DCL
( TTP,IXL,IXR,K ) FIXED BIN(31,0), /* SCRATCH VARIABLES */
SHRD CHAR(27) VAR, /* LETTERS OF SECTYPES WANTED */
SECS CHAR(27) VAR, /* LEGITIMATE SECTION TYPE CHARS */
SERRT(0 .. 1) CHAR(3) STATIC INIT('E='), /* FOR ERR/NOERR SECTION DUMP HEADER MSG SWITCH */
ILLEGAL STATIC INIT(0), /* COUNTER FOR ILLEGAL CHAR STATS */
IRENCNT FIXED BIN(31,0) STATIC INIT(0), /* STATISTIC */
/* TOTAL OF ILLEGAL DELIM ERROR MESSAGES */
NOPT INIT(1) STATIC FIXED BIN(31,0), /* NEXT LINK IN */
/* OPTION CHAIN */
CORBIT STATIC INIT(1) /* CORRECTION TEXT SKIP SWUtCH. */
FIXED BIN(15,0),
NOUT STATIC INIT(0) FIXED BIN(31,0), /* TOTAL NO. OF */
/* OUTPUT CHARS */
CAPCHK CHAR(80), /* FOR COUNTING UPPERCASE CHARs. */
/* CAPDEF IS DEFINED ON CAPCHK */
CAPDEF(80) CHAR(1) DEF CAPCHK, /* FOR COUNTING UPPERCASE */
NCAPS STATIC FIXED BIN(31,0) INIT(0), /* STATISTICS - */
/* NUMBER OF UPPERCASE CHARs IN DATA, ACCUMULATOR */
NCHARS STATIC FIXED BIN(31,0) INIT(0), /* STATISTICS - */
/* NUMBER OF CHARACTERS OF LOGICAL DATA INPUT, EXCLUDIN */
/* TYPIST ENTRIES, AND INCLUDING EXTRA BLANKS. EXCEPT */
/* FOR EXTRa BLANK LINES. */
M INIT(0) STATIC /* STRUCTURED OUTPUT RECORD COUNTER */
FIXED BIN(31,0), /* FOR CROSS-REFERENCE. */
INDEXS STATIC INIT(0) FIXED BIN(31,0), /* STATISTICS - */
/* NUMBER OF INDEX ENTRIES */
TP CHAR(1), /* CURRENT TYPIST'S LETTER */
CD CHAR(1), /* TEMP. */
PBIT BIT(1), /* PUNCTUATION BIT. */
QBIT BIT(1), /* QUOTE BIT (SINGLE OR DOUBLE). */
A CHAR(80) VAR, /* INPUT BUFFER. */
B CHAR(3625) VAR, /* WORK BUFFER. */
F CHAR(80) VAR, /* TEMP. */
STR CHAR(3625) VAR INIT(' '), /* STRUCTURED OUTPUT BUFR */
TR CHAR(080) VAR, /* TRANSLATE BUFFER. */
C CHAR(140) VAR, /* INDEX ENTRY BUFFER. */
CC CHAR(1), /* TEMP FOR DELIMITED CHARACTERS. */
DTMP CHAR(2), /* BASE FOR CTMP AND ETMP OVERLAYS. */
CTMP CHAR(1) DEF DTMP, /* TEMP. */
ETMP CHAR(1) DEF DTMP POS(2), /* TEMP. */
ROMAN CHAR(4) VAR, /* POTENTIAL ROMAN PAGE NUMBER. */
ROMAN CHAR(4) VAR, /* ACTUAL CURRENT ROMAN PAGE NO. */
DAY CHAR(2), /* DAY STRING. */
MMTHDAY CHAR(2), /* MONTH STRING. */
YRNDAY CHAR(4), /* YEAR. */
TDATE CHAR(10), /* TEMP FOR POTENTIAL DATE AREA. */
TDATE(10) CHAR(1) DEF TDATE, /* ACCESS TO TDATE CHARs */
DATEMP CHAR(10) VAR, /* TEMP FOR POTENTIAL DATE CHARS */
ICE INIT(0), /* ICE IS DEFINED ON ICE, POS(2). */
ICE CHAR(1) DEF ICE POS(2), /* POS ICE FILLING ICE. */
CP CHAR(1) DEF TCP POS(2), /* FOR PAGE NO. CONVERT */
THEN GOTO MORE   /* THAT HAS ALREADY BEEN FOUND THEN */
ND=INDEX(STRUBR(TR,NC+1),STAR)   /* ELSE SEARCH FOR NEXT DLM */
IF ND=0 THEN
   /* IF NONE LEFT IN BUFFER */
MORE..DO...
   /* THEN GET MORE INPUT */
B=SUBS(B,LTT)   /* DELETE ALREADY PROCESSED DATA */
   /* FROM WORK BUFFER EXCEPT LAST DLM */
IX=LENGTH(B)+1   /* IX NOW PTS TO WHAT WILL BE THE */
   /* CHAR IN B WHICH WILL CORRESPOND TO WHAT WILL BE THE */
   /* 1ST CHAR OF THE TR OVERLAY ON E. */
LTT=1   /* RESET PTR TO MOST RECENT DELIM SCANNED. */
NC=0   /* RESET PTR TO LAST IF CHAR ALREADY SCANNED. */
REREAD...   /* GET PRIMARY INPUT. */
IF IOPT(7) THEN DO...
    NPAGESAV=NPAGE   /* SAVE FOR PAGE CHANGE CHECK. */
    GET FILE(CLIN) EDIT(TP,NEAGE,IEGCNT,J,LEN,A)
(RFORM)...
    RFORM..FORMAT(A(1),F(4),F(5),F(6),F(4),A(16))
    IF NEAGE NE NPAGESAV THEN IF IOPT(6) THEN CALL PAGELCG.. 
    IF IOPT(3) THEN   /* LIST INPUT LINES CFTION. */
        CALL LINCLOG,
    IF IPGCNT LT 0 THEN SIGNAL ENDFILE(IN)   /* END */
    ELSE DO...   /* FOR STRAIGHT I/O INPUT. */
        READ FILE(IN) INTO(A)   /* READ ONE LINE. */
        IF A=(80) THEN GOTO REREAD   /* DISREGARD BLANK LINES. */
        J=J+1   /* END. */
        IF J=NOPT THEN DO...   /* FOR OPTION CHAIN. */
ENTER...
    /* ENTRY TO MAIN LOGC. */
    GET FILE(SYSIN) EDIT(OPAR,NOFT,SECS,SMOD)
(B(16),F(7),A(27),A(30))...
    K=INDEX(SMOD,'1')   /* FIND BLANK AFTER SECTYPE LETTERS */
    SMOD=SUBSTR(SMOD,1,K-1)   /* CHOP OFF BLANKS. */
    K=INDEX(SECS,'1')   /* SEARCH FOR BLANK ONLY FOR SECONDS */
    SECS=SUBSTR(SECS,1,K-1)   /* CHOP OFF EXTRA BLANKS. */
    IF IOPT(1) THEN SIGNAL ENDFILE(IN)   /* STOP OPTION. */
END...
    IF IOPT(15) THEN GOTO REREAD...   /* THE LINE SKIP CFTION. */
    IF IOPT(7) THEN GOTO CLEANSIN...
    IF IOPT(11) THEN DO...   /* ILLEGAL CHARACTER CHECK. */
        BEXCHKS..K=VERIFY(A,LEGITS)  /* SEARCH FOR ILLEGAL CHAR. */
        IF K NE O THEN DO...
            IF NCOUNT THEN...
            CP=SUBSTR(A,K-1)   /* GET ILLEGAL CHAR INTO CP EICI. */
            PUT FILE(SYSPRINT) SKIP(2) EDIT
               (**ILLEGAL CHAR,***,CP,**,IN LINE,J,IIP)
            SUBSTR(A,K-1)="?"   /* CHANGE ILLEGAL CHAR TO QUESTION */
            /* MARK TO CHK FOR OTHER ILLEGAL CHARS ON SAME LINE. */
            ILLEGAL=ILLEGAL+1   /* COUNT ILLEGAL CHARS FOR STATS. */
            GOTO BEXCHKS..END...
        IF IOPT(3) THEN   /* LIST INPUT LINES CFTION. */
            CALL LINCLOG...
        NUL=75   /* CURRENTLY PRESUMED NUMBER OF */
           /* SIGNIFICANT CHARACTERS IN LINE. */
    NULLS..IXR=INDEX(A,'1')   /* SEARCH FOR A NULL CHAR (PREFIX). */
    IF IXR GT 0 THEN DO...   /* IF FOUND ONE THEN DO */
        A=SUBSTR(A,1,IXR-1) CAT SUBSTR(A,IXR+1)   /* DELETE IT. */
        NUL=NUL-1   /* COUNT THE DELETION DOWN. */
        GOTO NULLS..END...   /* CHECK IF MORE NULLS. */
        DTM=0   /* TEMP, SEE BLOCK SHOULD. */
        /* INDICATES NO TYPEIST ENTITY FOUND */
        IF DTM=0 THEN...   /* YET ON THIS LINE, CHECK FOLLOW. */
            IF IOPT(11) THEN...   /* FIRST PART OF TYPEIST'S ENTITY CHECK. */
            IF VERIFY(ETM,CAPS)=0   /* AND 2ND PART OF CHECK. */
                THEN...   /* PROCESS APPARENT TYPEIST ENTITY. */
            IF IPGCNT NE 30 THEN PUT FILE(SYSPRINT) SKIP(1) EDIT
               (********APPARENT TYPEIST ENTITY IS,IEGCNT,
LINES FROM PREVIOUS LINE, 'J', PREVIOUS PAGE WAS.

NPAGE = IPAGE

ITP = ITP + 2

NPAGE = IPAGE

IF ITP > 1 THEN DO

/* IF EVEN PARTLY LEGIT */

/* PAGE NO. EXISTS */

GET STRING (SUBSTR(A, 3, ITP - 1)) EDIT (IPAGE) (F(ITP-1))

IF IPAGE NE NPAGE THEN IF IPAGE OE 0 THEN

PUT FILE(SYSPRINT) SKIP(1) EDIT

('$$$$$$ BAD PAGE SEQUENCE', NPAGE, IPAGE, 'IN LINE', 'J')

/* END */

IF ITP LE 1 OR IXL = 0 THEN

PUT FILE(SYSPRINT) SKIP(1) EDIT

('$$$$$$ BAD TYPIST ENTRY', 'LINE', 'J', 'A')

END

/* END OF TYPIST'S ENTRY PROCESSING */

IF CORBIT = 0 THEN GOTO REREAD

/* SKIPPING CORRECTION TEXT */

IXR = INDEX(A, '@@!')

/* SRCH FOR TRIPLE '@!' */

IF IXR OE 0 THEN DO

/* IF FOUND LINE DELETE THEN DO */

F = SUBSTR(A, MIN(IXR+3, LENGTH(A)))

/* GET REMAINDER OF LINE. */

IF VERIFY(F, '!') NE 0 THEN PUT FILE(SYSPRINT) SKIP(1) EDIT

('$$$$$$ BAD TRIPLE '"' AT"' IN REC', 'J', 'A')

GOTO REREAD

END

/* SKIP FURTHER PROCESSING OF LINE. */

IF DTMP = '"' THEN SUBSTR(A, 1, 2) = '"'

/* REPLACES 2 */

/* '"' SIGNS AT BEGINNING OF LINE WITH SPACES. */

IF DTMP = '"' THEN SUBSTR(A, 1, 1) = ' ';

/* 'AT'S IN 1ST COLUMN TO BLANKS. */

IF DTMP = '"' THEN END

/* 'AT'S IN 1ST COLUMN TO BLANKS. */

DUBLQUT

/* DOUBLE QUOTE CLEANUP. */

IXR = INDEX(A, '????')

/* SRCH FOR A DOUBLE QUOTE. */

IF IXR GT 0 THEN DO

/* IF FOUND A PAIR */

A = SUBSTR(A, 1, IXR - 1) CAT '"' CAT SUBSTR(A, IXR+2)

GOTO DUBLQUT

END

/* CHECK FOR MORE. */

DO IXL = LENGTH(A) TO LENGTH(A) - 4

/* SRCH LAST FIVE CHARS */

IF SUBSTR(A, IXL, 1) NE '"' THEN DO

/* OF LINE AND */

NUL = IXL

GOTO CHOPOFF

END

/* BLANK THEN INCLUDE IT & CHAR TO ITS LEFT. */

CHOPOFF

A = SUBSTR(A, ITP + 1, NUL - ITP)

/* CHECK OFF TRAILING */

/* BLINKS, AND LEADING TYPIST ENTRY, IF ANY. */

/* ITP WILL BE 0 IF NO TYPIST ENTRY ON LINE. */

DEBLATSCH

/* REENTRY POINT AFTER ENDFILE(IN) ON UNIT. */

IXR = INDEX(A, '@@')

/* SEARCH FOR DOUBLE 'AT'. */

IF IXR NE 0 THEN DO

/* IF FIND DOUCLE 'AT' THEN DO */

IXR = IXR + 2

/* PIRS TO GUESS WHERE. */

BLNKSCH

/* DOUBLE 'AT' UNTIL START OF LINE. */

IXL = IXL - 1

/* DECREMENT CHAR PTR. */

IF IXL GT 0 THEN IF SUBSTR(A, IXL, 1) NE '"' THEN GOTO BLNKSCH

A = SUBSTR(A, 1, IXL) CAT SUBSTR(A, MIN(IXR, LENGTH(A)));

MAX (0, LENGTH(A) - IXR + 1)

/* DELETE WORD. */

GOTO DEBLATSCH

END

/* TOP OF LOOP FOR SINGLE 'AT'S. */

ONEAT
IXR=INDEX(A,'@') /* SEARCH FOR A SINGLE 'AT'. */ IF IXR GT 0 THEN DO /* IF FOUND ONE THEN */ IF IXR LT LENGTH(A) THEN F=SUBSTR(A,IXR+1) ELSE F='' /* AVOIDS STEG PROBLEM. */ ELSE IF IXR LE 2 THEN A=F ELSE A=SUBSTR(A,1,IXR-2) CAT F GOTO ONEAT END /* CHECK FOR MORE SINGLE 'AT'S. */ CLEANIN
IF IOPT(14) THEN DO LEN=LENGTH(A)
PUT FILE(CLOUT) EDIT(TP,NPAGE,IPGCNT,J,L,A) (E(RFORM)) END,
CAPCHK=TRANSLATE(A, /* FCF UPPERCASE CHAR COUNT STATS */ 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA''
ABCDEFGHIJKLMNOPQRSTUVWXYZ+/@#$8!()[-][":%]
DO IXL=1 TO LENGTH(A) /* UPPERCASE CHAR COUNTING ICCP */ IF CAPDF(I XL)='A' THEN NCAPS=NCAPS+1 END.
NCAPS=NCAPS+LENGTH(A) /* COUNT CLEANED INPUT */ E=E' CAT A ' /* ADD CLEANED INPUT TO WORK BUFR */ TR=
TRANSLATE(A,'123456789',')='(1)[%'] /* ABOVE STATEMENT SETS UP THE DELM BUFFER */ GOTO R E S C H /* CONTINUE SEARCHING FOR A DLM */ END /* END OF INPUT COROUTINE */ ND=ND+NC /* SET UP ACTUAL PTR TO DLM MARKER IN TR BUFR */ CC=SUBSTR(B,IXL+ND-1,1) /* GET DLM ITSELF FROM B */ ICO=IC /* SAVE PREVIOUS DLM NC. */ DO IC=1 TO 9 /* NOSUBRG */ /* SEARCH FOR NEW DLM NC. */ IF CC=DLM(IC) THEN GOTO DLMHIT /* IF THIS IS THE NUMBER */ END /* CODE OF THE CHARACTER. */ DLMHIT /* CONTINUE INITIAL PROCESSING OF ANY HIT. */ LOS=LT /* RESET PTR TO OLD LAST SCANNED CHAR. */ LTT=IXL+ND-1 /* SET UP PTR TO NEW LAST SCANNED DLM */ IF IOPT(9) THEN /* FCF DEBUG. */ PUT FILE(SYSPRINT) SKIP(1) EDIT('CC='','IC='',' MODE='',MODER,' LCS='','LOS'
(3 A,3 (A,F(5))) /* MODE=0=STRUCTURED MODE, MODE GT 0 = AN EXTRANEOUS */ /* TEXT MODE, MODE LT 0 = AN INDEX MODE, EXCEPT THAT -5 */ /* IS 'INDEX IN STRUCTURED TEXT' CODE. */ IF MODE=0 OR MODE=-5 THEN DO /* IN EXCLUDING LEFT DELM UNLESS ASTERISK */ /* AS IN THE STATEMENT BELOW (IC=1=ASTERISK). */ IF ICO=1 THEN DO /* IF START/END OF SECTION. */ CD=SUBSTR(B,LOS+1,1) /* SECTION TYPE LETTER. */ IF I END NE 0 THEN /* EXCEPT FOR DUMMY LAST SECTION. */ IF INDEX(SECS,CD)=0 THEN DO /* IF NOT LEGAL SECTYPE. */ PUT FILE(SYSPRINT) SKIP(2) EDIT('********* ILLEGAL SECTION '','CD',' IN LINE',J) (A) /
SUBSTR(B,LOS+1)=TRANSLATE(CD,S MALLS,CAES) /* ABOVE STATEMENT CONVERTS POSSIBLE CAPITAL */ /* ILLEGAL SECTIONS TO SMALL LETTER SECTIONS. */ ISERR=0 END /* TURNS ON ERROR MESSAGE SWITCH. */ K=3 /* INITIAL POINTER FOR DOUBLE BLANK SEARCH. */ TWOBLNK /* TOP OF DOUBLE BLANK CLEANING ICCP. */ 
IXL=0 /* IF K LT LENGTH(STR) THEN */ IXL=INDEX(SUBSTR(STR,K),')') /* SEARCH FOR A DOUBLE BLANK */ IF IXL GT 0 THEN DO /* IF FOUND DOUBLE BLANKS */ K=K+IXL /* K NOW PTS TO 2ND BLNK OF PAIR. */ DTM=SUBSTR(STR,K-3,2) /* DTM=2 CHARS PRECEDING */ /* THE PAIR OF BLANKS. CTMP = 1ST CF THESE, ETMP=2ND. */ IF ETMP='.' OR ETMP='1' OR ETMP='1' THEN GOTO TWOBLNK /* OR, ALLOW DOUBLE BLANKS TO LIVE. */ PBIT=C TMP='.' OR CTMP='1' OR CTMP='1' QBIT=ETMP='.' OR ETMP='1' IF NOT (QB IT AND PB IT) THEN STR=SUBSTR(STR+1,K-2) CAT SUBSTR(STR,K) GOTO TWOBLNK /* BOTTOM OF DOUBLE BLANK CLEANING */
IF LENGTH(STR) GT 2 THEN DO,
  STR=STR CAT (15) ' ' /* APPEND 2 EOLNS FC STRUCT.,*/
  /* ANE 13 MORE FC STRUCTURE BLK. */
  /* THUS AT LEAST 18 BYTES ON TAPE. */
  IF SUBSTR(STR,3,1)='P' THEN DO, /* IF PLAY SECTION.*/
    K=MIN(10,LENGTH(STR)-4) ; /* TEMP USED 3 TIMES.*/
    TDATE=SUBSTR(STR,4,K) ; /* GET POTENTIAL DATE AREA.*/
    IXL=0 ; /* INITIAL CHAR PTR OFFSET BY -1. */
    FS=FSRCH IXL=IXL+1 ; /* ADVANCE PTR TO NEXT CHARACTER.*/
    IF IYL LE K THEN DO, /* WHILE WITHIN DATE'S LENGTH.*/
      CE=TDATA(IYL) ; /* ICAD ICE WITH DEC VAL. */
      IF ICE=64 THEN GOTO FS; /* SKIP IF LEADING BLANK. */
      IF ICE GE 240 THEN DO, /* IF FCODE 1ST DIGIT. */
        ITR=IXL ; /* REMEMBER ITS POSITION. */
      END; /* IF ONLY A BLANK THEN */
      IML=ITP+1 ; /* GOTO GSRCH. */
      /* SEARCH FOR MORE DIGITS. */
    END ; /* WHILE IN RANGE OF TDATE. */
    CE=TDATA(ITP) ; /* USE SUBSCRIPT RATHER THAN SUBSTR */
    IF IGE 240 THEN DO, /* IF DEC DIGIT THEN DO. */
      ITR=ITP ; /* GOTO FS. */
      END ; /* REMEMBER ITS POSITION. */
    IF ICE=64 THEN DO, /* IF ONLY A BLANK THEN */
      ITP=ITP+1 ; /* GOTO GSRCH. */
    /* SEARCH FOR MORE DIGITS. */
    END ; /* IF INPAGE(2) OR (IOPT(5) AND ISERRI=0) THEN */
    PUT FILE(SYSPRINT) SKIP(1) EDIT(ISERRI,ISERRI),
      'M,STR, 0 0 0' (A,F(8),2 N) ;
    /* IF IML=ITP THEN DO, */
    /* WHILE STILL IN RANGE OF TDATE. */
    IF ICE GE 240 THEN DO, /* IF STRING LENGTH IS GT 0 THEN. */
      DDATE=SUBSTR(TDATE,IXL,K) ; /* MOVE DATE INTO TEME. */
      IF K LE 2 THEN D DATE=D DATE ; /* IF 2 THEN ONLY. */
      ELSE DO, /* IF LE 5 THEN D DATE=D DATE. */
      IF YR= SUBSTR(DDATE,K-1) ; /* YR 3 DIGITS */
      ELSE DO, /* IF LE 5 THEN M NTHDAY=D DATE. */
      IF M NTHDAY=SUBSTR(DDATE,6,2) ; /* M NTHDAY=SUBSTR(DDATE,6,2) ; */
      IF VFRIFY(SUBSTR(STR,3,1),SMAD)=0 THEN DO, /* IF A TYPE OF */
        /* SECTION SELECTED FOR OUTPUT. */
      IF IOPT(2) OR (IOPT(5) AND ISERRI=0) THEN
      IF CD='C' THEN DO, /* IF SMALL C FOLLOWS ASTERISK.*/
        IF IC=3 THEN DO, /* IF IT'S AN "=". */
          C=C CAT SUBSTR(B,LOS+IXL,LIT-LOS-IXL) ; /* ADD PIECE TO */
          END; /* IF SECTION, 1ST PEICE CF SEC. OF COURSE. */
          /* ADD PEICE TO */
        END; /* END OF CREATING INDEX. */
      ELSE IF MODE=8 THEN DO, /* IF IN AN INDEX MODE */
        IF IC=3 THEN DO, /* IF IT'S AN "=". */
          C=C CAT SUBSTR(B,LOS,LT-LCS+1) ; /* CREATE INDEX ENTRY. */
          INDN=INDN+1 ; /* COUNT INDEX ENTRIES PUT CUT. */
          IF IOPT(13) THEN
          IF MODECHNG ELSE DO ,/* GET NEXT PIECE. */
            /* MODECHNG, */
            /* IF IC=3 THEN DO, */
            END; /* ADD PEICE TO INDEX. */
          ELSE EXTRAN . . . /* AN EXTRANEOUS TXT MODE */
          CALL PAGECHK(SUBSTR(B,LOS,LT-LCS)) ; /* END. */
          MODECHNG ; /* TRANSFORMS MODE, DEPENDING ON OLD MODE & */
          /* CURRENT DLM, EVEN IF ILLEGAL IN CONTEXT. */
          MODOLD-MODE ; /* SAVE CURRENT MODE (NOT YET USED) */
          /* IF ] » 1 THEN DO, */
          /* IF CONTENTS FOLLOWING ] ARE (#, 
  "=" OR "=" THEN DO, */
          /* IF CONTENTS FOLLOWING ] ARE (#, 

NEWFRK(MODE,IC)=NEWFRK(MODE,IC)+1

/* ABOVE STMT IS FREQUENCY TABLE OF DELIMITERS FOUND */
/* IN VARIOUS MODES, ILEGAL AND LEGAL. */
IF ERRAY(MODE,IC) THEN DO .
   IF IOPT(12) THEN
      PUT FILE(SYSPRINT) SKIP(1) EDIT
      ('/','/// ILEGAL DELIM,'CC','LINE',J,'MODE=',MODE)
      (4 A,X(14),A,F(2)) ,
   IF IOPT(8) THEN PUT FILE(SYSPRINT) EDIT
      (SUBSTR(B,LCS,ITT-LOS+1)) (X(2),A) ,
   TERRCNT= TERRCNT+1 ,
   /* TOTAL COUNT OF ILEGAL DELIMS. */
   ISERRI=0 ,
   /* INDICATE DUMPING ON PRINTER OF THIS */
   END ,
   /* SECTION IF IOPT(5) IS ALSO SPECIFIED. */
   IF IEND=0 THEN DO ,
      /* IF REAL OR SIGNALLED ENDFILE(IN */
      PUT FILE(SYSPRINT) SKIP(2) EDIT
      ('INPUT LINFS=',J,
      'CLEAN INPUT CHARS=',NCHARS,
      'CLEAN UPPERCASE=',NCAPS,
      'INDEX ENTRIES=',INDEXS,
      'STRUCTURED OUTPUT CHAR=',NOUT,
      'TOTAL ILEGAL DELIMS=',TERRCNT,
      'OUTPUT SECS=',N,'ILEGAL CHAR=',ILLEGAL,
      'LOGICAL OUTPUT CHAR=',NOUT-15*M,
      'TOTAL INPUT CHAR=',80*J,
      'ACCOUNTABLE INPUT CHAR=',4*J/15+NCAES+NCHARS)
      (A,CCL(27),F(8),SKIP)
      ((DLM(IXL) DO IXL=1 TO 9)) (COI(B),9 (A,X(7)))
      ((NEWFRK(IXL,IXR) DO IXF=1 TO 9),
      (ERRAY(IXL,IXR) DO IXR=7 TO 9) DO IXL=-5 TO 4)
      (SKIP,9 F(8)) ,
   IF IOPT(14) THEN
      PUT FILE(CLOUT) EDIT(TP,NPAGE,-1,J,LEN,A) (R(EFORM)) ,
      RETURN ,
         /* END OF PROGRAM EXECUTION. */
      ND=ND ,
      GOTO RESEARCH ,
      /* LOCK FOR NEXT DL. */
      PAGESCH ,
      /* CHECKS A SEGMENT OF TEXT FOR */
      /* PAGE ENTRIES, RCMAN OR DECIMAL. */
   PROC(TXT) ,
   /* DCL TEXT CHAR(3625) VAR ,
   LP=0 ,
   /* INITIALIZE FTF (OFFSET BY -1). */
   LOCKAGIN ,
   /* TOP OF DECIMAL PAGE NUMBER LOOP. */
   IF IP LE LENGTH(TEXT)-3 THEN DO ,
      /* TST TO AVOID STG ERR. */
      TP=INDEX(SUBSTR(TEXT,LP+4),',',P') ,
      /* IF FOUND */
      /* BLINK-SCALE P */
      IF IP GT 0 THEN DO ,
      LP=LP+4 ,
      NUMP=0 ,
      DO IF=LP TO MIN(LP+4,LENGTH(TEXT)) ,
      CP=SUBSTR(TEXT,IP,1) ,
      /* GET POSSIBLE DECIMAL CHAR. */
      IF ICP GT 240 THEN IF ICP=64 THEN IF NUMP GT 0 THEN DO ,
      NEWPAGE=NUMP ,
      /* SAVE NEW PAGE NUMBER. */
      PUT FILE(SYSPRINT) SKIP(1) EDIT('NEW PAGE',NEWPAGE)
      (A,F(5)) ,
      GOTO LOCKAGIN ,
         /* CHK IF MORE PAGE NO.'S */
      ELSE GOTO LOCKAGIN ,
      IF ICP GT 249 THEN GOTO LOCKAGIN ,
      NUMP=NUMP+10+ICP-240 ,
      /* ADD IN VALUE OF NEW DIGIT ETC. */
   END ,
   PUT FILE(SYSPRINT) SKIP(1) EDIT('PAGE NUMBER TOO LONG IN CR BEFORE LINE',J) (A) ,
   ISERRI=0 ,
   /* INDICATE ERR IF SECTION LISTED. */
   GOTO LOCKAGIN ,
         /* END. */
   LP=0 ,
   /* CHECK FOR RCMAN PAGE NUMBERS. */
   IF IP LE LENGTH(TEXT)-3 THEN DO ,
      /* TST TO AVOID STG ERR. */
      IP=INDEX(SUBSTR(TEXT,LP+1),',') ,
   IF IP GT 0 THEN DO ,
      IP= TP4+LP+2 ,
      /* NUMP=0 ,
      DO IF=LP TO MIN(LP+4,LENGTH(TEXT)) ,
      CP=SUBSTR(TEXT,IP,1) ,
      IF ICP=64 THEN IF NUMP GT 0 THEN DO ,
      NEWPAGE=NUMP ,
      PUT FILE(SYSPRINT) SKIP(1) EDIT('NEW PAGE',NEWPAGE)
      (A,F(5)) ,
      GOTO LOCKAGIN ,
         /* CHK IF MORE PAGE NO.'S */
      ELSE GOTO LOCKAGIN ,
      IF ICP GT 249 THEN GOTO LOCKAGIN ,
      NUMP=NUMP+10+ICP-240 ,
      /* ADD IN VALUE OF NEW DIGIT ETC. */
   END ,
   PUT FILE(SYSPRINT) SKIP(1) EDIT('PAGE NUMBER TOO LONG IN CR BEFORE LINE',J) (A) ,
   ISERRI=0 ,
   /* INDICATE ERR IF SECTION LISTED. */
   GOTO LOCKAGIN ,
         /* END. */
   LP=0 ,
   /* CHECK FOR RCMAN PAGE NUMBERS. */
   IF IP LE LENGTH(TEXT)-3 THEN DO ,
      /* TST TO AVOID STG ERR. */
      IP=INDEX(SUBSTR(TEXT,LP+1),',') ,
   IF IP GT 0 THEN DO ,
      IP=TP4+LP+2 ,
      /* NUMP=0 ,
      DO IF=LP TO MIN(LP+4,LENGTH(TEXT)) ,
      CP=SUBSTR(TEXT,IP,1) ,
      IF ICP=64 THEN IF LENGTH(RCMAN) GT 0 THEN DO ,

ROMAN=FCMAN
/* ACCEPT NEW ROMAN NUMERAL PAGE NO. */
PUT FILE(SYSPRINT) SKIP(1) EDIT
(**** NEW PAGE 'ROMAN' (A),
ISERRT=0,
GOTO NROMAN ; END ;
IF ICP=229 OR ICP=231 OR ICP=201 THEN FCMAN=FCMAN CAT CP,
ELSE GOTO PAGEERR ; END ;
PAGERR .
/* BAD ROMAN NUMERAL PAGE ENTRY. */
PUT FILE(SYSPRINT) SKIP(1) EDIT
(*********** BAD PAGE IN OR BEFORE RECCD 'M' (A,F(5)),
GOTO NROMAN ; END ;
END PAGECHECK ;
END LILINELOG ;
*/ LINE LISTING PROCEDURE. */
PUT FILE(SYSPRINT) SKIP(1) EDIT
(A,'**',J,YRMDAY,'/',MNTHDAY,'/',DAY,'=',M)
(2 A, COL(83), F(7), X(1), 5 A, COL(101), A, F(5)),
END LILINELOG ;
PAGelog .. PROC . ; /* PCC TO LOG PAGE CHANGES ETC. */
PUT FILE(SYSPRINT) SKIP(1) EDIT
(** NEWPAGE 'TP, TNPAGE' ON LINE', J, YRMDAY, '/-',
MNTHDAY, '/', DAY, '=' M ) (5 A, X(2)),
END PAGELOG ;
END SCAN ;
/*
// STRUCT USE 999999211, WCB, MSGLEVEL=(1,1)
// S EXEC ISPSUPGEN, DSSP=OLD, SSE=20, MEM=STBCT.
// PARM, PLIL=C48, NT, A, X, EW, OPT=0, SIZE=90K
// BAKUP SYSUT1 DD *, DCB=ELKSIZE=80
/* STRUCTURE */
*/
/
* NOTE THAT, SINCE THE ONE LINE MESSAGE FOR EACH SECTION IS */
* PUT OUT AFTER MOST PROCESSING OF THE SECTION, THE ERROR */
* MESSAGES WILL GENERALLY PRECEDE THE SECTION MESSAGE LINE. */
* SEQL OUTPUT IS VERY SLOW. CORE/TIME TRADEOFF COULD VERY */
* POSSIBLY BE MADE. */
STRUCT . PHC OPTIONS(MAIN) .
G . FORMAT(A, F(4)) .
H . FORMAT(A, F(5)) .
DCL /* DECLARE STATEMENT FOR CONSTANTS & SUCH. */
MONTHSMALLS CHAR(24) STATIC INIT('abcdefghijklmnopqrstuvwxyz'),
MONTHCAPS CHAR(21) STATIC INIT('ABCDEFGHIJKLMNOPQRSTUVWXYZ'),
/* ABOVE 2 CONSTANTS ARE FOR CONVERTING NAME OF MONTH IN */
/* LACER REF TO ALL UPPER CASE. */
STYPEP(26) CHAR(1) STATIC INIT('p', 'a', 'k', 'i', 'o', 't', 'u',
'p', 'p', 'p', 'p'), /* TITLED SECTIONS ARE THOSE FROM 1 */
/* IN THIS ARRAY, UNTITLED ARE FROM 14 TO 26. */
L . /* ALL P'S EXCEPT THE FIRST ARE JUST FOR FILL. */
FA CHAR(1) INITIAL('?'), /* ITEM-START DELIMITER */
FC CHAR(1) INITIAL('[]'), /* GROUP-START DELIMITER */
ERRMSG ENTRY(CHAR(4), FIXED BIN(15,0)),
STRO ENV(REGIONAL(3)) SEQL OUTPUT KEY;
COTO ENTRY(FIXED BIN(15,0), FIXED BIN(15,0), FIXED BIN(15,0),
LABEL),
GTCK ENTRY(FIXED BIN(15,0), CHAE(1)), /* ROUTINE TO */
/* GET ITEM & DETERMINE IF IT IS A */
/* LACER ENTRY, AND PROCESS IT IF */
/* IS A LEGITIMATE ONE. */
GITM ENTRY(FIXED BIN(15,0), CHAE(1)),
SD CHAR(2) INITIAL('[]'), /* FLANK-SECTION DELIM */
SYSPRINT PRINT ENV(F(133,133)),
DCL /* ERROR MESSAGE COUNT. */
IS, /* PTR TO CURRENT SEASON-END SPECS. */
ISNOW, /* SECTION NO. WITHIN PERFORMANCE. */
ISCOUNT(20), /* SECTION NUMBER FOR SEASON END. */
ISDATE(20,3), /* SEASON-END DATES */
THEATRE FOR SEASONAL CHECKS.
MONTH STRING ARRAYS.
TIMES FOR POTENTIAL NEW DATES.
ARRAY OF ACTUAL CURRENT DATE.
IDBITS POSITION.
CHK C.
FOR GTCK'S 'SEE' CHECKING, CH3
IS DEFINED ON CH3.
FOR GTCK'S 'AS' CHECKING
CURRENT PAGE NO. AS CHAR STRING
DATE IN LADDER E
ITEM IN C
TEMP SAVE AREA FOR C
TEMP.
USED IN CGGC ONLY
C2 IS DEFINED ON MNUM.
TEMP, ON LADDER E TYPE CHAR.
C, L, OR S.
IS DEFINED ON CHOGC.
CHOGA IS DEFINED ON CHOGA.
DCL
RECORDED KEY FOR STRU FILE HERE.
BASE FOR CHIDATE'S 3 DATE CHAR.
SEE ABOVE.
OPTION ETS ARRAY
STATISTIC, MAX OUTPUT BLKSIZE.
KEEPER OF THE CURRNT PAGE.
AS ABOVE, BUT FOR MAP/LOG.
MAP/LOG.
MONTH FOR MAP/LOG.
ERRORS FOR MAP/LOG.
NEW SECTION AND PAGE.
INDICATORS FOR MAP/LOG.
FOR TITLE TIME ENTRY.
FIRST REP DONE BIT. '0'=DONE.
TITLE TIME ENTRY BIT. '1'='YES
THERE IS A TITLE TIME ENTRY.
'SECTION', '0'=MEANS OPPOSITE.
ERROR BIT='1' IF ERRMSG CALLED
DURING PROCESSING OF THIS BLK.
IF='1' B THEN CAST TIME ENTRIES NOT ALLOWED
INDICATES NON-FLY TITLE SECTION.
INBUF. P STANDS FOR 'PURE'.
OUTPUT BUFR. 'S' STANDS FOR
'STRUCTURED'. GCES INTO STRU.
CURRENT SECTYPE INDICATOR CHAR.
THEATRE ABBREVIATIONS HELD HERE SKEY & KEY
.NULL, LAD, NSEE, NCGLE) FIXED BIN (31,0) STATIC INIT(0),
ABOVE 4 VARIABLES FOR STATISTICS, NUMBER OF OUTPUT CHARS,
NUMBER OF LADDER AS REFS, SEE REFS, CGLE'S RESPECTIVELY.
OUTPUT BECCE COUNTER.
NUMBER OF CURRENT INPUT BLK.
DEBUG CHAIN LINK VARIABLE.
CURRENT REL TRK IN STRU FILE
GTCK & GTCM EXIT FOR NO ITEM CASE.
TEMP SAVE E FOR NONEXIT ABOVE.
CONVERSION GOTO CONGO.
LABEL VARIABLE SWITCH.
END.
GET FILE (SIN) EDIT(K,C,SMOD)
(P(1), A(9), A(70))
MONSTR(L) = SUBSTR(C(1), K) END /* SHORTEN INTO MONSTR. */
DO IS=1 TO 20
GET FILE(SYSIN) EDIT(ISDATE(IS,*), THTR(IS), ISCNT(IS), SMOD)
(F(4), 2 F(2), A(6), F(2), A(62))
IF ISCNT(IS)=0 THEN GOTO SEASONEND END

SEASONEND..IS=1 /* PTR TO SEASON END SPECS, INITIAL */
READ FILE(PURE) INTO(F) /* TOP OF MAIN LOOP. */
J=J+1 /* L=LENGTH(P) */
ERR=0 /* COUNTS INPUT RECORDS */
IDAYS=IDAY.. /* MONTHS=1 */
IPAGES=1 /* FOR ELK MAP */
ERRORS=1 /* FOR ELK MAP */
NOTES=1 /* FOR ELK MAP */
PUT STRING(IPAGES(0)) EDIT(IPAG) (F(4)) /* FOR ELK MAP */
PUT STRING(MONTHS(0)) EDIT(IMONTH) (F(2)) /* FOR ELK MAP */
IF J=NOPT THEN GET FILE(SYSIN) EDIT /* FOR OPTION-*/
(IOPT,NOPT,SMOD) (32 B(1), F(6), A(42)) /* CNTRL CHAIN. */
IF IOPT(25) THEN GOTO INPUT /* THE SKIP FUNCTION. */
IF IOPT(5) THEN SD=1 /* SCANNER TYPE SEC-DIM */
IF IOPT(1) THEN /* RAW INPUT DUMP OPTION */
PUT FILE(SYSPRINT) SKIP(2) EDIT(F) (A) /* FOLLOWING STATEMENT IS REPLACED BY MTST CARDS FOR USE WITH*/
/* MTST DAT. */ GOTO NOTAPE /* TOP OF SECTION PROCESSING LOCP */
NEXTSEC /* TOP OF SECTION PROCESSING LOCP */
TO=TO+1.. /* SET UP RECORDED KEY */
INC=INCR /* INCREMENT COUNTER */
IF IOPT(2) OR ERR THEN /* ONE LINE SECTION HEADER MESSAGE */
(STR)
PUT FILE(SYSPRINT) SKIP(1) EDIT
(IN='J', OUT='I', TO='E', L='I', SUBSTR(P, ISI+2, MIN(94, ISR-ISL-1)),
** , (IRDATE(L) DO L=3 TO 1 EY -1))
(COL(3), (A,F(4)), X(1), 2 A, COL(123), F(4), 2 F(3))
IF IOPT(04) OR ERR THEN
PUT FILE(SYSPRINT) SKIP(1) LIST(S)
IF LENGTH(S) GT 0 THEN /* AVOID WRITING 2FCR LENGTH RECORD */
WRITO DO.. /* GENERAL OUTPUT DCGROUP. */
NOT=NOT+LENGTH(S) /* TOTAL OUTPUT CHAS STATISTICS. */
MAXOUT=MAX(MAXOUT, LENGTH(S)) /* FOR MAX BLK SIZE STATS. */
IF IOPT(7) THEN /* OUTPUT ONE SECTION AS KEYYED */
S='; CAT SKEY CAT S' /* BLK WITH 2 BLANKS FOR MIN 18 */
/* BYTE TAPE BLKS., WITH 'KEY' IN 3. */
WRITE FILE(STRS) FROM(S) /* ONLY WAY FOR TAPE. */
S=SUBSTR(S, 15).. END /* REMOVE SKEY FROM S. */
IF IOPT(8) THEN /* OUTPUT ONE SECTION AS A KEYED PHYSICAL BLK */
WHITE FILE(STRU) FROM(S) 'KEYINCM(SKEY CAT KSTR)'.. END /* ISNOW=ISCNT(IS) THEN /* START CHECKING /* CTMP=THTR(IS) THEN /* SEASON CHANGE. */
IF IRDATE(1)=ISDATE(IS, 3) THEN /* CHECK DAY OF MONTH. */
IF IRDATE(2)=ISDATE(IS, 2) THEN /* SEASON CHANGE. */
IS=IS+1,
S=SKEY='XXXXXXXXXXX' /* END OF SEASON DELIMITER. */
PUT SKIP(3) EDIT('NEW SEASON' '***************') (A)
GOTO WRITO.. END /* IF LAST SECTION DONE */
IF ISR=1 THEN /* IF LAST SECTION DONE */
IF IOPT(3) OR ERR THEN
PUT FILE(SYSPRINT) SKIP(1) EDIT /* PRINT MAP OF INPUT BLK */
((ERRORS(L/100), L/100, SUBSTR(F, L, MIN(100, L-1))),
IPAGES(L/100),
NOTES(L/100), (K DC K=1 TO 9),
MONTHS(L/100), IDAYS(L/100) DO T=1 TO I EY 100))
(SKIP(1), A(4), F(3), X(3), A(100), X(2), A(4), X(1), A(3),
SKIP(1), COL(20), 9 FP(T), X(5)),
X(6), A(2), F(3))
GOTO INPUT.. END /* GET NEXT INPUT BLK AND PROCCES */
ISTSEC /* ENTRY POINT FOR SECTION PROCESSING LOCP */
** INITIALIZE THE OUTPUT BUFFER **

```
ISL=ISR  ;  /* SET END OF LAST SEC=START OF NEXT SECTION */
/* IF THERE IS A NEXT SECTION THEN ISL IS NOW */
/* POINTING AT THE BLANK FACING THE SECTION */
/* DELIM, ' ' OR PREFIX */
ISR=INDEX(STRSUB(PS, ISL+1), SD) ; /* GET END OF CURRENT SEC */
IF ISR=0 THEN ISR=IT-2 ; /* IF IT'S THE LAST SECTION THEN */
ELSE ISR=ISR+ISL  ; /* ACTUAL PTR TO SEC'S RIGHT END */
SMOD=STRSUB(PS, ISL+2) ; /* GET SECTION TYPE CODE */
NOTES(ISL/100)=NOTES(ISL/100) CAT SMOD ; /* NOTE NEW SEC */
BITSEC='0'B ; /* CLEAR 'TIME ENTRY ALLOWED' BIT */
ITBIT='0'B ; /* CLEAR 'NON-PLAY TITLED SECTION' BIT */
ITBIT=1 ; /* INDICATES NO ROLES OR ACTIONS PUT OUT YET */
/* IN CURRENT SECTION, HEISE DEEM CGLE'S */
ITL=1 ; /* SET ONLY TO AVOID STRG ERR IN CASE CF */
/* NOITEM INTERRUPT ON 1ST GITM CALL PCR BLK */
```

** DO NS=1 TO 26 **

```
IF STYPE(NS)=SMOD THEN GOTO SECFIND ; END ;
CALL ERRMSG('ISEC', ISL+2) ; /* ILLEGAL SECTION TYPE */
IO=IO-1 ; /* DECREMENT FECD R CUTOUT COUNTER */
GOTO NEXTSEC ; /* SINCE THIS ILLEGAL SEC SKIPPED */
```

** SECFIND **

```
IF NS=1 THEN /* IF PLAY SEC (STARTING NEW PERFORMANCE) */
ISNOW=0 ; /* RESET 'SEC NO. WITHIN PERFORMANCE' COUNTER */
ELSE IF NS LE 13 THEN ITBIT='1'B ;
/* ITBIT='1'B INDICATES A NON-PLAY TITLED SECTION */
ISNOW=ITNOW41 ; /* COUNT SEC NO.'S WITHIN RFEB */
ITR=ITL+3 ; /* SET ITR TO POINT AT START OF SEC */
/* FOLLOWING THE SCETYPE LETTER */
IGR=ISL+2 ; /* IGR NOW PTS AT SCETYPE LETTER */
IF NS GT 13 THEN GOTO NEXTGRF ; /* IF UNTITLED SECTION */
/* PROCESS DATE, THEATRE, AND TITLE */
BITSEC='1'B ; /* SPECIFY THAT THIS KIND OF SECTION */
/* HAS NO TIME ENTRIES */
S=S CAT FC CAT SMOD ; /* TITLE GBE INDICATOR */
ISC1=INDEX(STRSUB(PS, ISL+1, ISL+1)) ;
ISC2=INDEX(STRSUB(PS, ISL+1, ISL+4)) ;
IF ISC2 GT 0 AND (ISC2 LT ISC OR ISC=0) THEN ISC=ISC2 ;
IF ISC=0 THEN DO ; /* IF NO END OF TITLE DELIMITER */
CALL ERRMSG('NTL', ISL+2) ;
ISC=ISC-1 ; /* NO TITLE, SET UP WHOLE SECTION=HEALER */
END ;
```

** NONEXIT=NOITEM **

```
/* GIVE ERROR MESSAGE IF FIND NO ITEM */
ISC=ISC+ISL ; /* NOW ACTUAL END TO END OF TITLE */
IF ITBIT THEN GOTO TITLE ; /* IF NOT A PLAY SECTION */
CALL GITM(ISC, 'A') ; /* TAKE CARE OF POSSIBLE PAGE ENTRY */
ITR=ITL ; /* AND RESTORE STATUS FOR CCCE */
BIT1='0'B ;
```

** TUBITS=0 **

```
DO IX=1 TO 4 ; /* THIS LOOP GETS UP TO 3 DATE PARTS */
CALL CGOU(ITR, ISC, IPDATE(IX), DATEDONE) ;
BIT1='1'B ; /* INDICATES THAT AT LEAST ONE DATE ENTRY HAS */
/* BEEN FOUND */
```

** IF IX=1 THEN IDPTR=NBL/100 ; */
```
/* ETS TC LINE FOR DATE */
/* ENTRY LOGGING */
/* CALL ERRMSG('XDATE', IDPTR*100+1) ; */
/* 4TH PART OF DATE */
/* IX=4 ; */
/* HAS BEEN FOUND, SO SIMULATE ONLY 3 PARTS */
DATEDONE ; /* DATE UPDATING, CHECKING, AND SETTING UP */
M=IX ; /* OF DATE SIGNIFICANCE FITS OCCURS BELOW */
DO IX=1 TO IX-1 ;
```

** IF IPDATE(M) LT IDATCHK(IX, 1) OR **
```
IPDATE(M) GT IDATCHK(IX, 2) THEN
```
CALL ERMSG(IDRT, IDPTR*10041) /* PART OF DATE IS OUT OF */           
               /* PERMISSIBLE RANGE. */           
ELSE /* IF THIS PART IS INSIDE PERMISSIBLE RANGE. */           
IRDATE(IX)=IDPTR(M) /* ACCEPT THIS PART OF DATE. */           
END                
      
IDATE= (IDATE(3)-1659)*372+(IDATE(2)-1)*31 + 
       IDATE(1)-1*8+IDBITS         /* CONVERT NEW DATE TO */           
               /* CODED DATE FOR DA KEY OR EQUIV. */           
THEN /* RELATIVE DATE CHECK */           
CALL ERMSG('DCHG',ITL) /* DATE ADVANCED TOO MUCH */           
               /* OR WENT BACKWARDS. */           
 ELSE /* THEATRE AFTER A FULL DATE ENTRY. */           
THEATRE:. 
CTMP=C /* SAVE 8-BYTE THEATRE FCG RECORD KEY STRING */           
      
TITLE:. 
IF BIT1 THEN DO /* IF DATE WAS UPDATED IN THIS SEC */           
               /* THEN DO UPDATE FCG MAP/LOG. */           
PUT STRING(MONTHS(IDPTR)) EDIT(IDDATE(2)) (F(2))                       
DO K=IDPTR TO 36; IDAYS(K)=IDDATE(1); END; END; 
CALL GITM(ISC,' ') /* GET TITLE OF SECTION */           
L=INDEX(C,';') /* SEARCH FOR SEMICOLON, IF ANY */           
               /* WHICH DEFINES START OF SUBTITLE, */           
IF L GT 1 THEN /* AND IF FIND A GOOD ONE THEN */           
C=SUBSTR(C,1,L-1) /* DELETE THE SUBTITLE. */           
               /* CTRL BITS FOR TITLE TIME ENTRIES */           
M=INDEX(C,'.') /* SRCH FOR END OF TITLE TIME ENTRY */           
IF M GT 1 THEN DO /* IF FOUND POSSIBLE TITLE TIME */           
TITLTM=SUBSTR(C,1,M-1) /* GET TITLE TIME ENTRY. */           
               /* THEN DELETE, LEAVING ONLY TITLE. */           
C=SUBSTR(C,M+2) END; 
S=S CAT FA CAT SMD CAT C /* PUT OUT TITLE ITEM */           
               /* INITIALIZE OLD RIGHT GROUP POINTER */           
NXTGRP: /* TOP OF LOOP PROCESSING CAST LISTS. */           
               /* END OF DATE, THEATRE, & TITLE PROCESSING. */           
IGL=IGR /* OLD RIGHT GRP PTR = NEW LEFT GRP PTR. */           
               /* AT THE START OF A SECTION IGL FTS AT THE */           
               /* CHAR FOLLOWING THE SECTION TYPE LETTER. */           
(STRG) /* TEMP ERR CHECK, REMOVE IF YOU SEE THIS CARD */           
               /* IGR=INDEX(SUBSTR(P,IGL+1,ISL-IGL),';') */           
               /* SEMICOLON */           
IF IGR NE 0 THEN GOTO VFRSTGRP /* IF SEMICOLON WAS FOUND */           
               /* SET GROUP-END = SECTION-END */           
IF VERIFY(SUBSTR(P,IGL+1,ISL-IGL),' ;!?.-E()') NE 0 
               /* GENERALLY, IF YOU FIND A LETTER */           
               /* OR A NUMBER THEN GOTO ISTGRP */           
               /* AT THIS POINT THERE ARE APPARENTLY NO MORE GRS IN THE */           
               /* SEC, SO IF NO GRPS AT ALL WERE IN THE SECTION, AND IF */           
               /* THIS SECTION HAS A TITLE TIME ENTRY THEN PUT IT OUT. */           
               /* IF FGBIT THEN S=S CAT FC CAT 'G' CAT FA CAT 'T' CAT TITLTM, */           
               /* GOTO NEXTSEC. */           
               /* VFRSTGRP. */           
               /* ACTUAL PTR TO END OF GRP DELIM */           
               /* ISTGRP. */           
               /* S=S CAT FC CAT 'G', */           
               /* ANOTHER GROUP EXISTS, SO PROCESS */           
               /* IF BITSEC THEN GOTO PREHOLE */           
               /* * APEREICE OF PERFORMANCE SECTION */           
               /* NONEXIT=NOITEM /* EVEN IF NO TIME ENTRY THERE MUST */           
               /* BE AN ITEM SINCE THERE IS A GRP */           
               /* ISP=ITR /* SAVE TTT FC FOR REUSE IF NO TIME ENTRY */           
               /* CALL GITM(IGR,'') */           
               /* GET POTENTIAL TIME ENTRY (BLANK) */           
               /* L=LENGTH(C) */           
               /* IF SUBSTR(C,L,1) NE '!' THEN DO */           
               /* IF HAS NO COLON THEN */           
               /* ITR=ISP */           
               /* GOTO PREHOLE */           
               /* * IT'S NO TIME ENTRY, SO */           
               /* END */           
               /* SEE IF IT'S A ROLE */           
               /* K=VERIFY(C,';U123456789TVX@ADEF') */           
               /* LEGAL TIME ENTRY CHANS */           
               /* IF K NE L THEN CALL ERMSG('UNVT',ITR) */           
               /* IF THE 1ST
GOTO GITOUT ..

CHECK .. ILD(2)=L

CALL COGO(ITR,IPM,ILD(3),NCYEARE)

IF ILD(3) GT 1800 OR ILD(3) LT 1659 THEN DC ..

CALL ERRMSG('EDYR',ITR) .. GOTO GITOUT .. END ..

PUTAS ..

C=' '  /* SET C TO LENGTH OF 8. */

PUT STRING(C) EDIT(ILD) (2 F(2),F(4)) ..

IF ITBIT=0 THEN DO .. /* INDICATES CGLE. */

C3='C' .. NCGLE=NCGLE+1 .. END .. /* AND COUNTS CGLE'S. */

IF C3='L' THEN NLAD=NLAD+1 .. /* COUNTS 'AS' REFS. */

ELSE IF C3='S' THEN NSEE=NSEE+1 .. /* COUNTS 'SEE' REFS. */

S=S CAT PA CAT C3 CAT C .. /* PUT CUT LADDER ENTRY. */

ITR=ITR+1 .. NONEXIT=EXITSV .. /* RESTORE GIM'S ENV. */

/* EXCEPT FOR C, THAT EXISTED AFTER THE FIRST */

/* CALL TO GITM AFTER ENTRY TO GTCK (AFTER). */

/* THIS SHOULD SET UP THE CALL FOR THE ITEM */

/* FOLLOWING THE ENTIRE LADDER REFERENCE. */

IF ITBIT=0 THEN DO .. /* IF THIS IS A CAST GNE ENTRY THEN */

NXTIT .. /* SRC FOR ANY SYNTACTIC FACES PREVIOUSLY */

/* THOUGHT TO HAVE BEEN SYNTACTIC ACTORS. */

IP=INDEX(SUBSTR(S,IL+1),T?) .. IF IF IP GT 0 THEN DO .. /* IF YOU FIND ONE THEN ... */

SUBSTR(S,IL+1,IP,1)=R' .. /* CORRECT THE TAG. */

GOTO NXTIT .. END ..

GOTO SECNDGRP .. /* ONLY ACTORS REMAIN, ROLES DONE. */

END ..

GOTO TRY2 ..

NCYEARE .. ILD(3)=IRDATE(3) .. GOTO PUTAS ..

NCMN ..

CALL ERRMSG('NCMN',ITL) .. GOTO PUTAS ..

GITOUT ..

C=CSV .. ITR=ITR+1 .. NONEXIT=EXITSV .. RETURN ..

ENDTCK .. END GTCK ..

GITM .. PROC(LST,DLM)

/* ON NORMAL EXIT ITL POINTS TO LEFT CHAR OF ITEM, ITR PTS */

/* TO THE NEXT CANDIDATE FOR A GROUP OF ITEM TO BE SEARCHED. */

DCL LST FIXED BIN(15,0), /* LST PTS TO AN OUT OF RANGE DELIM */

DLM CHAR(1) .. /* DELIMITS ITEM SOUGHT, UNLESS THE */

/* ITEM IS THE LAST ONE IN RANGE. */

C='ERRORIOUS ITEM' .. /* SHOULD SHOW IN CERTAIN ERRORS. */

RENT .. /* RE-ENTRY POINT AFTER PAGE ENTRY PROCESSING */

/* ITR POINTS TO THE CHARACTER WHICH IS */

/* 1ST CANDIDATE FOR START OF ITEM. */

IF ICP(20) THEN PUT FILE(SYSPRINT) EDIT(LST,' ','DIM',''')

(F(4), 3 A) SKIP ..

IF ITRGT'LST THEN GOTO NONEXIT .. /* ITEM TOO SMALL. */

L=VERIFY(SUBSTR(P,ITR,LST-ITL),',';',?;:-') ..

/* ABOVE STATEMENT SKIPS LAST PUNCTUATION */

/* FOLLOWING LAST ITEM PROCESSED. PURPOSE IS */

/* TO SEARCH FOR BEST CANDIDATE FOR LEFTMOST */

/* CHARACTER OF CURRENTLY SOUGHT ITEM. */

/* PUNCTUATION INCLUDES COMMA, BLANK, PERIOD. */

/* EXCLAMATION POINT, MINUS SIGN, QUESTION */

/* MARK, AND COLUMN. */

IF L=0 THEN GOTO NONEXIT .. /* IF NO DESCRIPTIVE ITEMS */

ITL=INDEX(SUBSTR(P,ITR+1,LST-ITL-1),DLM) ..

IF ITR=U THEN TIF=LST-ITL .. /* POINTS TO LEFTMOST CHARACTER OF ITEM */

ELSE ITR=ITR+1 .. /* POINTS TO DELIM FOLLOWING ITEM */

ITF=ITR .. /* POINTS TO DELIM OR LAST CHARACTER */

/* IF NO EXPLICIT DELIMITER IS FOUND IN RANGE THEN ITR */

/* PTS TO THE LAST CHAR OR THE ITEM, ELSE ITR PTS TO */

/* THE DELIMITER ITSELF AT THIS TIME. */

ITRACK .. C1=SUBSTR(P,ITR,1)

IF C1=' ' OR C1='.' OR C1='!' OR C1='?' OR C1='DL' THEN
DO
  /* CHOP TRAILING BLANKS, PERIODS, & COMMAS */
  IPL=tpl-1
  /* MOVING BACK PTR, AND THEN GOING BACK TO */
  GOTO ITBACK
  /* SEE IF ANY MORE TO CHOP. */
  IF IPL-ITL LT 0 THEN DO
    /* IF CHOPPED TOO MANY. */
    CALL ERRMSG('NL11',ITL).
    /* NEGATIVE LENGTH ITEM */
    CALL ERRMSG('NL12',IPL-ITL+1).
    GOTO ENDTGIM.
  END.
  IF SUBSTR(P,IPL-3,4)='but' THEN /* TAKE CUT TRAILING EUTS */
    DO
      IPL=IPL-4
      GOTO ITBACK.
    END.
  IF SUBSTR(P,ITL-1,2)='p' THEN
    DO
      /* CHECK FOR PAGE CHANGE */
      I=ITL/100
      /* COMPUTE AND SAVE LINE NUMBERS FOR MAE */
      K=ITL+1
      /* SET UP NON-DUMMY PARM FOR PAGE NUMBER */
      CALL COGO(K,IPL+1,N,NOPAGE).
      IF NL ITAG CR IPAG GT N+1 THEN CALL ERRMSG('SUSP',ITL)
      IPAG=N,
      /* NEW PAGE FOUND AND CONVEYED SUCCESSFULLY */
      ITR=K
      /* SET UP TRK TO GET ITEM AFTER PAGE ENTRY */
      PUT STRING(CHPAGE,EDIT(IPAG)(F(4))).
      IPAGES(L)=CHPAGE.
      S=S CAT FA CAT 'B' CAT CHPAGE.
      /* ENTRY TO STRUCT FILE */
      NOTES(L)=NOTES(L) CAT '*'
      IF ITOP(6) THEN
        PUT FILE(SYSPRINT) SKIP(2) EDIT
        (** NEW PAGE 'IPAG,'') (COL(21),A,F(4),SKIP(1),A),
        GOTO RENT.
        /* TRY FOR ITEM AGAIN */
      NOPAGE.
      END.
      IF IPL-ITL GT 3 THEN DO
        /* CHECK LEADING EUTS */
        CH4=SUBSTR(P,ITL,4)
        IF CH4='But' OR CH4='but' THEN DO /* CAPITAL & SMALL */
          ITR=ITL+4.
          GOTO RENT.
        END.
      END.
      C=SUBSTR(P,ITL,IPL-ITL+1)
      /* DELIVER ALL ITEMS IN C */
      IF DLM NE ' ' THEN ITR=ITR+1
      /* MOVE FOR FAST DELIMITEE. */
      IF ITOP(21) THEN PUT FILE(SYSPRINT) EDIT
        ('',C,'') (A).
      ENDTGIM.
      ENDTGIM.
    END.
    CIGO.
    /* PROC(IPTR,IPRT,IVAL,WGO) */
    /* SEARCH FOR AND ATTEMPT */
    /* TO CONVERT A NUMBER TO INTERNAL BINARY */
    /* IN CALLING SEQUENCE IPRT PTS TO FIRST */
    /* POSSIBLE CHAR OF NUMBERS. IPRT PTS TO ONE */
    /* CHAR PAST THE POSSIBLE END OF THE NUMBER. */
    /* EXIT VIA WGO IF NO FIND OF E1D CONVERT. */
    DCL WGO LABEL.
    IINENG=MIN(IPRT-IPTR,20).
    IF IINENG LT 1 THEN GOTO WGO /* STRING TOO SMALL. */
    CHGO=SUBSTR(P,IPTR,IINENG)
    DO NB=1 TO IINENG/* SRC FORWARD FOR 1ST NONELANK. */
    IF CHGOA(NB) 'NE'' THEN GOTO STST /* CHG PAST THE POSSIBLE END OF THE NUMBER. */
    END.
    GOTO WGO
    /* SINCE NOTHING BUT BLANKS IN RANGE. */
    /* IF 1ST NONELNK CHAR IS */
    /* NOT A DIGIT. */
    /* PUT CONVERTED VALUE INTO EAM FOR RETURN. */
    /* NOW PTS TO 1ST CHAR AFTER Digits */
    /* END CIGO. */
    ERMSG PROCEDURE(ABBRV,LOC)
    /* SEE NOTE AT IFC CP LISTING */
    /* EFFECT MESSAGE ROUTINE */
**DCL**

`ABBRV` CHAR(4), 
`LOC` FIXED BIN(15,0) /* AVODING STG PROBLEM. */ 
`ERR=MAX(1,LOC-49)` /* (STGS) PUT FILE(SYSRT) SKIP(2) EDIT */ 
`('ERROR NOTE *** ABBRV, AT LOC, **')` 
`SUBSTR(P, IERR, MIN(100, I-IERR+1)))' */ 
`A(2,F(5),3)A` 
`IF IOPT(27) THEN ERR='1'B` /* FCE CTHFR ERR MSG'S. */ 
`IERRNO='IERRNO+1` /* FOR ERROR STATISTICS. */ 
END ERRMSG

**ENDP**

`PUT FILE(SYSRT) SKIP(2) EDIT` 
`('MAX BLKSIZE='2'MAXOUT,'ERRMSG ERRORS='2'IERRNO,'TRKS='2'KSTRU,'OUTPUT CHARS='2'NOUT,'**AS','REFS='2'NLAD,'` 
`'CGLE='2'NCGLE,'SEE REFS='2'NSEE)'` 
`A(8,A,SKIP)`

END

/*
** LADDER  JCB 99999211, WCD, MSGLEVEL=1
** L EXEC LSEPUDGEN, MD=M=LADDR,
** PARM.PLLN='C48,NT,A,X,FW,OPT=0,SIZE=88K,E,LC=62',
** PARM.IKED='MAP',IST*,
** BAKUP.SYSUT1 DD *,DCB=BLKSIZE=80
** LADDER **/
** TLE='TITLE LADDER ENTRY'.
** SLE='SEE TYPE LADDER ENTRY'.
** CGLE='CAST GRP LADDER ENTEY'.
** LADDER **/ 

**DCL**

/* THE NEXT 8 ARRAYS FORM THE GROUP CONTROL BLOCKS (GCB'S)*/ 
`GBIT(64) BIT(1), /* 1'B INDICATES GCP DELETE FCN */` 
`GTM(150) FIXED BIN(15,0), /* PTS TO TIME ENTRY ICE, */` 
`OR IS O IF NO TIME ENTRY IN GCP. */` 
`GB(150) FIXED BIN(15,0), /* PTS TO ROLE ICE CHAIN */` 
`GA(150) FIXED BIN(15,0), /* PTS TO ACTOR ICE CHAIN */` 
`GEN(150) FIXED BIN(15,0), /* COUNT OF ROLES IN CHAIN*/` 
`GAN(150) FIXED BIN(15,0), /* ACTOR COUNT IN CHAIN*/` 
`GC(150) FIXED BIN(15,0), /* CAST GRP ENTRY CR ZERO */` 
`G(150) FIXED BIN(15,0), /* Foward CHAIN PTR */` 
`IGPSIZ INITIAL(064) STATIC, /* SIZE OF GCB ARRAYS. */` 
/* THE NEXT 5 ARRAYS FORM THE ITEM CONTROL BLOCKS (ICB'S)*/ 
`ITMSIZ INITIAL(012) STATIC, /* SIZE OF ICE ARRAYS. */` 
`IPBIT(128) BIT(1), /* ITEM ADD INDICATOR */` 
`ITMBIT(128) BIT(1), /* ITEM DELTE INDICATOR */` 
`IP (128), /* ACTUAL PTS TC ACTUAL ITEMS */` 
`IL (128), /* ACTUAL LENGTHS OF ACTUAL ITEMS */` 
`ILK (128), /* ITEM CONTROL EK CHAIN LINK WRDS */` 
**DCL** 
`ERR BIT(1), /* DCL STMT FOR FILE STORAGE & SEARCHES */` 
`ERROR INDICATOR BIT FOR SECTION. */` 
`ICOLDATE FIXED BIN(31,0), /* OLDKEY BIT(16) BELOW, IS */` 
`DEFINED ON ICOLDATE POS(3). */` 
`OLDKEY BIT(16) 'DEF OLDKEY POS(3)'
`IDTRK (0000,371), /* DAY OF YEAR TC TRK MAP. */` 
`MAP, GIVES 1ST TRACK TO SEARCH */` 
`FOR ANY SECTION WITH A CERTAIN */` 
`DATE, ELEMENT='=' IF NO SUCH */` 
`DATE IS IN THE LADA FILE. */` 
`NEWDATE FIXED BIN(31,0)'INITIAL(0)' STATIC,` 
`CRNKEY BIT(16) 'DEF NEWDATE POS(3)'
`LADA ENV (REGIONAL(3)) KEYED DIRECT,` 
`SKEY CHAR(12), /* RECORDED PART OF SEARCH KEY FOR LADA BLKS */` 
`REGU CHAR(8), /* TRACK NUMBER IN CHAR STRING FOR SUCH */` 
`/* KEY IN LADA FILE SEARCH. */` 
`LAD, /* 'CURRENT TRACK FOR SEQUENTIAL */` 
`LADA OUTPUT. */`
ON ERR SNAP PUT EDIT(IGF,IGF,ITP,GGC,S,IXX,T,J) (A).
ON KEY(LADA) BEGIN.

IF ONCODE=51 THEN TO..
CALL ERMSG(KRF, M-L4) /* KEYED RECORD NOT FOUND */
GOTO NOMATCH; /* ERR- SEC NOT FOUND. */
PUT SKIP(3) EDIT('KEY ONCODE=',ONCODE) (A); /* END ERR. */
END.;
CN ENDFILE(STRS) GOTO PROCEND; /* AVOID SEASON CHANGE MESSAGE. */
SEASON. /* TOP OF SEASON LOOP. */
PUT SKIP(6) EDIT('SEASON CHANGE $$$$$$$ TRK=','KLAD') (A);
IF IOPT(2) THEN ISTSEASON DO.
OPEN FILE(LADA) KEYED DIRECT OUTPUT;
CLOSE FILE(LADA).
OPEN FILE(LADA) KEYED DIRECT UPDATE;
KLAD=0; /* RESET OUTPUT TRK PTR TO START. */
IDTRK=-1; /* CLEAR DATE/TRACK MAP FOR PREVIOUS SEASON. */
TRANSBTS=11111111111111111111111111111111B; /* ASSURES NEW */
/* IDTRK ENTRY DUE TO APPARENT DATE CHANGE AT START OF */
/* EVERY SEASON, INCLUDING THE FIRST. */
INPUT. /* TOP OF SECTION LOOP. */
ERR=0'B; /* RESET BLK ERR INDICATOR TO 'NO ERRORS YET' */
DO IGF=1 TO IGPSZ; /* CLF(IGF)=IGF+1; */
   GGC(IGF)=0; /* GBT(IKP)=0'B; END; */
   OF GGB'S.
DO ITP=1 TO ITMSZ; /* IKL(ITEP)=1TP+1; END; */
END; /* PREVIOUS KEY IN LOWER 16 BITS. */
IF J(IOPT-1) THEN GET FILE(SYSIN) EDIT(ICFT,NCFT,C1)
   (32'B(1),F(6),X(41),A(1)); /* CTGNC CARTE CHAIN. */
IF IOPT(6) THEN DO; /* UNKEYED SEC FORUM INPUT OPTION */
READ FILE(STRS) INTO(Z);
STRUKEY=SUBSTR(Z, 2, 12); /* EXTRACT 'KEY'. */
S=SUBSTR(Z, 15, END); /* REMOVE 'KEY' & BLANKS. */
ELSE /* DA KEYED SEC INPUT. */
READ FILE(STRU) INTO(S) KEYTO(STRUKEY);
J=J+1; /* STRU INPUT BLK CTE */
IF IOPT(3) THEN GOTO INPUT; /* SKIP INPUT BLKS OPT. */
IF STRUKEY='XXXXXXXXXX' THEN DO;
   J=J-1; /* DON'T COUNT IT AS A SECTION. */
CLOSE FILE(LADA); /* SEASON END. */
IF IS=LENGTH(S); /* 'CURRENT ACCESSIBLE LENGTH' = */
   ISBE=1; /* 'BORDERLINE' FOR FREE GGBS. */
   NTITB='0'B; /* 'NOT NON-PLAY TITLED SECTION'. */
   DO NS=1 TO 26; /* INVESTIGATE SECTYPE & PROPERTIES */
   IF STYPE(NS)=SMOD THEN GOTO SECPOUND; /* END. */
   CALL ERMSG(1SBC,J) /* GOTO 'INPUT'. */
   /* ILLEGAL SEC. */
SECPOUND. /* IF NS=1 THEN STBTR='0'B; */
   IF NS LE '13 THEN NTITB='1'F; /* ALL BITS = 0 IF PLAY */
   ELSE DO; IF NS LE '13 THEN NTITB='1'F; /* END. */
   /* NTITB='1'B INDICATES A NON-PLAY TITLED SECTION. */
   TRANSBTS=STRUDEF; /* KEYBITS NOW CONTAINS THE CODED */
   /* DATE FROM STRUKEY (SEE DECLARE */
   /* STATEMENTS FOR 'SYN DEFINING). */
   CRNTKEY=KEYBITS; /* NEWDATE NOW HAS NEW CODED DATE */
   IPBIT,IMBIT='0'B; /* REPEATED TITLE LADDER LOCKOUT */
   LBIT='0'B; /* INDICATE 'NO TITLES' DONE YET IN */
   CRTITL='1'; /* THIS SECTION. (FLAT, OFFPA, CR */
   /* 'AFTERPEICE TITLES, THAT IS). */
   IS, /* SECOND LENGTH, LENGTH AFTER LADDER READ IN */
   IGP2, IGP3, IGP4=0; /* INITIAL GCB CHAINS' HEAD & TAILS */
   IXX=0; /* INDEX TO CHK GCB'S TO SEE IF */
   /* THEY REFER TO A CGBL THAT HAS */
   /* BEEN SUCCESSFULLY READ INTO CORE. */
   IGR, /* INITIAL RIGHT GRP PTR */
   IGP, /* INITIALIZE GBPTR */
   /* END PROG. */
ITP+1        */ INITIALIZE ICB PTR
IF I<5 THEN GOTO FORMOUT */ SINCE REQUIRES NC */
                           */ FURTHER PROCESSING. */
S=S CAT FC .  */ I NOW=LENGTH(S)-1. */
GOTO ISTGRP .  */ ENTRY TO GRC LOOP. */
GROUPS .  */ TOP OF GRC LOOP. */
IF IGH=I+1 THEN DO  */ IF END OF SOME BLC GR BLC PART */
IF I=IS THEN DO .  */ IF END OF ILE THEN DE */
   IF IGP<GT IGPZ THEN DO  */ IF AT LEAST 1 TILE GRC. */
      IGP3=IGPZ+1 ;  IGP4=IGP ; END .
      */ NOW GC & CHK FCB SGLE'S. */
      ELSE IF IXX=0 THEN DO .  */ END OF REFERRING GRP. */
         IGPZ=IGP ;
      IF IZ = GT I THEN DE
         T=IS ;  GOTO CONTGRPS ; END ; END .
         DO IXX=IXXY+1 TO IGPZ ;
      IF GC(G(IXX)) NE 0 THEN DO  ;  */ IF THIS GRP HAS READ A */
         ;  */ CGE SUCCESSFULLY INTG CORE FCB SETUP. */
         I=GC(G(IXX)) ;  GC(G(IXX))=IGPZ+1 ;
      GOTO CONTGRPS ; END .
      IGPMAX=MAX(IGPMAX,IGP) ;  */ MAX GCB'S USED IN RUN. */
      IPMAX=MAX(IPMAX,IGP) ;  */ MAX ILE'S USED IN RUN. */
      GOTO OUTPUT ;  */ ALL (ALL) INPUT/SETUP DONE, NOW PROCESS. */
END ;
CONTGRPS .
IF IGP GE IGPZ THEN DO .  */ CHECK OVERFLOW USAGE OP*/
                         */ GROUP CONTROBL BLOCKS. */
CALL ERRMSG('NGC',IGR) ;  IGR=I+1 ;  GOTO GROUPS ;
END ;
IGP=IGPZ+1 ;  */ SET PTR TO NEXT FREE GCE. */
ISTGRP .  */ ENTRY TO GRC LOOP FROM SECTION LOOP. */
KNTIME=0 ;  */ CLEAR MULTIPLE TIME ENTRY DETECTOR/COUNTER */
IGL=IGR ;  */ LAST BYTE+1 OF PREVIOUS GRC= IPTM CST */
         ;  */ BYTE OF CURRENT/NEW GROUP, IF ANY. */
GRB(IGP),GTM(IGP),GA(IGP),GRN(IGP),GAN(IGP)=0 ;
IGR=INDEX(S,IGR+1,IGP)+IGL ;  */ FIND GRP-END */
IF IGR-IGL LE 2 THEN DO .  */ GROUP TOO SHORT. */
   CALL ERRMSG('NGR',IGL) ;  IGR=IGP-1 ;
      GOTO GROUPS ; END .
   IF SUBSTR(S,IGR-1,1)=| THEN GEIT (IGP)=1'B ;
   GTRB=SUBSTR(S,IGR,2) ;  */ GET GRP DELIM CR1 CHAR8. */
   ITR=IGL+2 ;  */ RESET ITEM PTR. */
ITEMS .
IF ITR=IGR THEN DO .  */ IF LAST ITEM OF GRP IS DONE. */
   IF GAN(IGP)+GRN(IGP)+GTM(IGP)=0 THEN IGR=IGP-1 ;
GOTO GROUPS ;  */ THE ABOVE TEST IS FCB GRPS THAT HAVE NO */
END ;  */ "SIGNIFICANT ITEMS." TITLE GRPS USUALLY. */
ITR=ITR
ITYP=SUBSTR(S,ITR,2) ;
IF ITYP1 NE FA THEN DO .
   CALL ERRMSG('NEFA',ITR)
   PUT EDIT(S ,'' ,1 ,IGP,ITYP) (A) . END .
   ITR=INDEX(S,ITR+1,IGE-ITL ),FA .
IF ITR=0 THEN ITR=IGR ELSE ITR=ITR+ITL .
IP(ITP)=ITL+1 ;  */ SAVE PTR TO ITEM */
IL(ITP)=ITER-ITL-2 ;  */ SAVE LENGTH OF ITEM */
C1=SUBSTR(S,ITL+2,1) ;  */ GET 1ST CHAR OF ITEM. */
IMBIT(ITP)=(C1=1) ;  */ IBIT (ITP)=(C1='1') ;
IF IMBIT(ITP) OR IPEIT(ITP) THEN DO .  */ IF SIGNED */
   IP(ITP)=IP(ITP)+1 ;  */ INCREMENT PTR PAST SIGN */
   IL(ITP)=IL(ITP)+1 ;  */ AND DECREASE LENGTH BY ONE */
   END .
IF SUBSTR(S,IP(ITP)+IL(ITP)+1,1)=| . THEN IL(ITP)=IL(ITP)-1 ;
       */ ABOVE STMT TAKES CARE OF TRAILING LASH. */
       /* PRESUMABLY DUE TO GRC DELETE FUNCTION. */
IF ITYP2='A' THEN DO .  */$$$$$$$$$$$$$$$$ ACTOR $$$$$$$$$$$*/
GANN(IGP)=GANN(IGP)+1 /* ADD 1 TO GRF ACTOR CNT */
IF GANN(IGP)=1 THEN GA(IGP)=ITP /* PIR TO ETF TO 1ST */
/* ACTOR IN A GFE IS NOW UP */
GOTO ITEMDONE END .
IF ITYP2='H' THEN DO ./*$$$$$$$ ROLE $$$$$$$*/
GRN(IGP)=GRN(IGP)+1 /* ADD 1 TO GRF ROLE CNT */
IF GRN(IGP)=1 THEN GR(IGP)=ITP /* IF 1ST ROLE IN THE GRF */
GOTO ITEMDONE END .
DO M=1 TO 13 /* FOR ALL TITLE SECTION LETTERS */
IF ITYP2=STYPE(M) THEN DO ./*$$$$$$$ TITLE $$$$$$$*/
IF CRTITL='1' THEN /* IF NO PREVIOUS TITLE */
CRTITL=SUBSTR(S,IP(ITP),IL(ITP)) . GOTO ITEMS . END .
/* END OF TITLE TITLE PROCESSING */
IF ITYP2='L' OR ITP3='C' OR ITYP2='S' THEN DO /*$$$$$$$ 'AS' GFE $$$$$$$*/
IF LEUT THEN IF ITYP2 NE 'C' THEN DC .
CALL ERRMSG('XLDR',ITL) . GOTO ITEMS . END .
IF ITYP2 NE 'C' THEN /* EXCEPT FOR CGLE'S */
LEBT='1'B . /* LOCKOUT MORE THAN ONE TITLE LADDER REF */
Z=SUBSTR(S,IP(ITP),IL(ITP)) . /* TEMP FOR 3 USES SCON */
GET STRING(Z) EDIT
(IDAY,IMONTH,YEAR) (2 F(2),F(4)) . /* CONVETS */
/* FULL LADDER REF DATE TO EINARY */
imonth=imonth-1 . iday=iday-1 . /* SAVES CALCS */
idate=((YYEAR-1659)*372+IMONTH*31+IDAY)*8 /* SET UP */
/* ENCODED DATE FOR SCHR KEY */
skey=chidate CAT R00FKEY . /* SET UP RECORD KEY */
M=IMONTH*31+IDAY . /* GET ADDRESS OF IDTRK ENTRY FOR */
/* DATE OF LADDER REF SOUGHT */
L=IDTRK(M) . /* NOW L = 1ST TRACK TO SEARCH FOR THIS BLK */
IF L=-1 THEN DO . /* IF NO SECTION AT ALL ON THAT */
/* DATE IS ON THE DISK THEN */
/* CALL ERRMSG('NTRK',IDAY+1) . */
PUT EDIT(Z,SMOD,CRTITL) (3 (X(2),A)) .
GOTO ITEMS . END .
ITER=0 /* ITERATION FACTOR FOR EXTRA READ ATTEMPTS */
IF CHIDATE GE PEPCHK THEN DO .
CALL ERRMSG('PREF',ITL) /* FORWARD LADDER BEFERNC */
PUT FILE(SYSPRINT) SKIP(1) EDIT
(Z,SMOD,CRTITL) (3 (X(2),A)) .
GOTO ITEMS . END .
IF TITBIT THEN DO /* FOR NON-FILL TITLED SECTIONS */
GETITER . /* COMPUTE NUMBER OF EXTRA READ ATTEMPTS */
IF M=371 THEN M=0 ELSE M=M+1 .
IF IDTRK(M)=1 THEN GOTO GETITER .
ITER=IDTRK(M)-L . END .
DO M=L TO L+ITER /* TRACK SEARCH LOCFF */
PUT STRING(REGB) EDIT(M) (F(8)) .
READ FILE(LADA) INTO (Z) KEY(SKEY CAT REGB) .
IF IOCTL(4) THEN PUT EDIT(Z) (A) .
/* AT LEAST A 0-LENGTH TITLE IS REQUIRED HERE */
IF LENGTH(Z) GT 4 THEN DO .
ITER=INDEX(SUBSTR(Z,5),FC) /* SEARCH FOR END OF TITLE */
/* EVEN UNTITLED SECTIONS HAVE AT LEAST A NULL TITLE */
IF ITER THEN DO /* IF REFERRED-TO GRF HAS NC ROLES */
CALL ERRMSG('LTER',J) /* OR ACTORS THEN WARN */
ITER=LENGTH(Z)-3 . END . /* PUT GC AHEAD NORMALLY */
/* ANYWAY, BUT AVOID STRG */
CMTB=SUBSTR(Z,5,ITER-1) . /* MOVE TITLE TO CMTB */
/* TITLE MAY BE NULL */
IF IOCTL(17) THEN
PUT FILE(SYSPRINT) SKIP(1) EDIT
('CRITCLT='CRITCLT,'OLDTITL='CTMP) .
(3 'A',COL(40),3 'A') .
IF LENGTH(CRTITL)=0 THEN CRTITL=CTMP .
IF CRTIL=CTMP THEN GOTO TITLESMATCH ,
END ,
/* BOTTOM OF LADDER SEARCH LOOP. */
CALL ERMSG(’TTLC’,ITL)
PUT FILE(SYSPRINT) EDIT(‘ OLD=’ ,CTMP,’’’,’’’) (A) ,
GOTO NOMATCH ,

TITLESMATCH ...
IF ITYP2=’C’ THEN DO ,/* IF CAST GREF LADDER ENTRY THEN DO */
ITER=INDEX(Z,’?R’ CAT SUBSTR(S,IP(GR(IGE)),IL(GR(IGE)))) ,
IF ITER=0 THEN DO ,/* CALL ERMSG(’NSCN’,IF(GR(IGE))) */
GOTO ITEMS ,END ,/* NC MATCH CN 1ST ROLE. */
DO K=ITER TO 1 BY -1 ,/* 1ST ROLE MATCHED. */
IF SUBSTR(Z,K,1)=FC THEN GOTO FCPGND ,END ,
CALL ERMSG(’RNMP’,ITER) ,/* FGM LOGIC ERROR. */
FCPND ...
/* NORMAL EXIT FROM LEFT-HAND FC SEARCH LOOP. */
ITER=INDEX(SUBSTR(Z,K+1,FC)) ,/* SEARCH FOR END CP */
IF ITER=0 THEN ITER=LENGTH(Z)+1 ,/* MATCHING ROLE’S GREF. */
ELSE ITER=ITER+K ,
S=S CAT SUBSTR(Z,K+1,ITER-K-1) ,
GCG(IGP)=LENGTH(S) ,/* SAVE FOR USE AS I. */
S=S CAT FC ,
GOTO ITEMS ,END ,
S=SUBSTR(S,1,1) CAT Z CAT FC ,/* APPEND SIE OR TIE. */
IS=LENGTH(S)-1 ,/* SAVE FOR USE AS I. */
IF ITYP2=’L’ THEN ISEE=IS ,
/* ABOVE DEFINES BORDERS BETWEEN SEEN GRES & FIRM ONES. */
/* IF NO UNFIRM GRES THEN ISEE IS STILL SET APPROPRIATE */
/* ISEE LESS THAN IS DEFINES SEEN GRES EXISTENCE. */
GOTO ITEMS ...
NOMATCH ...
PUT FILE(SYSPRINT) EDIT(‘ CRTIL=’ ,CRTIL,’’’,’’’) (A) ,
GOTO ITEMS ,END ,
IF ITYP2=’T’ THEN DO ,/* $$$$$$ TIME $$$$$$$ */
IF KNTIME GT 0 THEN DO ,/* CALL ERMSG(’2TIM’,ITL) */
GOTO ITEMS ,END ,
KNTIME=KNTIME+1 ,/* COUNT TIME ENTRIES WITHIN GREF. */
GTM(IGP)=ITP ,/* MAKE GCB ECNT TO TIME’S ICE. */
GOTO ITEMDONE ,END ,
IF ITYP2=’B’ THEN GOTO ITEMS ,/* FAGES ARE IGNORED, BUT */
/* THIS SAVES SPACE AND AVOIDS ITEM-CHAIN- */
/* COUNT PROBLEMS (PAGE NUMBER ENTRY */
/* REPLACING AN ITEM). */
ITEMDONE ...
ITP=ITP+1 ,/* RESET TO PT TO NEXT FREE ICE, IF ANY. */
IF ITP GE ITMSZ THEN DO ,/* IF NO ICB’S LEFT. */
CALL ERMSG(’NYCB’,ITE) ,IGP=I+1 ,GOTO GROUPS ,
END ,
GOTO ITEMS ,/* GOTO TOP OF ITEMS LOOP. */
OUTPUT ... IGP=0 ,/* YOU COME HERE AFTER ALL (ALL) DATA HAS */
/* BEEN READ IN AND GCB/ICE PRTS SET UP. ‘FIG-*/
/* URING OUT’, THAT IS ADDITION, DELETION ETC */
/* ARE NOW TO BE DONE. ALSO CALLED PROCESSIN */
MORUPDAT ...
/* TOP OF INTERREPRESENTATION LOOP. */
IF INTCHN=0 THEN DO ,/* IF CGLE JUST PROCESSED THEN */
INTCHN=1 ,/* RESET THAT INDICATOR. */
IF IGP3SV=0 THEN /* IF NO GRES IN OUTPUT CHAIN YET */
IGP3,IGP4=GCG(IGP) ,/* THEN MAKE CGLE THE OUTPUT CHAIN. */
ELSE DO ,/* OTHERWISE MERELY */
GP(GCG(IGP))=IGP3SV ,/* ADD CGLE TO CURRENT OUTPUT CHAIN, */
IGP3=GCG(IGP) ,/* RESTORE OUTPUT CHAIN HEAD PRT, */
IGP4=IGP4SV ,END ,/* AND OUTPUT CHAIN TAIL PRT. */
END ,
IGP=IGP+1 ,/* INCREMENT TO NEXT REGULAR GREF. */
IF IGP GT IGP2 THEN GOTO PURMOUT ,/* IF ALL */
/* REFEREING GRES, IF ANY, HAVE BEEN PROCESSED THEN */
IF GTM(IGP)+GCHN(IGP)+GAN(IGP)=0 THEN DO ,
CALL ERMSG(’0GCB’,IGP) ,GOTO McRUDAT ,END ,
IF GRN(IGP)+GAN(IGP)=0 THEN CALL SUBGRADD ; TIME RECHAIN *
IF GCG(IGP) NE 0 THEN DO ; IF CAST GRP LADDER GRP *
IGP3SV=IGP3 ; GPG4SV=IGP4
IGP3,IGP4=GCG(IGP) ; TEMPORARILY SET US THE REFEREE-
INTRCHN=0 ; INDICATE CGLF 'JUST PROCESSED'.
SPITE=0101'B ; SET FOR KEYING CN RCLES ONLY.
DO TTL=1 TO GRN(IGP) ; FOR ALL RCLES IN REFEREED GRP.
CALL KEYS(IGP,TTL) ; SRCH REFEREED-TO GRP FOR MATCH.
IF IRG =0 THEN DO ; IF NO MATCH THEN
CALL ERMSG('INKY',J) ; GOTO MDRUPDAT ; END ; END ;
DO TTL=GA(IGP) TO GAN(IGP)+GA(IGP)-1 ; TO AVOID
/ * REPLACEMENT PROCESSING, SET ALL ACTORS IN REFEREED */
/ * GRP TO HAVE I?S FOR PAIRED ACTOR ADDITION PROCESSIN */
IPBIT(TTL)= NOT IMBIT(TTL) ; END ; / * UNLESS MINUS *
END ;
/ * IN 'SEE' LADDER CASE, BELOW, DELETE FIRST DUP RCLE. *
/ * IF ANY, ENCOUNTERED FROM ITS UNFIRM GRP. AND THEN *
/ * PUT OUT BOTH FIRM AND UNFIRM GRP(S). *
IF ISEE=IS THEN DO ; IF THIS SECTION
/ * HAS A 'SEE' LADDER REF IN IT THEN DO
DO TTL=1 TO GRN(IGP)
SPITE=111'B ; CALL KEYS(IGP,0)
IF IRG =0 THEN DO ; CALL ERMSG('CELT',IGP)
GOTO MDRUPDAT ; END ;
CALL GRPDLT ;
GRPDLT ; PROC ; THE GRP DELETE FUNCTION.
IF INTRCHN=0 THEN GTM(IGP3),GRN(IGP3),GAN(IGP3)=0 ;
ELSE
IF IG4=IGP3 THEN TGP3=0 ;
ELSE DO ; IF IGL=0 THEN TGP3=GIF(IGP)
ELSE DO ; IF IGR=GIF4 THEN IGR4=IGL
ELSE GF(IGL)=GF(IGP) ; END
END GRPDLT ;
GOTO MDRUPDAT ; END ; /* END OF GRP DELETE CODE */
/* UNPAIRED ROLE SUBGRP RECHAIN FOCILS BELOW */
IF GAN(IGP)=0 THEN IF NOT IMBIT(GR(IGP)) THEN CALL SUBGRADD ;
ELSE DO ; TRY GRP DELETE FOR UNPAIRED ROLE GRP.
SBITE=1010'B
CALL KEYS(IGP,1) ; IF IRG NE 0 THEN
DO ; GR(IGP)=ILK(GR(IGP)) ; GAN(IGP)=GRN(IGP)-1
GR(IGP)=ILK(GR(IGP)) ; GAN(IGP)=GRN(IGP)-1 ; END
ELSE CALL ERMSG('URDT',IGP)
GOTO MDRUPDAT ; END
/* UNPAIRED ACTOR SUBGRP RECHAIN FOCILS BELOW */
IF GRN(IGP)=0 THEN IF NOT IMBIT(GA(IGP)) THEN CALL SUBGRADD ;
ELSE DO ; TRY GRP DELETE FOR UNPAIRED ACTOR GRP.
SBITE=0010'B
CALL KEYS(IGP,1) ; IF IRG NE 0 THEN
DO ; GR(IGP)=ILK(GA(IGP)) ; GAN(IGP)=GAM(IGP)-1
GR(IGP)=ILK(GA(IGP)) ; GAN(IGP)=GAM(IGP)-1 ; END
ELSE /* NO MATCH IN ATTEMPTED UNPAIRED ACTOR GRP */
/* DELETION. */
CALL ERMSG('UADT',IGP) ; GOTO MDRUPDAT ; END
IF IPBIT(GR(IGP)) OR IMBIT(GR(IGP)) /* IF RCLE IS */
THEN GOTO 'ACTORKEY' ; /* SIGNED THEN KEY ON THE */
/* ACTOR PART OF GROUP */
SBITE='0101B' /* ROLE IS NOT SIGNED, SC */
EMSG='KEY' /* */
CALL PAYES(GA,GAN) /* */
ACTORKEY /* ECLE IS SIGNED, SC */
SBITE='0011B' /* SET UP FOR ACTOR SRCH */
EMSG='AK' /* */
CALL PAYES(GE,GRN) /*叫 */
PAS .. PROC (GA,GAN) /* THIS PROC, PROPERLY MASSAGED, */
* DOES MOST OF THE WORK FOR ORDINARY PAIRED GRFS, ITEM */
* ADDITION, REPLACEMENT, & DELETION, & GRF RELINK IF */
* NO MATCH. RELINKING IS TO OUTPUT CHAIN, AND IS THE */
* MOST FREQUENTLY PERFORMED OPERATION. */
DCL (GA(*),GAN(*)) FIXED BIN(15,0),
CALL KEYS(IGP,1) /* KEY ON IT. */
IF IRG=0 THEN IF IPBIT(GA(IGP)) OR IMBIT(GA(IGP)) THEN DO /* */
* NO MATCH FOR A ROLE IN A PAIRED GRF WITH A SIGNED ACTOR */
CALL ERRMSG(EMSG,IGP) /* */
GOTO MORUPDAT /* END. */
ELSE DO /* ADD GRF IF BOTH UNSIGNED & KEY SEARCH FAIL */
CALL SUBGRADD /* GOTO MORUPDAT /* END */
ELSE DO /* IRG NE 0 SO KEY SEARCH SUCCEEDED, SO DO */
* THE 'T', 'F', 'F', AND REPLACE FCNS. */
RPLUSMINUS ...
IF IPBIT(GA(IGP)) THEN DO /* SINGLE ACTOR ADDITION */
L=GA(IGP) /* GA(IGP)=ILK(L) /* GAN(IGP)=GAN(IGP)-1 */
ILK(L)=GA(IGP) /* GA(IGP)=L */
GAN(IGP)=GAN(IGP)+1 /* */
GOTO RPLUSMINUS /* END */
IF IMBIT(GA(IGP)) THEN DO /* IF ACTOR DELETE THEN */
/* SRCH IRG GRF FCN */
L=GA(IGP) /* ITEM EQUALLING GA(IGP) */
K=GA(IGP) /* /* SAVES SUBSCRIPTING. */
IIN=0 /* THIS LOCF SEARCHES FOR THE ITEM TO BE */
* DELETED IN THE REFERRED-TO GRF. */
IF IIN GE GAN(IGP) THEN DO /* */
CALL ERRMSG('ADLT',IP(L)) /* */
GOTO MORUPDAT /* END */
IF SUBSTR(S,IP(K),IL(K))=SUBSTR(S,IP(L),IL(L)) /* */
THEN DO /* */
IF K=GA(IGP) THEN GA(IGP)=ILK(K) /* */
ELSE ILK(K)=ILK(K) /* */
GAN(IGP)=GAN(IGP)-1 /* GAN(IGP)=GAN(IGP)-1 */
GAN(IGP)=GAN(IGP) /* */
GOTO RPLUSMINUS /* END */
K=ILK(K) /* */
GOTO MORESBRCH /* END */
* THE FAMOUS REPLACE FUNCTION PoILCWS. */
GANN(IGP)=GAN(IGP) /* GA(IGP)=GA(IGP) */
GOTO MORUPDAT /* END */
SUBGRADD .. PROC /* */
* SUBGRP RECHAIN FCN UNPAIRED GRF */
IF IGP4 NE 0 THEN GIP(IGP4)=IGP /* */
IGP4=IGP /* IF IGPF=0 THEN IGPF=IGP4 */
GOTO MORUPDAT /* END */
FORMOUT ...
Z=PC CAT SMOD CAT FA CAT SMOD CAT CRTITL /* THIS STATE- */
/* MENT BEGINS THE FORMATTING OF FINAL CUTPUT IN THE Z BUFF. */
IF IGPF=0 THEN GOTO ENDOUT /* IF EMPTY OUTPUT CHAIN. */
LBT='1'B /* INITIALIZE PTR TO 1ST GCR IN OUTPUT CHAIN. */
IGF=IGP3 /* */
DO WHILE(LEBIT) /* INDICATE LAST GCR NOT FOUND YET. */
/* LBIT='0'B=LAST GCR NOT ENCOUNTERED */
IF IGF=IGP4 THEN LBIT='0'B /* IF LAST GCR ENCOUNTERED */
IF GTM(IXG)+GEN(IXG)+GAN(IXG) GT 0 THEN /* THIS STMT */
Z=Z CAT 'G' /* */
ELSE GOTO ENDT /* */
IF GTM(IXG) NE 0 THEN /* IF THIS GRF HAS A TIME ENTRY. */
Z=Z CAT 'T' /* CAT SUBSTR(S,IF(GTM(IXG)),IL(GTM(IXG))) */
IXI=GR(IXG) /* */
DO IXX=IXI TO IXI+GNX(IXG)-1.
Z=Z CAT 'R' CAT SUBSTR(S,IP(IXI),Il(IXI)).
IXI=ILK(IXI) END.
IXI=GA(IXG).
DO IXX=IXI TO IXI+GNX(IXG)-1.
Z=Z CAT 'A' CAT SUBSTR(S,IP(IXI),Il(IXI)).
IF IP(IXI) GT ISEE THEN Z=Z CAT '?'.
IXI=ILK(IXI) END.
ENDIN.
IXG=GF(IXG).
IF IXG=0 THEN CALL ERRMSG('IXG0',IXI).
END.
ENDOUT.
/* AT THIS POINT THE OUTPUT ELK, IN Z, IS */
/* READY, AND ONLY HOUSEKEEPING REMAINS BEFORE */
/* THE ACTUAL WRITE STATEMENT. */
IF IOPT(24) OR ERR THEN
PUT FILE(SYSPRINT) EDIT
(J, 'OUTPUT=' T Z ) (X(3),? A).
RL=LENGTH(Z)+32.
RESIZE.
IF SP LT 0 OR (TITBIT AND STBT(NS)) THEN FC.
IF RL GT 3589 THEN CALL ERRMSG('OTSZ',LENGTH(Z)).
SP=3625.
KLAD=KLAD+1.
STBT='0'E.
GOTO RESIZE.
END.
STBT(NS)=1'B.
PUT STRING(REG8) EDIT(KLAD) (F(8)).
WRITE FILE(LADA) FROM(Z) KEYPROM STRUKY CAT REG8.
SP=SE-(537*RL(20))/512469.
IF IOPT(1) OR ERR THEN PUT FILE(SYSPRINT) EDIT
((L,SUBSTR(S,L*100+1,MIN(100,L*1CC))
DO L=0 TO (I-1)/100).
IF IOPT(5) OR ERR THEN
PUT FILE(SYSPRINT) SKIP(1) EDIT
('B='J,'LENGTH='I,'IP='F,'KEY='T,B9OFKEY,...
SUBSTR(S,1,MIN(83,1)).,....
,NEWDATE,372+1659.
TRUNC(MOD(NEWDATE,372)/31)+1,
MOD(NEWDATE,31)+1
(A,F(5),F(4),5 A,COL(123),F(4),2 F(3).
MAXOUT=MAX(MAXOUT,LENGTH(Z)). /* FOR MAX BLKSIZE STATS. */
NOUT=NOW+LENGTH(Z). /* TOTAL OUTPUT CHAR STATS. */
LATR=MAX(LATR,KLAD).
IF IOPT(7) THEN DO.
S='CAT STRUKY CAT Z. /* KEY, ALSO AT LEAST 18 */
/* BYTES FOR TAPE ELCK. */
WRITE FILE(LADS) FROM(S).
END.
IF OLKEY NE KEYBITS THEN DO. /* EACH TIME DATE CHANGES */
IOLDATE=O. /* CLEAR LEFT HALF OF IOLDATE. */
OLKEY=KEYBITS. /* PUT CODED DATE IN RIGHT HALF. */
N=MOD(IOLDATE,372) /* GET TABLE ADDRESS FOR DAY OF YE. */
IDTRK(N)=KLAD. /* SEARCHES FCE SECTIONS THAT WERE */
END. /* ON DAY OF YEAR N WILL START AT TRACK KLAD. */
GOTO INPUT. /* REACCESS NEXT SECTION, IF ANY. */
KEYS..PROC(TGRF,TOFF). /* GENERAL PURPOSE ITEM KEYING. */
/* ON EXIT IREG PTS TO THE GCB IN WHICH THE FOUND ITEM RESIDES. */
IF IREG=0 THEN NO MATCH WAS FOUND. ALSO IGL PTS TO THE PREVIOUS GCB ON THE CHAIN, IF ANY, OR IS 0. SIMILARLY IROF PTS TC THE ICB OF THE ITEM ON WHICH A MATCH WAS MADE. IT IS NOT NECESSARY 0 IN CASE OF A NO MATCH. ANI IIL PTS TO THE PREVIOUS ICB ON THE CHAIN, IF ANY, OR ELSE TO O. RBITS INDICATES THE TYPE OF ITEM ON WHICH KEYING WAS DONE.
IRG,IROF=O. /* RBITS='0'B. */
IF IGP3=0 THEN RETURN.
IF SBITS(2) THEN DO.
CALL RASH(GR,GARN).
IF IREG NE 'O' THEN DO.
/* RBITS (2) = '1'B. RETURN. END. */
IF SBITS(3) THEN DO.
CALL RASH(GA,GAN).
/* RBITS (3) = '0'B. */
/* EXIT. */
IF IRG NE 0 THEN DO ..
   RBITS(3)='1'B .. RETURN .. END ..
END ..
IF NOT SBITS(3) OR GTM(IGRP)=0 THEN RETUN ..
IGL=0 .. IGR=IGP3 ..
COMPTM=SUBSTR(S,IP(GTM(IGRP)),IL(GTM(IGRP))) ..
MORETIME ..
IF SUBSTR(S,IP(GTM(IGRP)),IL(GTM(IGRP)))=COMPTM THEN RETURN ..
IF IGR=IGP4 THEN DO ..
   IGL=IGR .. IGR=GF(IGR) .. GOTC MORETIME .. END ..
IGR=0 ..
   RBITS=’0'B ..
RASH .. PROC(IT,ITN) .. /* RCLE CH ACTE SEARCH PRCC */
DCL (IT(*), ITN(*)) FIXED BIN(15,0) ..
IRG, IROP=0 ..
   RBITS=’0'B ..
IF ITN(IGRP)=0 THEN RETURN .. /* IF THE SUBGROUP HAS NO */
   /* ITEMS TO MATCH ON THEN */
   IROP=IT(IGRP) .. /* IROP NOW PTS TO 1ST ICE IN THE */
   /* SUBGRP CHAIN, THE ITEM’S */
   /* EXISTENCE IS GUARANTEED BY THE */
   /* STATEMENT ABOVE. */
DO ITN-1 TO IOFF-1 .. IROP=ILK( IROP) .. END ..
/* THE ABOVE DOLOOP MOVES ALONG SUBGRP CHAIN TO THE */
/* PROPER LINK OF THE SUBGRP CHAIN AS DEFINED BY THE */
/* IOFF PARM IN THE CALL TO KEYS. */
COMPTM=SUBSTR(S,IP(IRG),IL(IRG)) ..
   /* SET UP COMPARISON TEME OUTSIDE OF ICCF. */
   IGL=0 ..
   IGR=IGP3 ..
   GCHK=’1'B ..
DO IOUT-1 TO IGPSIZ WHILE(GCHK) .. /* THE OUTER SEARCH ICCF */
IF IRG=IGP4 THEN GCHK=’0'B ..
IF ITN(IGR) NE 0 THEN DO ..
   IIL=0 ..
   IROP=IT(IGR) ..
   DO IIN=1 TO ITN(IGR) .. /* THE INNER SEARCH LOOP. */
   IF COMPTM=SUBSTR(S,IP(IRG),IL(IRG)) THEN RETURN ..
   IIL=IROF ..
   IROP=ILK(IRG) ..
   END ..
   IGL=IGR ..
   IGR=GF(IGR) ..
END ..
END ..
IRG, IROP=0 .. RETURN ..
END RASH ..
END KEYS ..
ERRMSG .. PROC(NOTE,LOC) .. /* GENERAL PURPOSE ERRB Messages. */
   DCL NOTE CHAR(4) ..
   IERR=MIN(MAX(LOC-TO, 1), LENGTH(S)) ..
   PUT FILE(SYSPRINT) SKIP(1) EDIT
   (’*’ ,’NOTE’,LOC,’ ‘,************ ‘)
   SUBSTR(S, IERR, MIN(1-IOPT+1,27)),’**’)
   (2*A, F(5,3) A) ..
   IF IOPT(27) THEN ERR=’1'B .. /* SET ERR BIT FOR MGE. */
   IERRNO=ERRNO+1 .. /* CCOUNT ERRSES. */
END ..
PROGEND .. /* FOR NORMAL END OF PGM OR EB ERROR ON COND. */
   PUT FILE(SYSPRINT) SKIP(1) EDIT
   (’ERRORS=’ ,’IERRNO,’ MAX OUTPUT BLKSIZE=’ ,’MAXCUT,
   MAX ITEMS=’ ,’ITPMAX,’ MAX GPSS=’ ,’IGEMAX,
   MAX REL. TRK=’ ,’LASTTRK,’ OUTPUT CHAIRS=’ ,’NCUT)
   (8*A, SKIP) .. END ..
*/ROOT
   JOB 999999211, WCD, MSGLEVEL= (1,1)
   /* EXEC LSPHACMP, MEM=ROOT, FARM, PLIL=’C48, NT, SIZE=50K’
   /* BAKUP, SYSUT1 DD *
   /* ROOBT .. PROC OPTIONS(MAIN) ..
   DCL IBPIT EXIT, /* INDICATES NO ENDFILE ON LADDER INPUT YET. */
   TAR EXIT, /* PTR TO 1ST FREE UNUSED ELC IN X.S. */
   ICHP EXIT, /* PTR TO ‘LAST CHAR RETURNED’ BY GCH. */
   /* */
ISTOP EXIT INIT(0), /* PROGRAM STOP SWITCH. */

IOPT (80) BIT(1) EXT ..

ON ERROR SNAP PUT FILE(SYSPRINT) SKIP(1) EDIT
('ROOT ERR ONCODE=",ONCODE", (A) ..
PUT FILE(SYSPRINT) SKIP(1) EDIT("BAD INPUT.", (A) ..
IF BIT=0 /* INDICATES PRIME INPUT ECF BCT NOT FOUND YET. */
IAR=1 .. /* PTS TO LAST SE ENTRY+1 .. */
ICH8=80 /* SET TO CAUSE GCHAR TC PRIME CARD BUFF. */

ON ENDFILE(SYSIN) GOTO SYsinEND ..
GET FILE(SYSIN) EDIT(IOPT) (B(1)). ..
NXTREG .. CALL CTRLORD ..

IF IOPT(10) THEN
PUT FILE(SYSPRINT) SKIP(1) EDIT("GOT BACK FROM CTRLORD") (A) ..
IF BIT NE O THEN DO ..
PUT FILE(SYSPRINT) SKIP(1) EDIT("IFEXIT=1, END OF PGM") (A) ..
RETURN END ..

IF ISTOP NE O THEN GOTO SYsinEND .. /* FOR TYPE 9 */
/* STATEMENT INTERPRETED BY CTRLORD */
CALL ITEMGET ..
IF IOPT(10) THEN
PUT FILE(SYSPRINT) SKIP(1) EDIT("GOT BACK FROM ITEMGET") (A) ..
GOTO NXTREG ..
SYsinEND ..
PUT FILE(SYSPRINT) SKIP(1) EDIT("ENDFILE ON SYsin") (A) ..
END ..

/* PL11.SYSLIN DD DISP=(NEW,KEEP), SPACE=(TRK,(50,10,10)) */

//CTRLORD JOB 99999211, WCD, MSGLEVEL= (1,1)
//C EXECS LSPBACMP_MEM=CTRLORD, FARM.PL11='C48,MT,A,X,SIZE=90K,FW,M'
//BACKUP.SYSUTT DD * /
/* CTRL PROGRAM */ (STRG, SUBRG) ..
/* THIS OVERLAY INTERPRETS SELECTION CCNTOOL CARDS FOR THE */
/* ITEMGET OVERLAY. */

CTRLORD .. FPROC ..

%DCL SUPERSSIZE FIXED ..
%DCL SIZECHAR0000 CHAR ..
%DCL SUPERSSPLUS FIXED ..
%DCL MANYSSSIZE FIXED ..
%SUPERSSIZE=140 ..
%MANYSSSIZE=SUPERSSIZE*2 ..
%SUPERSSPLUS=SUPERSSIZE+1 ..
%SIZECHAR0000= CAT SUPERSSSIZE CAT "***" ..
ON ERROR SNAP PUT FILE(SYSPRINT) SKIP(1) EDIT
("CTRL ERR ONCODE=",ONCODE", (A) ..
DCL
ISTOP EXIT INIT(0), /* SWITCH TO STOP PGM ON STMT 9. */
KITEM FIXED BIN(31,0) STATIC EXT /* CUTFUT REC COUNT. */
INIT(0),
J FIXED BIN(31,0) INIT(0) STATIC EXT,
IEBC FIXED BIN(31,0) STATIC INIT(0) EXT, /* FOR TOTAL */
/* POSSIBLE RECORDS GENERATED, VALID*/
/* IF ALL LEVEL 1 INCLUSIVENESS ONLY*/
MANY(2, SUPERSSIZE) STATIC INIT((MANYSSSIZE) 0) EXT
FIXED BIN(31,0),
/* ABOVE IS HIT COUNT + REC COUNT FOR STATISTICS. */
IDATE FIXED BIN(31,0), /* FOR CODED DATE CONVERSION OVERLAY*/
ILDAT(6) EXT, /* DATE RANGE TEMP. */
FX EXT, /* SCF TABLE. */
2 IDR(2) FIXED BIN(31,0), /* DATE RANGE FOR ITEMGET */
2 D, /* CURRENT DEFAULTS FOR SCF TABLE. */
3 INC CHAR(1),
3 SEC CHAR(1) VAR,
3 TIME CHAR(8) VAR,
3 ROLE CHAR(40) VAR,
3 ACTOR CHAR(40) VAR,
3 TITLE CHAR(83) VAR,
3 THEATRE CHAR(8) VAR,
2 'S(SUPERSIZE) LIKE D, /* S'S THEMSELVES. */
IBUPBIT EXT INIT(0),
IOPT(80) BIT(1) EXT, /* OPTION BITS. */
IAR EXT, /* PTR TO 1ST FREE UNUSED SLCT IN X.S. */
CARD CHAR(80) EXT, /* GCHAR'S SYSSN BUFFER. */
CAR(80) CHAR(1) DEF CARD, /* ACCESS TO CARD. */
ICHP EXT, /* PTR TO 'LAST CHAR RETURNED' BY GCHAR. */
CABS CHAR(26) INITIAL('ABCDEFGHIJKLMNOPQRSTUVWXYZ') STATIC,
SMALLS CHAR(26) STATIC INITIAL('abcdefghijklmnopqrstuvwxyz'),
SCHAR CHAR(1),
CCHT CHAR(83) VAR, /* TEMP PCE BUILDING NEW ITEMS. */
ITMSMT LABEL ... /* MEANS PRINT SCE ONLY IN SUMMARY. */
IF IBUPBIT NE 0 THEN CALL 'SUMMARY', /* INDICATES PRINT SCE + STATS. */
NXTBLNK ... CALL GCHAR ... /* LOCATE TO FIND INCLUSIVENESS CHAR */
IF CCHAR=' ' THEN GOTO NXTBLNK, /* SKIES FAST ELANKS */
IF CCHAR='9' THEN DO ;
ISTOP=1, /* RETURN 'END */
IF CCHAR='7' THEN /* 'GC' STATEMENT TO RUN ITEMGET. */
SETUPIN ... DO ... CALL 'SUMMARY' ... RETURN ... END ... SUMMARY ... PROC ... IF LF=0 OR (LF=1 AND IOPT(11)) THEN DO ; /* LIST SCE OPT */
PUT FILE(SYSPRINT) SKIP(2) EDIT(IIDAT) (F(5)) ;
PUT FILE(SYSPRINT) SKIP(1) EDIT
('NO. INC SEC TIME THEATRE TITLE', 'ROLE', 'ACTOR')
(A,COL(74),A,COL(106))
(('K,S,INC(K),S,SEC(K),S,TIME(K),S,THEATRE(K),S,TITLE(K),
S,ROLE(K),S,ACTOR(K) DO K=1 TO IAR-1))
(SKIP(1),F(3),X(1),A(4),A(1),A(9),A,CCL(74),A,
COL(104),A,COL(29))) ;
END ;
IF LF=0 THEN DO ;
PUT FILE(SYSPRINT) SKIP(2) EDIT
('POSSIBLE RECS=',IREC,'INPUT BLKS=',J,
'OUTPUT RECS=',KITM,'NC. HITS RECS PROCED BY HITS')
((J,MANY(1,J),MANY(2,J) DC J=1 TO TAB-1))
(SKIP(1),F(3),F(8),F(6)) ;
IF IOPT(12) THEN MANY=0 ; /* END */
END SUMMARY ...
IF 'CCHAR='0' THEN DO ; /* CLEAR ALL DEFAULTS. */
D=' ' ... GOTO NXTBLNK ... END ;
IF 'CCHAR='8' THEN DO ; /* CLEAR DEFAULTS AND SE TABLE. */
D=' ' ...
DO K=1 TO IAR ; S(K)=D ; /* END */
IAR=1 ; GOTO NXTBLNK ; /* END */
IF CCHAR='5' THEN DO ; /* IF DATE RANGE STATEMENT */
GET STRING(CARD) EDIT(IIDAT)
((X(1),2 'X(1),F(4),'2 'X(1),F(2))) ;
DO IS=1 TO 2 ; /* PCE UPPER AND LOWER DATE RANGES. */
IDATE= (IIDAT((IS-1)*3+1)-1659)*372 +
(IIDAT((IS-1)*3+2)-1)*31 +
IIDAT((IS-1)*3+3)-1 ;
LDR(IS)=IDATE ; /* END */
ICHP=24 ; /* CHAR AFTER FIXED FORMAT DATE */
IF IAR GE SUPERPLUS THEN DO ; /* IF SCE OVERFLOW. */
PUT FILE(SYSPRINT) SKIP(1) EDIT
('IAR OVERFLOW',SIZECHAR(0000)) (A) ;
ELSE DO ; /* END */
IF IAR GE SUPERPLUS THEN DO ; /* IF SCE OVERFLOW. */
PUT FILE(SYSPRINT) SKIP(1) EDIT
('IAR OVERFLOW',SIZECHAR(0000)) (A) ;
ELSE DO ; /* END */
GOTO SETUPPIN END

IF CCHAR='6' THEN DO

D='* CLEAR ALL DEFaulTS AND SET UP */
GET STRING(CARD) EDIT(ISE,LSE) (X(2),2 F(4)) /* FOR REPETITIVE SPECIFICATIONS. */
CALL ITMLOOP END

ELSE DO

IF CCHAR GE '1' THEN IF CCHAR LE '4' THEN DO.
S.INC(IAR)=CCHAR /* ISE,LSE=IAR */
CALL ITMLOOP END END END

GOTO NXTBNK

ITMLOOP .. PROC /* THIS PROCEDURE INTERPRETS SELECTION ENTRY */

ITM CURT='1' /* STATEMENTS EXCEPT FOR THE INCLUSIVENESS */
CALL GCHAR /* INITIALIZE NEW ITEM TO */
SCHAR=CCHAR /* GET THE ITEM TYPE CHAR */
NXTCHAR /* SAVE ITEM TYPE CHARACTER */

CALL GCHAR /* GET A CHAR. */

IF CCHAR='A' THEN DO /* INDICATE NONFLANK ENCOUNTERED. */
IBIT=0 CALL GCHAR CURL=CURT CAT CCHAR END
ELSE DO
ITMSTMT=ITM /* INDICATE NOT END OF STATEMENT YET*/
IF CCHAR='B' THEN GOTO CLOSITM /* END OF ITEM, SET IT UP */
IF CCHAR=' ' THEN DO /* IF BLANK THEN. */
IF IBIT=1 THEN DO /* IF 2ND CONTIGUOUS BLANK THEN */
CURT=SUBSTR(CURT,1,LENGTH(CURT)-1)
ITMSTMT=STMT /* FOR END OF STMT, NOT JUST ITEM. */
GOTO CLOSITM END
IBIT=1 /* INDICATE ONE BLANK ENCOUNTERED. */
CURT=CURT CAT CCHAR END
ELSE DO
CURT=CURT CAT TRANSLATE(CHAR,SMALLS,CAPS)
IBIT=0 END END
GOTO NXTCHAR

CLOSITM

IBIT=0 /* RESET FOR START OF NEXT ITEM. */
IF SCHAR='A' THEN D.ACTOR=CURT /* ELSE DC */
IF SCHAR='R' THEN D.ROLE=CURT /* ELSE DC */
IF SCHAR='T' THEN DO
IF OPT(4) THEN D.TITLE=TRANSLATE(CURT,CASES,SMALLS)
ELSE D.TITLE=CURT END /* ELSE DC */
IF SCHAR='C' THEN D.TIME=CURT /* ELSE DC */
IF SCHAR='W' THEN D.THEATRE=CURT /* ELSE DC */
IF SCHAR='S' THEN D.SEC=CURT END END END
GOTO ITMSTMT

STMT

DO IS=ISE TO LSE

S.INC(IAR)=S.INC(IS)

IF LENGTH(D.ACTOR)=0
THEN S.ACTOR(IAR)=S.ACTCR(IS)
ELSE S.ACTOR(IAR)=D.ACTOR
IF LENGTH(D.ROLE)=0
THEN S.ROLE(IAR)=S.ROLE(IS)
ELSE S.ROLE(IAR)=D.ROLE
IF LENGTH(D.SEC)=0
THEN S.SEC(IAR)=S.SEC(IS)
ELSE S.SEC(IAR)=D.SEC
IF LENGTH(D.TIME)=0
THEN S.TIME(IAR)=S.TIME(IS)
ELSE S.TIME(IAR)=D.TIME
IF LENGTH(D.TITLE)=0
THEN S.TITLE(IAR)=S.TITLE(IS)
ELSE S.TITLE(IAR)=D.TITLE
GOTO NXTCHAR
IF LENGTH(D.THEATRE)=0
  THEN S.THEATRE(IAR)=S.THEATRE(IS) ..
  ELSE S.THEATRE(IAR)=D.THEATRE ..
IAR=IAR+1 ..
IF IAR GE SUPERSSIZE THEN DO .. /* IF SCE OVERFLOW IMINENT*/
  PUT FILE(SYSPRINT) SKIP(1) EDIT
  ('SCE OVERFLOW';SIZECHAR(9999)) (A) .. GOTO SETUPIN .. END ..
END ..
END ITMLOOP ..
GCHAR .. PROC .. /* THIS IS AN INPUT BUFFERING PROCEDURE. IT */
      /* AVOIDS REPEATED CALLS TO DATA MANAGEMENT */
      /* AND THUS SAVES BOTH SPACE AND TIME */
IF ICHF GE 80 THEN DO ..
  GET FILE(SYSIN) EDIT(CARD) (A(80)) ..
  ICHF=0 .. END ..
  ICHF=ICHF+1 ..
CCHAR=CAR(ICHP) .. END GCHAR ..
END CTRLCD ..
* 
/*ITEMGET  JOB 99999211,WCD,MSGLEVEL=(1,1) 
/*ITEMGET PROGRAM */ (SUBRG,STRG) .. 
/* MUCH OF THE CODING IN THE ITEMGET PROCEDURE CF THIS FGM */
/* WAS COPIED FROM IADDR AND THEN MODIFIED, THUS RESULTING */
/* IN SOME ODDITIES AND INEFFECTIVENESS IN ORDER TO SAVE */
/* PROGRAMMING TIME. */
ITEMGET .. PROC ..
%DCL SUPERSSIZE FIXED ..
%DCL MANYSSIZE FIXED ..
%SUPERSSIZE=140 ..
%MANYSSIZE=SUPERSSIZE*2 ..
DCL
  CAPS CHAR(26) INITIAL('ABCDEFGHIJKLMNOPQRSTUVWXYZ') STATIC,
  DATE8 CHAR(8) DEF Z POS(2), /* CN DATE IN OUTPUT BUF */
  IOPT(80) BIT(1) EXT, /* OPTION BITS ARRAY, READ BY RCCT. */
  IAR EXT, /* PTH TO 1ST FREE UNUSED SICT IN X.S. */
  1 X EXT, /* MAIN INTERFACE BETWEEN ITEMGET & CTRLCD. */
  2 IDR(2) FIXED BIN(31,0), /* CURRENT DATE RANGE. */
  2 D, /* CURRENT SELECTION DEFAULTS. */
  3 INC CHAR(1), /* INCLUSIVENESS TAG. */
  3 SEC CHAR(1) VAR, /* SECTION TYPE. */
  3 TIME CHAR(8) VAR,
  3 ROLE CHAR(40) VAR,
  3 ACTOR CHAR(40) VAR,
  3 TITLE CHAR(83) VAR,
  3 THEATRE CHAR(8) VAR,
  2 S(SUPERSIZE) LIKE D, /* SELECTION ENTRY ARRAY. */
  IBUFBIT EXT INIT(0), /* FLAG INDICATING 1ST OVERLAY CR NO*/
  IBIT EXT, /* END OF FILE FLAG, 1 = END OF */
  /* FILE HAS BEEN ENCOUNTERED. */
  LEV CHAR(1) INIT('1'), /* CURRENT LEVEL CF INCLUSIVENESS. */
  TKNO CHAR(8) , /* TRACK NO. TEMP. */
  KEYAR(20) CHAR(12), /* ARRAY OF KEYS CF CURRENT PERFORM */
  TKEY(20), /* REL TRKS FOR PEOVE KEYS */
  KKEY CHAR(2) DEF K, /* RECORDED KEY FOR ITDM REWRITES. */
DCL
  SECTYPE CHAR(1) DEF Z,
  IYR CHAR(4) DEF Z POS(2),
  IMON CHAR(2) DEF Z POS(6),
  IDY CHAR(2) DEF Z POS(8),
  THETR CHAR(8) DEF Z POS(10),
  TIME CHAR(8) DEF Z POS(181),
  GB FIXED BIN(15,0), /* THESE VARIABLES ARE DECLARED */
  GS FIXED BIN(15,0), /* STATIC TO SAVE CCFD SINCE THEY */
  GUM FIXED BIN(15,0), /* ARE NOW OVERLAIN IN STATIC CSECT */
GSN FIXED BIN(15,0),
GBN FIXED BIN(15,0),
/* THE NEXT 2 ARRAYS FORM THE ITEM CONTROL BLOCKS (ICB'S) */
IP (70),
IL (70)) STATIC,
DCL ( /* DCL STMT FOR CONSTANTS ETC. (ALL STATIC). */
THLEGAL CHAR(80) VAR INITIAL( /* ILEGAL THEATRE CHARS. */
'abcdefghijklmnopqrstuvwxyz ' ' ')',
RLEGAL CHAR(80) VAR INITIAL( 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ ' '12' '),
ACLEGAL CHAR(80) VAR INITIAL( 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ ?6' ),
TLLEGAL CHAR(80) VAR INITIAL( 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZ ' ' ' '),
PA CHAR(1) INITIAL('?'),
/* ITEM-START DELIMITER */
PC CHAR(1) INITIAL('?'),
/* GROUP-START DELIMITER */
BBP CHAR(1) INITIAL('P') STATIC, /* SMALL 'P' (EBCDIC) */
SMALLS CHAR(26) STATIC INITIAL('abcdefghijklmnopqrstuvwxyz'),
CAPITL CHAR(83) VAR, /* TEMP FOR CAPITAL TITLE */
SYSPRINT PRINT ENV(P(133,133)),
LADA ENV(REGIONAL(3)) KEYED INPUT SEQL,
LADD ENV(REGIONAL(3)) DIRECT INPUT, /* FOR REREADS. */
ITEM ENV(REGIONAL(3)) OUTPUT SEQL KEYED UNEBUF,
LADS INPUT SEQL,
ITSM OUTPUT UNEBUF,
ITDM ENV(REGIONAL(3)) DIRECT UPDATE, /* FOR REWRITES */
ERRMSG ENTRY(CHAR(4),FIXED BIN(15,0)),
DCL /* DCL STMT FOR WRK & CTLB VARIABLES */
GTP CHAR(2), /* GTYP2 IS DEFINED ON THIS. GTYP CONTAINS */
/* THE CURRENT GSF CONTROL CHARS. */
GTP2 CHAR(1) DEF GTYP POS(2), /* GSF TYPE LETTER. */
ITYP CHAR(2), /* CURRENT ITEM DLM CHARS. */
ITYP2 CHAR(1) DEF ITYP POS(2), /* ITEM TYPE LETTER. */
S CHAR(2000) VAR EXT, /* PRIMARY INPUT BUF, E WRK BUFFER */
/* EXTERNAL FOR OVERLAY SURVIVAL. */
/* SINCE STATIC CSECT IS OVERLAIN. */
Z CHAR(188), /* OUTPUT BUFFER */
(CRTITL CHAR(83)) VAR, /* CURRENT SECTION TITLE, IF ANY. */
ROL CHAR(83) VAR, /* CURRENT 'ROLE', IF ANY. */
ACTR CHAR(83) VAR, /* CURRENT 'ACTOR', IF ANY. */
TBIT BIT(1), /* LOCKOUT BIT FOR MORE THAN ONE TIME ENTRY */
/* WITHIN ONE GROUP. 2ND ENTRY NOT PROCESSED */
(KITM INIT(0) EXT, /* CURRENT ITEM OUTPUT FILE BLK */
(KITM INIT(-1)), /* NUMBER OF HIGHEST BLK YET */
FIXED BIN(31,0), /* WRITTEN OUT, IF ANY. */
IT2 INIT(0), /* SOFTWARE FLAG INDICATING WHETHER WAIT STMT */
/* IS REQUIRED FOR ITE EVENT. */
K, /* HAS KEEF DEFINED ON IT. */
/* KEFP IS PTR TO ELEMENT OF GLD KEY ARRAY FOR LEVEL 4 */
/* RETURN OF PERFORMANCE. */
(ICPERF,ICSEC,ICGRP) FIXED BIN(31,0),
J FIXED BIN(31,0) INIT(0) EXT) STATIC; /* INPUT BLKS. */
DCL /* DCL STMT FOR KEY & DATE MANIPULATION ETC. */
ITREC FIXED BIN(31,0) STATIC INTT(0) EXT, /* FOR CTOTAL */
/* POSSIBLE RECORDS GENERATED, VALID*/
/* IF ALL LEVEL 1 INCLUSIVENESS ONLY*/
MAN Y(2,SUPERSIZE) STATIC INIT((((MANYSSIZE) 0) EXT
FIXED BIN(31,0)), /* ABOVE IS HIT COUNT + REC COUNT FOR STATISTICS. */
STRU KEY CHAR(12), /* INPUT BUF FOR KEY IN STRU READS */
SMOD CHAR(1) DEF STRUKEY POS(4),
REOFKEY CHAR(8) DEF STRUKEY POS(5),
STRUDEF BIT(24) DEF STRUKEY,
TRANSBITS BIT(24) INIT('1111111111111111111111' 'E'),
KEYBITS BIT(16) DEF TRANSBITS POS(6),
IF ILOPT(4) THEN ICAPO=0 ELSE ICAPO=1 /* FOR FASTER */ /* ACCESS TO OPTION NO. 4. */

ON ERROR SNAP BEGIN...
PUT FILE(SYSPRINT) SKIP(1) EDIT ('ITEMS ERR ONCODE='ONCODE') (A),
GOTO ENDPREC END
ON ENDPREC(LADS) GOTO ENDPREC
ON ENDPREC(LADA) GOTO ENDPREC

IF ILOPT(7) THEN DO /* IF POSSIBILITY OF LEVEL 4 INC- */ /* LUSIVENESS PROCESSING. */

OPEN FILE(LADD) TITLE('LADA')
OPEN FILE(ITCM) TITLE('ITEM') END
IF IBUFBIT=0 THEN /* IF THIS IS THE FIRST TIME THAT */ /* THE ITEMGET OVERLAY HAS BEEN */ /* CALLED. (PRIMES INPUT BUFFER). */ /* READ, OR IF NECESSARY, READ A SECTION. */
IF LEV='4' AND KREPT LE 'KTMP' THEN DO /* IF READ. */ STRUKEY=KEYAR(KREPT) /* RECORDED KEY FOR READ. */
READ FILE(LADD) INTO(S) KEY(STROKEY CAT 'C000000')
KREPT=KREPT+1 /* END. */ /* INCREMENT KEY ARRAY PTR*/
ELSE DO
IF ILOPT(8) THEN DO /* IF SEQUENTIAL INPUT OPTION. */
READ FILE(LADS) INTO(S),
STRUKEY=SUBSTR(S,3,12) /* EXTRACT ENCODED DATE. */ /* SEC-LETTER, & THEATRE. */
S=SUBSTR(S,15) /* RESET S TO GET RIL OF KEY */ END ELSE READ FILE(LADA) INTO(S) KEYTO(STRKEY)
IBUFBIT=1
ICSEC=KTM /* SAVE PTR TO START OF SECTION'S OUTPUT RECS */
IF SMOD=EBP THEN DO /* IF NEW PERFORMANCE SECTION. */ KTMP=0
ICPERF=KTM /* FORGET PREVIOUS ONE. */
LEV=1 /* JAM SET LEVEL. */
KTMP=KTMP+1

KEYAR(KTMP)=STRUKEY /* FOR FUTURE DEVICE FEEDBACK. */

J=J+1 /* END. */
TRANSBIT=STRUDEF /* KEYBITS NOW CONTAINS THE CODED */ /* FROM STRUKEY (SEE DECLARE */ /* STATEMENTS FOR SLY DEFINING). */
IOLDATE=0 /* CLEAR LEFT HALF OF IOLDATE. */
OLDKEY=KEYBITS /* CONVERSION, PUT CODED DATE INTO */ /* IOLDATE VIA DEFINING. */

IF IOLDATE LT IDR(1) THEN GOTO INPUT /* ABOVE STATEMENT SKIPS PROCESSING RECORDS */ /* THAT PRECEDE THE CURRENT DATE RANGE. */

IF IOLDATE GT IDR(2) THEN DO IF ILOPT(10) THEN
PUT FILE(SYSPRINT) SKIP(1) EDIT ('END OF DATE RANGE, EXITING FROM ITEMGET.') (A),
RETURN END

I=LENGTH(S)
IF ILOPT(1) THEN PUT FILE(SYSPRINT) SKIP(1) EDIT ((K DO K=1 TO 5)) (SKIP, COL(10), 9 (X(9), Y(1)))
((L, SUBSTR(S,L*100+1, MIN(100, I-1*100)))
DO L=0 TO (I-1)/100) (SKIP, Y(2), COL(10), A(1CC))
S=S CAT FC /* I NOW=LENGTH(S)-1. */

IF LEV='3' THEN LEV='1'

WSEEK ..
/* INDICATE NO TITLES DONE YET IN */ /* THIS SECTION. (PLAY, OPERA, CR */ /* ANTEREPISE TITLES, THAT IS). */

IGR=1 /* INITIAL RIGHT GRP PTR */
/* RESTART INITIAL RIGHT PTE, (GRF RESTART). */
/* INITIALIZE ICB PTE */

IF ITP(5) THEN
  PUT FILE(SYSPRINT), SKIP(1), EDIT
  ("INPUT BLK", 'J', 'LENGTH=', 'I', 'KEY=', 'STRUKEY', '***' '***',
   SUBSTR(S, MIN(60, I), '***' '***') (2 (A, F(4)), 5 A)),
GOTO ISTGRP
GROUPS . .
  /* TCP OF LCOF FOR GROUP ES, ENTERED AT END OF */
/* INPUT SETUP FOR EACH GDEF. */
/* IF GBN=4, GSN=4, GUM GT 0 OR IGR=I+1 THEN DO . . . . /* IF THIS GRP */
/* HAS AT LEAST ONE RCIE, ACTCF, OR TIME ENTRY */
/* OR IF IT'S THE LAST CF ONLY GEF IN THE */
/* SECTION THEN . . . . . */
ICGRP=KITM . .
  /* SAVE CURRENT NEXTREC OUTPUT PTE IN CASE OF */
/* GROUP RESTART. */
IF ITI=1 THEN DO . . . WAIT(188), . . . . END . . .
Z=REPEAT('1', 188), . . .
SECTYPE=SMOD.
IF GUM NE 0 THEN TIME=SUBSTR(S, IP(GUM), IL(GUM)),
  /* ABOVE STMTS MOVE TIME STRING INTO OUTPUT */
  /* BUFFER Z VIA DEFINING, IF IT EXISTS. */
  /* THE FOLLOWING PUT STRING CONVERTS THE DATE */
  /* TO AN 8 BYTE CHAR STRING IN Z. */
PUT STRING(DATE8), EDIT
  (IOLDATE/372+1659, TRUNC(MOD(IOLDATE, 372)/31)+1,
   MOD(IOLDATE, 31)+1) (F(4), 2 F(2) ),
THEATR=REGKEY,
  /* THEATRE FROM INPUT KEY TO Z EURO */
SUBSTR(Z, 1, 83)=CRTITL,
IR=MAX(GSN, 1), . . .
IRP=GS . .
  /* MOVE PTE TO 1ST ROLE IN GRF INTO A VARIABLE */
MORGES . .
IF GSN GT 0 THEN ROL=SUBSTR(S, IP(IRF), IL(IRF)),
ELSE ROL=' ',
IA=MAX(GBN, 1),
IAF=GB . . .
MORACT .
IF GBN GT 0 THEN ACTR = SUBSTR(S, IP(IAF), IL(IAF)),
ELSE ACTR=' ',
IF LEV GT '1' THEN GOTO PUTREC . . /* IF IN A 2ND PASS MODE. */
IF ICAP=0 THEN CAPTITL = TRANSLATE(CRTITL, CAES, SMALLS),
DO IX=1 TO IAR-1,
  /* SCE SEARCH LOOP. */
IF LENGTH(S, ACTOR(IX)) GT 0 THEN
IF S.ACTOR(IX) NE ACTR THEN GOTO NOHIT . .
IF LENGTH(S, ROLE(IX)) GT 0 THEN
IF S.ROLE(IX) NE ROL THEN GOTO NOHIT , .
IF LENGTH(S, TITLE(IX)) GT 0 THEN
IF ICAP=0 THEN DO . .
IF CAPTITL NE S.TITLE(IX) THEN GOTO NOHIT . . END . .
ELSE DO . .
IF CRTITL NE S.TITLE(IX) THEN GOTO NOHIT . . END . . END .
IF LENGTH(S, SEC(IX)) GT 0 THEN
IF S.SEC(IX) NE SMOD THEN GOTO NOHIT . .
IF LENGTH(S, THEATRE(IX)) GT 0 THEN
IF S.THEATRE(IX) NE THEAT THEN GOTO NOHIT . .
MANY(1, IX)=MANY(1, IX)+1 . . /* COUNT ACTUAL HITS DUE */
/* DUE TO SE(IX). */
LEV=S.INC(IX), . . . . . /* ASSIGN NEW PASS LEVEL */
IF LEV='1' THEN GOTO PUTREC . . /* RECORD INCLUSIVENESS. */
IF LEV='2' THEN DO . . /* GROUP INCLUSIVENESS */
  IGR=IGRDL,
  KITM=ICGRP . .
  GOTO EGRP . END . .
IF LEV='3' THEN DO . . /* SECTION INCLUSIVENESS. */
  KITM=ICSEC ,
  GOTO 'RESEC', . . END . . /* SO, LEV MUST EQUAL '4', PERFORMANCE INCLUSIVENESS. */
KITM=ICPERF . .
KREPT=1 . .
GOTO INPUT . . /* START ANOTHER PASS THROUGH THE PERFORMANCE */
<table>
<thead>
<tr>
<th>FE85</th>
<th>IF IAF=IAP1 THEN GOTO NO_BACT.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE86</td>
<td>IF IAF=IAP1 THEN GOTO NO_BACT.</td>
</tr>
<tr>
<td>FE87</td>
<td>IF IAF=IAP1 THEN GOTO NO_BACT.</td>
</tr>
<tr>
<td>FE88</td>
<td>IF IAF=IAP1 THEN GOTO NO_BACT.</td>
</tr>
<tr>
<td>FE89</td>
<td>IF IAF=IAP1 THEN GOTO NO_BACT.</td>
</tr>
<tr>
<td>FE8A</td>
<td>IF IAF=IAP1 THEN GOTO NO_BACT.</td>
</tr>
<tr>
<td>FE8B</td>
<td>IF IAF=IAP1 THEN GOTO NO_BACT.</td>
</tr>
<tr>
<td>FE8C</td>
<td>IF IAF=IAP1 THEN GOTO NO_BACT.</td>
</tr>
<tr>
<td>FE8D</td>
<td>IF IAF=IAP1 THEN GOTO NO_BACT.</td>
</tr>
<tr>
<td>FE8E</td>
<td>IF IAF=IAP1 THEN GOTO NO_BACT.</td>
</tr>
<tr>
<td>FE8F</td>
<td>IF IAF=IAP1 THEN GOTO NO_BACT.</td>
</tr>
</tbody>
</table>
ITEMS...
  IF ITR=IGR THEN GOTO GROUPS
  ITR=ITR+1
  ITRY=SUBSTR(S,ITL,2)
  IF ITRY NE FA THEN CALL ERMSP ('NEFA',ITL)
  ITR=INDEX(SUBSTR(S,ITL+1,ITR-ITL),FA)
  IF ITR=0 THEN ITR=IGR
  ELSE ITR=ITR+1
  IF ITP=ITL+2 THEN /* SAVE PTR TO ITEM */
  IL(ITP)=ITR-ITL-2 /* SAVE LENGTH OF ITEM */
  IF IOPT(2) THEN
    PUT FILE(SYSPRINT) SKIP(1) EDIT
    ('ITEM =',SUBSTR(S,ITL,ITR-ITL)) (A)
    IF ITP=2 THEN DO
      GBN=GBN+1
    IF GBN=1 THEN GH=ITP
      /* ACTOR IN A GRF IS NOW UP */
    GOTO ITEMDONE
  END
  IF ITP=2 THEN DO
    GBN=GBN+1
  IF GBN=1 THEN GS=ITP
  GOTO ITEMDONE
  IF ITP=2 THEN DO
    GBN=GBN+1
  IF GBN=1 THEN GS=ITP
  GOTO ITEMDONE
  IF TBIT THEN DO
    CALL ERMSP ('2TIM',ITL)
    GOTO ITEMS
  END
  TBIT='1'B /* NOTE TIME ENTRY ENCOUNTERED IN CURRENT GRF */
  GUM=ITP
  GOTO ITEMDONE
END
  IF CRTITLE NE ' ' THEN CALL ERMSP ('2TTL',IF(ITP))
  CRTITLE=SUBSTR(S,IP(ITP),IL(ITP))
  GOTO ITEMS
/* NO LONGER NEED ICE FOR TITLE. */
ITEMDONE...
  ITP=ITP+1
  GOTO ITEMS
ERMSG -- PROC (NOTE,LOC)
DCL NOTE CHAR (4),
PUT FILE(SYSPRINT) SKIP(1) EDIT
('******************************************** ') (A)
(NOTE,LOC) (A(4),X(2),F(6)) END
ENDPROC
  IF KITM LE MKITM THEN CALL ERMSP ('FSEC',MKITM)
  IPBIP=1 /* INDICATE END OF FILE ENCOUNTERED */
END ITEMGET..
DCL 1 Z,
5 SECTYPE CHAR(1),
5 YEAR CHAR(4),
5 MONTH CHAR(2),
5 DAY CHAR(2),
5 THEATRE CHAR(8),
5 TITLE CHAR(83),
5 ROLE CHAR(40),
5 ACTOR CHAR(40),
5 TIME CHAR(8),
TITLEB CHAR(83),
ROLEB CHAR(40),
ACTORB CHAR(40),
TITLEA(83) CHAR(1) DEF TITLEB,
ROLEA(40) CHAR(1) DEF ROLEB,
ACTORA(40) CHAR(1) DEF ACTORB,
IOTP(32) BIT(1),
SYSPRINT PRINT ENV(F(233,133))..;
GET FILE(SYSIN) EDIT(IOTP) B(1));
ON ENDFILE(SRTD) GOTO ENDP;,
INPUT..;
READ FILE(SRTD) INTO(Z);..
TITLEB=TITLE .. ROLEB=ROLE .. ACTORB=ACTOR ..
IT=IT3;
TITLESIZE ..
IF TITLEA(IT) NE ' ' THEN GOTO PROLESIZE ..
IT=IT-1 ..
IF IT GT O THEN GOTO TITLESIZE ..
PROLESIZE ..
IR=40 ..
ROLESIZE ..
IF ROLEA(IR) NE ' ' THEN GOTO PRACTBSIZE ..
IR=IR-1 ..
IF IR GT O THEN GOTO ROLESIZE ..
PRACTBSIZE ..
IA=40 ..
ACTORSIZE ..
IF ACTORA(IA) NE ' ' THEN GOTO SIZEDUP ..
IA=IA-1 ..
IF IA GT O THEN GOTO ACTORSIZE ..
SIZEDUP ..
PUT FILE(SYSPRINT) SKIP(1) EDIT
(SECTYPE,YEAR,MONTH,DAY,TIME,THEATRE,
SUBSTR(TITLE,'IT'),
SUBSTR(ROLE,'IR'),SUBSTR(ROLEA,1,IA))
('N,X(1)'),(1,COL(78),'A,COL(104)',A) ..
IF IOTP(1) THEN PUT DATA ..
GOTO INPUT ..
ENDP ..
END FORMAT ..
BIT1 BIT(1),
COMMENTS (4) CHAR(77),
HX(0 .. 15) CHAR(1) DEFINED HXDF,
PRM CHAR(100) VAR,
CARD (80) CHAR(1) INITIAL((14) (1) "",""),
C1 CHAR(1),
C2 CHAR(1),
C3 CHAR(1),
SYSPRINT PRINT ENV(F(133, 133)),
CFROM CHAR(256),
CRHALF (2) CHAR(128) DEF CFROM,
CRM (256) CHAR(1) DEF CRM,
GETDIGIT ENTRY (CHAR(1)),
TBL(0 .. 255) CHAR(1) INITIAL((256) (1) ""),
CNVAL CHAR(1) DEFINED NVAL POSITION(2),
INCNT, MAP = 1,
L = LENGTH (PRM),
IF L GE 5 THEN DO.,
PRM = PRM CAT ' ;' ;
GET STRING (PRM) DATA (INCNT, MAP),
END,
DO INCNT = 1 TO INCNT,
COMMENTS = ' ',
IC = 1,
DO NVAL = 0 TO 255,
CRM (NVAL + 1) = CNVAL,
END,
I = 0,
SRCCHX,
GET FILE(SYSIN) EDIT (C2) (A(1)),
NEXTEX,
IF C2 = ' ** THEN DO,
GET FILE(SYSIN) EDIT (COMMENTS (IC)) (A(77)),
IC = MIN (IC + 1, 4),
GOTO SRCCHX,
END,
IF C2 = ' ' OR C2 = ' ' THEN GOTO SRCCHX,
IF C2 = ' ; ' THEN GOTO PUNCH,
/* SEMICOLON DELIMITS END OF */
IF VERIFY (C2, '0123456789ABCDEF') = 1 THEN
PUT FILE(SYSPRINT) SKIP(1) EDIT ("ILLEGAL CHAR", C2) (A, X(2)),
C1 = '0',
GET FILE(SYSIN) EDIT (C3) (A(1)),
IF VERIFY (C3, '0123456789ABCDEF') = 1 THEN BIT1 = '1'E,
ELSE DO,
BIT1 = '0'B,
C1 = C2,
C2 = C3,
END,
CALL GETDIGIT (C1),
NVAL = IVAL * 16,
CALL GETDIGIT (C2),
NVAL = NVAL + IVAL,
TBL(I) = CNVAL,
IF I = 255 THEN GOTO PUNCH,
I = I + 1,
IF BIT1 THEN GOTO NEXTEX,
GOTO SRCCHX,
PUNCH,
J = 0,
DO I = 1 TO 8,
DO K = 16 TO 71 WHILE (J LT I * 32),
CARD(K) = TBL(J),
IF CARD(K) = "******" THEN DO,
K = K + 1,
CARD(K) = "******",
END,
J = J + 1,
END,
IF J NE I * 32 OR K GT 70 THEN DO,
PUT FILE(SYSPRINT) SKIP(1) EDIT
"OVERFLOW DUE TO TOO MANY QUOTES, I = 'I', I = 'I', F(4)",
RETURN,

END
CARD(K) = ****
CARD(K+1) = ****
DO L=K+2 TO 80
CARD(L) = ****
END

PUT FILE(SYSPRINT) EDIT(CARE) (A) 
END

DO MIP=1 TO MAP
IF MIP+INCNT GT 2 THEN PUT PAGE
PUT FILE(SYSPRINT) EDIT(COMMENTS) (CCL(25),A) ,
PUT FILE(SYSPRINT) SKIP ,
PUT FILE(SYSPRINT) SKIP(1) EDIT((HX(NN) DC NN=0 TO 7)
(COL(2),8 (A(1),X(15))) ,
PUT FILE(SYSPRINT) SKIP(1) EDIT('***',CBAIF(1),'***') (A) ,
PUT FILE(SYSPRINT) SKIP(1) EDIT
('***', (TBL(NN) DC NN=0 TO 127),'***') (A) ,
PUT FILE(SYSPRINT) SKIP(1) EDIT((HX(NN) DC NN=8 TO 15))
(COL(2),8 (A(1),X(15))) ,
PUT FILE(SYSPRINT) SKIP(1) EDIT('***',CBAIF(2),'***') (A) ,
PUT FILE(SYSPRINT) SKIP(1) EDIT
('***', (TBL(NN) DO NN=128 TO 255),'***') (A) ,
PUT FILE(SYSPRINT) SKIP(2) EDIT
((CNVAL,HX (CNVAL/16),HX (MOD(CNVAL,16)))
DO CVNVAL=K TO K+240 BY 16,
(TBL(CNVAL),
HX(UNSPEC(TBL(CNVAL)/16),
HX(MOD(UNSPEC(TBL(CNVAL)),16))
DO CVNVAL=K TO K+240 BY 16) DO K=0 TO 15)
(R(LN),SKIP(1),R(LN),SKIP(2)) ,
IN . . . FORMAT(16 (X(3),A,X(17),A,A)) .
END

END

GETDIGIT . . PROC(C3) . .
DCL C3 CHAR(1) ,
DO K=0 TO 15 ,
IF C3 = HX(K) THEN DO ,
IVAL=K ,
RETURN ,
END ,
END ,

PUT FILE(SYSPRINT) SKIP(1) EDIT
('****', BAD CHAR INPUT, '****',C3, '****') (A) ,
END ,

END

/*
LNDSTG JOB 99999211, WCD, MSGLEVEL=1
* EXEC LSPUPGEN, MEM=MAPRINT
* BAKUP.SYSUT1 DD *
CHARDEM . . PROC OPTIONS(MAIN) . .
DCL (C,D) CHAR(240) VAR ,
ON ENDFILE(SYSPRINT),
GET LIST(J,C) ,
D=C ,
I=LENGTH(C) ,
DO K=1 TO J ,
DO L=1 TO I ,
DO M=1 TO I SUBSTR(D,M,1)=SUBSTR(C,L,1) ,
END ,
PUT SKIP(1) LIST(D) ,
PUT SKIP(0) LIST(C) ,
END ,
PUT PAGE ,
END ,
DOCPRT...
PROC OPTIONS (MAIN)...
DCL CARD CHAR(80)...
DCL CARDBYTE (80) CHAR(1) DEF CARD...
DCL CARD1 CHAR(1) DEF CARD...
ON ENDFILE(SYSIN) GOTO PROGEND...

NEXTCARD...
GET FILE(SYSIN) EDIT(CARD) (A(80))...
IF CARD1='1' THEN PUT PAGE...
IF CARD1='0' THEN PUT SKIP...
IF CARD1='-' THEN PUT SKIP(2)...
IF CARD1='4' THEN PUT SKIP(0)...
IF CARD1='8' THEN DO...
DO I=80 TO 2 BY -1...
IF CARDBYTE(I) NE ' ' THEN GOTO THEPUT...
END...
GOTO NEXTCARD...

THEPUT...
PUT EDIT(SUBSTR(CARD,2,I-1)) (A)...
GOTO NEXTCARD...
END...

GOTO NEXTCARD...

PROGEND...
END...

COPYCAT...
PROC (PRM) OPTIONS (MAIN)...
DCL PRM CHAR(100) VAR...
DCL CARDIN CHAR(80),CARDOUT CHAR(80) INITIAL ((80) ' ')
ON ENDFILE(SYSIN) GOTO ENDPREG...

NEXTCARD...
GET FILE(SYSIN) EDIT(CARDIN) (A(80))...
IF LENGTH(PRM) GE 1 THEN SUBSTR(CARDIN,8C,1)=PRM...
PUT FILE(SYSPRNCH) EDIT(CARDOUT,CARDIN) (2 A(80))...
GOTO NEXTCARD...

ENDPREG...
END...

LSPMAIL...
PROC OPTIONS (MAIN)...
ON ERROR SNAP 'BEGIN'...
PUT FILE(SYSPRINT) SKIP(1) LIST ('ONPCODE=1', 'CNPCODE')...
PUT FILE(SYSPRINT) DATA...
END...

DCL
THESARC ENTRY(FIXED BIN(31,0)),
THESRTD ENTRY(CHAR(60),CHAR(28),FIXED BIN(31,0),
FIXED BIN(31,0),ENTRY,ENTRY),
IOTP(76) BIT(1),
IRCNT INIT(0),
ENT15 RETURNS(CHAR(240)),
SPFIELD CHAR(60),RFIELD CHAR(28),
SORTREC CHAR(240),
SORTCRD(3) CHAR(80) DEF 'SORTREC',
ISIZE FIXED BIN(31,0) INIT(45056),
ITRCDO FIXED BIN(31,0),
GET FILE(SYSIN) EDIT(IOTP,IRPT) (76 E(1),E(4))...

IF IOPT(1) THEN
  PUT FILE(SYSPRINT) SKIP(1) LIST('ISEMAIL STARTED') ,
  SFIELD=
    ' SORT FIELDS=(1,40,CH,A) ' ,
  RFIELD=
    ' RECORD TYPE=F,LENGTH=240 ' ,
  IBIT=1 ,
  CALL IHESTD(SFIELD,RFIELD,ISIZE,IHTCD,ENT15,ENT35) ,
  IF IOPT(1) THEN
    PUT FILE(SYSPRINT) SKIP(1) LIST('RETURN FROM IHESTD.') ,
  END
  ICNT=1 ,
  SORTREC=' ' ,
  IF IBIT=0 THEN DO ,
    CALL IHEARC(8) ,
    PUT FILE(SYSPRINT) EDIT('F15 GET 08') (A) ,
    RETURN(SORTREC) ,
  END ,
  IRCNT=IRCNT+1 ,
NXTCD ,
  IF ICNT GE 4 THEN DO ,
    IAV=40 ,
    PUT FILE(SYSPRINT) SKIP(1) EDIT
      ('ADDRESS NO.,IRCNT+1, TOC LONG.') (A,P(4)) ,
    GOTO FOURSCORE ,
  END
  GET FILE(SYSIN) EDIT(SORCRD(ICNT)) (A,B1) ,
  IAV= INDEX(SORCRD(ICNT),'**') ,
  IF IAV GT 0 THEN DO ,
    IF IOPT(1) THEN
      IF IOPT(1) THEN
        PUT FILE(SYSPRINT) SKIP(1) EDIT('E15 GET 00') (A) ,
        IBIT=1 ,
        CALL IHEARC(0) ,
        RETURN ,
      END ,
    ELSE
      FOURSCORE DO ,
        SUBSTR(SORCRD(ICNT),IAV,1)=' ' ,
        IF IOPT(1) THEN
          PUT FILE(SYSPRINT) SKIP(1) EDIT('E15 GET 12') (A) ,
          CALL IHEARC(12) ,
          RETURN(SORTREC) ,
        END ,
    END ,
  ICNT=ICNT+1 ,
  GOTO NXTCD ,
END ,
ENT35 ,
  PROC(SORTED) ,
  DCL SORTED CHAR(240) ,
  SORCRD(240) CHAR(1) DEF SORTED,
  SORTDF1 CHAR(40) DEF SORTED,
  SORTDF2 CHAR(40) DEF SORTED POS(4) ,
  IF IOPT(1) THEN
    PUT FILE(SYSPRINT) SKIP(1) LIST('IN ENT35') ,
    DO L=40 TO 1 BY -1 ,
    IF SORCRD(L) NE ' ' THEN GOTO GOTFSIZ ,
    END ,
    PUT FILE(SYSPRINT) SKIP(2) EDIT('BLANK FIRST LINE.') (A) ,
GOTFSIZ ,
  IAV= INDEX(SORTDF1,'/') ,
  DO J=1 TO IOPT ,
  IF LINO(SYSPRINT) GT 55 THEN PUT FILE(SYSPRINT) PAGE ,
  IF IAV GT 0 THEN
    PUT FILE(SYSPRINT) SKIP(2) EDIT
      (SUBSTR(SORTDF1,IAV+1,1-IAV),SUBSTR(SORTDF1,1,IAV-1))
      (A,X(1)) ,
    ELSE PUT SKIP(2) FILE(SYSPRINT) EDIT
      (SORTDF1) (A) ,
    PUT FILE(SYSPRINT) EDIT((SORTDF2(J) DC J=1 TO 5)) (SKIP(1),A) ,
  END ,
  IF IOPT(2) THEN DO ,
  END
  IF IOPT(2) THEN DO ,
THE PURPOSE OF THE LONDON STAGE PROJECT INPUT SYSTEM IS TO READ IN DATA IN THE FORMAT DESCRIBED IN 'LOGICAL STRUCTURE OF INPUT DATA' AND, AFTER PERFORMING LADDER UPDATES ON 'AS', 'SEE', AND CAST GROUP LADDER ENTRIES, TO PRODUCE A DATA BASE CONSISTING OF SORT RECORDS WITH FIELDS RESERVED FOR THE FOLLOWING ITEMS ON EACH RECORD:

1) DATE
2) SECTION TYPE
3) THEATRE
4) SYNTACTIC TITLE, IF ANY
5) TIME, IF ANY
6) SYNTACTIC ROLE, IF ANY
7) SYNTACTIC ACTOR, IF ANY

ALL POSSIBLE COMBINATIONS WILL BE PRODUCED IN THESE SORT RECORDS, THAT IS IF THERE IS A CAST GROUP WITH 2 ROLES AND 3 ACTORS THEN 6 SORT RECORDS WILL BE PRODUCED.

SCANNER INPUT DATA FORMAT

SCANNER INPUT IS ORIGINALLY IN THE FORM OF FIXED LENGTH BLOCKS 800 BYTES LONG, DIVIDED INTO 70 LINES OF 80 CHARACTERS EACH. THE LAST 5 CHARACTERS OF EACH LINE ARE SUPPOSED TO ALWAYS BE BLANK. THEY ARE TO BE IGNORED AND DELETED BY THE SCANNER INPUT EDITING PROGRAM, ICISCAN.

DIFFERENT BLOCKING IS PERMISSIBLE AS LONG AS THE LIFEC IS 80.

BATCHES OF INPUT DATA (ALL THE DATA INPUT TO ONE RUN OF A PROGRAM) MUST START WITH A PERFORMANCE AND MUST CONTAIN ONLY COMPLETE PERFORMANCES IN ORDER TO OBTAIN COMPLETE ACCURATE PROCESSING.

THE OUTPUT IS TO BE U FORMAT BLKS ACCEPTABLE AS INPUT TO A SYNTACTICAL PARSING PROGRAM CALLED STRCT (SHORT FOR 'STRUCTURE').

SINCE THE ICI SCANNER INTERPRETS THE DOUBLE QUOTE CHARACTER AS 2 SINGLE QUOTE CHARACTERS, ICISCAN WILL CONVERT ANY 2 CONSECUTIVE SINGLE QUOTE CHARACTERS IT FINDS INTO ONE DOUBLE QUOTE CHARACTER.

OTHER FUNCTIONS OF ICISCAN INCLUDE 'AT' SIGN PROCESSING (DEFINED ELSEWHERE), DELETING MULTIPLE BLANKS EXCEPT FOR 2 AFTER PERIODS, QUESTION MARKS, AND EXCLAMATION POINTS, CHECKING FOR ILLEGAL CHARACTERS, AND REBLOCKING INTO AN INTEGRAL NUMBER OF SECTIONS PER BLOCK, EXCEPT IN ADDITION, DOUBLE BLANKS WILL BE RETAINED AFTER PERIODS, QUESTION MARKS, OR EXCLAMATION POINTS WHEN ANY OF THESE IS FOLLOWED IMMEDIATELY BY EITHER SINGLE OR DOUBLE QUOTES IN THE LOGICAL TEXT.

STARTING WITH THE FIRST, EVERY THIRD BLK OF SCAN INPUT MUST START WITH THE TYPO'S INITIAL AND A PAGE NUMBER. THE ENTRY MUST BE THE VERY FIRST PRINTED CHARACTERS ON EACH TYPED PAGE, AND MUST BE IN THE FOLLOWING FORMAT:

1) AN ASTERISK
2) A CAPITAL LETTER
3) 1 TO 5 DECIMAL DIGITS
A blank

Typist entries on on a line will be processed before any
'AT' sign processing, so typist entries must be correct without the
help of 'AT' signs. This should

be easy since only the first few characters of the page are involved.
In input to the scan

program the deletions defined by

'AT' signs will be performed line by line in the following order:

1) If 3 contiguous 'AT' signs are found anywhere on the line then the
entire line will be deleted forthwith.

2) If 2 contiguous 'AT' signs are encountered then the 'AT' signs
and all characters between the 'AT' signs and the first blank to
the left of the 'AT' signs will be deleted.

The first blank to the left of the 'AT' signs will not be deleted.
This will be done left to right across the line repeatedly
until no double 'AT' signs are left.

3) If an isolated 'AT' sign is then found then the 'AT' sign and the
character to its left will be deleted.

This will be done left to right across the line repeatedly
until no 'AT' signs are left.

Subtleties not done by China Data..

1) Abnormal (uncertain or nonexistent) dates.

2) Some box tagging, especially very abnormal boxes.

3) Subtitle tagging.

4) Ladder changes except for replacement and addition as is
most commonly found in the text.

(that is no signed ladder changes)

5) Index tagging

6) Time notation

Items get selection/exclusion specifications

**************************

The itemget program will have the ability to select individual
sort records, groups of sort records, sections of sort records, entire
performances of sort records, and to exclude sort records outside of
certain date ranges. All of the preceding capabilities may be combined
although it is envisioned that only 1 or 2 capabilities will be used
in any one run. Selection may be based on any combination
of the items present in a sort record. These items are...

section type letter, date, theatre, title, time entry, role,
and actor.

Processing to be done for indirect sorts:

1) Mr, Mrs, Miss, Mlle, Sg, SGA, quotes, and capital followed by
a period will be placed behind the actual name, including
combinations of these. Spaces will then be taken out and
all letters converted to uppercase.

leading 'the', 'a', and 'an' will be treated similarly for titles.

Suffixes will not be affected.

*** The logical structure of input data ***

**************************

The largest component of input data is the 5/60 logical record,
referred to hereafter in this report as a Elk (EICCK). For safeet
input each blk should consist of one or more complete sections, also
called boxes. Examples of sections are...

play sections, afterpiece sections, dance sections, opera sections,
song sections, etc. The sections are grouped into performances, which
consist of one play section (mainpiece play) & all the other sections
that follow the play section and precede the next play section.

Another way to think of a performance is 'all of the boxes associated
with one theatre on one date'.

All of the above also applies to scanned input except
that there is no restriction on sections being contained
within one scanner input blk, etc.

It should be noted that sometimes a play section is used for
special purposes other than providing information about a play.
Each section starts with a three character sequence that defines both the start of a section and the end of a previous section if there was a previous one in the same block. The first char of this tag is a blank. The second char is a unique character that is never to occur following a blank and immediately preceding a small letter anywhere in the text except to define the start of a section. It is called the section tag character. For MTST input this character is the prefix. For scanner input an asterisk is used. The third character of the tag is a small letter. Only certain letters are allowed. These are listed elsewhere.

All sections except comment sections may contain 2 kinds of text, structured text, and extraneous text. A comment section consists entirely of extraneous text. Extraneous text contains only one kind of item of significance to programming, the MTST tag. Index entries occur only in extraneous text. They are delimited on the left either by a '$' or by a '}'. And on the right by an equal sign followed by 1 or 2 tag characters. If there are 2 tag characters then the first one is a decimal digit. If the tag is a blank then the type of index entry is unknown. Otherwise the tag indicates semantically the type of index entry, e.g., bold, author, street name, etc. The '$' type index entry indicates a person or place, the '}' type indicates an italicised index entry.

Index entries with a double character tag may not exceed 125 characters in length. Similarly index entries with one-character tags may not exceed 124 characters in length. Each of the above limits include the index delimiters and the equal sign.

For saved index entries always start with a blank-prefix-large letter or quote. An untaged or unknown type of index entry can end with another prefix-small letter or quote. A tagged index entry on the MTST will be indicated by an equals sign as the right-hand delimiter with the tag immediately following the equals sign. A blank in this location is another way of indicating an unknown type of index entry, as with the scanner.

Except for comment sections, extraneous text is defined in either of the following 2 ways...

1) On the left a left bracket, and on the right by one of the following...
   A) First occurrence of a percent sign following the left bracket.
   B) First occurrence of a right bracket following the left bracket provided that another left bracket does not lie between the original left bracket and the right bracket.

2) On the left with a left paren, and on the right with a right paren.

Extraneous text delimited in one of these 2 ways may contain delimiters used by the other method. That is, for example, extraneous text delimited by brackets may contain parentheses.

The delimiters are considered part of the extraneous text. Each pair of delimiters must define extraneous text to be entirely within one section. If there is no matching (right paren for left paren, right bracket or percent sign for left bracket) right delimiter following the left delimiter in the same section then the entire remainder of the section following the left delimiter will be considered extraneous text, however, an error message will be generated.

Page entries are a way of informing the computer of what page of the London stage any item of text came from. Page entries are identified anywhere in the text by a 'P' (blank-small p) followed immediately by an unsigned decimal number which is ended by a blank on the right... For example 'P132'. Note that this page entry is 6 characters long including the blanks, which are important and necessary.

Page entries can occur in front of most items, however the following restrictions and conventions must be followed... A) A page entry may not occur inside a date entry, it must either precede the entire date or else precede the theatre entry which follows the date.
SIMILARLY PAGE ENTRIES CANNOT OCCUR INSIDE A LADDER ENTRY.

PAGE ENTRIES MAY NOT FOLLOW A 'BUT' - THAT IS 'BUT P312 ITEM' IS NOT ALLOWED, EVEN THOUGH THE PAGE ENTRY DOES IMMEDIATELY PRECEDE AN ITEM.

THE PAGE ENTRY MUST IMMEDIATELY PRECEDE THE ITEM, SEPARATED FROM THE ITEM ONLY BY ONE OR MORE BLANKS OR THE EQUIVALENT (CARRIAGE RETURNS, LINES FEEDS, AND MTST RUBOUT CHARACTERS).

NOTE THAT A SECTION DELIMITER SEQUENCE OF ANY OTHER TYPE OF PUNCTUATION IS NOT AN ITEM.

PAGE ENTRIES MAY NOT FOLLOW A TIME ENTRY. INSTEAD THEY MUST EITHER PRECEDE THE TIME ENTRY OR PRECEDE THE ITEM THAT FOLLOWS THE TIME ENTRY.

ROMAN PAGE ENTRIES, ANALOGOUS TO ORDINARY PAGE ENTRIES, ARE USED IN EXTRANEOUS TEXT DERIVED FROM INTRODUCTIONS. IT IS SIMILAR TO THE ORDINARY PAGE ENTRY IN THAT IT IS PRECEDEED AND FOLLOWED BY AT LEAST ONE BLANK. IT IS SYNTACTICALLY DEFINED BY A BLANK-EXCLAMATION POINT COMBINATION IMMEDIATELY PRECEDING A SERIES OF ONE OR MORE LEGITIMATE ROMAN NUMERAL CHARACTERS AND ENDING WITH A BLANK. EXCEPT THAT IT IS ALLOWED ONLY IN INTRODUCTIONS, IT MAY OCCUR ANYWHERE THAT AN ORDINARY PAGE ENTRY COULD OCCUR IN A COMMENT SECTION.

THERE ARE 4 BASIC KINDS OF SECTIONS...

1) PLAY SECTIONS. THIS KIND OF SECTION IS REQUIRED TO HAVE A THEATRE PRECEDING ITS TITLE. IT MAY ALSO HAVE ONE OR MORE PARTS OF A DATE PRECEDING THE THEATRE.

2) SIMPLE TITLED SECTIONS DO NOT HAVE A DATE, THEATRE, OR TITLE TIME ENTRY. ALL TITLED SECTIONS EXCEPT PLAY SECTIONS MAY BE SIMPLE TITLED SECTIONS.


   A) ALL GROUPS IN THE CAST LIST WITH AT LEAST ONE ROLE OR ACTOR HAVE THE TIME ENTRY APPENDED TO THE GROUP.

   B) IF THERE ARE NO ROLES OR ACTORS IN THE CAST LIST THEN A GROUP CONTAINING ONLY THE TIME ENTRY WILL BE APPENDED TO THE END OF THE CAST LIST.

4) UNTITLED SECTIONS DO NOT HAVE SECTION TITLES, DATE ENTRIES, OR THEATRE ENTRIES. IF ANY OF THESE SORTS OF ITEMS HAPPEN TO BE IN AN UNTITLED SECTION THEN IT WILL BE TREATED AS SOMETHING IN A CAST GROUP.

   THERE MAY BE MORE THAN ONE TITLED SECTION ONE ONE DATE AT ONE THEATRE (IN ONE PERFORMANCE), BUT THERE MAY NOT BE MORE THAN ONE EACH OF ANY UNTITLED KIND OF SECTION. THERE MUST BE EXACTLY ONE PLAY SECTION, AND OF COURSE, IT COMES FIRST IN EACH PERFORMANCE.

THE FOLLOWING SECTION TYPES ARE SPECIFIED...

AFTERPIECE TITLED NO TIME ENTRIES OF ANY TYPE ALLOWED AT ALL.

AN OPERA SECTION IS REALLY A SONG SECTION HAVING A SYNTACTICAL TITLE AND CAST LIST INSTEAD OF JUST A CAST LIST. IT WILL NOT BE DISTINGUISHED BY CHINA DATA FROM OTHER SONG SECTIONS.
ANY OF THE FOLLOWING IS TO BE CONSIDERED A GROUP OF STRUCTURED
INPUT...

PERFORMANCE HEADER - DATE, IF ANY, AND THEATRE (A PAGE
ENTRY IS GENERALLY NOT CONSIDERED
A PART OF THE GROUP).

TITLE OF A SECTION
BALLET PART-BALLERINA/BALLERINO
MUSICAL INSTRUMENT-MUSICIAN
OPERA PART-OPERATOR
TRICK PART-TRICKSTER
ROLE-ACTOR GROUPS
ENTERTAINMENT-ENTERTAINER GROUPS
SONG-SINGER GROUPS
DANCE-DANCER GROUPS
MUSIC-MUSICIAN GROUPS
'AS' TYPE LADDER REFERENCES
'SEE' TYPE LADDER REFERENCES

FOR THE PURPOSES OF THE TYPIST ANY OF THE FOLLOWING THINGS IS
AN ITEM...

ROLE
ACTOR
TIME ENTRY
SYNTACTIC TITLES
SYNTACTIC SUBTITLES
(PLAY, AFTERPEICE, OPERA, BALLET, TRICK, ETC.)
OPERA
BALLET
INSTRUMENTAL MUSIC
TRICK
THEATRE

EACH PART OF A DATE (BUT FOR SOME PURPOSES THE ENTIRE DATE
IS CONSIDERED TO BE AN ITEM)

LADDER REFERENCE
INDEX ENTRY
DANCE
DANCER
MUSICAL PEICE
MUSICIAN
SONG
SINGER
ENTERTAINMENT
ENTERTAINER

THE UNTITLED TYPES OF SECTIONS (E, D, M, S) MAY EITHER CONTAIN A
CAST LIST OR THEY MAY BE EMPTY, EXISTING ONLY TO SHOW THAT THERE
WAS SOME OF THEIR PARTICULAR TYPE OF ACTIVITY ON THE GIVEN DATE
AT THE GIVEN THEATRE.

A SYNTACTIC TITLE MUST ALWAYS BE PRESENT IN A TITLED SECTION.
IF THE TITLE IS NOT KNOWN THEN EITHER 'NONE' OR 'LADR' MUST
BE PUT IN AS THE TITLE. IF SUCH A SECTION HAS A TITLE LADDER
REFERENCE THEN 'LADR' IS USED; OTHERWISE 'NONE' IS USED.
THESE ARE SPelled WITH THE FIRST LETTER CAPITALIZED, THE REST SMALL.
THEY ARE USED BOTH TO INDICATE THESE SPECIAL CASES AND TO SATISFY
THE REQUIREMENT THAT EACH TITLED SECTION HAVE A SYNTACTIC TITLE.

THE SAME THING GOES FOR THEATRE ENTRIES IN PERFORMANCE
SECTIONS, EXCEPT THAT 'NONE' IS ALWAYS USED, AND IT IS TYPED
ENTIRELY IN SMALL LETTERS. SIMILARLY, IF ALL PART OF A DATE IS
UNKNOWN THEN '0' MUST BE PUT IN FOR EACH UNKNOWN PART. THE VALUE OF
EACH PART OF A DATE THAT IS WRITTEN AS ZERO REMAINS THE SAME AS IT
WAS BEFORE, BUT THAT PART OF THE DATE IS MARKED IN THE OUTPUT DATA AS
BEING ACTUALLY UNKNOWN. THE ENTIRE DATE THEN SERVES ONLY AS AN ESTIMATE
OR GUESS FOR THE REAL DATE. SINCE NO PART OF A DATE THAT IS WRITTEN
AS 'ZERO' IS CHANGED, YOU MUST WRITE SUCCESSIVE UNCERTAIN DATES
EXPLICITLY WITH 0'S IN THE UNCERTAIN PARTS OF THE DATE.
LADDER REFERENCES ARE OF THE FORM...

'AS/SEE XX MONTH (XXXX)'  

NO SECTION MAY HAVE MORE THAN ONE TITLED TYPE LADDER REFERENCE IN IT. NO TITLE TYPE LADDER REFERENCE CAN BE INCLUDED BY ANY GROUPS IN THE SAME CAST LIST. OF COURSE IT MAY BE PRECEDED BY A TITLE, THEATRE, OR DATE ENTRY IN THE CASE OF A PERFORMANCE SECTION OR A TITLE IN THE CASE OF A TITLED SECTION.

THE END OF A LADDER ENTRY MAY BE DELIMITED EITHER BY A SEMICOLON, COMMA, MINUS SIGN, OR END OF SECTION, DEPENDING ON CIRCUMSTANCES.

'AS' TYPE LADDER REFERENCES WILL BE DONE AS DESCRIBED IN THE 5 MAY CCRB II 'WRITING INSTRUCTIONS FOR THE LONDON STAGE PROJECT'.

THE LADDER CHANGE CAPABILITIES ARE TO BE...

1) UNATTACHED ROLE SUBGROUP ADD
2) UNATTACHED ROLE SUBGROUP DELETE
3) UNATTACHED ACTOR SUBGROUP ADD
4) UNATTACHED ACTOR SUBGROUP DELETE
5) GROUP DELETE (PAIRED OR UNPAIRED)
6) ACTOR SUBGROUP & ROLE SUBGROUP REPLACE
7) ATTACHED ROLE ADD NO QUESTIONS ASKED
8) ATTACHED ROLE DELETE
9) ATTACHED ACTOR ADD NO QUESTIONS ASKED
10) ATTACHED ACTOR DELETE
11) PAIRED GROUP ADD / SAME FORMAT AS REPLACE

'SEE' TYPE TITLE LADDER ENTRIES

1) ALL FIRM GROUPS (THOSE THAT ARE NOT BROUGHT FORWARD BY THE LADDER REFERENCE) WILL REMAIN AS THEY ARE, ALL ITEMS AND GROUPS BEING CONSIDERED UNSIGNED.
2) A MATCH WILL BE ATTEMPTED ON EACH ROLE IN THE FIRM GROUPS.
   IF THE MATCHING ATTEMPT SUCCEEDS FOR ANY ROLE THEN THAT ROLE IS DELETED FROM THE QUESTIONABLE SET BEING BROUGHT FORWARD.
   IF THE ROLE DELETED IS THE ONLY ROLE IN ITS GROUP THEN THE ENTIRE GROUP IS DELETED.

CAST GROUP LADDER ENTRIES

1) MUST IMMEDIATELY PRECEDE THE FIRST DASH IN THE GROUP IF THE GROUP HAS A DASH IN IT (BECAUSE EVERY ITEM FOLLOWING THE CAST GROUP LADDER REFERENCE IS A SYNTACTIC ACTOR, NO USE CONFUSING PECULIAR).
2) KEYING IS DONE ON ALL SYNTACTIC ROLES IN THE GROUP DOING THE REFERRING. IF SOME UNMATCHED ROLES ARE STILL LEFT OVER IN THE FIRST GROUP HAVING A MATCH IN THE SECTION REFERRED TO THEN NO ERROR IS GENERATED. IF ANY OF THE ROLES IN THE REFERRING GROUP DO NOT FIND A MATCH IN THE FIRST GROUP FOUND HAVING ANY MATCH IN THE REFERRED TO SECTION THEN AN 'INKY' ERROR MESSAGE IS ('INKY' MEANS 'INCOMPLETE KEYING IN GROUP IN WHICH SOME KEYING HAS ALREADY BEEN ACHIEVED') GENERATED. IF NO MATCHES AT ALL ARE FOUND THEN A 'NCGM' ERROR ('NCGM' MEANS THAT NO MATCH FOR ANY ROLE IN THE LADDER GROUP WAS FOUND IN THE REFERRED TO SECTION) MESSAGE IS GENERATED. THE SUCCESS OF THIS SCHEME DEPENDS ON
NOT HAVING A SYNTACTIC ROLE MATCHING ANY OF THE ITEMS IN THE REFERRED GROUP PRESENT PRECEDING THE ACTUAL GROUP ON WHICH THE KEYING IS DESIGNED IN THE REFERRED TO SECTION

RELATIVELY MINOR PROGRAMMING CAN SUBSTANTIALLY ELIMINATE THIS RESTRICTION.

3) ONLY 'AS' TYPE REFERENCES ARE ALLOWED.
4) TIME ENTRIES WILL BE BROUGHT FORWARD INTACT UNLESS THE CURRENT GROUP HAS ITS OWN.
5) ARE ALLOWED ONLY IN UNTITLED SECTIONS.
6) AFTER SUCCESSFUL KEYING IS ACHIEVED, THEN ANY ACTCF IN THE REFERRED-TO GRP THAT IS MATCHED BY AN ACTOR PRECEDED BY A MINUS SIGN IN THE REFERRED GRP IS DELETED, ALONG WITH THE ACTOR PRECEDED BY THE MINUS SIGN IN THE REFERRED GRP, OF COURSE.
   IF THERE IS AN ACTOR PRECEDED BY A MINUS SIGN IN THE REFERRED GRP THAT HAS NO MATCH IN THE REFERRED-TO GROUP THEN A 'CGAD', 'CAST GROUP ACTOR DELETE' ERROR MESSAGE WILL BE GENERATED.
   (NOT YET IMPLEMENTED).
7) THE RESULT OF A SUCCESSFUL MATCH IS ALL OF THE SYNTACTIC ACTORS IN BOTH GROUPS, INCLUDING POSSIBLE REPEETITIONS WITH NO QUESTIONS ASKED, EXCEPT FOR DELETIONS AS NOTED ABOVE. REPEETITIONS MAY BE DELETED WITH RELATIVELY MINOR PROGRAMMING LATER.

1) PROGRAMMING SPECIFICATIONS
   *********************************************************
   SPECIFICATIONS TO BE DEVELOPED...

1) SYSTEM FOR TAGGING SUBTITLES TO BE USED IN LADDER KEYING.
2) SYSTEM FOR PATCHING, IF ANY.

NOTES...

1) UNTITLED SECTIONS MAY HAVE 'TITLE' TYPE LADDER REFERENCES. NATURALLY, ONLY ONE UNTITLED SECTION OF ANY ONE TYPE IS ALLOWED IN ANY PERFORMANCE.
2) NO REPEATED TITLES, INCLUDING 'NONE', ARE ALLOWED IN ANY PERFORMANCE (FOR THE SAME TYPE OF SECTION).
3) IN LADDER UPDATES TIME ENTRIES WILL BE BROUGHT FORWARD INTACT UNLESS THE CURRENT GROUP ALREADY HAS A TIME ENTRY.
4) THE MAXIMUM BLKSIZE ALLOWED FOR ANY OF THE PROGRAMS IS GUARANTEED TO BE LESS THAN OR EQUAL TO 3625, HOWEVER THE DATA LENGTH OF BLOCKS IN THE LADDER PROGRAM IS ALLOWED TO BE AS SMALL AS 3593 BYTES INCLUDING CONTROL CHARACTERS. LARGER SECTIONS THAN THIS WILL REQUIRE SPECIAL EDITING AND SMALL TO MEDIUM SIZED CHANGES IN THE EDIT, SAVEST, STRUCTUR, LADDER, & ITEMSGET PROGRAMS.
5) A SLASH WILL BE USED TO INDICATE THE END OF A LINE OF POETRY IN ALL CASES. A SLASH WILL BE USED TO INDICATE THE FIRST LINE OF A POEM. A SLASH HAS NO PROGRAMMING SIGNIFICANCE.
6) IN THE FUTURE ITALICS THAT DO NOT SIGNIFY TITLES WILL PROBABLY BE WIPED OUT.
7) TIME ENTRIES ARE NOT ALLOWED IN ANY CF AFTERPIECE SECTIONS, BUT AN ERROR MESSAGE WILL NOT NECESSARILY BE GENERATED ON DETECTING A TIME ENTRY IN THESE SECTIONS.

FOR THE MTST SYSTEM THERE ARE 7 PROGRAM STEPS IN THE DATA ENTRY AND RETRIEVAL CYCLE...

1) EDIT
2) SAVEST
3) STRUCTUR
4) LADDER
5) ITEMSGET
6) SORT
7) FORMAT

A PATCH STAGE WOULD PROBABLY WORK BEST FOLLOWING THE LADDER STEP. PERHAPS PATCHES COULD BE IN A FORMAT LIKE THE LADDER CHANGES THEMSELVES.

*** DEFINITIONS ***

***************
BALLET SECTION - A TITLED SECTION TAGGED WITH THE SMALL LETTER 'B'. ITS DATA SHOULD COME FROM A DANCE BOX.
CAST GROUP IS A CHAR STRING IN THE CAST LIST PART OF A SECTION.
IT IS DELIMITED ON THE RIGHT EITHER BY THE END OF THE SECTION OR
BY A SEMICOLON. IT IS DELIMITED ON THE LEFT BY EITHER THE
BEGINNING OF THE CAST LIST OR THE END OF THE PREVIOUS GROUP
(BY A SEMICOLON).

CAST GROUP LADDER ENTRY A LADDER ENTRY IN AN UNTITLED SECTION THAT
FOLLOWS A SYNTACTIC ROLE IN A CAST GROUP AND IMMEDIATELY
PRECEDES THE FIRST DASH IN THE GROUP IF THE GROUP HAS A DASH.

CAST LIST ITEM TITLE LADDER ENTRY ('AS' OR 'SEE'), CAST GROUP LADDER
ENTRY, CAST GROUP TIME ENTRY, SYNTACTIC ROLE, OR SYNTACTIC ACTOR.

CAST LIST ENTRY CAST GROUP.

CAST LIST TIME ENTRY AN ITEM IN A CAST GEP PRECEDING ALL ROLES
AND ACTORS IN THE GRP. IT IS DELIMITED ON THE RIGHT BY
BY A COLON-BLANK. IT CONSISTS OF THE COLON-BLANK AND
USUALLY SOME OTHER CHARACTERS FROM THE SET, '0123456789ABCDTIVX',
WHERE A, B, D, AND T ARE SMALL LETTERS. ALSO CALLED A
CAST GROUP TIME ENTRY.

EXTRANEOUS TEXT A) COMMENT SECTIONS
B) TEXT IN OTHER SECTIONS THAT IS DELIMITED AS
EXTRANEOUS TEXT.

MONOLOGUE WITH PARTS A TITLED SECTION TAGGED WITH THE SMALL
LETTER 'U'.

OPERA SECTION A TITLED SECTION TAGGED BY THE SMALL LETTER 'O'. ITS
DATA SHOULD COME FROM A SONG BOX IN THE CALENDER.

PERFORMANCE ALL OF THE SECTIONS STARTING WITH A PLAY SECTION
(EITHER REAL OR SPECIAL) AND ENDING EITHER WITH END OF DATA OR
START OF ANOTHER PLAY SECTIONS. ALTERNATIVELY ALL OF
THE SECTIONS FOR ONE THEATRE ON ONE DATE.

SECTION ALL CHARACTERS BETWEEN ONE BLANK-ASTERISK-SMALL LETTER
SEQUENCE AND THE NEXT SUCH SEQUENCE, EXCEPT THAT THE LAST SECTION
IN A BLK ENDS AT THE END OF THE BLK. THE FIRST SECTION IN A
BLK STARTS WITH THE FIRST BLANK-ASTERISK-SMALL LETTER SEQUENCE
IN THE BLK. ALL CHARACTERS OF LOGICAL DATA preceding this
SEQUENCE ARE IGNORED, EXCEPT FOR ILLEGAL CHAR CHECKS.
IN MTST INPUT THE '*' IS REPLACED BY THE PREFIX. IN
SCANNER INPUT THE BLANK PRECEDING THE ASTERISK IS NOT
ALWAYS REQUIRED, BUT TO BE PROPER IT SHOULD ALWAYS
BE THERE.

SECTION HEADER THE ITEMS FOLLOWING THE SECTION DELIMITER
CHARACTER AND PRECEDING THE CAST LIST, IF ANY. FOR UNTITLED
SECTIONS THIS IS THE SECTION TYPE LETTER. FOR TITLED
SECTIONS THE HEADER STARTS WITH THE SECTION TYPE LETTER
AND ENDS WITH THE FIRST OCCURRENCE OF ANY OF THE FOLLOWING ..
1) END OF THE SECTION.
2) PERIOD-BLANK-BLANK 3 CHAR SEQUENCE, '.', '.'.
3) QUESTION MARK-BLANK-BLANK 3 CHARACTER SEQUENCE, '? '?.
4) EXCLAMATION POINT-BLANK-BLANK 3 CHAR SEQUENCE, '! '!.

SECTION TYPE LETTER (SECTYPE) THE CHARACTER FOLLOWING THE
BLANK-ASTERISK OR BLANK-PREFIX DEFINING THE START OF A
SECTION. THIS CHARACTER SHOULD BE A SMALL LETTER DESIGNATED
TO REPRESENT A PARTICULAR TYPE OF SECTIONS.

PERFORMANCE DATE ENTRY IN A PLAY SECTION 1 TO 3 NUMBERS
SEPARATED FROM EACH OTHER AND ALL OTHER ITEMS BY AT
LEAST ONE BLANK. THE 1 TO 3 NUMBERS ARE CONSIDERED
ONE ITEM, AND HENCE ANY PAGE ENTRY MUST PRECEDE ALL
OF THEM. IF PRESENT THE DATE MUST LIE BETWEEN THE
SECTION TYPE LETTER AND THE THEATRE, IF ANY (ABSENCE OF A
THEATRE IN A PLAY SECTION IS AN ERROR).

THEATRE ENTRY A CHARACTERS STRING IN THE HEADER PART OF
A PERFORMANCE SECTION THAT FOLLOW THE DATE ENTRY, IF
ANY, SEPARATED FROM IT BY AT LEAST ONE BLANK, AND
PRECEDES THE TITLE, IF ANY, SEPARATED FROM IT BY AT LEAST
ONE BLANK. THE THEATRE BEGINS WITH THE FIRST NONBLANK
CHARACTER, EXCEPTING PERIODS AND CERTAIN OTHER PUNCTUATION
CHARACTERS, FOLLOWING THE DATE ENTRY, IF ANY. THE THEATRE
ENDS WITH THE FIRST BLANK FOLLOWING ITS START, OR ELSE
ENDS WITH THE END OF THE SECTION, WHICHEVER COMES FIRST.
THE LEGAL CHARACTERS FOR A THEATRE ARE THE SMALL LETTERS
AND THE APOSTROPHE.

TIME ENTRY TITLE TIME ENTRY OR CAST LIST TIME ENTRY.

TITLE AREA FOR A PLAY SECTION THE AREA FOLLOWING THE THEATRE
ENTRY AND ENDING WITH THE END OF THE SECTION HEADER. FOR
ANY OTHER TITLED SECTION THE AREA BETWEEN THE SECTYPE
LETTER AND END OF SECTION HEADER.

FULL TITLE STRING A CHARACTER STRING IN THE TITLE AREA.
IT STARTS WITH THE FIRST CHARACTER IN THE AREA THAT IS
NOT A PERIOD, BLANK, COMMA, SEMICOLON, QUESTION MARK,
MINUS SIGN, OR EXCLAMATION POINT. IT ENDS WITH THE END
OF THE TITLE AREA.

TITLE (SECTION TITLE) FOR PLAY AND AFTERPIECE SECTIONS THIS
IS THE PART OF THE FULL TITLE STRING PRECEDING THE SUBTITLE
IN THE STRING, IF ANY. FOR OTHER TITLED SECTIONS THIS IS
THE PORTION OF THE FULL TITLE FOLLOWING THE TIME ENTRY, IF
ANY, AND PRECEDING THE SUBTITLE, IF ANY.
HOWEVER ONLY 120 CHARS ARE PUT OUT FROM THE STRUCTURAL PROGRAM,
AND ONLY 83 ARE AVAILABLE IN SORTS.

TITLE TIME ENTRY A CHARACTER STRING IN THE FULL TITLE OF
A TITLED SECTION. IF PRESENT IT STARTS WITH THE FULL
TITLE STRING AND ENDS WITH THE FIRST 2 CHARACTER SEQUENCE
OF COLON-BLANK, ': ', IN THE FULL TITLE STRING. THE
STRING ITSELF MUST BE A LEGITIMATE CAST LIST TIME ENTRY
IN OTHER RESPECTS.

TRICK SECTION A TITLED SECTION TAGGED BY THE SMALL LETTER 'T'. ITS
DATA SHOULD COME FROM TITLED ENTERTAINMENT BOXES.

TITLED SECTION A SECTION THAT IS SUPPOSED TO HAVE A TITLE.
AT PRESENT THESE SECTIONS ARE TAGGED BY...
A, P, I, O, T, U. DISCOUNTING PAGE ENTRIES, ALL THESE SECTIONS
SHOULD HAVE A TITLE OF AT LEAST THE SPECIAL VARIETY.

STRUCTURED TEXT TEXT THAT IS NOT EXTRANEOUS TEXT.

STRUCTURED SECTION ANY KIND OF SECTION EXCEPT A COMMENT SECTION.

TITLE LADDER ENTRY A LADDER ENTRY IN EITHER A TITLED OR UNTITLED
SECTION THAT IS NOT PRECEDED BY ANY CAST GROUP ITEM. ...

THERE ARE TWO TYPES ... 'SEE' & 'AS'.

SYNTACTIC ROLE IS ANY ITEM THAT IS ...
1) IN A CAST GROUP THAT HAS A DASH OR CAST GROUP LADDER
ENTRY IN IT.
2) TO THE LEFT OF ALL DASHES AND THE CAST GROUP LADDER
ENTRY, IF ANY, IN ITS GROUP.
3) IS NOT A TIME ENTRY.
4) IS NOT A PAGE ENTRY.
5) IS NOT A LADDER ENTRY.

SYNTACTIC ACTOR IS ANY ITEM IN A CAST LIST GROUP THAT ...
1) IS TO THE RIGHT OF THE 1ST DASH IN THE GROUP IF THE
GROUP HAS A DASH IN IT, OR IS TO THE RIGHT OF THE
CAST GROUP LADDER ENTRY, IF ANY, IN A GRP THAT
HAS NO DASH.
3) IS NOT A TIME ENTRY.
4) IS NOT A LADDER ENTRY.
5) IS NOT A PAGE ENTRY.

THE RIGHT HAND DELIMITER OF A SYNTACTIC ACTOR MAY BE ...
1) A SEMICOLON.
2) END OF GROUP (EITHER SEMICOLON CF END OF SECTION).
3) COMMA.

*** STANDARD DATA FILES ***

THERE ARE 11 BASIC KINDS OF FILES ENCOMPASSING 7 GENERAL
DATA FORMATS IN USE BY THE LSP INPUT SYSTEM. THE 11 TYPES OF
FILE ARE NAMED AND BRIEFLY DESCRIBED BELOW. MORE DETAILED
DESCRIPTIONS ARE GIVEN IN THE INDIVIDUAL PROGRAM WRITEUPS.

LSPIC ORIGTIAL DATA FILE OF SCANNER OUTPUT. LSPIC FILES
ARE INPUTS TO THE ICISCAN, ICFRONT, AND ICIFIX PROGRAMS.
AND OUTPUT FROM THE ICFIX PROGRAM.
CORRECTION TEXT PORTION OF AN LSFIC FILE. FOR USE WITH
WITH ICIFRON. ALSO PRODUCED BY THE SIASH PROGRAM FROM
CARD FORMAT INPUT.

LSPDT
DATA TEXT PORTION OF AN LSFIC FILE. INPUT FOR
ICISCAN OR ICIFRON OR ICIFIX. OUTPUT FROM ICIFIX OF
SCANNER (OR LSPCNLSI ETC.).

LSPMT
ORIGINAL DATA FILE. LSPMT IS AN INPUT FOR THE SAVEDT PROGRAM.
IF IT IS DESIRED TO CREATE A BACKUP OF THE INUT THEN
AN LSPMT FILE CAN ALSO BE OUTPUT FROM THE SAVEDT PROGRAM.
SIMILARLY LSTEM FILES ARE BOTH INPUT AND OUTPUT FOR DDT.
STANDS FOR 'CLEANED'. THIS IS OPTINAL INPUT/OUTPUT OF
THE ICISCAN PROGRAM. IT IS A LINE BY LINE FORM OF TEXT
THAT HAS BEEN PUT THROUGH THE LINE READIN PROCESS AND
CLEANED OF 'AT' SIGNS AND TYPIST ENTRIES ETC. EACH LINE
IS HEADED BY THE FOLLOWING PIECES OF INFORMATION IN A
FIXED FORMAT:
1) TYPIST'S LETTER, 1 CHAR.
2) PAGE NUMBER, 4 CHAR.
3) LINE NUMBER WITHIN PAGE, 5 CHAR.
4) ABSOLUTE LINE NUMBER, 6 CHAR.
5) LENGTH OF LINE, 4 CHAR.

THUS A LSPCL FILE IS SUITABLE FOR USE AS A STREAM FILE
ALLOWING EASY MANIPULATION IN A VARIETY OF
WAYS BY A FORTRAN PROGRAM, EITHER BATCH OR RAX.

LSPPU
STANDS FOR 'PURE'. THIS IS THE MAIN OUTPUT OF SAVEDT OR ICISCAN
AND INPUT TO THE PROGRAM. IT IS TEXT CLEANED OF COMMENT
SECTIONS, AND EXTRANEOUS TEXT, AS WELL AS BUOUCUT CHARs,
ILLEGAL CHARs, AND CERTAIN CONTROL TEXT. EACH ELK SHOULd BE
1 OR MORE COMPLETE SECTIONS, AND EACH ELK SHOULD CONTAIN
ONLY COMPLETE SECTIONS, NOT JUST PART OF A SECTION.

LSPST
STANDS FOR STRUCTURED TEXT. IT IS OUTPUT FROM THE STRUCTURE
PROGRAM, AND INPUT TO THE LADDER PROGRAM. THE DATA IN THIS
FILE CONSISTS OF TAGGED GROUPS AND ITEMS, ONE SECTIONS PER ELK.

THIS IS A REGIONAL (3) U FORMAT KEYED SEQUENTIAL FILE.

LSPSS
STANDS FOR 'STRUCTURED SEQUENTIAL', LIKE LSEP, EXCEPT
THAT IT HAS THE DIRECT ACCESS KEY OF THE LSPST FILE APPENDED
TO THE FRONT OF THE BLK, AND IT IS PAIRED IN PECNT OF THE
KEY WITH 2 BLANKS TO MAKE A MINIMUM BLKSIZE OF 18 BYTES.
THIS FORMAT IS INTENDED FOR STRUCTURED OUTPUT TO TAPE.

LSPLD
STANDS FOR 'LADDER'. THIS IS OUTPUT FROM THE LADDER PROGRAM AND
AND INPUT TO THE ITEMSGET PROGRAM. IT IS A KEYED DIRECT
ACCESS FILE HOLDING SECTIONS WITH LADDER
REFERENCES CONVERTED TO EXPLICIT CAST LISTS AND TITLES.
LSPLD FILES ARE VERY SIMILAR TO LSPST FILES.

LSPS
STANDS FOR 'LADDER SEQUENTIAL'. JUST LIKE LSPLD EXCEPT
THAT, AS IN THE LSPSS FILE, THE KEY IS APPENDED TO
THE FRONT OF THE BLK AND PRECEDED BY 2 BLANKS TO MAKE THE
18 BYTE MINIMUM BLKSIZE FOR TAPE. THIS IS AN ORDINARY
U FORMAT SEQUENTIAL FILE.

LSPIT
STANDS FOR 'ITEMS'. IT IS OUTPUT FROM THE ITEMSGET PROGRAM
AND INPUT TO THE SORT STEP. THE FORMAT IS FIXED
UNBLOCKED KEYED REGIONAL (3) DIRECT ACCESS,
AND EACH RECORD USUALLY CORRESPONDS TO A SYNTACTIC ROLE/ACTOR
PAIR. HOWEVER THE RECORD MAY HOLD AS LITTLE AS THE SECTION
TYPE, DATE, AND THEATRE.

LSPIS
STANDS FOR 'ITEMS SEQUENTIAL'. SIMILAR TO LSPIT, BUT
INTENDED FOR UNKEYED SEQUENTIAL TAPE OUTPUT FROM THE
ITEMGET PROGRAM. THIS IS AN UNBLOCKED F FORMAT FILE.

LSPSR
STANDS FOR 'SORTED'. OUTPUT FROM THE SORT PROGRAM.
SAME FORMAT AS LSPIS, EXCEPT THAT MAY BE BLOCKED, IF
BLOCKED, IS USUALLY BY A FACTOR OF 19. INPUT FOR THE
FORMAT PROGRAM.

LSPIX
SIX INDEX ENTRIES. OUTPUT FROM THE SAVEDT FGm. AT PRESENT
ONLY CONTAINS THE INDEX ENTRY ITSELF. SHOULD ALSO CONTAIN

CURRENT STATUS OF PROGRAMMING

PROGRAMMING TO BE DONE --

1) INDIRECT SORT SYSTEM.
2) OPTIONAL CORRECTION OF ROLES OR ACTORS THAT START WITH 'END' ETC.
3) IMPLEMENT QUESTION MARK AS TITLE DELIMITER FOR TITLED SECTIONS.
4) ALTER LADDER TO PREVENT CGLE UPDATES FROM HAVING QUESTION MARKS APPENDED TO THEM.
5) MINOR IMPROVEMENTS IN ICIFRONT.
6) PUT RECORD COUNT IN FRMAT, OPTIONAL PRINT OUTPUT, AND OPTIONAL CHECK FOR IDENTICAL RECORDS.

STATUS OF PROGRAMS ..

CARDIN DISCONTINUED
EDT DONE
SAVEDT DONE
ICIFIX DONE
ICIFRONT MAINT STAGE
ICISCAN MAINT STAGE
STRUCTUR MAINT STAGE
LADDER MAINT STAGE
ITEMGET MAINT STAGE
FRMAT MAINT STAGE
LSPRINT DONE
DOCPRTN DONE
PLIST DONE
TRT DONE
MAP DONE
COPYCAT DONE
ICICORDER NOT DONE
SLASH MAINT STAGE

DOCUMENTATION ..

PROGRAMMING SPECS MAINT STAGE
INPUT SPECS MAINT STAGE
GENERAL HALF DONE
BUG NOTES NOT DONE
CHARACTER SET TABLES DONE
OPERATION INSTRUCTIONS MAINT STAGE
LISTINGS DONE
RECOMMENDATIONS NOT DONE

PROCEDURES ..

LSPCNLSL DONE
LSPFICSLX DONE
LSPSLASH NOT DONE
LSPFRONT DONE
LSPSLCTN DONE
LSPRINTS DONE
LSPISCAN DONE
LSPSTRUCT DONE
LSPLADDL DONE
LSPITEMS DONE
LSPFRMAT DONE
LSPSRT4 DONE
LSPSCRTT DONE
LSPMERGE DONE
LSPBACMP DONE
LSPUTGEN DONE
LSPUPGEN DONE
LSPROPCUP DONE
THE GROOVE

FOR STANDARDIZATION AND SIMPLICITY OF OPERATION, IT IS RECOMMENDED THAT THE FOLLOWING UNITS, VOLUMES, AND DSN'S BE USED..

<table>
<thead>
<tr>
<th>PROCNAME</th>
<th>UNIT</th>
<th>VOLUME</th>
<th>DSNNAME</th>
<th>UNIT</th>
<th>VOLUME</th>
<th>DSNNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEHINITT</td>
<td>ANY</td>
<td>VARIES</td>
<td>(NONE)</td>
<td>SL</td>
<td>180</td>
<td>VARIES</td>
</tr>
<tr>
<td>LSEPBACKUP</td>
<td>NL</td>
<td>182</td>
<td>-------</td>
<td>SL</td>
<td>180</td>
<td>VARIES</td>
</tr>
<tr>
<td>LSPICF</td>
<td>SL</td>
<td>182</td>
<td>VARIES</td>
<td>SL</td>
<td>180</td>
<td>VARIES</td>
</tr>
<tr>
<td>LSPFRONT</td>
<td>SL</td>
<td>181</td>
<td>VARIES</td>
<td>LSPDT</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>LSPISCL</td>
<td>SL</td>
<td>180</td>
<td>VARIES</td>
<td>SL</td>
<td>182</td>
<td>VARIES</td>
</tr>
<tr>
<td>LSPSCLIN</td>
<td>SL</td>
<td>180</td>
<td>VARIES</td>
<td>LSPCL</td>
<td>SL</td>
<td>182</td>
</tr>
<tr>
<td>LSPSTCT</td>
<td>SL</td>
<td>182</td>
<td>LSF010</td>
<td>LSEPUD</td>
<td>SL</td>
<td>180</td>
</tr>
<tr>
<td>LSPSTADD</td>
<td>SL</td>
<td>180</td>
<td>LSF010</td>
<td>STS</td>
<td>SL</td>
<td>182</td>
</tr>
<tr>
<td>LSPISTMS</td>
<td>SL</td>
<td>182</td>
<td>LSF010</td>
<td>LSPLD</td>
<td>SL</td>
<td>180</td>
</tr>
<tr>
<td>LSPSORT</td>
<td>SL</td>
<td>182</td>
<td>LSF013</td>
<td>LSPIT</td>
<td>SL</td>
<td>182</td>
</tr>
<tr>
<td>LSPMERGE</td>
<td>SL</td>
<td>ANY</td>
<td>VARIES</td>
<td>LSPSR</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>LSPFRMAT</td>
<td>SL</td>
<td>182</td>
<td>LSF014</td>
<td>LSPSR</td>
<td>SL</td>
<td>ANY</td>
</tr>
<tr>
<td>LPRINTS</td>
<td>SL</td>
<td>181</td>
<td>LSF002</td>
<td>SYSSOUT</td>
<td>SL</td>
<td>PRINTF</td>
</tr>
</tbody>
</table>

OPTION BITS FOR ICIFRONT .. (IOPT)

1) CLEAN INPUT FROM FILE DATX INSTEAD OF SCANNEE FORMAT INPUT.
2) ENABLE LOG MESSAGES.
3) ENABLE 'ORG FOR C/D' LOG MESSAGES.
   EVERY COLUMN OF THE OPTION CARD SHOULD HAVE A '0' IN IT UNLESS IT IS INDICATING A '1' OPTION.

OPTION BITS FOR ICISCAN ...

1) SIMULATE END OF DATA INPUT IMMEDIATELY (STCF).
2) PRINT ALL STRUCTURED OUTPUT SECTIONS ALONG WITH THEIR LINE NUMBERS.
3) LIST INPUT LINES AND CROSS REFERENCE INFO ON SYSPRINT.
4) SUPPRESS 'BAD TYPIST ENTRY' MESSAGE FOR ENTRIES ENDING WITH '$@' OF THE USUAL VARIETY.
5) LIST SECTIONS THAT ARE PROBABLY IN ERROR DUE TO THEIR ASSOCIATION WITH AN ILLEGAL DELIM OR THEIR NOT HAVING A SMALL LETTER AFTER THE '$@'.
6) NOTE TYPIST PAGE ENTRIES AS THEY ARE ENCUENTERED IN LINE READIN.
7) CLEANED INPUT FROM FILE CLIN.
8) PRINT PIECE OF TEXT PRECEDING ILLEGAL DELIM, FROM DELIM BEFORE ILLEGAL ONE TO THE ILLEGAL ONE INCLUSIVE, IMMEDIATELY FOLLOWING THE ILLEGAL DELIM MESSAGE ITSELF.
9) PRINT A LINE OF CERTAIN LIUBUGGING INFO ON ENCOUNTERING A DELIM.
10) PERFORM ILLEGAL CHAR CHECK.
11) ENABLE ILLEGAL DELIM ERROR MESSAGE.
12) PRINT INDEX ENTRIES LEGITIMATELY PROCESSED. CERTAIN INCORRECTLY DELIMITED ONES OR OUT OF CONTEXT ONES ARE NOT CAUGHT.
13) CLEANED OUTPUT TO FILE CLOUT.
14) SKIP INPUT LINES, BUT STILL COUNT THEM AS USUAL - IN PARTICULAR BLANK LINES ARE STILL NOT COUNTED.
15) REVERT TO REGULAR TEXT MODE IF IN CORRECTION TEXT MODE ONLY UPON ENCOUNTERING A PAGE ENTRY ON A LINE THAT HAS NO SLASHES (DESIRABLE FOR FIRST RUN WHEN CORRECTION TEXT IS LIKELY TO BE PRESENT).

OPTION BITS FOR SAVEDT .. (IOPT)

1) PRINT TRANSLATED INPUT BLKS CLEANED OF EXTRA FEED CHAR.
2) PRINT ERR MSG CONCERNING MTST ENTRY STARTING TCC FAR AFTER BEGINNING OF BLK.
4) PRINT ERR MSG CONCERNING LACK OF '1/2' CHAR WITHIN 70 BYTES OF EOB.
5) PRINT ERR MSG CONCERNING ILLEGAL(C.9.6) CHAR IN TEXT.
6) PRINT MSG CONCERNING START OF EACH NEW SECTION.
7) PRINT MAP OF EACH INPUT BLK.
8) PRINT EACH PURIFIED TEXT BLK WRITTEN TO 'FURED' FILE.
11) PRINT EXPANDED ERR MSG CONCERNING NO MATCHING RIGHT BRACKET.
11) PRINT LOG-TYPE ERR MESSAGES ...
14) ENABLE 'PUMP' ERROR MESSAGES.
15) PRINT ERR MSG CONCERNING NONCONSECUTIVE MST TAPES.
17) LIST INDEX ENTRIES PRODUCED.

OPTION BITS FOR STRUCTUR .. (IOPT)
1) PRINT EACH RAW INPUT SECTION.
2) PRINT ONE LINE SECTION HEADER MESSAGE.
3) PRINT MAP OF INPUT BLK WITH COORDINATES, ETC FOR LOCATING ERRORS.
4) PRINT EACH SECTION OF RAW OUTPUT.
5) SCANNER TYPE INPUT, WITH ' * ' INSTEAD OF EEFIX FOR SECTION DLN.
6) PRINT MSG AFTER SUCCESSFULLY PROCESSING ICHEMN STAGE PAGE ENTRY.
7) CREATE UNKEYED SEQL OUTPUT, KEY DATA APPENDED TO FRONT OF BLKS.
8) PUT OUT DIRECT ACCESS KEYED BLKS.
9) ACCEPT MONTHS IN LADDER REFERENCES IN ANY COMBINATION OF UPPER OR
    LOWER CASE (USED IN CONJUNCTION WITH ALL UPPER CASE MONSTR CARDS).
21) LIST EACH ITEM RETURNED BY GITM.
25) SKIP INPUT BLK OPTION.
27) SET INDICATOR TO CAUSE PRINTING OF MAE AND RAW OUTPUT FOR
    SECTIONS ASSOCIATED WITH ERRORS.

OPTION BITS FOR LADDER ..
1) PRINT MAP WITH COORDINATES OF EACH INPUT BLK.
2) CLEAR LADA FILE AT START OF EACH NEW SEASON. USED IN ORDER TO
    PROCESS UNLIMITED SIZE FILES ON TAPE.
3) SKIP INPUT BLKS OPTION (ALSO IGNORES SEASON CHANGES). FOR
    DEBUGGING OR SKIPPING AROUND SECTIONS THAT CAUSE PROGRAM
    TO EOMB ETC.
4) PRINT EACH BLK OF RAW INPUT READ DURING LADDER SEARCH.
5) ONE LINE HEADER FOR EACH INPUT BLK, GIVING LENGTH, SECTYPE,
    THEATRE, BLK NUMBER, DATE, AND 1ST PART OF BLK ITSELF.
6) SEQUENTIAL FORMAT INPUT FROM FILE STRS.
7) SEQUENTIAL FORMAT OUTPUT TO FILE IADS.
17) PRINT OLD AND NEW TITLES AFTER LADDER REFERENCE READ-IN.
24) PRINT EACH BLK OF RAW OUTPUT.
27) SET INDICATOR FOR PRINTING OF EXTRA INFO FOR SECTIONS WITH
    ERROR MESSAGES.

LADDAR HAS A DEBUG CHAIN IN COLS 33-36.

OPTION BITS FOR ITEMGET .. (IOPT)
1) PRINT MAP OF EACH INPUT SECTION IN LINES OF 100 CHARACTERS EACH WITH
    COORDINATES FOR EASY LOCATION OF CHARACTERS IN BERR.
2) PRINT EACH ITEM AS IT IS ENCOUNTERED IN INPUT/SETUP. (DEBUG).
3) PRINT EACH GRP AS IT IS ENCOUNTERED IN INPUT SETUP. (DEBUG)
4) SECTION TITLES IN CAPS.
5) PRINT HEADER FOR EACH INPUT SECTION, INCLUDES INPUT SECTION NUMBER,
    KEY UNTIL IT'S TAKEN OUT, LENGTH OF SECTION, AND FIRST FEW
    CHARACTERS OF SECTION ITSELF.
7) ALLOW FOR PERFORMANCE SELECTION VIA LEVEL 4 INCLUSIVENESS.
8) UNKEYED SEQUENTIAL INPUT FROM LADS FILE INTENDED FOR TAPE INPUT.
9) UNKEYED SEQUENTIAL OUTPUT TO ITSM FILE (INTENDED FOR TAPE OUTPUT).
10) PRINT CONTROL FLOW TRACE FOR SELECTED TRANSFERS.
11) PRINT THE SELECTION CONTROL BLK BEFORE EACH EXIT FROM THE CONTROL
    CARD INTERPRETER IN NEAT FORMAT WITH READINGS.
12) CLEAR STATS ARRAY AFTER EACH PRINTING OF IT.
19) PRINT EACH OUTPUT RECORD IN FORMAT WHICH USUALLY FITS ON 1 LINE.
29-32) ALLOW ERROR MESSAGE FOR ILLEGAL CHARACTER TO BE PRINTED
    THEATRE, TITLE, ROLE, AND ACTOR RESPECTIVELY.

OPTION BITS FOR FORMAT (IOPT) ...
1) LIST IN DATA DIRECTED FORMAT ALL VARIABLES IN FRCGEM
OPTION BITS FOR LSPRINT .. (ICFT)

64)  GET DATA TO BE LISTED FROM FILE SYSIN RATHER THAN FILE IN.
65)  NO PRINTER CARRIAGE CONTROL CHARACTER.
66)  TYPE NUMBER OF LINES COUNTED ON 1052 AFTER EACH ENDPAGE OR
     END OF RANGE.
69)  PUT LINE NUMBERS IN F(7) FORMAT STARTING IN COL(83) OF OUTPUT.
71)  DO NOT COUNT ENTIRELY BLANK LINES (CARRIAGE CTRL CHAR IS
     CONSIDERED HERE).
72)  CREATE BACKUP FILE OF INPUT.
73)  NO TRANSLATION FOR FIRST PRINTING OF LINE.
74)  NO OVERPRINT ON TOP OF FIRST LNE.
75)  ENABLE STANDARD PL/1 SYSTEM ACTION FOR ENDPAGE(SYSPRINT)
     ON-CONDITION, I.E. DON'T IGNORE ENDPAGES.
LSPRINT'S OPTION CARD FOLLOWS THE TRANSLATION TABLES. THERE
IS NO OPTION CHAIN, AND LSPRINT CHECKS FOR NO ERRORS.

PLIST OPTIONS
PLIST HAS ONLY 1 OPTION .. LIST LENGTH OF INPUT RECORDS.
THIS LENGTH LISTING IS SPECIFIED BY PUTTING ONE OR MORE CHARACTERS
IN THE PARM FIELD OF THE EXEC STATEMENT. THERE SHOULD BE NO PARM
FIELD IF A PLAIN LISTING IS DESIRED.

ICIFRONT ..
1)  NO '*' AT LOGICAL BEGINNING OF CORRECTION STATEMENT.
   WHEN LOOKING FOR THE '*' THAT DEFINES THE START OF
   EACH CORRECTION STATEMENT, THE PROGRAM FOUND ANOTHER
   NONBLANK CHARACTER FIRST. THIS IS A FLUSHING ERROR.

2)  ILLEGAL END OF PAGE ENTRY IN LINE N
   WHERE N IS THE CURRENT ABSOLUTE LINE NUMBER OF THE
   DATA TEXT. THIS IS A FAIRLY UNLIKELY ERROR INDICATING
   THAT SOMETHING IS HORRIBLY WRONG EITHER WITH THE
   CORRECTION TEXT OR WITH THE PROGRAM. THE PROGRAM
   STOPS AFTER THIS MESSAGE.

3)  NO FCN
   AFTER THE BEGINNING OF A STATEMENT NO FUNCTION
   OR COMMAND CHARACTER CANDIDATES COULD BE FOUND AT ALL.
   A FATAL ERROR.

4)  BAD PAGE SEQUENCE M,N IN LINE X.
   PAGE N FOLLOWS PAGE M IN THE DATA TEXT. PAGE N
   STARTS ON ABSOLUTE LINE X.

5)  BAD TYPIST ENTRY IN LINE N.
   IN ABSOLUTE LINE N THERE IS AN APPARENT TYPIST ENTRY
   (AN ASTERISK-CAPITAL LETTER IN COLS 1-2), BUT EITHER
   IT DIDN'T HAVE ANY DIGITS OR ELSE ITS DIGIT PORTION
   ENDED WITH A NONBLANK CHARACTER THAT WAS NOT PART
   OF A '$@' SEQUENCE.

6)  ERRONEOUS END OF CORRECTION.
   THIS MESSAGE OCCURS IN THE EVENT OF AN UNSUCCESSFUL
   FLUSHING ERROR. THE PROGRAM STOPS AFTER THIS MESSAGE.

7)  ILLEGAL FCN CHAR
   AN APPARENT FUNCTION OR COMMAND CHARACTER THAT IS
   NOT A C, D, OR I, EITHER SMALL OR CAPITAL. THIS IS
   A FLUSHING ERROR.

8)  NONBLANK FOLLOWS FCN CHAR
   THE CHARACTER FOLLOWING A LEGAL APPARENT FUNCTION
   CHARACTER IS NOT A BLANK. THIS IS A FLUSHING ERROR.
9) NO LINE NUMBER
IN AN APPARENT CORRECTION STATEMENT WITH
A LEGAL FUNCTION CHAR FOLLOWED BY A BLANK THERE IS
NOT A LEGAL LINE NUMBER SPECIFICATION. THIS IS A
FLUSHING ERROR.

10) MULTILINE CMD
A CORRECTION STATEMENT WITH A FUNCTION CODE OF 'I' AND
MULTILINE LINE SPECS. THIS IS A FLUSHING ERROR.

11) NEGATIVE 1ST WORD SPEC.
IN A PHRASE COMMAND THE LOGICAL LOCATION OF THE FIRST
WORD SPECIFIED IS BEFORE THE FIRST WORD OF THE LOGICAL
LINE. THE LOGICAL LINE STARTS AFTER THE TYPIST ENTRY,
IF ANY. THIS ERROR USUALLY OCCURS ON A LINE WITH A
TYPIST ENTRY WHEN THE CORRECTION TRIES TO CORRECT THE
TYPIST ENTRY.

12) NEGATIVE 2ND WORD SPEC
SIMILAR TO THE PRECEDING ERROR, BUT EVEN THE LAST WORD
OF THE PHRASE PRECEDES THE FIRST LOGICAL WORD OF THE LINE.

13) NO ';' (SLASH-BLANK)
EITHER THERE WAS NO ';' TO DELIMIT
THE END OF NEW CHANGE OR INSERTION TEXT IN A CORRECTION
STATEMENT, OR ELSE THE ';' OCCURRED
IMMEDIATELY SO THAT THERE WAS TEXT OF ZERO LENGTH.
THIS IS A FLUSHING ERROR.

ICISCAN ERROR MESSAGES

'***** ILLEGAL CHAR 'X' IN LINE J, Y'   X IS THE ILLEGAL CHAR, J IS
THE LINE NUMBER, AND Y IS THE DECIMAL VALUE OF THE CHARACTER
IN CASE THE CHARACTER DOESN'T PRINT (AS IS USUALLY THE CASE WITH
ILLEGAL CHARACTERS). THE ILLEGAL CHARACTER IS CONVERTED TO A SLASH
SO THAT IT MAY BE NOTICED MORE EASILY IN LATER STAGES OF THE
INPUT SYSTEM.

'$$$$$$$$ APPARENT TYPIST ENTRY IS 'X' LINES FROM PREVIOUS LINE J,
PREVIOUS PAGE WAS N' SELF EXPLANATORY. OCCURS WHENEVER A NEW
PAGE DOES NOT START 30 LINES FROM THE PREVIOUS ONE.

'$$$$$$ BAD PAGE SEQUENCE XY'   X IS THE PREVIOUS PAGE AND Y IS
THE CURRENT PAGE. THIS OCCURS WHENEVER THE NUMBER OF A PAGE
DOES NOT FOLLOW THAT OF THE PREVIOUS PAGE.

'$$$$$ BAD TYPIST ENTRY, LINE J ILLLL' WHERE 'ILLLL' IS AN IMAGE
OF THE LINE AS IT CURRENTLY EXISTS IN MEMORY AND J IS THE LINE
NUMBER. UNLESS OPTION 4 IS SPECIFIED THIS MESSAGE IS GENERATED
WHENEVER THERE IS AN ASTERISK-CAPITAL LETTER PAIR IN THE FIRST
2 PLACES OF A LINE AND THESE ARE NOT FOLLOWED BY ONE OR MORE
DECIMAL DIGITS ENDING WITH A BLANK. IF OPTION 4 IS SPECIFIED THEN
THE MESSAGE IS NOT GENERATED IN THE SPECIFIC INSTANCE THAT THE DECIMAL
DIGITS END WITH A '$' (DOLLAR SIGN-AT SIGN).
NOTE THAT ALL 3 TYPIST ENTRY ERROR MESSAGES START WITH DOLLAR SIGNS,
AND THAT ONLY TYPIST ENTRY ERROR MESSAGES START WITH DOLLAR SIGNS.

'********* BAD TRIPLE 'AT' IN REC J, ILLL' WHERE J IS THE LINE
OR RECORD NUMBER, AND 'ILLLL' IS THE CURRENT IMAGE OF THE LINE IN
MEMORY. THIS MESSAGE OCCURS WHEN A LINE HAS A TRIPLE 'AT' FOLLOWED
BY ANY NONBLANK CHARACTERS SUCH AS A FOURTH 'AT' ON THE SAME LINE.
THE LINE IS ENTIRELY LEFT OUT, HOWEVER PAGE ENTRY PROCESSING HAS
ALREADY BEEN DONE.

'********* ILLEGAL SECTION 'X' IN LINE J WHERE X IS THE CHAR FOLLOWING
AN ASTERISK IN THE STRUCTURED TEXT THAT IS NOT A SMALL LETTER.'
'E=XLILLL' WHERE 'X' IS THE SECTION TYPE CHARACTER AND 'LILLL' IS THE BODY OF THE SECTION. THIS MESSAGE, SIMILAR TO THE 'M=*XLILLL' MESSAGE, IS USED IN PLACE OF 'M=*XLILLL' WHEN THE SECTION IS ASSOCIATED WITH AN ILLEGAL DELIM OR ILLEGAL SECTYPE ERROR. IF X IS A SMALL LETTER FOR AN ILLEGAL SECTYPE ERROR THEN THE ORIGINAL CHARACTER WAS CAPITAL X.

'///// ILLEGAL DELIM 'X' IN LINE J, MCDE=N' WHERE X IS ONE OF THE 9 BASIC DELIMITERS USED BY ICISCAN, AND N IS THE MODE OF THE PIECE OF TEXT PRECEDING THE ILLEGAL DELIMITER. THE MCDE OF A PIECE DETERMINES WHAT THAT TEXT IS CONSIDERED TO BE. FOR EXAMPLE IN MODE 0 THE TEXT IS CONSIDERED PART OF SOME STRUCTURED OUTPUT SECTION, HENCE MODE 0 IS CALLED 'STRUCTURED TEXT MCDE'. MCDE -3 IS (INTENDED) FOR INDEX ENTRIES FOLLOWING 2 LEFT BRACKETS. THE MODE -3 STARTS WITH A PLUS SIGN OR DOLLAR SIGN, AND ENDS WITH AN EQUAL SIGN (PREFERABLY), HOWEVER IT MAY ALSO END WITH A RIGHT BRACKET, PERCENT SIGN, OR ASTERISK. NATURALLY, ENDING WITH ANYTHING OTHER THAN AN EQUAL SIGN CAUSES AN ILLEGAL DELIM ERROR.

'*** PAGE NUMBER TOO LONG IN OR BEFORE LINE J' THIS MESSAGE IS FOR A LONDON STAGE PAGE NUMBER OF THE DECIMAL VARIETY WITH MORE THAN 4 DIGITS. THE NEW PAGE NUMBER IS NOT ACCEPTED. NO ERROR MESSAGE IS GIVEN FOR APPARENT PAGE ENTRIES NOT ENDING WITH BLANKS - THEY ARE IGNORED.

'******** BAD PAGE IN OR BEFORE LINE J' AN APPARENT ROMAN PAGE ENTRY HAD A CHARACTER IN IT THAT WAS NOT A CAPITAL I, V, OR X. NO ERROR MESSAGE IS GIVEN FOR APPARENT ROMAN PAGE ENTRIES WITH BLANKS. THEY ARE IGNORED.

- SAVED LOG-TYPE ERROR MESSAGES .-
FUNP 'FUNNY PAGE NUMBERING' PAGE ENTRIES, OTHER THAN THE FIRST ONE IN A RUN ARE NOT IN CONSECUTIVE ORDER. THIS CHECKING IS UNRELATED TO MTST TAPE ORDER OR DATE SEQUENCE CHECKING.

MTCC NONCONSECUTIVE MTST ENTRIES.
NCRE NO RIGHT BRACKET.
IXOB INDEX OUTSIDE OF BRACKETS.
NORX NO RIGHT HAND DELIMITER OF AN INDEX ENTRY THAT HAS A LEFT
ICSE INDEX CAUGHT CROSSING SECTION OR BRACKET
BNDX LAST CHARACTER OF AN INDEX ENTRY IS NOT A SMALL LETTER OR QUOTE.

STRUCT ERRORS NOTE/LOG TYPE MESSAGE CODES .-
BDYR IN A LADDER REFERENCE THE APPARENT YEAR WAS EARLIER THAN 1660 OR LATER THAN 1800. THE LADDER REFERENCE WAS TREATED AS A CAST ITEM.
NL11 NEGATIVE LENGTH ITEM. VERY RARE. FAIRLY HARMLESS TO STRUCTURE, BUT DON'T KNOW IF IT WILL HARM LADDER. MAY OCCUR WHEN THE DELIMITER IS THE FIRST CHAR OF A POTENTIAL ITEM (IF THAT FIGURES) IN A GITM CALL AND THE DELIMITER IS FOLLOWED ONLY BY BLANKS OR PERIODS OR OTHER DELIMITERS WITHIN THE RANGE. IF IN ANY OTHER CASE YOU MAY WANT TO CALL WILL, ESPECIALLY IF LADDER BOMBS ON THE DATA. NL11 IS FOLLOWED INVARIABLY BY NL12.
NL12 RIGHT HAND DATA FOR NL11 ERROR. NL11 GIVES LEFT HAND DATA AS WITH NIT1/NIT2 PAIR OF ERROR MESSAGES.

NSEC BLK HAS NO SECTION DELIMITER. THE BLK IS OVERRIDDEN ANYWAY. IF THIS ERROR OCCURS WITH ICISCAN OUTPUT THEN ICISCAN SHOULD BE AT FAULT, IF WITH SAVED OUTPUT THEN SAVED'S INPUT DATA IS MORE LIKELY TO BE THE CAUSE.

NIT1 FIRST ERROR MESSAGE FOR MISSING ITEM WHERE ONE IS REQUIRED. THE TEXT SHOULD BE THE LEFT HAND AREA ASSOCIATED WITH THE MISSING ITEM. NIT2 ALWAYS FOLLOWS NIT1.

NIT2 SECOND ERROR MESSAGE FOR MISSING ITEM. ALWAYS FOLLOWS NIT1.
GIVES THE RIGHT HAND AREA ASSOCIATED WITH THE ITEM. THE NUMBERS ARE SUPPOSED TO BE THE LEFT AND RIGHT HAND PTS TO THE ITEM RESPECTIVELY.

LEGAL SECTION TYPE. A BLANK-PREFIX WAS ENCOUNTERED THAT WAS NOT FOLLOWED BY ONE OF THE SMALL LETTERS THAT DELIMITS A LEGAL TYPE OF SECTION.

A PERFORMANCE OR AFTERPRICE SECTION HAS NO ' ' THREE CHAR LENGTH STRING IN IT (THE 1ST OF WHICH DELIMITS THE PLAY TITLE IN THE SECTION).

4 DECIMAL NUMBERS WERE FOUND IN THE PRE-THEATRE PART OF A PLAY SECTION. THE 1ST THREE WERE INTERPRETED AS YEAR, MONTH, & DATE RESPECTIVELY. THE FOURTH WAS IGNORED.

ONE OR MORE PARTS OF THE MOST RECENT NEW DATE IS OUTSIDE ITS LEGAL RANGE. THE OLD VALUE OF THE OUT OF RANGE PART IS NOT CHANGED.

THE DIFFERENCE BETWEEN THE LAST PREVIOUS DATE AND THE NEWEST ONE IS GREATER THAN ABOUT 100 DAYS, OR ELSE IT'S EARLIER THAN THE OLD DATE. THE NEW DATE IS ACCEPTED ANYWAY. TOO BAD, IT'S PROBABLY WRONG, BUT MAYBE IT WILL BE SET RIGHT AGAIN SOON.

IN A SYNTACTIC ROLE-ACTOR GROUP AN APPARENT TIME ENTRY CONTAINED A CHARACTER THAT IS NOT ALLOWED IN TIME ENTRIES.

THE APPARENT TIME ENTRY WAS TAKEN AS A TIME ENTRY ANYWAY, COMPLETE WITH THE ILLEGAL CHARACTER.

IN AN APPARENT LADDER REFERENCE THE DAY OF THE MONTH WAS GREATER THAN 31. THE APPARENT LADDER REFERENCE WAS TREATED AS A SYNTACTIC ROLE OR ACTOR.

IN AN APPARENT LADDER REFERENCE THE MONTH OF THE YEAR COULD NOT BE DETERMINED. THE MONTH MUST BE STATED BY ONE OF THE FOLLOWING CODES . . JAN, FEB, MARCH, APRIL, MAY, JUNE, JULY, AUG, SEPT, OCT, NOV, DEC.

EACH MONTH MUST START WITH A CAPITAL, WITH THE OTHER LETTERS BEING SMALL. A PERIOD MAY FOLLOW ANY OF THESE CODES.

STILL TO BE EXPLAINED.

A PAGE NUMBER ENTRY WAS FOUND IN WHICH THE NEW PAGE NUMBER WAS MORE THAN ONE GREATER THAN THE PREVIOUS NUMBER OR ELSE WAS SMALLER THAN THE PREVIOUS PAGE NUMBER.

THE NEW PAGE NUMBER IS ACCEPTED ANYWAY.

LADDER . .

SOMETHING IS WRONG. PROGRAM ERROR. VARIABLE IXG HAS BECOME 0 DURING FINAL OUTPUT FORMATTING WHILE INTERPRETING THE CUTFUT CHAIN. CALL WILL. (NEAR LABEL ENDR, STM 521).

ON INPUT A GROUP WITH A LENGTH OF 0 WAS ENCOUNTERED. IT WAS IGNORED AND PROCESSING WAS CONTINUED WITH THE NEXT GROUP, IF ANY. (AFTER LABEL INSTGRP, NEAR STM 130).

SOMETHING WRONG. A CHAR WHICH IS SUPPOSED TO BE HEX 'FA' IS NOT HEX 'FA'. EITHER LADDER OR STRUCTURE IS TO BLAME.

NO MATCH FOR ROLE IN KEYING ATTEMPT IN A PAIRED ROLE-ACTOR GROUP THAT HAS A SIGNED ACTOR. THE GROUP IS IGNORED. PROCESSING CONTINUES WITH THE NEXT GRP. THE NUMBER POINTS TO THE ROLE.

NO MATCH IN KEY SEARCH KEYING ON AN ACTOR IN A PAIRED GRP IN WHICH THE 1ST ROLE IS SIGNED. THE GRP IS IGNORED.

PROCESSING CONTINUES WITH THE NEXT GRP. THE NUMBER POINTS TO THE 1ST ROLE IN THE GROUP.

NO MORE GROUP CONTROL BLOCKS LEFT. BEGONIES RECOMPILEATION WITH NUMBER OF GCB'S INCREASED. THE NUMBER PTS TO THE START OF THE 1ST GRP LEFT OUT. BE SURE TO CHECK THE NUMBER OF TCB'S TOO.

SUCCEEDING GRPS ARE NOT PROCESSED, AND THUS NOT WRITTEN OUT.

SAME AS NCGB, BUT FOR ITEM CONTROL BLOCKS.

EXTRA TITLE LADDER ENTRY ENCOUNTERED IN SECTION. ONLY ONE LADDER LADDER ENTRY IS ALLOWED IN A TITLED SECTION. THE SECOND AND LATER ENTRIES ARE IGNORED.

ILLEGAL SECTION LETTER.

LADDER GRP REFERRED-TO HAS NOTHING TO DELIMIT END OF THE TITLE, IF ANY, THEREFORE THE TITLE, IF ANY, WAS THE ONLY DATA THAT MIGHT HAVE BEEN GAINED BY THE LADDER REF, USUALLY LADDER GRPS ARE FOR MORE THAT JUST A TITLE.

'CAST GRP NO MATCH'. A REFERRED-TO SECTION FOR A CAST GRP LADDER REF WAS FOUND, BUT THE SECTION HAD NC MATCH FOR THE FIRST LEVEL
ROLE IN THE REFERRING GRP.

ROLE NO MATCH PROGRAM ERROR. SHOULD NEVER OCCUR. CALL WILL.

OGCE A COMPLETELY EMPTY GRP HAS OCCURRED OF BEEN FOUND SOMEWHERE IN
THE MIDST OF LADDER PROCESSING. IT MAY BE AN ERROR, BUT IN ANY
CASE SHOULD CAUSE LITTLE TROUBLE. CALL WILL.

INKY INCOMPLETE KEYING ON A CGLE. ONE OR MORE BOLES DO NOT MATCH
WHILE AT LEAST THE FIRST ROLE IN THE REFERRING GRP DOES HAVE A MATCH.

GDLT INABILITY TO KEY ON ANYTHING IN A GRP DELETION ATTEMPT.

URDT INABILITY TO KEY ON ROLE TC BE DELETED IN UNFAIRED ROLE
DELETION ATTEMPT.

UADT SAME AS URDT, BUT FOR ACTOR.

KRNP KEYED RECORD NOT FOUND. A LADDER SEARCH FAILED TO FINE THE
SECTION SOUGHT BECAUSE IT WAS NOT ON THE DISK. IF THIS MESSAGE IS
IMMEDIATELY PRECEDED BY ONE OR MORE 'CRIT=/*GRDTITL=' MESSAGES
THEN IT MAY BE JUST THE BAD MATCHES INDICATED IN THOSE MESSAGES,
THAT IS, IT FOUND THE RECORD, BUT GOT A EAL COMPARE ON THE 1ST
TRY, SO IT TRIED AGAIN ON THE NEXT TRY, BUT THEN THERE WAS NO
RECORD AT ALL WITH THE RIGHT KEY. THE NUMBER TELLS ON WHICH
TRY THE FAILURE OCCURRED. 1=1ST TRY AND INDICATES NOT EVEN
A BAD COMPARE.

IDLT IN AN ITEM DELETION IN A PAIRED GRP THE ITEM TO BE DELETED
COULD NOT BE FOUND IN THE REFERRED-TO GRP. THE ORIGINAL ITEM
IS LEFT INTACT BUT WITHOUT THE MINUS SIGN (THAT IS, IT IS NOT
DELETED EITHER), AND NO FURTHER PROCESSING IS ONE ON THE GRP.

NTEK THE DATE/TRACK MAP INDICATES THAT NO SECTION HAS BEEN WRITTEN
IN THE LADA FILE WITH THE DATE FOUND IN A LADDER REFERENCE KEY.
THE LADDER REFERENCE IS IGNORED, AND PROCESSING CONTINUES
WITH THE NEXT ITEM.

FREP FORWARD REFERENCE IN A LADDER ENTRY. THE DATE INVOLVED IS
ON OR LATER THAN THE DATE OF THE SECTION IN WHICH THE LADDER
ENTRY OCCURS. THE ENTRY IS IGNORED. PROCESSING CONTINUES WITH
THE NEXT ITEM.

TTLH NO TITLE MATCHING THE ONE IN A ECX CONTAINING A LADDER ENTRY
WAS FOUND IN ANY OF THE LADDER BLOCKS SEARCHED AS A RESULT OF
THE LADDER ENTRY. THE NUMBER POINTS TO THE LEFT-HAND
CHARACTER OF THE LADDER ENTRY.

2TIM MORE THAN ONE TIME ENTRY ENCOUNTERED IN A SINGLE GROUP.
THE SECOND ENTRY IS IGNORED. PROCESSING CONTINUES WITH
THE NEXT ITEM.

OTSZ SIZE OF AN OUTPUT BLK IS GREATER THAN 3588 BYTES, SO IT CAN'T
FIT ON A TRACK. REPROGRAMMING WILL BE NECESSARY UNLESS THE
BLOCK'S SIZE CAN BE REDUCED TO 3588 OR LESS (OR SOMETHING).
THE BLK WAS NOT WRITTEN AT ALL. THE BLK WAS PROCESSED
COMPLETELY IN OTHER RESPECTS. PROCESSING CONTINUES NORMALLY.
THE NUMBER IS THE BLK'S SIZE.

ITEMSGET ...

ITHE ILLEGAL CHARACTER IN THEATER, USUALLY A CAEITAL

ITIT ILLEGAL CHARACTER IN TITLE

1ROL ILLEGAL CHARACTER IN ROLE, OFTEN AN 'EC'

IACT ILLEGAL CHARACTER IN ACTOR

0GEP 'GRP' LENGTH IS LESS THAN 2, INCLUDING CONTROL CHARS.

NEFA PROGRAMMING ERROR SOMEWHERE, EITHER IN ITEMSGET OR BEFORE.

2TTL TWO TITLES IN THE SAME BLK, OF AT LEAST 2 ITEMS WHICH ARE NOT
ROLE, ACTORS, OR TIME ENTRIES, WHICH ARE SUPPOSED TO BE ALL
THAT EXIST DESIDES TITLES AT THIS STAGE.

2TIM 2 TIME ENTRIES ENCOUNTERED IN ONE GROUP, SECOND ONE IS IGNORED.

FSEQ PROGRAM ERROR. ITEM OUTPUT HAS NOT CAUGHT UP TO ITEM CUTFUT
ON THE DA OUTPUT FILE AT THE END OF THE PROGRAM. APEARENT
FAILURE ASSOCIATED WITH 2ND PASS FOR INCLUSIVENESS LEVELS
2, 3, OR 4. ONLY QUALITY OF OUTPUT SHOULD BE AFFECTED, BUT
CALL WILL DUE TO PROGRAM ERROR. RE-RUNING WITH INCLUSIVENESS
LEVELS 2, 3, AND 4 LEFT OUT MAY HELP.

FORMAT ..
FORMAT OF SORT RECORDS

BYTE 1  SECTION TYPE CHARACTER.  1 BYTE
BYTES 2-9  DATE IN FORMAT 'YYYYMMDD'.  8 BYTE
BYTES 10-17  THEATRE, PADDLED ON RIGHT WITH BLANKS, IN LOWER CASE.  8 BYTE
BYTES 18-100  TITLE PADDLED ON RIGHT WITH BLANKS.  83 BYTE
BYTES 101-140  ROLE, INCLUDING 2 CHAR WHICH INDICATE QUESTIONABLE STATUS (VIA 'SEE' TYPE LADDER REFERENCE) 40 BYTE
BYTES 141-180  ACTOR, IN SAME FORMAT AS ROLE.  40 BYTE
BYTES 181-188  TIME ENTRY  8 BYTE

TOTAL LENGTH IS 188 BYTES. PREFERABLY BLOCKED BY FACTOR OF 19.

/*
//OLSPS  JOB 99999211,WCD,MSGLEVEL=(1,1)
//D EXEC LSPRINTS,OPT=CARDOCSN,MEM=DOCOLSPS
//PRINT.IN DD DSNNAME=INSYS
//PRINT.INSYS DD *
OPERATING THE LONDON STAGE PROJECT INPUT SYSTEM
******************************************************************************

SUMMARY OF STEPS
******************************************************************************

*** INITIAL PROCESSING ***

LSPIBOKUP  TO CREATE SL BACKUP OF NL TAPE FROM SCANNER COMPANY.
ICIFIX  TO PRODUCE A FILE OF DATA TEXT FROM THE SL BACKUP,
DELETED ON BLANK LINES, AND LOCALE BAD TYPST ENTRY.
ICIFIX  TO PRODUCE A FILE OF CORRECTION TEXT FROM THE SL,
BACKUP, DELETING BLANK LINES.
LSPRINT  TO LIST THE FILE OF CORRECTION TEXT.
ICISCAN  TO PRINT ONTO TAPE A LIST/LOG OF THE DATA TEXT AND TO
PRODUCE PURE FILE OUTPUT FOR THE STRUCTURE PROGRAM.
LSPRINT  TO LIST ICISCAN'S LIST/LOG PRINTOUT WITH OVERPRINTING
ON THE SNA CHAIN FOR ERROR ANALYSIS AND CRCSS REFERENCE.
STRCT  FOR LOCATING DCHG ERRORS AND SEASON CHANGES THAT
DON'T WORK RIGHT.
LADDR  TO LOCATE ANY SINGLE ERRORS THAT CAUSE MANY ERROR
MESSAGES, BUT ARE EASY TO FIX. THIS STEP SHOULD
NOT BE RUN UNLESS THE STRCT STEP HAS NO BAD DCHG ERRORS,
AND NO SEASON CHANGES MISSED.

*** FIRST CYCLE ***

ANALYSIS  (DONE BY THE OPERATOR) TO LOCATE IMPORTANT ERRORS,
AND DETERMINE REMEDIAL ACTION FOR THEM.
THE FOLLOWING SPECIAL AREAS FOR THE OPERATOR TO
CHECK ARE LISTED IN ORDER OF IMPORTANCE:
1) THE FIRST LINE.
2) THE ENDING OF EACH SEASON EXCEPT THE LAST IN THE RUN.
3) THE START OF EACH MONTH OR YEAR.
4) TYPST ENTRIES.
5) SEASONS NOT STARTING WITH A PLAY SECTION. IN THIS
EVENT THE ITEMGET PROGRAM MAY BOMB OUT, BUT LADDR
SHOULD WORK PRETTY WELL.
ICIFIX  TO FIX SERIOUS ERRORS, AND POSSIBLY OTHER ERRORS,
DISCOVERED BY ICIFIX, ICISCAN, STRCT, AND POSSIBLY LADDR.
ICIFRONT  (AN OPTIONAL STEP) TO APPLY CORRECTIONS TO THE DATA TEXT.
ICISCAN  (EITHER USING LSPIBOKUP FOR CLEANED FORMAT INPUT OR
LSPIBOKUP FOR SCANNER FORMAT DATA TEXT) TO CHECK FOR
SERIOUS ERRORS, ESPECIALLY FROM APPLICATION OF CORRECTIONS.
OPTION 6 SHOULD BE USED INSTEAD OF OPTION 3.
STRCT  SAME AS FOR INITIAL PROCESSING STAGE.
LADDR  SAME AS FOR INITIAL PROCESSING STAGE.

*** SECOND (E LATER) CYCLES ***

ANALYSIS  AS IN FIRST CYCLE, BUT MAY INVOLVE DEVELOPMENT OF FIXES
ON CARDS FOR CLEANED FORMAT OUTPUT FROM ICIFRONT, TO
ICIFIX
FOR TEXT. AS IN FIRST CYCLE, BUT MAY BE SUPERSEDED
BY USE OF SLASH PROGRAM.

ICIFIX
FOR CORRECTION TEXT. MAY BE SUPERSEDED BY SLASH PROGRAM.

SLASH
MAY BE USED TO CONVERT CORRECTION TEXT ON CARDS TO
PROPER FORMAT INPUT TO ICIFRONT’S CORRECTION TEXT FILE.

ICIFRONT
MAY BE USED (OR NOT) EITHER WITH SLASH OUTPUT OR
ORIGINAL CORRECTION TEXT, EITHER WITH CLEANED FORMAT
INPUT OR SCANNER FORMAT INPUT (4 COMBINATIONS).

ICISPAN
SAME AS FIRST CYCLE.

SRTCT
SAME AS OTHER RUNS.

LADDR
SAME AS OTHER RUNS.

*** RETRIEVAL STEPS ***

ITEMS
TO SELECT SOME OR ALL OF SORT RECORDS REPRESENTED
BY LADDR OUTPUT.

SORT
TO SORT SORT RECORDS IF NECESSARY.

MERGE
(ACTUALLY PART OF SORTING PROCESS) TO COMBINE OUTPUT
OF SEVERAL SORT RUNS IF NECESSARY.

FRMAT
TO LIST SORT RECORDS.

**** INITIAL PROCESSING SEQUENCE ****

WHEN THE ORIGINAL MAGTAPE ARRIVES FROM THE SCANNER COMPANY
THE FIRST THING TO DO IS TO MAKE A BACKUP COPY ON A STANDARD LABELED
TAPE USING THE LSPCNLSL PROCEDURE. BE SURE TO TAKE THE WRITE
RING OFF THE SCANNER COMPANY’S TAPE BEFORE USING IT, FOR SAFETY.
THREE PARAMETERS MAY HAVE TO BE SPECIFIED ON THE EXEC CARD
IN THE USUAL CASE...

1) OUTVOL= IS THE VOLUME SERIAL NUMBER OF THE STANDARD
LABELED OUTPUT TAPE. DEFAULT IS 'A'.

2) OUTSEQ= IS THE OUTPUT FILE SEQUENCE NUMBER. DEFAULT IS 1.

3) OUTDSN= IS THE DATASET NAME OF THE OUTPUT FILE. THE
DEFAULT IS 'LSPIC'.

LSPIC FILE NAMES ARE DESIGNED TO DESCRIBE THE INFORMATION IN
THE FILE. THEY ARE 8 CHARACTERS LONG. EACH CHARACTER HAS
THE FOLLOWING MEANING...

(1) THE YEAR IN WHICH THE DATA IS STORED.
   A=71, B=72 ETC.

(2) THE MONTH IN THIS YEAR IN HEX, 5 = MAY, C = DEC., ETC.

(3) THE DAY OF THE MONTH BY THE FOLLOWING SYSTEM .. 1-9 = 1-9,
   A=10, B=11, ETC.

(4-6) THE LAST THREE DIGITS OF THE YEAR IN WHICH THE DATA IN THE
FILE STARTS.

(7) THE MONTH IN WHICH THE DATA STARTS, USING THE SAME CODES AS
    AS FOR CHARACTER (2) ABOVE.

(8) THE DAY ON WHICH THE DATA STARTS, SAME CODES AS (3) ABOVE.

OF COURSE USE OF THIS NAMING CONVENTION IS NOT REQUIRED, IT IS
ONLY INTENDED FOR SAFETY AND CONVENIENCE IN KEEPING TRACK OF
TAPE. IF THE USER FINDS IT INCONVENIENT THEN IT SHOULD BE IGNORED.

THE SCANNER COMPANY’S TAPE SHOULD BE MOUNTED ON UNIT 182,
AND THE OUTPUT TAPE ON UNIT 180. THESE CAN BE ALTERED BY USING
INUNIT= AND OUTUNIT= ON THE EXEC CARD. EXAMPLE

//JJK EXEC LSPCNLSL,OUTVOL=LPFU09,OUTDSN=A4E777C6,INUNIT=181
HERE THE INPUT UNIT IS 181 INSTEAD OF THE DEFAULT 182, AND
THE OUTSEQ= PARAMETER WAS NOT NEEDED SINCE THE FILE SEQUENCE
WAS TO BE 1.

AFTER THE BACKUP STEP IS COMPLETED REMOVE THE SCANNER
COMPANY’S TAPE. KEEP IT UNTIL 2 OTHER COPIES OF ITS DATA HAVE
BEEN MADE AND CHECKED.

AFTER THE BACKUP, THE NEXT STEP IS TO SEPARATE THE CORRECTION
TEXT FROM THE DATA TEXT, CREATING TWO NEW FILES WITH THE
ICTIDYFX PROGRAM. WHEN CREATING THE DATA TEXT ENABLE TYPEST ENTRY
CHECKING AND LEAVE A RECORD OF BAD PAGE ENTRIES ON THE CONSOL
DO NOT ALTER THE TEXT IN ANY WAY EXCEPT FOR SEPARATING THE TWO KINDS OF TEXT AND POSSIBLY DELETING BLOCK FILLER AT THE END OF THE FILE. THE REASON FOR NOT TAMPERING IS TO PRESERVE THE ORIGINAL DATA TEXT FOR ACCOUNTING PURPOSES.

ICEPICX OPERATIONS ARE DESCRIBED IN THE ICEPICX PROGRAM WRITEUP.

WHEN A REVISION IS MADE OF A BATCH THEN THE DATE OF STORAGE, GIVEN IN COLUMNS 1-3 OF THE NEW DSN, SHOULD REFLECT THE DATE OF THE REVISION.

AFTER SEPARATION PRINT THE CORRECTION TEXT USING THE LSPRINTS PROCEDURE WITH OPT=DIRECSN. IF YOU ARE ANXIOUS TO SEE THE ORIGINAL DATA TEXT THEN, OF COURSE, GO RIGHT TO THE ICISCAN LIST/LOG STEP.


1) INVOL INPUT VOLUME LABEL.
2) INDSN INPUT DATASET NAME.
3) INSEQ INPUT FILE SEQUENCE NUMBER, DEFAULT IS 1.
4) OUTSEQ OUTPUT FILE SEQUENCE NUMBER, DEFAULT IS 1.

THE RUN DECK SHOULD HAVE 2 DD CARDS FOR SYSPRINT, 1 FOR THE PRINTER DIRECTLY, THE OTHER FOR OUTPUT TO LSC02 ON UNIT 181. PLACE THE ONE THAT YOU WANT TO BE USED AHEAD OF THE OTHER ONE IN THE DECK. IF YOU ARE USING TAPE FOR SYSPRINT THEN IT IS SOMEWHAT MORE EFFICIENT TO PLACE THE LSPRINT STEP IMMEDIATELY AFTER THE ICISCAN STEP AND HAVE THE TAPE PRINTED IMMEDIATELY.


PRINTING OF ICISCAN'S LIST/LOG SHOULD BE DONE WITH THE LSPRINTS PROCEDURE USING OPT=PRINTSN.

IT IS ALLRIGHT IF THE PRINTER IS ALLOWED TO RUN OUT OF PAPER IN THIS STEP. THE OUTPUT WILL BE BULKY ANYWAY, BUT THE NUMBER OF PIECES SHOULD NOT EXCEED 3 IF POSSIBLE. THE PRINTOUT IS INTENDED MAINLY FOR REFERENCE IN THE ERROR CORRECTION PROCESS.

IF THE QUALITY IS NOT TOO LOW THEN, AFTER CORRECTION NOTES RTC. HAVE BEEN MADE ON IT, THIS PRINTOUT SHOULD BE SENT TO DR. SCHNEIDER. SCRAP PAPER SHOULD BE USED FOR THIS PRINTOUT IF THERE ARE ANY SIZEABLE CHUNKS AVAILABLE.

HERE IS A LIST OF ITEMS REQUIRING SPECIAL ATTENTION IN RUNNING STRCT, ESPECIALLY FOR THE FIRST TIME ON A NEW BATCH .

1) CHECK THE MONTH ABBREVIATIONS USED IN LADDER REFERENCES, AND CHANGE THE 'MOMENT' CARDS ACCORDINGLY. SOME FARTS USE 'AER' WHILE OTHERS USE 'APRIL', ETC. (MARCH, APRIL, MAY, JUNE, JULY, AND AUGUST ARE THE ONES TO WATCH).

2) IT IS IMPORTANT TO BE SURE ABOUT THE SEASON ENDS IF YOU HOPE TO RUN THE OUTPUT THROUGH LADDNR. ONE LIKELY ERROR IS COUNTING SECTIONS IN THE LAST PERFORMANCE OF A SEASON THAT ARE NOT PUT OUT BY ICISCAN. ANOTHER IS OVERLOOKING THE DELETION OF AN ASTERISK BY AT SIGNS. ILLEGAL SECTIONS THAT HAVE CAPITAL LETTERS ARE PUT OUT BY ICISCAN AS REGULAR SECTIONS WITH LEGAL SMALL
3) Check the JCL cursorly to make sure no unwanted override cards are there. Struct and ladder are somewhat dangerous in this respect since they are executed by straight JCL, not procedures. Special cards left over from the previous run are likely to cause trouble.

The ladder step should not be run until the structure step produces no bad 'dchg' messages. This is one in which the date is off by 5 days or more. Additionally, all season changes must occur properly. A mere change is not enough, it must be entirely correct.

*** ERROR CORRECTION CYCLES ***

After the first cycle the error correction ICPF analysis step may include analysis of correction text errors and a comparison of iciscan and/or struct errors in using icifront vs. not using it. The most likely serious error in correction text involves a correction specification that indicates an incorrect typist or advances the page number too much. This can usually be spotted when there are very many 'corrections' done to one line in the update log printout, or when the program stops and there are still corrections left to do.

The icifix data text step is as described for the initial processing sequence, except that the input is the most recent version of the data text.

If icifront is to be used again and errors have been found previously then the icifrony correction text correction text step will have to be done. Do not enable typist entry checking for this step. This must be done by hand for correction text.

Use the lspfront procedure to run icifront. The standard units are 181 for data text, 182 for correction text, and 180 for cleaned format output. The results may look fairly bad at first, but initially icifront's output improves dramatically with just a small amount of fixing in both the data and correction texts.

If icifront was not used then the iciscan step should be run with the lpsiscan procedure as in the initial processing stage. The iciscan case is described in the following paragraphs.

LSPSCLIN is used to run the cleaned format output through the iciscan program. The input file is named clin. The default input unit is 180 as with the lpsiscan procedure. The standard output unit and volume are also identical to those of the lpsiscan run. To avoid overwriting previous output you may use a higher outseq number (as long as there is room on the tape).

Icifront seems to increase the error rate when used on raw data and correction text. However when the data text has had all of its bad typist entries fixed and certain horrible correction text errors have been fixed then use of icifront is likely to be beneficial. The real stickler is that it may create a new bad dchg error. Another problem is that an error may be corrected by both icifront and icifix. The solution here is to delete the correction from the correction text and correct the data text with icifix.

Although not essential, it is probably better to put the sysprint output on tape and use lpsprint rather than print directly, since brackets are so relevant at this stage.

After running iciscan a decision must be made as to whether to go on with the present cycle or to go back and use icifix to correct major errors and then rerun iciscan again. This decision should be based primarily on the same details as are described in the analysis step. Usually it is best to go ahead and run struct regardless of errors.
IN RUNNING STRUCTURES IT IS PROBABLY BEST TO JUST PRINT
DIRECTLY ON THE PRINTER RATHER THAN USE UNIT 181 UNLESS IT
IS ONE OF THE FINAL CYCLES OF THE DATA ENTRY STAGE. THERE IS
NO TAPE PROCEDURE FOR THIS STEP, A STRAIGHT PROGRAM IS USED
SINCE IT IS ANTICIPATED THAT NO FLEXIBILITY WILL BE NEEDED IN
TAPE ALLOCATION ETC.

IN RUNNING STRUCT THE SAME CONSIDERATIONS FOR OVERWRITING
PREVIOUS OUTPUT APPLY AS WITH THE IClSCAN PROGRAM. SINCE STRUCT
IS RUN BY STRAIGHT JCL THE ACTUAL DD CARD IN THE RUN DECK
WILL HAVE TO BE REPLACED OR SUPERSEDED BY ONE WITH THE SAME
NAME IN THE DECK.

CONDITIONS FOR CONTINUING WITH LADDER RATHER THAN RESTARTING
AT THE ANALYSIS STEP ARE ALWAYS THE SAME - NO EAD EVENTS ERRORS,
AND NO SEASON CHANGES MISSED.

*** RETRIEVAL STEPS ***

IF SORT RECORDS ARE TO BE PRODUCED THEN AFTER
A SUCCESSFUL LADDER RUN YOU SHOULD ALMOST CERTAINLY
RUN ITEMS RATHER THAN RESTART THE CYCLE. ONLY RESTART IF THERE
IS AN EASY WAY TO GET RID OF A SUBSTANTIAL PORTION OF ERRORS.
TO SELECT ALL RECORDS IN A BATCH USE THE FOLLOWING STATEMENTS
AFTER THE OPTION CARD .
1) DATE RANGE, '5 1659 07 01 1800 12 31'
2) SE CARD TO CAUSE SELECTION OF ALL RECORDS, '11'
3) GO CARD, '7'
4) STOP CARD, '9'

ITEMS SHOULD TAKE APPROXIMATELY 40 MINUTES PER OUTPUT REEL, PLUS
SEVERAL MINUTES CHANGEOVER. BATCH 1 TOOK APPROXIMATELY 2 REELS.
BEFORE RUNNING CHECK AND SEE IF ANY BATCH (ESP NO. 3) HAS A
MUCH BIGGER OUTPUT CHAR COUNT IN THE LADDER PROGRAM, AND IF SO
THEN ADD MORE NAMES TO THE OUTPUT VOLUME LIST ON THE ITSMD DD
STATEMENT. THE STEPNAM IS 'GO'.

THE SORT STEP, IF ANY, NEED NOT USE ANYTHING PECULIAR TO
THE LONDON STAGE PROJECT SINCE THE PROGRAM INVOLVED IS SUPPLIED
BY IBM, HOWEVER THERE ARE 2 PROCEDURES PROVIDED, LS3FSCRT AND
LS3PSRTT. LS3PSRTT IS USED FOR SORTS HAVING TAPE INPUT AND
OUTPUT, USING DISKS AS INTERMEDIATE STORAGE AREAS. WITH SUFFICIENT
SPACE ON THE DISKS IT CAN SORT APPRX 50,000 RECORDS PER RUN.
THIS IS MORE THAN 1 TAPEFULL (UNBLOCKED) BUT LESS THAN 2 TAPE
FULL. BATCH NUMBER 1 (WITH ONLY SLIGHT CORRECTIONS IN IT)
HAS APPROXIMATELY 64,000 SORT RECORDS. A FAIRLY GOOD WAY
TO GET AROUND THIS IS TO 1) DO N SEPARATE SORTS, EACH PRODUCING
ONE TAPE OF OUTPUT FROM ONE TAPE OF INPUT, AND 2) THEN DO
SUCCESSIVE 2 WAY MERGES OF THESE UNTIL ONLY ONE FILE (CF 1 OR
MORE TAPEs) IS LEFT. THIS MERGING PROCESS IS FACILITATED BY
THE LS3PSRTT PROCEDURE.

THE OUTPUT OF THE SORT STEP CAN BE
USED AS INPUT TO ANOTHER SORT STEP, BUT IF OUTPUT IS BLOCKED THEN
ONE SORT STEP MAY NOT BE ABLE TO HANDLE AN ENTIRE TAPE AT ONE
SHOT. ONE SOLUTION IS TO USE THE LS3PBBLK PROCEDURE TO UNBLOCK
THE DATA ONTO SEVERAL TAPEs. AN UNBLOCKED TAPE CAN HOLD APPRX.
32,000 RECORDS. A TAPE BLOCKED BY A FACTOR OF 16 CAN HOLD APPRX.
96,000 RECORDS.

THE PRMTU PROGRAM PRINTS, IN A HOPEFULLY NEAT MANNER, OUTPUT
FROM THE ITEMS OR SORT STEPS. THE TAPE PROCEDURE FOR THIS IS
LS3PSRTT. THE NORMALLY REQUIRED PARAMETERS ARE INUNIT=, & INVOI=.
INVOI= MAY ACTUALLY BE A LIST OF SEVERAL VOLUMES ENCLOSED IN
QUOTES, FOR EXAMPLE .
"F EXEC LS3PSRTT,INUNIT=180,INVOI='LSF018,LSF034'
DEFAULTS ARE INUNIT=182, INVOI=LSF018.

/*
*ICIFORM JOB 99999211, WCD, MSGLEVEL=(1,1)


**** ORIGINAL MAGTAPE INPUT FROM SCANNER ****

THE SCANNER COMPANY PRODUCES NONLABELLED TAPE FORMATTED THUS...

DCB=(RECFM=PB,BLKSIZE=800,IRECL=80).

Each logical record nominally corresponds to a line of text on the China data sheets. There may be an occasional extra line of 80 blanks, usually occurring at the end of a page. Each page is supposed to have 30 lines. Each line of ten 80-byte records is one third of a page. If there is an extra blank line then the other lines following it are pushed down even into the next blk if necessary to keep the blksize at 800.

The typed portion of each line is nominally the first 75 chars of the line, the remaining 5 chars being blank on the China data sheets except in case of error. After scanning to tape this is still generally true, but often a typed double quote char is scanned as 2 apostrophes, thus causing the 75th typed char to spill over into column 76. To solve this problem any pair of contiguous apostrophes found on a line is replaced with a double quote char by the iciscan program. Occasionally nonblank chars are put into column 76 or beyond for other reasons, all errors of some sort. These errors are accepted up to the last nonblank char anyway. No error message is generated for this.

The characters are standard ebcDIC except for the following...

The British pound sign is an ebcDIC number sign, hex '7E'.

It is printed by the lsprint sn tables as an 'I' overprinted on a minus sign. In the icifix program the 1052 prints it as just what it is internally, the number sign. Neither the BN nor the SN chain can print the number sign character.

The left and right brackets are hex 'AD' and 'BD' respectively. They print as the parentheses overprinted with the minus sign using the LSPRINT SN tables. On the 1052 they are printed as the 'LESS THAN' and 'GREATER THAN' symbols respectively by ICIFIX.

The character the scanner company is supposed to use for nonecs is the hex '00'. This is printed by the SN LSPRINT Tables as a zero overprinted on the prefix char. Other illegal chars print as the usual 'H' on 'I' char.

/*
//FILXLIST JOB 99999211,WCMD,MSGLEVEL=(1,1)
//D EXEC LSPRINTS,OPT=CARDOCSN,MEM=DOCPIX
//PRINT.IN DD DDNAME=INSYS
//PRINT.INSYS DD *

ICIFIX GENERAL DESCRIPTION

ICIFIX IS AN INTERACTIVE TEXT EDITING PROGRAM OPERATED FROM THE OPERATOR'S CONSOLE OF AN OS/360 SYSTEM. IT IS LINE ORIENTED AND SPECIALIZED FOR FIXING A SMALL NUMBER OF ERRORS IN RAW DATA IN CI FORMAT. ITS INPUT IS A SEQUENTIAL FILE IN, WITH AN LRECL OF 80. PRINCIPAL OUTPUT IS TO ANOTHER SEQUENTIAL FILE, TOUT, WITH THE SAME LRECL. OUTPUT MAY ALSO GO TO THE PRINTER, AND, IF CCUSE, COMMANDS AND PROGRAM RESPONSES ARE MADE ON THE CONSOLE.

HERE IS A DESCRIPTION OF THE COMMANDS...

(PARTS OF COMMAND EXAMPLES INSIDE PARENTHESES ARE OPTIONAL)

1) 'CHOP' TERMINATES AN EDITING SESSION IMMEDIATELY, NOT PUTTING OUT ANY MORE LINES TO THE OUTPUT FILE. THE CURRENT LINE IS NOT PUT OUT. LINES PRECEDING THE CURRENT LINE HAVE ALREADY BEEN PUT OUT. 'CHOP' IS TYPED IN SMALL LETTERS.

2) 'BACKUP' TRANSFERS CURRENT LINE AND REMAINDERS OF INPUT FILE TO OUTPUT FILE, THEN TERMINATES PROGRAM. TYPED IN SMALL LETTERS.

3) 'ND' WHERE 'N' IS A NONNEGATIVE INTEGER, AND 'D' IS A SMALL 'D'. CAUSES THE DELETION OF N LINES BY READING N LINES FROM THE INPUT FILE AND NOT PUTTING THEM OUT TO THE OUTPUT FILE. IF 'D' ALONE IS TYPED THEN '1D' IS ASSUMED.

4) 'N' WHERE 'N' IS A NONNEGATIVE INTEGER. READS N LINES, PUTTING INPUT INTO THE OUTPUT FILE, STARTING WITH THE CURRENT LINE. THIS IS CALLED THE 'ADVANCE' COMMAND.
5) 'NL' WHERE 'N' IS A NONNEGATIVE INTEGER, AND 'L' IS THE SMALL LETTER, 'L'. SETS THE LINE COUNT LIMIT FOR A SEARCH TO N. N=0 MEANS NO SEARCH AT ALL. N=1 MEANS SAME LINE ONLY.

6) 'C{CENT SIGN-C'} 'G' IS THE CENT SIGN. 'C' IS ANY CHARACTER STRING EXCEPT ONE CONTAINING A CENT SIGN OR ONE CONSISTING ONLY OF BLANKS. THIS COMMAND CAUSES THE PROGRAM TO SEARCH FOR THE CHARACTER (NOT THE STRING) FOLLOWING THE CENT SIGN FOR UP TO THE NUMBER OF LINES ALLOWED BY THE SEARCH LIMIT (SEE COMMAND NO. 5, ABOVE). IF THE CHAR IS FOUND THEN THE LINE IS TYPED UP TO AND INCLUDING THE CHAR OF THE SEARCH. YOU MAY THEN OVERLAY NEW TEXT, BEGINNING WITH THE SEARCH CHAR, UP TO THE END OF THE LINE. THIS IS DESCRIBED IN THE REPLACE COMMAND BELOW.

7) 'x{CENT SIGN-x} (Y{CENT SIGN-y})' THE REPLACE (AND POSSIBLY SEARCH AGAIN) COMMAND. THE BASIC FORM OF THIS COMMAND IS 'x{CENT SIGN-x}' WHERE 'X' IS A CHAR STRING, AND '{CENT SIGN-y}' IS THE CENT SIGN. THIS COMMAND IS INTENDED TO BE GIVEN AFTER A SEARCH COMMAND, FOR EXAMPLE COMMAND NO. 6. THE REPLACE COMMAND ALLOWS YOU TO REPLACE CHARACTERS, BEGINNING WITH THE SEARCH CHAR ITSELF (THE LAST CHAR TO BE TYPED IF THE SEARCH IS SUCCESSFUL) AND EXTENDING FOR THE LENGTH OF X OR TO THE END OF THE LINE, WHICHEVER IS SHORTER. IF THIS COMMAND IS NOT GIVEN IMMEDIATELY AFTER & IN RESPONSE TO A SEARCH COMMAND OR EQUIVALENT THEN THE REPLACEMENT STARTS WITH THE FIRST CHAR OF THE CURRENT LINE.


8) 'xx{CENT SIGN-x} (Y{CENT SIGN-y})' A CENT SIGN AS THE FIRST CHAR OF A COMMAND CONTAINING ANOTHER CENT SIGN MEANS TO CREATE A NEW LINE INSERTED BEFORE THE CURRENT ONE. THE NEW LINE MAY BE OF ANY LENGTH UP TO 75 CHARACTERS. IT WILL BE PADDED ON THE RIGHT WITH NULL CHARACTERS (PREFIXES, OR THE PL/1 'CR' SYMECL). IF THERE ARE ANY NONBLANK CHARACTERS AFTER THE SECOND CENT SIGN THEN THE CHAR AFTER THE CENT SIGN IS TAKEN TO BE A NEW SEARCH CHAR AS IN THE REPLACE COMMAND.

CREATING A LINE, OR DELETING THEM FOR THAT MATTER IS TO BE AVOIDED IF POSSIBLE SINCE IT CAUSES MORE ICIFIX ERROR MESSAGES AND UPSETS THE LINE NUMBER CROSREFERENCE SYSTEM.

9) 'CHECK' TURNS ON THE ERROR CHECKING FUNCTION.

This is typed in small letters.

10) 'NOCHECK' TURNS OFF THE ERROR CHECKING FUNCTION.

WHEN THE PROGRAM STARTS, THIS FUNCTION IS OFF.

Typed in small letters.

11) 'PRINT' AFTER EXECUTION OF THIS COMMAND LINES PUT INTO THE OUTPUT FILE WILL ALSO BE PUT INTO THE SYSPRINT FILE (IN OTHER WORDS, THEY WILL BE PRINTED ON THE PRINTER). THIS COMMAND IS TYPED IN SMALL LETTERS. THE 'PRINT' FUNCTION IS ON WHEN THE PROGRAM STARTS. IT IS VERY EASY TO FORGET AND LEAVE THE PRINT ON WHEN EXECUTING A BIG LINE CHANGE COMMAND, THUS PRINTING A LARGE PORTION OF A BATCH, UNLESS YOU CANCEL. THIS IS THE MOST IMPORTANT POINT TO REMEMBER IN OPERATING THE PROGRAM, AS IT GENERALLY MEANS RESTARTING THE EDITING SESSION.

12) 'NOPRINT' THIS TURNS OFF THE PRINT FUNCTION ABOVE. THE 'PRINT' FUNCTION IS OFF WHEN ICIFIX BEGINS EXECUTION.

Typed in small letters.

13) 'NG' WHERE 'N' IS A NONNEGATIVE INTEGER AND 'G' IS THE SMALL 'G'. IF N IS GREATER THAN THE CURRENT LINE NUMBER THEN ADVANCE CURRENT LINE TO N. OTHERWISE DO NOTHING.

14) 'V' WHERE 'V' IS THE SMALL 'V'. MEANS TYPE THE ENTIRE CURRENT LINE ON THE CONSOLE. THE PURPOSE IS TO VERIFY CURRENT CONTENTS OF A LINE.
ICIFIX OVERALL OPERATION

The simplest way to use units for ICIFIX, in order to keep unit usage for later programs in the 180-182 group is to put the input tape for ICIFIX ON UNIT 182 before running. This involves at most one removal and one mounting. The new tape will then be on 180 ready for immediate use by ICISCAN after the fix.

Due to the waste of computer time involved with ICIFIX use, only the following kinds of fixes should, in general, be done:

1) Date wrong over a period of several days (usually either a year or a month wrong).
2) Season changes not working.
3) Sections whose error causes a large number of subsequent ladder failures, say 20.

To save computer time, be sure to plan ahead, listing specific corrections, search chars, if any, and line numbers, and putting markers in the pages of the data listing where errors are to be corrected. Remember that ICIFIX corrections may be only temporary. All may be redone after corrections in correction text have been applied to the original unrevised backup file from ICI. About 2 corrections with ICIFIX per year of data sufficed quite well for batch No. 1.

It is not desirable to delete the correction text, just get around it with ICISCAN somehow. We may want to have correction text conveniently available associated with its revised data text at some point. Delete the correction text if necessary, though.

If you must mess with it you might as well delete it all.

You may want to print the corrections and stick this in the last full listing of data for reference.

When ICIFIX starts it displays the following message (the standard message) on the console...

N '17C'

Where '17C' is the line number of the current line (always line 1 at the start), and '17C' represents a character string consisting of the first 17 chars of line N.

This standard message is the response, except for errors to every command not resulting in a search. The standard message is also given after each displayed error message.

The other non-error message is the search message, which is identical to the standard message except that its first line is usually not 17 chars long, and there is a capital 'S' at the end of the message, in order to verify that it is indeed the successful search message. The last char of a successful search message is the search char itself. If a search is unsuccessful then the standard message is given, except in the event that the search goes to the end of the file, in which case no indication is given.

The standard sequence for correction by replacement is as follows...

1) Go to the line involved with an 'RNG' command.
2) If you have doubts, then verify the line with the 'Y' command, although this shouldn't be necessary since you should have the listing to refer to during the session.
3) Locate either at or shortly before the char(s) to be replaced. If the chars are near the beginning of the line then you are already located there since a replace command not immediately following a search-type command (immediately) will start replacing...
4) Replace the erroneous chars with new ones. If the chars are simply to be deleted then type the straight up and down char above the comma key. This is called the null char, or prefix, or EL/1 'OB' symbol. It will take up space in the input to ICISCAN, but ICISCAN will delete it very quickly, and it will not be part of the logical text.

5) You may want to verify the change.

ICIFIX error messages

ICIFIX has 4 standard error messages which are displayed on the 1052:

'NO COMMAND' the last response typed in did not have any non-blank chars in it.

'NO CENT SIGN' the response did not have a cent sign, and it was not one of the legal commands not containing a cent sign.

'BAD TYPIST'S ENTRY' an asterisk in column 1 of a line was followed by a capital letter, but either the letter was not followed by 1 or more decimal digits or else there were 1 or more digits but they were not followed by and immediately terminated by a blank. This message is displayed only in the 'check' mode. When the program begins execution it is in this mode.

'TYPIST'S ENTRY OUT OF ORDER X Y' the last entirely legitimate typist entry encountered by ICISCAN in check mode did not have a number 1 less than the value of the current legitimate page entry. The numbers, X and Y, are the previous and current page numbers respectively.

ICIFIX operates with STRINGRANGE and SUBSCRIPTRANGE enabled, and has 'PUT DATA' type SNAP ON CONDITIONS for these features.

/**
//DOCFRONT JOB 99999211, WCD, MSGLEVEL=(1,1)
//B EXEC LSPRINTS, OPT=CARDOCSN, DEM=DOCFRONT, DSSF=OLD
//PRINT.IN DD DNAME=INSYS
//PRINT.INSYS DD *

*** ICIFRONT and Slash ***

ICIFRONT is an appendage to the data entry system. ICIFRONT's main purpose is to perform corrections specified either by the CHINA data correction system specifications or the instructions for the London stage project. The differences between these instructions and those for card input are described here.

The main problem with cards is that keypunches with lower case capabilities are not available to us. To remedy this, the following rule is used...

Any character that is to be left unaltered is preceded by the slash character, when the slash is present then the character following it (even another slash) will be left exactly as it is, and the slash will be deleted. To get one single slash in the output use two consecutive slashes. All letters will be translated to lower case unless they are preceded by a slash. In this case the slash will be removed and the letter will not be altered.

This slash processing will be done by a program called 'slash'. Slash's input file is 'sysin'. The output file is 'crtct'. Slash has 1 option, option 1. If option 1 is specified then the output will be listed on sysprint in addition to going to the file CRTCT.

The SYSIN file is a stream file. The first 80 bytes must be binary digits, 0's except for possibly option no. 1. The 81st char must be a '*' starting the first correction statement. The data must be followed by at least 80 blanks (one card) at
THE END OF THE FILE. THERE IS NO PROCEDURE FOR SLASH.

ICIFRONT'S PROCEDURE IS CALLED LSFRONT. ITS DEFAULTS ARE.
DATUNIT=181, DATVOL=LSF025, DATSN=LSPRT, DATSEC=1,
CORUNIT=182, CORVOL=LSF026, CORSN=LSPCR, CORSEC=1,
OUTUNIT=180, OUTVOL=LSF027, OUTSN=LSCPL, OUTSEC=1

THE PARAMETERS USUALLY REQUIRED ON THE EXEC CARD ARE.
DATVOL, CORVOL, & OUTVOL.
IT IS SUGGESTED THAT, IF ICIFRONT IS TO BE USED, IT BE
TIED FIRST IN THE FIRST CYCLE AFTER THE INITIAL PROCESSING STAGE.
/*
//SCANLIST JOB 99999211, WCD, MSGLEVEL=(1,1)
//D EXEC LSPRINTS, OPT=CARDOCSN, MEM=DOCISCAN
//PRINT.IN DD DDDNAME=INSYS
//PRINT.INSSYS DD *

GENERAL OVERVIEW OF ICISCAN

ICISCAN'S GENERAL PURPOSE IS TWO-FOLD... 1) TO SERVE AS A FRONT
END FOR THE ODDITIES OF THE REMOTE EDITOR / CHINA DATA / OPTICAL
SCANNER DATA PREPARATION SYSTEM. 2) TO DIVIDE THE TEXT INTO ITS
3 KINDS OF COMPONENTS: STRUCTURED SECTIONS, EXTRANEOUS TEXT, AND
INDEX ENTRIES IN EXTRANEOUS TEXT. COMMENT SECTIONS ARE CONSIDERED
EXTRANEOUS TEXT, BUT WITH THE FRONT AND BEAR ENDLIMETERS REMOVED.
ICISCAN ALSO CHECKS FOR PAGE ENTRIES - BOTH DECIMAL AND ROMAN, BUT
THOSE ARE NOT SUPPOSED TO BE IN THE DATA AT PRESENT. THEY ARE
INTENDED FOR FUTURE USE IN THE LONDON STAGE INDEX FOR INSTRUCTIONS.
ICISCAN ALSO COLLECTS STATISTICAL DATA FOR GENERAL REFERENCE AND
SYSTEM OPTIMIZATION PURPOSES AS WELL AS FOR CALCULATING PAYMENTS TO
THE TYPING AND SCANNING COMPANIES. ICISCAN OPTIONALLY PRODUCES
EXTENSIVE PRINTOUTS OF PROCESSING TO FACILITATE LATE CORRECTION OF
INPUT TEXT BY A HUMAN.

ICISCAN HAS AN OPTION CHAIN IN COLUMNS 17-23. OPTIONS ON A CARD
ARE IN EFFECT UNTIL THE LINE WITH THE LINE NUMBERS IN THE CHAIN
IS READ IN. AT THAT TIME THE PROGRAM, REALIZING THAT NEW OPTIONS
ARE TO BE APPLIED, READS IN THE NEXT OPTION CARD FROM THE SYSSIN
FILE. THESE OPTIONS ARE THEN IN EFFECT AS THE LINE RECEIVES
PROCESSING IS CARRIED OUT. TO MAKE OPTIONS ON A CARD REMAIN IN
EFFECT UNTIL THE END OF THE PROGRAM, PUT 0'S IN COLS 17-23, SINCE
LINE 0 DOES NOT EXIST AND WILL THEREFORE NEVER BE READ IN.

IN ADDITION TO OPTIONS AND THE LINE NUMBERS FOR THE NEXT
SET OF OPTIONS, IF ANY, EACH ICISCAN OPTION CARD HAS 2 OTHER
THINGS ON IT... 1) A LIST OF THE SMALL LETTERS THAT ARE
ACCEPTABLE AS SECTION TYPE LETTERS, LOCATED IN COLS 24-49, AND
2) A LIST OF THE SECTION TYPES THAT ARE TO BE PUT CUTOFF BY ICISCAN,
IN COLS 51-76. BOTH LISTS OF SMALL LETTERS ARE REQUIRED TO
START IN THE FIRST COLUMN OF THEIR RESPECTIVE FIELDS AND TO
HAVE AT LEAST THE 'F' IN THEM. THE LIST OF LEGAL SECTION
TYPES, IN FACT, SHOULD ALWAYS BE THE SAME. AT PRESENT THIS MEANS
THE LETTERS, 'ABCDDEMOUSTU'.

AT THE HEAD OF THE OPTION CHAIN IS A SPECIAL OPTION CARD
CALLED THE STARTER CARD. IT IS MUCH LIKE OTHER OPTION CARDS,
BUT ITS PURPOSE IS ONLY TO START THE PROGRAM WITH THE RIGHT
INPUT FILE, CLIN OR IN. WHEN ICISCAN STARTS, IT FIRST READS
IN THE STARTER CARD AND PROCESSES IT JUST LIKE ANY OTHER OPTION
CARD. THIS HAPPENS BEFORE ANY OTHER INPUT. THE OPTION CARD
SHOULD HAVE A 1 IN COLUMN 15, THE APPROPRIATE OPTION FOR CLIN
OR IN IN COLUMN 7, AND 0000001 IN COLS 17-23. IT SHOULD ALSO
HAVE A NONBLANK CHAR IN COL 24, AND IN COL 51. IT MAY AS WELL
HAVE THE FULL SET OF SECTION LETTERS IN THESE LAST TWO FIELDS.

THE FOLLOWING OPTIONS ARE RECOMMENDED FOR A LIST/LOG OF
ICIFIXED SCANNED FORMAT INPUT... 3, 4, 11, 12, 16.

THE OPTION CARDS FOR THIS RUN WOULD BE
'00110000001100110000001ABCDDEMOUSTU' AND
'00110000001100110000001ABCDDEMOUSTU'

(THE LETTERS BEING SMALL LETTERS FONNE BY USE OF THE
MULTIPUNCH BUTTON).
GENERAL LOGIC OVERVIEW OF ICISCAN

THERE ARE 2 MUTUALLY EXCLUSIVE PRIME INPUT FILES, CLIN & IN.
IN'S INPUT IS IN THE FORM OF 80 BYTE LOGICAL RECORDS. EACH
INPUT BLK (OR LINE) IS NOMINALLY 75 BYTES OF TYPING FOLLOWED BY 5 BYTES
OF BLANKS. A NUMBER OF THINGS MAY AFFECT THIS, HOWEVER. THE
PROCEDURES APPLIED TO EACH INPUT LINE ARE DESCRIBED IN 'ORDER OF LINE
READIN PROCESSING'.
CLIN'S LINE IS OF VARIABLE LENGTH AND IS IN A STREAM FILE.
EACH LINE STARTS WITH A 20 BYTE HEADER OF CCMTEC INFORMATION.
THE FORMAT IS DOCUMENTED IN 'STANDARD DATA FILES'. THE LAST
LINE OF A FILE IS INDICATED BY A NEGATIVE LINE NUMBER WITHIN
CURRENT PAGE. THIS LAST LINE CONTAINS NO DATA, ALTHOUGH ITS
DATA PORTION MAY BE OF NONZERO LENGTH.
SINCE THE INPUT LINE IS OFTEN ONLY PART OF A SECTION, ICISCAN
CONSISTS ESSENTIALLY OF 2 'COROUTINES', THE INPUT COROUTINE AND THE
PROCESSING & OUTPUT COROUTINE. EXCEPT FOR EMPLOYING IN THE INPUT
ROUTINE, THE OUTPUT AND PROCESSING ROUTINE IS THE DOMINANT ONE,
INVOKING THE LINE READIN ROUTINE WHENEVER THERE ARE NO MORE DELIMITEBs
LEFT IN THE CURRENT BUFFER.
PURE FILE OUTPUT IS U-FORMAT BLKS UP TO 3625 BYTES IN LENGTH, EACH
BLK BEING ONE SECTION. AT PRESENT, INDEX ENTRIES ARE NOT PUT OUT,
ALTHOUGH THEY MAY BE OPTIONALLY PRINTED, SINCE THE FULL INFORMATION
NEEDED TO PROCESS THEM HAS NOT YET BEEN EDITED INTO THE TEXT.
CLOUT FILE OUTPUT IS WRITTEN AFTER ALL LINE READIN CHANGES HAVE
BEEN MADE TO THE INPUT LINE. IT IS POSSIBLE TO CREATE A
BACKUP OF A CLEAN FILE BY SPECIFYING OPTIONS 7 & 14 SIMULTANEOUSLY.
THE PROCESSING ROUTINE CONCERNS ITSELF PRIMARILY WITH MCODES AND
DELIMITERS. AT THE BEGINNING OF A RUN, TEXT STARTS IN STRUCTURED MODE,
AND ALL TEXT FROM THE BEGINNING OF THE DATA UNTIL THE FIRST DELIMITER
IS STRUCTURED TEXT. DEPENDING ON THE DELIMITER THE NEXT PIECE OF TEXT
(BETWEEN THE DELIMITER AND THE DELIMITER AFTER IT) WILL BE IN SOME
PARTICULAR MODE, FOR EXAMPLE 'STRUCTURED', OR 'EXTRANEOUS DUE TO LEFT
PARENT'. IN THE GENERAL CASE THERE ARE TWO FACTORS DETERMINING
A NEW MODE. 1) THE PREVIOUS MODE. 2) THE DELIMITER DEFINING THE START
OF THE NEW TEXT. AS A SPECIAL CASE, IF THE DELIMITER IS A "**" AND IT
IS IMMEDIATELY FOLLOWED IN THE CLEANED LOGICAL TEXT BY A SMALL 'C',
THEN THE NEW MODE WILL BE 'COMMENT SECTION'. THERE ARE 10 MCODES,
INDICATED BY VALUES RANGING FROM -5 TO +4 IN THE VARIABLE, 'MODE'.
THE 10 MODES ARE NAMED IN THE LISTING BESIDE THE ARRAY, 'NEWMCDE',
WHICH DEFINES THE NEW MODE OF FOLLOWING TEXT (EXCEPT FOR COMMENT
SECTIONS AS NOTED ABOVE) BASED UCFN THE PREVIOUS MCODE AND THE
DELIMITER ENDING THE PREVIOUS PIECE OF TEXT.

ORDER OF LINE READIN PROCESSING FOR ICISCAN
1) READ LINE. IF CLEANED INPUT (OPTION 7) THEN GO TO 4.
2) IF BLANK LINE THEN GO TO 1, NOT COUNTING IT AS A LINE.
3) COUNT IT AS A LINE.
4) IF LINE NUMBER IS SAME AS ON CURRENT OPTION CARD THEN READ IN
A NEW OPTION CARD AND PROCESS IT; DOING AN IMMEDIATE
STOP IF OPTION NO. 1 IS SPECIFIED.
5) IF OPTION 15, THE LINE SKIP OPTION, THEN GC TO 1.
6) IF OPTION 7, THE CLEANED INPUT OPTION THEN GC TO LINE 22.
7) COUNT AS A LINE ON CURRENT PAGE.
8) IF ILLEGAL CHAR CHECK OPTION NO. 11 THEN DO THE CHECK, CONVERTING
EACH ILLEGAL CHAR IN LINE TO A 'QUESTION MARK'.
9) IF OPTION 3 THEN LIST LINE AS IT PRESENTLY STANDS.
10) DELETE ANY NULL CHARS IN LINE, KEEPING CCOUNT.
11) CHECK FOR REGULAR TYPST ENTRY (VIA 'AT' SIGN-CAPITOL LETTER) &
PROCESS IT IF PRESENT. POSSIBLY REENTERING REGULAR TEXT MODE
FROM CORRECTION TEXT MODE.
12) GO TO 1 IF IN CORRECTION TEXT MODE.
13) TAKE CARE OF TRIPLE 'AT'S.
14) IF LINE STARTS WITH 2 'AT'S THEN CHANGE THEM TO 2 SPACES.
15) IF LINE STARTS WITH 1 'AT' THEN CHANGE IT TO A SPACE.
IF LINE STARTS WITH '***' THEN ENTER CORRECTION TEXT MODE AND
GO TO 1 AGAIN.

CONVERT PAIRS OF SINGLE QUOTES TO A DOUBLE QUOTE CHAR, KEEPING
COUNT AS NULL CHAR.

CHECK LAST 5 CHARS OF LINE FOR NONBLANK CHAR AND KEEP
ANY FOUND THERE.

CHOP OFF TRAILING BLANKS AND TYPIST ENTRY, IF ANY, FROM LINE.

PROCESS DOUBLE 'AT'S.

PROCESS SINGLE 'AT'S.

IF OPTION 14 THEN PUT OUT LINE TO FILE CLOUT.

COUNT UPPER CASE CHAR REMAINING IN LINE (FOR CLEANED CAP COUNT).

COUNT TOTAL CHAR REMAINING IN LINE (FOR TOTAL CLEANED CHAR COUNT).

APPEND TO WORK BUFFER.

NOTES ..

1) ILLEGAL SECTYPE CHARACTERS THAT ARE CAPITAL LETTERS ARE
   CHANGED TO SMALL UPPERCASE. THE RESULTING SECTIONS ARE THEN
   TREATED LIKE OTHER SECTIONS. IF THE SECTION IS OF A LEGAL
   TYPE THEN IT WILL BE PUT OUT.

2) AS NOTED IN LINE READIN PROCESS NO. 18, NONBLANK CHARACTERS IN
   FOLLOWING THESE NONBLANK CHARACTERS ARE NOT RETAINED.

STRUDOC JOB 9999211,WCD,MSLEVEL=(1,1)
/B EXEC LSPRINTS,OPT=CARDOCN,REM=STRUDOC
//PRINT.IN DD DDDNAME=INSYS
//PRINT.INSYS DD *

OVERVIEW OF STRUCTUR

STRUCTUR (PGM NAME= STRCT) TAKES OUTPUT FROM ICISCAN
(PURIFIED TEXT) AND SYNTACTICALLY ANALYZES IT INTO ITS
COMPONENT ITEMS AND GROUPS. EACH ITEM PUT OUT WILL BE EITHER
A ROLE, ACTOR, TIME ENTRY, LONDON STAGE PAGE ENTRY, SECTION
TITLE, THEATRE, DATE, CAST GROUP LADDER ENTRY, 'SEE' LADDER
ENTRY, OR TITLE LADDER ENTRY. THERE ARE TWO KINDS OF GROUPS,
TITLE GROUPS, AND CAST GROUPS. A TITLE GROUP CONTAINS THE
TITLE, IF ANY, OF A SECTION, AND THE TITLE LADDER ENTRY OR 'SEE'
LADDER ENTRY, IF ANY, OF A SECTION. IT MAY, LIKE CAST GEPs,
CONTAIN A LONDON STAGE PAGE ENTRY. A CAST GEP MAY CONTAIN
0 OR 1 TIME ENTRIES, 0 OR MORE ROLES, 0 OR 1 CAST GEP LADDER
ENTRIES, AND 0 OR MORE ACTORS IN THAT ORDER, EXCEPT THAT
ROLES MAY FOLLOW CAST GROUP LADDER ENTRIES AS LONG AS ONE
ROLE PRECEDES THE CGLG IN THE GROUP. THIS EXCEPTION MAY BE DUE
TO A BUG, BUT SHOULDN'T DO NO GREAT HARM. A CAST GEGG ALWAYS HAS
AT LEAST ONE ITEM, AND ALL ITEMS IN A CAST GEGG SHOULD BE
OF NONZERO LENGTH IN THEIR DATA RECTIONS. A TITLE ITEM MAY BE
OF ZERO LENGTH IN THE DATA PORTION, IN FACT EACH UNTITLED SECTION
HAS A TITLE OF ZERO LENGTH IN ITS OUTPUT BK.

EACH INPUT BK MUST CONTAIN 1 OR MORE COMPLETE SECTIONS.

EACH OUTPUT BK IS ONE SECTION. PARTS OF SECTIONS ARE NOT
ALLOWED, AND SHOULD NOT BE PRODUCED BY ICISCAN. EACH OUTPUT
BK OF SAVED, ON THE OTHER HAND, SHOULD CONSIST OF 1 OR MORE
COMPLETE SECTIONS, BUT OCCASIONALLY DUE TO BAD INPUT SAVED WILL
PUT OUT A SECTIONLESS BK OR ONE WITH PART OF A SECTION IN IT.

FOR SEQUENTIAL OUTPUT (OPTION 7) THE keys (12 BYTES) ARE
APPENDED TO THE FRONT OF EACH BK, AND 2 BLANKS ARE APPENDED IN
FRONT OF THE keys TO MAKE AN ABSOLUTE MINIMUM OUTPUT BLKSIZE
OF 18 BYTES INCLUDING THE MINIMUM POSSIBLE OF 4 DATA BYTES CONSISTING
OF AN EMPTY TITLE GEGG AND TITLE ITEM. THE BLANKS MAY BE
CONSIDERED 'RESERVED' SINCE AT PRESENT THEY ARE DISCARDED ON
INPUT BY LADDER.

STRUCTUR REQUIRES 3 SETS OF RECORDS (HEREAFTER CALLED 'CARDS')
IN ITS SYSTN FILE. THE FIRST 12 CARDS, CALLED MONST E CARDS,
DESCRIBE THE NAMES USED IN THE BATCH FOR THE MONTH PART OF A
LADDER REFERENCE. THE MONTH NAME MAY BE EITHER IN ALL CAPS
OR ELSE ENTERED EXACTLY AS IN THE DATA IN UPPER AND LOWER
CASE, DEPENDING ON WHETHER OPTION 9 IS SPECIFIED. EACH CARD
HAS 2 DATA FIELDS. THE FIRST FIELD IS COLUMN 1, AND CONTAINS
A NUMBER BETWEEN 1 AND 9 DEFINING THE LENGTH OF THE MONTH NAME.
THE NAME FIELD ITSELF STARTS IN COLUMN 2. THE REMAINDER OF THE CARD IS IGNORED. THE MONTH NAMES MUST BE IN ORDER FROM JANUARY TO DECEMBER, AND ALL MUST BE PRESENT FOR EVERY RUN OF STRCT.

IF OPTION 9 IS USED, THE MONTH NAMES MUST BE COMPLETELY UPPERCASE.

FOLLOWING THE MONSTR CARDS ARE 1 OR MORE SEASON CHANGE CARDS. THE LAST SEASON CHANGE CARD CONTAINS, STARTING IN COLUMN 1, THE FOLLOWING PATTERN OF 0'S AND X'S: '00000000000000'. THE REMAINDER OF THE CARD IS IGNORED. ALL OTHER SEASON CHANGE CARDS ARE IN THE FOLLOWING FORMAT, STARTING IN COLUMN 1:

YYYYMMDDTTTTTTTTL WHERE 'YYYYMMDD' IS THE DATE THE LAST PERFORMANCE TOOK PLACE.

'LL' IS THE NUMBER OF SECTIONS THAT ICISCAN PUT OUT FOR THE PERFORMANCE. IT IS EASY TO MAKE A BIG MISTAKE IN THESE CARDS BY 1) INCORRECTLY FIGURING (ESPECIALLY OVERESTIMATING) THE NUMBER OF SECTIONS PRESENT AT THE LAST PERFORMANCE BY FAILING TO NOTICE SOMETHING LIKE A DOUBLE 'AT', THAT DELETES THE ASTERISK STARTING THE SECTION (UNDERESTIMATING IS NOT SO SERIOUS). 2) FAILING TO ACCOUNT FOR THE KINDS OF SECTIONS THAT ARE NOT SELECTED FOR OUTPUT BY ICISCAN, THAT IS, IF THE 'SECTION TYPES WANTED' FIELD OF ICISCAN'S OPTION CARD CONTAINS ONLY 'AP', THEN 'L' SECTIONS WILL NOT BE PUT OUT BY ICISCAN OR COUNTED BY STRCT. 'TTTTTTTT' IS THE THEATRE, TYPED IN LOWER CASE AND PADDED ON THE RIGHT WITH BLANKS. THE REMAINDER OF EACH CARD FOLLOWING THE 'LL' FIELD IS IGNORED.

FOLLOWING THE LAST SEASON CHANGE CARD IS THE OPTION CHAIN, WHICH CONSISTS OF 1 OR MORE CARDS. THE OPTION CARD HAS 32 OPTION LOCATIONS IN COLS 1-32. IN COLS 33-38 IS THE LINK TO THE INPUT BLK, IF ANY, FOR WHICH THE NEXT OPTION CARD WILL BE USED. IF THERE IS NO NEXT SET OF OPTIONS THEN 0'S ARE PREPARED IN COLS 33-38. THE OPTIONS ON THE NEXT CARD ARE READ IMMEDIATELY AFTER THE READING OF THE BLK WHOSE NUMBER IS LISTED IN THE CURRENT OPTION CARD. IN PARTICULAR, THE BLK NUMBERED IN THE CURRENT OPTION CARD WILL BE SKIPPED IF THE NEXT OPTION CARD SPECIFIES THE SKIP OPTION. THE VERY FIRST OPTION CARD IN THE CHAIN IS READ IN AND TAKES EFFECT RIGHT AFTER THE FIRST BLK OF INPUT FROM THE FILE, PURE.

RECOMMENDED OPTIONS FOR ERROR CHECKING ARE: 5, 6, 7, 9, 27.

IN BATCH 1, STRCT PRODUCED RELATIVELY FEW ERROR MESSAGES, BUT IF ERROR MESSAGES ARE FREQUENT IN SOME OTHER BATCH THEN OPTION 27 SHOULD BE THE FIRST TO GO.

PROGRAM NOTES

M IS A SCRATCH VARIABLE USED ONLY IN SHORT SEQUENCES.
IX IS A SCRATCH VARIABLE USED IN NO INNER PROCEDURE OR ON UNIT.
K AND L ARE SCRATCH VARIABLES, THEY ARE USED IN STRUCT, GTCK, AND GITM, BUT NOT IN COGO OR ERRMSG OR ANY ON UNIT.

** SPECIFICATIONS FOR THE STRU/LATA FILES **

THE FOLLOWING 9 TYPES OF GROUPS OCCUR IN THE STRU FILE:

A. AFTERPEICE TITLE GROUP, ALWAYS THE FIRST IN ITS SECTION
B. PERFORMANCE TITLE GROUP, ALWAYS THE FIRST GROUP IN ITS SECTION
C. INSTRUMENTAL TITLE, ALWAYS 1ST GEE IN SECTION.
D. OPERA TITLE, ALWAYS 1ST GEE IN SECTION.
E. MONOLUGE WITH PARTS TITLE, ALWAYS 1ST GEE IN ITS SECTION.
F. BALLET TITLE, ALWAYS 1ST GRP IN ITS SECTION.
G. 'ROLE-ACTOR GROUP', WITH TIME, ROLE, AND ACTOR ITEMS
H. LADDER GROUP, WITH AN L OR S ITEM.

ITEMS ARE TAGGED BY THE FOLLOWING CHARACTERS:

B = PAGE ENTRY
A = ACTOR (ALSO MUSICIAN, DANCER, SINGER ETC.)
R = ROLE (ALSO SONG, ENTERTAINMENT, DANCE ETC.)
T = TIME ENTRY
A = (SMALL LETTER) AFTERPEICE TITLE
P = (SMALL LETTER) PLAY TITLE (MAINEPEICE)
U = (SMALL LETTER) TITLE OF MONOLUGE WITH PARTS
B = (SMALL LETTER) BALLET TITLE
ALL LADA FILE BLKS HAVE A TITLE OF AT LEAST THE NULL VARIETY. THE TITLE 'UN' IS TREATED JUST LIKE ANY OTHER TITLE. PERHAPS STRUCTR SHOULD CONVERT 'UN' TITLES TO ' ' TITLES.

STRU/STRS FILES DO NOT HAVE TITLE GRPS EXCEPT FOR TITLED SECTION.

THE TIME ENTRY, IF PRESENT, MUST PRECEDE ALL ROLE AND ACTOR ENTRIES IN THE GROUP.

ALL ROLE ENTRIES PRECEDE ANY ACTOR ENTRIES.

THE TITLE, IF ANY, OF AN AFTERPICE, SONG, OR PERFORMANCE SECTION MUST BE THE FIRST GROUP IN ITS SECTION. THE TITLE ITEM MAY BE PRECEDED BY A PAGE ENTRY, HOWEVER, SINCE PAGE ENTRIES MAY OCCUR ANYWHERE. A LADDER REFERENCE MAY NOT PRECEDE A PLAY, OPERA, OR AFTERPICE TITLE. A CGLE MUST BE PRECEDED BY AT LEAST ONE ROLE.

STRU FILE NOTES . . 9/12/71

1) EVERY TITLED SECTION HAS AT LEAST FC CAT SMCD AS 1ST 2 CHAR.
2) IF A SECTION HAS A TITLE THEN IT IS IN ITEM FORM, & THIS MAY BE OF LENGTH 0.
3) EVERY TITLED SECTION HAS AT LEAST A 0 LENGTH TITLE, (IF THERE ARE ANY CANDIDATES AT ALL AFTER THE THEATRE).
4) 'P GF' TYPES OF SECTIONS HAVE NO TITLE AT ALL, BUT THEATRE IS PRESERVED IN KEY.
5) AN RA GRP WITH NO ITEMS IS POSSIBLE DUE TO INPUT DATA ERROR, ELSE NOT.
6) NULL TIME ENTRIES CANNOT EXIST. THEY ALWAYS HAVE AT LEAST THE COLON.
7) NULL ROLE OR ACTORS (0 LENGTH) EXIST ONLY IF GTK DELIVERS A 0 LENGTH C TO THE ROLE OR ACTOR LOOP.
   (GTM ALWAYS DELIVERS AT LEAST 'ERRONEOUS ITEM'.)
8) LADDER REFERENCES HAVE ONLY 'S OR 'L OR 'C AS THEIR ITEM TYPE LETTERS, AND CAN OCCUR ONLY WHERE A ROLE OR AN ACTOR IS EXPECTED, THAT IS IN G GROUPS.
9) PAGE ITEMS ARE FIXED LENGTH (4 DATA, 6 INC ITEM CTRL) IF PRESENT.

FORMAT OF SECTION/BLK KEYS FOR STRUCTURE/LADDER/ITEMGET . .

RECORDED KEY LENGTH IS 12 BYTES.
BYTES 0 TO 2 ARE THE CODED DATE AND DATE SIGNIFICANCE BITS.
THE RIGHTMOST 3 BITS DEFINE WHETHER THE CORRESPONDING PART OF THE DATE (YEAR, MONTH, AND DAY RESPECTIVELY) IS DEFINITE OR UNCERTAIN. A ZERO BIT MEANS DEFINITE, ONE MEANS UNCERTAIN. THE BITS CORRESPOND TO YEAR, MONTH, AND DAY RESPECTIVELY LEFT TO RIGHT.
THE ACTUAL DATE REPRESENTATION IS ENCODED BY THE FORMULA,
IDATE= ((IYEAR-1660) * 3724 (IMONTH-1) * 31 + IDAY-1) * 8 .

BINARY NULLWORD, IDATE. AFTER THE SIGNIFICANCE BITS ARE ADDED TO IDATE, THE LAST 3 BYTES OF IDATE ARE CONCATENATED TO THE REST OF THE KEY. THE FULL RECORDED KEY IS WRITTEN FROM THE VARIABLE KEY.

BYTE 3 CONTAINS THE SECTION TYPE LETTER (SMCD) . .

A) AFTERPICE SECTION
B) BALLET SECTION
D) DANCING SECTION
E) ENTERTAINMENT SECTION
I) INSTRUMENTAL MUSIC SECTION
M) MUSIC SECTION
O) OPERA SECTION
P) PERFORMANCE SECTION
S) SINGING SECTION
T) TRICK SECTION
U) MONOLOGUE WITH PARTS SECTION

BYTES 4 TO 11 CONTAIN THE THEATRE ABBREVIATION, WHICH IS RIGHT-ADJUSTED AND PADDED ON THE RIGHT WITH BLANKS, IF NECESSARY.

/ * 
//LADRDAC DOC 99999211, WCD, MSGLEVEL=(1,1)
LADDER OVERVIEW

LADDER HAS THE TWO FUNCTIONS OF RETRIEVING LADDER REFERENCES AND OF PERFORMING COMPLEX UPDATES WITH THESE LADDER REFERENCES. THERE ARE 3 KINDS OF LADDER REFERENCES DISCUSSED SEPARATELY BELOW.


AFTER ALL (ALL) INPUT/SETUP IS COMPLETE THE UPDATE STAGE STARTS. IN THIS STAGE LADDRE GOES THROUGH THE INPUT CHAIN, AND HANDLES EACH GRP IN IT ON AN INDIVIDUAL BASIS. FOR CORRECT TLE CASES AND NON-LADDER CASES PROCESSING IS AS DESCRIBED IN THE SPECIFICATIONS, 'TYPING INSTRUCTIONS FOR THE INDCN STAGE PROJECT', 'DESCRIPTION OF THE CALENDER', 5 MAY OCBF II. THIS IS CARRIED OUT BY RECHAINING GRPS AND ITEMS FROM THE INPUT CHAIN TO THE OUTPUT CHAIN, AND DELETING GRPS AND ITEMS FROM THE OUTPUT CHAIN. TLE PROCESSING OR SLE PROCESSING IS CARRIED ON SIMULTANEOUSLY WITH CGLE PROCESSING ON A GEF BY GEF BASIS. ALTHOUGH THE UPDATE STAGE HAS NOT BEEN THOROUGHLY TESTED IT SHOULD WORK FOR CHINA DATA MATERIAL SINCE THIS PRESUMABLY DOES NOT HAVE ANY UPDATES EXCEPT THE REPLACE VARIETY WHICH HAS BEEN THOROUGHLY TESTED. ONE PROBLEM WHOSE SOLUTION HAS NOT BEEN UNDERTAKEN IS THAT OF REKEYING IN THE OUTPUT CHAIN.

USING THE FACT THAT NAMES ARE SUPPOSED TO BE UNIQUE IN ANY ONE CONTEXT, IN THIS CASE A SECTION WITH ALL ITS LADDER REFERENCES, SIMPLIFIES THE PROGRAMMING QUITE A BIT. THE SIMPLIFICATION LIES IN THAT UPDATING AN ITEM INVOLVES ONLY ALTERNING, IN SOME WAY, THE OUTPUT CHAIN. THE REFERREING ITEM IN THE INPUT CHAIN IS THEREAFTER 'UNREFERENCEABLE', BUT THE ITEM IN THE OUTPUT CHAIN CAN STILL BE AFFECTED BY A LATER ITEM IN THE INPUT CHAIN. FOR EXAMPLE IF THERE IS AN ACTOR, JOE, AT THE FAR END OF THE OUTPUT CHAIN, AND THERE ARE 2 REFERENCES TO IN THE INPUT, THE FIRST ONE BEING AN ADDITION OF SOME SORT, AND THE SECOND BEING A DELETION, THEN THE ORIGINAL JOE AT THE FAR END OF THE OUTPUT CHAIN WOULD SURVIVE AND THE JOE JUST ADDED WOULD BE DELETED. DELETIONS SHOULD PRECEDE ADDITIONS, HOWEVER EVEN THIS WILL NOT NECESSARILY PREVENT ALL TRAGEDIES.

'SEE' LADDER REFERENCES ARE PROCESSED MUCH LIKE TLE'S IN INPUT/SETUP. IN THE UPDATE STAGE, HOWEVER, THE PROCESSING IS AS DESCRIBED IN THE JUNE 9 GENERAL SPECIFICATIONS. THIS HAPPENS ON AN INPUT GRP BY INPUT GRP BASIS, AND PRECEDES REGULAR PROCESSING AND RECHAINING TO THE OUTPUT CHAIN FOR THE GEF. A CGLE GRP WILL HAVE SEE TYPE PROCESSING DONE ON IT WITH RESPECT TO THE REFERRED-TO GRP BEFORE OTHER PROCESSING IS DONE. THIS BUG SHOULD NOT CAUSE MUCH TROUBLE, ESPECIALLY AT PRESENT, SINCE THE MATERIAL APPEARS TO HAVE SO FEW 'SEE' REFERENCES (ONLY 1 IN BACH 1).

TIME ENTRIES ARE NOT HANDLED IN 'SEE' REFERENCES. THEY ARE TAKEN FROM THE CURRENT TIME ENTRY, IF ANY, REGARDLESS.

IN THE OUTPUT FORMATTING STAGE ACTORS THAT WERE READ IN BY A 'SEE' LADDER REFERENCE AND NOT DELETED ARE FORMATTED CUT WITH A QUESTION MARK FOLLOWING THEM. AT PRESENT THIS IS TRUE OF CGLE ACTORS AS WELL. THIS IS DONE IN ORDER TO LOCATE 'CGLE'S MORE EASILY ON THE PRINTOUT SINCE CGLE'S HAVE NOT BEEN TESTED THOROUGHLY, AND
SUCCESSFUL ONES ARE SCARCE ON THE ERRCH PRENTCUTS.
AFTER OUTPUT CHAIN SETUP, ANY CGLE'S LEFT ARE PUT THROUGH
INPUT/SETUP PROCESSING AND CHAINED TO THE CAST GGP'S THAT REFERRED
TO THEM. THE LINK IS IN THE REFERRED GGP'S GCB IN THE GCG
ARRAY, AND POINTS TO THE REFERRED-TO GGP'S GCE.
IN INPUT/SETUP REFERRED-TO CGLE GGP'S ARE READ IN AND APPENDED TO
THE S BUFFER AS THEIR REFERRING ITEMS ARE ENCOUNTERED. AFTER
THIS THE CGS ELEMENT OF THE REFERRED GGP'S GCE IS SET TO
CONTAIN THE LAST LGG OF THE REFERRED-TO GGP IN THE S BUFFER.
AFTER THE TLE OR SLR SETUP STAGE, IF ANY, EACH GCE IN THE INPUT
CHAIN IS EXAMINED TO SEE IF ITS CGS ELEMENT POINTS TO A REFERRED-TO
CAST GGP IN THE S BUFFER. IF SO, THIS REFERRED-TO GGP IS ALSO
SET UP, WITH THE CGS PTR IN THE REFERRED GGP NOW RESET TO PCINT
TO THE REFERRED-TO GGP'S GCB.
WHEN UPDATE PROCESSING IS DONE ON AN INPUT CHAIN GGP, IT IS
AGAIN CHECKED TO SEE IF IT HAS A CGLE. IF SO, THEN THE REFERRED-TO
GGP IS LOGICALLY SUBSTITUTED FOR THE OUTPUT CHAIN TEMPORARILY,
AND THEN NORMAL UPDATE PROCESSING IS DONE ON IT. AFTER THE GGP
IS PROCESSED, THE RESULT IS CHAINED TO THE ACTUAL OUTPUT CHAIN.
IF BLKSIZE PROBLEMS OCCUR FOR LADA IT WILL BE NECESSARY TO
CHANGE BOTH THE DBC ON THE DD CARD AND THE S & 2 BUFFERS IN LADD.
THIS, AS FAR AS I CAN SEE, IS AS SIMPLE AS IT LOOKS, EXCEPT THAT
CORE PROBLEMS MAY ARISE. THIS IS NEITHER A HIGH NOR A LOW PROBABILITY.
LADD'S OPTIONS ARE IN COLS 1-32, AND THE OPTION CHAIN IS IN
COLS 33-38. THE REMAINDER OF EACH CARD IS IGNORED.
RECOMMENDED OPTIONS FOR A FIRST RUN ARE 2, 6, 7, AND 27.

GENERAL DESCRIPTION OF FILE USAGE

<table>
<thead>
<tr>
<th>DDNAME</th>
<th>USE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSin</td>
<td>ALWAYS USED FOR OPTION CHAIN.</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>ALWAYS USED FOR 'PRINT FILE'.</td>
</tr>
<tr>
<td>LADA</td>
<td>ALWAYS USED, INTERMEDIATE STORAGE, DA KEYED OUTPUT.</td>
</tr>
<tr>
<td>STRU</td>
<td>USED FOR DA KEYED INPUT WHEN OPTION 6 IS 0.</td>
</tr>
<tr>
<td>STS</td>
<td>USED FOR SEQL UNKEYED INPUT WHEN OPTION 6 IS 1.</td>
</tr>
<tr>
<td>LADS</td>
<td>USED ONLY FOR SEQL UNKEYED OUTPUT WHEN OPTION 7 IS 1.</td>
</tr>
</tbody>
</table>

LADA IS USED FOR INTERMEDIATE STORAGE OF A SEASON'S
PERFORMANCES, AND, IF OPTION 2 IS NOT SPECIFIED AND THE OUTPUT
WANTED IS ONLY IN THE LAST SEASON PROCESSED, THEN LADA MAY ALSO
SERVE AS THE OUTPUT FILE. THIS WOULD NECESSITATE CHANGING THE
DISP PARAMETER TO 'KEEP' FOR THE LADA DD STATEMENT.
STRU IS THE MAIN (STRUCTURED DATA) INPUT FILE IF OPTION 6
IS NOT SPECIFIED, OTHERWISE STRS IS. IF USE OF STRU, LADS, OR
STRS IS NOT SPECIFIED ON ANY OPTION CARD THEN ITS DD CARD NEED
NOT BE PRESENT IN THE JCL DECK.
LADD'S DATA OUTPUT IS IN THE SAME FORMAT AS THE STRU
OR STRS FILE INPUT EXCEPT THAT LADDER ENTRIES AND PAGE ENTRIES
ARE NO LONGER PRESENT. IN THE LADA FILE THE BLKS ARE LIKE
THE KEYED STRU FILE BLKS. IN THE LADS FILE BLKS ARE UNKEYED,
WITH THE KEY AND 2 BLANKS APPENDED TO THE FRONT OF THE BLK JUST
AS WITH THE STRS FILE INPUT.

INTERNAL PROGRAMMING NOTES

FOR LADDER K AND L ARE SCRATCH VARIABLES USED ONLY IN SHORT
SEQUENCES OF INSTRUCTIONS.
IGP1 = 0 IF NO PRE-LADDER GRP'S, =1 IF THERE IS AT LEAST ONE PRE-LADDER
GROUP.
IGP2 PTS TO LAST PRE-LADDER GGP, =0 IF NO PRE-LADDER GRP'S.
IGP3 PTS TO 1ST LADDER GGP, =0 IF NO LADDER GRP'S.
IGP4 PTS TO LAST LADDER GGP, =0 ONLY IF NO RA GRP'S AT ALL EXIST.

THE GCB (GROUP CONTROL BLOCK) LINK FORWARD VARIABLE IS GF.
THE 'ICB' (ITEM CONTROL BLOCK) LINK FORWARD FIELD IS ILK.
THE LENGTH OF EACH ACTOR OR ROLE CHAIN IS KEPT IN THE CHAIN'S GCB.
THE KEYS SUBROUTINE SEARCHES THE OUTPUT CHAIN FOR AN ITEM CORRESPONDING TO ONE DESCRIBED IN THE CALL. THE CALLING SEQUENCE IS:

1ST PARAMETER  GRP IN WHICH KEY RESIDES (GCB NUMBER)
2ND PARAMETER  OFFSET IN THE SUBGROUP OF THIS GROUP.
0 = 1ST ON CHAIN AS DOES 1.

AN IMPLICIT PARAMETER IS THE ARRAY, SBITS, DEFINED ON SEITE. IN THIS ARRAY IF THE CORRESPONDING BIT IS A 1 THEN KEYING IS ATTEMPTED ON THAT PARTICULAR KIND OF ITEM. IF MORE THAN ONE OF THE BITS IS 1 THEN THE ORDER OF SEARCHING IS... ROLE THEN ACTOR, THEN TIME (SECOND BIT, THIRD BIT, THEN FIRST BIT).

THE VALUES RETURNED ARE IN IRG, IROP, AND PEETS RESPECTIVELY. IRG IS 0 IF THE SEARCH WAS UNSUCCESSFUL. OTHERWISE IRG IS THE NUMBER OF THE GCB FOR THE GRP WHICH CONTAINS THE KEY, AND IROP IS THE ITEM CONTROL BLOCK NUMBER (NOT OFFSET WITHIN SUEGEB, AS IN CALLING SEQUENCE, BUT ABSOLUTE). PEETS INDICATES WHICH KIND OF ITEM THE KEYING WAS ACHIEVED ON, BEING INTERPRETED THE SAME AS SBITS.

IN ADDITION, IGL INDICATES THE PREVIOUS GRP ON THE LADDER CHAIN IF ANY (=0 IF 1ST GRP), AND IIL INDICATES THE ABSOLUTE ITEM CONTROL BLOCK OF THE PRECEDING ITEM IN THE SUBGROUP, IF ANY (=0 IF NONE).

OTHER MORE EXTENSIVE, BUT LESS RELIABLE NOTES CAN BE FOUND IN THE FILING CABINET IN THE LAWRENCE ROOM.

/*ITEMHF  JOB 99999211,WCD,MSGLIVE=(1,1)
/D EXEC LSPRINTS,OPT=CARDOCSN,REM=DOCITEMS
/PRINT IN DD DDNAME=INSYS
/PRINT.INSYS DD */

CONTROL OF THE SELECTION PROCESS AT ANY ONE POINT IN THE RUN LIES IN THE SELECTION CONTROL BLOCK (SCB), WHICH CONSISTS OF ONE DATE RANGE ENTRY AND 0 OR MORE SELECTION STATEMENT ENTRIES. A SELECTION STATEMENT ENTRIES CONTAINS THE FOLLOWING ITEMS OF INFORMATION:

A) INCLUSIVENESS (A DIGIT FROM 1 TO 4)
B) SECTION TYPE S
C) THEATRE M
D) TIME C
E) TITLE T
F) SYNTACTIC ROLE R
G) SYNTACTIC ACTOR A

EACH ITEM, EXCEPT INCLUSIVENESS IS A VARYING LENGTH CHARACTER STRING. MATCHING OF A SORT RECORD WITH ANY SELECTION ENTRY IN THE PROPER DATE RANGE CAUSES THE RECORD AND POSSIBLY THE SORT RECORDS OF THE GROUP, SECTION, OR PERFORMANCE IN WHICH IT IS FOUND TO BE PUT INTO THE OUTPUT FILE. WHETHER ONLY THE MATCHING RECORD ITSELF OR THE ENTIRE GROUP, SECTION, OR PERFORMANCE IS PUT OUT DEPENDS ON THE INCLUSIVENESS ITEM OF THE SELECTION ENTRY ON WHICH THE MATCH WAS MADE.

1=RECORD, 2=GROUP, 3=SECTION, 4=PERFORMANCE.

THE LETTER FOLLOWING EACH TYPE OF ITEM IN THE ABOVE TABLE IS THE ITEM TAG USED IN INPUT CONTROL STATEMENTS TO SPECIFY THE TYPE OF EACH ITEM.

SELECTION BY A SELECTION ENTRY IS DETERMINED IN THE FOLLOWING WAY:
FIRST THE DATE OF THE SORT RECORD MUST BE IN THE DATE RANGE OF THE SCB. SECOND, EACH ITEM IN THE SELECTION ENTRY OTHER THAN INCLUSIVENESS, THAT HAS A NONZERO LENGTH MUST MATCH THE CORRESPONDING ITEM IN THE SORT RECORD.

THE SELECTION ENTRIES WILL BE DEFINED BY CONTROL RECORDS IN THE FORM OF SELECTION CONTROL STATEMENTS IN THE FILE SYSTIN. SINCE IT IS EXPECTED THAT THIS FILE WILL CONSIST OF CARDS, EACH LETTER READ IN THE DATA PORTION OF A SELECTION ENTRY ITEM WILL BE TRANSLATED TO A LOWER CASE LETTER IF IT IS ENTERED IN UPPER CASE UNLESS THE LETTER IS PRECEDED BY A SLASH, IN WHICH CASE THE SLASH WILL BE LEFT OUT AND THE CHARACTER FOLLOWING IT WILL NOT BE TRANSLATED. A CHARACTER IMMEDIATELY FOLLOWING AN INITIAL SLASH, INCLUDING ANOTHER SLASH OR A BLANK, IS TAKEN AS IS AND
A SELECTION STATEMENT IS ONE THAT HAS AN INCLUSIVENESS TAG OF 1, 2, 3, 4, OR 6. AN ORDINARY SELECTION STATEMENT IS ONE THAT HAS AN INCLUSIVENESS TAG OF 1, 2, 3, OR 4.

EACH STATEMENT CONSISTS OF THE INCLUSIVENESS ITEM PLUS 0 OR MORE OTHER ITEMS, THE STATEMENT ENDING WITH 2 CONSECUTIVE BLANKS. EACH ITEM WILL START WITH A CHARACTER INDICATING WHAT SORT OF ITEM IT IS, TITLE, THEATRE, ROLE, ETC., FOLLOWED BY THE ITEM ITSELF, AND, EXCEPT FOR THE LAST ITEM IN A STATEMENT, ENDING WITH THE 'E' SIGN. THE LAST ITEM WILL BE DELIMITED BY THE 2 CONSECUTIVE BLANKS THAT END THE STATEMENT. AN ENTIRE GROUP OF STATEMENTS THAT FORM AN SCE WILL BE DELIMITED BY A STATEMENT WITH AN INCLUSIVENESS TAG OF 7.

FOLLOWING THIS STATEMENT PROCESSING OF SCE RECORDS WILL PROCEED UNTIL A DATE GREATER THAN THE 2ND DATE IN THE DATE RANGE ENTRY OF THE SCE IS ENCOUNTERED. THIS SELECTION SPECIFICATIONS SHOULD BE SET UP WITH DATE RANGES IN CHRONOLOGICAL ORDER.

EACH ORDINARY SELECTION STATEMENT CORRESPONDS TO ONE SCE ENTRY. EACH REPEAT STATEMENT CORRESPONDS TO AS MANY SELECTION ENTRIES AS ARE DEFINED BY THE REPETITIVE SPECIFICATIONS.

AN ORDINARY SE WILL BE FORMED FROM THE ITEMS IN THE SELECTION STATEMENT AS WELL AS THE LAST ITEM OF EACH OTHER TYPE USED IN A PREVIOUS SELECTION STATEMENT UNLESS SPECIFIED OTHERWISE. AN EXAMPLE SHOULD ILLUSTRATE THIS:

5 1738 05 23 1738 06 14
1MILFET/HAMLET, /PRINCE OF /DENMARK
4MCG

THE STATEMENT ON THE FIRST LINE, LIKE ALL STATEMENTS, STARTS WITH THE STATEMENT TAG, IN THIS CASE 5, WHICH INDICATES THAT IT IS A DATE RANGE STATEMENT. THE DATE RANGE ITSELF IS IN A FIXED LENGTH FORMAT AND HENCE NEEDS NO DELIMITERS AT IT ITS END.


AS A RESULT OF THE FIRST 2 STATEMENTS ANY SCE RECORD WITH A THEATRE ENTRY OF 'DL' AND A TITLE ENTRY OF '/HAMLET, /PRINCE OF /DENMARK' BETWEEN THE DATES OF MAY 23 1738 AND JUNE 14 1738 WILL BE PUT OUT. IN ADDITION THE STATEMENT INTERPRETER HAS REMEMBERED THE TWO ITEMS, THEATRE, AND TITLE, IN CASE OF LATER NEED. THIS USE IS ILLUSTRATED BY THE SELECTION ENTRY RESULTING FROM THE THIRD STATEMENT. THIS STATEMENT HAS ONLY ONE ITEM, A 'CG' THEATRE ENTRY, BUT SINCE IT HAS NOT BEEN TOLD OTHERWISE, THE STATEMENT INTERPRETER INTERPRETS THIS TO MEAN THAT THE OTHER KINDS OF ITEMS THAT HAVE BEEN DEFINED PREVIOUSLY, IN THIS CASE ONLY THE TITLE, '/HAMLET, /PRINCE OF /DENMARK', ARE TO REMAIN THE SAME.

THAT IS, THE ENTRY CAUSES NOT ONLY MATCHING ON 'CG' BUT ON '/HAMLET, /PRINCE OF /DENMARK' AS WELL, EVEN THOUGH THE THIRD STATEMENT DOES NOT MENTION '/HAMLET, /PRINCE OF /DENMARK'. THE NUMBER 4, THE INCLUSIVENESS INDICATOR FOR THE THIRD STATEMENT INDICATES THAT IF A MATCH IS FOUND FOR THIS SELECTION ENTRY THEN THE ENTIRE SET OF SCE RECORDS RESULTING FROM THAT PERFORMANCE ARE TO BE PUT OUT. 3 WOULD MEAN THE ANALOGOUS THING FOR THE SECTION, AND 2 FOR THE GROUP. THE WAY TO 'CLEAR' THE INTERPRETER'S MEMORY OF A PREVIOUSLY DEFINED ITEM IS TO REDEFINE IT (THE ITEM) AS A ZERO LENGTH CHARACTER STRING, FOR EXAMPLE:

IT WOULD CLEAR THE TITLE AS THOUGH IT HAD NOT BEEN MENTIONED PREVIOUSLY.

IT SHOULD BE REMEMBERED, HOWEVER, THAT THE AECVE STATEMENT WOULD
RESULT IN A SELECTION ENTRY. CLEARING WORKS JUST AS WELL IN A STATEMENT THAT HAS OTHER ITEMS IN IT.

ITEM MAY

WOULD CLEAR THE TITLE, AND PROVIDE FOR SELECTION OF SECTIONS OCCURRING AT 'HAY' ON THE SAME BASIS AS THE PREVIOUS DEFAULTS EXCEPT THAT ANY TITLE WOULD BE ACCEPTABLE.

A STATEMENT BEGINNING WITH AN INCLUSIVENESS TAG OF '6' IS A REPEAT STATEMENT. THE REPEAT STATEMENT MAY CONTAIN ANYTHING THAT AN ORDINARY SELECTION ENTRY STATEMENT HAS, BUT IN ADDITION IT CONTAINS, IMMEDIATELY FOLLOWING THE INCLUSIVENESS TAG, 2 NUMBERS THAT FORM ITS REPETITIVE SPECIFICATION. THESE NUMBERS DEFINE RESPECTIVELY THE FIRST AND LAST PREVIOUSLY DEFINED SELECTION ENTRIES THAT ARE TO BE USED IN THE REPETITION. THESE DEFINED SELECTION ENTRIES ARE REPEATED EXCEPT FOR CHANGESDEFINED IN THE USUAL WAY IN THE REMAINDER OF THE REPETITIVE STATEMENT. EXAMPLE, SUPPOSE THIS WERE THE 4TH STATEMENT.

6 1 2 A JONES R HAMLET


THE INCLUSIVENESS TAG OF EACH SELECTION ENTRY THUS FORMED WILL BE THE SAME AS THAT OF THE ORIGINAL ENTRY UPON WHICH IT WAS BASED.

THE MEMORY OF THE STATEMENT INTERPRETER IS CLEARED BY A REPEAT STATEMENT EXCEPT FOR THE ITEMS USED IN THE STATEMENT.

THE CLEARING IS DONE BEFORE THE ITEMS IN THE STATEMENT ARE PROCESSED, THUS NO DEFAULTS ARE USED.

REPEAT STATEMENTS AND DATE RANGE STATEMENTS MUST BEGIN IN COLUMN 1 OF A CARD OR THE EQUIVALENT.

ITEMGET REFERENCE SHEET

INCLUSIVENESS AND STATEMENT TYPE NUMBERS..

0 CLEAR ALL DEFAULTS FOR SCB
1 SELECT RECORDS
2 SELECT GROUPS
3 SELECT SECTIONS
4 SELECT PERFORMANCES
5 SET UP DATE RANGE STATEMENT
6 START OF REPEAT TYPE STATEMENT
7 INITIATE SELECTION STAGE, END OF SELECTION SETUP STAGE.
8 CLEAR ALL DEFAULTS AND SCB, EXCEPT FOR DATE RANGE.
9 STOP EXECUTION IMMEDIATELY.

ITEM TYPE CHARACTERS..
S SECTION LETTER
M THEATRE (MOVIE)
C TIME ENTRY (CLOCK)
T TITLE
R ROLE
A ACTOR

ITEMS ARE DELIMITED BY THE 'E' IF FOLLOWED BY ANOTHER ITEM IN THE SAME SE, OTHERWISE THEY ARE DELIMITED BY THE DOUBLE BLANK THAT DELIMITS THE END OF THE STATEMENT. STATEMENTS ARE DELIMITED EXCLUSIVELY BY THE DOUBLE BLANK. SCB'S ARE DELIMITED BY A TYPE 7 STATEMENT.

FORMAT FOR A REPEAT SPECIFICATION IS..

COL 1 '6'
COL 2 IGNORED
COL 3-6 FIRST SE, RIGHT JUSTIFIED
COL 7-10 2ND SE, RIGHT JUSTIFIED
COL 11-13 IGNORED
COL 14 ITEM TYPE CHAR FOR 1ST ITEM

FORMAT FOR DATE RANGE STATEMENT..
IN ORDER TO SAVE CORE FOR MORE SE'S THE ITEMGET PROGRAM (NAME = CRTI) IS CONSTRUCTED AS AN OVERLAY OF 3 PROCEDURES, THE ROOT, THE CONTROL CARD INTERPRETER, AND THE SELECTOR.

CARD INPUT (SYSIN FILE) IS TO THE ROOT SECTION FOR THE OPTION CARD WHICH HAS 80 BINARY OPTION LOCATIONS. ALL OTHER SYSIN INPUT IS TO THE CONTROL CARD INTERPRETER. THIS INTERPRETER CONSIDERS ALL INPUT TO BE IN THE FORM OF STATEMENTS. OFFICIALLY A STATEMENT STARTS WITH A DIGIT FROM 0 TO 8, AND ENDS WITH 2 OR MORE CONSECUTIVE BLANKS. ACTUALLY THE INTERPRETER MAY BE SOMEWHAT MORE LENIENT, BUT DON'T COUNT ON IT.

/*
//TRTDOC JOB 99999211, WCD, MSGLEVEL=(0, 0)
//LSPRINTS PROC LIB=SRCCLB, MEM=ERROR, OUTVOL=GWSSJR1, DSSF=MOD, SSF=300,
// OPT=DIRECTS, CHAIN=SN

//PRINT EXEC PGM=LSPRINT
//STEPLIB DD UNIT=231, VOL=SER=GWSSJR1, LSN=0001, DISP=OLD
//SYSIN DD SYSOUT=A, UCS=(&CHAIN)
//RAKUP DD UNIT=231, VOL=SER=ECOUTVOL, DSN=6LIE, (MEM),
// DISP=(EDSSE, KEEP), SPACE=(TRK, (ESS5, 50, 10)),
// DCB=(RECFM=FB, ELKSIZE=360C, LRECL=60)
//SYSIN DD UNIT=231, VOL=SER=GWSSJR1, DSN=SYS1.FRCCLIE(5CP),
// DISP=OLD
//PEND
//LSPRINT EXEC LSPRINTS, MEM=TRTDOC
//PRINT IN DD DATA

*** TRT PROGRAM ***

THIS PROGRAM IS USED TO PRODUCE & DOCUMENT TRANSLATION TABLES. IT HAS 2 PARAMETERS, BOTH SPECIFIED ON THE EXEC CARD.

MAP= & INCNT=.

MAP= SPECIFIES HOW MANY MAPS OF EACH TRANSLATION TABLE ARE TO BE PRINTED. NUMBERS LESS THAN 1 WILL PRODUCE NO MAPS. INCNT= SPECIFIES HOW MANY TRANSLATION TABLES ARE TO BE EXPECTED IN THE INPUT. AGAIN NUMBERS LESS THAN 1 WILL PRODUCE NO TRANSLATE TABLES.

TRT CHECKS FOR 2 ERRORS . .
1) ILLEGAL CHARACTERS. ANYTHING besides COMMA, BLANK, HEX DIGIT, OR SEMICOLON UNLESS IN A COMMENT.
2) OVERFLOW DUE TO TOO MANY QUOTES. THIS HAPPENS WHEN THE SIZE OF THE OUTPUT ON ONE CARD BECOMES TOO BIG DUE TO THE LARGE NUMBER OF APOSTROPHE CHARACTERS (QUOTES). EACH QUOTE IS PUNCHED AS 2 QUOTES ON THE CARD, EXCEPT THE FIRST AND LAST QUOTES, WHICH ARE DELIMITERS.

ALTHOUGH CERTAIN OTHER FORMS OF INPUT ARE ACCEPTABLE, TRT IS DESIGNED TO ACCEPT INPUT AS A LIST OF HEX NUMBERS IN PL1 LIST DIRECTED FORMAT. INPUT OF A TABLE MAY BE DELIMITED BY A SEMICOLON.

IF LESS THAN 256 CHARACTERS ARE SPECIFIED FOR A TABLE THEN THE REMAINDER OF THE 256 BYTE TABLE PRODUCED IS UNDEFINED. THE INPUT ROUTINE IS DESIGNED TO ACCEPT COLS 3-80 OF A CARD AS A COMMENT IF THERE IS AN ASTERISK IN COL 3. THERE MAY BE UP TO 4 OF THESE COMMENT CARDS INTERSPERSED THROUGH THE TEXT. THEY ARE PRINTED
LSP MAILING LIST PROGRAM

INPUT TO THIS PROGRAM IS IN THE FORM OF 80 BYTE RECORDS
(CARDS) THAT ARE DIVIDED INTO TWO 40 BYTE FIELDS, CORRESPONDING TO
ONE LINE EACH OF AN ADDRESS. THE MAXIMUM NUMBER OF CARDS FOR
ONE ADDRESS IS 3, CONSTITUTING A LIMIT OF 6 LINES FOR AN ADDRESS.
THE DELIMITER FOR SEPARATING ONE ADDRESS FROM ANOTHER IS THE "*"
WHICH APPEARS SOMEWHERE IN THE LAST CARD OF EACH ADDRESS.
THE LAST ADDRESS MUST BE FOLLOWED BY A CARD HAVING AN ASTERISK
IN COLUMN ONE.
THREE MODIFICATIONS ARE PERFORMED ON THE ADDRESS LIST BEFORE IT IS
PRINTED.
1) THE DELIMITING ASTERISK IS REMOVED.
2) THE ADDRESSES ARE SORTED ON THE 40 BYTES OF THE FIRST LINE.
3) IF THERE IS A '/' IN THE FIRST FIELD OF ANY ADDRESS THEN
   THE CHARACTERS PRECEDING THE SLASH ARE PLACED AFTER ALL THE
   OTHER NONBLANK CHARACTERS FOLLOWING THE '/' IN THE
   SAME FIELD, SEPARATED FROM
   THE NONBLANK CHARACTERS BY ONE BLANK. ALSO THE SLASH IS
   REMOVED AND THE RESULTING CHARACTER STRING IS PLACED LEFT-ADJUSTED
   ON THE FIRST PRINTED LINE OF THE ADDRESS.

*** THE FORMAT PROGRAM ***

THE PROGRAM, CALLED FRMT, IS USED TO PRINT SORT RECORDS.
IT CAN PRINT THE OUTPUT OF EITHER THE SORT/MERGE PROCESS OR
THE ITEMS STEPS. THE PRINTOUT IS FORMATTED SO AS TO PRINT MOST
SORT RECORDS ON A SINGLE LINE. IF, HOWEVER, AN ITEM IS TOO LONG
FOR ITS FIELD IN THE LINE THEN THE REST OF THE ITEMS FOLLOWING
WILL STILL BE PRINTED IN THEIR PROPER COLUMNS, BUT ON THE
NEXT LINE. IF A SYNTACTIC ACTOR OVERFLOWS ITS FIELD THEN THE
LAST PART OF THE ACTOR IS PRINTED AT THE BEGINNING OF THE NEXT
LINE. MOST SUCH SYNTACTIC ACTORS ARE IN ERROR.
FRMT HAS 2 INPUT FILES, SRDD & SYIN. SRDD IS FOR SORT
RECORD INPUT. SYIN IS FOR THE OPTION CARD. THERE ARE 32 OPTION
LOCATIONS IN COLS 1-32. ONLY OPTION 1 IS MEANINGFUL. IT SHOULD
BE A 0, EXCEPT FOR PROGRAM DEBUGGING. THE OTHER 31 OPTION
LOCATIONS MUST CONTAIN EITHER 0'S OR 1'S. THE OUTPUT FILE
IS SYSPRINT. IT IS FIXED LENGTH AND UNBLOCKED. ELFSIZE IS 133.

*** SAVEDT PROGRAM ***

THIS PROGRAM CAN BE USED FOR MTST INPUT. MTST MATERIAL
IS NO LONGER BEING PRODUCED, SO THIS PROGRAM IS UNLIKELY TO
BE USED ANY MORE, AND IS THUS NOT VERY
BE USED ANY MORE, AND IS THUS NOT DOCUMENTED VERY WELL. ITS
OUTPUT CAN BE USED BY THE STRUCTUR PROGRAM IF 2 THINGS ARE
DONE: 1) EITHER REMOVE ALL MTST TAPE ENTRIES OR PUT IN THE
MTST TAPE ENTRY PROCESSING STATEMENTS MENTIONED IN THE STRUCTUR
LISTING AND RECOMPILE (IN WHICH CASE SCANNER T SECTIONS WILL NOT WORK TOO WELL). 2) SET OPTION 5 IN STRICT TO 0.

/*
/EDTDOC JOB 99999211, WCD, MSGLEVEL=(1,1)
/D EXEC LSPRINTS, OPT=CARDOCSN, HEM=DOEDIT
/PRINT,IN DD DDNAME=INSTS
/PRINT, INSYS DD *

EDT IS AN INTERACTIVE EDITING PROGRAM DESIGNED TO MANIPULATE MTST ORIGINAL DATA FILES. IT IS NOT WELL DOCUMENTED, COMMENTED, OR EVEN DEBUGGED NECESSARILY, BECAUSE IT IS NO LONGER USED DUE TO THE CHANGE TO THE SCANNER SYSTEM. IT IS REPLACED BY ICIFIX, AND POSSIBLY BY RAXED. EDT DOES WORK THOUGH, AND HAS BEEN EXTENSIVELY USED TO ENHANCE THE LARGE MTST SAMPLE A NUMBER OF TIMES THROUGH THE CYCLE.

TREATMENT OF '1/2' CHARACTER IN EDT PROGRAM ...

TREATMENT FOR FINISH COMMAND IS SAME AS FOR SUCCESSIVE READ, THEN WRITE COMMANDS.


THE CHARACTER IS LEFT UNALTERED, BUT IT MAY TRANSLATE WRONG ON OUTPUT OR CAUSE TROUBLE IN GENERAL. IT SHOULD BE CORRECTED BY EDITING, IF POSSIBLE.

/TAPELIST JOB 99999211,WCD, MSGLEVEL=(1,1)
/JOBLIB DD UNIT=2311, VOL=SER=GWSJR1, DISP=OLD, DSN=UTLIB
/G EXEC PGM=DOCPRT
/SYSPRINT DD SYSOUT=A, UCS=(HN, FOLD)
/SYSIN DD *

*** LONDON STAGE PROJECT TAPE USAGE LIST ***

LSP01 NL, FREE
LSP02 SL, PRINT FILE TAPE, DSN=SYSOUT
LSP03 SL, FREE
LSP04 NL, FREE
LSP09 SL, FREE (FORMER REV 2 OF BATCH 1)
LSP10 SL, ICISCAN SCRATCH OUTPUT, DSN IS LSPFU
LSP11 SL, STRUCT SCRATCH OUTPUT, DSN IS STES
LSP12 SL, LADDER SCRATCH OUTPUT, DSN IS LSPLD
LSP13 SL, ITEMGET SCRATCH OUTPUT, DSN IS LSPIT, VCL NO. 1.
LSP14 SL, ITEMGET SCRATCH OUTPUT, DSN IS LSPIT, VCL NO. 2.
LSP15 SL, ITEMGET SCRATCH OUTPUT, DSN IS LSPIT, VCL NO. 3.
LSP16 SL, ITEMGET SCRATCH OUTPUT, DSN IS LSPIT, VCL NO. 4.
LSP17 SL, REV 3 OF BATCH 1 A897776C. 3 MORE FIXES, 6069, 7198, 9015.
LSP18 SL, DEFAULT SORT OUTPUT.
LSP19 NL, FREE
LSP20 NL, FREE
LSP21 NL, BACKUP OF GWSJR1 9/10/71
LSP22 NL, GWSJR1 BACKUP 9/20/71 AM
LSP23 SL, LONGTERM FILE TAPE.
   1 GENSPECS.JNO971, GENERAL SPECS SIGNED ON JUNE 9TH, 1971
   2 LSPMT.MY017110 MTST CORRECTED SAMPLE.
   3 UNALTERED RELOCKED (3200) BATCH 1 A987776C UNCHRD
LSP24 SL, FREE
LSP25 SL, DATATEXT A177776 FROM REV 3 OF BATCH 1, CHK.
LSP26 SL, CORRTXT ZA177776 FROM REV 3 OF BATCH 1, CHK.
LSP27 SL, STANDARD ICIFRONT OUTPUT, DSN VARIES.
LSP28 NL, GWSJR1 BACKUP 10/2/71 AM.
LSP29 NL, GWSJR1 BACKUP 10/9/71 AM.
LSP30 SL, LSPCR IS DSN, AA877896, CHOPPED FROM LSP026 ZA177776
THE SYSTEM WAS ORIGINALLY DESIGNED TO USE THE MTST FORMAT, BUT IN APRIL 1970 THE DECISION WAS MADE TO USE A DATA PREPARATION FIRM IN HONGKONG AND OPTICALLY SCAN THEIR TYPED WRITTEN OUTPUT. THIS METHOD WAS MUCH CHEAPER THAN THE MTST METHOD BUT IT WAS SLOWER, MORE ERROR PRONE, AND IT REQUIRED NEW PROGRAMES. IN EMPLOYING THIS METHOD IT WAS THOUGHT WISER NOT TO HAVE THE GIRLS IN HONGKONG ATTEMPT TO DO SOME OF THE MORE SCHEMATIC TRICKS INVOLVED IN MAKING A PRECISE LOGICAL REFLECTION OF THE SEMANTIC INFORMATION, UNDERSTANDABLE ONLY TO HUMANS, IN THE LONDON STAGE.

THE GENERAL SPECIFICATIONS OF JUNE 9, 1971 DESCRIBE THE ORIGINAL SYSTEM OF SCANNER-ORIENTED PROGRAMS, WHICH WERE ALMOST IDENTICAL TO THE MTST PROGRAMS EXCEPT THAT SAVED (MTST) WAS REPLACED BY ICISCAN (SCANNER).

SOME TIME LATER THE DATA PREPARATION COMPANY INFORMED US OF ITS NEED FOR AN ERROR CORRECTION SYSTEM. APPARENTLY THEIR ERROR RATE, AS THEY DETERMINED IT, WAS NOT WITHIN THE AGREED LIMIT. ON JULY 21, 1971 THEY WERE SENT A COPY OF 'CORRECTION INSTRUCTIONS FOR THE LONDON STAGE PROJECT', DESCRIBING THE SYSTEM WHICH WE DEVELOPED.
SOME TIME AFTER THE FIRST SCANNED SAMPLE WITH CORRECTION TEXT
ARRIVED ON AUGUST 20, 1871, HOWEVER, WORK ON THE CORRECTION
PROGRAM WAS SUSPENDED BECAUSE THE RATIO OF CORRECTION TEXT TO DATA
TEXT IN THE SAMPLE WAS ONLY 1 PERCENT. THIS WAS SO SMALL THAT IF
THE RATIO HOLD FOR ALL DATA THEN ONLY ABOUT 75 DOLLARS OF TYING
WOULD BE PUT INTO CORRECTION TEXT. THIS AMOUNT WOULD JUSTIFY
ONLY A FEW DAYS OF PROGRAMMING. LATER Batches PROVED TO HAVE LARGER
PROPORTIONS OF CORRECTION TEXT, BUT OTHER WORK KEPT THE CORRECTION
SYSTEM FROM BEING COMPLETED UNTIL EARLY OCTOBER 1971, SHORTLY
BEFORE THE FIRST SAMPLE OF BATCH 2 ARRIVED. (UNFORTUNATELY THIS
SAMPLE HAD NO CORRECTION TEXT IN IT).

FORMATS FOR NONSTANDARD TIME ENTRIES ...
1) END/ FIRST CAP.
2) END/ RMN/ FIRST CAP.
3) END OF ACT/ RMN/ FIRST CAP
4) IN ACT/ RMN/ FIRST CAP
1) FOR 'END' OR 'END OF ACT' SET UP SMALL 'A'. FOR 'IN ACT' SMALL 'D'.
2) CHECK FOR RMN. IF FIND THEN ACCEPT AND PUT OUT AS TIME ENTRY.
ELSE TAKE FIRST CAP IF ANY & PUT OUT WITH TIME ENTRY.
ELSE TAKE FIRST CAP IF ANY & PUT OUT AS TIME ENTRY.
3) IF NO CAP THEN TAKE AS ROLE OR ACTOR.

LISP SYSTEM GENERATION
1) IPL NOT USING LSP'S PROCEDURE ON GWSJR1.
2) VARY GWSJR1 (TO BE) OFFLINE.
3) IEHDASDR AS SINGLE JOB IN STREAM. VTCC=2, EXTENT=3.
4) VARY GWSJR1 ONLINE AGAIN.
5) RUN IEBCOPY AS SINGLE JOB WITH NEW SPACE=
6) RE-IPL USING SYS1.PROCLIB ON GWSJR1.
7) RUN PLIST COMP USING LSPUTGEN AND DTE=NEW, USE=XXXX. ALSO
OVERRIDING TEMP DATASET VOL PARMS TO ENSURE UTILTE WILL
BE ALLOCATED AT BOTTOM OF GWSJR1.
8) RUN ICFIX WITH LSPUTGEN AND DSSP=NEW, SSF=
ALSO OVERRIDE TEMP DATASET ALLOCS AGAIN, SAME REASON.
9) RUN OTHER COMPILES, INC ITEMS. SAVE COMPLETED DISK
ON 2 SEPARATE TAPES.
10) DELETE OLIB, SRCLIB.
11) SAVE DISK AGAIN ON ONE OR BOTH OF THE TAPES.
ICFIX, ICISCAN, STRCT, LADDER, ITEMS, FBMAT, DOCPRINT,
PLIST, LPRINT, LSPMAIL.

/*
//EDIT JOB 99999211, WCD, MSGLEVEL=1
//E EXEC LSPBACMP, DSSP=OLD, DOSP=OLD, SSP=10, OSE=30, LSD=ELT
//BAKUP.SYNTT DD *, DCB=BLKSIZE=30
(SUBRG,STRG) ...
EDIT .. PROC OPTIONS(MAIN) ..
/* THE NUMBERS REPRESENTED BY 'XX' IN THE COMMAND EXAMPLES ARE
VARIABLE IN LENGTH. */

/* UPDATE COMMANDS ARE SMALL LETTERS.
 'F' FINISH, READ AND WRITE CONSECUTIVE BIKS UNTIL END OF FILE.
 'L' LIST, LIST ENTIRE CURRENT BIK IN MTST FORMAT WITH LINE NUMBERS.
 'DXX' DELETE, DELETE LINE XX.
 'R' READ, READ NEXT BIK INTO CORE.
 'W' WRITE, WRITE OUT CURRENT BIK TO OUTPUT FILE.
 'S' SWITCH, SWITCH PRINT ON BIK I/C TO OPPOSITE STATE.
 THIS IS USED FOR BOTH REGULAR I/C, AND IN THE
 UPDATE MODE TO CONTROL LISTING IN MTST FORMAT.
 THE SWITCH IS INITIALLY IN THE 'OFF' STATE.
 CONTROLS LISTING OF RAW INPUT FOR THE 'R' AND 'F' COMMANDS
 CONTROLS RAW OUTPUT LISTING FOR 'W' AND 'F' COMMANDS.
 CONTROLS MTST FORMAT LISTING FOR 'U' COMMAND.
 'IXXX YYYY$' INSERT, INSERT STRING 'YYYY' AFTER LINE XX.
 LINE 0 DOES NOT EXIST, BUT YOU CAN PUT
 SOMETHING AFTER IT ANYWAY.
 'CXX YYYY$' CHANGE, SAME AS DELETE FOLLOWED BY INSERT WITH
 THE SAME LINE NUMBR.
 'U' UPDATE, CLOSE CURRENT OUTPUT FILE, THEN RECEIVE IT FOR

ALL COMMANDS MAY FOLLOW ANY OTHER COMMAND. THE END OF A COMMAND LIST IS INDICATED BY END OF INPUT STRING.

HOWEVER THE 'F' AND 'U' COMMANDS RESULT IN A RETURN TO THE 'OK' MESSAGE AFTER COMPLETION. COMMANDS FOLLOWING THE F & U COMMANDS ARE NOT EXECUTED.

TO END THE PROGRAM TYPE A CAPITAL 'R' AS A SPECIAL COMMAND.

THERE IS A PROVISION FOR EASY CORRECTION OF TYPING ERRORS THAT YOU DETECT RIGHT AFTER MAKING. IF YOU HAVE TYPED A WRONG CHARACTER THEN THAT CHARACTER CAN BE DELETED BY TYPING AN 'AT' CHARACTER IMMEDIATELY AFTER THE INCORRECT CHARACTER. IN FACT YOU CAN DELETE MORE PREVIOUS CHARACTERS BY TYPING MORE 'AT' SIGNS. THUS 2 SUCCESSIVE 'AT' SIGNS WILL DELETE THE TWO CHARACTERS PRECEDING THE 'AT' SIGNS.

WHEN YOU WANT TO INSERT A QUOTE INTO THE TEXT YOU MUST TYPE IT AS 2 SUCCESSIVE QUOTES. A SINGLE QUOTE IS USED BY THE CONTROL PROGRAM TO DELIMIT THE END OF ALL INPUT, AND THE EDITING PROGRAM IS NOT OF THE QUOTE OR ANY CHARACTER FOLLOWING IT.

NOTE: TWO QUOTES FOLLOWED BY AN 'AT' SIGN RESULTS IN A SINGLE QUOTE OF INPUT, I.E. YOU NEED 2 'AT' SIGNS TO DELETE A DOUBLE QUOTE.

THE INPUT STRING MUST BE 126 CHARACTERS OR LESS. NO BLANKS ARE ALLOWED ANYWHERE EXCEPT AFTER THE 'IXX' OR THE 'CXX', WHERE ONE IS REQUIRED, AND IN THE STRING, 'YYYYY'. /*

DCL (A,B) CHAR(3625) VARYING,
HEX ENTRY(CHAR(132) VAR) RETURNS (CHAR(264) VAR),
SYSPRINT PRINT ENV(F(133,133)),
P CHAR(3625) VAR,
ISTACK1 INITIAL(05),
IOPT(32) BIT(1),
IO INITIAL(1),
HEX00 CHAR(1) INITIAL(' '),
HEX0D CHAR(1) INITIAL('?'),
HEX25 CHAR(1) INITIAL('?'),
HEX62 CHAR(1) INITIAL('?'),
/* THE '1/2' CHARACTER */
EBC CHAR(1) INITIAL('c'),
EBL CHAR(1) INITIAL('l'),
CCHAR(9) CHAR(1) INITIAL('f','l','d','r','w','s','i','c','u'),
PROCED(9) LABEL INITIAL(BUTFHRU,LISTALL,DELETE,READLK,
WRTBLK,SRCH,INSERT,CHANGE,UPDATE),
T1DEF(8) CHAR(32) INITIAL(
'z567091378u4tnekkbbxudc?l?l?',
'>.<1??swmzsa??o?j=pq ?y-gf?,??;',
'z-0/2?) 1|6*?j?tnekk?hexudc?l?',
'??1??swmzsa??o?j=pc y-y_gf?"???:',
'?????????????????????????????',
'?????????????????????????????',
'?????????????????????????????',
'T1 CHAR(256) DEFINED T1DEF,
T2DEF(8) CHAR(32) INITIAL(
'?????????????????????????????????????????????????????-????',
'?????????????????????????????????????????????????????',
'?????????????????????????????????????????????????????',
'?????????????????????????????????????????????????????',
'?????????????????????????????????????????????????????',
'?????????????????????????????????????????????????????',
'?????????????????????????????????????????????????????',
'T2 CHAR(256) DEFINED T2DEF,
J INITIAL(0),
IBADCHAR INITIAL(0),
BADCHAR CHAR(1) 'DEF' IBADCHAR POS(2),
LNS DEF LINESOUTH,
BIT1 BIT(1) DEF IOPT,
C CHAR(126) VARYING,
STR CHAR(126) VARYING,
ICMND INITIAL(0),
CMND CHAR(1) DEF ICMND PCS(2),
SMOD CHAR(1) 
ON ENDFILE(IN) BEGIN
DISPLAY('END OF FILE') . . GOTO OK . . END 
ON ENDFILE(SYSIN) 
GET FILE(SYSIN) EDIT(IOPT) (E(1)) .
ON ERROR SNAF CALL PUTDATA :

PUTDATA . . PROC 
PUT FILE(SYSPRINT) SKIP(1) LIST('ERROR CODE= ',CNCODE) 
PUT DATA . . END .
IF IOPT(30) THEN
DO I=1 TO 8

DISPLAY(T1DEF(I) CAT ' ' ' ' CAT T2DEF(I) CAT ' ' ' ')
END .

OK . .
C=REPEAT(HEX00,126) . . /* INITIALIZE C WITH HEX ZEROS FOR */
DISPLAY('OK') REPLY(C) . . /* DELIMITING END OF INPUT CN 1052. */

ICL=INDEX(C,HEX00) . .
IF IOPT(21) THEN PUT SKIP DATA(ICL,C) . .
IF ICL=0 THEN ICL=126 ELSE ICL=ICL-1 .
IF IOPT(21) THEN PUT DATA(C,ICL)
C=SUBSTR(C,1,ICL) . .
IF IOPT(20) THEN DO 

DISPLAY(ICL CAT ' ' ' ' CAT SUBSTR(C,1,12) CAT ' ' ' ' CAT HEX(SUBSTR(C,1,10)))
END .
ON STRG . .
GETRUB . . N=INDEX(C,')') . .
IF N GT 1 THEN DO

C=SUBSTR(C,1,N-2) CAT SUBSTR(C,N+1)
ICL=LENGTH(C) . . GOTO GETRUB . . END .

N=0 . .
GETQUOTE . .
I=N . .
N=INDEX(SUBSTR(C,1+1),''''''') . .
IF N GT 0 THEN DO

N=L+N . . C=SUBSTR(C,1,N) CAT SUBSTR(C,N+2)
GOTO GETQUOTE . . END .

REVERT STRG .
N=0 . .
IF IOPT(20) THEN PUT FILE(SYSPRINT) SKIP(1) EDIT ('FIXED STRING='''',C,'''') (A . .

NXTCMND . .
IF N GE ICL THEN GOTO OK . .
N=N+1 . .
LINESOUGHT=20000 . . /* SET LINE SOUGHT CUT OF SIGHT */
CMND=SUBSTR(C,N,1) . . /* GET COMMAND LETTER */
IF IOPT(9) THEN DO

PUT SKIP EDIT('CMND='',CMND,ICMND) (A . .

DISPLAY(CMND CAT ICMND) . .
END .
CALL FIX . .
IF CMND GE '0' THEN IF CMND LE '9' THEN GOTO STACKUP .
IF CMND='.' THEN DO

ISTACK2=ISTACK1 . . N=N+1 . . GOTO STACKUP . . END .
IF CMND='.' THEN DO

PUT FILE(SYSPRINT) SKIP(1) EDIT
((IX DO IX=1 TO 9),
(SUBSTR(E,L,MIN(100,I-L+1)) DO L=1 TO I EY 100)
(9 P(10),99 (SKIP,A)) . .
GOTO NXTCMND . . END .

IF CMND='B' THEN DO:
  A=B...
  GOTO NXTCMND...
END.

IF CMND='A' THEN DO:
  B=A...
  GOTO NXTCMND...
END.

IF CMND='L' THEN DO:
  MORELIST...
CMND=1'

CALL INPUT...
CALL MTLIST...
GOTO MORELIST...
END.

IF CMND='P' THEN DO:
  PUT SKIP LIST('BEGINNING PUT',ISTACK2,ISTACK1)...
  P=SUBSTR(B,1,ISTACK2-1) CAT SUBSTR(E,ISTACK1+1)...
  E=SUBSTR(B,ISTACK2,ISTACK1-ISTACK2+1)...
  CALL OUTPUT...
  B=P...
  GOTO NXTCMND...
END.

IF CMND='T' THEN DO:
  CALL GETLINE...
  PUT SKIP LIST('DOING TYPE',LINX,SUBSTR(E,K,INTST-K))...
  DISPLAY(LINO CAT SUBSTR(E,K,INTST-K))...
  GOTO NXTCMND...
END.

DO IX=1 TO 9...
  IF CCHAR(IX)=CMND THEN GOTO PROCED(IX)...
END...

IF CMND='R' THEN RETURN...
IF CMND='!' THEN GOTO OK...
DISPLAY('UNDEFINED COMMAND' ' CAT '...
  ' CAT CMND CAT ' ')
GOTO OK...

STACKUP...
N=N-1...
CALL GETLINE...
ISTACK1=LNS...
PUT SKIP DATA(ISTACK1,ISTACK2)...
N=N-1...
GOTO NXTCMND...

CHANGE...
INSERT...

CALL GETLINE...

IX=INDEX(SUBSTR(C,N+1),'$')...
IF IX=0 THEN DO...
DISPLAY('UNABLE TO FIND "$" DELIMITER')...
PUT SKIP (3) DATA(IX,C,N)...
GOTO OK...
END...

IX=IX+1...
B=SUBSTR(C,N,IX-N) CAT '??'...
N=IX...
/* SET PTR TO NEXT COMMAND FOR CONTINUOUS CMND*/
IX=-1...
/* SET PTR TO CAUSE INSERTION ONLY. */
IF CMND=EBE THEN DO...
/* MESSAGE, AND THEN RESETING A */
/* MESSAGE, AND THEN RESETING A */
/* MESSAGE, AND THEN RESETING A */
/* MESSAGE, AND THEN RESETING A */
/* PUT FILE(SYSPRINT) SKIP(1) EDIT
('LINE',LINX,'...SUBSTR(B,K,L-K) ...DELETED BY CHANGE')
(A,F(3),3 A)...
IX=K...
/* RESET PTR TO CAUSE DELETION TOO. */
END...

B=SUBSTR(B,1,IX-1) CAT STR CAT SUBSTR(E,L)...
CALL FIX...

B=SUBSTR(B,1,K-1) CAT SUBSTR(B,1)...
CALL FIX...

DELETE...
CALL GETLINE...

STR=SUBSTR(E,K,L-K+1)...
B=SUBSTR(B,1,K-1) CAT SUBSTR(B,1)...
CALL FIX...

PUT FILE(SYSPRINT) SKIP(1) EDIT
('LINE',LINX, '...SUBSTR(E,K,L-K) ...DELETED BY DELETE')
(A,F(3),3 A)...
N=N-1...
GOTO NXTCMND 

SRCH 

CALL GETLINE 

IOPT(LNS)=NOT IOPT(LNS) 

DISPLAY('OPT' CAT LNS CAT ' = ' CAT IOPT(LNS)) 

N=N-1 

GOTO NXTCMND 

PUTTHRU 

PUT SKIP LIST('************ FINISH COMMAND GIVEN') 

PUTPUT 

CALL INPUT 

CALL OUTPUT 

GOTO PUTPUT 

GETLINE .. PROC /* ASSUMES DECIMAL NUMEER CHAF STRING STARTS */ /* IN C AT OFFSET=(N+1) FROM START OF C */ 

CALL FIX 

IRC=VERIFY(SUBSTR(C,N+1),"0123456789") 

IF IRC LE 1 THEN DO 

DISPLAY('SUSPICIOUS NUMERICAL PARAMETER') 

PUT SKIP(3) DATA(IRC,C,N) 

GOTO OK 

END 

IRC=IRC+N 

/* SETS IRC TO PT TO 1ST CHAR AFTER NUMBER */ /* STRING'S END. */ 

GET STRING(SUBSTR(C,N+1,IRC-N-1)) EDIT(LINESought) 

(F(IRC-N-1)) 

N=IRC 

/* NOW N PTS TO 1ST CHAR AFTER NUMEER STRING */ 

SMOD=SUBSTR(C,N,1) 

IF SMOD=' ' THEN N=N+1 

CALL MLIST /* FIND LINE WHOSE NUMBER IS HELD IN */ /* THE VARIABLE LINESought */ 

END 

MLIST .. PROC /* FINDS A LINE AND SETS UP PTERs TO THE */ /* BEGINNING AND END OF IT. ALSO LISTS LINES */ /* AS IT IS SEARCHING IF THE COMMAND LETTER */ /* IN THE VARIABLE, CMD, IS 'I'. */ /* ON RETURN K PTS TO 1ST CHAR OF LINE. L PTS TC 1ST */ /* CHAR OF NEXT LINE, IF ANY. IF NO NEXT LINE THEN L */ /* PTS TO T+1 WHERE T IS THE CURRENT LENGTH OF THE F */ /* BUFFER. BOTH K & L =1 IF LINE SPECIFIED IS LINE 0. */ /* IF THE LINE NUMBER SPECIFIED IS GREATER THAN THE */ /* NUMBER OF THE ACTUAL LAST LINE THEN THE LAST LINE */ /* IS DELIVERED INSTEAD. */ /* IF CMD=ERL THEN */ 

PUT FILE(SYSPRINT) SKIP(2) EDIT 

('*********** LISTING REQUESTED OF INPUT BLK NC.' ,J, 
', OUTPUT BLK NO. WILL BE' ,IC,' LENGTH=',L) 

(2 (A,F(4)),A,F(5)) 

LINO=0 

L=1 

K=1 

ON STRG SNAP PUT DATA(K,L,I) 

NEXTLINE .. IF LINO=LINESought THEN RETURN .. 

LINO=LINO+1 

K=L 

L=INDEX(SUBSTR(B,K,I-K+1),'?') 

/* 12-9-8-5 EBCDIC CR OD */ 

IF L=0 THEN L=1 ELSE L=L+K-1 

IF CMD='L' THEN PUT FILE(SYSPRINT) SKIP(1) EDIT 

(LINO, SUBSTR(B,K,L-K)) (F(3), X(4), A) 

INTST=L 

FINDSTRT 

IF L=I THEN RETURN 

SMOD=SUBSTR(B,L/I) 

IF SMOD NE '?' THEN IF SMOD NE '?' THEN GTCIC NEXTLINE .. 

L=L+1 

GOTO FINDSTRT
END

FIX .. PROC ..
I=LENGTH(B) ..

CUTBACK ..
IF I=0 THEN GOTO APPEND ..
SMOD=SUBSTR(B,1,1) ..
IF SMOD=HEXOD OR SMOD=HEX25 THEN DO ..
I=I-1 .. B=SUBSTR(B,1,I) .. GOTO CUTBACK .. END ..

CUTFORWARD ..
IF I=0 THEN GOTO APPEND ..
SMOD=SUBSTR(B,1,1) ..
IF SMOD=HEXOD OR SMOD=HEX25 THEN DO ..
I=I-1 .. B=SUBSTR(B,2) .. GOTO CUTFORWARD .. END ..

APPEND ..
B=B CAT HEXOD CAT HEX25 .. I=I+2 .. END FIX ..

INPUT .. PROC ..
READ FILE(IN) INTO(A) ..
IF IOPT(20) THEN
PUT SKIP LIST('AFTER INPUT READ') ..
I=LENGTH(A) ..
B=TRANSLATE(A,T1) ..
J=I+1 ..
IF BIT1 THEN
PUT SKIP LIST(B) ..
DO K12=I TO MAX(I-70,1) BY -1 ..
IF SUBSTR(B,K12,1)=HEX62 THEN GOTO FOUND12 .. END ..
DISPLAY('NO ' '1/2' ' CHAR WITHIN 70 BYTES OF ECE') ..
GOTO LENFIXED ..

FOUND12 ..
I=K12-1 ..
B=SUBSTR(B,1,1) ..

LENFIXED ..
A=B ..
DO K12=1 TO I ..
IF SUBSTR(A,K12,1)=HEX62 THEN SUBSTR(A,K12,1)=?' .. END ..
K12=0 ..
IERRCNT=0 ..

RESEARCH ..
L=INDEX(SUBSTR(A,K12+1),?'') ..
IF L=0 THEN RETURN ..
IERRCNT=IERRCNT+1 ..
K12=L+K12 ..
BADCHAR=SUBSTR(B,K12,1) ..
DISPLAY('BAD CHAR: ' 'AT' ' CAT K12 CAT HEX(BADCHAR)'), ..
IF K12=I THEN RETURN ..
IF IERRCN'T GT 1 THEN RETURN ..
GOTO RESEARCH ..
END ..

OUTPUT .. PROC ..
B=B CAT HEX62 .. /* REINSERT '1/2' CHAR CN OUTPUT */
IF IOPT(20) THEN
PUT SKIP LIST('BEGINNING OUTPUT PROCESSING.') ..
A='A IS THIS SENTENCE JUST BEFORE OUTPUT TRANSLATION.' ..
A=TRANSLATE(B,T2) ..
IF BIT1 THEN
DO ..
PUT SKIP LIST(B) ..
PUT SKIP LIST(TRANSLATE(A,T1)) ..
END ..
WRITE FILE(TOUT) FROM(A) ..
IO=IO+1 ..
END ..

LISTALL .. CALL MLIST ..
PUT SKIP(32) ..
GOTO NXTCMDND ..
BIT 1 BIT(1)

SMOD CHAR(1) VARYING, /* CURRENT SECTYPE INDICATOR CHAR */
EBC CHAR(1) INITIAL('C') /* SMALL EBCDIC 'C' */
EBSM CHAR(26) INITIAL('abcdefghijklmnopqrstuvwxyz')
EBSMQ CHAR(27) INITIAL('abcdefghijklmnopqrstuvwxyz')
EBLBRK CHAR(1) INITIAL('I')
EBRBRK CHAR(1) INITIAL('I')
PO 9 6H26 CHAR(1) INITIAL('?') /* EOF GARBAGE CODES IN MT */
CH2 CHAR(2),
CH3 CHAR(3),
C1 CHAR(1) DEFINED CH3 POSITION(3),
C2 CHAR(1) DEFINED CH3,
M12 CHAR(1) INITIAL('?') /* 11.0.9.2 H'62' FCR MT & EBCDIC */
MTPRDCR CHAR(2) INITIAL('?')
MTQSTCR CHAR(2) INITIAL('?')
MTEXCR CHAR(2) INITIAL('?')
MTLF CHAR(1) INITIAL('?') /* 11.9.5 PUNCHES H'15' */
MTBLNK CHAR(3) INITIAL('?')
MTASC CHAR(17) INITIAL('('17') ?')
LRECD CHAR(18),
T1DEF(8) CHAR(32) INITIAL('z256 ?091378?[4?tek? hbxudc??1?'
', '].[12?SMMVRA??o?]=pq ?y-gf?,??';
', 'Z-%/ (?)(1+8?)$('?tek? HBXUDC??1?'
', '?*?'?i?SMMVRA??o?J@eq? y-gf?"?'?';
', '?????????????????????????????????
', '????????????????????????????????
', '????????????????????????????????
', '???????????????????????????????')
T1 CHAR(256) DEFINED T1DEF, /* MTST TC EBCDIC TR TABL */
ISKIP INITIAL(0),
J INITIAL(0),
ON ENDFILE(IN) GOTO ENDFILE,,
ON CONVERSION GOTO CONGO, /* LAFPL VARIABLE */
ON ENDFILE(SYSIN)
GET FILE(SYSIN) EDIT(IOPT) (B(1)),
NP=0

INPUT .. READ FILE(IN) INTO(A),
J=J+1, /* J NOW = THE NUMBER OF GCCD INPUT BLOCKS */
I=LENGTH(A),
IF I LE 20 THEN DO,
PUT FILE(SYSPRINT) SKIP(4) EDIT
('BLK', 'J,' IS SHORT, LENGTH =', 'I') (2 (F(G))),
('***', 'TRANSLATE(A,T1)', '***') (SKIP, A, A, A),
J=J-1,
GOTO INPUT,
END
ELSE WRITE FILE(TOUT) FROM(A),
IPAGES=NP,
B=TRANSLATE(A,T1),
PUT FILE(SYSPRINT) SKIP(3) EDIT
('BLK', 'J,' HAS A LENGTH OF', 'I', '***',
SUBSTR(B, 1, MIN(I, 70)), '***',
R(G), R(H), 3 A),
K=2

PERIODS .. M=K, /* SEARCH FOR A FEED CODE */
IF K=0 THEN GOTO TABS, /* IF NOT FIND THEN ALL DONE */
K=4*K,
/* NOW K COUNTS TO THE FEED */
CH2=SUBSTR(A, K-2, 2),
IF CH2 NE MTPRDCR AND CH2 NE MTEXCR AND CH2 NE MTQSTCR THEN
DO, /* REMOVE A FEED CHAR */
K=K-1,
A=SUBSTR(A, 1, K) CAT SUBSTR(A, K+2),
END,
GOTOC PERIODS, /* DELETE REMAINING FEEDS, IF ANY */
ILX=INDEX(B, 'I!')  /* CHECK FOR NEW MTST TAPE ENTRY */
IF ILX=0 THEN GOTO NOTAPE  
IRX=INDEX(SUBSTR(B,ILX+2,5), 'I')  /* GET NEXT BLNK*/
IF IRX LT 2 THEN GOTO NOTAPE  /* WRONG LENGTH, GIVE UP */
C=SUBSTR(B,ILX+2,IRX-1)  /* SET UP FOR CONVERSION */
ION=N  /* SAVE LAST MTST TAPE NUMER */
CONGO=NOTAPE  /* DEFINE WHERE TO GO CN CONVERSION INTERRUPT */
GET STRING(C) EDIT (N) (F(IRX-1))  /* N RESERVED */
MTNO=0  /* RESET BLK CNT FOR MTST TAPE */
IF IOPT(2) THEN
IF ILX GT 9 THEN PUT FILE(SYSPRINT) SKIP(1) EDIT
('POSSIBLE ERROR - MT TAPE ENTRY STARTS AT',ILX) (A,F(4))  .
IF IOPT(15) THEN
IF N NE ION+1 AND ION NE 0 THEN DO .
PUT FILE(SYSPRINT) SKIP(2) EDIT
('POSSIBLE ERROR ***** NONCONSECUTIVE MTST ENTRIES',
ION,' ',N) (2 R(G))  .
CALL ERREMSG('MTOO',ILX,2,SMCD,'0'E)  .
END  .
NOTAPE  .
MTNO=MTNO+1  /* INCREMENT MTST TAPE BLK CNT */
PUT FILE(SYSPRINT) SKIP(1) EDIT
('REF. CODE,'MTNO,' OF MTST TAPE NO.'N) (2 E(G))  .
IF IOPT(1) THEN PUT FILE(SYSPRINT) SKIP(2) EDIT(B) (A)  .
L=INDEX(B,'?')  .
IF L NE 0 THEN PUT FILE(SYSPRINT) SKIP(2) EDIT
('ILLEGAL(0.8.2) CHAR FOUND IN BLK',J,
' AT POSITION',L) (2 (R(G)))  .
I=LENGTH(A)  .
DO M=1 TO MAX(1,I-70) BY -1  /* SEARCH BACKWARDS FOR ',' */
IF SUBSTR(A,M,1)=M12 THEN DO .
K=INDEX(A,M12)  .
END  .
IF K NE M THEN DO  .
PUT FILE(SYSPRINT) SKIP(1) EDIT
('POSSIBLE ERROR- ',M12,' CHAR IN BLK',J,' AT',M)
(2 (R(H)))  .
PUT DATA(K,M12)  .
END  .
GOTO PRINTIN  .
END  .
END  .
IF IOPT(4) THEN
PUT FILE(SYSPRINT) SKIP(1) EDIT
('ERROR- NO ','1/2',' CHAR WITHIN 70 BYTES OF ECE') (A)  .
M=1  /* USE WHOLE BLK */
PRINTIN  .
A=M13BLNK CAT SUBSTR(A,1,M-1) CAT M13BLNK  .
E=TRANSLATE(A,TI)  /* TRANSFORM MTST TO EECIC */
K=INDEX(B,PO.9_6H26)  /* CHECK FOR ILLEGAL CODES IN TEXT */
IF IOPT(5) THEN
IF K NE 0 THEN PUT FILE(SYSPRINT) SKIP(1) EDIT
('BLK',J,' HAS AN ILLEGAL(0.9.6) CODE IN LOCATION',K)
(A,F(4)) (A,F(4))  .
D=TRANSLATE(A,TI)  .
A=REPEAT(' ',3625)  .
ERRFILE=""  .
NOTE=""  .
I=LENGTH(B)  .
ISR=1  /* INITIAL OLD RIGHT SECTION DELIMITER */
IRX=1  /* INITIAL INDEX PTE TO START SCANNING */
/* END OF BLK INPUT PROCESSING AND CHECKS */
NEXTSEC  .
IF ISR NE 1 THEN
IF IOPT(6) THEN
PUT FILE(SYSPRINT) SKIP(1) EDIT
(3 A, F(5))
ISL=ISR
/* SAVE OLD RIGHT SEC-DELIM AS NEW LEFT ONE */
NEXTSECPRT...
IF ISR=1-2
/* IF LAST SEC ALREADY PROCESSED */
DO
/* DO PAGE AND CUTFUT SETUP PROCES */
IF NOT IOPT(7) THEN GOTO SKIPOPT7
PUT FILE(SYSPRINT) SKIP(2) EDIT((K DC R=10 TO 90 BY 10)
(COL(20), 9 (F(2), X(8))))..

SKIPOPT7..
ILX=1
CONG=GETPAG

GETPAG. IRX=INDEX(SUBSTR(B, ILX+1), P').
/* 12.11.7 H'97 */
IF IRX=0 THEN GOTO GETPAG
/* ALL PAGE ENTRIES DONE */
IRX=IRX+ILX+1
/* IRX NOW POINTS TO THE 'P' OF A */
/* PROSPECTIVE PAGE ENTRY */
ILX=IRX
/* SAVE PTR TO START NEXT PAGE SRCH */
IRB=INDEX(SUBSTR(B, IRX+1, 5), '1')..
IF IRX LT 2 THEN GOTO GETPAG
/* NC NUNMEE FCNED */
C=SUBSTR(B, IRX+1, IRX-1)
/* SET UP CONVERSION TRY */
NOP=NP
/* SAVE PAGE NUMMEE FCE ERROR CHECK */
GET STRING(C) EDIT(NP) (F(IRE-1))
/* ATTEMPT CONV */
L=(IRX+1)/100
/* CONVERSION ATTEMPT SUCCEEDED, SO */
NOTEILIST(L)=NOTEILIST(L) CAT '*'
/* NOTE PAGE ENTRY IN MRG */
DO K=L TO 36
IPAGES(K)=NE
END
IF IOPT(14) THEN IF NP LT 1 CE (NP LT NC) THEN
CALL EREMSG('PUNED', IRX+1, 2, SMOD, '1'B)
SUBSTR(A, IRX, 1)='P'
GOTO GETPAG

GOTPG
IF IOPT(7) THEN PUT FILE(SYSPRINT) SKIP(1) EDIT
((ERRFILE(L/100), L/100, SUBSTR(B, L, 100),
IPAGES(L/100), NOTEILIST(L/100),
SUBSTR(A, L, 100), ( K DO K=1 TO 9)
DC L-1 TO I BY 100)
(SKIP(1), A(4), F(3), X(3), A(100), F(6), X(1), A(3),
SKIP(1), COL(11), A(100), SKIP(1), COL(11), 9 (X(9), F(1)))
P=''
IRX=1
/* EURE & INDEX EDITING & CUTFUT */
EDITOUT..
IF IRX=1 THEN GOTO PUTOUT
/* FINISHED, WRITE OUT */
ILX=IRX
/* NOW POINTS TO 1ST BYTE OF NEW AREA */
SMOD=SUBSTR(A, ILX, 1)
/* GET AREA TYPE */
IF SMOD='P' THEN DO
/* IF IT'S A PAGE CHANGE ENTRY */
IRX=INDEX(SUBSTR(B, ILX+1, 5), '1') ILX
/* FIND #1'S END */
C=SUBSTR(B, ILX+1, IRX-ILX-1)
/* SET UP FOR CONVERSION */
TOP=NP
/* SAVE PAGE NO. FOR ERR CHK */
GET STRING(C) EDIT(NP) (F(IRX-ILX-1))
/* ATTEMPT CONV */
PUT STRING(CHPAGE) EDIT(NP) (P(4))
P=P CAT SUBSTR(B, ILX-1, IRX-ILX+2)
IRX=IRX+1
GOTO EDITOUT
END
IRX=VERIFY(SUBSTR(A, ILX+1), SMOD)
/* SET RELATIVE */
/* POINTER TO 1ST BYTE OF NEXT AREA, IF ANY */
IF IRX=0 THEN IRX=1
/* LAST AREA FOUND */
ELSE IRX=IRX+ILX
/* NOT LAST AREA */
IF SMOD='T' THEN DO
/* BLANK (UNDERLINING STRUCTED DATA */
P=P CAT SUBSTR(B, ILX, IRX-ILX)
END
ELSE IF SMOD='T' THEN DO
C2=SUBSTR(B, IRX-1, 1)
IF C2 NE '1' THEN DO
IRX=IRX-1
SMOD=C2
END

C=SMOD CAT SUBSTR(B,ILX+1,IRX-ILX-2) ..
IF SMOD NE 'I' THEN IRX=IRX+1 ..
IF LENGTH(C) LT 20 THEN DO .. /* FAD FCE SETTING .. */
CH14=C ..
C=CH14 ..
END ..
IF IOPT(17) THEN
PUT FILE(SYSPRINT) SKIP(1) LIST(C) ..
WRITE FILE(INDX) PRCM(C) ..
END ..
GOTO EDITOUT ..
PUTOUT ..
IF IOPT(8) THEN
PUT FILE(SYSPRINT) SKIP(1) DATA(P) ..
WRITE FILE(PURE) PRCM(P) ..
GOTO INPUT ..
END ..
/* SAVE INDEX TO CALCULATE NEW ISR */
ISR=INDEX(SUBSTR(B,ISR+1),'1') .. /* FIND NEXT ELNK-ERFX */
IF ISR=0 THEN DO .. /* SET UP LAST SECTION AND PROCESS IT */
ISR=1-2 ..
GOTO DOLASTSEC ..
END ..
ISR=ISR+IX .. /* SET PTR TO ACTUAL LOC OF RIGHT SEC-DELIM */
IF VERIFY(SUBSTR(B,ISR+2,1),ESML) THEN GOTO NEXTSECFR .
/* TRY AGAIN IF THIS IS NOT THE END OF THE SECTION */
DOLASTSEC ..
SMOD=SUBSTR(B,ISL+2,1) .. /* OBSERVE TYPE OF THE SECTION */
/* ABOVE STAT MARKS SECTION START IN PAF MRGN */
NOTE LIST(ISR/100)=NOTE LIST(ISR/100) CAT SUBSTR(B,ISR+2,1) ..
IF SMOD=memcpy THEN /* IF IT'S A COMMENTS SECTION THEN .. */
DO ..
ILB=ISL+2 ..
IRB=ISR-2 ..
DO L=ISL TO ISR-1 .. SUBSTR(A,L,1)='=' .. END ..
INXTRN=NEXTSEC ..
GOTO NEXTIND ..
END ..
IRB=ISL+2 ..
NEXTSECFR ..
ILB=INDEX(SUBSTR(B,IRB+1,ISR-IRB-1),EELBRKT) ..
IF ILB=0 THEN GOTO NEXTSEC ..
ILB=ILB+IRB ..
IRB=INDEX(SUBSTR(B,ILB+1,ISR-IRB-1),EELBRKT) ..
IF IRB=0 THEN DO ..
IF IOPT(11) THEN
PUT SKIP(1) FILE(SYSPRINT) EDIT
( 'ERROR- NO RIGHT BRACKET, LEFT IS AT',ILB ) (B(H)) ..
CALL ERRMSG('NORE',ILB,'JMOD','IB') ..
IRB=ISR-ILB .. /* EXTEND 'BRACKET' TO END OF SECTION */
/* INDEXES SHOULD GC CK, SORRY IF ANY */
/* STRUCTURED DATA IS LCST */
END ..
IRB=IRB+ILB ..
IF SUBSTR(B,ILB-1,1)=':' THEN ILB=ILB-1 ..
DO L=ILB TO IRB .. SUBSTR(A,L,1)='X' .. END ..
INXTRN=NEXTBRKTS ..
NEXTIND ..
ILX=INDEX(SUBSTR(B,IRX+1,IRX-IRX-1),VL) ..
IF ILX=0 THEN GOTO INDXTRN .. /* EXIT-RETURN FROM INDXN */
ILX=ILX+IRX .. /* SET UP C1 AND C2 */
CH3=SUBSTR(B,ILX-1,3) ..
IF VERIFY(C1,ESML)=1 THEN BIT1='1'B .. ELSE BIT1='0' ..
IF C2=' ' AND NOT BIT1 OR C1=EELBRKT OR C1=EERKRT THEN
DO .. /* SO THIS WASN'T THE START OF AN INDEX AFTER */
IRX=ILX .. /* ALL, BUT RESET THE SEARCH ECINTER AND */
GOTO NEXTIND / * KEEP SEARCHING FOR START OF INDEX */
END
IF ILX LE ILB THEN DO
PUT FILE (SYSPRINT) SKIP (1) EDIT
('INDEX OUTSIDE OF BRACKETS STARTING AT ', ILX) (2 (H))
CALL ERRMSG ('IXOB', ILX, '1', SMOD, '1'B)
END
IRX=INDEX (SUBSTR (B, ILX + 1, IRB - ILX - 1), 'I')
IF IRX=0 THEN DO
IRX=INDEX (SUBSTR (B, ILX + 1, IRB - ILX - 1), 'I'
IF IRX=0 THEN DO
PUT FILE (SYSPRINT) SKIP (1) EDIT
('ERROR - NO MATCHING INDEX CLOSER, ILX=', ILX, ' IRE=', IRE)
(2 (H))
CALL ERRMSG ('NORX', ILX, '1', SMOD, '1'B)
IRX=ILX
GOTO INDEXRTEN
END
IRX=IRX+ ILX
GOTO VERISMALL
END
IRX=IRX+ ILX
CH3=SUBSTR (B, ILX - 1, 3) / * SET UP C1 AND C2 */
IF VERIFY (C1, EBSM)=1 THEN BIT1='1'E / ELSE BIT1='0'E
IF C2=' ' AND NOT BIT1 OR C1=EBBRT OR C1=EBRRT THEN
DO
PUT FILE (SYSPRINT) SKIP (1) EDIT
('CAUGHT INDEX CROSSING SECTION OR BRACKET AT ', ILX
, ', IRB=', IRB) (A, F (4), A, F (4))
('INDEX BEGINS AT ', ILX) (A, F (4))
CALL ERRMSG ('YCSB', ILX, '1', SMOD, '1'B)
GOTO INDEXRTEN
END
VERISMALL
IF VERIFY (SUBSTR (B, IRX + 1, 1), EBSM)=0 THEN / * INDEX DID */
/ * END WITH A SMALL LETTER OR QUOTE */
DO
PUT FILE (SYSPRINT) SKIP (1) EDIT
('BAD END OF INDEX VERIFIED AT ', ILX, ' INDEX STARTED AT ', ILX
(2 (H))
CALL ERRMSG ('BNDX', ILX, '1', SMOD, '1'B)
END
DO L=ILX TO IRX+1 .. SUBSTR (A, L, 1)= 'I' .. END
IF SUBSTR (B, IRX, 1)= '/' THEN GOTO NEXTIND
SUBSTR (B, IRX, 1)=SUBSTR (B, IRX + 1, 1)
SUBSTR (B, IRX + 1, 1)= ' ' / * FOR NEATNESS */
GOTO NEXTIND

ERRMSG . . PROCEDURE (ABBREV, LOC, ITYPE, SECTYPE, PRINTOPT),
DCL ABBREV CHAR (120) VARYING,
LOC FIXED BIN (15, 0),
PRINTOPT BIT (1),
ITYPE FIXED BIN (15, 0),
MSG (2) CHAR (8) VARYING INITIAL ('', 'FCSSIBLE'),
SECTYPE CHAR (1) VARYING ,
IF IOPT (12) THEN IF PRINTOPT THEN
PUT FILE (SYSPRINT) SKIP (2) EDIT
(MSG (ITYPE), 'ERROR **** ' , ABBREV, ' IN ' , SECTYPE, ' LOC=', LOC
(SKIP (2) 6 A, F (5))
ERRFILE (LOC/100)= ABBREV
END ERRMSG
ENDFPROG . . END SAVEDIT

//LSFMAIL JOB 999992711, WCD; MSGLEVEL=(1, 1)
// EXEC PL1LG, PARM='*NCP'NT'
//GO.SYSLIB DD
//DD DSN=SYS1.SORTLIB, DISP=SHF