

ARIZONA DEPARTMENT OF TRANSPORTATION

REPORT NUMBER: FHWA-AZ88-207

# **SIMPLIFIED BRIDGE LOAD RATING METHODOLOGY USING THE NATIONAL BRIDGE INVENTORY FILE**

**Volume II: Program Listing**

**Prepared by:**

Roy A. Imbsen  
Robert A. Schomber  
Imbsen & Associates, Inc.  
3213 Ramos Circle  
Sacramento, California 95827

**August 1987**

**Prepared for:**

Arizona Department of Transportation  
206 South 17th Avenue  
Phoenix, Arizona 85007  
In cooperation with  
U.S. Department of Transportation  
Federal Highway Administration

The contents of this report reflect the views of the authors who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the Arizona Department of Transportation or the Federal Highways Administration. This report does not constitute a standard, specification, or regulation. Trade or manufacturer's names which may appear herein are cited only because they are considered essential to the objectives of the report. The U.S. Government and the State of Arizona do not endorse products or manufacturers.

TECHNICAL REPORT DOCUMENTATION PAGE

1. REPORT NO. FHWA-AZ87-207, II		2. GOVERNMENT ACCESSION NO.		3. RECIPIENT'S CATALOG NO.	
4. TITLE AND SUBTITLE SIMPLIFIED BRIDGE LOAD RATING METHODOLOGY USING THE NATIONAL BRIDGE INVENTORY FILE VOLUME II: PROGRAM LISTING				5. REPORT DATE AUGUST, 1987	
				6. PERFORMING ORGANIZATION CODE	
7. AUTHOR(S) Roy A. Imbsen, Robert A. Schomber				8. PERFORMING ORGANIZATION REPORT NO.	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Imbsen & Associates, Inc. 3213 Ramos Circle Sacramento, CA 95827				10. WORK UNIT NO.	
				11. CONTRACT OR GRANT NO. HPR-PL-1(31)ITEM 207	
12. SPONSORING AGENCY NAME AND ADDRESS ARIZONA DEPARTMENT OF TRANSPORTATION 206 S. 17TH AVENUE PHOENIX, ARIZONA 85007				13. TYPE OF REPORT & PERIOD COVERED Final Report	
				14. SPONSORING AGENCY CODE	
15. SUPPLEMENTARY NOTES Prepared in cooperation with the U.S. Department of Transportation, Federal Highway Administration					
16. ABSTRACT <p>The purpose of this research was to develop a computerized system to determine the adequacy of a bridge or group of bridges to carry specified overload vehicles. The system utilizes two levels of analysis. The Level 1 analysis is the basic rating system for the Arizona Department of Transportation. This analysis computes the overload capacity with a limited amount of data. A Level 2 evaluation, which conducts a more detailed evaluation, uses an enhanced NBIF database, Standard Plans, or a more detailed analysis similar to Brass. A Special Level 2 analysis was also developed to analyze continuous slab bridges by utilizing data from standard slab plans.</p> <p>Detailed case studies were conducted on twenty-five typical Arizona bridges to verify the methodology used in the Level 1 procedure and to correlate the bridge plans with data in the NBIF. The Level 1 procedure gives ratings which are within 10% of the Level 2 procedure for bridges which satisfy the level one assumptions. Similarly, the Special Level 2 analysis for reinforced concrete continuous bridges gives ratings within 10% of the Level 2 analysis. The NBIF compared well with general bridge plans and is applicable to Level 1 analysis for "typical bridges".</p> <p>The research results are reported in three documents:</p> <ol style="list-style-type: none"> <li>1. Final Report</li> <li>2. Volume I: Users Manual</li> <li>3. Volume II: Program Listing</li> </ol>					
17. KEY WORDS Overload, Bridge Rating, National Bridge Inventory Analysis			18. DISTRIBUTION STATEMENT Document is available to the U.S. public through the National Technical Information Service, Springfield, Virginia 22161		
19. SECURITY CLASSIF. (of this report) Unclassified		20. SECURITY CLASSIF. (of this page) Unclassified		21. NO. OF PAGES 142	22. PRICE

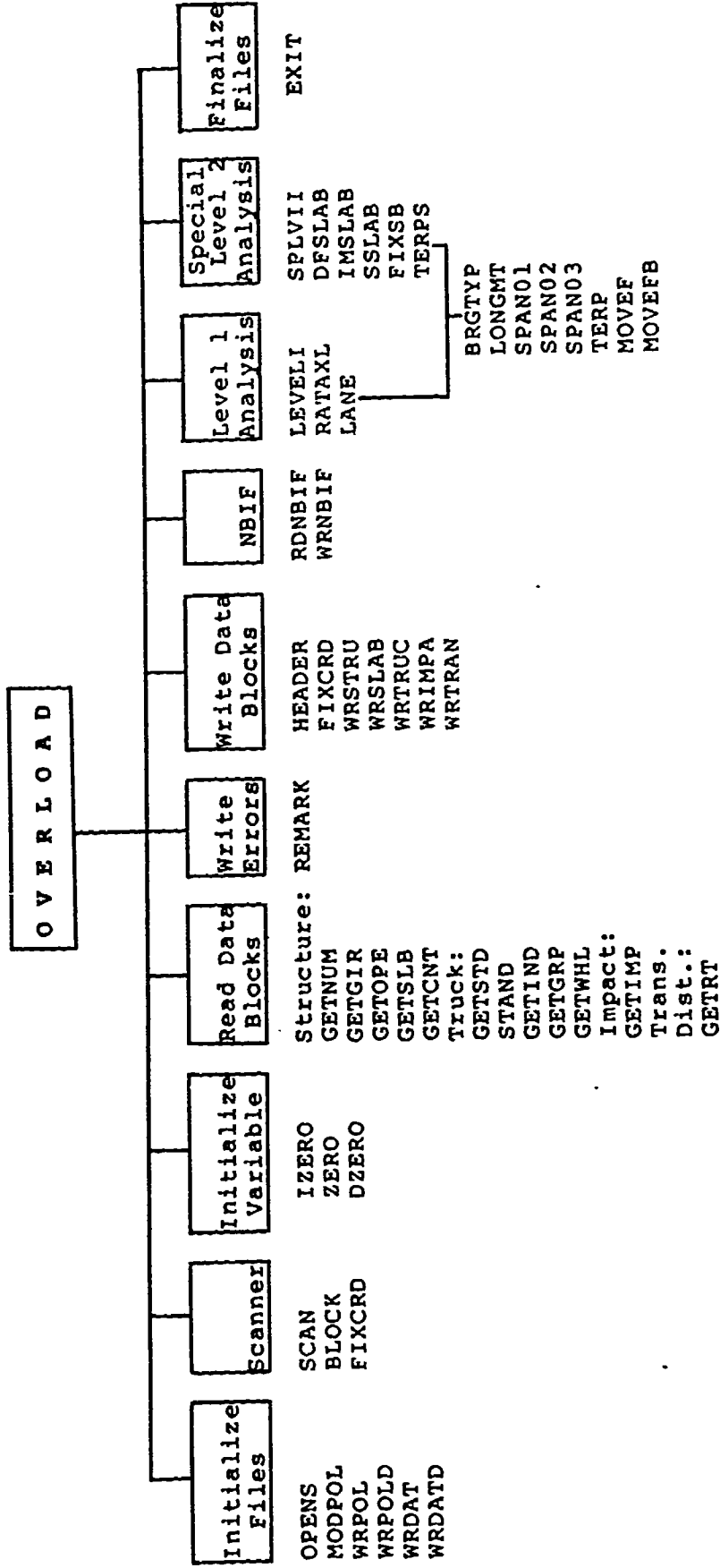


FIGURE 1. OVERLOAD SUBROUTINE CHART

C  
C  
C THIS PROGRAM CREATES A SEQUENTIAL FILE OF A REDUCED NBIF  
C  
C

C  
C CALLS TO : NONE  
C  
C

C  
C VARIABLE DEFINITION  
C

C N - NUMBER OF RE ORDS IN THE NBIF  
C REC1 - STATE CODE  
C IREC2 - STRUCTURE NUMBER  
C REC3A - INVENTORY ROUTE  
C REC3B - INVENTORY ROUTE  
C REC4 - STATE HIGHWAY DEPARTMENT DISTRICT  
C REC5 - COUNTY (PARISH)  
C REC6 - CITY/TOWN CODE  
C REC7 - FEATURES INTERSECTED  
C REC8 - FACILITY CARRIED BY STRUCTURE  
C REC9 - LOCATION  
C IREC10 - INVENTORY ROUTE, MIN. VERTICAL CLEARANCE  
C IREC11 - MILEPOINT  
C REC12 - ROAD SECTION NUMBER  
C REC13 - BRIDGE DESCRIPTION  
C IREC14 - DEFENSE MILEPOINT  
C IREC15 - DEFENSE SECTION LENGTH  
C IREC16 - LATITUDE  
C IREC17 - LONGITUDE  
C IREC18 - PHYSICAL VULNERABILITY  
C IREC19 - BYPASS, DETOUR LENGTH  
C IREC20 - TOLL  
C IREC21 - CUSTODIAN  
C IREC22 - OWNER  
C REC23 - FEDERAL-AID PROJECT NUMBER  
C IREC24 - HIGHWAY SYSTEM  
C IREC25 - ADMINISTRATIVE JURISDICTION  
C IREC26 - FUNCTIONAL CLASSIFICATION  
C IREC27 - YEAR BUILT  
C IREC28 - LANES ON AND UNDER STRUCTURE  
C IREC29 - AVERAGE DAILY TRAFFIC  
C IREC30 - YEAR OF AVERAGE DAILY TRAFFIC  
C IREC31 - DESIGN LOAD  
C IREC32 - APPROACH ROADWAY WIDTH  
C IREC33 - BRIDGE MEDIAN  
C IREC34 - SKEW  
C IREC35 - STRUCTURE FLARED  
C REC36 - TRAFFIC SAFETY FEATURES  
C IREC37 - NAVIGATION CONTROL  
C IREC38 - NAVIGATION VERTICAL CLEARANCE  
C IREC39 - NAVIGATION HORIZONTAL CLEARANCE  
C REC40 - STRUCTURE OPEN, POSTED, CLOSED TO TRAFFIC  
C IREC41 - TYPE SERVICE  
C IREC42 - STRUCTURE TYPE, MAIN  
C IREC43 - STRUCTURE TYPE, APPROACH SPANS

C

C IREC44 - NUMBER OF SPANS IN MAIN UNIT  
 C IREC45 - NUMBER OF APPROACH SPANS  
 C IREC46 - TOTAL HORIZONTAL CLEARANCE  
 C IREC47 - LENGTH OF MAXIMUM SPAN  
 C IREC48 - STRUCTURE LENGTH  
 C IREC49 - CURB OR SIDEWALK WIDTHS  
 C IREC50 - BRIDGE ROADWAY WIDTH, CURB TO CURB  
 C IREC51 - DECK WIDTH, OUT TO OUT  
 C IREC52 - MIN. VERTICAL CLEARANCE OVER BRIDGE ROADWAY  
 C IREC53 - MIN. VERTICAL UNDERCLEARANCE  
 C IREC54 - MIN. LATERAL UNDERCLEARANCE ON RIGHT  
 C IREC55 - MIN. LATERAL UNDERCLEARANCE ON LEFT  
 C REC56 - WEARING SURFACE - PROTECTIVE SYSTEM  
 C REC57 - DECK  
 C REC58 - SUPERSTRUCTURE  
 C REC59 - SUBSTRUCTURE  
 C REC60 - CHANNEL AND CHANNEL PROTECTION  
 C REC61 - CULVERT AND RETAINING WALLS  
 C IREC62 - ESTIMATE REMAINING LIFE  
 C IREC63 - OPERATING RATING  
 C REC64 - APPROACH ROADWAY ALIGNMENT  
 C IREC65 - INVENTORY RATING  
 C REC66 - STRUCTURAL CONDITION  
 C REC67 - DECK GEOMETRY  
 C REC68 - UNDERCLEARANCES, VERTICAL AND HORIZONTAL  
 C REC69 - SAFE LOAD CAPACITY  
 C REC70 - WATERWAY ADEQUACY  
 C REC71 - APPROACH ROADWAY ALIGNMENT  
 C IREC72 - YEAR NEEDED  
 C IREC73 - TYPE OF SERVICE  
 C IREC74 - TYPE OF WORK  
 C IREC75 - LENGTH OF IMPROVEMENT  
 C IREC76 - PROPOSED DESIGN LOADING OF IMPROVEMENT  
 C IREC77 - PROPOSED ROADWAY WIDTH  
 C IREC78 - PROPOSED NUMBER OF LANES  
 C IREC79 - DESIGN ADT  
 C IREC80 - YEAR OF ESTIMATED ADT  
 C IREC81 - YEAR OF PROPOSED ADJACENT ROADWAY IMPROVEMENTS  
 C IREC82 - TYPE OF PROPOSED ADJACENT ROADWAY IMPROVEMENTS  
 C IREC83 - COST OF IMPROVEMENTS  
 C IREC84 - P.E. COST  
 C REC85 - DEMOLITION COST  
 C IREC86 - SUBSTRUCTURE COST  
 C IREC87 - SUPERSTRUCTURE COST  
 C IREC88 - DATE OF LAST INSPECTION  
 C REC89 - SUFFICIENCY RATING  
 C REC90 - RESERVED SPACE FOR WASHING OFFICE

C

C

C

---

 MODIFICATION LOG

C

C

 07/01/1986 RAS INITIAL CODING
 

---

C

C

C

REAL\*8 REC50

C

INTEGER I,II,N,IREC2I

INTEGER IREC2,IREC10,IREC11,4

IREC14,IREC15,

C

```

1      IREC16, IREC17, IREC18, IREC19, IREC20, IREC21, IREC22,
2          IREC24, IREC25, IREC26, IREC27, IREC28, IREC29,
3      IREC30, IREC31, IREC32, IREC33, IREC34, IREC35,
4      IREC37, IREC38, IREC39,          IREC41, IREC42, IREC43,
5      IREC44, IREC45, IREC46, IREC47, IREC48, IREC49, IREC50,
6      IREC51, IREC52, IREC53, IREC54, IREC55,
7          IREC62, IREC63,
8      IREC65,
9      IREC72, IREC73, IREC74, IREC75, IREC76, IREC77, IREC78,
A      IREC79, IREC80, IREC81, IREC82, IREC83, IREC84,
B      IREC86, IREC87, IREC88

```

C

```

INTEGER KK(17)
CHARACTER*1 CK1, CK5, CK6, CK7, CK8, CK9, CK10
CHARACTER*9 CK2
CHARACTER*18 CK3
CHARACTER*25 CK4
CHARACTER*1 BL1, BL5, BL6, BL7, BL8, BL9, BL10
CHARACTER*9 BL2
CHARACTER*18 BL3
CHARACTER*25 BL4

```

C

```

CHARACTER*3 REC1, REC5, REC85
CHARACTER*8 REC3B
CHARACTER*2 REC4, REC13
CHARACTER*4 REC6, REC36
CHARACTER*18 REC8
CHARACTER*25 REC7, REC9
CHARACTER*7 REC23
CHARACTER*1 REC3A, REC40, REC56, REC57, REC61, REC66, REC67, REC68
CHARACTER*1 REC69, REC70, REC71, REC64, REC60, REC58, REC59
CHARACTER*5 REC12, REC89
CHARACTER*28 REC90

```

C

```

DATA BL1 /' '/
DATA BL2 /' '/
DATA BL3 /' '/
DATA BL4 /' '/
DATA BL5 /' '/
DATA BL6 /' '/
DATA BL7 /' '/
DATA BL8 /' '/
DATA BL9 /' '/
DATA BL10 /' '/

```

C

C

```
OPEN THE FILES
```

C

```
CALL INIT
```

C

```
READ(5,10) N
```

10

```
FORMAT(15)
```

C

```
IREC2=0
IREC21=0
```

C

C

```
READ IN THE NBIF
```

C

```
DO 500 I1=1,N
```

C

```
READ(5,20) REC1, IREC2, REC3A, REC3B, REC4, REC5, REC6, REC7,
```

C

```

1      REC8, REC9,      IREC10,IREC11, REC12, REC13,IREC14,
2      IREC15,IREC16,   IREC17,IREC18,IREC19,IREC20,IREC21,
3      IREC22, REC23,   IREC24,IREC25,IREC26,IREC27,IREC28,
4      IREC29,IREC30,   IREC31,IREC32,IREC33,IREC34,IREC35,
5      REC36,IREC37,    IREC38,IREC39, REC40,IREC41,IREC42,
6      IREC43,IREC44,   IREC45,IREC46,IREC47,IREC48,IREC49,
7      IREC50,IREC51,   IREC52,IREC53,IREC54,IREC55, REC56,
8      REC57, REC58,    REC59, REC60, REC61,IREC62,IREC63,
9      REC64,IREC65,    REC66, REC67, REC68, REC69, REC70,
A      REC71,IREC72,    IREC73,IREC74,IREC75,IREC76,IREC77,
B      IREC78,IREC79,   IREC80,IREC81,IREC82,IREC83,IREC84,
C      REC85,IREC86,    IREC87,IREC88, REC89, REC90

```

C

```

C      CHECK FOR UNDERCROSSINGS

```

C

```

C      IF( IREC2 .EQ. IREC21 .AND. REC3A .EQ. '2') GOTO 400

```

C

```

C      NI = IREC21 + 1

```

C

```

C      IDENTIFY STRUCTURES THAT ARE SEQUENTIALLY MISSING
C      BY USING THE BRIDGE NUMBER

```

C

```

C      IF(IREC2 .EQ. NI) GOTO 100

```

```

C      NF = IREC2 - 1

```

```

C      DO 50 I=1,17

```

```

C          KK(1)=0

```

50

```

C          CONTINUE

```

```

C          CK1 = BL1

```

```

C          CK2 = BL2

```

```

C          CK3 = BL3

```

```

C          CK4 = BL4

```

```

C          CK5 = BL5

```

```

C          CK6 = BL6

```

```

C          CK7 = BL7

```

```

C          CK8 = BL8

```

```

C          CK9 = BL9

```

```

C          CK10= BL10 .

```

C

```

C      ZERO OUT INFORMATION FOR MISSING STRUCTURES

```

C

```

C      DO 60 I = NI,NF

```

```

C          WRITE(6,70)KK(1),CK1,CK2,CK3,CK4,(KK(J),J=2,7),CK5,

```

```

C          1      (KK(J),J=8,15),CK6,CK7,CK8,CK9,(KK(J),J=16,17),

```

```

C          2      CK10

```

60

```

C          CONTINUE

```

C

```

C      WRITE OUT SEQUENTIAL REDUCED NBIF

```

C

```

C      100 WRITE(6,70)      IREC2,REC3A,REC3B,

```

```

C          1      REC8, REC9,      IREC11,

```

```

C          2

```

```

C          3

```

```

C          4      IREC29,IREC30,      IREC32,      IREC27,

```

```

C          5      IREC29,IREC30,      IREC32,      IREC34,

```

```

C          6      IREC43,IREC44,      IREC45,      REC40,      IREC42,

```

```

C          7      IREC43,IREC44,      IREC45,      IREC47,IREC48,

```

```

C          8      IREC50,      IREC52,      REC56,

```

```

C          9      REC57, REC58,      REC59,      IREC63,

```

```

C          9      IREC65,      REC66

```

C

```

C      400 IREC21= IREC2

```



C

500 CONTINUE

C

20	FORMAT (	A3,	115,	A1,A8,	A2,	A3,	A4,	A25,
1		A18,	A25,	14,	15,	A5,	A2,	14,
2		13,	15,	16,	11,	12,	11,	11,
3		11,	A7,	12,	11,	12,	14,	14,
4		16,	12,	11,	13,	11,	12,	11,
5		A4,	11,	13,	14,	A1,	12,	13,
6		13,	13,	14,	13,	14,	16,	16,
7		14,	14,	14,	14,	13,	13,	A1,
8		A1,	A1,	A1,	A1,	A1,	12,	13,
9		A1,	13,	A1,	A1,	A1,	A1,	A1,
A		A1,	12,	11,	13,	16,	11,	14,
B		12,	16,	12,	12,	11,	15,	13,
C		A3,	15,	15,	16,	A5,	A28)	

C

70 FORMAT(115,A1,A8,A18,A25,15,14,16,12,13,  
 1 12,A1,313,214,16,214,4A1,213,A1)

C

CLOSE THE FILES

C

CALL CLOSIT

C

STOP  
END

\$DEBUG

C \*\*\*\*\*

C DATABASE - CREATE ARIZONA DIRECT ACCESS FILE FROM REDUCED NBIF

C CALLS TO : NONE

C VARIABLE DEFINITION

- C NREC - NO. RECORDS IN NBIF
- C KK(1) - STRUCTURE NUMBER
- C CK1 - INVENTORY ROUTE
- C CK2 - INVENTORY ROUTE
- C CK3 - FACILITY CARRIED BY STRUCTURE
- C CK4 - LOCATION
- C KK(2) - MILEPOINT
- C KK(3) - YEAR BUILT
- C KK(4) - AVERAGE DAILY TRAFFIC
- C KK(5) - YEAR OF AVERAGE DAILY TRAFFIC
- C KK(6) - APPROACH ROADWAY WIDTH
- C KK(7) - SKEW
- C CK5 - OPEN, CLOSED, POSTED TO TRAFFIC
- C KK(8) - STRUCTURE TYPE, MAIN
- C KK(9) - STRUCTURE TYPE, APPROACH SPANS
- C KK(10) - NO. OF SPANS IN MAIN UNIT
- C KK(11) - NO. OF APPROACH SPANS
- C KK(12) - LENGTH OF MAXIMUM SPAN
- C KK(13) - STRUCTURE LENGTH
- C KK(14) - BRIDGE WIDTH, CURB TO CURB
- C KK(15) - MINIMUM VERTICAL CLEARANCE
- C CK6 - WEARING SURFACE
- C CK7 - DECK CONDITION
- C CK8 - SUPERSTRUCTURE CONDITION
- C CK9 - SUBSTRUCTURE CONDITION
- C KK(16) - OPERATING RATING
- C KK(17) - INVENTORY RATING
- C CK10 - STRUCTURAL CONDITION

C MODIFICATION LOG

C 07-01/1986 RAS INITIAL CODING

C  
 C INTEGER KK(17),NREC,1,J  
 C CHARACTER\*1 CK1,CK5,CK6,CK7,CK8,CK9,CK10  
 C CHARACTER\*8 CK2  
 C CHARACTER\*18 CK3  
 C CHARACTER\*25 CK4  
 C CHARACTER\*20 FLNM  
 C LOGICAL ST

C WRITE(0,(' SEQUENTIAL FILE 8 :',\*))

```
      READ(0,'(A)')FLNM
      OPEN(UNIT=8,FILE=FLNM,STATUS='OLD')
C
      WRITE(0,('( DIRECT ACCESS FILE:',*))
      READ(0,'(A)')FLNM
      OPEN(UNIT=9,FILE=FLNM,STATUS='NEW',RECL=126,ACCESS='DIRECT')
C
      WRITE(0,('( DATABASE NOW EXECUTING', ))
C
      NREC=1

      DO 100 I=1,NREC
         II=1
         READ(8,10)KK(1),CK1,CK2,CK3,CK4,(KK(J),J=2,7),CK5,
1             (KK(J),J=8,15),CK6,CK7,CK8,CK9,(KK(J),J=16,17),
2             CK10

10      FORMAT(115,A1,A8,A18,A25,15,14,16,12,13,
1          12,A1,313,214,16,214,4A1,213,A1)

         WRITE(9,REC=II)KK(1),CK1,CK2,CK3,CK4,(KK(J),J=2,7),CK5,
1             (KK(J),J=8,15),CK6,CK7,CK8,CK9,(KK(J),J=16,17),
2             CK10
100     CONTINUE
C
      CLOSE(8)
      CLOSE(9)
C
      STOP
      END
```

\$DEBUG

C \*\*\*\*\*

C OVERLOAD MAIN PROGRAM

C A BRIDGE RATING PROGRAM USING A LEVEL 1 PROCEDURE

C IMBSEN AND ASSOCIATES, INC.  
C 3213 RAMOS CIRCLE  
C SACRAMENTO, CA 95827

C THIS PROGRAM MAY USE THE WINDOWS OPTION FOR POL INPUT  
C THIS PROGRAM USES THE 'SCAN' INTERPRETER TO READ INPUT

C -----  
C CALLS TO :

C OPENS  
C SETIN (SCAN ROUTINE)  
C SETOUT (SCAN ROUTINE)  
C SCINIT (SCAN ROUTINE)  
C ZERO  
C READSC (SCAN ROUTINE)  
C EXIT  
C IZERO  
C DZERO  
C GETNUM  
C GETGIR  
C GETOPE  
C GETSLB  
C GETCNT  
C GETIND  
C GETGRD  
C GETWHL  
C GETSTD  
C STAND  
C GETIMP  
C GETRT  
C HEADER  
C FIXCRD  
C WRSTRU  
C WRTRUC  
C WRIMPA  
C WRTRAN  
C RDNBIF  
C WRNBIF  
C LEVEL1  
C SPLV11  
C REMARK

C -----  
C VARIABLE DEFINITION

C MAXILD - MAXIMUM AXLE LOADS PERMITTED  
C MAXSTR - MAXIMUM NUMBER STRUCTURES PERMITTED  
C NSTRUC(MAXSTR) - STRUCTURE NUMBER

C OPRING(MAXSTR) - OPERATING RATING  
 C ISPLAN(MAXSTR) - SLAB PLANS  
 C CONTIN(MAXSTR) - CONTINUITY  
 C GRDSPC(MAXSTR) - GIRDER SPACING  
 C FACIMP(MAXSTR) - IMPACT FACTOR  
 C RT(MAXSTR) - TRANSVERSE DISTRIBUTION RATIO  
 C IGROUP(MAXILD) - GROUP IDENTIFICATION  
 C PL(MAXILD) - AXLE LOADS  
 C DX(MAXILD) - DISTANCE TO THE AXLES  
 C WHO(MAXILD) - OUTER WHEEL GAGE  
 C WHI(MAXILD) - INNER WHEEL GAGE  
 C IFL - NUMBER OF INDIVIDUAL AXLES ENTERED  
 C JFG - NUMBER OF GROUP AXLES ENTERED  
 C ISTRUC - NUMBER OF STRUCTURES ENTERED  
 C IPAGE - PAGE NUMBER  
 C ICAP - CAPTION IDENTIFICATION  
 C NREC - NUMBER RECORDS IN NBIF  
 C KK(17) - INTEGER VARIABLES FROM REDUCED NBIF  
 C WIDTH - WIDTH OF TRUCK  
 C CK(1-10) - CHARACTER VARIABLES FROM REDUCED NBIF

-----  
 C  
 C MODIFICATION LOG

C 06/11/1986 RAS INITIAL CODING  
 C 07/02/1986 RAS POL MODIFICATION

\*\*\*\*\*

COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN  
 COMMON /SCANCT/ ECHAR, ECHO, LABEL, LIMIT, MARK, PROMT,  
 1 POINT, RECSIZ, INIT, LEOF, EOL, MENU,  
 2 AUTORD, COMMENT, SIGNED  
 COMMON /SCANIO/ INUNIT, IOUT, FILES(10), FILPT, FILLIM, LREMOT  
 COMMON /SCANLN/ COL, JUMP, NENT, PSTATE, SKIP, DOREAD, INCOL,  
 1 JSTART(80), Ibuff(81), Jbuff(81), IDIGIT(81),  
 2 CARD(81)  
 COMMON /SCANTB/ NSEPTB, NCLASS, ITAB(256), ICLASS(256)  
 COMMON /SCANIM/ NCPW, BLANK, INTZER, INTNIN, INTC, INTCOM, INTBLK  
 EQUIVALENCE (IVAL(1), VALUE, IVALUE)  
 INTEGER COL, SKIP, PSTATE, NBLANK, RECLN, ENDCHR, DUMMY  
 LOGICAL PROMSW, ECHOSW, COMSW, ATRDSW, EOLSW, EOF SW, MENUSW, PTSW,  
 1 SIGNSW, NEXT, MATCH, DOREAD, ENDFIL, NUMI, NUMR, STRING,  
 2 ENDCRD, SEP, INTEGR

C INTEGER MAXILD, MAXSTR  
 C PARAMETER (MAXILD=30, MAXSTR=40)

C INTEGER NSTRUC(MAXSTR), OPRING(MAXSTR), ISPLAN(MAXSTR)  
 C INTEGER CONTIN(MAXSTR)  
 C REAL \*8 GRDSPC(MAXSTR), FACIMP(MAXSTR), RT(MAXSTR)

C INTEGER IGROUP(MAXILD)  
 C REAL \*8 PL(MAXILD), DX(MAXILD), WHO(MAXILD), WHI(MAXILD)

C INTEGER I, IMSG, INTEG, INUNIT, IOUT, IER  
 C INTEGER IPAGE, ICAP, LEVOUT, ISTRUC, ISLAB  
 C INTEGER IFL, JFG  
 C INTEGER NREC, KK(17), ICSLAB

```

REAL*4 REEL
REAL*8 WIDTH
CHARACTER*1 CK1,CK5,CK6,CK7,CK8,CK9,CK10
CHARACTER*9 CK2
CHARACTER*18 CK3
CHARACTER*25 CK4
CHARACTER*4 BLANK1,TITLE(20)
DATA BLANK1/' '/

C
NREC =9861

C
IMSG =6
INUNIT=5
IOUT =6
IER =0
IPAGE =0
IOVLD =0

C
C OPEN FILES
C
CALL OPENS

C
C INITIALIZE SCAN ROUTINES
C
NBLANK=80
RECLEN=80
ENDCHR= '$'
PROMSW=.FALSE.
ECHOSW=.TRUE.
COMSW=.FALSE.
ATRDSW=.FALSE.
EOLSW=.TRUE.
EOFSW=.TRUE.
MENUSW=.FALSE.
PTSW=.FALSE.
SIGNSW=.FALSE.
CALL SETIN(INUNIT)
CALL SETOUT(IOUT)
CALL SCINIT(NBLANK,RECLEN,ENDCHR,PROMSW,ECHOSW,COMSW,ATRDSW,
1          EOLSW,EOFSW,MENUSW,PTSW,SIGNSW)

C
C READ OVERLOAD BLOCK DATA
C
CALL ZERO(TITLE,20,BLANK1)
CALL READSC
IF(ENDFIL(DUMMY)) CALL EXIT
IF(.NOT. MATCH('OVERLOAD',3)) GOTO 8000
IF(ENDCRD(DUMMY)) GOTO 40
IF(.NOT. STRING(DUMMY)) GOTO 8001
DO 30 I=1,NWD
    TITLE(I)=ENTITY(I)
    ENTITY(I)=BLANK1
30 CONTINUE
40 CALL READSC
IF(.NOT. MATCH('OUTPUT',3)) GOTO 8024
IF(.NOT. MATCH('LEVEL',3)) GOTO 8027
IF(.NOT. INTEGR(INTEG)) GOTO 8023
LEVOUT=INTEG
IF(LEVOUT.LT.1.OR.LEVOUT.GT.3) GOTO 8029

C

```

```

C READ STRUCTURE DATA BLOCK
C
  CALL READSC
  CALL IZERO(NSTRUC,MAXSTR,0)
  CALL DZERO(GRDSPC,MAXSTR,0.DO)
  CALL IZERO(OPRING,MAXSTR,0)
  CALL IZERO(ISPLAN,MAXSTR,0)
  CALL IZERO(CONTIN,MAXSTR,0)
  IF(.NOT. MATCH('STRUCTURES',3))GOTO 8002
  IF(.NOT. MATCH('NUMBER',3) )GOTO 8003
  CALL GETNUM(NSTRUC,ISTRUC,MAXSTR,IMSG)
  CALL READSC
  IF(MATCH ('GIRDER',3)      ) GOTO 100
  GOTO 120
100 IF(.NOT. MATCH('SPACING',3) ) GOTO 8005
  CALL GETGIR(GRDSPC,ISTRUC,IMSG)
  CALL READSC
120 IF(MATCH('OPERATING',3))      GOTO 130
  GOTO 135
130 IF(.NOT. MATCH('RATING',3))  GOTO 8026
  CALL GETOPE(OPRING,ISTRUC,IMSG)
  CALL READSC
135 IF(MATCH('SLAB',3))          GOTO 137
  GOTO 140
137 IF(.NOT. MATCH('PLANS',3))   GOTO 8028
  CALL GETSLB(ISPLAN,ISLAB,ISTRUC,IMSG)
  CALL READSC
140 IF(MATCH('CONTINUITY',3))    GOTO 142
  GOTO 145
142 CALL GETCNT(CONTIN,ISTRUC,IMSG)
  CALL READSC
C
C READ TRUCK DATA BLOCK
C
145 CALL IZERO(IGROUP,MAXILD,0)
  CALL DZERO(PL      ,MAXILD,0.DO)
  CALL DZERO(DX      ,MAXILD,0.DO)
  CALL DZERO(WHO,MAXILD,0.DO)
  CALL DZERO(WHI,MAXILD,0.DO)
  IF(.NOT. MATCH('TRUCK',3) ) GOTO 8007
  IF( MATCH('STANDARD',3) ) GOTO 205
  IF( MATCH('INDIVIDUAL',3) ) GOTO 150
  IF(.NOT. MATCH('GROUP',3) ) GOTO 8008
C
C GET LOAD CONFIGURATION
C
  CALL GETGRP(PL,DX,IGROUP,IFL,JFG,MAXILD,IMSG)
  GOTO 160
150 CALL GETIND(PL,DX,IGROUP,IFL,JFG,MAXILD,IMSG)
C
160 CALL READSC
C
C READ OUTER WHEEL CONFIGURATION
C
  IF( .NOT. MATCH('OUTER',3)) GOTO 8009
  IF( .NOT. MATCH('WHEEL',3)) GOTO 8010
  IF( .NOT. MATCH('SPACING',3)) GOTO 8011
  CALL GETWHL(IGROUP,WHO,IFL,JFG,MAXILD,IMSG)
C
  CALL READSC

```

```

C
C   READ INNER WHEEL CONFIGURATION
C
    IF(MATCH('INNER',3))          GOTO 200
      GOTO 210
200  IF( .NOT. MATCH('WHEEL',3))  GOTO 8012
    IF( .NOT. MATCH('SPACING',3)) GOTO 8013
    CALL GETWHL(IGROUP,WHI,IFL,JFG,MAXILD,IMSG)
    GOTO 208
C
C   READ STANDARD VEHICLES
C
205  IF(.NOT. INTEGR(INTEG))      GOTO 8023
      IOVLD=INTEG
      IF( IOVLD .LT. 1 .OR. IOVLD .GT. 11) GOTO 8030
    CALL STAND(IOVLD,IGROUP,PL,DX,WHO,WHI,IFL,JFG,IMSG)
C
208  CALL READSC
210  IF(.NOT. MATCH('VEHICLE',3)) GOTO 8014
    IF(.NOT. MATCH('WIDTH',3))   GOTO 8015
    IF(.NOT. NUMR(REEL))          GOTO 8016
      WIDTH=DBLE(REEL)
C
C   READ IMPACT DATA BLOCK
C
    CALL READSC
    CALL DZERO(FACIMP,MAXSTR,0.DO)
    IF( MATCH('IMPACT',3))       GOTO 220
      GOTO 230
220  CALL GETIMP(FACIMP,ISTRUC,IMSG)
C
C   READ TRANSVERSE DISTRIBUTION BLOCK
C
    CALL READSC
230  CALL DZERO(RT,MAXSTR,1.DO)
    IF( MATCH('TRANSVERSE',3))  GOTO 235
      GOTO 240
235  IF(.NOT. MATCH('DISTRIBUTION',3)) GOTO 8020
    IF( .NOT. MATCH('RATIO',3))    GOTO 8019
    CALL GETRT(RT,ISTRUC,IMSG)
C
    CALL READSC
240  IF(.NOT. MATCH('FINISH',3))   GOTO 8021
C
    CALL HEADER(TITLE,1,IPAGE)
    REWIND 5
    CALL FIXCRD
C
    CALL HEADER(TITLE,2,IPAGE)
    CALL WRSTRU(NSTRUC,GRDSPC,OPRING,CONTIN,ISTRUC)
C
    IF(ISLAB .EQ. 0) GOTO 250
    CALL HEADER(TITLE,2,IPAGE)
    CALL WRSLAB(ISPLAN,ISLAB)
C
250  CALL HEADER(TITLE,2,IPAGE)
    CALL WRTRUC(PL,DX,IFL,IGROUP,WHO,WHI,WIDTH,IOVLD)
C
    CALL HEADER(TITLE,2,IPAGE)
    CALL WRIMPA(FACIMP,NSTRUC,ISTRUC)

```



```

C      CALL HEADER(TITLE,2,IPAGE)
      CALL WRTRAN(NSTRUC,RT,ISTRUC)
C
      ICSLAB=0
      DO 500 I=1,ISTRUC
      CALL RDNBIF(NREC,NSTRUC(I),KK,CK1,CK2,CK3,CK4,CK5,CK6,CK7,CK8,CK9,
1         CK10)
      IF(LEVOUT .NE. 3)CALL HEADER(TITLE,3,IPAGE)
      IF(LEVOUT .NE. 3)CALL WRNBIF(NREC,NSTRUC(I),KK,CK1,CK2,CK3,CK4,
1         CK5,CK6,CK7,CK8,CK9,CK10,LEVOUT)
C
      IF(LEVOUT .EQ. 2)GOTO 500
      CALL HEADER(TITLE,4,IPAGE)
      IF( KK(8) .EQ. 201) GOTO 350
      GOTO 400
350     ICSLAB = ICSLAB + 1
      IF(1SPLAN(ICSLAB).GT.0)GOTO 450
400     CALL LEVEL1(NREC,KK,CK1,CK2,CK3,CK4,CK5,CK10,PL,
1         DX,GRDSPC(1),WHO,WHI,WIDTH,FACIMP(1),IGROUP,IFL,
2         OPRING(1),NSTRUC(1),RT(1),CONTIN(1),IOVLD)
      GOTO 500
450     CALL SPLV11(NREC,KK,CK1,CK2,CK3,CK4,CK5,CK10,PL,
1         DX,GRDSPC(1),WHO,WHI,WIDTH,FACIMP(1),IGROUP,IFL,
2         NSTRUC(1),RT(1),1SPLAN(ICSLAB),IOVLD)
500     CONTINUE
C
      CALL EXIT
C
C      ERROR MESSAGES
C
8000 CALL REMARK(1MSG,'ERROR - THE WORD <OVERLOAD> IS MISSING.      ')
      GOTO 8990
8001 CALL REMARK(1MSG,'ERROR - TITLE IS MISSING.                    ')
      GOTO 8990
8002 CALL REMARK(1MSG,'ERROR - THE WORD <STRUCTURE> IS MISSING.     ')
      GOTO 8990
8003 CALL REMARK(1MSG,'ERROR - THE WORD <NUMBER> IS MISSING.        ')
      GOTO 8990
8004 CALL REMARK(1MSG,'ERROR - INCORRECT NUMERIC INTEGER INPUT.     ')
      GOTO 8990
8005 CALL REMARK(1MSG,'ERROR - THE WORD <SPACING> IS MISSING.       ')
      GOTO 8990
8006 CALL REMARK(1MSG,'ERROR - INCORRECT NUMERIC REAL INPUT.        ')
      GOTO 8990
8007 CALL REMARK(1MSG,'ERROR - THE WORD <TRUCK> IS MISSING.         ')
      GOTO 8990
8008 CALL REMARK(1MSG,'ERROR - THE WORD <GROUP> IS MISSING.         ')
      GOTO 8990
8009 CALL REMARK(1MSG,'ERROR - THE WORD <OUTER> IS MISSING.         ')
      GOTO 8990
8010 CALL REMARK(1MSG,'ERROR - THE WORD <WHEEL> IS MISSING.         ')
      GOTO 8990
8011 CALL REMARK(1MSG,'ERROR - THE WORD <SPACING> IS MISSING.       ')
      GOTO 8990
8012 CALL REMARK(1MSG,'ERROR - THE WORD <WHEEL> IS MISSING.         ')
      GOTO 8990
8013 CALL REMARK(1MSG,'ERROR - THE WORD <SPACING> IS MISSING.       ')
      GOTO 8990
8014 CALL REMARK(1MSG,'ERROR - THE WORD <VEHICLE> IS MISSING.       ')

```

```

      GOTO 8990
8015 CALL REMARK(IMG, 'ERROR - THE WORD <WIDTH> IS MISSING.      ')
      GOTO 8990
8016 CALL REMARK(IMG, 'ERROR - INCORRECT NUMERIC REAL INPUT.    ')
      GOTO 8990
8017 CALL REMARK(IMG, 'ERROR - THE WORD <IMPACT> IS MISSING.     ')
      GOTO 8990
8018 CALL REMARK(IMG, 'ERROR - INCORRECT NUMERIC REAL INPUT.    ')
      GOTO 8990
8019 CALL REMARK(IMG, 'ERROR - THE WORD <RATIO> IS MISSING.      ')
      GOTO 8990
8020 CALL REMARK(IMG, 'ERROR - THE WORD <DISTRIBUTION> IS MISSING. ')
      GOTO 8990
8021 CALL REMARK(IMG, 'ERROR - THE WORD <FINISH> IS MISSING.     ')
      GOTO 8990
8023 CALL REMARK(IMG, 'ERROR - INCORRECT NUMERIC INTEGER INPUT.    ')
      GOTO 8990
8024 CALL REMARK(IMG, 'ERROR - THE WORD <OUTPUT> IS MISSING.      ')
      GOTO 8990
8026 CALL REMARK(IMG, 'ERROR - THE WORD <RATING> IS MISSING.      ')
      GOTO 8990
8027 CALL REMARK(IMG, 'ERROR - THE WORD <LEVEL> IS MISSING.      ')
      GOTO 8990
8028 CALL REMARK(IMG, 'ERROR - THE WORD <PLANS> IS MISSING.     ')
      GOTO 8990
8029 CALL REMARK(IMG, 'ERROR - INCORRECT OUTPUT LEVEL INPUT.    ')
      GOTO 8990
8030 CALL REMARK(IMG, 'ERROR - INCORRECT STANDARD VEHICLE INPUT.    ')
C
8990 CALL EXIT
      END

```

```

C
C   OPENS - OPEN FILES
C
C   THIS SUBROUTINE OPENS THE FILES
C
C-----
C
C   CALLS TO :  MODPOL
C
C-----
C
C   VARIABLE DEFINITION
C
C   PASSED :  NONE
C
C   RETURNED : NONE
C
C-----
C
C   MODIFICATION LOG
C
C       06/11/1986  RAS  INITIAL CODING
C       07/14/1986  RAS  ADDED MENU FILES
C
C *****
C
C   CHARACTER*3  QUES
C   CHARACTER*20 FLNM
C   LOGICAL ST
C
C   WRITE(0,('( ' ' ' , ))
C   WRITE(0,('( ' ADOT OVERLOAD BRIDGE RATING (VERSION 1.3)' , ))
C   WRITE(0,('( ' ' ' , ))
C   WRITE(0,('( ' Do you want to read in the MENU file?' , ))
C   WRITE(0,('( ' (Enter YES or NO)' , * ))
C   READ(0, '(A)') QUES
C   WRITE(0,('( ' ' ' , ))
C   IF(QUES .EQ. 'YES' .OR. QUES .EQ. 'Y ') GOTO 100
C   IF(QUES .EQ. 'NO ' .OR. QUES .EQ. 'N ') GOTO 200
C   WRITE(0,('( ' Error in Input - execution terminated' , ))
C   STOP
C
C 100 OPEN(UNIT=9, FILE='ADOT.POL', STATUS='OLD')
C
C   INQUIRE(FILE='OVER.DAT', EXIST=ST)
C   IF(ST) THEN
C     OPEN(UNIT=5, FILE='OVER.DAT', STATUS='OLD')
C     CLOSE(UNIT=5, STATUS='DELETE')
C   ENDIF
C   OPEN(UNIT=5, FILE='OVER.DAT', STATUS='NEW')
C
C   IOUTER=5
C
C   CALL MODPOL(IOUTER)
C   REWIND 5
C   GOTO 300

```

```

C
200 WRITE(0,('' Input File Name :'',*))
    READ(0,('A'))FLNM
    OPEN(UNIT=5,FILE=FLNM,STATUS='OLD')
C
300 WRITE(0,('' Scan Echo Name :'',*))
    READ(0,('A'))FLNM
    INQUIRE(FILE=FLNM,EXIST=ST)
    IF(ST) THEN
        OPEN(UNIT=6,FILE=FLNM,STATUS='OLD')
        CLOSE(UNIT=6,STATUS='DELETE')
    ENDIF
    OPEN(UNIT=6,FILE=FLNM,STATUS='NEW')
C
    WRITE(0,('' Output File Name:'',*))
    READ(0,('A'))FLNM
    INQUIRE(FILE=FLNM,EXIST=ST)
    IF(ST) THEN
        OPEN(UNIT=7,FILE=FLNM,STATUS='OLD')
        CLOSE(UNIT=7,STATUS='DELETE')
    ENDIF
    OPEN(UNIT=7,FILE=FLNM,STATUS='NEW')
C
    OPEN(UNIT=8,FILE='ARIZONA.DIR',STATUS='OLD',RECL=126,
1      ACCESS='DIRECT')
C
    WRITE(0,(''      '',))
    WRITE(0,('' OVERLOAD NOW EXECUTING'',))
    WRITE(0,(''      '',))
    RETURN
    END

```

```

$DEBUG
SUBROUTINE MODPOL(IOUTER)
C
C *****
C
C MODPOL - MODIFY POL
C
C THIS SUBROUTINE MODIFIES THE POL
C FROM THE RUNFILE OF MENU WINDOWS
C
C -----
C
C CALLS TO :
C
C WRPOL
C WRPOLD
C WRDAT
C WRDATD
C
C -----
C
C VARIABLE DEFINITION
C
C PASSED :
C
C IOUTER - FILE UNIT NUMBER
C
C -----
C
C MODIFICATION LOG
C
C 07/14/1986 RAS INITIAL CODING
C
C *****
C
C CHARACTER*1 AB(80),AE(80),IBK(80)
C INTEGER I,INITL
C
C INITL=0
C
C READ(9,1) (AB(I), I=1,80)
1 FORMAT(80A1)
C
5 READ(9,1,END=999) (AE(I), I=1,80)
C
IF(AB(1).NE.AE(1).OR.AB(2).NE.AE(2).OR.AB(3).NE.AE(3))GOTO 100
GOTO 200
100 IF(INITL .EQ. 0) GOTO 125
GOTO 150
125 CALL WRPOL(AB,IOUTER)
C
DO 140 I=1,80
IBK(I) = AB(I)
AB(I) = AE(I)
140 CONTINUE
C
GOTO 5
C
150 CALL WRDAT(AB,IOUTER)
C

```

```

      DO 175 I=1,80
        IBK(I) = AB(I)
        AB(I) = AE(I)
175  CONTINUE
C
      INITL=0
      GOTO 5
C
200  IF(INITL .EQ. 0) GOTO 250
      GOTO 300
C
250  CALL WRPOLD(AB, IOUTER)
      INITL=1
      DO 270 I=1,80
        IBK(I) = AB(I)
        AB(I) = AE(I)
270  CONTINUE
C
      GOTO 5
C
300  CALL WRDATD(AB, IOUTER)
      INITL=1
      DO 350 I=1,80
        IBK(I)= AB(I)
        AB(I) = AE(I)
350  CONTINUE
      GOTO 5
C
999  IF(AB(1).EQ.IBK(1) .AND. AB(2).EQ.IBK(2)
1    .AND. AB(3).EQ.IBK(3)) GOTO 900
      CALL WRPOL(AB, IOUTER)
      GOTO 950
900  CALL WRDAT(AB, IOUTER)
C
950  WRITE(IOUTER,1000)
1000 FORMAT('FINISH')
      END

```

```

$DEBUG
SUBROUTINE WRPOL(AB, IOUSER)
C
C *****
C
C   WRPOL - WRITE POL
C
C   THIS SUBROUTINE WRITES OUT THE POL EXACT FROM
C   THE RUNFILE OF MENU WINDOWS
C
C -----
C
C   CALLS TO : NONE
C
C -----
C
C   VARIABLE DEFINITION
C
C   PASSED :
C
C       AB - BEGINNING LINE WITH 80 CHARACTERS
C
C -----
C
C   MODIFICATION LOG
C
C       07/14/1986  RAS  INITIAL CODING
C
C *****
C
C   CHARACTER*1 AB(80),ABW(80)
C   INTEGER I, ICNT
C
C   IF( AB(1).EQ. 'O' .AND.AB(2) .EQ. 'V' .AND.AB(3).EQ. 'E')GOTO 100
C
C   DO 50 I=1,80
C     IF( AB(I) .EQ. ';' ) GOTO 25
C     GOTO 30
25  ABW(I) = ' '
C     GOTO 50
30  ABW(I) = AB(I)
50  CONTINUE
C
C   WRITE(IOUSER, 75)(ABW(I), I=1,80)
75  FORMAT(80A1)
C   RETURN
C
C   100 ICNT=0
C     DO 150 I=1,80
C       IF( AB(I) .EQ. ';' ) GOTO 120
C       GOTO 125
120  ICNT = ICNT + 1
C     ABW(I) = ' '
C     IF( ICNT .EQ. 2) ABW(I) = '''
C     IF( ICNT .EQ. 3) ABW(I) = ''''
C     GOTO 150
125  ABW(I) = AB(I)
150  CONTINUE
C
C   WRITE(IOUSER, 200)(ABW(I), I=1,80)

```

200 FORMAT(80A1)  
RETURN  
END



```

C
C WRPOLD - WRITE POL WITH A DASH AT THE END
C
C THIS SUBROUTINE WRITES OUT THE POL EXACT WITH A DASH AT THE
C END FROM THE RUNFILE OF MENU WINDOWS
C
C -----
C
C CALLS TO : NONE
C
C -----
C
C VARIABLE DEFINITION
C
C PASSED :
C
C AB - BEGINNING LINE WITH 80 CHARACTERS
C
C -----
C
C MODIFICATION LOG
C
C 07/14/1986 RAS INITIAL CODING
C
C *****
C
C CHARACTER*1 AB(80),ABW(80)
C INTEGER I,ICNT
C
C ICNT=0
C
C DO 50 I=1,80
C IF( AB(I) .EQ. ';' ) GOTO 25
C GOTO 30
25 ICNT=ICNT+1
C ABW(I) = ' '
C IF(ICNT .EQ. 4) ABW(I) = '-'
C GOTO 50
30 ABW(I) = AB(I)
50 CONTINUE
C
C WRITE(10,75)(ABW(I),I=1,80)
75 FORMAT(80A1)
C
C RETURN
C
C END

```

\$DEBUG

SUBROUTINE WRDAT (AB, IOUTER)

```
C *****
C
C WRDAT - WRITE DATA ONLY
C
C THIS SUBROUTINE WRITES OUT ONLY
C FROM THE RUNFILE OF MENU WINDOWS
C -----
C
C CALLS TO : NONE
C -----
C
C VARIABLE DEFINITION
C
C PASSED :
C
C AB - BEGINNING LINE WITH 80 CHARACTERS
C -----
C
C MODIFICATION LOG
C
C 07/14/1986 RAS INITIAL CODING
C *****
C
C CHARACTER*1 AB(80),ABW(80)
C INTEGER I,ICNT, IDELB, IDELE
C
C ICNT=0
C
C DO 50 I=1,80
C IF( AB(I) .EQ. ';' ) GOTO 25
C GOTO 50
25 ICNT=ICNT+1
C IF(ICNT .EQ. 2) IDELB=1
C IF(ICNT .EQ. 3) IDELE=1
50 CONTINUE
C
C DO 100 I=1, IDELB
C ABW(I) = ' '
100 CONTINUE
C
C DO 200 I=IDELB+1, IDELE-1
C ABW(I) = AB(I)
200 CONTINUE
C
C DO 300 I=IDELE,80
C ABW(I) = ' '
300 CONTINUE
C
C WRITE(IOUTER,400)(ABW(I),I=1,80)
400 FORMAT(80A1)
C RETURN
C END
```

```

$DEBUG
SUBROUTINE WRDATD(AB,IOUTER)
C
C *****
C
C WRDATD - WRITE DATA ONLY WITH A DASH AT THE END
C
C THIS SUBROUTINE WRITES OUT ONLY THE DATA WITH A DASH AT THE
C END FROM THE RUNFILE OF MENU WINDOWS
C
C -----
C
C CALLS TO : NONE
C
C -----
C
C VARIABLE DEFINITION
C
C PASSED :
C
C AB - BEGINNING LINE WITH 80 CHARACTERS
C
C -----
C
C MODIFICATION LOG
C
C 07/14/1986 RAS INITIAL CODING
C
C *****
C
C CHARACTER*1 AB(80),ABW(80)
C INTEGER I,ICNT, IDELB, IDELE
C
C ICNT=0
C
C DO 50 I=1,80
C IF( AB(I) .EQ. ';' ) GOTO 25
C GOTO 50
25 ICNT=ICNT+1
C IF(ICNT .EQ. 2) IDELB=1
C IF(ICNT .EQ. 3) IDELE=1
50 CONTINUE
C
C DO 100 I=1,IDELE
C ABW(I) = ' '
100 CONTINUE
C
C DO 200 I=IDELEB+1,IDELE-1
C ABW(I) = AB(I)
200 CONTINUE
C
C ABW(IDELE) = ' '
C ABW(IDELE+1)= '-'
C
C DO 300 I=IDELE+2,80
C ABW(I) = ' '
300 CONTINUE
C
C WRITE(IOUTER,400)(ABW(I),I=1,80)
400 FORMAT(80A1)

```

RETURN  
END

```

$DEBUG
C *****
C *****
C *****
C **
C **
C **          S C A N
C **          =====
C **
C **          A FREE FORM INPUT FORTRAN SUBPROGRAM SYSTEM
C **          IBM-AT AND XT VERSION
C **
C **
C **          CIVIL ENGINEERING SYSTEMS LABORATORY
C **          DEPARTMENT OF CIVIL ENGINEERING
C **          UNIVERSITY OF ILLINOIS AT URABANA-CHAMPAIGN
C **
C **
C **          VERSION MODIFIED AT E. C. C. FOR USE ON THE IBM-AT
C **
C **
C *****
C *****
C *
C * SCAN
C *
C *****
C          SUBROUTINE SCAN
C
C          SCAN
C
COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
COMMON /SCANCT/ ECHAR, ECHO, ILABEL, LIMIT, MARK, PROMT,
1 POINT, RECSIZ, INIT, LEOF, EOL, MENU,
2 AUTORD, COMMNT, SIGNED
COMMON /SCANIO/ INUNIT, IOUT, FILES(10), FILPT, FILLIM, LREMOT
COMMON /SCANLN/ COL, JUMP, NENT, PSTATE, SKIP, DOREAD,
1 INCOL, JSTART(80), Ibuff(81), Jbuff(81),
2 IDIGIT(81), CARD(81)
COMMON /SCANTB/ NSEPTB, NCLASS, ITAB(256), ICLASS(256)
COMMON /SCANIM/ NCPW,BLANK,INTZER,INTNIN,INTC,INTCOM,INTBLK
EQUIVALENCE (IVAL(1),VALUE,IVALUE), (FLT,INT)
REAL F,FLT,SEIDSH
INTEGER RECSIZ, FILES, FILPT, FILLIM, COL, SKIP, PSTATE
INTEGER ISTATE(110), STATE, EXP, OCOL, NDOT, ISIG, NSIG
INTEGER NBLANK, LIM, INT
LOGICAL ECHO, PROMT, POINT, INIT, LEOF, EOL, MENU,
1 COMMNT, AUTORD, SIGNED, DOREAD, NEXT
DATA OCOL/0/, NDOT/0/
DATA ISIG/1/,NSIG/1/,INT/0/,F/.1/,EXP/0/
C          D + - E A Q . S B $
C          I          L U     E L
C          G          P O     P A
C          I          H T          N
C          T          A E          K
C          1      GET
C          2      +-
C          3      INTEGER
C          4      REAL
C          5      START EXP

```

```

C      6      EXP
C      7      LABEL
C      8      NAME, STRING
C      9      STRING
C      A      ANYTEXT
C      9      ANYTEXT
C      DATA ISTATE / 18, 17, 16, 26, 26, 30, 32, 29, 6, 7,
1      18, 17, 16, 26, 26, 13, 32, 29, 6, 7,
2      18, 11, 11, 11, 11, 11, 35, 11, 11, 11,
3      19, 11, 11, 28, 28, 11, 20, 11, 11, 11,
4      21, 10, 10, 22, 28, 10, 28, 10, 10, 10,
5      25, 24, 23, 28, 28, 9, 28, 9, 9, 9,
6      25, 9, 9, 28, 28, 9, 28, 9, 9, 9,
7      4, 8, 8, 4, 4, 8, 27, 8, 8, 8,
8      4, 14, 14, 4, 4, 14, 27, 14, 14, 14,
9      4, 4, 4, 4, 4, 12, 4, 4, 4, 33,
A      4, 8, 8, 4, 4, 8, 4, 8, 8, 8 /

```

```

C
C
2      NENT=NENT+1
      NCHAR = 0
      STATE = PSTATE
      NBLANK = 0
      ENTITY(1) = BLANK
      JSTART(NENT)=0
      OCOL=COL
      LIM=LIMIT

```

```

C
C GO TO ACTION

```

```

C
1      GOTO (10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,
1      170,180,190,200,210,220,230,240,250,260,270,280,290,
2      300,310,320,330,340,350), JUMP

```

```

C
C GET A LINE

```

```

C
5      DOREAD = .TRUE.
10     IF(DOREAD)CALL SCANRD
      COL = 0
      NENT = 1
      JSTART(1) = 0
      JUMP=2
      LIM=80
      IF(MODE.EQ.10)GOTO 12
      GOTO 20

```

```

C
C END OF FILE

```

```

C
12     FILPT = FILPT-1
      IF(FILPT.GT.0)NUNIT = FILES(FILPT)
      IF(FILPT.EQ.0)MODE=-10
      IF(LEOF)GO TO 13
      IF(FILPT.GT.0)GO TO 5
      CALL SCANMS(1)
      CALL EXIT
13     JUMP = 1
      DOREAD = .TRUE.
      GOTO 150

```

```

C
C NEXT CHARACTER

```

```

C
 20  COL = COL+1
C
C  GET CHARACTER
C
 30  JUMP = ISTATE(10*STATE+IBUFF(COL))
      GOTO 1
C
C  SET JUMP TO NEXT CHARACTER
C
 40  JUMP = 2
C
C  PACK CHARACTER
C
 50  NCHAR = NCHAR + 1
      GOTO 1
C
C  COUNT BLANKS
C
 60  NBLANK = NBLANK + 1
      IF(NBLANK.LE.LIM)GOTO 20
C
C  CHECK IF END OF STRING
C
      IF(PSTATE.EQ.1)GOTO 130
C
C  END LINE
C
 70  IF(.NOT.EOL)GO TO 75
      JUMP = 3
      IF(AUTORD)JUMP = 1
      IF(AUTORD)DOREAD = .TRUE.
      ENTITY(1)=BLANK
      MODE = 9
      NCHAR=0
      NWD = 0
      COL=MAX0(OCOL,1)
      JSTART(NENT)=COL
      GOTO 150
 75  NCHAR = 0
      NWD = 0
      NBLANK = 0
      STATE = PSTATE
      GOTO 5
C
C  PACK BLANK EXIT
C
 80  JUMP = 3
 85  ICOLMN = 0
      IF(NENT.NE.0)ICOLMN = JSTART(NENT)
      CALL SCANPK
      IF(ENTITY(1).EQ.SEIDSH)GOTO 75
      IF(MODE.EQ.6 .AND. IVALUE.EQ.16)GOTO 2
      GOTO 150
C
C  END EXPONENT
C
 90  FLT = FLT * 10. ** EXP
      EXP = 0
      NSIG = 1

```

```

C
C END REAL
C
100 F = 0.1
C
C END INTEGER
C
110 IVALUE = INT
    ISIG = 1
    INT = 0
    GOTO 80
C
C END STRING
C
120 SKIP = COL
340 JUMP = 2
    GOTO 85
C
C END GET STRING
C
130 MODE = 8
    PSTATE = 0
    JUMP = 2
    GOTO 150
C
C END NAME
C
140 NCHAR = NDOT + 1
    GOTO 80
C
C RETURN
C
150 RETURN
C
C MINUS FOUND
C
160 ISIG = -1
C
C PLUS FOUND
C
170 STATE = 2
    MODE = 6
    INT = ITAB(JBUFF(COL)+1)
    IVALUE = INT
    GOTO 310
C
C START INTEGER
C
180 STATE = 3
    INT = 0
    MODE = 1
    IF (JSTART(NENT).EQ.0) JSTART(NENT)=COL
C
C CONTINUE INTEGER
C
190 INT = INT*10 + IDIGIT(COL) * ISIG
    GOTO 40
C
C CHANGE TO REAL
C

```



```

200 STATE = 4
    MODE = 2
    FLT = INT
        GOTO 40
C
C CONTINUE REAL
C
210 FLT = FLT + FLOAT(IDIGIT(COL)*ISIG)*F
    F = F * 0.1
        GOTO 40
C
C START EXPONENT
C
220 STATE = 5
        GOTO 40
C
C MINUS EXP
C
230 NSIG = -1
C
C PLUS EXP
C
240 STATE = 6
        GOTO 40
C
C CONTINUE EXP
C
250 EXP = EXP * 10 + IDIGIT(COL)*NSIG
    STATE = 6
        GOTO 40
C
C START LABEL
C
260 STATE = 7
    NDOT = 0
    MODE = 3
        GOTO 310
C
C START NAME
C
270 STATE = 8
    NDOT = NDOT + 1
    NCHAR = NDOT
    MODE = 4
    JSTART(NENT+NDOT) = COL+1
        GOTO 20
C
C ANY TEXT
C
280 MODE = 5
    STATE = 10
    ISIG = 1
    INT = 0
    EXP = 0
    F = 0.1
    NSIG = 1
        GOTO 40
C
C SEPARATER
C

```

```

290 INT = ITAB(JBUFF(COL)+1)
    IVALUE = INT
    MODE = 6
    JUMP = 34
    JSTART(NENT) = COL
    ISIG = 1
    GOTO 50
C
C START STRING
C
300 STATE = 9
    MODE = 7
305 JSTART(NENT) = COL
    GOTO 20
C
C SET BEGINNING COL NUMBER
C
310 JSTART(NENT) = COL
    IF(MODE.EQ.6.AND.SIGNED)GO TO 290
    GOTO 40
C
C START REAL
C
320 JSTART(NENT) = COL
    INT = 0
    GOTO 200
C
C END LINE AND END STRING
C
330 SKIP=COL
    GOTO 80
C
C START SIGNED REAL
C
350 INT=0
    GOTO 200
    END
C *****
C *
C * RDLINE
C *
C *****
C SUBROUTINE RDLINE
C
C READ A CARD
C
COMMON /SCANLN/ COL, JUMP, NENT, PSTATE, SKIP, DOREAD,
1 INCOL, JSTART(80), Ibuff(81), JBUFF(81),
2 IDIGIT(81), CARD(81)
INTEGER COL, SKIP, PSTATE
LOGICAL DOREAD
JUMP = 1
CALL SCANRD
RETURN
END
C *****
C *
C * SCANMS
C *
C *****

```

```

SUBROUTINE SCANMS( ERRNO )
C
C      ISSUE ERROR MESSAGES
C
COMMON /SCANIO/ INUNIT, IOUT, FILES(10), FILPT, FILLIM, LREMOT
INTEGER ERRNO, FILES, FILPT, FILLIM
C
C      SEND ERROR MESSAGE AND RETURN
C
GO TO ( 10, 20, 30 ), ERRNO
10 WRITE(IOUT,1001)
GO TO 9999
20 WRITE(IOUT,1002)
GO TO 9999
30 WRITE(IOUT,1003)
GO TO 9999
9999 RETURN
C
1001 FORMAT(40H0 .....END OF FILE - PROGRAM TERMINATED /)
1002 FORMAT(40H0 .....INPUT ERROR - PROGRAM TERMINATED /)
1003 FORMAT(40H0 .....FILE LIST OVERFLOW /)
END
C *****
C *
C * SETIN
C *
C *****
C      SUBROUTINE SETIN(IN)
C
C      SET SCAN INPUT UNIT
C
COMMON /SCANIO/ INUNIT, IOUT, FILES(10), FILPT, FILLIM, LREMOT
INTEGER IN, FILES, FILPT, FILLIM
INUNIT = IN
RETURN
END
C *****
C *
C * SETOUT
C *
C *****
C      SUBROUTINE SETOUT(OUT)
C
C      SET SCAN ECHO OUTPUT UNIT
C
COMMON /SCANIO/ INUNIT, IOUT, FILES(10), FILPT, FILLIM, LREMOT
INTEGER OUT, FILES, FILPT, FILLIM
IOUT = OUT
RETURN
END
C *****
C *
C * SETREM
C *
C *****
C      SUBROUTINE SETREM(IUNIT)
C
C      SET THE REMOTE UNIT FOR PROMPTING
C
COMMON /SCANIO/ INUNIT, IOUT, FILES(10), FILPT, FILLIM, LREMOT

```

```

    INTEGER IUNIT, FILES, FILPT, FILLIM
    LREMOT = IUNIT
    RETURN
    END
C *****
C *
C * SETFIL
C *
C *****
C     SUBROUTINE SETFIL(FILIST,NFILES)
C
C     STACK A LIST OF INPUT FILES
C
C     COMMON /SCANIO/ IUNIT, IOUT, FILES(10), FILPT, FILLIM, LREMOT
C     INTEGER J, K, NFIL, NFILES, FILIST(10), FILES, FILPT, FILLIM
C     NFIL = NFILES
C     IF(FILPT+NFIL.LE.FILLIM) GO TO 10
C     NFIL=FILLIM-FILPT
C     CALL SCANMS(3)
C 10 J=FILPT+NFIL+1
C     IF(NFIL.EQ.0)GO TO 30
C     DO 20 K=1,NFIL
C 20 FILES(J-K)=FILIST(K)
C     FILPT=FILPT+NFIL
C     IUNIT=FILES(FILPT)
C 30 RETURN
C     END
C *****
C *
C * SCINIT
C *
C *****
C     SUBROUTINE SCINIT(NBLANK, RECLEN, ENDCHR, PROMSW, ECHOSW, COMSW,
C     1     ATRDSW, EOLSW, EOF SW, MENU SW, PT SW, SIGN SW )
C
C     INITIALIZE
C
C     COMMON /SCANCT/ ECHAR, ECHO, I LABEL, LIMIT, MARK, PROMT,
C     1     POINT, RECSIZ, INIT, LE OF, EOL, MENU,
C     2     AUTORD, COM MNT, SIGNED
C     INTEGER RECSIZ, RECLEN
C     LOGICAL ECHO, PROMT, POINT, INIT, LE OF, EOL, MENU,
C     1     COM MNT, AUTORD, SIGNED
C     LOGICAL PROMSW, ECHOSW, COMSW, ATRDSW, EOLSW, EOF SW, PT SW,
C     1     MENU SW, SIGN SW
C     INIT = .TRUE.
C     ECHAR = ENDCHR
C     ECHO = ECHOSW
C     LE OF = EOF SW
C     EOL = EOLSW
C     MENU = MENU SW
C     AUTORD = ATRDSW
C     COM MNT = .NOT.COMSW
C     PROMT = PROMSW
C     POINT = PT SW
C     SIGNED = SIGN SW
C     LIMIT = MINO(RECSIZ,NBLANK)
C     MARK = MINO(RECSIZ,RECLEN)
C     RETURN

```

```

      END
C *****
C *
C * ADDLAB
C *
C *****
      SUBROUTINE ADDLAB(LABEL)
C
C      ADD LABEL FOR SCAN ECHO
C
      COMMON /SCANCT/ ECHAR, ECHO, ILABEL, LIMIT, MARK, PROMT,
1          POINT, RECSIZ, INIT, LEOF, EOL, MENU,
2          AUTORD, COMMNT, SIGNED
      INTEGER RECSIZ, LABEL
      LOGICAL ECHO, PROMT, POINT, INIT, LEOF, EOL, MENU,
1          COMMNT, AUTORD, SIGNED
      ILABEL = LABEL
      RETURN
      END
C *****
C *
C * BACKSP
C *
C *****
      SUBROUTINE BACKSP(NUMENT)
C
C      BACKSPACE THE SCANNER
C
      COMMON /SCANLN/ COL, JUMP, NENT, PSTATE, SKIP, DOREAD,
1          INCOL, JSTART(80), Ibuff(81), Jbuff(81),
2          IDIGIT(81), CARD(81)
      INTEGER COL, SKIP, PSTATE, NUMENT
      LOGICAL DOREAD
      IF(NENT.LT.NUMENT) GO TO 10
      NENT = NENT - NUMENT
      COL = JSTART(NENT + 1)
      JUMP = 3
      GO TO 20
10 COL = 0
      JUMP = 2
20 RETURN
      END
C *****
C *
C * GETSTR
C *
C *****
      SUBROUTINE GETSTR
C
C      PARSE A STRING
C
      COMMON /SCANLN/ COL, JUMP, NENT, PSTATE, SKIP, DOREAD,
1          INCOL, JSTART(80), Ibuff(81), Jbuff(81),
2          IDIGIT(81), CARD(81)
      INTEGER COL, SKIP, PSTATE
      LOGICAL DOREAD
      COL = JSTART(NENT)
      PSTATE = 1
      JUMP = 2
      RETURN

```

```

      END
C *****
C *
C * RESET
C *
C *****
      SUBROUTINE RESET
C
      START LINE OVER
C
      COMMON /SCANIM/ NCPW,BLANK,INTZER,INTNIN,INTC,INTCOM,INTBLK
      COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
      COMMON /SCANLN/ COL, JUMP, NENT, PSTATE, SKIP, DOREAD,
1          INCOL, JSTART(80), IBUFF(81), JBUFF(81),
2          IDIGIT(81), CARD(81)
      EQUIVALENCE (IVAL(1),VALUE,IVALUE)
      INTEGER COL, SKIP, PSTATE
      LOGICAL NEXT, DOREAD
      COL = 0
      JUMP = 2
      NENT = 0
      ENTITY(1) = BLANK
      RETURN
      END
C *****
C *
C * SKPSTR
C *
C *****
      SUBROUTINE SKPSTR
C
      SKIP TO END OF STRING
C
      COMMON /SCANLN/ COL, JUMP, NENT, PSTATE, SKIP, DOREAD,
1          INCOL, JSTART(80), IBUFF(81), JBUFF(81),
2          IDIGIT(81), CARD(81)
      INTEGER COL, SKIP, PSTATE
      LOGICAL DOREAD
      COL = SKIP
      RETURN
      END
C *****
C *
C * SEPTAB
C *
C *****
      SUBROUTINE SEPTAB(NEWTAB)
C
      CHANGE SEPERATOR TABLE
C
      INTEGER I, NEWTAB(1)
      COMMON /SCANTB/ NSEPTB, NCLASS, ITAB(256), ICLASS(256)
      DO 10 I = 1, NSEPTB
10 ITAB(I) = NEWTAB(I)
      RETURN
      END
C *****
C *
C * DOSCAN
C *

```

```

C *****
C LOGICAL FUNCTION DOSCAN(DUMMY)
C
C ADVANCE THE SCANNER NOW
C
COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
EQUIVALENCE (IVAL(1),VALUE,IVALUE)
INTEGER DUMMY
LOGICAL NEXT
IF(NEXT)CALL SCAN
NEXT = .FALSE.
DOSCAN = .TRUE.
RETURN
END
C *****
C *
C * ENDCRD
C *
C *****
C LOGICAL FUNCTION ENDCRD(DUMMY)
C
C IS THE CURRENT SCAN ENTITY AN END OF LINE
C
COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
EQUIVALENCE (IVAL(1),VALUE,IVALUE)
INTEGER MODEOL, DUMMY
LOGICAL NEXT
DATA MODEOL/9/
IF(.NOT.NEXT)GO TO 10
CALL SCAN
NEXT = .FALSE.
10 ENDCRD = .FALSE.
IF(MODE.NE.MODEOL)GO TO 20
ENDCRD = .TRUE.
NEXT = .TRUE.
20 RETURN
END
C *****
C *
C * ENDFIL
C *
C *****
C LOGICAL FUNCTION ENDFIL(DUMMY)
C
C IS THE CURRENT SCAN ENTITY AN END OF FILE
C
COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
EQUIVALENCE (IVAL(1),VALUE,IVALUE)
LOGICAL NEXT
INTEGER MODEOF, DUMMY
DATA MODEOF/10/
IF(.NOT.NEXT)GO TO 10
CALL SCAN
NEXT = .FALSE.
10 ENDFIL = .FALSE.
IF(MODE.NE.IABS(MODEOF))GO TO 20
ENDFIL = .TRUE.
NEXT = .TRUE.
20 RETURN
END

```

```

C *****
C *
C * ENTIT
C *
C *****
C     SUBROUTINE ENTIT(TEXT,NC,NW)
C
C         RETURN THE CURRENT CHARACTERS FROM ENTITY
C
C     COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
C     EQUIVALENCE (IVAL(1),VALUE,IVALUE)
C     DIMENSION TEXT(1)
C     INTEGER I, NC, NW
C     LOGICAL NEXT
C     NC = NCHAR
C     NW = NWD
C     DO 10 I = 1, NW
C 10 TEXT(I) = ENTITY(I)
C     RETURN
C     END
C *****
C *
C * INTEGR
C *
C *****
C     LOGICAL FUNCTION INTEGR(INTEG)
C
C         IS THE CURRENT SCAN ENTITY AN INTEGER
C
C     COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
C     EQUIVALENCE (IVAL(1),VALUE,IVALUE)
C     INTEGER MODINT, INTEG
C     LOGICAL NEXT
C     DATA MODINT/1/
C     IF(.NOT.NEXT)GO TO 10
C     CALL SCAN
C     NEXT = .FALSE.
C 10 INTEGR = .FALSE.
C     IF(MODE.NE.MODINT)GO TO 20
C     INTEG = IVALUE
C     INTEGR = .TRUE.
C     NEXT = .TRUE.
C 20 RETURN
C     END
C *****
C *
C * LABEL
C *
C *****
C     LOGICAL FUNCTION LABEL(DUMMY)
C
C         IS THE CURRENT SCAN ENTITY A LABEL
C
C     COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
C     EQUIVALENCE (IVAL(1),VALUE,IVALUE)
C     INTEGER MODLAB, DUMMY
C     LOGICAL NEXT
C     DATA MODLAB/3/
C     IF(.NOT.NEXT)GO TO 10
C     CALL SCAN

```



```

NEXT = .FALSE.
10 LABEL = .FALSE.
   IF(MODE.NE.MODLAB)GO TO 20
   LABEL = .TRUE.
   NEXT = .TRUE.
20 RETURN
END
C *****
C *
C * MATCH
C *
C *****
C LOGICAL FUNCTION MATCH(STRING,NC)
C
C MATCH A CHARACTER ARRAY AGAINST THE CURRENT SCAN ENTITY
C
COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
EQUIVALENCE (IVAL(1),VALUE,IVALUE)
DIMENSION STRING(1)
INTEGER NC
LOGICAL NEXT, SCANMC
MATCH = .FALSE.

IXT = .TRUE.
20 RETURN
END
C ***** *
C *
C ***** *
C LOGICAL FUNCTION NAME(DUMMY)
C
C IS THE CURRENT SCAN ENTITY A NAME
C
COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
EQUIVALENCE (IVAL(1),VALUE,IVALUE)
INTEGER MODNAM, DUMMY
LOGICAL NEXT
DATA MODNAM/4/
IF(.NOT.NEXT)GO TO 10
CALL SCAN
NEXT = .FALSE.
10 NAME = .FALSE.
   IF(MODE.NE.MODNAM)GO TO 20
   NAME = .TRUE.
   NEXT = .TRUE.
20 RETURN
END
C *****
C *
C * NOSCAN
C *
C *****
C LOGICAL FUNCTION NOSCAN(DUMMY)

```

```

C
C      DO NOT ADVANCE THE SCANNER
C
COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
EQUIVALENCE (IVAL(1),VALUE,IVALUE)
INTEGER DUMMY
LOGICAL NEXT
NEXT = .FALSE.
NOSCAN = .TRUE.
RETURN
END
C *****
C *
C * NUM1
C *
C *****
      LOGICAL FUNCTION NUM1(INTEGR)
C
C      IS THE CURRENT SCAN ENTITY A NUMBER, CONVERT TO INTEGER
C
COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
EQUIVALENCE (IVAL(1),VALUE,IVALUE)
INTEGER MODREL, INTEGR
LOGICAL NEXT
DATA MODREL/2/
IF(.NOT.NEXT)GO TO 10
CALL SCAN
NEXT = .FALSE.
10 NUM1 = .FALSE.
IF (MODE .GT. MODREL) GO TO 20
IF (MODE .EQ. MODREL ) INTEGR = VALUE
IF (MODE .NE. MODREL ) INTEGR = IVALUE
NUM1 = .TRUE.
NEXT = .TRUE.
20 RETURN
END
C *****
C *
C * NUMR
C *
C *****
      LOGICAL FUNCTION NUMR(REAL)
C
C      IS THE CURRENT SCAN ENTITY A NUMBER, CONVERT TO REAL
C
COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
EQUIVALENCE (IVAL(1),VALUE,IVALUE)
REAL REAL
INTEGER MODREL, MODINT
LOGICAL NEXT
DATA MODREL/2/, MODINT/1/
IF(.NOT.NEXT)GO TO 10
CALL SCAN
10 NEXT = .FALSE.
NUMR = .FALSE.
IF (MODE .GT. MODREL) GO TO 20
IF (MODE .EQ. MODINT ) REAL = IVALUE
IF (MODE .NE. MODINT ) REAL = VALUE
NUMR = .TRUE.
NEXT = .TRUE.

```

```

20 RETURN
END
C *****
C *
C * READSC
C *
C *****
C SUBROUTINE READSC
C
C ADVANCE THE SCANNER NOW
C
COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
COMMON /SCANCT/ ECHAR, ECHO, I LABEL, LIMIT, MARK, PROMT,
1 POINT, RECSIZ, INIT, LEOF, EOL, MENU,
2 AUTORD, COMMNT, SIGNED
EQUIVALENCE (IVAL(1),VALUE,IVALUE)
INTEGER RECSIZ, MODEOL
LOGICAL ECHO, PROMT, POINT, INIT, LEOF, EOL, MENU,
1 COMMNT, AUTORD, SIGNED, NEXT
DATA MODEOL/9/
IF(.NOT.AUTORD.OR.(AUTORD.AND.MODE.NE.MODEOL))CALL RDLINE
NEXT = .TRUE.
RETURN
END
C *****
C *
C * REALN
C *
C *****
C LOGICAL FUNCTION REALN(REAL)
C
C IS THE CURRENT SCAN ENTITY A REAL
C
COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
EQUIVALENCE (IVAL(1),VALUE,IVALUE)
LOGICAL NEXT
REAL REAL
INTEGER MODREL
DATA MODREL/2/
IF(.NOT.NEXT)GO TO 10
CALL SCAN
NEXT = .FALSE.
10 REALN = .FALSE.
IF(MODE.NE.MODREL)GO TO 20
REAL = VALUE
REALN = .TRUE.
NEXT = .TRUE.
20 RETURN
END
C *****
C *
C * SEP
C *
C *****
C LOGICAL FUNCTION SEP(SEPTYP)
C
C IS THE CURRENT SCAN ENTITY A SEPERATOR
C
COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
EQUIVALENCE (IVAL(1),VALUE,IVALUE)

```

```

    INTEGER SEPTYP, MODSEP
    LOGICAL NEXT
    DATA MODSEP/6/
    IF(.NOT.NEXT)GO TO 10
    CALL SCAN
    NEXT = .FALSE.
10  SEP = .FALSE.
    IF(MODE.NE.MODSEP)GO TO 20
    SEPTYP = IVALUE
    SEP = .TRUE.
    NEXT = .TRUE.
20  RETURN
    END
C *****
C *
C * STRING
C *
C *****
C     LOGICAL FUNCTION STRING(DUMMY)
C
C     IS THE CURRENT SCAN ENTITY A STRING
C
C     COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
C     EQUIVALENCE (IVAL(1),VALUE,IVALUE)
C     INTEGER MODSTR, DUMMY
C     LOGICAL NEXT
C     DATA MODSTR/7/
C     IF(.NOT.NEXT)GO TO 10
C     CALL SCAN
C     NEXT = .FALSE.
10  STRING = .FALSE.
    IF(MODE.NE.MODSTR)GO TO 20
    STRING = .TRUE.
    NEXT = .TRUE.
20  RETURN
    END
C *****
C *
C * TRUE
C *
C *****
C     LOGICAL FUNCTION TRUE(DUMMY)
C
C     ADVANCE THE SCANNER BEFORE THE NEXT TEST
C
C     COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
C     EQUIVALENCE (IVAL(1),VALUE,IVALUE)
C     INTEGER DUMMY
C     LOGICAL NEXT
C     NEXT = .TRUE.
C     TRUE = .TRUE.
C     RETURN
C     END
C *****
C *
C * SCANCL
C *
C *****
C     SUBROUTINE SCANCL
C

```

```

C      CLASSIFY THE INPUT CHARACTERS
C
COMMON /SCANCT/ ECHAR, ECHO, ILABEL, LIMIT, MARK, PROMT,
1          POINT, RECSIZ, INIT, LEOF, EOL, MENU,
2          AUTORD, COMMNT, SIGNED
COMMON /SCANLN/ COL, JUMP, NENT, PSTATE, SKIP, DOREAD,
1          INCOL, JSTART(80), Ibuff(81), Jbuff(81),
2          IDIGIT(81), CARD(81)
COMMON /SCANTB/ NSEPTB, NCLASS, ITAB(256), ICLASS(256)
COMMON /SCANIM/ NCPW, BLANK, INTZER, INTNIN, INTC, INTCOM, INTBLK .
INTEGER I, NN, J, K, RECSIZ, COL, SKIP, PSTATE
LOGICAL ECHO, PROMT, POINT, INIT, LEOF, EOL, MENU,
1          COMMNT, AUTORD, SIGNED, DOREAD
CHARACTER*1 XCARD(320), Xbuff(320)
C      EQUIVALENCE ( XCARD(1), CARD(1) ), ( Xbuff(1), Jbuff(1) )
EQUIVALENCE ( XCARD(1), CARD(1) )
J = 1
K = NCPW
NN = MINO(INCOL, MARK)
NN = NN+1
CARD(NN) = ECHAR
DO 10 I = 1, NN
C      Xbuff(K) = XCARD(J)
Jbuff(I) = ICHAR(XCARD(J))
IDIGIT(I) = -1
IF(Jbuff(I).GE.INTZER.AND.Jbuff(I).LE.INTNIN)
1  IDIGIT(I) = Jbuff(I)-INTZER
J = J+NCPW
K = K+NCPW
Ibuff(I) = ICLASS(Jbuff(I)+1)
IF(CARD(I).EQ.ECHAR)Ibuff(I) = 10
10 CONTINUE
RETURN
END
C *****
C *
C * SCANIN
C *
C *****
C      SUBROUTINE SCANIN(IN, BUFF, RECLN, ERRCOD, IOUT, ECHO)
C
C      INPUT A RECORD FROM A DEVICE
C
COMMON /SCANIM/ NCPW, BLANK, INTZER, INTNIN, INTC, INTCOM, INTBLK
COMMON /SCANLN/ COL, JUMP, NENT, PSTATE, SKIP, DOREAD,
1          INCOL, JSTART(80), Ibuff(81), Jbuff(81),
2          IDIGIT(81), CARD(81)
INTEGER IN, RECLN, ERRCOD, IOUT
INTEGER COL, SKIP, PSTATE
INTEGER I, IDUM
REAL*4 BUFF(1)
LOGICAL DOREAD, ECHO
ERRCOD = 0
IDUM = RECLN
C      CALL SCANDB
READ(IN, 1001, ERR = 20, END = 30)(BUFF(I), I = 1, RECLN)
DO 10 I = 1, RECLN
IF(.NOT.(BUFF(IDUM).EQ.BLANK))GOTO 40
IDUM = IDUM-1
10 CONTINUE

```

```

                IF(ECHO)WRITE(1OUT,1002)
                GOTO 1
20             ERRCOD= 1
                GOTO 40
30             ERRCOD= -1
40             RECLEN=MAX0(IDUM,2)
                RETURN
1001          FORMAT(80A1)
1002          FORMAT(/)
                END
C *****
C *
C * SCANPK
C *
C *****
                SUBROUTINE SCANPK
C
C             PACK THE SCANNER RESULT IN ENTITY AT 4 CHARACTERS PER WORD
C
                COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
                COMMON /SCANLN/ COL, JUMP, NENT, PSTATE, SKIP, DOREAD,
1                 INCOL, JSTART(80), IBUFF(81), JBUFF(81),
2                 IDIGIT(81), CARD(81)
                COMMON /SCANIM/ NCPW,BLANK,INTZER,INTNIN,INTC,INTCOM,INTBLK
                CHARACTER*1 XENTIT(80), XCARD(320)
                EQUIVALENCE ( XENTIT(1), ENTITY(1) ), ( XCARD(1), CARD(1) )
                EQUIVALENCE (IVAL(1),VALUE,IVALUE)
                INTEGER I, ISTART, COL, SKIP, PSTATE
                LOGICAL NEXT, DOREAD
                NWD = (NCHAR+NCPW-1)/NCPW
                IF(NWD.LE.0)NWD = 1
                ENTITY(NWD) = BLANK
                IF(NWD.LT.20)ENTITY(NWD+1) = BLANK
                ISTART = (ICOLMN-1)*NCPW+1
                IF(NCHAR.EQ.0)NWD = 0
                IF(NCHAR.EQ.0)GO TO 20
                IF(MODE.EQ.7)ISTART = ISTART+NCPW
                DO 10 I = 1, NCHAR
                XENTIT(I) = XCARD(ISTART)
                ISTART = ISTART+NCPW
                IF(MODE.EQ.4)ISTART = (JSTART(NENT+1)-1)*NCPW+1
10             CONTINUE
20             RETURN
                END
C *****
C *
C * SCANRD
C *
C *****
                SUBROUTINE SCANRD
C
C             INPUT FOR SCAN
C
                COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
                COMMON /SCANCT/ ECHAR, ECHO, ILABEL, LIMIT, MARK, PROMT,
1                 POINT, RECSIZ, INIT, LEOF, EOL, MENU,
2                 AUTORD, COMMNT, SIGNED
                COMMON /SCANIO/ INUNIT, IOUT, FILES(10), FILPT, FILLIM, LREMOT
                COMMON /SCANLN/ COL, JUMP, NENT, PSTATE, SKIP, DOREAD,
1                 INCOL, JSTART(80), IBUFF(81), JBUFF(81),

```

```

2          IDIGIT(81), CARD(81)
COMMON /SCANIM/ NCPW,BLANK,INTZER,INTNIN,INTC,INTCOM,INTBLK
EQUIVALENCE (IVAL(1),VALUE,IVALUE)
INTEGER I, NN, IERR, RECSIZ, FILES, FILPT, FILLIM
INTEGER COL, SKIP, PSTATE
LOGICAL ECHO, PROMT, POINT, INIT, LEOF, EOL, MENU,
1          COMMNT, AUTORD, SIGNED, NEXT, DOREAD, REREAD

C
C          READ A RECORD
C
DOREAD = .FALSE.
10  IF(PROMT)WRITE(LREMOT,1003)
REREAD = .FALSE.
INCOL = RECSIZ
C          CALL SCANDB
CALL SCANIN (INUNIT,CARD,INCOL,IERR,IOUT,ECHO)
IF(IERR)20,40,30

C
C EOF,ERR
C
20  MODE=10
      GOTO 100
30  CALL SCANMS(2)
      CALL EXIT

C
C CLASSIFY
C
40  CALL SCANCL
      MODE = 0

C
C COMMENT 'C ' IN CC 1&2 = ZC3, Z40 = 195, 64
C
IF(JBUFF(1).NE.INTC.OR.JBUFF(2).NE.INTBLK)GO TO 50
IF(COMMNT)REREAD = .TRUE.
      GOTO 70

C
C POINT 'DDDD,DDDD' IN CC1-9 = (4@ZF0-F9),Z68,(4@ZF0-F9)
C                               = (4@Z40-249),107,(4@Z40-249)
C
50  IF(.NOT.POINT)GO TO 70
      IF(INCOL.NE.9)GO TO 70
      IF(JBUFF(5).NE.INTCOM)GO TO 70
      DO 60 I=1,4
          IF(IDIGIT(I).EQ.-1.OR.IDIGIT(I+5).EQ.-1)GOTO 70
60  CONTINUE
      MODE=11
      GOTO 70

C
C ECHO (NOT A POINT WITH MENUING)
C
70  IF(MODE.EQ.11.AND.MENU)GO TO 100
      IF(.NOT.ECHO)GO TO 90
      NN = MINO(INCOL,MARK)
      IF(ILABEL.EQ.0)GO TO 80
      WRITE(IOUT,1001)ILABEL,(CARD(I),I=1,NN)
      ILABEL = 0
      GOTO 90
80  WRITE(IOUT,1002)(CARD(I),I=1,NN)
90  IF(REREAD)GOTO 10
100 RETURN

```

```

1001 FORMAT(1X,15,1X,80A1)
1002 FORMAT(7X,      80A1)
1003 FORMAT(3H ? )
      END
C *****
C *
C * SCANMC
C *
C *****
      LOGICAL FUNCTION SCANMC(TEXTA,TEXTB,NCHAR).
C
C      CHARACTER MATCH OF A AND B FOR NCHARS
C
      COMMON /SCANIM/ NCPW,BLANK,INTZER,INTNIN,INTC,INTCOM,INTBLK
      DIMENSION TEXTA(1), TEXTB(1)
      REAL JA, JB, JC, JD
      CHARACTER*1 AA(4), BB(4), CC(4), DD(4)
      EQUIVALENCE (AA(1),JA), (BB(1),JB), (CC(1),JC), (DD(1),JD)
      INTEGER I, NW, NR
      SCANMC = .FALSE.
      NW = NCHAR/NCPW
      NR = NCHAR-NW*NCPW
      IF(NW.EQ.0)GO TO 20
      DO 10 I = 1, NW
      IF(TEXTA(I).NE.TEXTB(I))GO TO 50
10 CONTINUE
20 IF(NR.EQ.0)GO TO 40
      JA = TEXTA(NW+1)
      JB = TEXTB(NW+1)
      JC = BLANK
      JD = BLANK
      DO 30 I = 1, NR
      CC(4) = AA(1)
      DD(4) = BB(1)
      IF(JC.NE.JD)GO TO 50
30 CONTINUE
40 SCANMC = .TRUE.
50 RETURN
      END
C *****
C *
C * SCANDB
C *
C *****
      SUBROUTINE SCANDB
C
C      DEBUG - ALL CHARACTER OUTPUT AT 4 PER WORD
C
      COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
      COMMON /SCANCT/ ECHAR, ECHO, I LABEL, LIMIT, MARK, PROMT,
1 POINT, RECSIZ, INIT, LEOF, EOL, MENU,
2 AUTORD, COMMENT, SIGNED
      COMMON /SCANIO/ INUNIT, IOUT, FILES(10), FILPT, FILLIM, LREMOT
      COMMON /SCANLN/ COL, JUMP, NENT, PSTATE, SKIP,-DOREAD,
1 INCOL, JSTART(80), Ibuff(81), Jbuff(81),
2 IDIGIT(81), CARD(81)
      EQUIVALENCE (IVAL(1),VALUE,IVALUE)
      INTEGER RECSIZ, FILES, FILPT, FILLIM, COL, SKIP, PSTATE
      LOGICAL ECHO, PROMT, POINT, INIT, LEOF, EOL, MENU,
1 COMMENT, AUTORD, SIGNED, NEXT, DOREAD

```



```

WRITE( 6,901)
901 FORMAT(' A NEW STATEMENT')
C901 FORMAT ( 1H1,/, ' *****'/,
C 1 ' * S C A N D E B U G *'/,
C 2 ' *****'/)
WRITE(IOUT,1002)RECSIZ, LIMIT, MARK, ECHAR, ECHO, PROMT, ILABEL,
1 INIT, POINT, MENU, LEOF, EOL, AUTORD, COMMT,
2 SIGNED
WRITE(IOUT,1003)INUNIT, IOUT, LREMOT, FILPT, FILLIM, FILES
WRITE(IOUT,1004)COL, JUMP, NENT, PSTATE, SKIP, DOREAD, INCOL,
3 CARD, JBUFF, IBUFF, IDIGIT
WRITE(IOUT,1005)ENTITY, MODE, VALUE, IVALUE, IVAL, NCHAR, NWD,
1 NEXT, ICOLMN
RETURN
1002 FORMAT( 5X, 25HC O N T R O L S T A T E /
1 10X, 14HRECORD SIZE - ,110/
2 10X, 15HRECORD LIMIT - , 12/
3 10X, 13HRECORD END - , 12/
4 10X, 23HRECORD END CHARACTER - , A1/
5 10X, 14HECHO SWITCH - , L1/
6 10X, 16HPROMPT SWITCH - , L1/
7 10X, 13HLINE LABEL - , 15/
8 10X, 24HINITIALIZATION SWITCH - , L1/
9 10X, 15HPOINT SWITCH - , L1/
A 10X, 14HMENU SWITCH - , L1/
B 10X, 21HEND OF FILE SWITCH - , L1/
C 10X, 21HEND OF LINE SWITCH - , L1/
D 10X, 19HAUTO READ SWITCH - ,L1/
E 10X, 17HCOMMENT SWITCH - , L1/
F 10X, 14HSIGN SWITCH - , L1/)
1003 FORMAT( 5X, 17H I / O S T A T E /
1 10X, 13HINPUT UNIT - , 12/
2 10X, 14HOUTPUT UNIT - , 12/
3 10X, 14HREMOTE UNIT - , 12/
4 10X, 21HFILE STACK POINTER - , 12/
5 10X, 13HFILE LIMIT - , 12/
6 10X, 13HFILE STACK - , 10(12, 2X)/)
1004 FORMAT( 5X, 19H S C A N S T A T E /
1 10X, 9HCOLUMN - , 12/
2 10X, 7HJUMP - , 12/
3 10X, 16HENTITY NUMBER - , 12/
4 10X, 15HSTRING STATE - , 12/
5 10X, 13HSKIP STATE - , 12/
6 10X, 14HREAD SWITCH - , L1/
7 10X, 14HRECORD SIZE - , 12/
8 10X, 13HINPUT LINE - , 81A1/
9 10X, 18HINTERNAL RECORD - , 20(13,1X)/28X,20(13,1X)/
A 28X,20(13,1X)/28X,21(13,1X)/
B 10X, 8HCLASS - , 20(13,1X)/18X,20(13,1X)/18X,20(13,1X)/
C 18X,21(13,1X)/
D 10X, 9HDIGITS - , 20(13,1X)/19X,20(13,1X)/19X,20(13,1X)/
E 19X,21(13,1X)/)
C
C THE A4 FORMAT IS MACHINE DEPENDENT
C
1005 FORMAT( 5X, 11H E N T I T Y /
1 10X, 9HENTITY - , 20A4/
2 10X, 8HVALUE - , G20.10/
3 10X, 9HIVALUE - , 120/
4 10X, 7HIVAL - , 2120/

```

```
5      10X, 7HMODE - , 12/  
6      10X, 12HCHARACTER - , 12/  
7      10X, 8HWORDS - , 12/  
8      10X, 7HNEXT - , L1/  
9      10X, 9HCOLUMN - , 12//)  
END
```

```

$DEBUG
C *****
C *
C * BLOCK DATA
C *
C *****
      BLOCK DATA
C
C      INITIALIZE THE SCANNER
C
      COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
      COMMON /SCANCT/ ECHAR, ECHO, ILABEL, LIMIT, MARK, PROMT,
1      POINT, RECSIZ, INIT, LEOF, EOL, MENU,
2      AUTORD, COMMNT, SIGNED
      COMMON /SCANIO/ INUNIT, IOUT, FILES(10), FILPT, FILLIM, LREMOT
      COMMON /SCANLN/ COL, JUMP, NENT, PSTATE, SKIP, DOREAD,
1      INCOL, JSTART(80), Ibuff(81), Jbuff(81),
2      IDIGIT(81), CARD(81)
      COMMON /SCANTB/ NSEPTB, NCLASS, ITAB(256), ICLASS(256)
      COMMON /SCANIM/ NCPW,BLANK,INTZER,INTNIN,INTC,INTCOM,INTBLK
      EQUIVALENCE (IVAL(1),VALUE,IVALUE)
      EQUIVALENCE (BLANK,TORAK1),(ENTITY,TORAK2),(CARD,TORAK3),
1      (ECHAR,TORAK4)
      INTEGER RECSIZ, FILES, FILPT, FILLIM, COL, SKIP, PSTATE
      LOGICAL ECHO, PROMT, POINT, INIT, LEOF, EOL, MENU,
1      COMMNT, AUTORD, SIGNED, NEXT, DOREAD
      CHARACTER*4 TORAK1,TORAK2(20),TORAK3(81),TORAK4
C
C      /SCANNER/
C
      DATA TORAK2/20*'  ', IVAL/2*0/, MODE/0/, NCHAR/0/,
1      NWD/0/, NEXT/.FALSE./, ICOLMN/0/
C
C      /SCANCT/
C
      DATA TORAK4/'$ ', ECHO/.TRUE./, ILABEL/0/, LIMIT/80/, MARK/80/,
1      POINT/.FALSE./, RECSIZ/80/, INIT/.FALSE./, LEOF/.FALSE./,
2      EOL/.TRUE./, MENU/.FALSE./, AUTORD/.FALSE./,
3      COMMNT/.TRUE./, SIGNED/.FALSE./, PROMT/.FALSE./
C
C      /SCANIO/
C
      DATA INUNIT/5/, IOUT/6/, FILES/10*0/, FILPT/0/, FILLIM/10/,
1      LREMOT/6/
C
C      /SCANLN/
C
      DATA COL/0/,- JUMP/1/, NENT/1/, PSTATE/0/, SKIP/1/,
1      DOREAD/.TRUE./, INCOL/80/, JSTART/80*0/,
2      Ibuff/81*9/, Jbuff/81*9/, IDIGIT/81*0/,
3      TORAK3/81*'  '/
C
C      /SCANTB/
C
      DATA NSEPTB/256/, NCLASS/256/
      DATA ITAB/8*0,
2      8*0,
4      00,00,26,22,09,17,07,24, 04,11,10,05,16,03,02,15,
6      8*0,
8      23,00,00,00,00,00,00,00, 00,00,21,12,00,25,19,20,
8*0,

```

A	8*0,	00,00,00,01,00,08,13,00,
C	8*0,	8*0,
E	8*0,	00,00,00,00,06,00,00,00,
1	8*0,	8*0,
3	8*0,	8*0,
5	00,00,26,22,09,17,07,24,	04,11,10,05,16,03,02,15,
7	8*0,	00,00,21,12,00,25,19,20,
9	23,00,00,00,00,00,00,00,	8*0,
B	8*0,	00,00,00,01,00,08,13,00,
D	8*0,	8*0,
F	8*0,	00,00,00,00,06,00,00,00/

DATA ICLASS/8*8,	8*8,
2	8*8,
4	9,8,6,8,8,8,8,6,
6	8*1,
8	8,5,5,5,5,4,5,5,
A	8*5,
C	8,5,5,5,5,5,5,5,
E	8*5,
1	8*8,
3	8*8,
5	9,8,6,8,8,8,8,6,
7	8*1,
9	8,5,5,5,5,4,5,5,
B	8*5,
D	8,5,5,5,5,5,5,5,
F	8*5,

C  
C  
C  
  
C

/SCANIM/

DATA NCPW/4/, TORAK1/' ', INTZER/48/,  
1 INTNIN/57/, INTC/67/, INTCOM /44/, INTBLK/32/

END

```

$DEBUG
SUBROUTINE FIXCRD
C
C *****
C
C FIXCRD - FIX CARD
C
C THIS SUBROUTINE READS FROM THE SCAN INPUT FILE AND
C WRITES TO THE OUTPUT FILE
C
C -----
C
C CALLS TO : NONE
C
C -----
C
C VARIABLE DEFINITION
C
C PASSED : NONE
C
C RETURNED: NONE
C
C -----
C
C MODIFICATION LOG
C
C 06/11/1986 RAS INITIAL CODING
C
C *****
C
C CHARACTER*4 CARD(20)
C INTEGER IR, IW
C
10 READ(5,20,END=100) (CARD(IR),IR=1,20)
20 FORMAT(20A4)
WRITE(7,30)(CARD(IW),IW=1,20)
30 FORMAT(10X,20A4)
GOTO 10
100 CONTINUE
C
RETURN
END

```

```

$DEBUG
SUBROUTINE IZERO (A,N,VALUE)
C
C *****
C
C IZERO - INTEGER ZERO
C
C THIS SUBROUTINE INITIALIZES THE INTEGER ARRAY TO VALUE
C -----
C
C CALLS TO : NONE
C -----
C
C VARIABLE DEFINITION
C
C PASSED :
C
C VALUE - INTEGER VALUE
C N - SIZE OF ARRAY
C
C RETURNED :
C
C A - INTEGER ARRAY
C -----
C
C MODIFICATION LOG
C
C 06/11/1986 RAS INITIAL CODING
C *****
C
C INTEGER VALUE,A(N)
C INTEGER I,N
C DO 100 I=1,N
C A(I) = VALUE
100 CONTINUE
C RETURN
C END

```

```

$DEBUG
  SUBROUTINE ZERO (A,N,VALUE)
C
C *****
C
C   ZERO - REAL ZERO
C
C   THIS SUBROUTINE INITIALIZES THE REAL*4 ARRAY TO VALUE
C
C -----
C
C   CALLS TO : NONE
C
C -----
C
C   VARIABLE DEFINITION
C
C   PASSED :
C
C     VALUE - REAL*4 VALUE
C     N     - SIZE OF ARRAY
C
C   RETURNED :
C
C     A     - REAL*4 ARRAY
C
C -----
C
C   MODIFICATION LOG
C
C     06/11/1986  RAS  INITIAL CODING
C
C *****
C
C   REAL*4 VALUE,A(N)
C   INTEGER I,N
C     DO 100 I=1,N
C       A(I) = VALUE
100  CONTINUE
C   RETURN
C   END

```

```

$DEBUG
SUBROUTINE DZERO (A,N,VALUE)
C
C *****
C
C DZERO - DOUBLE PRECISION ZERO
C
C THIS SUBROUTINE INITIALIZES THE REAL*8 ARRAY TO VALUE
C
C -----
C
C CALLS TO : NONE
C
C -----
C
C VARIABLE DEFINITION
C
C PASSED :
C
C VALUE - REAL*8 VALUE
C N - SIZE OF ARRAY
C
C RETURNED :
C
C A - REAL*8 ARRAY
C
C -----
C
C MODIFICATION LOG
C
C 06/11/1986 RAS INITIAL CODING
C
C *****
C
REAL*8 VALUE,A(N)
INTEGER I,N
DO 100 I=1,N
A(I) = VALUE
100 CONTINUE
RETURN
END

```



```

$DEBUG
C
C *****
C THIS SUBROUTINE GETS THE STRUCTURE NUMBERS *
C *****
C
C SUBROUTINE GETNUM(NSTRUC, ISTRUC, MAXSTR, IMSG)
C
C *****
C
C GETNUM - GET STRUCTURE NUMBER
C
C THIS SUBROUTINE GETS THE STRUCTURE NUMBER
C
-----
C
C CALLS TO : READSC
C           REMARK
C           EXIT
C
-----
C
C VARIABLE DEFINITION
C
C PASSED :
C
C MAXSTR - MAXIMUM NUMBER OF STRUCTURES PERMITTED
C IMSG - ERROR MESSAGE FILE NUMBER
C
C RETURNED :
C
C NSTRUC - STRUCTURE NUMBER
C ISTRUC - NUMBER OF STRUCTURES TO BE RATED
C
-----
C
C MODIFICATION LOG
C
C 06/11/1986 RAS INITIAL CODING
C
C *****
C
C COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
C EQUIVALENCE (IVAL(1),VALUE,IVALUE)
C INTEGER IVALUE
C REAL*4 VALUE
C
C INTEGER DUMMY,INTEG
C INTEGER NSTRUC(1),IMSG,ISTRUC
C LOGICAL SEP,ENDCRD,INTEGR,NEXT
C
C ISTRUC=1
10 IF (.NOT. INTEGR(INTEG)) GOTO 8000
C NSTRUC(ISTRUC) = INTEG
C IF( ENDCRD(DUMMY)) RETURN
C IF(SEP (DUMMY)) GOTO 25
C GOTO 30
25 IF(DUMMY .NE. 3) GOTO 8001
C CALL READSC
30 ISTRUC= ISTRUC+1

```

```

                IF( ISTRUC .GT. MAXSTR)      GOTO 8002
                GOTO 10
C
C  ERROR MESSAGES
C
8000 CALL REMARK(IMG, 'ERROR - INCORRECT NUMERIC INTEGER INPUT.      ')
      GOTO 8990
8001 CALL REMARK(IMG, 'ERROR - ILLEGAL CONTINUATION SEPARATOR.      ')
      GOTO 8990
8002 CALL REMARK(IMG, 'ERROR - INCREASE MAXSTR TO:                    ')
      WRITE(IMG,8003)ISTRUC
8003 FORMAT(10X,110)
      GOTO 8990
8990 CALL EXIT
      END

```

\$DEBUG

SUBROUTINE GETGIR(GRDSPC, ISTRUC, IMSG)

C \*\*\*\*\*

C \*\*\*\*\*

C GETGIR - GET GIRDER SPACING

C THIS SUBROUTINE GETS THE GIRDER SPACING

C

C -----

C

C CALLS TO : READSC

C REMARK

C EXIT

C

C -----

C

C VARIABLE DEFINITION

C

C PASSED :

C ISTRUC - NUMBER OF STRUCTURES TO BE RATED

C IMSG - ERROR MESSAGE FILE NUMBER .

C

C RETURNED :

C

C GRDSPC - GIRDER SPACING

C

C -----

C

C MODIFICATION LOG

C

C 06/11/1986 RAS INITIAL CODING

C \*\*\*\*\*

C \*\*\*\*\*

C

C COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN

C EQUIVALENCE (IVAL(1),VALUE,IVALUE)

C INTEGER IVALUE

C REAL\*4 VALUE

C

C INTEGER DUMMY

C INTEGER IMSG,ISTRUC

C REAL\*4 REEL

C REAL\*8 GRDSPC(1)

C LOGICAL SEP,ENDCRD,NUMR,NEXT

C

C I=1

C 10 IF( .NOT. NUMR(REEL)) GOTO 8000

C GRDSPC(1) = DBLE(REEL)

C IF( ENDCRD(DUMMY)) GOTO 40

C IF(SEP (DUMMY)) GOTO 25

C GOTO 30

C 25 IF(DUMMY .NE. 3) GOTO 8001

C CALL READSC

C 30 I=I+1

C IF(I .GT. ISTRUC) GOTO 8002

C GOTO 10

C 40 IF(I .LT. ISTRUC) GOTO 8003

RETURN

C

C

ERROR MESSAGES

C

```
8000 CALL REMARK(1MSG,'ERROR - INCORRECT NUMERIC REAL INPUT.      ')
      GOTO 8990
8001 CALL REMARK(1MSG,'ERROR - ILLEGAL CONTINUATION SEPARATOR.    ')
      GOTO 8990
8002 CALL REMARK(1MSG,'ERROR - TOO MANY GIRDER SPACINGS ENTERED.  ')
      GOTO 8990
8003 CALL REMARK(1MSG,'ERROR - TOO FEW GIRDER SPACINGS ENTERED.  ')
      GOTO 8990
8990 CALL EXIT
      END
```

```

$DEBUG
SUBROUTINE GETOPE(OPRING,ISTRUC,IMSG)
C
C *****
C
C   GETOPE - GET OPERATING RATING
C
C   THIS SUBROUTINE GETS THE OPERATING RATING
C
C -----
C
C   CALLS TO :  READSC
C               REMARK
C               EXIT
C
C -----
C
C   VARIABLE DEFINITION
C
C   PASSED :
C
C       ISTRUC - NUMBER OF STRUCTURES TO BE RATED
C       IMSG   - ERROR MESSAGE FILE NUMBER
C
C   RETURNED :
C
C       OPRING - OPERATING RATING
C
C -----
C
C   MODIFICATION LOG
C
C       06/11/1986  RAS  INITIAL CODING
C
C *****
C
COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
EQUIVALENCE (IVAL(1),VALUE,IVALUE)
INTEGER IVALUE
REAL*4  VALUE
C
INTEGER DUMMY,INTEG
INTEGER OPRING(1),IMSG,ISTRUC
LOGICAL SEP,ENDCRD,INTEGR,NEXT
C
I=1
10  IF( .NOT. INTEGR(INTEG)) GOTO 8000
    OPRING(1) = INTEG
    IF( ENDCRD(DUMMY)) GOTO 40
    IF(SEP (DUMMY)) GOTO 25
    GOTO 30
25  IF(DUMMY .NE. 3) GOTO 8001
    CALL READSC
30  I=I+1
    IF(I .GT. ISTRUC) GOTO 8002
    GOTO 10
40  IF(I .LT. ISTRUC) GOTO 8003
    RETURN
C
C   ERROR MESSAGES

```

C  
8000 CALL REMARK(1MSG,'ERROR - INCORRECT NUMERIC INTEGER INPUT. ')  
GOTO 8990  
8001 CALL REMARK(1MSG,'ERROR - ILLEGAL CONTINUATION SEPARATOR. ')  
GOTO 8990  
8002 CALL REMARK(1MSG,'ERROR - TOO MANY OPERATING RATINGS ENTERED. ')  
GOTO 8990  
8003 CALL REMARK(1MSG,'ERROR - TOO FEW OPERATING RATINGS ENTERED. ')  
GOTO 8990  
8990 CALL EXIT  
END

```

$DEBUG
C
C *****
C THIS SUBROUTINE GETS THE SLAB PLAN NUMBERS *
C *****
C
C SUBROUTINE GETSLB(ISPLAN,ISLAB,ISTRUC,MSG)
C
C *****
C
C GETSLB - GET SLAB PLANS
C
C THIS SUBROUTINE GETS THE SLAB PLANS
C
C -----
C
C CALLS TO : READSC
C REMARK
C EXIT
C
C -----
C
C VARIABLE DEFINITION
C
C PASSED :
C
C ISTRUC - MAXIMUM NUMBER OF STRUCTURES ENTERED
C MSG - ERROR MESSAGE FILE NUMBER
C
C RETURNED :
C
C ISPLAN - SLAB PLAN NUMBER
C ISLAB - NUMBER OF SLAB PLANS INPUT
C
C -----
C
C MODIFICATION LOG
C
C 07/02/1986 .RAS INITIAL CODING
C
C *****
C
C COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
C EQUIVALENCE (IVAL(1),VALUE,IVALUE)
C INTEGER IVALUE
C REAL*4 VALUE
C
C INTEGER DUMMY,INTEG
C INTEGER ISPLAN(1),MSG,ISLAB,ISTRUC
C LOGICAL SEP,ENDCRD,INTEGR,NEXT
C
C ISLAB=1
10 IF( .NOT. INTEGR(INTEG)) GOTO 8000
   ISPLAN(ISLAB) = INTEG
   IF( ENDCRD(DUMMY)) RETURN
   IF(SEP (DUMMY)) GOTO 25
   GOTO 30
25 IF(DUMMY .NE. 3) GOTO 8001
   CALL READSC
30 ISLAB= ISLAB+1

```

```
      IF( ISLAB .GT. ISTRUC)      GOTO 8002
      GOTO 10
C
C   ERROR MESSAGES
C
8000 CALL REMARK(IMG, 'ERROR - INCORRECT NUMERIC INTEGER INPUT.      ')
      GOTO 8990
8001 CALL REMARK(IMG, 'ERROR - ILLEGAL CONTINUATION SEPARATOR.      ')
      GOTO 8990
8002 CALL REMARK(IMG, 'ERROR - TOO MANY SLAB PLANS ENTERED.          ')
      GOTO 8990
8990 CALL EXIT
      END
```



```

$DEBUG
SUBROUTINE GETCNT(CONTIN,ISTRUC,IMSG)
C
C *****
C
C GETCTN - GET CONTINUITIES
C
C THIS SUBROUTINE GETS THE CONTINUITIES
C
C -----
C
C CALLS TO : READSC
C            REMARK
C            EXIT
C
C -----
C
C VARIABLE DEFINITION
C
C PASSED :
C
C     ISTRUC - NUMBER OF STRUCTURES TO BE RATED
C     IMSG   - ERROR MESSAGE FILE NUMBER
C
C RETURNED :
C
C     CONTIN - CONTINUITY
C
C -----
C
C MODIFICATION LOG
C
C     06/11/1986  RAS  INITIAL CODING
C
C *****
C
C COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
C EQUIVALENCE (IVAL(1),VALUE,IVALUE)
C INTEGER IVALUE
C REAL*4 VALUE
C
C INTEGER DUMMY,INTEG
C INTEGER CONTIN(1),IMSG,ISTRUC
C LOGICAL SEP,ENDCRD,INTEGR,NEXT
C
C I=1
10 IF( .NOT. INTEGR(INTEG)) GOTO 8000
   CONTIN(1) = INTEG
   IF( ENDCRD(DUMMY)) GOTO 40
   IF(SEP (DUMMY)) GOTO 25
   GOTO 30
25 IF(DUMMY .NE. 3) GOTO 8001
   CALL READSC
30 I=I+1
   IF(I .GT. ISTRUC) GOTO 8002
   GOTO 10
40 IF(I .LT. ISTRUC) GOTO 8003
   RETURN
C
C ERROR MESSAGES

```

C  
8000 CALL REMARK(1MSG,'ERROR - INCORRECT NUMERIC INTEGER INPUT. ')  
GOTO 8990  
8001 CALL REMARK(1MSG,'ERROR - ILLEGAL CONTINUATION SEPARATOR. ')  
GOTO 8990  
8002 CALL REMARK(1MSG,'ERROR - TOO MANY CONTINUITY ITEMS ENTERED. ')  
GOTO 8990  
8003 CALL REMARK(1MSG,'ERROR - TOO FEW CONTINUITY ITMES ENTERED. ')  
GOTO 8990  
8990 CALL EXIT  
END

```

$DEBUG
  SUBROUTINE GETSTD(IOVLD,MSG)
C
C *****
C
C   GETSTD - GET STANDARD VEHICLES
C
C   THIS SUBROUTINE GETS THE STANDARD VEHICLES TO BE RATED
C
C -----
C
C   CALLS TO : READSC
C              REMARK
C              EXIT
C
C -----
C
C   VARIABLE DEFINITION
C
C   PASSED :
C
C       MSG - ERROR MESSAGE FILE NUMBER
C
C   RETURNED :
C
C       IOVLD - STANDARD VEHICLE IDENTIFICATION
C
C -----
C
C   MODIFICATION LOG
C
C       06/11/1986  RAS  INITIAL CODING
C
C *****
C
C   COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
C   EQUIVALENCE (IVAL(1),VALUE,IVALUE)
C   INTEGER IVALUE
C   REAL*4  VALUE .
C
C   INTEGER DUMMY
C   INTEGER MSG,IOVLD
C   LOGICAL SEP,ENDCRD,MATCH,NEXT
C
C   IF(MATCH('H20',3))      GOTO 110
C   IF(MATCH('HS20',4))     GOTO 120
C   IF(MATCH('ALTERNATE',3))GOTO 130
C   IF(MATCH('TYPE',3))     GOTO 140
C   IF(MATCH('P5',2))       GOTO 150
C   IF(MATCH('P7',2))       GOTO 160
C   IF(MATCH('P9',2))       GOTO 170
C   IF(MATCH('P11',3))      GOTO 180
C   IF(MATCH('P13',3))      GOTO 190
C   GOTO 8000
110 IOVLD=1
    RETURN
120 IOVLD=2
    RETURN
130 IF(.NOT. MATCH('MILITARY',3))GOTO 8001
    IOVLD=3

```

```

      RETURN
140  IF( MATCH('3S2',3))GOTO 145
      IF( MATCH('33' ,2))GOTO 146
      IF( MATCH('3'  ,1))GOTO 147
      GOTO 8002
145  IOVLD=5
      RETURN
146  IOVLD=6
      RETURN
147  IOVLD=4
      RETURN
150  IOVLD=7
      RETURN
160  IOVLD=8
      RETURN
170  IOVLD=9
      RETURN
180  IOVLD=10
      RETURN
190  IOVLD=11
      RETURN
C
C  ERROR MESSAGES
C
8000 CALL REMARK(IMG,'ERROR - INCORRECT STANDARD VEHICLE INPUT.  ')
      GOTO 8990
8001 CALL REMARK(IMG,'ERROR - THE WORD <MILITARY> IS MISSING.  ')
      GOTO 8990
8002 CALL REMARK(IMG,'ERROR - <3S2>, <33>, OR <3> IS MISSING.  ')
      GOTO 8990
8990 CALL EXIT
      END

```

```

$DEBUG
SUBROUTINE STAND(IOVLD,IGROUP,PL,DX,WHO,WHI,IFL,JFG,IMSG)
C *****
C
C STAND - STANDARD OVERLOAD/RATING VEHICLES
C
C THIS SUBROUTINE READS IN STANDARD VEHICLES TO BE RATED
C -----
C
C CALLS TO : NONE
C -----
C
C VARIABLE DEFINITION
C
C PASSED :
C
C IOVLD - OVERLOAD VEHICLE IDENTIFICATION
C
C RETURNED:
C
C IGROUP - GROUP IDENTIFICATION
C PL - AXLE LOADS
C DX - DISTANCES TO THE AXLES
C WHO - OUTER WHEEL GAGE
C WHI - INNER W-----
C -----
C
C MODIFICATION LOG
C
C 06/11/1986 RAS INITIAL CODING
C *****
C
C INTEGER IGROUP(1),IFL,JFG,IMSG,IOVLD,I
C REAL*8 PL(1),WHO(1),WHI(1),DX(1)
C INTEGER IGRPA(2),IGRPB(3),IGRPC(2),IGRPD(3),IGRPE(5)
C INTEGER IGRPF(6),IGRPG(3),IGRPH(4),IGRPI(5),IGRPJ(6)
C INTEGER IGRPK(7)
C REAL*8 TRUCKA(2),TRUCKB(3),TRUCKC(2),TRUCKD(3),TRUCKE(5)
C REAL*8 TRUCKF(6),TRUCKG(3),TRUCKH(4),TRUCKI(5),TRUCKJ(6)
C REAL*8 TRUCKK(7)
C REAL*8 DXA(2),DXB(3),DXC(2),DXD(3),DXE(5)
C REAL*8 DXF(6),DXG(3),DXH(4),DXI(5),DXJ(6)
C REAL*8 DXK(7)
C REAL*8 WHOA(2),WHOB(3),WHOC(2),WHOD(3),WHOE(5)
C REAL*8 WHOF(6),WHOG(3),WHOH(4),WHOI(5),WHOJ(6)
C REAL*8 WHOK(7)
C
C DATA IGRPA / 1, 2/
C DATA TRUCKA / 8.00, 32.00/
C DATA DXA / 0., 14.00/
C DATA WHOA / 6.0, 6.00/
C

```

```

DATA IGRPB / 1, 2, 3/
DATA TRUCKB / 8.00, 32.00, 32.00/
DATA DXB / 0. , 14.00, 28.00/
DATA WHOB / 6. , 6. , 6. /

C
DATA IGRPC / 1, 2/
DATA TRUCKC / 24.00, 24.00/
DATA DXC / 0. , 4.00/
DATA WHOC / 6.0, 6.00/

C
DATA IGRPD / 1, 2, 3/
DATA TRUCKD / 16.00, 17.00, 17.00/
DATA DXD / 0. , 15.00, 19.00/
DATA WHOD / 6. , 6. , 6. /

C
DATA IGRPE / 1, 2, 3, 4, 5/
DATA TRUCKE / 10.00, 15.50, 15.50, 15.50, 15.50/
DATA DXE / 0. , 11.00, 15.00, 37.00, 41.00/
DATA WHOE / 6. , 6. , 6. , 6. , 6. /

C
DATA IGRPF / 1, 2, 3, 4, 5, 6/
DATA TRUCKF / 12.00, 12.00, 12.00, 16.00, 14.00, 14.00/
DATA DXF / 0. , 15.00, 19.00, 34.00, 50.00, 54.00/
DATA WHOF / 6. , 6. , 6. , 6. , 6. , 6. /

C
DATA IGRPG / 1, 2, 3/
DATA TRUCKG / 26.00, 48.00, 48.00/
DATA DXG / 0. , 18.00, 36.00/
DATA WHOG / 6. , 6. , 6. /

C
DATA IGRPH / 1, 2, 3, 4/
DATA TRUCKH / 26.00, 48.00, 48.00, 48.00/
DATA DXH / 0. , 18.00, 36.00, 54.00/
DATA WHOH / 6. , 6. , 6. , 6. /

C
DATA IGRPI / 1, 2, 3, 4, 5/
DATA TRUCKI / 26.00, 48.00, 48.00, 48.00, 48.00/
DATA DXI / 0. , 18.00, 36.00, 54.00, 72.00/
DATA WHOI / 6. , 6. , 6. , 6. , 6. /

C
DATA IGRPJ / 1, 2, 3, 4, 5, 6/
DATA TRUCKJ / 26.00, 48.00, 48.00, 48.00, 48.00, 48.00/
DATA DXJ / 0. , 18.00, 36.00, 54.00, 72.00, 90.00/
DATA WHOJ / 6. , 6. , 6. , 6. , 6. , 6. /

C
DATA IGRPK / 1, 2, 3, 4, 5, 6, 7/
DATA TRUCKK / 26.00, 48.00, 48.00, 48.00, 48.00, 48.00, 48.00/
DATA DXK / 0. , 18.00, 36.00, 54.00, 72.00, 90.00, 108.00/
DATA WHOK / 6. , 6. , 6. , 6. , 6. , 6. , 6. /

C
GOTO(100,200,300,400,500,600,700,800,900,1000,1100)10VLD

C
C
C
C
100 IFL=2
JFG=2
DO 110 I=1,IFL
IGROUP(I) = IGRPA(I)
PL(I) = TRUCKA(I)
DX(I) = DXA(I)

```

```

        WHO(1) = WHOA(1)
        WHI(1) = 0.00
110 CONTINUE
RETURN
C
C   HS20
C
200 IFL=3
    JFG=3
    DO 210 I=1,IFL
        IGROUP(1) = IGRP(1)
        PL(1) = TRUCKB(1)
        DX(1) = DXB(1)
        WHO(1) = WHOB(1)
        WHI(1) = 0.00
210 CONTINUE
RETURN
C
C   ALTERNATE MILITARY
C
300 IFL=2
    JFG=2
    DO 310 I=1,IFL
        IGROUP(1) = IGRPC(1)
        PL(1) = TRUCKC(1)
        DX(1) = DXC(1)
        WHO(1) = WHOC(1)
        WHI(1) = 0.00
310 CONTINUE
RETURN
C
C   TYPE 3
C
400 IFL=3
    JFG=3
    DO 410 I=1,IFL
        IGROUP(1) = IGRPD(1)
        PL(1) = TRUCKD(1)
        DX(1) = DXD(1)
        WHO(1) = WHOD(1)
        WHI(1) = 0.00
410 CONTINUE
RETURN
C
C   TYPE 3-S2
C
500 IFL=5
    JFG=5
    DO 510 I=1,IFL
        IGROUP(1) = IGRPE(1)
        PL(1) = TRUCKE(1)
        DX(1) = DXE(1)
        WHO(1) = WHOE(1)
        WHI(1) = 0.00
510 CONTINUE
RETURN
C
C   TYPE 3-3
C
600 IFL=6

```

```

JFG=6
DO 610 I=1,IFL
  IGROUP(1) = IGRPF(1)
  PL(1)      = TRUCKF(1)
  DX(1)      = DXF(1)
  WHO(1)     = WHOF(1)
  WHI(1)     = 0.00
610 CONTINUE
RETURN
C
C   P5
C
700 IFL=3
    JFG=3
    DO 710 I=1,IFL
      IGROUP(1) = IGRPG(1)
      PL(1)      = TRUCKG(1)
      DX(1)      = DXG(1)
      WHO(1)     = WHOG(1)
      WHI(1)     = 0.00
710 CONTINUE
RETURN
C
C   P7
C
800 IFL=4
    JFG=4
    DO 810 I=1,IFL
      IGROUP(1) = IGRPH(1)
      PL(1)      = TRUCKH(1)
      DX(1)      = DXH(1)
      WHO(1)     = WHOH(1)
      WHI(1)     = 0.00
810 CONTINUE
RETURN
C
C   P9
C
900 IFL=5
    JFG=5
    DO 910 I=1,IFL
      IGROUP(1) = IGRPI(1)
      PL(1)      = TRUCKI(1)
      DX(1)      = DXI(1)
      WHO(1)     = WHOI(1)
      WHI(1)     = 0.00
910 CONTINUE
RETURN
C
C   P11
C
1000 IFL=6
     JFG=6
     DO 1010 I=1,IFL
       IGROUP(1) = IGRPJ(1)
       PL(1)      = TRUCKJ(1)
       DX(1)      = DXJ(1)
       WHO(1)     = WHOJ(1)
       WHI(1)     = 0.00
1010 CONTINUE

```



```
      RETURN
C
C      P31
C
1100  IFL=7
      JFG=7
      DO 1110 I=1,IFL
          IGROUP(I) = IGRP(K(I))
          PL(I)      = TRUCKK(I)
          DX(I)      = DXK(I)
          WHO(I)     = WHOK(I)
          WHI(I)     = 0.00
1110  CONTINUE
      RETURN
      END
```

```

$DEBUG
SUBROUTINE GETIND(PL,DX,IGROUP,IFL,JFG,MAXILD,IMSG)
C
C *****
C
C GETIND - GET INDIVIDUAL TRUCK CONFIGURATION
C
C THIS SUBROUTINE GETS THE INDIVIDUAL TRUCK CONFIGURATION
C
C -----
C
C CALLS TO : READSC
C             REMARK
C
C -----
C
C VARIABLE DEFINITION
C
C PASSED:
C
C     MAXILD - MAXIMUM AXLE LOADS PERMITTED
C     MSG    - ERROR MESSAGE FILE NUMBER
C
C RETURNED:
C
C     PL     - AXLE LOADS
C     DX     - DISTANCES TO THE AXLES
C     IGROUP - GROUP IDENTIFICATION
C     IFL    - NUMBER OF INDIVIDUAL AXLES
C     JFG    - NUMBER OF GROUP AXLES
C
C -----
C
C MODIFICATION LOG
C
C     06/11/1986  RAS  INITIAL CODING
C
C *****
C
C COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
C EQUIVALENCE (IVAL(1),VALUE,IVALUE)
C INTEGER IVALUE
C REAL*4  VALUE
C
C     INTEGER  ILD,JGP,DUMMY,MAXILD,IFL,JFG,IMSG
C     INTEGER  IGROUP(1)
C     REAL*4   REEL
C     REAL*8   DXTOT,PLOAD,AS
C     REAL*8   PL(1),DX(1)
C     LOGICAL  SEP,ENDCRD,NUMR,NEXT
C
C     ILD=1
C     JGP=1
C     DXTOT=0.0D0
C
C     SINGLE LOAD
C
C 15 IF( .NOT. NUMR (REEL) ) GOTO 8000
C     PLOAD= DBLE(REEL)
C     PL(ILD)= PLOAD

```

```

      IGROUP(ILD)=JGP
      DX(ILD)= DXTOT
      IFL=ILD
      JFG=JGP
      IF( ENDCRD(DUMMY)) RETURN
C
C   SINGLE AXLE SPACING
C
      IF( .NOT. NUMR (REEL) ) GOTO 8000
      AS=DBLE(REEL)
      DXTOT= DXTOT + AS
      IF(ENDCRD (DUMMY))      GOTO 8001
      IF( SEP(DUMMY) )GOTO 25
      GOTO 30
25  IF( DUMMY .NE. 3)      GOTO 8011
      CALL READSC
30  ILD=ILD+1
      JGP=JGP+1
      IF(ILD .GT. MAXILD)  GOTO 8009
      GOTO 15
C
8000 CALL REMARK(IMG,'ERROR - INCORRECT NUMERIC REAL INPUT.      ')
      GOTO 8990
8001 CALL REMARK(IMG,'ERROR - AXLE LOAD MISSING.                  ')
      GOTO 8990
8009 CALL REMARK(IMG,'ERROR - INCREASE MAXILD TO:                ')
      WRITE(IMG,8100)IFL
8100  FORMAT(10X,110)
      GOTO 8990
8011 CALL REMARK(IMG,'ERROR - ILLEGAL CONTINUATION SEPARATOR.    ')
      GOTO 8990
C
8990 CALL EXIT
      END

```

```

$DEBUG
SUBROUTINE GETGRP(PL,DX,IGROUP,IFL,JFG,MAXILD,IMSG)
C
C *****
C
C GETGRP - GET GROUP TRUCK CONFIGURATION
C
C THIS SUBROUTINE GETS THE INDIVIDUAL TRUCK CONFIGURATION
C
C -----
C
C CALLS TO : READSC
C             REMARK
C
C -----
C
C VARIABLE DEFINITION
C
C PASSED:
C
C     MAXILD - MAXIMUM AXLE LOADS PERMITTED
C     IMSG   - ERROR MESSAGE FILE NUMBER
C
C RETURNED:
C
C     PL     - AXLE LOADS
C     DX     - DISTANCES TO THE AXLES
C     IGROUP - GROUP IDENTIFICATION
C     IFL    - NUMBER OF INDIVIDUAL AXLES
C     JFG    - NUMBER OF GROUP AXLES
C
C -----
C
C MODIFICATION LOG
C
C     06/11/1986  RAS  INITIAL CODING
C     07/02/1986  RAS  POL MODIFICATION
C
C *****
C
C COMMON /SCANNER/ ENTITY(20),IVAL(2),MODE,NCHAR,NWD,NEXT,ICOLMN
C EQUIVALENCE (IVAL(1),VALUE,IVALUE)
C INTEGER IVALUE
C REAL*4  VALUE
C
C     INTEGER  ILD,JGP,INTEG,DUMMY,NAS,MAXILD,IFL,JFG,IMSG
C     INTEGER  IGROUP(1)
C     REAL*4   REEL
C     REAL*8   DXTOT,PLOAD,AS
C     REAL*8   PL(1),DX(1)
C     LOGICAL  SEP,ENDCRD,INTEGR,NUMR ,NEXT
C
C     ILD=1
C     JGP=1
C     IFL= ILD
C     JFG= JGP
C     DXTOT=0.000
C     NAS=0
C
C     GROUP AXLE LOAD

```

```

C
5  IF( .NOT. NUMR (REEL) ) GOTO 8000
   PLOAD= DBLE(REEL)
   IF( ENDCRD (DUMMY)) GOTO 10
   GOTO 20
10  PL(ILD) =PLOAD
   IGROUP(ILD) = JGP
   DX(ILD) =DXTOT
   RETURN

C
C  NUMBER AXLES PER GROUP
C
20  IF( .NOT. INTEGR(INTEG)) GOTO 8001
   NAS= INTEG
   IF( ENDCRD (DUMMY)) GOTO 8002

C
C  GROUP AXLE SPACING
C
   IF( .NOT. NUMR (REEL)) GOTO 8000
   AS=DBLE(REEL)
   IF( ENDCRD (DUMMY)) GOTO 40
   GOTO 75
40  IFL = ILD + NAS -1
   JFG = JGP
   IF(IFL .GT. MAXILD) GOTO 8009
   DO 45 I=ILD,IFL
     PL(I) = PLOAD
     IGROUP(I) = JGP
     DX(I) = DXTOT
     DXTOT = DXTOT + AS
45  CONTINUE
     DXTOT = DXTOT -AS
   RETURN

C
C  SPACING TO NEXT GROUP
C
75  IF(.NOT. NUMR (REEL)) GOTO 8000
   SNG= DBLE(REEL)
   IFL = ILD + NAS -1
   JFG = JGP
   IF(IFL .GT. MAXILD) GOTO 8009

C
   DO 100 I=ILD,IFL
     PL(I) = PLOAD
     IGROUP(I) = JGP
     DX(I) = DXTOT
     IF(IFL .EQ. 1) GOTO 100
     DXTOT = DXTOT + AS
100 CONTINUE
     DXTOT = DXTOT - AS
     DXTOT = DXTOT + SNG
     IF( SEP (DUMMY)) GOTO 120
     GOTO 8012
120 IF( DUMMY .NE. 3) GOTO 8011
     CALL READSC
     ILD = ILD + NAS
     JGP = JGP + 1
     IFL = ILD
     JFG = JGP
     IF(IFL .GT. MAXILD) GOTO 8009

```

```

      GOTO 5
C
8000 CALL REMARK(IMG, 'ERROR - INCORRECT NUMERIC REAL VALUE.      ')
      GOTO 8990
8001 CALL REMARK(IMG, 'ERROR - INCORRECT NUMERIC INTEGER VALUE.  ')
      GOTO 8990
8002 CALL REMARK(IMG, 'ERROR - AXLE SPACING MISSING.              ')
      GOTO 8990
8009 CALL REMARK(IMG, 'ERROR - INCREASE MAXILD TO:                ')
      WRITE(IMG, 8100) IFL
8100  FORMAT(10X, I10)
      GOTO 8990
8011 CALL REMARK(IMG, 'ERROR - ILLEGAL CONTINUATION SEPARATOR.    ')
      GOTO 8990
8012 CALL REMARK(IMG, 'ERROR - CONTINUATOR MISSING.                ')
      GOTO 8990
C
8990 CALL EXIT
      END

```



```

      GOTO 17
15  IF(DUMMY .NE. 3)   GOTO 8010
      CALL READSC
17  JGP = JGP +1
      ILD = ILD +1
      GOTO 10
C
C  GROUP WHEEL SPACINGS
C
40  IF(JGP .GT. JFG)   GOTO 8004
      IF( .NOT. NUMR (REEL)) GOTO 8005
      WHL(ILD) = DBLE(REEL)
45  IF(ILD .EQ. IFL) GOTO 50
      ICHCK1 = IGROUP(ILD)
      ICHCK2 = IGROUP(ILD+1)
      IF( ICHCK1 .NE. ICHCK2) GOTO 50
      WHL(ILD+1) = WHL(ILD)
      ILD = ILD + 1
      GOTO 45
50  IFWG = JGP
      IFWO = ILD
      IF (ENDCRD(DUMMY)) GOTO 400
      IF( SEP(DUMMY))   GOTO 100
      GOTO 150
100 IF(DUMMY .NE. 3 ) GOTO 8010
      CALL READSC
150 JGP=JGP + 1
      ILD=ILD + 1
      GOTO 40
C
300 IF( IFWO .NE. IFL)   GOTO 8008
      RETURN
400 IF( IFWG .NE. JFG)   GOTO 8009
      RETURN
C
8000 CALL REMARK(IMG, 'ERROR - SEPARATOR NOT ALLOWED.      ')
      GOTO 8990
8001 CALL REMARK(IMG, 'ERROR - TOO MANY SINGLE WHEEL SPACINGS. ')
      GOTO 8990
8002 CALL REMARK(IMG, 'ERROR - INCORRECT SINGLE WHEEL SPACING. ')
      GOTO 8990
8004 CALL REMARK(IMG, 'ERROR - TOO MANY GROUP WHEEL SPACINGS. ')
      GOTO 8990
8005 CALL REMARK(IMG, 'ERROR - INCORRECT GROUP WHEEL SPACING. ')
      GOTO 8990
8008 CALL REMARK(IMG, 'ERROR - INCORRECT NO. SINGLE WHEEL SPACINGS. ')
      GOTO 8990
8009 CALL REMARK(IMG, 'ERROR - INCORRECT NO. GROUP WHEEL SPACINGS. ')
      GOTO 8990
8010 CALL REMARK(IMG, 'ERROR - ILLEGAL CONTINUATOR.      ')
      GOTO 8990
C
8990 CALL EXIT
      END

```



```

$DEBUG
SUBROUTINE GETIMP(FACIMP, ISTRUC, IMSG)
C
C *****
C
C GETIMP - GET IMPACT FACTOR
C
C THIS SUBROUTINE GETS THE IMPACT FACTOR
C
C -----
C
C CALLS TO : READSC
C           REMARK
C           EXIT
C
C -----
C
C VARIABLE DEFINITION
C
C PASSED :
C
C ISTRUC - NUMBER OF STRUCTURES TO BE RATED
C IMSG - ERROR MESSAGE FILE NUMBER
C
C RETURNED :
C
C FACIMP - IMPACT FACTOR OF OVERLOAD VEHICLE
C
C -----
C
C MODIFICATION LOG
C
C 06/11/1986 RAS INITIAL CODING
C
C *****
C
C COMMON /SCANNER/ ENTITY(20), IVAL(2), MODE, NCHAR, NWD, NEXT, ICOLMN
C EQUIVALENCE (IVAL(1), VALUE, IVALUE)
C INTEGER IVALUE
C REAL*4 VALUE
C
C INTEGER DUMMY
C INTEGER IMSG, ISTRUC
C REAL*4 REEL
C REAL*8 FACIMP(1)
C LOGICAL SEP, ENDCRD, NUMR, NEXT
C
C
C I=1
10 IF( .NOT. NUMR(REEL)) GOTO 8000
   FACIMP(1) = DBLE(REEL)
   IF( ENDCRD(DUMMY)) GOTO 40
   IF(SEP (DUMMY)) GOTO 25
   GOTO 30
25 IF(DUMMY .NE. 3) GOTO 8001
   CALL READSC
30 I=I+1
   IF(I .GT. ISTRUC) GOTO 8002
   GOTO 10
40 IF(I .LT. ISTRUC) GOTO 8003
   RETURN

```

```
C
C   ERROR MESSAGES
C
8000 CALL REMARK(IMG, 'ERROR - INCORRECT NUMERIC REAL INPUT.      ')
      GOTO 8990
8001 CALL REMARK(IMG, 'ERROR - ILLEGAL CONTINUATION SEPARATOR.    ')
      GOTO 8990
8002 CALL REMARK(IMG, 'ERROR - TOO MANY IMPACT FACTORS ENTERED.   ')
      GOTO 8990
8003 CALL REMARK(IMG, 'ERROR - TOO FEW IMPACT FACTORS ENTERED.    ')
      GOTO 8990
8990 CALL EXIT
      END
```

\$DEBUG

SUBROUTINE GETRT(RT, ISTRUC, MSG)

C \*\*\*\*\*

C GETRT - GET TRANSVERSE DISTRIBUTION RATIO

C THIS SUBROUTINE GETS THE TRANSVERSE DISTRIBUTION RATIO

C -----  
C CALLS TO : READSC  
C REMARK  
C EXIT

C -----  
C VARIABLE DEFINITION

C PASSED :

C ISTRUC - NUMBER OF STRUCTURES TO BE RATED  
C MSG - ERROR MESSAGE FILE NUMBER.

C RETURNED :

C RT - TRANSVERSE DISTRIBUTION RATIO

C -----  
C MODIFICATION LOG

C 06/11/1986 RAS INITIAL CODING

C \*\*\*\*\*

C COMMON /SCANNER/ ENTITY(20), IVAL(2), MODE, NCHAR, NWD, NEXT, ICOLMN  
C EQUIVALENCE (IVAL(1), VALUE, IVALUE)  
C INTEGER IVALUE  
C REAL\*4 VALUE

C INTEGER DUMMY  
C INTEGER MSG, ISTRUC  
C REAL\*4 REEL  
C REAL\*8 RT(1)  
C LOGICAL SEP, ENDCRD, NUMR, NEXT

C  
10 I=1  
IF( .NOT. NUMR(REEL)) GOTO 8000  
RT(1) = DBLE(REEL)  
IF( ENDCRD(DUMMY)) GOTO 40  
IF(SEP (DUMMY)) GOTO 25  
GOTO 30  
25 IF(DUMMY .NE. 3) GOTO 8001  
CALL READSC  
30 I=I+1  
IF(I .GT. ISTRUC) GOTO 8002  
GOTO 10  
40 IF(I .LT. ISTRUC) GOTO 8003  
RETURN

```
C
C   ERROR MESSAGES
C
8000 CALL REMARK(IMG, 'ERROR - INCORRECT NUMERIC REAL INPUT.      ')
      GOTO 8990
8001 CALL REMARK(IMG, 'ERROR - ILLEGAL CONTINUATION SEPARATOR.    ')
      GOTO 8990
8002 CALL REMARK(IMG, 'ERROR - TOO MANY RATIOS ENTERED.          ')
      GOTO 8990
8003 CALL REMARK(IMG, 'ERROR - TOO FEW RATIOS ENTERED.          ')
      GOTO 8990
8990 CALL EXIT
      END
```

```

$DEBUG
SUBROUTINE REMARK(UNIT,MSG)
C
C *****
C
C   REMARK - WRITE REMARKS
C
C   THIS SUBROUTINE WRITES REMARKS TO OUTPUT UNIT
C
C -----
C
C   CALLS TO : NONE
C
C -----
C
C   VARIABLE DEFINITION
C
C   PASSED :
C
C       UNIT  - ERROR MESSAGE FILE NUMBER
C       MSG   - ERROR MESSAGE
C
C -----
C
C   MODIFICATION LOG
C
C       06/11/1986  RAS  INITIAL CODING
C
C *****
C
C   INTEGER UNIT
C   CHARACTER*45 MSG
C
C   WRITE(UNIT,100)MSG
100  FORMAT(10X,A45)
C   RETURN
C   END

```

```

$DEBUG
SUBROUTINE HEADER(TITLE, ICAP, IPAGE)
C
C *****
C
C   HEADER - HEADINGS
C
C   THIS SUBROUTINE WRITES OUT THE HEADER TO THE OUTPUT FILE
C
C -----
C
C   CALLS TO : NONE
C
C -----
C
C   VARIABLE DEFINITION
C
C   PASSED :
C
C       ICAP  - CAPTION IDENTIFICATION
C
C   RETURNED :
C
C       TITLE - TITLE HEADING
C       IPAGE - PAGE NUMBER
C
C -----
C
C   MODIFICATION LOG
C
C       06/11/1986  RAS  INITIAL CODING
C
C *****
C
C   CHARACTER*4 TITLE(20)
C   CHARACTER*50 CAP(4),KAP
C   INTEGER IPAGE, ICAP, I
C   DATA CAP / ' ', ' ', ' ', ' ',
C   1          ' ', ' ', ' ', ' ',
C   2          ' ', ' ', ' ', ' ',
C   3          ' ', ' ', ' ', ' ',
C
C   OVERLOAD SCAN ECHO           ',
C   OVERLOAD DATA CHECK        ',
C   NATIONAL BRIDGE INVENTORY FILE ',
C   RESULTS                       '/
C
C   IPAGE = IPAGE + 1
C   KAP = CAP(ICAP)
C
C   WRITE(7,10)IPAGE,(TITLE(I),I=1,20)
10  FORMAT('1',///,67X,'PAGE',14,/,
1     10X,'***** OVERLOAD *****',11X,'(VERSION 1.3)',/,
2     10X,'ADOT OVERLOAD BRIDGE RATING',3X,
3     ' ' BY IMBSEN AND ASSOCIATES, INC.',//,
4     10X,20(A4))
C
C   WRITE(7,20)KAP
20  FORMAT(/,10X,65('*'),/,10X,A50,/,10X,65('*'),/)
C
C   RETURN
C   END

```

\$DEBUG

SUBROUTINE FIXCRD

C

C \*\*\*\*\*

C

C FIXCRD - FIX CARD

C

C THIS SUBROUTINE READS FROM THE SCAN INPUT FILE AND  
C WRITES TO THE OUTPUT FILE

C

C

C

C CALLS TO : NONE

C

C

C

C VARIABLE DEFINITION

C

C PASSED : NONE

C

C

C RETURNED: NONE

C

C

C

C MODIFICATION LOG

C

C 06/11/1986 RAS INITIAL CODING

C

C

C

C

C CHARACTER\*4 CARD(20)  
C INTEGER IR, IW

C

10 READ(5,20,END=100) (CARD(IR), IR=1,20)

20 FORMAT(20A4)

WRITE(7,30) ( CARD(IW), IW=1,20)

30 FORMAT(10X,20A4)

GOTO 10

100 CONTINUE

C

RETURN

END

```

$DEBUG
SUBROUTINE WRSTRU(NSTRUC,GRDSPC,OPRING,CONTIN,ISTRUC)
C
C *****
C
C WRSTRU - WRITE STRUCTURE BLOCK
C
C THIS SUBROUTINE WRITES THE STRUCTURE DATA BLOCK INPUT
C
C -----
C
C CALL TO : NONE
C
C -----
C
C VARIABLE DEFINITION
C
C PASSED:
C
C NSTRUC - STRUCTURE NUMBER
C GRDSPC - GIRDER SPACING INPUT
C OPRING - OPERATING RATING INPUT
C CONTIN - CONTINUITY
C ISTRUC - NUMBER OF STRUCTURES TO BE RATED
C
C RETURNED : NONE
C
C -----
C
C MODIFICATION LOG
C
C 06/11/1986 RAS INITIAL CODING
C
C *****
C
C INTEGER NSTRUC(1),OPRING(1),CONTIN(1),ISTRUC
C REAL*8 GRDSPC(1)
C
C WRITE(7,85)
85 FORMAT(/,10X, '***** STRUCTURE DATA *****')
C WRITE(7,90)
90 FORMAT(/, 10X,' STRUCTURE GIRDER OPERATING CONTINUITY',
1 /, 10X,' NUMBER SPACING RATING (0- NBIF ',
2 'DEFAULT)',
3 /, 10X,' (FT) (0- NBIF (1-SIMPLE)',
4 /, 10X,' DEFAULT) (2-CONTIN',
5 'UOUS)',
6 /)
C DO 100 I=1,ISTRUC
C WRITE(7,95)NSTRUC(I),GRDSPC(I),OPRING(I),CONTIN(I)
95 FORMAT(13X,110,F10.2,1X,110,2X,110)
100 CONTINUE
C RETURN
C END

```



```

$DEBUG
SUBROUTINE WRSLAB(ISPLAN,ISLAB)
C
C *****
C
C WRSLAB - WRITE SLAB DATA
C
C THIS SUBROUTINE WRITES THE STANDARD SLAB PLAN DATA
C
C -----
C
C CALL TO : NONE
C
C -----
C
C VARIABLE DEFINITION
C
C PASSED:
C
C ISPLAN - SLAB PLAN CODE
C ISLAB - NUMBER OF STANDARD SLAB PLAN CODES ENTERED
C
C RETURNED : NONE
C
C -----
C
C MODIFICATION LOG
C
C 07/07/1986 RAS INITIAL CODING
C
C *****
C
C INTEGER ISPLAN(ISLAB),ISLAB,ITOGGL,ICODE,ICODEP
C CHARACTER*10 PLAN(9)
C CHARACTER*15 ANALYS(3)
C
C DATA PLAN / 'NA ', 'CS-2-15 ', 'CS-2-20 ',
1 'CS-3-20 ', 'CS-4-20 ', 'CS-5-20 | ',
2 'CS-20 ', 'CS-20 | ', 'CODE ERROR'/
C
C DATA ANALYS/ 'LEVEL 1 ', 'SPECIAL LEVEL 2',
1 'ERROR IN INPUT '/
C
C WRITE(7,85)
85 FORMAT(/,10X, '***** SLAB PLAN DATA *****')
WRITE(7,90)
90 FORMAT(/, 10X,' SLAB SLAB SLAB'
1 /, 10X,' SEQUENCE PLAN PLAN TYPE OF'
2 /, 10X,' NUMBER CODE ACTUAL ANALYSIS'
3 /)
DO 100 I=1,ISLAB
ITOGGL=1
ICODE = ISPLAN(I)
ICODEP=ICODE+1
IF(ICODEP .GT. 8)ICODEP=9
IF(ICODE.GT. 0) ITOGGL=2
IF(ICODE.GT. 7) ITOGGL=3
WRITE(7,95)I,ICODE,PLAN(ICODEP),ANALYS(ITOGGL)
95 FORMAT(11X, I10, 1X, I10, 5X, A10, 1X, A15)
100 CONTINUE

```

```
WRITE(7,110)
110 FORMAT(/,10X,'NOTE: SPECIAL LEVEL 2 DOES NOT CONSIDER A',/
1      10X,'      SLAB WITH HINGES')
RETURN
END
```

```

$DEBUG
SUBROUTINE WRTRUC(PL,DX,IFL,IGROUP,WHO,WHI,WIDTH,IOVLD)
C
C *****
C
C WRTRUC - WRITE TRUCK BLOCK
C
C THIS SUBROUTINE WRITES THE TRUCK BLOCK DATA INPUT
C
C -----
C
C CALLS TO : NONE
C
C -----
C
C VARIABLE DEFINITION
C
C PASSED :
C
C PL - AXLE LOADS
C DX - DISTANCES TO THE AXLES
C IFL - NUMBER OF INDIVIDUAL AXLES
C IGROUP - GROUP IDENTIFICATION
C WHO - OUTER WHEEL GAGE
C WHI - INNER WHEEL GAGE
C WIDTH - VEHICLE WIDTH
C IOVLD - STANDARD OVERLOAD IDENTIFICATION
C
C RETURNED : NONE
C
C -----
C
C MODIFICATION LOG
C
C 06/11/1986 RAS INITIAL CODING
C
C *****
C
C INTEGER IGROUP(1),IFL,1,IOVLD
C REAL*8 PL(1),DX(1),WHO(1),WHI(1),WIDTH
C CHARACTER*18 SV(11)
C
C DATA SV / ' H20 ' ' HS20 '
1 'ALTERNATE MILITARY' ' TYPE 3
2 ' TYPE 3-S2 ' ' TYPE 3-3
3 ' P5 ' ' P7
4 ' P9 ' ' P11
5 ' P13 '/'
WRITE(7,85)
85 FORMAT(/,10X,'***** TRUCK DATA *****')
IF(IOVLD.GT.0) WRITE(7,97) SV(IOVLD)
97 FORMAT(10X,' STANDARD VEHICLE: ',A18)
WRITE(7,100)
100 FORMAT(/,11X,'AXLE AXLE AXLE WHEEL SPACING AXLE',
1 /,11X,'LOAD LOAD LAYOUT OUTER INNER GROUP',
2 /,10X,' ID (KIPS) (FT) (FT) (FT) ID',
3 /)
DO 300 I=1,IFL
WRITE(7,200)I,PL(I),DX(I),WHO(I),WHI(I),IGROUP(I)
200 FORMAT(10X,I5,3F10.2,F9.2,2X,I5)

```

```
300 CONTINUE
    WRITE(7,400)WIDTH
400  FORMAT(/,10X, '    VEHICLE WIDTH      ',F10.2,' (FT)')
    RETURN
    END
```

```

$DEBUG
SUBROUTINE WRIMPA(FACIMP,NSTRUC,ISTRUC)
C
C *****
C
C WRIMPA - WRITE IMPACT DATA BLOCK
C
C THIS SUBROUTINE WRITES THE IMPACT DATA BLOCK INPUT
C
C -----
C
C CALLS TO : NONE
C
C -----
C
C VARIABLE DEFINITION
C
C PASSED :
C
C FACIMP - IMPACT FACTOR OF OVERLOAD VEHICLE
C NSTRUC - STRUCTURE NUMBER
C ISTRUC - NUMBER OF STRUCTURES TO BE RATED
C
C RETURNED : NONE
C
C -----
C
C MODIFICATION LOG
C
C 06/11/1986 RAS INITIAL CODING
C
C *****
C
C INTEGER I,NSTRUC(1),IMESS
C REAL*8 FACIMP(1)
C CHARACTER*1 IFLAG(2),STAR
C
C DATA IFLAG/ ' ', '*' /
C
C IMESS=0
C WRITE(7,100)
100 FORMAT( /,10X, '***** IMPACT DATA *****')
C WRITE(7,150)
150 FORMAT( /,10X, ' STRUCTURE IMPACT',
1 /,10X, ' NUMBER FACTOR',
2 /)
C DO 180 I=1,ISTRUC
C STAR=IFLAG(1)
C IF(FACIMP(1) .EQ. 0.D0)IMESS=IMESS+1
C IF(FACIMP(1) .EQ. 0.D0)STAR=IFLAG(2)
C WRITE(7,175)NSTRUC(1),FACIMP(1),STAR
175 FORMAT(10X,110,F10.2,1X,A1)
180 CONTINUE
C IF(IMESS .GT. 0)WRITE(7,185)
185 FORMAT(/,10X,'ASTERISK (*) INDICATES DEFAULTS TO AASHTO IMPACT',
1 ' FORMULA')
C RETURN
C END

```

```

C
C      WRTRAN - WRITE TRANSVERSE DISTRIBUTION BLOCK
C
C      THIS SUBROUTINE WRITES THE TRANSVERSE DISTRIBUTION DATA
C      BLOCK INPUT
C
C-----
C
C      CALLS TO : NONE
C
C-----
C
C      VARIABLE DEFINITION
C
C      PASSED :
C
C          NSTRUC - STRUCTURE NUMBER
C          RT      - TRANSVERSE DISTRIBUTION RATIO
C          ISTRUC - NUMBER OF STRUCTURES TO BE RATED
C
C      RETURNED : NONE
C
C-----
C
C      MODIFICATION LOG
C
C          06/11/1986 RAS  INITIAL CODING
C
C      *****
C
C      INTEGER NSTRUC(1), ISTRUC
C      REAL*8 RT(1)
C
C      WRITE(7,85)
85  FORMAT(/ ,10X, '***** TRANSVERSE DISTRIBUTION DATA *****')
      WRITE(7,90)
90  FORMAT(/, 10X,'          STRUCTURE          RATIO',
1    /, 10X,'          NUMBER          ',
2    /)
      DO 100 I=1, ISTRUC
          WRITE(7,95) NSTRUC(I), RT(I)
95  FORMAT(17X, I10, F10.2)
100 CONTINUE
      RETURN
      END

```

\$DEBUG

SUBROUTINE RDNBIF(NREC,NSTRUC,CK,CK1,CK2,CK3,CK4,CK5,CK6,  
1 CK7,CK8,CK9,CK10)

C \*\*\*\*\*

RDNBIF - READ NBIF (NATIONAL BRIDGE INVENTORY FILE)

THIS SUBROUTINE READS IN SELECTED DATA FROM A REDUCED  
NBIF (DIRECT ACCESS FILE)

-----  
CALLS TO : NONE  
-----

VARIABLE DEFINITION

PASSED :

NREC - NUMBER OF RECORDS IN THE NBIF  
NSTRUC - ARIZONA STRUCTURE NUMBER

RETURNED :

	NBIF ITEM	DESCRIPTION	NO. BYTES
KK(1)	- 8	STRUCTURE NUMBER	4
CK1	- 5	INVENTORY ROUTE ) SPLIT	1
CK2	- 5	INVENTORY ROUTE )	8
CK3	- 7	FACILITY CARRIED BY STRUCT	18
CK4	- 9	LOCATION	25
KK(2)	- 11	MILEPOINT	4
KK(3)	- 27	YEAR BUILT	4
KK(4)	- 29	AVERAGE DAILY TRAFFIC	4
KK(5)	- 30	YEAR OF ADT	4
KK(6)	- 32	APPROACH ROADWAY WIDTH	4
KK(7)	- 34	SKEW	4
CK5	- 41	STRUCTURE OPEN,POSTED,CLOSED	1
KK(8)	- 43	STRUCTURE TYPE, MAIN	4
KK(9)	- 44	STRUCTURE TYPE, APPROACH	4
KK(10)	- 45	NUMBER SPANS, MAIN	4
KK(11)	- 46	NUMBER APPROACH SPANS	4
KK(12)	- 48	LENGTH MAXIMUM SPAN	4
KK(13)	- 49	STRUCTURE LENGTH	4
KK(14)	- 51	BRIDGE WIDTH (C-TO-C)	4
KK(15)	- 53	MINIMUM VERTICAL CLEARANCE	4
CK6	- 57	WEARING SURFACE	1
CK7	- 58	DECK	1
CK8	- 59	SUPERSTRUCTURE	1
CK9	- 60	SUBSTRUCTURE	1
KK(16)	- 64	OPERATING RATING	4
KK(17)	- 66	INVENTORY RATING	4
CK10	- 67	STRUCTURAL CONDITION	1

-----  
MODIFICATION LOG

06/11/86 RAS INITIAL CODING

```

C
C *****
C
  INTEGER KK(1),NREC,NSTRUC
  CHARACTER*1 CK1,CK5,CK6,CK7,CK8,CK9,CK10
  CHARACTER*8 CK2
  CHARACTER*18 CK3
  CHARACTER*25 CK4
C
  IF(NSTRUC .GT. NREC) RETURN
  READ(8,REC=NSTRUC)KK(1),CK1,CK2,CK3,CK4,(KK(J),J=2,7),
1      CK5,(KK(J),J=8,15),CK6,CK7,CK8,CK9,
2      (KK(J),J=16,17),CK10
C
  RETURN
  END

```



```

$DEBUG
SUBROUTINE WRNBIF(NREC,NSTRUC,CK1,CK2,CK3,CK4,
1          CK5,CK6,CK7,CK8,CK9,CK10,LEVOUT)
C
C *****
C
C WRNBIF - WRITE NBIF (NATIONAL BRIDGE INVENTORY FILE)
C
C THIS SUBROUTINE WRITES THE INFORMATION FROM THE REDUCED
C NBIF
C
C -----
C
C CALLS TO : REMARK
C
C -----
C
C VARIABLE DEFINITION
C
C PASSED :
C
C CK(1-10) - CHARACTER VARIABLES FROM NBIF
C KK       - INTEGER VARIABLES FROM NBIF
C NREC     - NUMBER OF RECORDS IN NBIF
C NSTRUC   - STRUCTURE NUMBER
C LEVOUT   - LEVEL OF OUTPUT FLAG
C
C RETURNED : NONE
C
C -----
C
C MODIFICATION LOG
C
C 06/11/1986 RAS INITIAL CODING
C *****
C
C INTEGER NREC,NSTRUC,CK(1)
C REAL*8 XKK2,XKK14
C CHARACTER*1 CK1,CK5,CK6,CK7,CK8,CK9,CK10
C CHARACTER*8 CK2
C CHARACTER*18 CK3
C CHARACTER*25 CK4
C
C IF(NSTRUC .GT. NREC) GOTO 8000
C XKK2 = DBLE(KK(2))/100.DO
C XKK14 = DBLE(KK(14))/10.DO
C WRITE(7,85)
85  FORMAT(/,10X, '***** STATE OF ARIZONA *****')
C WRITE(7,200)
200 FORMAT(/,12X,'ITEM ITEM ',
3     /,11X,'NUMBER NAME ',/)
C
C WRITE(7,210)KK(1)
210 FORMAT(13X,'0B',4X,'STRUCTURE NO. ....',16)
C
C WRITE(7,220)CK1,CK2
220 FORMAT(13X,'05',4X,'INVENTORY ROUTE .....',A1,A8)
C
C WRITE(7,230)CK3

```

```

230 FORMAT(13X,'07',4X,'FACILITY CARRIED BY STRUCTURE.....',A18)
C
WRITE(7,240)CK4
240 FORMAT(13X,'09',4X,'LOCATION .....',A25)
C
WRITE(7,250)XKK2
250 FORMAT(13X,'11',4X,'MILEPOINT .....',F6.2)
C
WRITE(7,260)KK(3)
260 FORMAT(13X,'27',4X,'YEAR BUILT/RECONSTRUCTION .....',14)
C
WRITE(7,270)KK(4)
270 FORMAT(13X,'29',4X,'AVERAGE DAILY TRAFFIC .....',16)
C
WRITE(7,280)KK(5)
280 FORMAT(13X,'30',4X,'YEAR OF AVERAGE DAILY TRAFFIC ....',12)
C
WRITE(7,290)KK(6)
290 FORMAT(13X,'32',4X,'APPROACH ROADWAY WIDTH (FT) .....',13)
C
WRITE(7,300)KK(7)
300 FORMAT(13X,'34',4X,'SKEW (DEGREES) .....',12)
C
WRITE(7,310)CK5
310 FORMAT(13X,'41',4X,'OPEN(A),POSTED(P),CLOSED(C) .....',A1)
C
WRITE(7,320)KK(8)
320 FORMAT(13X,'43',4X,'STRUCTURE TYPE, MAIN .....',13)
C
WRITE(7,330)KK(9)
330 FORMAT(13X,'44',4X,'STRUCTURE TYPE, APPROACH SPANS ...',13)
C
WRITE(7,340)KK(10)
340 FORMAT(13X,'45',4X,'NO. OF SPANS IN MAIN UNIT .....',13)
C
WRITE(7,350)KK(11)
350 FORMAT(13X,'46',4X,'NO. OF APPROACH SPANS .....',14)
C
WRITE(7,360)KK(12)
360 FORMAT(13X,'48',4X,'LENGTH OF MAXIMUM SPAN (FT) .....',14)
C
WRITE(7,370)KK(13)
370 FORMAT(13X,'49',4X,'STRUCTURE LENGTH (FT) .....',16)
C
WRITE(7,380)XKK14
380 FORMAT(13X,'51',4X,'BRIDGE WIDTH,CURB-TO-CURB (FT) ...',F5.1)
C
WRITE(7,390)KK(15)
390 FORMAT(13X,'53',4X,'MINIMUM VERTICAL CLEARANCE .....',14)
C
WRITE(7,400)CK6
400 FORMAT(13X,'57',4X,'WEARING SURFACE .....',A1)
C
WRITE(7,410)CK7
410 FORMAT(13X,'58',4X,'DECK CONDITION .....',A1)
C
WRITE(7,420)CK8
420 FORMAT(13X,'59',4X,'SUPERSTRUCTURE CONDITION .....',A1)
C
WRITE(7,430)CK9

```

```

430  FORMAT(13X,'60',4X,'SUBSTRUCTURE CONDITION .....',A1)
C
    WRITE(7,440)KK(16)
440  FORMAT(13X,'64',4X,'OPERATING RATING .....',13)
C
    WRITE(7,450)KK(17)
450  FORMAT(13X,'66',4X,'INVENTORY RATING .....',13)
C
    WRITE(7,460)CK10
460  FORMAT(13X,'67',4X,'STRUCTURAL CONDITION .....',A1)
C
    IF(LEVOUT .EQ. 2)WRITE(7,500)
500  FORMAT(10X,/, '      ANALYSIS TERMINATED- ECHO CHECK ONLY')
    IF( KK(1) .EQ. 0)      GOTO 8001
    RETURN
8000 CALL REMARK( 7,'ERROR - STRUCTURE NUMBER > 9861.      ')
      GOTO 8990
8001 CALL REMARK( 7,'ERROR - STRUCTURE NUMBER NOT IN NBIF.  ')
      GOTO 8990
8990 RETURN
      END

```

\$DEBUG

SUBROUTINE LEVEL1(NREC, KK, CK1, CK2, CK3, CK4, CK5, CK10, PL, DX,  
1 GRDSPC, WHO, WHI, WIDTH, FACIMP, IGROUP, IFL,  
2 OPRING, NSTRUC, RTMT, CONTIN, IOVLD)

C \*\*\*\*\*

C LEVEL1 - LEVEL 1 ANALYSIS

C THIS SUBROUTINE DOES A LEVEL 1 ANALYSIS

C -----  
C CALLS TO :

C BRGTYP  
C LONGMT  
C DZERO  
C RATAXL  
C LANE  
C REMARK

C -----  
C VARIABLE DEFINITION

C PASSED :

C NSTRUC - STRUCTURE NUMBER  
C OPRING - OPERATING RATING  
C GRDSPC - GIRDER SPACING  
C FACIMP - IMPACT FACTOR  
C RTMT - TRANSVERSE DISTRIBUTION RATIO  
C CONTIN - CONTINUITY  
C PL - AXLE LOADS  
C DX - DISTANCE TO THE AXLES  
C WHO - OUTER WHEEL GAGE  
C WHI - INNER WHEEL GAGE  
C IGROUP - GROUP IDENTIFICATION  
C IFL - NUMBER OF INDIVIDUAL AXLES ENTERED  
C NREC - NUMBER RECORDS IN NBIF  
C KK(17) - INTEGER VARIABLES FROM REDUCED NBIF  
C WIDTH - WIDTH OF TRUCK  
C CK(1-10) - CHARACTER VARIABLES FROM REDUCED NBIF  
C IOVLD - STANDARD TRUCK CODE

C -----  
C MODIFICATION LOG

C 06/11/1986 RAS INITIAL CODING

C \*\*\*\*\*

C  
C INTEGER ITRUCK, MAINST, NSPAN, IGROUP(1), IFL, IOPER, INVEN, KK(1)  
C INTEGER NAXLE, NTYPE, NMAT, OPRING, NSTRUC, IOP, NSPALL, CONTIN  
C REAL\*8 OPERWT, SPANMX, PL(1), DX(1), GRDSPC  
C REAL\*8 WHO(1), WHI(1), WIDTH, FACIMP  
C REAL\*8 RATEWT, RL(6), RX(6)

```

REAL*8 RTMT,RIMT,RTGIMP,OVDIMP,RFOP,SPANRT
REAL*8 TRANS(3,3),OVLDRR(3,3),C(3,3)
REAL*8 MO(3,3),MR(3,3),LOCO(3,3),LOCR(3,3)
REAL*8 ML(3,3),MLR(3,3),RM(3,3),MLO(3,3)
REAL*8 XKK6,XKK14,XKK2
REAL*8 TEMP1,TEMP2
CHARACTER*1 SCON,CK1,CK5,CK10
CHARACTER*6 CONT
CHARACTER*9 LABEL1
CHARACTER*8 LABEL2
CHARACTER*15 LABELT
CHARACTER*8 CK2
CHARACTER*18 CK3
CHARACTER*25 CK4
CHARACTER*16 MESSAG
CHARACTER*32 IFLAG(2),IFLG

C
DATA IFLAG / '(NOTE- POSITIVE MOMENT CONTROLS)',
1          '(NOTE- NEGATIVE MOMENT CONTROLS)'/
C
IFLG = IFLAG(1)
DO 10 I=1,3
DO 5 J=1,3
MO(I,J) = 0.00
MR(I,J) = 0.00
LOCO(I,J) = 0.00
LOCR(I,J) = 0.00
ML(I,J) = 0.00
MLR(I,J) = 0.00
MLO(I,J) = 0.00
RM(I,J) = 0.00
C(I,J) = 0.00
OVLDRR(I,J)=0.00
TRANS(I,J)= 0.00
5 CONTINUE
10 CONTINUE
C
IBLT = KK(3)/100
IRNS = MOD(KK(3),100)
MAINST= KK(8)
NTYPE = MOD(MAINST,100)
NMAT = MAINST/100
IF(CONTIN .EQ. 0) GOTO 50
IF(CONTIN .EQ. 1) GOTO 40
IF(CONTIN .EQ. 2) GOTO 30
GOTO 50
C
30 IF(NMAT .EQ. 1 .OR. NMAT .EQ. 2) NMAT=2
IF(NMAT .EQ. 3 .OR. NMAT .EQ. 4) NMAT=4
IF(NMAT .EQ. 5 .OR. NMAT .EQ. 6) NMAT=6
GOTO 50
C
40 IF(NMAT .EQ. 1 .OR. NMAT .EQ. 2) NMAT=1
IF(NMAT .EQ. 3 .OR. NMAT .EQ. 4) NMAT=3
IF(NMAT .EQ. 5 .OR. NMAT .EQ. 6) NMAT=5
C
50 XKK2 = DBLE(KK(2))/100.00
C
CALL BRGTYP(NTYPE,NMAT,LABEL1,LABEL2,CONT)
C

```

```

WRITE(7,100)NSTRUC,IBLT,IRNS,CK1,CK2,CK(10),CK3,CK(11),CK4,CK(12),
1      XKK2,CK(13)
100 FORMAT( ,10X,'***** NBIF GENERAL INFORMATION *****',
1      //,10X,' STRUCTURE NO.',15X,110,      5X,'YR BLT/RECON ',
2      12,'/'12,
3      /,10X,' INVENTORY RT.',16X,A1,A8,      5X,'NO.SPANS,MAIN',16,
4      /,10X,' FAC. CARRIED ',7X,A18,      5X,'NO.SPANS,APP ',16,
5      /,10X,' LOCATION ',A25,      5X,'LEN. MAX SPAN',16,
6      /,10X,' MILEPOINT ',19X,F6.2,      5X,'STRUCTURE LEN',16)
IF(NSTRUC .GT. NREC)      GOTO 8000
IF( KK(1) .EQ. 0)      GOTO 8001

C
IF(CONTIN .EQ. 1 .OR. CONTIN .EQ. 2) GOTO 110
WRITE(7,105) LABEL1,LABEL2,CONT
105 FORMAT( /,12X,'STRUCTURE TYPE : ',
1      A9,1X,A8,1X,'(',A6,' -NBIF DEFAULT)')
GOTO 115
110 WRITE(7,111) LABEL1,LABEL2,CONT
111 FORMAT( /,12X,'STRUCTURE TYPE : ',
1      A9,1X,A8,1X,'(',A6,' -NBIF OVERWRITE)')

C
115 IF(NTYPE .EQ. 7) GOTO 700
IF(NTYPE .EQ. 8) GOTO 700
IF(NTYPE .EQ. 9) GOTO 700
IF(NTYPE .EQ. 10) GOTO 700
IF(NTYPE .EQ. 11) GOTO 700
IF(NTYPE .EQ. 12) GOTO 700
IF(NTYPE .EQ. 13) GOTO 700
IF(NTYPE .EQ. 14) GOTO 700
IF(NTYPE .EQ. 15) GOTO 700
IF(NTYPE .EQ. 16) GOTO 700
IF(NTYPE .EQ. 17) GOTO 700
IF(NTYPE .EQ. 18) GOTO 700
IF(NTYPE .EQ. 0) GOTO 700

C
C
XKK6= DBLE(KK(6))
XKK14= DBLE(KK(14))/10.D0
NSPAN= KK(10)
NSPALL= KK(10) + KK(11)
IF(OPRING .EQ.0)IOPER=KK(16)
IF(OPRING .GT.0)IOPER=OPRING
INVEN= KK(17)
SCON= CK10

C
IF(CK1 .EQ. '2')      GOTO 8002
IF(CK5 .EQ. 'P')WRITE(7,8003)
8003 FORMAT(/,10X,'WARNING - STRUCTURE IS POSTED.')
IF(CK5 .EQ. 'C')WRITE(7,8004)
8004 FORMAT(/,10X,'WARNING - STRUCTURE IS CLOSED.')
IF(IOPER .EQ. 800)      GOTO 8005
IF(IOPER .GE. 700 .AND. IOPER .LT. 800)GOTO 8006
IF(IOPER .GE. 900)      GOTO 8007
IF(IOPER .EQ. 0)      GOTO 8008
IF(XKK6 .LT. WIDTH) WRITE(7,8009)
8009 FORMAT(/,10X,'WARNING - APPROACH WIDTH < VEHICLE WIDTH.')
IF(XKK14 .LT. WIDTH)WRITE(7,8010)
8010 FORMAT(/,10X,'WARNING - BRIDGE WIDTH < VEHICLE WIDTH.')

C
ITRUCK= IOPER/100

```

```

      IOP=MOD(IOPER,100)
OPERWT= IOP
SPANMX=DBLE( KK(12))
SPANRT = ( DBLE(KK(13))-(NSPALL -2)*SPANMX )/(2.DO*SPANMX)
IF(NSPALL .EQ. 1) SPANRT = 0.DO
IF(NSPALL .EQ. 2) SPANRT = 1.DO
IF(SPANRT .GT. 1.DO) SPANRT = 1.00DO
C
C   CALCULATE MOMENT FOR OVERLOAD TRUCK
C
      CALL LONGMT(IFL,SPANMX,PL,DX,SPANRT,CONT,NSPALL,MO,LOCO)
C   WRITE(7,140)
C140  FORMAT(' OVERLOAD TRUCK MOMENTS')
C   WRITE(7,150)MO(1,1),MO(1,2),MO(1,3)
C   WRITE(7,150)MO(2,1),MO(2,2),MO(2,3)
C   WRITE(7,150)MO(3,1),MO(3,2),MO(3,3)
C150  FORMAT(3F10.2)
C
      CALL DZERO(RL,6,0.DO)
      CALL DZERO(RX,6,0.DO)
C
C   GET RATING VEHICLE AXLE CONFIGURATION
C
      CALL RATAXL(ITRUCK,NAXLE,RL,RX,RATEWT,LABELT)
C
C   CALCULATE MOMENT FOR RATING VEHICLE
C
      CALL LONGMT(NAXLE,SPANMX,RL,RX,SPANRT,CONT,NSPALL,MR,LOCR)
C   WRITE(7,141)
C141  FORMAT(' RATING TRUCK MOMENTS')
C   WRITE(7,150)MR(1,1),MR(1,2),MR(1,3)
C   WRITE(7,150)MR(2,1),MR(2,2),MR(2,3)
C   WRITE(7,150)MR(3,1),MR(3,2),MR(3,3)
      MO(1,1) = DMAX1(MO(1,1),MO(1,2))
C
C   CHECK FOR LANE LOAD CONTROL FOR H20 AND HS20 TRUCKS
C
      IF(1OVLD .EQ. 1 .OR. 1OVLD .EQ. 2) GOTO 150
      GOTO 155
C
C150  CALL LANE(SPANMX,SPANRT,CONT,NSPALL,MLO)
      MO(1,1) = DMAX1(MLO(1,1),MO(1,1) )
      MO(2,1) = DMAX1(MLO(2,1),MO(2,1) )
      TEMP1 = DABS (MLO(2,2) )
      TEMP2 = DABS ( MO(2,2) )
      MO(2,2) =-DMAX1(TEMP1,TEMP2 )
      MO(3,1) = DMAX1(MLO(3,1),MO(3,1) )
      MO(3,2) = DMAX1(MLO(3,2),MO(3,2) )
      TEMP1 = DABS(MLO(3,3) )
      TEMP2 = DABS( MO(3,3) )
      MO(3,3) =-DMAX1(TEMP1,TEMP2 )
C
C155  IF(ITRUCK .EQ.1) CALL LANE(SPANMX,SPANRT,CONT,NSPALL,ML)
      IF(ITRUCK .EQ.2) CALL LANE(SPANMX,SPANRT,CONT,NSPALL,ML)
C   WRITE(7,142)
C142  FORMAT(' LANE LOADING MOMENTS')
C   WRITE(7,150)ML(1,1),ML(1,2),ML(1,3)
C   WRITE(7,150)ML(2,1),ML(2,2),ML(2,3)
C   WRITE(7,150)ML(3,1),ML(3,2),ML(3,3)
C

```

```

MLR(1,1) = DMAX1(ML(1,1),MR(1,1),MR(1,2))
MLR(2,1) = DMAX1(ML(2,1),MR(2,1) )
TEMP1= DABS(ML(2,2))
TEMP2= DABS(MR(2,2))
MLR(2,2) =-DMAX1(TEMP1, TEMP2 )
MLR(3,1) = DMAX1(ML(3,1),MR(3,1) )
MLR(3,2) = DMAX1(ML(3,2),MR(3,2) )
TEMP1= DABS(ML(3,3))
TEMP2= DABS(MR(3,3))
MLR(3,3) =-DMAX1(TEMP1, TEMP2 )
C WRITE(7,143)
C143 FORMAT(' MAX LANE/RATING MOMENTS')
C WRITE(7,200) MLR(1,1),MLR(1,2),MLR(1,3)
C WRITE(7,200) MLR(2,1),MLR(2,2),MLR(2,3)
C WRITE(7,200) MLR(3,1),MLR(3,2),MLR(3,3)
C200 FORMAT(6F10.2)
C
C CALCULATE LONGITUDINAL RATIO
C
RM(1,1) = MO(1,1)/MLR(1,1)
IF(NSPALL .EQ. 1 .OR. CONT .EQ. 'SIMPLE')GOTO 300
IF(NSPALL .EQ. 2 .AND.CONT .EQ. 'CONTIN')GOTO 255
IF(NSPALL .GT. 2 .AND.CONT .EQ. 'CONTIN')GOTO 260
C
255 RM(2,1) = MO(2,1)/MLR(2,1)
RM(2,2) = MO(2,2)/MLR(2,2)
GOTO 300
C
260 RM(3,1) = MO(3,1)/MLR(3,1)
RM(3,2) = MO(3,2)/MLR(3,2)
RM(3,3) = MO(3,3)/MLR(3,3)
C
C CALCULATE CONTINUITY FACTOR
C
300 C(1,1) = 1.00
C(2,1) = RM(2,1)/ RM(1,1)
C(2,2) = RM(2,2)/ RM(1,1)
C(3,1) = RM(3,1)/ RM(1,1)
C(3,2) = RM(3,2)/ RM(1,1)
C(3,3) = RM(3,3)/ RM(1,1)
C
C CALCULATE IMPACT RATIO
C
RTGIMP = 50.00/(SPANMX + 125.00) + 1.00
IF(RTGIMP .GT. 1.3000)RTGIMP=1.3000
IF(NMAT .EQ. 7) RTGIMP=1.0000
IF(FACIMP .GT. 0.00)GOTO 350
OVDIMP=RTGIMP
GOTO 355
350 OVDIMP = FACIMP + 1.00
355 RIMT = OVDIMP/RTGIMP
C
C TRANSFORMATION FUNCTION
C
IF(NSPALL .EQ. 1 .OR. CONT .EQ. 'SIMPLE')GOTO 356
IF(NSPALL .EQ. 2 .AND.CONT .EQ. 'CONTIN')GOTO 357
IF(NSPALL .GT. 2 .AND.CONT .EQ. 'CONTIN')GOTO 358
C
356 TRANS(1,1) = 1.00/(RM(1,1)*C(1,1)*RTMT*RIMT)
GOTO 360

```



```

C
357 TRANS(2,1) = 1.D0/(RM(1,1)*C(2,1)*RTMT*RIMT)
    TRANS(2,2) = 1.D0/(RM(1,1)*C(2,2)*RTMT*RIMT)
    GOTO 360

C
358 TRANS(3,1) = 1.D0/(RM(1,1)*C(3,1)*RTMT*RIMT)
    TRANS(3,2) = 1.D0/(RM(1,1)*C(3,2)*RTMT*RIMT)
    TRANS(3,3) = 1.D0/(RM(1,1)*C(3,3)*RTMT*RIMT)

C
360 IF(INVEN .NE. IOPER) GOTO 400
    IF(SCON .EQ. '7') OPERWT=OPERWT*1.36D0
    IF(SCON .EQ. '8') OPERWT=OPERWT*1.36D0
    IF(SCON .EQ. '9') OPERWT=OPERWT*1.36D0
400 RFOP = OPERWT/RATEWT
    OVLDRR(1,1) = TRANS(1,1)*RFOP
    OVLDRR(2,1) = TRANS(2,1)*RFOP
    OVLDRR(2,2) = TRANS(2,2)*RFOP
    OVLDRR(3,1) = TRANS(3,1)*RFOP
    OVLDRR(3,2) = TRANS(3,2)*RFOP
    OVLDRR(3,3) = TRANS(3,3)*RFOP

C
    WRITE(7,510) LABELT,OPERWT,RATEWT
510 FORMAT( /,10X,'***** RATING VEHICLE INFORMATION *****',/,
1      ,26X,A15,
2      12X,'OPERATING WEIGHT (TONS) =',F12.2,/,
3      12X,'WT. RATING VEHICLE (TONS) =',F12.2)

C
    IF(OPRING .EQ. 0) WRITE(7,512)RFOP
512 FORMAT( 12X,'RF: OPERATING (NBIF DEFAULT) =',F12.2)
    IF(OPRING .GT. 0) WRITE(7,514)RFOP
514 FORMAT( 12X,'RF: OPERATING (NBIF OVERWRITE) =',F12.2)

C
    WRITE(7,515)
515 FORMAT(/,12X,' LIVE LOAD MOMENT PER LANE (K-FT) AASHTO',/
1      12X,'SPAN EXT INT BENT IMPACT')

C
    WRITE(7,516)MLR(1,1),RTGIMP
516 FORMAT(/,12X,'SIMPLE NA',F12.2,' NA',F7.2)

C
    IF(NSPALL.EQ.2.AND.CONT.EQ.'CONTIN')WRITE(7,520)MLR(2,1),MLR(2,2),
1      RTGIMP
520 FORMAT( 12X,'TWO NA',F12.2,F12.2,F7.2)
    IF(NSPALL.GT.2.AND.CONT.EQ.'CONTIN')WRITE(7,525)MLR(3,1),MLR(3,2),
1      MLR(3,3),RTGIMP
525 FORMAT( 12X,'THREE ',F12.2,F12.2,F12.2,F7.2)

C
    IF(INVEN .NE. IOPER) GOTO 550
    IF(SCON .EQ. '7') WRITE(7,540)
    IF(SCON .EQ. '8') WRITE(7,540)
    IF(SCON .EQ. '9') WRITE(7,540)
540 FORMAT(/, 10X,'NOTE: THIS BRIDGE WAS ADMINISTRATIVELY RATED.',
1      /, 10X,' THE OPERATING RATING WAS MULTIPLIED BY',
2      /, 10X,' 1.36 FOR STRUCTURAL CONDITIONS > 6.')
```

```

C
550 WRITE(7,560)
560 FORMAT( /,10X,'***** OVERLOAD VEHICLE INFORMATION *****')
    WRITE(7,562)
562 FORMAT(/,12X,' LIVE LOAD MOMENT PER LANE (K-FT) ',/
1      12X,'SPAN EXT INT BENT IMPACT')
```

```

IF(FACIMP .GT. 0.D0)GOTO 570
MESSAG= '(AASHTO DEFAULT)'
GOTO 575
570 MESSAG= '(OVERWRITE) '
C
575 WRITE(7,580)MO(1,1),OVDIMP,MESSAG
580 FORMAT(/,12X,'SIMPLE NA',F12.2,' NA',F7.2,1X,
1 A16)
C
IF(NSPALL.EQ.2.AND.CONT.EQ.'CONTIN')WRITE(7,585)MO(2,1),MO(2,2),
OVDIMP,MESSAG
1
585 FORMAT( 12X,'TWO NA',F12.2,F12.2,F7.2,1X,A16)
IF(NSPALL.GT.2.AND.CONT.EQ.'CONTIN')WRITE(7,590)MO(3,1),MO(3,2),
MO(3,3),OVDIMP,MESSAG
1
590 FORMAT( 12X,'THREE ',F12.2,F12.2,F12.2,F7.2,1X,A16)
C
WRITE(7,595)SPANRT
595 FORMAT( /,10X,'***** LEVEL 1 ANALYSIS *****',/
1 ,42X,' SPAN RATIO ',F6.3,/,
2 ,10X,' MOMENT IMPACT TRAN. ',/
3 10X,' RATIO CONTIN RATIO RATIO OVLDRR')
C
IF(NSPALL .EQ. 1 .OR. CONT .EQ. 'SIMPLE') GOTO 600
IF(NSPALL .EQ. 2 .AND. CONT .EQ. 'CONTIN') GOTO 625
IF(NSPALL .GT. 2 .AND. CONT .EQ. 'CONTIN') GOTO 650
C
600 WRITE(7,605) RM(1,1),C(1,1),RIMT,RTMT,OVLDRR(1,1)
605 FORMAT(/,12X,'SIMPLE',5F7.2)
CONTROL=OVLDRR(1,1)
GOTO 680
C
625 WRITE(7,630) RM(1,1),C(2,1),RIMT,RTMT,OVLDRR(2,1)
630 FORMAT(/,12X,' INT',5F7.2)
WRITE(7,635) RM(1,1),C(2,2),RIMT,RTMT,OVLDRR(2,2)
635 FORMAT( 12X,' BENT',5F7.2)
CONTROL=DMINI(OVLDRR(2,1),OVLDRR(2,2))
IF(OVLDRR(2,2) .LT. OVLDRR(2,1))IFLG=IFLAG(2)
GOTO 680
C
650 WRITE(7,655) RM(1,1),C(3,1),RIMT,RTMT,OVLDRR(3,1)
655 FORMAT(/,12X,' EXT',5F7.2)
WRITE(7,660) RM(1,1),C(3,2),RIMT,RTMT,OVLDRR(3,2)
660 FORMAT( 12X,' INT',5F7.2)
WRITE(7,665) RM(1,1),C(3,3),RIMT,RTMT,OVLDRR(3,3)
665 FORMAT( 12X,' BENT',5F7.2)
CONTROL=DMINI(OVLDRR(3,1),OVLDRR(3,2),OVLDRR(3,3))
IF(OVLDRR(3,3) .LT. OVLDRR(3,1) .AND.
1 OVLDRR(3,3) .LT. OVLDRR(3,2)) IFLG=IFLAG(2)
C
680 WRITE(7,685)CONTROL,IFLG
685 FORMAT(/,12X,'CONTROLLING OVLDRR =',F10.2,1X,A32)
C
IF(CONTROL .GE. 1.D0) WRITE(7,686)
IF(CONTROL .GE. 0.70D0 .AND. CONTROL .LT. 1.D0)WRITE(7,690)
IF(CONTROL .LT. 0.70D0) WRITE(7,695)
686 FORMAT( /,10X,' ***** THIS BRIDGE IS A GO ***** ' )
690 FORMAT( /,10X,' ***** LEVEL 2 ANALYSIS REQUIRED ***** ' )
695 FORMAT( /,10X,' ***** THIS BRIDGE IS A NO GO ***** ' )
RETURN
C

```

```

700 WRITE(7,690)
    RETURN
8000 CALL REMARK( 7,'ERROR - STRUCTURE NUMBER > 9861.      ')
      GOTO 8990
8001 CALL REMARK( 7,'ERROR - STRUCTURE NUMBER NOT IN NBIF.  ')
      GOTO 8990
8002 CALL REMARK( 7,'ERROR - STRUCTURE IS AN UNDERCROSSING. ')
      GOTO 8990
8005 CALL REMARK( 7,'ERROR - PEDESTRIAN LOADING ONLY.      ')
      GOTO 8990
8006 CALL REMARK( 7,'ERROR - RAILROAD LOADING ONLY.         ')
      GOTO 8990
8007 CALL REMARK( 7,'ERROR - GROSS LOADING ONLY.            ')
      GOTO 8990
8008 CALL REMARK( 7,'ERROR - OPERATING RATING IS NOT GIVEN. ')
      GOTO 8990
8990 RETURN
    END

```

```

$DEBUG
SUBROUTINE RATAXL (ITRUCK,NAXLE,RL,RX,RATEWT,LABELT)
C
C *****
C
C RATAXL - RATING AXLE CONFIGURATION
C
C THIS SUBROUTINE GETS THE AXLE LOADS AND CONFIGURATION
C FOR THE RATING TRUCK
C
C -----
C
C CALLS TO : NONE
C
C -----
C
C VARIABLE DEFINITION
C
C PASSED :
C
C ITRUCK - TRUCK TYPE CODE
C
C RETURNED :
C
C NAXLE - NUMBER OF AXLES
C RL - AXLE LOADS
C RX - DISTANCES TO THE AXLES
C RATEWT - TRUCK RATING WEIGHT
C LABELT - TRUCK LABEL
C
C -----
C
C MODIFICATION LOG
C
C 06/11/1986 RAS INITIAL CODING
C
C *****
C
C INTEGER ITRUCK,NAXLE
C REAL*8 RL(1),RX(1),RATEWT
C CHARACTER*15 LABELT,TR(6)
C
C DATA TR/ ' H20 TRUCK ',
1 ' HS20 TRUCK ',
2 'ALTERNATE TRUCK',
3 ' TYPE 3 TRUCK ',
4 'TYPE 3-S2 TRUCK',
5 'TYPE 3-3 TRUCK'/
C
C IF(ITRUCK .EQ. 1) GOTO 110
C IF(ITRUCK .EQ. 2) GOTO 120
C IF(ITRUCK .EQ. 3) GOTO 130
C IF(ITRUCK .EQ. 4) GOTO 140
C IF(ITRUCK .EQ. 5) GOTO 150
C IF(ITRUCK .EQ. 6) GOTO 160
C WRITE(7,200)
200 FORMAT(/,10X,'ERROR - RATING VEHICLE CODE INCORRECT IN',
1 ' SUBROUTINE RATAXL')
C STOP
C

```

```

C   H2O TRUCK
C
110 NAXLE = 2
    RL(1) = 8.00
    RL(2) = 32.00
    RX(1) = 0.00
    RX(2) = 14.00
    RATEWT= 20.00
    LABELT= TR(1)
    GOTO 170

C
C   HS20 TRUCK
C
120 NAXLE = 3
    RL(1) = 8.00
    RL(2) = 32.00
    RL(3) = 32.00
    RX(1) = 0.00
    RX(2) = 14.00
    RX(3) = 28.00
    RATEWT= 36.00
    LABELT= TR(2)
    GOTO 170

C
C   ALTERNATE MILITARY
C
130 NAXLE = 2
    RL(1) = 24.00
    RL(2) = 24.00
    RX(1) = 0.00
    RX(2) = 4.00
    RATEWT= 24.00
    LABELT= TR(3)
    GOTO 170

C
C   TYPE 3
C
140 NAXLE = 3
    RL(1) = 16.00
    RL(2) = 17.00
    RL(3) = 17.00
    RX(1) = 0.00
    RX(2) = 15.00
    RX(3) = 19.00
    RATEWT= 25.00
    LABELT= TR(4)
    GOTO 170

C
C   TYPE 3-S2
C
150 NAXLE = 5
    RL(1) = 10.00
    RL(2) = 15.500
    RL(3) = 15.500
    RL(4) = 15.500
    RL(5) = 15.500
    RX(1) = 0.00
    RX(2) = 11.00
    RX(3) = 15.00
    RX(4) = 37.00

```

RX(5) = 41.D0  
RATEWT= 36.D0  
LABELT= TR(5)  
GOTO 170

C

C

TYPE 3-3

C

160 NAXLE = 6  
RL(1) = 12.D0  
RL(2) = 12.D0  
RL(3) = 12.D0  
RL(4) = 16.D0  
RL(5) = 14.D0  
RL(6) = 14.D0  
RX(1) = 0.D0  
RX(2) = 15.D0  
RX(3) = 19.D0  
RX(4) = 34.D0  
RX(5) = 50.D0  
RX(6) = 54.D0  
RATEWT= 40.D0  
LABELT= TR(6)

C

170 RETURN  
END

```

$DEBUG
SUBROUTINE LANE(SPANMX,SPANRT,CONT,NSPAN,ML)
C
C *****
C
C LANE - LANE LOADING
C
C THIS SUBROUTINE CALCULATES THE LANE LOAD MOMENT
C FOR A H20 AND HS20 TRUCK
C
C -----
C
C CALLS TO : NONE
C
C -----
C
C VARIABLE DEFINITION
C
C PASSED :
C
C SPANMX - MAXIMUM SPAN LENGTH
C SPANRT - END SPAN RATIO
C NSPAN - NUMBER OF SPANS
C CONT - CONTINUITY
C
C RETURNED :
C
C ML - MOMENT
C
C -----
C
C MODIFICATION LOG
C
C 06/11/1986 RAS INITIAL CODING
C
C *****
C
C REAL*8 R1T(3),R2T(3),R3T(3)
C REAL*8 W,RIDER,SPANMX,C1,C2,SPANRT
C REAL*8 DD1,DD2,DD3,DD4
C REAL*8 ML(3,3)
C CHARACTER*6 CONT
C INTEGER NSPAN
C
C W=0.64D0
C RIDER=18.D0
C C1= W*SPANMX*SPANMX
C C2= RIDER*SPANMX
C CALL DZERO(R1T,3,0.D0)
C CALL DZERO(R2T,3,0.D0)
C CALL DZERO(R3T,3,0.D0)
C
C *****
C
C SIMPLE BEAM MOMENT
C *****
C
C ML(1,1)= 0.125D0 *C1 + 0.25D0 *C2
C IF(NSPAN .EQ. 1 .OR. CONT .EQ. 'SIMPLE') RETURN
C
C *****

```

```

C   TWO SPAN MOMENT
C *****
C
C   (POSITIVE MOMENT)
C   ML(2,1)= 0.095703D0*C1 + 0.207401D0*C2
C
C   (NEGATIVE MOMENT)
C   ML(2,2)= -0.125D0 *C1 - 0.1875D0 *C2
C   IF(NSPAN .EQ. 2) RETURN
C
C *****
C   THREE SPAN MOMENT
C *****
C
C   (POSITIVE MOMENT- 0.4 PT)
C 100 R1T(1) = 0.1000D0*C1 + 0.20416D0*C2
C     R1T(2) = 0.0581D0*C1 + 0.15648D0*C2
C     R1T(3) = 0.0269D0*C1 + 0.10740D0*C2
C
C   (POSITIVE MOMENT- 0.5 PT)
C     R2T(1) = 0.0750D0*C1 + 0.1750D0*C2
C     R2T(2) = 0.0694D0*C1 + 0.16667D0*C2
C     R2T(3) = 0.0625D0*C1 + 0.15625D0*C2
C
C   (NEGATIVE MOMENT)
C     R3T(1) = -0.1167D0*C1 -0.1824D0*C2
C     R3T(2) = -0.0884D0*C1 -0.1568D0*C2
C     R3T(3) = -0.0742D0*C1 -0.1380D0*C2
C
C   IF(SPANRT .GE. 0.75)GOTO 200
C   IF(SPANRT .GE. 0.50)GOTO 300
C   GOTO 400
C
C 200 DD4 = (SPANRT -0.75D0)/0.25D0
C     DD1 = R1T(1) - R1T(2)
C     DD2 = R2T(1) - R2T(2)
C     DD3 = R3T(1) - R3T(2)
C     ML(3,1)= R1T(2) + DD4*DD1
C     ML(3,2)= R2T(2) + DD4*DD2
C     ML(3,3)= R3T(2) + DD4*DD3
C     RETURN
C 300 DD4 = (SPANRT -0.50D0)/0.25D0
C     DD1 = R1T(2) - R1T(3)
C     DD2 = R2T(2) - R2T(3)
C     DD3 = R3T(2) - R3T(3)
C     ML(3,1)= R1T(3) + DD4*DD1
C     ML(3,2)= R2T(3) + DD4*DD2
C     ML(3,3)= R3T(3) + DD4*DD3
C     RETURN
C 400 ML(3,1)= R1T(3)
C     ML(3,2)= R2T(3)
C     ML(3,3)= R3T(3)
C     RETURN
C     END

```



\$DEBUG

SUBROUTINE SPLVII(NREC, KK, CK1, CK2, CK3, CK4, CK5, CK10, PL, DX,  
1 GRDSPC, WHO, WHI, WIDTH, FACIMP, IGROUP, IFL,  
2 NSTRUC, RTMT, ISPLAN, IOVLD)

C  
C \*\*\*\*\*

C SPLVII - SPECIAL LEVEL 2 ANALYSIS

C THIS SUBROUTINE DOES A SPECIAL LEVEL 2 ANALYSIS FOR  
C SLAB BRIDGES

C -----  
C CALLS TO :

C SSLAB  
C DFSLAB  
C IMSLAB  
C FIXSB

C -----  
C VARIABLE DEFINITION

C PASSED :

C NSTRUC - STRUCTURE NUMBER  
C GRDSPC - GIRDER SPACING  
C FACIMP - IMPACT FACTOR  
C RTMT - TRANSVERSE DISTRIBUTION RATIO  
C PL - AXLE LOADS  
C DX - DISTANCE TO THE AXLES  
C WHO - OUTER WHEEL GAGE  
C WHI - INNER WHEEL GAGE  
C IGROUP - GROUP IDENTIFICATION  
C IFL - NUMBER OF INDIVIDUAL AXLES ENTERED  
C NREC - NUMBER RECORDS IN NBIF  
C KK(17) - INTEGER VARIABLES FROM REDUCED NBIF  
C WIDTH - WIDTH OF TRUCK  
C CK(1-10) - CHARACTER VARIABLES FROM REDUCED NBIF  
C ISPLAN - STANDARD SLAB PLAN CODE  
C IOVLD - STANDARD TRUCK CODE

C -----  
C MODIFICATION LOG

C 07/07/1986 RAS INITIAL CODING

C \*\*\*\*\*

C INTEGER MAINST, NSPAN, IGROUP(1), IFL, KK(1)  
C INTEGER NTYPE, NMAT, NSTRUC, NSPALL  
C INTEGER ISPLAN  
C REAL\*8 SPANMX, BRDWT, PL(1), DX(1), GRDSPC  
C REAL\*8 WHO(1), WHI(1), WIDTH, FACIMP  
C REAL\*8 RTMT, OVDIMP, SPANRT  
C REAL\*8 MO(3,3), LOCO(3,3), MLO(3,3)

```

REAL*8  XKK6,XKK14,XKK2
REAL*8  DF(3),IM(3),SLABRR(3),RIMT(3),MCPDL(3),MLL(3),CONTROL
REAL*8  MAXS(5),ALPHAS(5),EXTS(5),INTS(5),NEGS(5)
REAL*8  TEMP1,TEMP2
CHARACTER*1  SCON,CK1,CK5,CK10
CHARACTER*6  CONT
CHARACTER*9  LABEL1
CHARACTER*8  LABEL2
CHARACTER*8  CK2
CHARACTER*18 CK3
CHARACTER*32 IFLAG(2),IFLG

C
DATA IFLAG / '(NOTE- POSITIVE MOMENT CONTROLS)',
1          '(NOTE- NEGATIVE MOMENT CONTROLS)'/

C
IFLG = IFLAG(1)

C
DO 10 I=1,3
DO 5  J=1,3
      MO(I,J) = 0.00
      MLO(I,J) = 0.00
      LOCO(I,J) = 0.00
5     CONTINUE
10    CONTINUE

C
IBLT = KK(3)/100
IRSN = MOD(KK(3),100)
MAINST= KK(8)
NTYPE = MOD(MAINST,100)
NMAT = MAINST/100
XKK2 = DBLE(KK(2))/100.00

C
CALL BRGTYP(NTYPE,NMAT,LABEL1,LABEL2,CONT)

C
WRITE(7,100)NSTRUC,IBLT,IRSN,CK1,CK2,CK(10),CK3,CK(11),CK4,CK(12),
1          XKK2,CK(13)
100  FORMAT(//,10X,'***** NBIF GENERAL INFORMATION *****',
1      //,10X,' STRUCTURE NO.',15X,110, 5X,'YR BLT/RECON ',
2          12,'/',12,
3          /,10X,' INVENTORY RT.',16X,A1,A8, 5X,'NO.SPANS,MAIN',16,
4          /,10X,' FAC. CARRIED ',7X,A18, 5X,'NO.SPANS,APP ',16,
5          /,10X,' LOCATION ',A25, 5X,'LEN. MAX SPAN',16,
6          /,10X,' MILEPOINT ',19X,F6.2, 5X,'STRUCTURE LEN',16)
IF(NSTRUC .GT. NREC) GOTO 8000
IF( KK(1) .EQ. 0) GOTO 8001
WRITE(7,105) LABEL1,LABEL2,CONT
105  FORMAT( /,10X,' STRUCTURE TYPE : ',A9,1X,A8,1X,'(',A6,')')

C
XKK6= DBLE(KK(6))
XKK14= DBLE(KK(14))/10.00
NSPAN= KK(10)
NSPALL= KK(10) + KK(11)
SCON= CK10

C
IF(CK1 .EQ. '2') GOTO 8002
IF(CK5 .EQ. 'P')WRITE(7,8003)
8003 FORMAT(/,10X,'WARNING - STRUCTURE IS POSTED.')
IF(CK5 .EQ. 'C')WRITE(7,8004)
8004 FORMAT(/,10X,'WARNING - STRUCTURE IS CLOSED.')
IF(XKK6 .LT. WIDTH) WRITE(7,8009)

```

```

8009 FORMAT(/,10X,'WARNING - APPROACH WIDTH < VEHICLE WIDTH. ')
      IF(XKK14 .LT. WIDTH)WRITE(7,8010)
8010 FORMAT(/,10X,'WARNING - BRIDGE WIDTH < VEHICLE WIDTH. ')
C
      SPANMX=DBLE( KK(12))
      BRDWT=XKK14
C
C   GET STANDARD SLAB PLAN CURVES
C
      CALL FIXSB(ISPLAN,SPANMX)
      CALL SSLAB(ISPLAN,MAXS,ALPHAS,EXTS,INTS,NEGS)
      CALL TERPS(SPANMX,MAXS,ALPHAS,SPANRT)
      CALL TERPS(SPANMX,MAXS,EXTS ,MCAPDL(1))
      CALL TERPS(SPANMX,MAXS,INTS ,MCAPDL(2))
      CALL TERPS(SPANMX,MAXS,NEGS ,MCAPDL(3))
C
C   CALCULATE MOMENT FOR OVERLOAD TRUCK
C
      CALL LONGMT(IFL,SPANMX,PL,DX,SPANRT,CONT,NSPALL,MO,LOCO)
C
C   CHECK FOR LANE LOAD CONTROL FOR H20 AND HS20 TRUCKS
C
      IF(10VLD .EQ. 1 .OR. 10VLD .EQ. 2) GOTO 150
      GOTO 155
C
150   CALL LANE(SPANMX,SPANRT,CONT,NSPALL,MLO)
      MO(1,1) = DMAX1(MLO(1,1),MO(1,1) )
      MO(2,1) = DMAX1(MLO(2,1),MO(2,1) )
      TEMP1 = DABS (MLO(2,2) )
      TEMP2 = DABS ( MO(2,2) )
      MO(2,2) =--DMAX1(TEMP1,TEMP2 )
      MO(3,1) = DMAX1(MLO(3,1),MO(3,1) )
      MO(3,2) = DMAX1(MLO(3,2),MO(3,2) )
      TEMP1 = DABS(MLO(3,3) )
      TEMP2 = DABS( MO(3,3) )
      MO(3,3) =--DMAX1(TEMP1,TEMP2 )
C
C   WRITE(7,140)
C140  FORMAT(' OVERLOAD TRUCK MOMENTS')
C     WRITE(7,150)MO(1,1),MO(1,2),MO(1,3)
C     WRITE(7,150)MO(2,1),MO(2,2),MO(2,3)
C     WRITE(7,150)MO(3,1),MO(3,2),MO(3,3)
C150  FORMAT(3F10.2)
C
C   CALCULATE AASHTO DISTRIBUTION FACTOR
C
155   CALL DFSLAB(SPANRT,SPANMX,DF(1),DF(2),DF(3))
C
C   CALCULATE AASHTO IMPACT FACTOR
C
      CALL IMSLAB(SPANRT,SPANMX,IM(1),IM(2),IM(3))
      IF(FACIMP .GT. 0.00) GOTO 500
      RIMT(1) = 1.000
      RIMT(2) = 1.000
      RIMT(3) = 1.000
      GOTO 550
500   OVDIMP = FACIMP + 1.00
      RIMT(1) = OVDIMP/IM(1)
      RIMT(2) = OVDIMP/IM(2)
      RIMT(3) = OVDIMP/IM(3)
C

```

```

C   CALCULATE OVERLOAD RATING FACTOR
C
550 MLL(1)   = MO(3,1)*DF(1)*IM(1)
    MLL(2)   = MO(3,2)*DF(2)*IM(2)
    MLL(3)   = MO(3,3)*DF(3)*IM(3)
    SLABRR(1) = MCAPDL(1)/(MLL(1)*RIMT(1)*RTMT)
    SLABRR(2) = MCAPDL(2)/(MLL(2)*RIMT(2)*RTMT)
    SLABRR(3) = -MCAPDL(3)/(MLL(3)*RIMT(3)*RTMT)
    CONTROL   = DMIN1(SLABRR(1),SLABRR(2),SLABRR(3))
    IF(SLABRR(3) .LT. SLABRR(1) .AND.
1   SLABRR(3) .LT. SLABRR(2)) IFLG=IFLAG(2)
C
    WRITE(7,560)
560 FORMAT(/,10X,'***** OVERLOAD VEHICLE INFORMATION *****')
    WRITE(7,573)MO(3,1),IM(1),DF(1),MO(3,2),IM(2),DF(2),
1     MO(3,3),IM(3),DF(3)
573 FORMAT(/,12X,'          L.L. MOMENT      AASHTO      AASHTO LOAD',/
1     12X,'          PER LANE          IMPACT      DISTRIBUTION',/
2     12X,'          (K-FT)              (LANES)',/
3     /,12X,'EXT SPAN ',F12.2,F12.2,3X,F12.4,      /
4     12X,'INT SPAN ',F12.2,F12.2,3X,F12.4,      /
5     12X,'BENT      ',F12.2,F12.2,3X,F12.4)
C
    WRITE(7,580)
580 FORMAT(/,10X,'***** SPECIAL LEVEL 2 ANALYSIS *****',/
1     /,10X,'          (HINGES NOT CONSIDERED )          ')
    WRITE(7,583)SPANRT
583 FORMAT(/,12X,'EXTERIOR TO INTERIOR SPAN RATIO ',F6.3)
    WRITE(7,585)MCAPDL(1),MLL(1),RIMT(1),RTMT,SLABRR(1),
1     MCAPDL(2),MLL(2),RIMT(2),RTMT,SLABRR(2),
2     MCAPDL(3),MLL(3),RIMT(3),RTMT,SLABRR(3)
585 FORMAT(/,12X,'          MOMENT - DEAD L.L.MOMENT  IMPACT  TRAN.',/
1     12X,'          CAPACITY LOAD  PLUS IMP.  RATIO  RATIO',/
2     '          OVLDRR'
3     12X,'          (K-FT)          (K-FT)          ',/
4     /,12X,'EXT SPAN ',F10.2,F12.2,F9.2,F8.2,F9.2,      /
5     12X,'INT SPAN ',F10.2,F12.2,F9.2,F8.2,F9.2,      /
6     12X,'BENT      ',F10.2,F12.2,F9.2,F8.2,F9.2)
C
    WRITE(7,590)CONTROL,IFLG
590 FORMAT(/,12X,'CONTROLLING OVLDRR =',F10.2,1X,A32)
C
    IF(CONTROL .GE. 1.00) WRITE(7,595)
    IF(CONTROL .GE. 0.7000 .AND. CONTROL .LT. 1.00)WRITE(7,596)
    IF(CONTROL .LT. 0.7000) WRITE(7,597)
595 FORMAT(/,10X,'          ***** THIS BRIDGE IS A GO ***** ')
596 FORMAT(/,10X,'          ***** LEVEL 2 ANALYSIS REQUIRED ***** ')
597 FORMAT(/,10X,'          ***** THIS BRIDGE IS A NO GO ***** ')
C
    RETURN
8000 CALL REMARK( 7,'ERROR - STRUCTURE NUMBER > 9861. ')
    GOTO 8990
8001 CALL REMARK( 7,'ERROR - STRUCTURE NUMBER NOT IN NBIF. ')
    GOTO 8990
8002 CALL REMARK( 7,'ERROR - STRUCTURE IS AN UNDERCROSSING. ')
    GOTO 8990
8990 RETURN
    END

```

```

$DEBUG
SUBROUTINE DFSLAB(SPANRT,SPANMX,DFEXT,DFINT,DFNEG)
C
C *****
C
C DFSLAB - DISTRIBUTION FACTOR FOR SLAB BRIDGES
C
C THIS SUBROUTINE CALCULATES THE AASHTO DISTRIBUTION FACTOR
C FOR A SLAB BRIDGE
C
C -----
C
C CALLS TO : NONE
C
C -----
C
C VARIABLE DEFINITION
C
C PASSED :
C
C SPANMX - MAXIMUM SPAN LENGTH
C SPANRT - END SPAN RATIO
C
C RETURNED :
C
C DFEXT - DISTRIBUTION FACTOR AT THE EXTERIOR SPAN
C DFINT - DISTRIBUTION FACTOR AT THE INTERIOR SPAN
C DFNEG - DISTRIBUTION FACTOR AT THE BENT
C
C -----
C
C MODIFICATION LOG
C
C 07/07/1986 RAS INITIAL CODING
C
C *****
C
C REAL*8 SPANRT,SPANMX
C REAL*8 DFEXT,DFINT,DFNEG
C
C DFEXT = 4.00 + 0.0600*SPANRT*SPANMX
C IF(DFEXT .GT. 7.00) DFEXT=7.00
C DFEXT = 1.00/(2.00*DFEXT)
C
C DFINT = 4.00 + 0.0600*SPANMX
C IF(DFINT .GT. 7.00) DFINT=7.00
C DFINT = 1.00/(2.00*DFINT)
C
C DFNEG = 4.00 + 0.0600*(SPANMX + SPANMX*SPANRT)/2.00
C IF(DFNEG .GT. 7.00) DFNEG=7.00
C DFNEG = 1.00/(2.00*DFNEG)
C
C RETURN
C END

```

```

$DEBUG
SUBROUTINE IMSLAB(SPANRT,SPANMX,IMEXT,IMINT,IMNEG)
C
C *****
C
C   IMSLAB - IMPACT FACTOR
C
C   THIS SUBROUTINE CALCULATES THE AASHTO IMPACT FACTOR
C
C -----
C
C   CALLS TO : NONE
C
C -----
C
C   VARIABLE DEFINITION
C
C   PASSED :
C
C       SPANMX - MAXIMUM SPAN LENGTH
C       SPANRT - END SPAN RATIO
C
C   RETURNED :
C
C       IMEXT - IMPACT FACTOR AT THE EXTERIOR SPAN
C       IMINT - IMPACT FACTOR AT THE INTERIOR SPAN
C       IMNEG - IMPACT FACTOR AT THE BENT
C
C -----
C
C   MODIFICATION LOG
C
C       07/07/1986  RAS  INITIAL CODING
C
C *****
C
C   REAL*8 SPANRT,SPANMX
C   REAL*8 IMEXT,IMINT,IMNEG
C
C   IMEXT = SPANRT*SPANMX + 125.00
C   IMEXT = 1.00 + 50.00/IMEXT
C   IF(IMEXT .GT. 1.300) IMEXT = 1.300
C
C   IMINT = SPANMX + 125.00
C   IMINT = 1.00 + 50.00/IMINT
C   IF(IMINT .GT. 1.300) IMINT = 1.300
C
C   IMNEG = (SPANMX*SPANRT + SPANMX)/2.00 + 125.00
C   IMNEG = 1.000 + 50.00/IMNEG
C   IF(IMNEG .GT. 1.300) IMNEG = 1.300
C
C   RETURN
C   END

```



```

C      DATA MAXC / 25.0D0, 30.0D0, 35.0D0, 40.0D0 /
      DATA ALPHAC / .820D0, .817D0, .814D0, .813D0 /
      DATA EXTC / 24.28D0, 29.75D0, 35.98D0, 43.56D0 /
      DATA INTC / 24.86D0, 30.59D0, 37.18D0, 45.06D0 /
      DATA NEGC / 34.09D0, 37.38D0, 41.70D0, 51.16D0 /

C      DATA MAXD / 25.0D0, 30.0D0, 35.0D0, 40.0D0, 45.0D0 /
      DATA ALPHAD / .820D0, .817D0, .814D0, .813D0, .811D0 /
      DATA EXTD / 24.27D0, 29.75D0, 37.07D0, 43.08D0, 49.83D0 /
      DATA INTD / 24.89D0, 30.59D0, 38.27D0, 44.48D0, 51.73D0 /
      DATA NEG D / 30.69D0, 35.46D0, 40.39D0, 48.56D0, 56.17D0 /

C      DATA MAXE / 25.0D0, 30.0D0, 35.0D0, 40.0D0, 45.0D0 /
      DATA ALPHAE / .820D0, .817D0, .814D0, .813D0, .811D0 /
      DATA EXTE / 28.43D0, 34.56D0, 41.62D0, 46.33D0, 51.63D0 /
      DATA INTE / 29.08D0, 35.46D0, 42.82D0, 47.93D0, 53.53D0 /
      DATA NEGE / 35.65D0, 40.39D0, 45.25D0, 50.92D0, 57.42D0 /

C      DATA MAXF / 25.0D0, 30.0D0, 35.0D0, 40.0D0 /
      DATA ALPHAF / .820D0, .817D0, .814D0, .813D0 /
      DATA EXTF / 36.90D0, 45.78D0, 56.89D0, 70.10D0 /
      DATA INTF / 37.62D0, 46.79D0, 58.19D0, 71.90D0 /
      DATA NEGF / 42.63D0, 49.34D0, 57.52D0, 72.27D0 /

C      DATA MAXG / 25.0D0, 30.0D0, 35.0D0, 40.0D0 /
      DATA ALPHAG / .820D0, .817D0, .814D0, .813D0 /
      DATA EXTG / 43.58D0, 53.20D0, 60.87D0, 70.10D0 /
      DATA INTG / 44.30D0, 54.21D0, 62.17D0, 71.90D0 /
      DATA NEGG / 50.80D0, 58.10D0, 62.10D0, 72.27D0 /

C      CALL DZERO(MAXS, 5,0.D0)
      CALL DZERO(ALPHAS,5,0.D0)
      CALL DZERO(EXTS, 5,0.D0)
      CALL DZERO(INTS, 5,0.D0)
      CALL DZERO(NEGS, 5,0.D0)

C
C      IF(ICURVE .EQ. 1) GOTO 10
C      IF(ICURVE .EQ. 2) GOTO 20
C      IF(ICURVE .EQ. 3) GOTO 30
C      IF(ICURVE .EQ. 4) GOTO 40
C      IF(ICURVE .EQ. 5) GOTO 50
C      IF(ICURVE .EQ. 6) GOTO 60
C      IF(ICURVE .EQ. 7) GOTO 70
      WRITE(7,200)
200  FORMAT(/,10X,'ERROR - INCORRECT STANDARD SLAB PLAN CURVE')
      STOP

C
C      STANDARD PLAN CS-2-15
C
10  DO 11 I=1,5
      MAXS(I) = MAXA(I)
      ALPHAS(I) = ALPHAA(I)
      EXTS(I) = EXTA(I)
      INTS(I) = INTA(I)
      NEGS(I) = NEGA(I)
11  CONTINUE
      RETURN

C
C      STANDARD PLAN CS-2-20

```



```

C
20 DO 21 I=1,5
    MAXS(I) = MAXB(I)
    ALPHAS(I) = ALPHAB(I)
    EXTS(I) = EXTB(I)
    INTS(I) = INTB(I)
    NEGS(I) = NEGB(I)
21 CONTINUE
    RETURN
C
C STANDARD PLAN CS-3-20
C
30 DO 31 I=1,4
    MAXS(I) = MAXC(I)
    ALPHAS(I) = ALPHAC(I)
    EXTS(I) = EXTC(I)
    INTS(I) = INTC(I)
    NEGS(I) = NEGC(I)
31 CONTINUE
    RETURN
C
C STANDARD PLAN CS-4-20
C
40 DO 41 I=1,5
    MAXS(I) = MAXD(I)
    ALPHAS(I) = ALPHAD(I)
    EXTS(I) = EXTD(I)
    INTS(I) = INTD(I)
    NEGS(I) = NEGD(I)
41 CONTINUE
    RETURN
C
C STANDARD PLAN CS-5-20 I
C
50 DO 51 I=1,5
    MAXS(I) = MAXE(I)
    ALPHAS(I) = ALPHAE(I)
    EXTS(I) = EXTE(I)
    INTS(I) = INTE(I)
    NEGS(I) = NEGE(I)
51 CONTINUE
    RETURN
C
C STANDARD PLAN CS-20
C
60 DO 61 I=1,4
    MAXS(I) = MAXF(I)
    ALPHAS(I) = ALPHAF(I)
    EXTS(I) = EXTF(I)
    INTS(I) = INTF(I)
    NEGS(I) = NEGF(I)
61 CONTINUE
    RETURN
C
C STANDARD PLAN CS-20 I
C
70 DO 71 I=1,4
    MAXS(I) = MAXG(I)
    ALPHAS(I) = ALPHAG(I)
    EXTS(I) = EXTG(I)

```

```
INTS(1) = INTG(1)
NEGS(1) = NEGG(1)
71 CONTINUE
RETURN
C
END
```

```

$DEBUG
SUBROUTINE FIXSB(ISPLAN,SPANMX)
C
C *****
C
C   FIXSB - FIX SLAB SPAN LENGTHS
C
C   THIS SUBROUTINE FIXES THE MINIMUM AND MAXIMUM SPAN
C   LENGTH CORRESPONDING TO THE STANDARD PLANS
C
C -----
C
C   CALLS TO : NONE
C
C -----
C
C   VARIABLE DEFINITION
C
C   PASSED :
C
C       ISPLAN - STANDARD SLAB PLAN CODE
C
C   RETURNED :
C
C       SPANMX - MAXIMUM SPAN LENGTH
C
C -----
C
C   MODIFICATION LOG
C
C       07/08/1986  RAS  INITIAL CODING
C
C *****
C
C   INTEGER ISPLAN
C   REAL*8 SPANMX
C
C   IF(ISPLAN .EQ. 1)GOTO 10
C   IF(ISPLAN .EQ. 2)GOTO 10
C   IF(ISPLAN .EQ. 3)GOTO 20
C   IF(ISPLAN .EQ. 4)GOTO 10
C   IF(ISPLAN .EQ. 5)GOTO 10
C   IF(ISPLAN .EQ. 6)GOTO 20
C   IF(ISPLAN .EQ. 7)GOTO 20
C   RETURN
C
C 10  IF(SPANMX .LT. 25.D0) GOTO 15
C     IF(SPANMX .GT. 45.D0) GOTO 18
C     RETURN
C
C 15  SPANMX = 25.D0
C     WRITE(7,100)SPANMX
C     RETURN
C
C 18  SPANMX = 45.D0
C     WRITE(7,200)SPANMX
C     RETURN
C
C 20  IF(SPANMX .LT. 25.D0) GOTO 25
C     IF(SPANMX .GT. 40.D0) GOTO 28

```

```
      RETURN
C
25  SPANMX = 25.D0
    WRITE(7,100)SPANMX
    RETURN
C
28  SPANMX = 40.D0
    WRITE(7,200)SPANMX
    RETURN
C
100 FORMAT(10X,' WARNING - FIXUP IN MAX SPAN LENGTH ',F10.3)
200 FORMAT(10X,' WARNING - FIXUP IN MAX SPAN LENGTH ',F10.3)
    END
```

```

$DEBUG
SUBROUTINE TERPS(SPANMX,X,Y,VALUE)
C
C *****
C
C   TERP  - LINEAR INTERPOLATION
C
C   THIS SUBROUTINE INTERPOLATES LINEARLY BETWEEN TWO POINTS
C
C -----
C
C   CALLS TO :  NONE
C
C -----
C
C   VARIABLE DEFINITION
C
C   PASSED :
C
C       SPANMX - MAXIMUM SPAN LENGTH
C       X      - X VALUES
C       Y      - Y VALUES
C
C   RETURNED :
C
C       VALUE - INTERPOLATED VALUE
C
C -----
C
C   MODIFICATION LOG
C
C       07/07/1986  RAS  INITIAL CODING
C
C *****
C
C   INTEGER J,K
C   REAL*8 SPANMX,XK,YK,X(5),Y(5),VALUE
C
C   K=0
C   DO 100 J=1,5
C   K=K+1
C   IF(SPANMX .LT. X(J) ) GOTO 200
100 CONTINUE
C
200 XK = X(K) - X(K-1)
   YK = Y(K) - Y(K-1)
   VALUE = Y(K-1) + YK*( SPANMX - X(K-1) )/XK
   RETURN
   END

```

\$DEBUG

SUBROUTINE BRGTYP(NTYPE,NMAT,LABEL1,LABEL2,CONT)

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

\*\*\*\*\*

BRGTYP - BRIDGE TYPE ID

THIS SUBROUTINE DETERMINES THE BRIDGE TYPE ID

-----  
CALLS TO : NONE

-----  
VARIABLE DEFINITION

PASSED :

NMAT - SUPERSTRUCTURE DESIGN TYPE AND MATERIAL TYPE  
NTYPE - SUPERSTRUCTURE CONSTRUCTION TYPE

RETURNED :

LABEL1 - MATERIAL TYPE LABEL  
LABEL2 - CONSTRUCTION TYPE LABEL  
CONT - DESIGN TYPE LABEL (SIMPLE OR CONTINUOUS)

-----  
MODIFICATION LOG

06/11/1986 RAS INITIAL CODING

\*\*\*\*\*

INTEGER NTYPE,NMAT  
CHARACTER\*9 LABEL1,TYPE1(7)  
CHARACTER\*6 CONT ,DET(2)  
CHARACTER\*5 LABEL2,TYPE2(20)

DATA TYPE1/ 'REIN CONC', 'STEEL ', 'P/S CONC ', 'TIMBER ',  
1 'MASONRY ', 'ALUMINUM ', 'OTHER '/

DATA TYPE2/ 'SLAB ', 'STRG ', 'GIRD ', 'TBEAM', 'BOX-M',  
2 'BOX-S', 'FRM ', 'ORTH ', 'TRS-D', 'TRS-T',  
3 'ARC-D', 'ARC-T', 'SUS ', 'STAY ', 'MOV-L',  
4 'MOV-B', 'MOV-S', 'TUN ', 'CULVT', 'OTHER'/

DATA DET / 'SIMPLE', 'CONTIN' /

IF(NMAT .EQ. 1) LABEL1=TYPE1(1)  
IF(NMAT .EQ. 2) LABEL1=TYPE1(1)  
IF(NMAT .EQ. 3) LABEL1=TYPE1(2)  
IF(NMAT .EQ. 4) LABEL1=TYPE1(2)  
IF(NMAT .EQ. 5) LABEL1=TYPE1(3)  
IF(NMAT .EQ. 6) LABEL1=TYPE1(3)  
IF(NMAT .EQ. 7) LABEL1=TYPE1(4)  
IF(NMAT .EQ. 8) LABEL1=TYPE1(5)  
IF(NMAT .EQ. 9) LABEL1=TYPE1(6)

```

C      IF(NMAT .EQ. 0) LABEL1=TYPE1(7)
      CONT=DET(1)
      IF(NMAT .EQ. 2) CONT =DET(2)
      IF(NMAT .EQ. 4) CONT =DET(2)
      IF(NMAT .EQ. 6) CONT =DET(2)
C
      IF(NTYPE .EQ. 1)LABEL2=TYPE2(1)
      IF(NTYPE .EQ. 2)LABEL2=TYPE2(2)
      IF(NTYPE .EQ. 3)LABEL2=TYPE2(3)
      IF(NTYPE .EQ. 4)LABEL2=TYPE2(4)
      IF(NTYPE .EQ. 5)LABEL2=TYPE2(5)
      IF(NTYPE .EQ. 6)LABEL2=TYPE2(6)
      IF(NTYPE .EQ. 7)LABEL2=TYPE2(7)
      IF(NTYPE .EQ. 8)LABEL2=TYPE2(8)
      IF(NTYPE .EQ. 9)LABEL2=TYPE2(9)
      IF(NTYPE .EQ.10)LABEL2=TYPE2(10)
      IF(NTYPE .EQ.11)LABEL2=TYPE2(11)
      IF(NTYPE .EQ.12)LABEL2=TYPE2(12)
      IF(NTYPE .EQ.13)LABEL2=TYPE2(13)
      IF(NTYPE .EQ.14)LABEL2=TYPE2(14)
      IF(NTYPE .EQ.15)LABEL2=TYPE2(15)
      IF(NTYPE .EQ.16)LABEL2=TYPE2(16)
      IF(NTYPE .EQ.17)LABEL2=TYPE2(17)
      IF(NTYPE .EQ.18)LABEL2=TYPE2(18)
      IF(NTYPE .EQ.19)LABEL2=TYPE2(19)
      IF(NTYPE .EQ. 0)LABEL2=TYPE2(20)
C
      RETURN
      END

```

```

$DEBUG
SUBROUTINE LONGMT(NA, SPAN, A, B, SPANRT, CONT, NSPAN, MM, LOC)
C
C *****
C
C LONGMT - LONGITUDINAL MOMENT
C
C THIS SUBROUTINE CALCULATES THE MAXIMUM MOMENT FOR A
C GENERAL TRUCK CONFIGURATION
C
C -----
C
C CALLS TO :
C
C DZERO
C SPAN01
C SPAN02
C SPAN03
C MOVEF
C MOVEFB
C
C -----
C
C VARIABLE DEFINITION
C
C PASSED :
C
C NA - NUMBER OF AXLES
C SPAN - MAXIMUM SPAN LENGTH
C A - AXLE LOAD
C B - DISTANCES TO THE AXLES
C SPANRT - END SPAN RATIO
C CONT - CONTINUITY
C NSPAN - NUMBER OF SPANS
C
C RETURNED :
C
C MM - MAXIMUM MOMENT
C LOC - DISTANCE TO LEAD AXLE AT MAXIMUM MOMENT
C
C -----
C
C MODIFICATION LOG
C
C 06/11/1986 RAS INITIAL CODING
C
C *****
C
C INTEGER NA, NSPAN, NN, KN
C REAL*8 SPAN, A(NA), B(NA)
C REAL*8 MM(3,3), LOC(3,3)
C REAL*8 H1(31), H2(31), H3(31), SX(31)
C REAL*8 SPANAD, SPANRT
C REAL*8 XINC(41), TRUCK
C CHARACTER*6 CONT
C
C TRUCK=B(NA)
C
C CALL DZERO(H1,31,0.D0)
C CALL DZERO(H2,31,0.D0)
C CALL DZERO(H3,31,0.D0)

```



```

CALL DZERO(SX,31,0.D0)
CALL DZERO(XINC,41,0.D0)
C
C   SIMPLE BEAM MOMENT
C
NN = 11
KN = 21
SPANAD = SPAN
CALL SPAN01(SPAN,H1,H2,SX,XINC,TRUCK)
C
C   0.5 POINT
C
CALL MOVEF (KN,NA,B,XINC,A,SPANAD,SX,H1,NN,SPAN,MM(1,1),LOC(1,1))
C
C   0.4 POINT
C
CALL MOVEFB(KN,NA,B,XINC,A,SPANAD,SX,H2,NN,SPAN,MM(1,2),LOC(1,2))
IF(NSPAN .EQ. 1 .OR. CONT .EQ. 'SIMPLE') RETURN
C
IF(NSPAN .EQ. 2) GOTO 200
IF(NSPAN .GE. 3) GOTO 300
C
C   TWO SPAN MOMENT
C
200 CALL DZERO(H1,31,0.D0)
CALL DZERO(H2,31,0.D0)
CALL DZERO(H3,31,0.D0)
CALL DZERO(SX,31,0.D0)
CALL DZERO(XINC,41,0.D0)
C
NN = 21
KN = 31
SPANAD = SPAN * 2.D0
CALL SPAN02(SPAN,H1,H3,SX,XINC,TRUCK)
C
C   0.4 POINT
C
CALL MOVEFB(KN,NA,B,XINC,A,SPANAD,SX,H1,NN,SPAN,MM(2,1),LOC(2,1))
C
C   BENT
C
CALL MOVEF (KN,NA,B,XINC,A,SPANAD,SX,H3,NN,SPAN,MM(2,2),LOC(2,2))
IF(NSPAN .EQ. 2) RETURN
C
C   THREE SPAN MOMENT
C
300 CALL DZERO(H1,31,0.D0)
CALL DZERO(H2,31,0.D0)
CALL DZERO(H3,31,0.D0)
CALL DZERO(SX,31,0.D0)
CALL DZERO(XINC,41,0.D0)
C
NN = 31
KN = 41
SPANAD = SPAN * (1.D0 + 2.D0 * SPANRT)
CALL SPAN03(SPAN,H1,H2,H3,SX,SPANRT,XINC,TRUCK)
C
C   0.4 POINT SPAN 1
C
CALL MOVEFB(KN,NA,B,XINC,A,SPANAD,SX,H1,NN,SPAN,MM(3,1),LOC(3,1))

```

```
C
C 0.5 POINT SPAN 2
C CALL MOVEF (KN,NA,B,XINC,A,SPANAD,SX,H2,NN,SPAN,MM(3,2),LOC(3,2))
C
C BENT
C CALL MOVEFB(KN,NA,B,XINC,A,SPANAD,SX,H3,NN,SPAN,MM(3,3),LOC(3,3))
RETURN
END
```

```

$DEBUG
SUBROUTINE SPAN01(SPAN,H1,H2,SX,XINC,TRUCK)
C
C *****
C
C SPAN01 - ONE SPAN INFLUENCE LINES
C
C THIS SUBROUTINE STORES OFF THE INFLUENCE LINE
C COEFFICIENTS FOR A ONE SPAN BRIDGE
C
C -----
C
C CALLS TO : NONE
C
C -----
C
C VARIABLE DEFINITION
C
C PASSED :
C
C SPAN - MAXIMUM SPAN LENGTH
C TRUCK - TRUCK LENGTH
C
C RETURNED :
C
C H2 - INFLUENCE LINE AT 0.4 POINT
C H1 - INFLUENCE LINE AT 0.5 POINT
C SX - TENTH POINT OF SPAN
C XINC - INCREMENT TO MOVE TRUCK
C
C -----
C
C MODIFICATION LOG
C
C 06/11/1986 RAS INITIAL CODING
C
C *****
C
C INTEGER I
C REAL*8 SPAN,H1(1),H2(1),SX(1),COEFP4(11),COEFP5(11),TENTH(11)
C REAL*8 XX1,XX2,XINC(1)
C
C DATA TENTH / 0.0D0, 0.1D0, 0.2D0, 0.3D0, 0.4D0, 0.5D0,
1 0.6D0, 0.7D0, 0.8D0, 0.9D0, 1.0D0/
C
C DATA COEFP4/ 0.0D0, .06D0, .12D0, .18D0, .24D0, .20D0,
1 .16D0, .12D0, .08D0, .04D0, .00D0/
C
C DATA COEFP5/ 0.0D0, .05D0, .10D0, .15D0, .20D0, .25D0,
1 .20D0, .15D0, .10D0, .05D0, .00D0/
C
C DO 100 I=1,11
C SX(I)=TENTH(I)*SPAN
C H1(I)=COEFP5(I)
C H2(I)=COEFP4(I)
100 CONTINUE
C
C XX1 = SPAN*0.1D0
C XX2 = TRUCK*0.1D0
C

```

```
DO 200 I=1,11
  XINC(I) = XX1
200 CONTINUE
DO 250 I=12,21
  XINC(I) = XX2
250 CONTINUE
C
  RETURN
  END
```

\$DEBUG

SUBROUTINE SPAN02(SPAN,H1,H3,SX,XINC,TRUCK)

C \*\*\*\*\*

C SPAN02 - ONE SPAN INFLUENCE LINES

C THIS SUBROUTINE STORES OFF THE INFLUENCE LINE  
C COEFFICIENTS FOR A TWO SPAN BRIDGE

C -----  
C CALLS TO : NONE

C -----  
C VARIABLE DEFINITION

C PASSED :  
C SPAN - MAXIMUM SPAN LENGTH  
C TRUCK - TRUCK LENGTH  
C  
C RETURNED :  
C H1 - INFLUENCE LINE AT 0.4 POINT  
C H3 - INFLUENCE LINE AT THE BENT  
C SX - TENTH POINT OF SPAN  
C XINC - INCREMENT TO MOVE TRUCK

C -----  
C MODIFICATION LOG  
C  
C 06/11/1986 RAS INITIAL CODING

C \*\*\*\*\*

C INTEGER I  
C REAL\*8 SPAN,H1(1),H3(1),SX(1),CURVL1(21),CURVL3(21),COEFXD(21)  
C REAL\*8 XINC(1),TRUCK,XX1,XX2

C DATA COFXD / 0.0D0, 0.1D0, 0.2D0, 0.3D0, 0.4D0, 0.5D0,  
1 0.6D0, 0.7D0, 0.8D0, 0.9D0, 1.0D0,  
2 1.1D0, 1.2D0, 1.3D0, 1.4D0, 1.5D0,  
3 1.6D0, 1.7D0, 1.8D0, 1.9D0, 2.0D0/

C DATA CURVL1/.000D0, .0501D0, .1008D0, .1527D0, .2064D0, .1625D0,  
1 .1216D0, .0843D0, .0512D0, .0229D0, .000D0,  
2 -.0171D0,-.0288D0,-.0357D0,-.0384D0,-.0375D0,  
3 -.0336D0,-.0273D0,-.0192D0,-.0099D0, .000D0/

C DATA CURVL3/.0D0,-.02475D0,-.0480D0,-.06825D0,-.0840D0,-.09375D0,  
1 -.0960D0, -.08925D0,-.0720D0,-.04275D0,.000D0,  
2 -.04275D0,-.0720D0, -.08925D0,-.0960D0,-.09375D0,  
3 -.0840D0, -.06825D0,-.0480D0, -.02475D0, .000D0/

C XX1 = SPAN\*0.1D0  
C XX2 = TRUCK\*0.1D0

```
C
DO 100 I=1,21
  SX(I)=COEFXD(I)*SPAN
  H1(I)=CURVL1(I)
  H3(I)=CURVL3(I)
100 CONTINUE
DO 125 I=1,21
  XINC(I)= XX1
125 CONTINUE
DO 150 I=22,31
  XINC(I)= XX2
150 CONTINUE
C
  RETURN
  END
```

\$DEBUG

SUBROUTINE SPAN03(SPAN,H1,H2,H3,SX,SPANRT,XINC,TRUCK)

C \*\*\*\*\*

C SPAN03 - THREE SPAN INFLUENCE LINES

C THIS SUBROUTINE STORES OFF THE INFLUENCE LINE  
C COEFFICIENTS FOR A THREE SPAN BRIDGE

C -----  
C CALLS TO : NONE  
C -----

C VARIABLE DEFINITION

C PASSED :

C SPAN - MAXIMUM SPAN LENGTH  
C TRUCK - TRUCK LENGTH  
C SPANRT - END SPAN RATIO

C RETURNED :

C H1 - INFLUENCE LINE AT 0.4 POINT, SPAN 1  
C H2 - INFLUENCE LINE AT 0.5 POINT, SPAN 2  
C H3 - INFLUENCE LINE AT THE BENT  
C SX - TENTH POINT OF SPAN  
C XINC - INCREMENT TO MOVE TRUCK  
C -----

C MODIFICATION LOG

C 06/11/1986 RAS INITIAL CODING

C \*\*\*\*\*

C INTEGER I

REAL\*8 SPAN,H1(1),H2(1),H3(1),SX(1)  
REAL\*8 CURVL1(31),CURVL2(31),CURVL3(31)  
REAL\*8 CURVQ1(31),CURVQ2(31),CURVQ3(31)  
REAL\*8 CURVH1(31),CURVH2(31),CURVH3(31)  
REAL\*8 COEFXD(31),DD1,DD2,DD3,DD4,SPANRT  
REAL\*8 XINC(1),TRUCK,XX1,XX2,XX3

C DATA COEFXD / 0.0D0, 0.1D0, 0.2D0, 0.3D0, 0.4D0, 0.5D0,  
1 0.6D0, 0.7D0, 0.8D0, 0.9D0, 1.0D0,  
2 0.1D0, 0.2D0, 0.3D0, 0.4D0, 0.5D0,  
3 0.6D0, 0.7D0, 0.8D0, 0.9D0, 1.0D0,  
4 0.1D0, 0.2D0, 0.3D0, 0.4D0, 0.5D0,  
5 0.6D0, 0.7D0, 0.8D0, 0.9D0, 1.0D0/

C DATA CURVL1/0.0D0, .04944D0, .09952D0, .15088D0, .20416D0, .1600D0,  
1 .11904D0, .08192D0, .04928D0, .02176D0, .000D0,  
2 -.01560D0,-.02560D0,-.03080D0,-.03200D0,-.0300D0,  
3 -.02560D0,-.01960D0,-.01280D0,-.00600D0, .000D0,  
4 .00456D0, .00768D0, .00952D0, .01024D0, .0100D0,

```

5          .0089600, .0072800, .0051200, .0026400, .00000/
C
DATA CURVL2/0.DO,-.009900,-.019200,-.027300,-.033600,-.037500,
1          -.038400,-.035700,-.028800,-.017100, .00000,
2          .0230000, .0520000, .0870000, .1280000, .175000,
3          .1280000, .0870000, .0520000, .0230000, .00000,
4          -.0171000,-.0288000,-.035700,-.0384000,-.037500,
5          -.0336000,-.0273000,-.0192000,-.0099000, .00000/
C
DATA CURVL3/0.DO,-.0264000,-.0512000,-.0728000,-.0896000,-.100000,
1          -.1024000,-.0952000,-.0768000,-.0456000, .00000,
2          -.0390000,-.0640000,-.0770000,-.0800000,-.075000,
3          -.0640000,-.0490000,-.0320000,-.0150000, .00000,
4          .0114000, .0192000, .023800, .0256000, .025000,
5          .0224000, .0182000, .0128000, .0066000, .00000/
C
DATA CURVQ1/0.DO, .0380700, .0765600, .1158900, .1564800, .1237500,
1          .0931200, .0650100, .0398400, .0180300, .00000,
2          -.0177600,-.0290100,-.0347200,-.0358400,-.0333300,
3          -.0281600,-.0212800,-.0136500,-.0062400, .00000,
4          .0034200, .0057600, .0071400, .0076800, .007500,
5          .0067200, .0054600, .0038400, .0019800, .00000/
C
DATA CURVQ2/0.DO,-.0061900,-.0120000,-.0170600,-.021000,-.0234400,
1          -.0240000,-.0223100,-.0180000,-.0106900, .00000,
2          .0200000, .0466700, .0800000, .1200000, .1666700,
3          .1200000, .0800000, .0466700, .0200000, .00000,
4          -.0106900,-.0180000,-.0223100,-.0240000,-.0234400,
5          -.0210000,-.0170600,-.0120000,-.0061900, .00000/
C
DATA CURVQ3/0.DO,-.0173300,-.0336000,-.0477800,-.058800,-.0656300,
1          -.0672000,-.0624800,-.0504000,-.0299300, .00000,
2          -.0444000,-.0725300,-.0868000,-.0896000,-.0833300,
3          -.0704000,-.0532000,-.0341300,-.0156000, .00000,
4          .0085500, .0144000, .0178500, .0192000, .0187500,
5          .0168000, .0136500, .0096000, .0049500, .00000/
C
DATA CURVH1/0.DO, .0262900, .0528000, .0797600, .1074000, .0859400,
1          .0656000, .0466100, .0292000, .0135900, .00000,
2          -.0207000,-.0336000,-.0399000,-.0408000,-.0375000,
3          -.0312000,-.0231000,-.0144000,-.0063000, .00000,
4          .0021400, .0036000, .0044600, .0048000, .0046900,
5          .0042000, .0034100, .0024000, .0012400, .00000/
C
DATA CURVH2/0.DO,-.0030900,-.0060000,-.0085300,-.010500,-.0117200,
1          -.0120000,-.0111600,-.0090000,-.0053400, .00000,
2          .0162500, .0400000, .0712500, .1100000, .1562500,
3          .1100000, .0712500, .0400000, .0162500, .00000,
4          -.0053400,-.0090000,-.0111600,-.0120000,-.0117200,
5          -.0105000,-.0085300,-.0060000,-.0030900, .00000/
C
DATA CURVH3/0.DO,-.0092800,-.0180000,-.0255900,-.031500,-.0351600,
1          -.0360000,-.0334700,-.0270000,-.0160300, .00000,
2          -.0517500,-.0840000,-.0997500,-.1020000,-.0937500,
3          -.0780000,-.0577500,-.0360000,-.0157500, .00000,
4          .0053400, .0090000, .0111600, .0120000, .0117200,
5          .0105000, .0085300, .0060000, .0030900, .00000/

```

```

XX1 = SPAN*SPANRT*0.100
XX2 = SPAN*0.100

```



```

      XX3 = TRUCK*0.1D0
C
      IF (SPANRT .GE. 0.75D0) GOTO 100
      IF (SPANRT .GE. 0.50D0) GOTO 200
      GOTO 300
C
100  DD4 = (SPANRT-0.75D0)/0.25D0
      DO 150 I=1,31
          DD1 = CURVL1(I)-CURVQ1(I)
          DD2 = CURVL2(I)-CURVQ2(I)
          DD3 = CURVL3(I)-CURVQ3(I)
          H1(I)=CURVQ1(I) + DD1*DD4
          H2(I)=CURVQ2(I) + DD2*DD4
          H3(I)=CURVQ3(I) + DD3*DD4
150  CONTINUE
      GOTO 500
C
200  DD4 = (SPANRT-0.50D0)/0.25D0
      DO 250 I=1,31
          DD1 = CURVQ1(I)-CURVH1(I)
          DD2 = CURVQ2(I)-CURVH2(I)
          DD3 = CURVQ3(I)-CURVH3(I)
          H1(I)=CURVH1(I) + DD1*DD4
          H2(I)=CURVH2(I) + DD2*DD4
          H3(I)=CURVH3(I) + DD3*DD4
250  CONTINUE
      GOTO 500
C
300  DO 350 I=1,31
          H1(I)=CURVH1(I)
          H2(I)=CURVH2(I)
          H3(I)=CURVH3(I)
350  CONTINUE
      GOTO 500
C
500  SX(1)= COEFD(1)
      XINC(1)=XX1
      DO 510 I=2,11
          SX(I)=SX(I) + COEFD(I)*SPAN*SPANRT
          XINC(I) = XX1
510  CONTINUE
      DO 520 I=12,21
          SX(I)=SX(11) + COEFD(I)*SPAN
          XINC(I) = XX2
520  CONTINUE
      DO 530 I=22,31
          SX(I)=SX(21) + COEFD(I)*SPAN*SPANRT
          XINC(I) = XX1
530  CONTINUE
      DO 540 I=32,41
          XINC(I) = XX3
540  CONTINUE
      RETURN
      END

```

```

$DEBUG
SUBROUTINE TERP(XDIS,X,Y,FX,NN)
C
C *****
C
C   TERP  -  LINEAR INTERPOLATION
C
C   THIS SUBROUTINE INTERPOLATES LINEARLY BETWEEN TWO POINTS
C
C -----
C
C   CALLS TO :  NONE
C
C -----
C
C   VARIABLE DEFINITION
C
C   PASSED :
C
C       XDIS  -  DISTANCE TO INTERPOLATE AT
C       X      -  X VALUES
C       Y      -  Y VALUES
C       NN     -  NUMBER OF POINTS TO INTERPOLATE BETWEEN
C
C   RETURNED :
C
C       FX    -  INTERPOLATED VALUE
C
C -----
C
C   MODIFICATION LOG
C
C       06/11/1986  RAS  INITIAL CODING
C
C *****
C
C   INTEGER J,K,NN
C   REAL*8 XDIS,XK,YK,X(NN),Y(NN),FX
C
C   K=0
C   DO 100 J=1,NN
C   K=K+1
C   IF(XDIS .LT. X(J) ) GOTO 200
100 CONTINUE
C
200 XK = X(K) - X(K-1)
   YK = Y(K) - Y(K-1)
   FX = Y(K-1) + YK*( XDIS - X(K-1) )/XK
   RETURN
   END

```

```

$DEBUG
SUBROUTINE MOVEF(KN,NA,B,XINC,A,SPANAD,SX,CURV,NN,SPAN,MOMENT,
1 LOCATE)

```

```

C *****
C

```

```

C MOVEF - MOVE FOWARD

```

```

C THIS SUBROUTINE MOVES THE TRUCK FOWARD

```

```

C -----

```

```

C CALLS TO : TERP

```

```

C -----

```

```

C VARIABLE DEFINITION

```

```

C PASSED :

```

- C KN - NUMBER OF TENTH POINTS ALONG BRIDGE AND TRUCK LENGTH
- C NA - NUMBER OF AXLES
- C XINC - INCREMENT TO MOVE TRUCK
- C A - AXLE LOADS
- C B - DISTANCES TO THE AXLES
- C SPANAD - BRIDGE LENGTH
- C SX - DISTANCE AT THE TENTH POINTS
- C CURV - INFLUENCE CURV
- C NN - NUMBER OF TENTH POINTS ALONG BRIDGE
- C SPAN - MAXIMUM SPAN LENGTH

```

C RETURNED :

```

- C MOMENT - MAXIMUM MOMENT
- C LOCATE - LOCATION OF TRUCK AT MAXIMUM MOMENT

```

C -----

```

```

C MODIFICATION LOG

```

```

C 06/11/1986 RAS INITIAL CODING

```

```

C *****

```

```

C INTEGER NA,NN,KN,K,I
C REAL*8 B(1),XINC(1),A(1),SPANAD,SX(1),CURV(NN),SPAN
C REAL*8 MOMENT,LOCATE
C REAL*8 SUM,XDIS,XLEAD,FX,P,XMMT

```

```

C MOMENT = 0.D0
C SUM= 0.D0

```

```

C DO 200 J=1,NA
C XLEAD= B(J) - XINC(1)

```

```

C DO 190 K=1,KN
C XLEAD=XLEAD + XINC(K)

```

```

C DO 150 I=1,NA
C P = A(I)

```

```

      XDIS = XLEAD - B(1)
      FX = 0.D0
      IF(XDIS.GT.1.D-4 .AND. XDIS.LE.SPANAD)
        CALL TERP(XDIS,SX,CURV,FX,NN)
      XMMT= SPAN*FX*P
      SUM = XMMT + SUM
C900    WRITE(6,900)SUM,XMMT,SPAN,FX,P,XDIS,XLEAD
      150    FORMAT(7F10.3)
C        CONTINUE
C        IF(DABS(SUM) .LT. DABS(MOMENT))GOTO 175
      MOMENT = SUM
      LOCATE = XLEAD
      175    SUM = 0.D0
C        CONTINUE
      190    CONTINUE
C        CONTINUE
      200    RETURN
      END

```

```

$DEBUG
SUBROUTINE MOVEFB(KN,NA,B,XINC,A,SPANAD,SX,CURV,NN,SPAN,MOMENT,
1 LOCATE)
C
C *****
C
C MOVEFB - MOVE FOWARD AND BACKWARD
C
C THIS SUBROUTINE MOVES THE TRUCK FOWARD AND BACKWARD
C
C -----
C
C CALLS TO : TERP
C
C -----
C
C VARIABLE DEFINITION
C
C PASSED :
C
C KN - NUMBER OF TENTH POINTS ALONG BRIDGE AND TRUCK LENGTH
C NA - NUMBER OF AXLES
C XINC - INCREMENT TO MOVE TRUCK
C A - AXLE LOADS
C B - DISTANCES TO THE AXLES
C SPANAD - BRIDGE LENGTH
C SX - DISTANCE AT THE TENTH POINTS
C CURV - INFLUENCE CURV
C NN - NUMBER OF TENTH POINTS ALONG BRIDGE
C SPAN - MAXIMUM SPAN LENGTH
C
C RETURNED :
C
C MOMENT - MAXIMUM MOMENT
C LOCATE - LOCATION OF TRUCK AT MAXIMUM MOMENT
C
C -----
C
C MODIFICATION LOG
C
C 06/11/1986 -RAS INITIAL CODING
C
C *****
C
C INTEGER NA,NN,KN,K,I
C REAL*8 B(1),XINC(1),A(1),SPANAD,SX(1),CURV(NN),SPAN
C REAL*8 MOMENT,LOCATE
C REAL*8 SUM,XDIS,XLEAD,FX,P,XMNT
C
C *****
C
C FOWARD *
C *****
C
C WRITE(6,6)LOCATE
C6 FORMAT (' LOCATE -PASSED THRU MOVEFB',F10.2)
C MOMENT = 0.00
C SUM= 0.00
C
C
C DO 200 J=1,NA
C XLEAD= B(J) - XINC(1)

```

```

C      DO 190 K=1,KN
      XLEAD=XLEAD + XINC(K)
C
C      DO 150 I=1,NA
      P      = A(I)
      XDIS  = XLEAD - B(I)
      FX    = 0.DO
      IF(XDIS.GT.1.D-4 .AND. XDIS.LE.SPANAD)
1          CALL TERP(XDIS,SX,CURV,FX,NN)
      XMMT= SPAN*FX*P
      SUM  = XMMT + SUM
C      WRITE(6,900)SUM,XMMT,SPAN,FX,P,XDIS,XLEAD
C900   FORMAT(7F10.3)
      150   CONTINUE
C
      IF(DABS(SUM) .LT. DABS(MOMENT))GOTO 175
      MOMENT = SUM
      LOCATE = XLEAD
      175   SUM = 0.DO
C
      190   CONTINUE
C
      200   CONTINUE
C      WRITE(6,4)MOMENT
C4     FORMAT (' MOMENT -BETWEEN           ',F10.2)
C      WRITE(6,5)LOCATE
C5     FORMAT (' LOCATE -BETWEEN           ',F10.2)
C
C *****
C      BACKWARD *
C *****
C
      SUM  = 0.DO
C
C      DO 400 J=1,NA
      XLEAD=-B(J) + SPANAD + XINC(1)
C
      DO 390 K=1,KN
      XLEAD=XLEAD - XINC(K)
C
C      DO 350 I=1,NA
      P      = A(I)
      XDIS  = XLEAD +B(I)
      FX    = 0.DO
      IF(XDIS.GT.1.D-4 .AND. XDIS.LE.SPANAD)
1          CALL TERP(XDIS,SX,CURV,FX,NN)
      XMMT= SPAN*FX*P
      SUM  = XMMT + SUM
C      WRITE(6,1000)SUM,XMMT,SPAN,FX,P,XDIS,XLEAD
C1000  FORMAT(7F10.3)
      350   CONTINUE
C
      IF(DABS(SUM) .LT. DABS(MOMENT) )GOTO 375
      MOMENT = SUM
      LOCATE = XLEAD
      375   SUM = 0.DO
C
      390   CONTINUE
C

```

```
400 CONTINUE
C      WRITE(6,998)MOMENT
C998   FORMAT (' MOMENT -AFTER CALC          ',F10.2)
C      WRITE(6,999)LOCATE
C999   FORMAT(' AFTER CALC IN MOVEFB',F10.2)
RETURN
END
```

```

$DEBUG
  SUBROUTINE EXIT
C
C *****
C
C   EXIT   - EXIT PROGRAM
C
C   THIS SUBROUTINE CLOSES THE FILES
C
C -----
C
C   CALLS TO : NONE
C
C -----
C
C   VARIABLE DEFINITION
C
C   PASSED : NONE
C
C   RETURNED : NONE
C
C -----
C
C   MODIFICATION LOG
C
C   06/11/1986  RAS  INITIAL CODING
C
C *****
C
C   CLOSE(5)
C   CLOSE(6)
C   CLOSE(7)
C   CLOSE(8)
C   STOP
C   END

```