DATA */ ;
```

AD-A067 217 COMPUTER SCIENCES CORP FALLS CHURCH VA F/G 17/2
INVESTIGATION OF THE VULNERABILITY/SURVIVABILITY OF SYSTEMS SUP--ETC(U)
JUN 78 H BLANK, G KINAL, M BASSMAN, D GAN DNA001-77-C-0015
UNCLASSIFIED DNA-4354F-1C-1 NL

3 OF 4
AD A067217

END
DATE FILMED
6-79
DDC

CONT

```
/*                          FUNCTION SETFLAG                                */
/*                                                                          */
/*       THIS FUNCTION IS CALLED TO PROCESS THE INPUT SPECIFICATIONS        */
/* FOR THE FLAG SUBFIELD OF CIRCUITS, TRUNKS, LINKS, OR SITES.              */
/* THE USER CAN SPECIFY THE FIELD AS 1) BLANK, 2) *, OR 3) A STRING         */
/* OF EIGHT 0'S OR 1'S.  SETFLAG PACKS THE EIGHT BITS INTO A SINGLE         */
/* ONE-BYTE FIELD AND STORES THE BLANK OR * AS THE EBCDIC REPRE-            */
/* SENTATION.  THIS WORKS BECAUSE THE EBCDIC CODES FOR ASTERISK             */
/* ('*' = 01011100) AND BLANK (' ' = 01000000) ARE NOT VALID                */
/* FLAG FIELD SETTINGS.                                                     */
/*                                                                          */
/*       THE PARAMETER FLAGCHARS IS THE EIGHT-BYTE INPUT FIELD,             */
/* POSSIBLY BLANK OR CONTAINING ONLY A LEADING *.  THE ONE-BYTE             */
/* EIGHT-BIT) RESULT IS RETURNED AS A FUNCTION VALUE.                       */

SETFLAG: PROCEDURE(FLAGCHARS) RETURNS(CHARACTER(1));

DECLARE FLAG      CHARACTER(1),  /* RETURNED VALUE */
        FLAGCHARS CHARACTER(8),  /* RECEIVED VALUE */
        FLAGBITS  BIT(8),        /* INTERMEDIATE VALUE */

        I         BINARY FIXED(15,0);

IF      FLAGCHARS='        '     THEN FLAG=' ';  /* '01000000'B */
ELSE IF FLAGCHARS='*       '     THEN FLAG='*';  /* '01011100'B */
ELSE IF VERIFY(FLAGCHARS,'01')=0 THEN
   DO;
      DO I=1 TO 8;
         SUBSTR(FLAGBITS,I,1) = SUBSTR(FLAGCHARS,I,1)='1';
      END;
      UNSPEC(FLAG)=FLAGBITS;
   END;
ELSE
   DO;
      FLAG=' ';
      CALL MSG(8,'INVALID FLAG FIELD SPECIFICATION '''||FLAGCHARS||
                 ''' IGNORED.  BLANK FIELD ASSUMED.');
   END;

RETURN(FLAG);

END /* SETFLAG */ ;


END /* READ_REQUEST */ ;
```

```
/*                    SUBROUTINE FLUSH                                  */
/*                                                                      */
/*       THIS PROCEDURE WILL READ AND LIST THE USER'S INPUT WITHOUT     */
/* PROCESSING IT.  TWO SECONDARY ENTRY POINTS ARE PROVIDED:             */
/*                                                                      */
/* FLUSH_ALL      -- FLUSHES AND PRINTS THE REMAINING INPUT UP TO THE   */
/*                   END OF FILE                                        */
/* FLUSH_REQUEST -- FLUSHES AND PRINTS THE INPUT UP TO THE BEGINNING    */
/*                   OF THE NEXT REQUEST                                */

FLUSH: PROCEDURE;

FLUSH_ALL: ENTRY;


DO WHILE(¬EOF);
   CALL INPUT(' ',ITEM);
END;
RETURN;

FLUSH_REQUEST: ENTRY;

DO UNTIL(VERIFY(SEPARATOR,VALID_END_SEP)=0);
   CALL INPUT(SEPARATOR,ITEM);
END;

RETURN;

END /* FLUSH */ ;
```

```
/*                    SUBROUTINE INPUT                                    */
/*                                                                        */
/*      THIS SUBROUTINE RETRIEVES THE USER'S INPUT ONE CHARACTER AT A     */
/* TIME BY INVOKING FUNCTION NEXTCHAR.  (NEXTCHAR, WHICH CALLS            */
/* GETCHAR, STRIPS OUT THE PARENTHETICAL COMMENTS, SO INPUT NEVER          */
/* SEES THEM.)                                                            */
/*                                                                        */
/*      INPUT HAS TWO PARAMETERS. P_SEP WILL BE SET TO THE SEPARATOR      */
/* ('@', '-', '#', '%', ',', ':', '&') PRECEDING THE DATA ITEM TO BE      */
/* READ; ON ENTRY, IT MAY BE                                              */
/*   '1' IF THIS IS THE FIRST CALL TO INPUT,                              */
/*   ' ' IF END-OF-FILE WAS PREVIOUSLY REACHED, OR INPUT IS BEING         */
/*       FLUSHED, OR                                                      */
/*   ':' IF END-OF-REQUEST WAS REACHED AND MORE DATA NOT YET NEEDED.      */
/*                                                                        */
/*      THE SECOND PARAMETER WILL BE THE DATA ITEM READ.                  */
/*                                                                        */
/*      LEADING BLANKS WILL BE STRIPPED FROM THE INPUT DATA ITEM.         */
/* TRAILING BLANKS ARE NOT STRIPPED.                                      */
/*                                                                        */
/*      SINCE THE SEPARATOR THAT FOLLOWS THE DATA ITEM IS NOT             */
/* RETURNED TO THE CALLING PROCEDURE UNTIL THE NEXT TIME THAT INPUT       */
/* IS INVOKED, THAT CHARACTER IS SAVED IN THE GLOBAL LOCATION             */
/* NEXT_SEPARATOR SO IT WILL BE AVAILABLE TO THE NEXT INVOCATION.         */
/*                                                                        */
/*      WHEN INPUT DETECTS THE END OF THE CURRENT REQUEST, IT CALLS       */
/* PRINT_RTQ TO UPDATE THE LISTING.                                       */

INPUT: PROCEDURE(P_SEP,DATA);

DECLARE /* PARAMETERS */
        P_SEP CHARACTER(1),              /* PRESENT SEPARATOR */
        DATA  CHARACTER(100) VARYING;    /* PRESENT DATA ITEM */

DECLARE CH    CHARACTER(1);              /* SINGLE CHARACTER BEING    */
                                         /*   "READ" BY NEXTCHAR      */

ON ENDFILE(SYSIN) GO TO NOMORE;

/* DATA ITEM MUST BE INITIALIZED TO A NULL STRING */
DATA='';
```

```
/* IF THIS IS THE FIRST CALL TO INPUT, NEXT_SEPARATOR CONTAINS A '1' */
/* RATHER THAN A TRUE SEPARATOR.  FIND THE FIRST NON-BLANK CHARACTER */
/* AND LET THAT BE THE SEPARATOR.  IGNORING COMMENTS (WHICH ARE      */
/* STRIPPED OUT BY NEXTCHAR), IF THE FIRST CHARACTER ISN'T A VALID   */
/* REQUEST CHARACTER, THERE IS AN ERROR.  FLAG IT AND TRY TO         */
/* CONTINUE.                                                         */
IF NEXT_SEPARATOR='1' THEN
   DO;

      /* INITIALIZE P_SEP */
      P_SEP=' ';

      DO WHILE(NEXTCHAR(CH)=' '); /* SKIP LEADING BLANKS */
      END;

      IF VERIFY(CH,VALID_REQ_SEP)¬=0 THEN
         DO;
            CALL MSG(8,'INPUT MUST START WITH VALID REQUEST.  '||
                       'INPUT BEGINNING '''||CH||''' IGNORED.');
            DO WHILE(VERIFY(NEXTCHAR(CH),VALID_REQ_SEP)¬=0);
            END;
         END;

      /* CURRENT CHARACTER IS VALID */
      NEXT_SEPARATOR=CH;

   END;

/* END OF SPECIAL PROCESSING FOR FIRST CALL TO INPUT */
```

```
/* IF THE PRESENT SEPARATOR IS NOT A BLANK (SET BY THE MAIN */
/* PROCESSING LOOP AS A GO-AHEAD SIGNAL) AND THE NEXT       */
/* SEPARATOR IS A VALID END-OF-REQUEST CHARACTER, THEN WE   */
/* HAVE JUST REACHED THE END OF A REQUEST.                  */
/*     DON'T READ ANYMORE INPUT YET, BECAUSE IF WE          */
/* DO, WE'LL FOUL UP THE LISTING.  SET THE PRESENT SEPARATOR*/
/* TO ";" (WHICH IS NOT A VALID DELIMITER AS FAR AS THE USER*/
/* IS CONCERNED) TO SIGNAL THE MAIN PROCESSING LOOP THAT    */
/* SUBROUTINE INPUT MUST BE CALLED AN EXTRA TIME BEFORE     */
/* READING THE NEXT REQUEST.   KEEP ON DOING THIS UNTIL THE */
/* STOP SIGNAL (P_SEP=';') IS TURNED OFF IN THE MAIN LOOP   */
/* AND THE GO-AHEAD SIGNAL (P_SEP=' ') IS TURNED ON.        */

IF P_SEP=';' | (P_SEP¬=' '&VERIFY(NEXT_SEPARATOR,VALID_END_SEP)=0) THEN
   DO;
      P_SEP=';';
      RETURN;
   END;

/* IF PRECEDING CALL TO INPUT RAISED THE ENDFILE CONDITION, THERE */
/* IS NOTHING MORE TO BE READ.  THIS SITUATION IS INDICATED BY A  */
/* BLANK NEXT_SEPARATOR.  INDICATE THIS TO THE CALLING PROGRAM BY */
/* SETTING THE EOF FLAG AND RETURNING A BLANK SEPARATOR.          */

P_SEP=NEXT_SEPARATOR;
IF P_SEP=' ' THEN
   DO;
      EOF='1'B;
      RETURN;
   END;

DO WHILE(NEXTCHAR(CH)=' '); /* SKIP LEADING BLANKS */
END;

/* ACCUMULATE DATA UNTIL END-OF-FIELD SEPARATOR OR END-OF-FILE */
/* DATA IS ALREADY SET TO A NULL STRING */
DO WHILE(VERIFY(CH,'@-#%,:&')¬=0);
   DATA=DATA||CH;
   CH=NEXTCHAR(CH);
END;

/* WE HAVE A VALID SEPARATOR.  SAVE IT FOR NEXT TIME. */
NEXT_SEPARATOR=CH;

/*IF ¬ASIS THEN DATA=UP(DATA);*/

/* PRINT THE CURRENT REQUEST IF WE ARE AT THE END. */
IF VERIFY(NEXT_SEPARATOR,VALID_END_SEP)=0 THEN CALL PRINT_REQ;

RETURN;
```

```
NOMORE: /* ENDFILE(SYSIN)                                              */
        /*      CONTROL CAN REACH THIS POINT IN TWO WAYS:              */


        /* (1) THERE WERE NO VALID REQUESTS IN THE INPUT STREAM, IN   */
        /*     WHICH CASE THIS IS THE FIRST CALL TO INPUT AND         */
        /*     NEXT_SEPARATOR IS STILL EQUAL TO '1'.  SET EOF AND     */
        /*     P_SEP AND THEN RETURN.                                 */

IF NEXT_SEPARATOR='1' THEN
   DO;
      EOF='1'B;
      P_SEP=' ';
      CALL PRINT_REQ;
      RETURN;
   END;


        /* (2) AT LEAST ONE VALID REQUEST CHARACTER HAS BEEN READ     */
        /*     (PROBABLY MORE) AND END-OF-FILE WAS REACHED WHILE      */
        /*     READING A DATA ITEM, WHICH MAY OR MAY NOT BE BLANK.    */
        /*     SET NEXT_SEPARATOR TO BLANK SO END-OF-FILE WILL BE     */
        /*     RECOGNIZED AND SET IMMEDIATELY NEXT TIME INPUT IS      */
        /*     CALLED, AND THEN RETURN FOR PROCESSING.                */

NEXT_SEPARATOR=' ';

/*IF ¬ASIS THEN DATA=UP(DATA);*/

CALL PRINT_REQ;
RETURN;
```

```
/*                  FUNCTION NEXTCHAR                                    */
/*                                                                       */
/*       NEXTCHAR IS THE INTERFACE BETWEEN SUBROUTINE INPUT AND          */
/* FUNCTION GETCHAR.  THE ONLY PURPOSE OF NEXTCHAR IS TO "EAT" COM-      */
/* MENTS.  IF COMMENTS WERE NOT PERMITTED, THIS INTERFACE COULD BE       */
/* ELIMINATED AND GETCHAR COULD BE RENAMED "NEXTCHAR."  (IN FACT,        */
/* BEFORE COMMENTS WERE PERMITTED, IT WAS.)                              */
/*                                                                       */
/*       AT PRESENT, NO WARNING IS GIVEN IF THE ENDFILE CONDITION IS     */
/* RAISED WHILE EATING A COMMENT.                                        */

NEXTCHAR: PROCEDURE(THISCHAR) RETURNS(CHARACTER(1));

DECLARE THISCHAR CHARACTER(1);

THISCHAR=GETCHAR(THISCHAR);
DO WHILE(THISCHAR='(');
   DO WHILE(GETCHAR(THISCHAR)¬=')');
   END;
   THISCHAR=GETCHAR(THISCHAR);
END;

RETURN(THISCHAR);
```

```
/*                        FUNCTION GETCHAR                                  */
/*                                                                          */
/*      THIS FUNCTION DOES THE ACTUAL WORK OF BUFFERING THE INPUT AND       */
/* EXTRACTING ONE CHARACTER AT A TIME.  GETCHAR GETS THE NEXT               */
/* CHARACTER FROM FILE SYSIN AND RETURNS IT BLINDLY (WITHOUT                */
/* EXAMINATION) TO THE POINT OF INVOCATION AS BOTH A PARAMETER AND          */
/* FUNCTION VALUE.                                                          */
/*                                                                          */
/*      MULTIPLE BUFFERING IS USED SO THAT A COMPLETE SINGLE REQUEST,       */
/* SPANNING MULTIPLE RECORDS, CAN BE PRINTED ON THE LISTING AT              */
/* ONE TIME BY PRINT_REQ.                                                   */
/*                                                                          */
/*      THE GLOBAL VARIABLES INREC, INREC_LEVEL, INRECLEN, BUFPOS,          */
/* BUFBEGIN, AND BUFEND ARE ALL ESTABLISHED IN THE MAIN PROCEDURE.          */
/*                                                                          */
/*      IF GETCHAR FILLS UP THE INPUT ARRAY INREC (A SINGLE                 */
/* REQUEST IS MORE THAN DIM(INREC,1) RECORDS LONG), IT CALLS                */
/* PRINT_INREC TO UPDATE THE LISTING AND PROVIDE MORE INPUT SPACE.          */


GETCHAR: PROCEDURE(THISCHAR) RETURNS(CHARACTER(1));

DECLARE THISCHAR CHARACTER(1);

/* ENDFILE(SYSIN) IS HANDLED IN SUBROUTINE INPUT */

IF BUFPOS>=BUFEND THEN
   DO;
      IF INREC_LEVEL>=DIM(INREC,1) THEN CALL PRINT_INREC;
      INREC_LEVEL=INREC_LEVEL+1;
      GET FILE(SYSIN) EDIT(INREC(INREC_LEVEL))
                          (A(INRECLEN));
      BUFPOS=BUFBEGIN-1;
   END;

BUFPOS=BUFPOS+1;
THISCHAR=SUBSTR(INREC(INREC_LEVEL),BUFPOS,1);
RETURN(THISCHAR);

END /* GETCHAR */ ;

END /* NEXTCHAR */ ;
```

```
/*                      SUBROUTINE PRINT_INPUT                          */
/*                                                                      */
/*     THIS SUBROUTINE HAS TWO ENTRY POINTS AND IS USED TO PRINT        */
/* THE USER'S INPUT ON THE OUTPUT LISTING.  IT IS CALLED BY INPUT       */
/* OR GETCHAR.                                                          */
/*                                                                      */
/* PRINT_INREC -- EMPTIES THE ENTIRE INREC ARRAY.                       */
/* PRINT_REQ   -- PRINTS ONLY THE CURRENT REQUEST.                      */

PRINT_INPUT: PROCEDURE;

DECLARE I BINARY FIXED(15,0);

PRINT_INREC: ENTRY;

DO I=1 TO INREC_LEVEL;
   PUT FILE(SYSPRINT) EDIT(INREC#,INREC(I))
                          (SKIP,COL(COL#),F(4),COL(COLREC),A(INRECLEN));
   INREC#=INREC#+1;
END;
INREC_LEVEL=0;
RETURN;


PRINT_REQ: ENTRY;

PUT FILE(SYSPRINT) SKIP(1);
DO I=1 TO INREC_LEVEL-1;
   PUT FILE(SYSPRINT) EDIT(INREC#,INREC(I))
                          (SKIP,COL(COL#),F(4),COL(COLREC),A(INRECLEN));
   INREC#=INREC#+1;
END;

INREC(1)=INREC(INREC_LEVEL);
INREC_LEVEL=1;

RETURN;


END /* PRINT_INPUT */ ;

END /* INPUT */ ;
```

```
/*                          SUBROUTINE: MSG                                 */
/*                                                                          */
/*       THIS SUBROUTINE PRINTS ALL MESSAGES AND KEEPS TRACK OF THE         */
/* HIGHEST SEVERITY CODE.  THIS CODE (SEE TABLE UNDER THE DECLARATION       */
/* OF MSGCLASS) WILL ULTIMATELY BE ISSUED AS THE JOB STEP RETURN CODE       */
/* AND CAN BE TESTED BY LATER STEPS.                                        */
/*       THE FIRST PARAMETER IS THE SEVERITY CODE TO BE ASSOCIATED          */
/* WITH THE CURRENT MESSAGE; IT SHOULD BE 0, 4, 8, 12, OR 16.  THE          */
/* SECOND PARAMETER IS THE TEXT OF THE MESSAGE.  AN ELEMENT OF              */
/* MSGCLASS WILL BE CONCATENATED TO THE FRONT OF THE TEXT ACCORDING         */
/* TO THE VALUE OF RET_CODE.                                                */
/*       THE STARTING COLUMN OF THE MESSAGE IS DETERMINED BY THE VALUE      */
/* OF THE GLOBAL VARIABLE MSGC.  IF THE RET_CODE IS 16 (TERMINAL            */
/* ERROR), THEN THE SUBROUTINE INITIATES IMMEDIATE TERMINATION BY           */
/* ESCAPING TO THE GLOBAL LABEL CLEANUP INSTEAD OF SIMPLY RETURNING         */
/* TO THE CALLING POINT.                                                    */

MSG: PROCEDURE(RET_CODE,TEXT);

DECLARE RET_CODE BINARY FIXED(31,0),
        TEXT     CHARACTER(111) VARYING;

DECLARE MSGCLASS(0:4) CHARACTER(21) VARYING
                      INITIAL(
                              /* CLASS                  CODE */

                        '',                            /* 00 */
                        '**** WARNING: ',              /* 04 */
                        '**** ERROR: ',                /* 08 */
                        '**** DEBUG ERROR: ',          /* 12 */
                        '**** TERMINAL ERROR: ');      /* 16 */

/* MAKE SURE RET_CODE IS 0, 4, 8, 12, OR 16. */
RET_CODE=4*(DIVIDE(RET_CODE+3,4,5,0));
IF RET_CODE>16 THEN RET_CODE=16;

PUT FILE(SYSPRINT) EDIT(MSGCLASS(DIVIDE(RET_CODE,4,5,0))||TEXT)
                       (SKIP,COL(MSGC),A);
IF RET_CODE>0 THEN MSG_COUNT=MSG_COUNT+1;
IF RC>12 THEN RETURN; /* IF ALREADY DONE, DON'T FLUSH AGAIN. */
RC=MAX(RET_CODE,RC);
IF RET_CODE>12 THEN
   DO;
      PUT FILE(SYSPRINT) EDIT('INPUT STREAM FLUSHED:')
                             (SKIP,COL(MSGC),A(21));
      CALL FLUSH_ALL;
      GO TO TERMERR;
   END;

RETURN;

END /* MSG */ ;
```

```
/*                    SUBROUTINE GARBAGE_COLLECT                          */
/*                                                                        */
/*      THIS SUBROUTINE PERFORMS GARBAGE COLLECTION ON THE LINK AND       */
/* SITE MASTER FILES; THAT IS, LINKS AND SITES THAT ARE USED BY NO         */
/* TRUNKS AND/OR CIRCUITS ARE DELETED FROM THE DATA BASE.                 */
/*                                                                        */
/*      GARBAGE_COLLECT READS EACH FILE IN TURN SERIALLY AND MAKES A      */
/* LIST OF ALL CONTROL KEYS IN THE GARBAGE COLLECTION WORK FILE           */
/* GCWORK.  IT THEN RE-READS THE LIST OF KEYS AND TRIES TO DELETE         */
/* EACH RECORD.  ONLY THOSE TO WHICH NO VARIABLE RECORDS ARE LINKED       */
/* WILL ACTUALLY BE DELETED.                                              */

GARBAGE_COLLECT: PROCEDURE;

DECLARE RECORDS  CHARACTER(5),          /* 'LINKS' OR 'SITES' */
        KEY      CHARACTER(11)VARYING, /* CONTROL KEY */
        KEYLEN   BINARY FIXED(15,0),   /* 5 FOR LINKS, 11 FOR SITES */
        #_DELS   BINARY FIXED(15,0),
        SOMELEFT BIT(1),               /* 1 => MORE KEYS TO PROCESS */
        GCWORK   FILE STREAM ENVIRONMENT
                             (VB,RECSIZE(1004),BLKSIZE(13000));


/* WORK FILE MAY BE MISSING */
ON UNDEFINEDFILE(GCWORK)
   BEGIN;
      CALL MSG(8,'INCOMPLETE OR NO DD STATEMENT FOR FILE GCWORK.');
      CALL MSG(0,'SPECIFY: //GCWORK DD UNIT=SYSDA,SPACE=(TRK,(1,1))');
      CALL MSG(4,'GARBAGE COLLECTION NOT PERFORMED.');
      GO TO RET;
   END;


ON ENDFILE(GCWORK) SOMELEFT='0'B;

DO RECORDS='LINKS','SITES';
   IF RECORDS='LINKS' THEN
      DO;
         FILEID='RMLK';
         ELMLIST='RMLKCTRLEND.';
         KEYLEN=5;
      END;
   ELSE IF RECORDS='SITES' THEN
      DO;
         FILEID='RMSI';
         ELMLIST='RMSICTRLEND.';
         KEYLEN=11;
      END;
   ELSE LEAVE;  /* IMPOSSIBLE CASE BECAUSE OF ITERATIVE DO */
```

```
   CALL MSG(0,'GARBAGE COLLECTION ON UNUSED '||RECORDS||':');

   /* READ THE LINK OR SITE MASTER FILE SERIALLY AND MAKE A LIST */
   /* OF ALL RECORDS IN THE FILE.                                */

   OPEN FILE(GCWORK) OUTPUT;
   FUNCT=RESTM;
   CALL DBIO(3,2);
   IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);
   FUNCT=SEQRM;
   CALL DBIO(6,2);
   DO WHILE(STATUS¬=ENDP);
      IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);
      PUT FILE(GCWORK) EDIT(SUBSTR(IOAREA,1,KEYLEN))
        (A(KEYLEN));
      CALL DBIO(6,2);
   END;
   STATUS=OK;
   CLOSE FILE(GCWORK);

   /* NOW READ THE WORK FILE AND TRY TO DELETE EACH RECORD IN TURN. */
   /* ONLY THOSE WITH NO LINKAGE TO VARIABLE RECORDS WILL BE       */
   /* DELETED BY TOTAL.                                            */

   OPEN FILE(GCWORK) INPUT;

   FUNCT=DEL_M;
   #_DELS=0;
   SOMELEFT='1'B;
   GET FILE(GCWORK) EDIT(KEY)(A(KEYLEN)); /* FIRST CONTROL KEY */
   DO WHILE(SOMELEFT);
      CONTROL=KEY;
      CALL DBIO(7,2);
      IF STATUS¬='IMDL' THEN
         DO;
            IF STATUS¬=OK THEN SIGNAL CONDITION(TOTERR);
            CALL MSG(0,'  '||KEY||' DELETED');
            #_DELS=#_DELS+1;
         END;
      ELSE STATUS=OK;
      GET FILE(GCWORK) EDIT(KEY)(A(KEYLEN));
   END;

   CALL MSG(0,CHAR(#_DELS)||' UNUSED '||RECORDS||' DELETED.');
   CLOSE FILE(GCWORK);

END;

RET: RETURN;

END /* GARBAGE_COLLECT */ ;


END /* DBUTIL */ ;
```

DISTRIBUTION LIST

DEPARTMENT OF DEFENSE

U.S. Documents Officer
AFSOUTH
ATTN: U.S. Documents Officer

Armed Forces Staff College
ATTN: Reference & Technical Services Branch

Assistant Secretary of Defense
Comm. Cmd. Cont. & Intell.
ATTN: Assistant for NATO C3
ATTN: Stra. & Theater C&C System
ATTN: Survl. & Warning System

Assistant Secretary of Defense
Program Analysis & Evaluation
ATTN: General Purpose Programs

Assistant to the Secretary of Defense
Atomic Energy
ATTN: B. Adams
ATTN: Executive Assistant

Command & Control Technical Center
Department of Defense
ATTN: C-330
ATTN: C-610
ATTN: C-650

Commander in Chief
U.S. European Command
ATTN: J-6
ATTN: J-5NPG
ATTN: J-417-LW
ATTN: J-2
ATTN: J-3
ATTN: ECJ6-PF
ATTN: J-2-FTD

Commander In Chief, Pacific
ATTN: J-6
ATTN: J-3
ATTN: J-2
ATTN: J-54

Defense Advanced Rsch. Proj. Agency
ATTN: TIO

Defense Communications Engineer Center
ATTN: Code R720
ATTN: Code R400
ATTN: Code 123

Defense Communications Agency
ATTN: Code 205
ATTN: Code 410
ATTN: Code 510
ATTN: Code 520
ATTN: Code 530
ATTN: Code 101B

Defense Documentation Center
12 cy ATTN: DD

DEPARTMENT OF DEFENSE (Continued)

Defense Intelligence Agency
ATTN: DT-4B
ATTN: RDS-3A
6 cy ATTN: DT-1B

Defense Nuclear Agency
ATTN: STSP
ATTN: STVL
ATTN: STRA
ATTN: OAPO
ATNN: DDST
ATTN: VLIS
ATTN: STNA
4 cy ATTN: TITL

Director Net Assessment
Office of the Secretary of Defense
ATTN: Military Assistants

Field Command
Defense Nuclear Agency
ATTN: FCPR
ATTN: FCPRK

Field Command
Defense Nuclear Agency
Livermore Division
ATTN:: FCPRL

Interservice Nuclear Weapons School
ATTN: TTV

Joint Chiefs of Staff
ATTN: J-3
ATTN: J-3, NMCC
ATTN: J-5, Nuclear Division
ATTN: J-3, WWMCCS Council Support Office
ATTN: SAGA
ATTN: J-3, WWMCCS Evaluation Office
ATTN: J-3, Nuclear Weapons Branch
ATTN: J-3, Nuclear Contingency Branch

Joint Strat. Tgt. Planning Staff
ATTN: JPST
ATTN: JLTW-2
ATTN: NRI-STINFO Library
ATTN: JSAS
2 cy ATTN: JL

Joint Tactical Communications Office
ATTN: TT-E-SS

National Communications System
Office of the Manager
ATTN: NCS-TS

National Security Agency
ATTN: R-52, O. Van Gunten
ATTN: TDL

U.S. Forces Korea
ATTN: DJ-AM-SM
ATTN: CJ-P-G

DEPARTMENT OF DEFENSE (Continued)

U.S. National Military Representative
SHAPE
ATTN: U.S. Documents Officer

Under Secretary of Def. for Rsch. & Engrg.
ATTN: Tactical Warfare Programs
ATTN: Strategic & Space Systems (OS)
ATTN: Research & Adv. Technology

WWMCCS System Engineering Org.
ATTN: WWMCCS/SEE

DEPARTMENT OF THE ARMY

Asst. Chief of Staff for Intelligence
Department of the Army
ATTN: DAMI-FI

Atmospheric Sciences Laboratory
U.S. Army Research & Development Command
ATTN: DELAS-AS

Deputy Chief of Staff for Ops. & Plans
Department of the Army
ATTN: DAMO-SSP
ATTN: DAMO-RQS
ATTN: Technical Advisor
ATTN: DAMO-TCW

Deputy Chief of Staff for Rsch. Dev. & Acq.
Department of the Army
ATTN: DAMA-CSS-N
ATTN: Advisor for RDA Analysis

Harry Diamond Laboratories
Department of the Army
ATTN: DELHD-N-NP
ATTN: DELHD-N-RBH
ATTN: DELHD-N-RBA
ATTN: DELHD-N-CO
ATTN: DELHD-N-TI
2 cy ATTN: DELHD-N-EM

Multi-Service Communications Systems
Department of the Army
ATTN: DRCPM-MSCS-APB, M. Francis

U.S. Army Ballistic Research Labs.
ATTN: VL
ATTN: TBL
ATTN: CAL

U.S. Army C-E Engrg. Installation Agency
ATTN: CC-EMED-PED
ATTN: CCC-SEO
ATTN: CCC-PRSO

U.S. Army Comb. Arms Combat Dev. Acty.
ATTN: ATCA-CFT
ATTN: ATCA-CA
ATTN: ATCA-CO

U.S. Army Comd. & General Staff College
ATTN: ATSW-TA-D

U.S. Army Communications Command
ATTN: ACC-OPS-WR
ATTN: ACC-OPS-SM

DEPARTMENT OF THE ARMY (Continued)

U.S. Army Communications System Agency
ATTN: CCM-AD-LB (Library)

U.S. Army Concepts Analysis Agency
ATTN: Code 605/606

U.S. Army Electronics Rsch. & Dev. Command
ATTN: DRCPM-TDS-SD
ATTN: DRCPM-ATC

Commander-In-Chief
U.S. Army Europe and Seventh Army
ATTN: AEAGB
ATTN: AEAGD-MM
ATTN: ODCSE-E AEAGE
ATTN: AEAGC-O
ATTN: AEAGC-DSW

U.S. Army Materiel Sys. Analysis Activity
ATTN: DRXSY-DS
ATTN: DRXSY-S

U.S. Army Missile R&D Command
ATTN: DRSMI-YDR

U.S. Army Nuclear & Chemical Agency
ATTN: ATCN-W
ATTN: Library

U.S. Army Satellite Comm. Agency
ATTN: TACSAT Office
ATTN: DRCPM-SC-11

U.S. Army TRADOC Systems Analysis Activity
ATTN: ATAA-TAC

U.S. Army Training and Doctrine Comd.
ATTN: ATORI-OP-SD
ATTN: Technical Library

U.S. Army War College
ATTN: Library

Commander
V Corps
Department of the Army
ATTN: AETVCE
ATTN: AETVGC
ATTN: AETVGB

Commander
VII Corps
Department of the Army
ATTN: AETSCE
ATTN: AETSGB-O
ATTN: AETSGC

DEPARTMENT OF THE NAVY

Naval Ocean Systems Center
ATTN: Research Library

Naval Surface Weapons Center
White Oak Laboratory
ATTN: Code F31

U.S. Naval Forces, Europe
ATTN: N3262 Nuclear Surety Officer

DEPARTMENT OF THE NAVY (Continued)

Office of the Chief of Naval Operations
ATTN: OP 96
ATTN: OP 94
ATTN: OP 604 E/F
ATTN: OP 941

U.S. Atlantic Fleet
Department of the Navy
ATTN: J611A
ATTN: J54

DEPARTMENT OF THE AIR FORCE

Aerospace Defense Command
ATTN: INW
ATTN: Commander
ATTN: ADCOM/INA

Air Force Weapons Laboratory, AFSC
ATTN: DYC
ATTN: SUL
ATTN: EL

Assistant Chief of Staff
Intelligence
Department of the Air Force
ATTN: INAK

Deputy Chief of Staff
Operations Plans and Readiness
Department of the Air Force
ATTN: AFXOXFM
ATTN: AFXOK

Deputy Chief of Staff
Research, Development, & Acq.
Department of the Air Force
ATTN: AFRDQSM

Deputy Chief of Staff
Programs & Analyses
Department of the Air Force
ATTN: PACGC
ATTN: PACSC

Electronic Systems Division, AFSC
ATTN: Technical Library

Foreign Technology Division, AFSC
ATTN: NIIS Library

Space & Missile Systems Organization
Air Force Systems Command
ATTN: SKA

Strategic Air Command
Department of the Air Force
ATTN: XPFS

Tactical Air Command
Department of the Air Force
ATTN: DRA

Commander In Chief
U.S. Air Forces in Europe
ATTN: XPXX

OTHER GOVERNMENT AGENCY

Central Intelligence Agency
ATTN: OSR/SF
ATTN: OSR/SEC
ATTN: OSI/LSD
ATTN: RD/SI

DEPARTMENT OF DEFENSE CONTRACTORS

BDM Corp.
ATTN: L. Jacobs
ATTN: W. Sweeney
ATTN: Corporate Library

Boeing Co.
ATTN: R. Scheppe

Commander 66th MI Group
ATTN: RDA

Computer Sciences Corp.
ATTN: H. Blank
ATTN: Library
ATTN: D. Gan
ATTN: M. Bassman
ATTN: G. Kinal

ESL, Inc.
ATTN: J. Marshall
ATTN: Library

General Electric Co.-TEMPO
Center for Advanced Studies
ATTN: DASIAC

General Electric Co.-TEMPO
Alexandria Office
ATTN: DASIAC

Institute for Defense Analyses
ATTN: D. Signori
ATTN: Classified Library

Kaman Sciences Corp.
ATTN: W. Long

Mission Research Corporation
ATTN: S. Gutsche

R & D Associates
ATTN: Technical Information Center
ATTN: R. Schaefer
ATTN: R. Poll

R & D Associates
ATTN: R. Latter

RCA Corp.
ATTN: E. Van Keuren

SRI International
ATTN: C. Shoens
ATTN: W. Jaye

TRW Defense & Space Sys. Group
ATTN: R. Webb

TRW Defense & Space Sys. Group
ATTN: J. Dyche

AD-A067 217 COMPUTER SCIENCES CORP FALLS CHURCH VA F/G 17/2
INVESTIGATION OF THE VULNERABILITY/SURVIVABILITY OF SYSTEMS SUP--ETC(U)
JUN 78 H BLANK, G KINAL, M BASSMAN, D GAN DNA001-77-C-0015
UNCLASSIFIED DNA-4354F-1C-1 NL

END
DATE FILMED
9-79
DDC

4

217

SUPPLEMEN

INFORMAT

ERRATA SHEET

for

DNA 4354F-1C-1 dated June 1978

Attached are a cover and a DD Form 1473 to replace the existing ones. The cover has been printed on adhesive-backed paper; please remove the back and place it over the existing cover. The DD Form 1473 has been printed on DAVAC; moisten the back and place it over the existing DD Form 1473.

DNA 4354F-1C-1

INTEGRATED NUCLEAR COMMUNICATIONS ASSESSMENT (INCA)

Transatlantic Trunk Utilization

Appendix A—Computer Program Documentation

Computer Sciences Corporation
6565 Arlington Boulevard
Falls Church, Virginia 22046

June 1978

Final Report for Period February 1977—October 1977

CONTRACT No. DNA 001-77-C-0115

THIS WORK SPONSORED BY THE DEFENSE NUCLEAR AGENCY
UNDER RDT&E RMSS CODE B363077464 099QAXCA10601 H2590D.

Prepared for
Director
DEFENSE NUCLEAR AGENCY
Washington, D. C. 20305

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1 REPORT NUMBER: DNA 4354F-1C-1	2 GOVT ACCESSION NO.
3 RECIPIENT'S CATALOG NUMBER	
4 TITLE (and Subtitle): INTEGRATED NUCLEAR COMMUNICATIONS ASSESSMENT (INCA) Transatlantic Trunk Utilization Appendix A—Computer Program Documentation	5 TYPE OF REPORT & PERIOD COVERED: Final Report for Period February 1977—October 1977
	6 PERFORMING ORG. REPORT NUMBER
7 AUTHOR(s): H. Blank, G. Kinal, M. Bassman, D. Gan	8 CONTRACT OR GRANT NUMBER(s): DNA 001-77-C-0115
9 PERFORMING ORGANIZATION NAME AND ADDRESS: Computer Sciences Corporation, 6565 Arlington Boulevard, Falls Church, Virginia 22046	10 PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Subtask 099QAXCA106-01
11 CONTROLLING OFFICE NAME AND ADDRESS: Director, Defense Nuclear Agency, Washington, D.C. 20305	12 REPORT DATE: June 1978
	13 NUMBER OF PAGES: 208
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

This work sponsored by the Defense Nuclear Agency under RDT&E RMSS Code B363077464 099QAXCA10601 H2590D.

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

Data Base	TOTAL®	Input
Primitive Matrix	Files	Output
Utility Program	Master Files	Parameter
Flow Chart	Variable Files	

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

This report contains the Computer Program Documentation for the Integrated Nuclear Communications Assessment (INCA) Transatlantic Trunk Utilization Study. It explains how the data base was created and defines each of the master files and variable files created. The individual programs utilized are explained in detail.

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)