```
COMMCIO
COMMON
 COMMCIO  CTEXT  COMMCIO - FILE ACTION MACROS.
 COMMCIO  SPACE  4
***       COMMCIO - FILE ACTION MACROS.
*         S. H. KEYSER.     08/15/73.
*         ADAPTED FROM G. R. MANSFIELDS CPCOM.
 COMMCIO  SPACE  4
***       FILE ACTION MACROS FORMAT I/O FUNCTION REQUESTS INTO
*         X REGISTERS AND RETURN JUMP TO PROCESSING SUBROUTINE CIO=.
*         THIS ENTRY POINT IS CONTAINED IN THE COMMON DECK -COMCCIO-.
 COMMCIO  SPACE  4
***     FUNCTION PARAMETERS   DESCRIPTION OF REQUEST.
*
*        BKSP    FILE,RCL     BACKSPACE 1 LOGICAL RECORD.
*        BKSPRU  FILE,CNT,RCL BACKSPACE PHYSICAL RECORDS.
*        CLOSE   FILE,TYP,RCL CLOSE FILE.
*        EVICT   FILE,RCL     RELEASE FILE SPACE.
*        OPEN    FILE,TYP,RCL OPEN FILE.
*        READ    FILE,RCL     READ FILE TO CIO BUFFER.
*        READAL  FILE,RCL     READALL. (65 WORD PRU S)
*        READER  FILE,RCL     READ TO END OF RECORD.(NUCC MUX FILE ONLY)
*        READIF  FILE,RCL     READ IF DATA READY.   (NUCC MUX FILE ONLY)
*        READN   FILE,RCL     READ NON-STOP. (S OR L TAPES ONLY)
*        READNS  FILE,RCL     READ NON-STOP. (MASS STORAGE ONLY)
*        READSK  FILE,RCL     READ SKIP.
*        RETURN  FILE,RCL     RETURN FILE TO SYSTEM.
*        REWIND  FILE,RCL     SKIP TO BEGINNING OF INFORMATION.
*        REWRT   FILE,RCL     REWRITE RECORD IN PLACE.
*        REWRTF  FILE,RCL     REWRITE END OF FILE IN PLACE.
*        REWRTR  FILE,RCL,LVL REWRITE END OF RECORD IN PLACE.
*        RPHR    FILE,RCL     READ PHYSICAL RECORD TO CIO BUFFER.
*        SKIPB   FILE,CNT,RCL SKIP RECORDS BACKWARD.
*        SKIPEI  FILE,RCL     SKIP TO END OF INFORMATION.
*        SKIPF   FILE,CNT,RCL SKIP RECORDS FORWARD.
*        SKIPFB  FILE,CNT,RCL SKIP FILES BACKWARD.
*        SKIPFF  FILE,CNT,RCL SKIP FILES FORWARD.
*        UNLOAD  FILE,RCL     UNLOAD FILE FROM JOB.
*        WPHR    FILE,RCL     WRITE 1 PHYSICAL RECORD FROM CIO BUFFER.
*        WRITAL  FILE,RCL     WRITEALL. (65 WORD PRU S)
*        WRITE   FILE,RCL     WRITE DATA FROM CIO BUFFER.
*        WRITEF  FILE,RCL     WRITE END OF FILE.
*        WRITEN  FILE,RCL     WRITE NON-STOP. (S OR L TAPES ONLY)
*        WRITER  FILE,RCL,LVL WRITE END OF RECORD.
*
 COMMCIO  SPACE  4,10D
***       FOLLOWING PARAMETERS APPLY TO ALL I/O FUNCTION REQUESTS:
*
*                FILE - THE PARAMETER *FILE* IS THE ADDRESS OF THE FET
*                       FOR THE FILE TO WHICH THE OPERATION IS REQUESTED.
*
*                RCL - IF THE PARAMETER *RCL* IS PRESENT IN THE MACRO
*                      CALL, (IE., NOT NULL), CONTROL WILL BE RETURNED
*                      TO THE USER PROGRAM WHEN THE REQUESTED FUNCTION
*                      IS COMPLETE.  IF *RCL* IS NOT SPECIFIED, CONTROL
*                      WILL BE RETURNED WHEN OPERATION IS REQUESTED.
*                LVL - LEVEL NUMBER OF RECORD TO BE WRITTEN.
 COMMCIO  SPACE  4
***       ADDITIONAL PARAMETERS APPLY TO CERTAIN I/O FUNCTION REQUESTS:
*
*                CNT - THE PARAMETER *CNT* ON SPECIFIED POSITIONING
*                      FUNCTIONS IS THE COUNT OF RECORDS OR FILES TO
*                      BE SKIPPED.  A VALUE OF -0 IS CONSIDERED INFINIT.
*
*                TYP - THE PARAMETER *TYP* IS SPECIFIED FOR OPEN AND
*                      CLOSE FUNCTIONS.  SEE SPECIAL NOTES BELOW.
*
 COMMCIO  SPACE  4
*****     CIO - INITIATE FILE ACTION.
*
 CIO      MACRO  FILE,CODE,RCL,IMMD,LAB,SCNT,SIMD
*
*         THE CIO MACRO FORMATS INTO REGISTERS A CALL TO CIO= TO
*         INITIATE A FILE ACTION.  THE SPECIFIC ACTION DEPENDS ON THE
*         VALUE OF THE -CODE- PARAMETER.
*
*         PARAMETERS:
*                FILE = ADDRESS OF THE FET FOR THE FILE TO WHICH THE
*                       ACTION IS DIRECTED.
*                CODE = CIO CODE SPECIFYING THE FILE ACTION.  DEPENDING
*                       ON THE PRESENCE OR ABSENCE OF THE PARAMETER
*                       -IMMD-, -CODE- IS EITHER THE ADDRESS OF A CIO
*                       ACTION CODE OR AN ABSOLUTE EXPRESSION.  IF AN
*                       ABSOLUTE EXPRESSION, THEN CODE<400000B.
*                RCL  = IF PRESENT, AUTO-RECALL IS REQUESTED.
*                IMMD = IF PRESENT, -CODE- IS THE ADDRESS OF A WORD
*                       CONTAINING THE CIO ACTION CODE, RATHER THAN
*                       THE CODE ITSELF.
*                LAB  = IF PRESENT, BIT 35 OF THE RA+1 CALL TO CIO IS
*                       SET.  IF EQUIPMENT ASSIGNMENT IS NEEDED, PP
*                       ROUTINE 2SU WILL USE THE FIVE WORD (OR LESS)
*                       MESSAGE IN RA+70B RATHER THAN CONSTRUCT ITS
*                       OWN.  (USED FOR TAPE OPEN REQUESTS.)
*                SCNT = IF PRESENT, A SKIP COUNT FOR THE FILE
*                       POSITIONING CIO CODES.  IF PARAMETER -SIMD-
*                       IS OMMITTED, THEN SCNT<400000B.
*                       IF ABSENT, NO SKIP COUNT IS INCLUDED.
*                SIMD = IF ABSENT, PARAMETER -SCNT- IS THE ACTUAL SKIP
*                       COUNT OR AN INCREMENT REGISTER EXPRESSION
*                       YEILDING A SKIP COUNT.
*                       IF PRESENT, -SCNT- IS THE ADDRESS OF A WORD
*                       CONTIANING A SKIP COUNT AS 42/0,18/COUNT.
*
*         USES   A1, X1, X2, A6, X6, X7.
*
*         CALLS  CIO=, WNB=, SYS=.
*****
          R=     X2,;A
          IFC    NE,  ;E ,3
          MX7    1
          LX7    35D+59D-58D
          BX2    X2+X7
          IFC    NE, ;F  ,10D
          IFC    EQ, ;G  ,6
          IFNG   ;F,3
          MX1    18D
          LX1    18D
          SKIP   3
          R=     X1,;F
          SKIP   1
          R=     A1,;F
          LX1    18D
          BX2    X1+X2
          IFC    NE,  ;D ,6
          R=     A1,;B
          IFC    NE,  ;C ,2
          BX7    -X1
          SKIP   1
          BX7    X1
          SKIP   4
          IFC    NE,  ;C ,2
          SX7    -;B
          SKIP   1
          SX7    ;B
          RJ     =XCIO=
          ENDM
 COMMCIO  SPACE  4,10
***       SCOPE OPEN AND CLOSE REQUESTS.
*
 COMMCIO  SPACE  4
***       OPEN   FILE,TYP,RCL
*
*         FOR OPEN THE *TYP* PARAMETER MAY BE:
*
*               (NULL)       NR
*                READ        READNR
*                REEL        REELNR
*                ALTER       ALTERNR
*                WRITE       WRITENR
*
 COMMCIO  SPACE  4
***       CLOSE  FILE,TYP,RCL
*
*         FOR CLOSE THE *TYP* PARAMETER MAY BE:
*
*               (NULL)
*                NR
*                UNLOAD
*                RETURN
*
 COMMCIO  SPACE  4
***               THE OPEN FUNCTION IS A FILE INITIALIZATION AND
*         STATUS CHECKING OPERATION.  THE USER NEED NOT ISSUE
*         AN OPEN UNLESS THE FOLLOWING PROCESSING IS DESIRED:
*
*                FOR ALL FILES, PRU SIZE AND RB SIZE IS RETURNED
*         TO THE FET FIELDS.  DEVICE TYPE AND DISPOSITION FIELDS
*         ARE SET BY THE SYSTEM ON ALL FILE ACTION REQUESTS.
*
*                FOR RANDOM FILES, THE SCOPE INDEX IS READ IF THE
*         R BIT IS SET AND AN INDEX BUFFER IS SPECIFIED.  IF THE
*         FILE IS NOT RANDOM, THE R BIT IS CLEARED.
*
*                FOR SEQUENTIAL FILES, FILE POSITION WILL BE SET
*         TO BEGINNING OF INFORMATION UNLESS NO REWIND IS SPECIFIED
*         BY USING THE *TYP* PARAMETER WITH AN *NR* SUFFIX.
*
*                FOR LABELED TAPE FILES, LABEL PROCESSING IS PERFORMED.
*         IF SPECIFIED LABEL INFORMATION IN THE FET MATCHES THE LABEL
*         ON THE TAPE, THE HDR1 LABEL IS RETURNED TO THE FET AND
*         WRITTEN INTO THE CIO BUFFER PROVIDED SUFFICIENT ROOM
*         IS AVAILABLE IN THE BUFFER (8 WORDS).
*
*                FOR THE *WRITE* AND *WRITENR* TYPE OPENS, THE
*         LAST CODE AND STATUS IN THE SYSTEM FNT IS SET TO WRITE.
*         THIS ENSURES THE CIO BUFFER WILL BE FLUSHED IF THE
*         JOB TERMINATES ABNORMALLY BEFORE BUFFER CONTENTS HAVE
*         BEEN TRANSFERED TO AN OUTPUT DEVICE.
*
 COMMCIO  SPACE  4,26
***               THE CLOSE FUNCTION IS A FILE TERMINATION OPERATION.
*         THE USER MUST EMPTY THE CIO BUFFER WHEN FILES ARE BEING
*         WRITTEN, CLOSE DOES NOT FLUSH THE BUFFER.  A CLOSE SHOULD
*         BE ISSUED WHEN THE FOLLOWING PROCESSING IS DESIRED:
*
*                IF THE *TYP* PARAMETER IS NULL, UNLOAD OR RETURN,
*         SEQUENTIAL FILES WILL BE REWOUND.  NR SPECIFIES THAT THE
*         FILE IS NOT TO BE REWOUND.
*
*                FOR LABELED TAPE FILES, AN EOF TRAILER LABEL IS
*         WRITTEN IF THE LAST OPERATION WAS A WRITE.
*
*                FOR UNLABELED TAPE FILES, AN EOF/EOI IS WRITTEN
*         IF THE LAST OPERATION WAS A WRITE.
*
*                FOR ALL TAPE FILES, THE TAPE IS UNLOADED UNLESS
*         *NR* IS SPECIFIED IN WHICH CASE THE FILE IS REPOSITIONED
*         BEFORE THE EOF/EOI IF ONE WAS WRITTEN.  A CLOSE/RETURN
*         DECREMENTS THE TAPE RESERVATION COUNT FOR THE JOB, A
*         CLOSE/UNLOAD DOES NOT DECREASE THIS VALUE.
*
*                FOR RANDOM FILES, A SCOPE INDEX IS WRITTEN IF THE
*         FET R BIT IS SET, AN INDEX BUFFER IS SPECIFIED AND THE
*         FILE CONTENTS HAVE BEEN ALTERED SINCE THE LAST OPEN.
*
 OPEN     SPACE  4,14D
 OPEN     MACRO  F,T,L
          R=  X2,;A
          ECHO   7,B=(,NR,ALTER,ALTERNR,READ,READNR,REEL,REELNR,WRITE,WR
,ITENR),C=(160B,120B,160B,120B,140B,100B,340B,300B,144B,104B)
          IFC    EQ, B ;B ,6
          STOPDUP
          IFC    EQ,  ;C ,2
          SX7    C
          SKIP   1
          SX7    -C
          SKIP   1
 OPEN     ERR    ILLEGAL TYPE SPECIFIED - ;B
          RJ     =XCIO=
          ENDM
 CLOSE    SPACE  4,13D
 CLOSE    MACRO  F,T,L
          R=  X2,;A
          ECHO   7,B=(,NR,RETURN,UNLOAD),C=(150B,130B,174B,170B)
          IFC    EQ, B ;B ,6
          STOPDUP
          IFC    EQ,  ;C ,2
          SX7    C
          SKIP   1
          SX7    -C
          SKIP   1
 CLOSE    ERR    ILLEGAL TYPE SPECIFIED - ;B
          RJ     =XCIO=
          ENDM
 POSMF    SPACE  4,10
***       POSMF - POSITION MULTI-FILE TAPE.
*
*
*         POSMF  FILE,RCL
*
*         THE MULTI-FILE SET, *FILE* IS POSITIONED TO THE MEMBER FILE
*         SPECIFIED BY THE FILE POSITION NUMBER IN THE FET LABEL
*         EXTENSION.  IF THE POSITION NUMBER IS BLANK, THE SET WILL BE
*         POSITIONED AT THE NEXT MEMBER FILE.  IF THE POSITION NUMBER
```

```
*         IS 999, THEN THE SET WILL BE POSITIONED AFTER THE LAST MEMBER
*         OF THE SET.  IF THE POSITION NUMBER IS GREATER THAN THAT OF
*         THE LAST MEMBER, END-OF-SET STATUS WILL BE RETURNED.


 POSMF    MACRO  F,L
*
          R=  X2,F
*
          IFC    NE,  L ,2
          SX7    -110B
          SKIP   1
          SX7    110B
*
          RJ     =XCIO=
*
 POSMF    ENDM
 COMMCIO  SPACE  4
**                COMPRESSED MACRO TEXT FOLLOWS.  FOR DETAILED
*         EXPANSIONS, ASSEMBLE WITH LIST OPTIONS *LO=EFX*.
 TEXT     SPACE  4,21D
          ECHO   19D,LLLLLL=(BKSPRU,SKIPB ,SKIPF ,SKIPFB,SKIPFF),O=(44B,
,640B,240B,740640B/2,740240B/2),X=(**,**,**,  ,  )
 ;A       SPACE  4,18D
 LLLLLL   MACRO  F,N,L
          R=  X2,F
          IF     -REG,N,1
          IFGT   N,,3
          R=  X1,N
          LX1    18D
          SKIP   3
          IFMI   N,3
          MX1    18D
          LX1    36D
          BX2    X1+X2
          IFC    EQ,  L ,2
          SX7    ;B
          SKIP   1
          SX7    -;B
;C        LX7    1
          RJ     =XCIO=
          ENDM
 SKIPEI   SPACE  4,3
 SKIPEI   MACRO  F,R
          SKIPF  ;A,-0,;B
          ENDM
 RPHR     SPACE  4,8D
 RPHR     MACRO  F,L
          R=  X2,;A
          IFC    EQ,  ;B ,2
          BX7    X7-X7
          SKIP   1
          BX7    -X7-X7
          RJ     =XCIO=
          ENDM
 TEXT     SPACE  4,12D
          ECHO   9D,LLLLLL=(BKSP  ,EVICT ,READ  ,READAL,READER,READIF,RE
,ADN ,READNS,READSK,RETURN,REWIND,REWRT ,REWRTF,UNLOAD,WPHR  ,WRITAL,WRI
,TE ,WRITEF,WRITEN),Y=(40B,114B,10B,510B,200B,204B,260B,250B,20B,174B,50
,B,214B,234B,60B,4,514B,14B,34B,264B)
 ;A       SPACE  3,8D
 LLLLLL   MACRO  F,L
          R=  X2,F
          IFC    EQ,  L ,2
          SX7    ;B
          SKIP   1
          SX7    -;B
          RJ     =XCIO=
          ENDM
          ECHO   23D,LLLLLL=(REWRTR,WRITER),Y=(224B,24B)
 LLLLLL   SPACE  4,10
 LLLLLL   MACRO  F,T,V
          IFC    EQ, T  ,2
          SX7    ;B
          SKIP   1
          SX7    -;B
          IFC    NE, V  ,13D
          IF     -REG,V,7
          SX1    V
          LX1    14D
          IFC    EQ, T  ,2
          BX7    X7+X1
          SKIP   1
          BX7    X7-X1
          SKIP   5
          L!V    14D
          IFC    EQ, T  ,2
          BX7    X7+V
          SKIP   1
          BX7    X7-V
          R=  X2,F
          RJ     =XCIO=
          ENDM
 COMMCIO  SPACE  4
          ENDX
COMMDTR
COMMON
 COMMDTR  CTEXT  COMMDTR - DATA TRANSFER MACROS.
 COMMDTR  SPACE  4
***       COMMDTR - DATA TRANSFER MACROS.
*         S. H. KEYSER.   08/17/73.
*         ADAPTED FROM G. R. MANSFIELDS CPCOM.
 COMMDTR  SPACE  4
***               DATA TRANSFER MACROS FORMAT REQUESTS INTO REGISTERS
*         AND RETURN JUMP TO SUBROUTINES FOR PROCESSING.  PROCESSING
*         CONSISTS OF MOVING DATA BETWEEN THE CIO BUFFER FOR A FILE
*         AND A WORKING BUFFER AREA FOR THE PROGRAM.
 COMMDTR  SPACE  4
***               CONVENTIONALLY, THE WORKING BUFFER FOR CODED DATA HOLDS
*         ONE UNIT RECORD FOR PROCESSING, (EG, A SINGLE CARD OR LINE
*         IMAGE).  FOR BINARY DATA, THE SIZE OF THE WORKING BUFFER
*         MAY BE DICTATED BY OTHER CONCERNS, GENERALLY DEPENDANT
*         ON THE DATA STRUCTURE.
 COMMDTR  SPACE  4
***               PROCESSING SUBROUTINES MAINTAIN IN AND OUT CIO BUFFER
*         POINTERS AND AUTOMATICALLY ISSUE I/O FUNCTION REQUESTS AS
*         NEEDED TO FILL OR EMPTY THE BUFFER.  USER PROGRAMMING IS
*         THUS SIMPLIFIED, AND SPECIAL CONCERN NEED ONLY BE GIVEN TO
*         THE INITIATION AND TERMINATION OF THE I/O TASK.
 COMMDTR  SPACE  4
***               ON INPUT, THE USER SHOULD INITIATE FILE ACTION BY
*         ISSUEING A READ REQUEST BEFORE DATA TRANSFER MACROS WILL
*         BE CALLED.  IN ADDITION, THE USER MUST RESTART FILE ACTION IF
*         DESIRED AFTER EOR/EOF HAS BEEN DETECTED.  NO SPECIAL CONCERN
*         NEED BE GIVEN TO THE TERMINATION OF INPUT.
*
 COMMDTR  SPACE  4
***               ON OUTPUT, THE USER IS RESPONSIBLE FOR FLUSHING THE
*         BUFFER AFTER ALL TRANSFERS HAVE BEEN MADE.  THIS IS GENERALLY
*         DONE WITH A WRITER FILE ACTION REQUEST.  SCOPE BUFFER FLUSHING
*         SET UP WITH AN OPEN/WRITE SHOULD NOT BE RELIED UPON UNDER
*         NORMAL TERMINATION CIRCUMSTANCES.
*
 COMMDTR  SPACE  4
***               MACROS WHICH READ DATA FROM A FILE RETURN STATUS
*         INFORMATION AS FOLLOWS:
*
*                 (X1) = +0, TRANSFER COMPLETE.
*                 (X1) = -1, EOF DETECTED ON FILE.
*                 (X1) > 0, EOR DETECTED ON FILE BEFORE TRANSFER WAS
*                           COMPLETED.  (X1) = LWA+1 TRANSFERED INTO
*                           BUFFER.
*
 COMMDTR  SPACE  4,20
***      FUNCTION PARAMETERS  DESCRIPTION OF REQUEST.
*
*         EXAMC  FILE,BUF,N  EXAMINE -C- FORMAT LINE.
*         EXAMO  FILE        EXAMINE ONE WORD.
*         EXAMW  FILE,BUF,N  EXAMINE WORDS.
*        FILLW   FILE,N      WRITE FILL WORDS.
*         READC  FILE,BUF,N  READ LINE IN -C- FORMAT.
*         READH  FILE,BUF,N  READ LINE IN -H- FORMAT.
*         READI  FILE,BUF,N  READ LINE INTERACTIVELY.
*         READM  FILE,TBL,N  READ AND APPEND WORDS TO MANAGED TABLE.
*         READO  FILE        READ ONE WORD INTO X6.
*         READS  FILE,BUF,N  READ LINE INTO STRING BUFFER.
*         READW  FILE,BUF,N  READ WORDS.
*         SKIPR  FILE        SKIP REST OF CURRENT RECORD.
*         SKIPW  FILE,N      SKIP WORDS.
*         WRITEC  FILE,BUF    WRITE LINE IN -C- FORMAT.
*         WRITEH  FILE,BUF,N  WRITE LINE IN -H- FORMAT.
*         WRITEO  FILE        WRITE ONE WORD FROM X6.
*         WRITES  FILE,BUF,N  WRITE LINE FROM STRING BUFFER.
*         WRITEW  FILE,BUF,N  WRITE WORDS.
*
 COMMDTR  SPACE  4
***      COMMDTR PARAMETERS:
*
*         FILE   ADDRESS OF THE FET FOR THE FILE TO WHICH THE TRANSFER
*                IS REQUESTED.
*
*         BUF    ADDRESS OF THE USER WORKING BUFFER.
*
*         N      WORD COUNT OF THE WORKING BUFFER.
*
*         TBL    MANAGED TABLE NAME OR REGISTER EXPRESSION
*                THE TABLE NUMBER.
 EXAMC    SPACE  4,10
**        EXAMC - EXAMINE -C- FORMAT LINE.
*
*                 EXAMC SAMPLES THE NEXT AVAILABLE -C- FORMAT LINE IN
*         THE CIRCULAR BUFFER WITHOUT ADVANCING THE OUT POINTER.


 EXAMC    MACRO  FN,FW,WC
          R=  B6,;B
          R=  B7,;C
          R=  X2,;A
          RJ     =XEXC=
 EXAMC    ENDM
 EXAMO    SPACE  4,10
**        EXAMO - EXAMINE ONE WORD.
*
*                 EXAMO SAMPLES THE NEXT AVAILABLE WORD IN THE CIRCULAR
*         BUFFER WITHOUT ADVANCING THE OUT POINTER.


 EXAMO    MACRO  FN
          R=  X2,;A
          RJ     =XEXO=
 EXAMO    ENDM
 EXAMW    SPACE  4,10
**        EXAMW - EXAMINE WORDS.
*
*                 EXAMW SAMPLES THE NEXT AVAILABLE N WORDS IN THE
*         CIRCULAR BUFFER WITHOUT ADVANCING THE OUT POINTER.


 EXAMW    MACRO  FN,FW,WC
          R=  B6,;B
          R=  B7,;C
          R=  X2,;A
          RJ     =XEXW=
 EXAMW    ENDM
 FILLW    SPACE  4,10
**        FILLW - WRITE FILL WORDS.


 FILLW    MACRO  FN,WC
          R=  B6,;B
          R=  X2,;A
          RJ     =XWFW=
 FILLW    ENDM
 READI    SPACE  4,10
**        READI - READ LINE INTERACTIVELY.


 READI    MACRO  FN,FW,WC
          R=  B6,;B
          R=  B7,;C
          R=  X2,;A
          RJ     =XRDI=
 READI    ENDM
 READM    SPACE  4,12
**        READM - READ AND APPEND WORDS TO MANAGED TABLE.


 READM    MACRO  F,T,W
          R=  B6,;C
          IF     REG,;B,2
          R=  B7,;B
          SKIP   1
          R=  B7,N.;B
          R=  X2,;A
          RJ     =XRDM=
 READM    ENDM
 READO    SPACE  4,10
**        READO - READ ONE WORD.


 READO    MACRO  F
          R=  X2,;A
          RJ     =XROW=
 READO    ENDM
```

```
 SKIPR    SPACE  4,7
**        SKIPR - SKIP REST OF CURRENT RECORD.


 SKIPR    MACRO  F
          R=  X2,;A
          RJ     =XSKR=
 SKIPR    ENDM
 COMMDTR  SPACE  4
**        SKIPW - SKIP WORDS.


 SKIPW    MACRO  A,B
          R=  B6,B
          R=  X2,A
          RJ     =XSKW=
          ENDM
 WRITEO   SPACE  4,10
**        WRITEO - WRITE ONE WORD.


 WRITEO   MACRO  F
          SA1    ;A+2
          RJ     =XWTO=
 WRITEO   ENDM
 TEXT     SPACE  4
**              COMPRESSED MACRO TEXT FOLLOWS.  FOR DETAILED
*         EXPANSIONS, ASSEMBLE WITH LIST OPTIONS *EFX*.
 TEXT     SPACE  4,16D
          ECHO    9D,A=(READ,WRITE),B=(RD,WT),CC=(;A,;A),D=( , ),E=( ,),
,F=( ,*)
 ;A;C;E   SPACE  4,8D
          ECHO   7,P=(C,H,S,W),Q=(;F, , , )
 ;A;C;E   SPACE  4,6
 ;A;C;E   MACRO  X,Y,Z
          R=  B6,Y
Q         R=  B7,Z
          R=  X2,X
          RJ     =X;B;C=
          ENDM
 COMMDTR  SPACE  4
 COMMDTR  ENDX
COMMJCM
COMMON
 COMMJCM  CTEXT  COMMJCM - JOB CONTROL MANAGER MACRO DEFINITIONS.
          SPACE  4
***       COMMJCM - JOB CONTROL MANAGER MACRO DEFINITIONS.
*         D. L. MAUSNER.  05/15/76.  NUCC.
          SPACE  4
***             *COMMJCM* CONTAINS DEFINITIONS OF MACROS  FOR USE WITH
*         *COMCJCM*, THE JOB CONTROL MANAGER INTERFACE.
*
*               THESE MACROS PLACE THEIR ARGUMENTS INTO REGISTERS  AND
*         RETURN-JUMP TO *JCM=*, WHICH FORMATS THE REQUIRED PPU STATUS.
*         UPON RETURN, REGISTERS CONTAIN STATUS INFORMATION AS FOLLOWS:
*
*               (X1) = CONTENTS OF STATUS WORD 1.
*               (X2) = CONTENTS OF STATUS WORD 2.
*               (X6) = 0, IF NO ERROR.
*                    = ERROR MESSAGE ADDRESS, OTHERWISE.
 INTL     SPACE  4,10
**        INTERNAL MACRO DEFINITIONS.
*
*
*         FOR EACH STATUS FORMAT, A *JCM=X* MACRO IS DEFINED.
*
*               X          FORMAT
*               -          ------
*               A          FUNCTION CODE.
*               B          ADDRESS/FUNCTION.
*               C          FILENAME/FIELD LENGTH/POSITION/FUNCTION.
 JCM=A    SPACE  4,10
**        JCM=A - STATUS FORMAT [A].


 JCM=A    MACRO  FN
          R=  X7,FN
          RJ     =XJCM=
          ENDM
 JCM=B    SPACE  4,10
**        JCM=B - STATUS FORMAT [B].


 JCM=B    MACRO  FN,BF,FL
          R=  X1,BF
          IFC    NE,,FL,X2,,1
          R=  A2,FL
          R=  X7,FN
          LX1    30D
          LX2    30D
          RJ     =XJCM=
          ENDM
 JCM=C    SPACE  4,10
**        JCM=C - STATUS FORMAT [C].


 JCM=C    MACRO  FN,LF,PS
          IFC    NE,,LF,X1,,1
          R=  A1,LF
          IFC    NE,,PS,X2,,1
          R=  A2,PS
          R=  X7,FN
          RJ     =XJCM=
          ENDM
          EJECT
 JCREAD   SPACE  4,10
***       JCREAD - READ CONTROL STATEMENT.
*
*
*         JCREAD BF
*         ENTRY  *BF* = BUFFER ADDRESS (ALLOW 8 WORDS).
*         EXIT   (X1) BIT 04 = 0, IF READ COMPLETE.
*                            = 1, OTHERWISE.


 JCREAD   MACRO  BF
          JCM=B  10B,BF
          ENDM
 JCENST   SPACE  4,10
***       JCENST - ENTER CONTROL STATEMENT.
*
*
*         JCENST BF
*         ENTRY  *BF* = BUFFER ADDRESS (AT MOST 8 WORDS).


 JCENST   MACRO  BF
          JCM=B  20B,BF
          ENDM
 JCEXST   SPACE  4,10
***       JCEXST - EXECUTE CONTROL STATEMENT.
*
*
*         JCEXST BF,FL
*         ENTRY  *BF* = BUFFER ADDRESS (AT MOST 8 WORDS).
*                *FL* = ADDRESS OF FIELD LENGTH REQUEST, >= 0.
*                     = *X2* IF (X2) = FIELD LENGTH REQUEST.
*         NOTE   USES ALL REGISTERS.


 JCEXST   MACRO  BF,FL
          JCM=B  30B,BF,FL
          ENDM
 JCBKSP   SPACE  4,10
***       JCBKSP - REVERSE-READ CONTROL STATEMENT.
*
*
*         JCBKSP BF
*         ENTRY  *BF* = BUFFER ADDRESS (ALLOW 8 WORDS).
*         EXIT   (X1) BIT 4 = 0, IF READ COMPLETE.
*                           = 1, OTHERWISE.


 JCBKSP   MACRO  BF
          JCM=B  40B,BF
          ENDM
 JCENPR   SPACE  4,10
***       JCENPR - ENTER CONTROL STATEMENT PROCEDURE.
*
*
*         JCENPR LF,PS
*         ENTRY  *LF* = ADDRESS OF PROCEDURE FILENAME.
*                     = *X1* IF (X1) = FILENAME.
*                *PS* = ADDRESS OF 24-BIT FILE POSITION.
*                     = *X2* IF (X2) = FILE POSITION.


 JCENPR   MACRO  LF,PS
          JCM=C  50B,LF,PS
          ENDM
 JCEXPR   SPACE  4,10
***       JCEXPR - EXECUTE CONTROL STATEMENT PROCEDURE.
*
*
*         JCEXPR LF,PS
*         ENTRY  *LF* = ADDRESS OF PROCEDURE FILENAME.
*                     = *X1* IF (X1) = FILENAME.
*                *PS* = ADDRESS OF 24-BIT FILE POSITION.
*                     = *X2* IF (X2) = FILE POSITION.
*         NOTE   USES ALL REGISTERS.


 JCEXPR   MACRO  LF,PS
          JCM=C  60B,LF,PS
          ENDM
 JCPOSF   SPACE  4,10
***       JCPOSF - POSITION JOB CONTROL FILE.
*
*
*         JCPOSF PS
*         ENTRY  *PS* = ADDRESS OF 24-BIT POSITION.
*                     = *X2* IF (X2) = FILE POSITION.


 JCPOSF   MACRO  PS
          JCM=C  70B,X1,PS
          ENDM
 JCEXIT   SPACE  4,10
***       JCEXIT - EXIT FROM CONTROL STATEMENT PROCEDURE.
*
*
*         JCEXIT
*         ENTRY  NONE.


 JCEXIT   MACRO
          JCM=A  100B
          ENDM
 JCSTAT   SPACE  4,10
***       JCSTAT - OBTAIN JOB CONTROL STATUS.
*
*
*         JCSTAT
*         ENTRY  NONE.


 JCSTAT   MACRO
          JCM=A  110B
          ENDM
          SPACE  4
 COMMJCM  ENDX
COMMCPM
COMMON
 COMMCPM  CTEXT  COMMCPM - CONTROL POINT MANAGER MACROS.
 COMMCPM  SPACE  4
***       COMMCPM - CONTROL POINT MANAGER MACROS.
*         C. G. FILSTEAD.  03/20/74.
*         REVISED -- CGF.  01/21/77.
 COMMCPM  SPACE  4,10
***       COMMCPM CONTAINS MACROS WHICH ARE USED TO COMMUNICATE WITH PP
*         ROUTINE *CPM* VIA *COMCCPM*.
*
*         USES   X - 1, 2, 6, 7.
*                B - NONE.
*                A - 1, 2, 6, 7.
*
*         CALLS  CPM=.
 COMMCPM  SPACE  4,10
***       THE FOLLOWING NOTATIONS ARE USED BELOW:
*
*
*         RECALL             A PARAMETER WHICH, IF NON-NULL, SPECIFIES
*                            THAT AUTO-RECALL WILL BE USED.
*
*         XREG               A PARAMETER  WHICH CAN BE ANY VALID BOOL-
*                            EAN ADDRESS EXPRESSION.
 ^?CPM"DE SPACE  4,10
**        ^?CPM"DE - DISABLE / ENABLE PROCESSING.
*
*
*         ^?CPM"DE F,(P),L
*
*         ENTRY  F = FUNCTION CODE.
*                P = LIST OF MNEMONICS.
```

```
*               L = NON-NULL IF AUTO-RECALL.


 ^?CPM"DE MACRO  FN,P,L
*
         ^?CPM"IR 2,P
         ^?CPM"M1 FN,P,L
         SKIP   2
         ^?CPM"ML (P),(C,D,F,M,R),(10B,1,2,4000B,4)
         ^?CPM"M1 FN,^?CPM"MV,L
*
 ^?CPM"DE ENDM
 ^?CPM"IR SPACE  4,10
**       ^?CPM"IR - TEST FOR REGISTER EXPRESSION.
*
*
*TAG     ^?CPM"IR LC,XP
*
*        SIMILAR TO:
*
*TAG     IF     REG,XP,LC


         MACRO  ^?CPM"IR,T,L,X
*
 ^?CPM"IR MICRO  1,8,*T       *
         IFC    NE,  T ,2
 ^?CPM"LC MICRO
         SKIP   2
 ^?CPM"LC DECMIC L+1
 ^?CPM"LC MICRO  1,, '^?CPM"LC'D
*
 '^?CPM"IR' IF     REG,X,'^?CPM"LC'
 ^?CPM"IR ENDM
 ^?CPM"ML SPACE  4,10
**       ^?CPM"ML - PROCESS MNEMONIC LIST.
*
*
*        ^?CPM"ML (PL),(ML),(VL)
*
*        ENTRY  PL = PARAMETER LIST.
*               ML = MNEMONIC LIST.
*               VL = VALUE LIST.
*
*        EXIT   ^?CPM"MV = THE LOGICAL SUM OF  THOSE ELEMENTS OF -VL-
*               WHICH CORRESPOND TO  THE ELEMENTS OF -ML- WHICH MATCH
*               THE FIRST CHARACTER OF ONE OF THE ELEMENTS OF -PL-.


 ^?CPM"ML MACRO  PL,ML,VL
*
 ^?CPM"MV SET    0
*
         IRP    PL
*
 ^?CPM"AA SET    -1
 ^?CPM"AA MICRO  1,1, PL
*
         ECHO   3,X=(ML),Y=(VL)
         IFC    EQ, X '^?CPM"AA' ,2
 ^?CPM"AA SET    Y
         STOPDUP
*
 ^?CPM"MV SIOR   ^?CPM"MV,^?CPM"AA
*
         IFLT   ^?CPM"AA,0,1
         ERR    ARGUMENT -PL- NOT ONE OF (ML)
*
         IRP
*
 ^?CPM"ML ENDM
 ^?CPM"M1 SPACE  4,10
**       ^?CPM"M1 - CALL CPM, OPTIONALLY CHECKING ARGUMENT.
*
*
*        ^?CPM"M1 F,P,L,LB,UB
*
*        ENTRY  F  = FUNCTION CODE.
*               P  = PARAMETER.
*               L  = NON-NULL IF AUTO-RECALL.
*               LB = LOWER BOUND IF NON-NULL.
*               UB = UPPER BOUND IF NON-NULL.


 ^?CPM"M1 MACRO  F,P,L,LB,UB
*
 ^?CPM"00 ^?CPM"IR ,P
*
         IFC    NE, P X2 ,1
         BX2    P
*
 ^?CPM"00 ELSE
*
         IFC    NE,  LB ,2
         IFLT   P,LB,1
         ERR    ARGUMENT (P) MUST BE # LB.
*
         IFC    NE,  UB ,2
         IFGT   P,UB,1
         ERR    ARGUMENT (P) CANT BE > UB.
*
         IFGE   P,1S17,2
         SA2    =P
         SKIP   1
         R=  X2,P
*
 ^?CPM"00 ENDIF
*
         IFC    NE,  L ,2
         R=  X7,-;AS1
         SKIP   1
         R=  X7,;AS1
*
         RJ     =XCPM=
*
 ^?CPM"M1 ENDM
 ^?CPM"SW SPACE  4,10
**       ^?CPM"SW - PROCESS SENSE SWITCHES.
*
*
*        ^?CPM"SW F,S,L
*
*        ENTRY  F = FUNCTION CODE.
*               S = SENSE SWITCHES.
*               L = NON-NULL IF AUTO-RECALL.


 ^?CPM"SW MACRO  F,S,L
*
         ^?CPM"IR 2,S
         ^?CPM"M1 F,S,L
         SKIP   6
 ^?CPM"AA SBIT   S
 ^?CPM"AB SAND   ^?CPM"AA,777601B
 ^?CPM"AA SET    ^?CPM"AA*1S5
         IFNE   ^?CPM"AB,0,1
         ERR    ONE OF (S) < 1 OR > 6.
         ^?CPM"M1 F,^?CPM"AA,L
*
 ^?CPM"SW ENDM
 ^?CPM"TL SPACE  4,10
**       ^?CPM"TL - SET TIME LIMIT.
*
*
*        ^?CPM"TL F,P,L,D,T1,T2
*
*        ENTRY  F  = FUNCTION CODE.
*               P  = PARAMETER.
*               L  = NON-NULL IF AUTO RECALL.
*               D  = NON-NULL IF NO ABORT.
*               T1 = MNEMONIC 1 (TL OR PTL)
*               T2 = MNEMONIC 2 (CP OR PT)


 ^?CPM"TL MACRO  F,P,L,D,T1,T2
*
 ^?CPM"00 ^?CPM"IR ,P
         IFC    NE, P X2 ,1
         BX2    P
*
 ^?CPM"00 ELSE
*
 ^?CPM"AA SET    0
*
         IFC    NE,  D ,1
 ^?CPM"AA SET    40B
 ^?CPM"AA MICRO  1,, P
*
 ^?CPM"AB MICRO  1,,+P+
 ^?CPM"AB MICCNT ^?CPM"AB
*
         IFC    EQ, T1 '^?CPM"AB' ,3
 ^?CPM"AA SET    ^?CPM"AA+1
 ^?CPM"AA MICRO  2+^?CPM"AB,, P
         SKIP   3
         IFC    EQ, T2 '^?CPM"AB' ,2
 ^?CPM"AA SET    ^?CPM"AA+2
 ^?CPM"AA MICRO  2+^?CPM"AB,, P
*
         R=  X2,'^?CPM"AA'
*
         IFGT   '^?CPM"AA',32767D,1
         ERR    T1 ('^?CPM"AA') MUST BE < 32768D.
*
         IFLT   '^?CPM"AA',0,1
         ERR    T1 ('^?CPM"AA') MUST BE POSITIVE.
*
         IFNE   ^?CPM"AA,0,3
         R=  X7,^?CPM"AA
         LX7    24D
         BX2    X2+X7
*
 ^?CPM"00 ENDIF
*
         ^?CPM"M1 F,X2,L
*
 ^?CPM"TL ENDM
 DISABLE  SPACE  4,10
***      DISABLE - DISABLE OPTIONS.
*
*
*        DISABLE (M1,M2,...,MN),RECALL
*
*        ENTRY  THE FIRST CHARACTER OF EACH  *MI* SPECIFIES AN OPTION
*               TO BE DISABLED.  (SEE BELOW)
*
*
*        DISABLE XREG,RECALL
*
*        ENTRY  *XREG* = BITS SET  CORRESPONDING TO THE OPTIONS WHICH
*               ARE TO BE DISABLED.  (SEE BELOW)


 DISABLE  MACRO  P,L
*
         ^?CPM"DE 11B,(P),L
*
 DISABLE  ENDM
 ENABLE   SPACE  4,10
***      ENABLE - ENABLE OPTIONS.
*
*
*        ENABLE (M1,M2,...,MN),RECALL
*
*        ENTRY  THE FIRST CHARACTER OF EACH  *MI* SPECIFIES AN OPTION
*               TO BE ENABLED.  (SEE BELOW)
*
*
*        ENABLE XREG,RECALL
*
*        ENTRY  *XREG* = BITS SET CORRESPONDING  TO THE OPTIONS WHICH
*               ARE TO BE ENABLED.  (SEE BELOW)


 ENABLE   MACRO  P,L
*
         ^?CPM"DE 12B,(P),L
*
 ENABLE   ENDM
 ENABLE   SPACE  4,10
***      DISABLE / ENABLE OPTIONS.
*
*
* LETTER        BIT         MEANING
*
*   C     XXX XXX XX1 XXX   CONTROL CARD DAYFILE
*   D     XXX XXX XXX XX1   AUTOMATIC DUMPS
*   F     XXX XXX XXX X1X   FIELD LENGTH CHANGE MESSAGES
*   M     1XX XXX XXX XXX   *MAGNET* TAPE PROCESSING
*   R     XXX XXX XXX 1XX   ROLLOUT MESSAGES
 ENDJOB   SPACE  4,10
***      ENDJOB - END THE JOB.
*
*
```

```
*         ENDJOB
*
*         ONLINE JOBS WILL BE LOGGED OUT.   BATCH JOBS WILL BE TERMIN-
*         ATED IMMEDIATELY.


 ENDJOB   MACRO
*
          ^?CPM"M1 17B,0,RECALL
*
 ENDJOB   ENDM
 NONSYS   SPACE  4,10
***       NONSYS - CLEAR SYSTEM OVERLAY LOAD FLAG.
*
*
*         NONSYS RECALL


 NONSYS   MACRO  L
*
          ^?CPM"M1 3,0,L
*
 NONSYS   ENDM
 OFFSW    SPACE  4,10
***       OFFSW - TURN OFF SENSE SWITCHES.
*
*
*         OFFSW  (S1,S2,...,SN),RECALL
*
*         ENTRY  EACH *SI* IS A SWITCH NUMBER, 1 - 6.
*
*
*         OFFSW  XREG,RECALL
*
*         ENTRY  XREG = 1 BIT PER SWITCH:  ... 000 654 321 000 000.


 OFFSW    MACRO  S,L
*
          ^?CPM"SW 5,(S),L
*
 OFFSW    ENDM
 ONSW     SPACE  4,10
***       ONSW - TURN ON SENSE SWITCHES.
*
*
*         ONSW   (S1,S2,...,SN),RECALL
*
*         ENTRY  EACH *SI* IS A SWITCH NUMBER, 1 - 6.
*
*
*         ONSW   XREG,RECALL
*
*         ENTRY  XREG = 1 BIT PER SWITCH:  ... 000 654 321 000 000.


 ONSW     MACRO  S,L
*
          ^?CPM"SW 4,(S),L
*
 ONSW     ENDM
 SETEM    SPACE  4,10
***       SETEM - SET EXIT MODE.
*
*
*         SETEM  MODE,RECALL
*
*         ENTRY  MODE = EXIT MODE, 0 - 7.


 SETEM    MACRO  M,L
*
          ^?CPM"M1 13B,M,L,0,7
*
 SETEM    ENDM
 SETDML   SPACE  4,10
***       SETDML - SET DAYFILE MESSAGE LIMIT.
*
*
*         SETDML LIMIT,RECALL
*
*         ENTRY  LIMIT = MESSAGE LIMIT, 0 - 4095D.


 SETDML   MACRO  M,L
*
          ^?CPM"M1 7,M,L,0,4095D
*
 SETDML   ENDM
 SETLDR   SPACE  4,10
***       SETLDR - SET LOADER CONTROL WORD.
*
*
*         SETLDR ADDR,IM,RECALL
*
*         ENTRY  ADDR = ADDRESS OF CONTROL WORD:  48/DATA, 12/X.
*                IM = NON-NULL IF CONTROL WORD IN X REGISTER *ADDR*.


 SETLDR   MACRO  A,I,L
*
          IFC    EQ,  I ,2
          R=  A2,A
          SKIP   1
          IFC    EQ, X2 A ,4
          LX2    -12D
          MX7    12D
          BX2    -X7*X2
          SKIP   3
          MX2    12D
          L;A    12D
          BX2    -X2*;A
*
          ^?CPM"M1 16B,X2,L
*
 SETLDR   ENDM
 SETMAP   SPACE  4,10
***       SETMAP - SET MAP OPTIONS.
*
*
*         SETMAP OPT,RECALL
*
*         ENTRY  THE FIRST CHARACTER OF *OPT* IS ONE OF THE FOLLOWING:
*
*                CODE   OPT  MEANING
*
*                0      O    NO LOAD MAP.
*                1      F    FULL MAP.
*                2      P    PARTIAL MAP.
*                3      D    DEFAULT MAP OPTION.
*
*
*         SETMAP XREG,RECALL
*
*         ENTRY  XREG = DESIRED MAP CODE.


 SETMAP   MACRO  M,L
*
          ^?CPM"IR 2,M
          ^?CPM"M1 1,M,L
          SKIP   2
          ^?CPM"ML M,(O,F,P,D),(0,1,2,3)
          ^?CPM"M1 1,^?CPM"MV,L
*
 SETMAP   ENDM
 SETMSL   SPACE  4,10
***       SETMSL - SET MASS STORAGE LIMIT.
*
*
*         SETMSL LIMIT,RECALL
*
*         ENTRY  LIMIT = MASS STORAGE LIMIT, 0 - 131071D.
*                        (PRUS / 64)


 SETMSL   MACRO  M,L
*
          ^?CPM"M1 10B,M,L,1,131071D
*
 SETMSL   ENDM
 SETPTL   SPACE  4,10
***       SETPTL - SET PROCESSOR TIME LIMIT.
*
*
*         SETPTL SSSSS,RECALL,D
*
*         ENTRY  SSSSS = NEW PT LIMIT.
*                D = NON-NULL IF NO ABORT ON MAXIMUM EXCEEDED.
*
*
*         SETPTL (PTL+SSSSS),RECALL,D
*
*         ENTRY  SSSSS = SECONDS TO ADD TO THE CURRENT PT LIMIT.
*
*
*         SETPTL (PT+SSSSS),RECALL,D
*
*         ENTRY  SSSSS = SECONDS TO ADD  TO THE CURRENT PROCESSOR TIME
*                TO FORM THE NEW PT LIMIT.
*
*
*         SETPTL XREG,RECALL
*
*         ENTRY  XREG = 30/0,1/D,5/CODE, 24/SSSSS
*                 D = 1, DON-T ABORT IF MAXIMUM EXCEEDED.
*                 CODE = 0, SSSSS
*                        1, PTL+SSSSS
*                        2, PT+SSSSS


 SETPTL   MACRO  P,L,D
*
          ^?CPM"TL 14B,(P),L,D,PTL,PT
*
 SETPTL   ENDM
 SETTL    SPACE  4,10
***       SETTL - SET TIME LIMIT.
*
*
*         SETTL  SSSSS,RECALL,D
*
*         ENTRY  SSSSS = NEW TIME LIMIT.
*                D = NON-NULL IF NO ABORT ON MAXIMUM EXCEEDED.
*
*
*         SETTL  (TL+SSSSS),RECALL,D
*
*         ENTRY  SSSSS = SECONDS TO ADD TO THE CURRENT TIME LIMIT.
*
*
*         SETTL  (CP+SSSSS),RECALL,D
*
*         ENTRY  SSSSS = SECONDS TO ADD TO THE CURRENT CP TIME TO FORM
*                THE NEW TIME LIMIT.
*
*
*         SETTL  XREG,RECALL
*
*         ENTRY  XREG = 30/0,1/D,5/CODE, 24/SSSSS
*                 D = 1, DON-T ABORT IF MAXIMUM EXCEEDED.
*                 CODE = 0, SSSSS
*                        1, TL+SSSSS
*                        2, CP+SSSSS


 SETTL    MACRO  P,L,D
*
          ^?CPM"TL 6,(P),L,D,TL,CP
*
 SETTL    ENDM
 SETWFL   SPACE  4,10
***       SETWFL - SET *WFL* CONTROL WORD.
*
*
*         SETWFL FN,(F1,F2,...,FN),RECALL
*
*         ENTRY  FN = FILE NUMBER, 0 - 777B.
*                FI = CONTROL MNEMONICS.  (SEE BELOW)
*
*
*         SETWFL XREG,,RECALL
*
*         ENTRY  XREG = ... XXX RNE SIA 000 NNN NNN NNN.  (BITS)
*                       NNN...N = FILE NUMBER.
*
*
*         WFL CONTROL FLAGS.
*
*         LETTER  MEANING
*
*          R     REPROCESS BIT DOES NOT STOP WFL CALL.
*          N     WFL TO BE CALLED UPON NORMAL TERMINATION.
*          E     WFL TO BE CALLED UPON ERROR TERMINATION.
```

```
*          S     ALL FLAGS ARE TO BE CLEARED AFTER THE FIRST WFL CALL.
*          I     THE FIRST WFL CALL IS TO BE INHIBITED.
*          A     ADVANCE THE FILE NUMBER AFTER EACH WFL CALL.
*
*
*         IF NO FILE AND FLAGS ARE SPECIFIED,  THEN WFL DUMPS ARE DIS-
*         ABLED.   IF A FILE IS SPECIFIED,  BUT NO FLAGS,  *E* AND *S*
*         ARE ASSUMED.


 SETWFL   MACRO  FN,FL,L
*
 ^?CPM"00 IF     REG,FN
*
          IFC    NE, FN X2 ,1
          BX2    FN
*
          ^?CPM"M1 15B,X2,L
*
 ^?CPM"00 ELSE
*
 ^?CPM"BA SET    FN
*
          IFGE   ^?CPM"BA,0,1
          IFGT   ^?CPM"BA,777B,1
          ERR    ILLEGAL FILE NUMBER - FN
*
          ^?CPM"ML (FL),(A,I,S,E,N,R),(1,2,4,10B,20B,40B)
*
          IFC    NE,  ;A;B ,2
          IFEQ   ^?CPM"MV,0,1
 ^?CPM"MV SET    14B
*
 ^?CPM"MV OCTMIC ^?CPM"MV*1S12+^?CPM"BA
          ^?CPM"M1 15B,'^?CPM"MV'B,L
*
 ^?CPM"00 ENDIF
*
 SETWFL   ENDM
 YOURLIB  SPACE  4,10
***       YOURLIB - SET ALTERNATE LIBRARY OPTIONS.
*
*
*         YOURLIB FNAME,FLAGS,RECALL
*                  X2    X7
*
*         ENTRY  FNAME = ADDRESS OF FILE NAME OR NULL.
*                FLAGS = ONE OF THE FOLLOWING:
*
*                (2)      FS - SATISFY FROM FNAME, THEN SYSTEM.
*                (3)      F  - SATISFY FROM FNAME ONLY.
*                (3)      N  - DO NOT SATISFY FROM SYSTEM.
*                (1)      S  - SATISFY FROM SYSTEM ONLY.
*                (1)      SF - SATISFY FROM SYSTEM, THEN FNAME.
*                (4)      X  - EXTENDED LIBRARY SCAN.
*
*         IF *FLAGS* IS NULL, THEN OPTIONS ARE CLEARED.
*
*
*         YOURLIB FNAME,XREG,RECALL
*
*         ENTRY  XREG = CODE FOR FLAGS. (SEE ABOVE)
*
*
*         ORDER  X - 7, 2.


 YOURLIB  MACRO  FN,P,L
*
 ^?CPM"BA SET    0
*
 ^?CPM"00 ^?CPM"IR ,P
*
          IFC    NE, X7 P ,1
          BX7    P
 ^?CPM"BA SET    -1
*
 ^?CPM"00 ELSE
*
          IRP    P
*
          ECHO   6,X=(SF,FS,F,X,N,S),Y=(1,2,3,4,3,1),Z=(3,3,3,4,3,3)
          IFC    EQ, P X ,5
 ^?CPM"BB SAND   ^?CPM"BA,Z
          ERRNZ  ^?CPM"BB OVERLAPPING SPECIFICATION - P
 ^?CPM"BA SIOR   ^?CPM"BA,Y
          STOPDUP
          SKIP   1
          ERR    UNRECOGNIZED OPTION - P
*
          IRP
*
*
 ^?CPM"00 ENDIF
*
          IFC    EQ,  FN ,2
          BX2    X2-X2
          SKIP   6
          R=  A2,FN
          MX6    -6
          AX2    12D
          BX2    X6*X2
          IFC    EQ,  P ,1
 ^?CPM"BA SET    2
*
*
          IFGE   ^?CPM"BA,0,1
          R=  X7,^?CPM"BA
*
          BX2    X2+X7
          ^?CPM"M1 2,X2,L
*
 YOURLIB  ENDM
 COMMCPM  SPACE  4,1
 COMMCPM  ENDX
COMMSBR
COMMON
 COMMSBR  CTEXT  COMMSBR - SUBROUTINE DEFINITION MACROS.
 COMMSBR  SPACE  4,10
***       COMMSBR - SUBROUTINE DEFINITION MACROS.
*         C. G. FILSTEAD.  03/04/74.
*         D. L. MAUSNER.   05/01/76.
 COMMSBR  SPACE  4,10
***       -COMMSBR- CONTAINS MACROS WHICH FACILITATE THE DEFINITION OF
*         VARIOUS TYPES OF SUBROUTINE ENTRY POINTS.
 QUAL     SPACE  4,10
**        QUAL - DEFINE QUALIFIED SYMBOL BLOCK.
*
*
*QNAME    QUAL   NAME
*
*
*         ENTRY  *NAME* = NAME OF THE BLOCK, OR ASTERISK.
*         EXIT   *QNAME* = A MICRO - 1,,*NAME*


 ^?QAL"CQ IF     -MIC,^?QAL"CQ

 QUAL.    OPSYN  QUAL

          MACRO  QUAL=,MNAME,SYM
*
          IFC    NE, MNAME  ,1
 MNAME    MICRO  1,, '^?QAL"CQ'
*
          IFC    NE, SYM * ,3
 ^?QAL"QS MICRO  1,,/'^?QAL"CQ' '^?QAL"QS'/
 ^?QAL"CQ MICRO  1,, SYM
          SKIP   3
 ^?QAL"CQ MICRO  1,, '^?QAL"QS'
 ^?QAL"CQ MICCNT ^?QAL"CQ
 ^?QAL"QS MICRO  ^?QAL"CQ+2,,/'^?QAL"QS'/
*
          QUAL.  '^?QAL"CQ'
          ENDM

 ^?QAL"QS MICRO
 ^?QAL"CQ MICRO

 QUAL     OPSYN  QUAL=
 ^?QAL"CQ ENDIF
 LJP      SPACE  4
***       LJP - DEFINE ABSOLUTE LONG JUMP INSTRUCTION.
*
*
*         LJP    ADDRESS
*         DEFINE AN ABSOLUTE LONG JUMP INSTRUCTION FOR USE BY OTHER
*         MACROS.
*         ENTRY  (ADDRESS) = JUMP ADDRESS.


          PURGMAC LJP
 LJP      PPOP   5,0100B
 SUBR     SPACE  4,30
***       SUBR - DEFINE SUBROUTINE ENTRY POINT.
*
*
* NAM     SUBR   T,U
*
*
*         ENTRY POINTS MAY BE DEFINED IN CPU OR PPU ASSEMBLY.  THE CAL-
*         LING SEQUENCE IS IDENTICAL.  SINCE NO EXIT SYMBOL IS REQUIRED
*         IN CPU ASSEMBLY, NONE IS DEFINED.
*
*
*         FOR PPU CODING, THREE TYPES OF ENTRY POINT ARE POSSIBLE:
*
* 1.      NAMX   LJM         *
*         NAM    EQU         *-1
*
* 2.             LJM         *
*         NAM    EQU         *-1
*
* 3.      NAM    BSS         0
*
*
*         FOR CPU CODING, TWO TYPES OF ENTRY POINT ARE POSSIBLE:
*
* 1.      NAM    EQ          *+1S17
*         NAMX   EQU         NAM
*
* 2.      NAM    BSS         0
*
*
*         *T* AND *U* FUNCTION AS FOLLOWS:
*
*         T      U           PPU TYPE  CPU TYPE
*
*         -      -           1         1
*         X      -           1 GLOBAL  1 GLOBAL
*         *      -           2         1
*         *      X           2 GLOBAL  1 GLOBAL
*         D      -           3         2
*         D      X           3 GLOBAL  2 GLOBAL
*
*         A GLOBAL SUBROUTINE IS ONE WHOSE NAME IS NOT QUALIFIED.


          PURGMAC SUBR
          MACRO  SUBR,N,T,U
*
          IFC    EQ,$T$D$,2
 N        BSS    0
 .1       SKIP
*
          IFPP   6
          IFC    EQ,$T$*$,2
          LJP    *
          SKIP   1
 N!X      LJP    *
 N        EQU    *-1
          SKIP   2
 N        EQ     *+1S17
 N!X      EQU    *
*
 .1       ENDIF
*
          IF     -DEF,QUAL$,5
          IFC    NE,$T$X$,1
          IFC    EQ,$U$X$,3
 ^?SBR"SQ QUAL
 N        EQU    /'^?SBR"SQ'/N
          QUAL   '^?SBR"SQ'
*
          ENDM

 COMMSBR  ENDX
COMMSYS
COMMON
 COMMSYS  CTEXT  COMMSYS - SYSTEM REQUEST MACROS.
 COMMSYS  SPACE 4
***       COMMSYS - SYSTEM REQUEST MACROS.
*         ADAPTED FROM G. R. MANSFIELDS CPCOM.
*         S. H. KEYSER.  08/14/73.
```

```
 COMMSYS  SPACE  4
***              SYSTEM REQUEST MACROS FORMAT REQUESTS INTO X REGISTERS
*         AND RETURN JUMP TO THE SYSTEM COMMUNICATION SUBROUTINES.  THE
*         COMMON DECK -COMCSYS- CONTAINS THESE SUBROUTINES.
*
*
*                ENTRY       FUNCTION.
*
*                SYS=        PROCESS SYSTEM REQUEST.
*                RCL=        PLACE PROGRAM ON RECALL.
*                WNB=        WAIT NOT BUSY.
*                DFM=        DAYFILE MESSAGE.
*                TIM=        PROCESS TIM FUNCTION.
*                MEM=        PROCESS MEMORY REQUEST.
*
*                TO PROVIDE COMPATABILITY WITH THE MACE VERSION OF
*         CPCOM THE ENTRY POINT MSG= IS AVAILABLE CONTAINED IN THE
*         SEPARATE COMMON DECK -COMCMSG-.
*
 COMMSYS  SPACE  4
***       COMMSYS CODING CONVENTIONS.
*
 COMMSYS  SPACE  4
***              WHERE THE FORMAL PARAMETER *RECALL* APPEARS IN
*         THE MACRO DEFINITION, THE PRESENCE OF A NON-NULL ACTUAL
*         PARAMETER ON THE MACRO CALL SPECIFIES THAT THE REQUESTED
*         OPERATION IS TO BE PERFORMED IN RECALL.
*
          SPACE  4
***              WHERE THE FORMAL PARAMETER *REPLY* APPEARS, THE
*         USER MAY SPECIFY AN ADDRESS FOR THE SYSTEM RESPONSE.
*         IF THE ACTUAL CALL PARAMETER IS NULL, THE RESPONSE IS
*         RETURNED IN REGISTERS ONLY.
*
          EJECT
 ABORT    SPACE  4,14D
*****     ABORT - ABNORMAL PROGRAM TERMINATION.
*
 ABORT    MACRO  MSG,SRC

*                MSG = IF PRESENT, ADDRESS OF DAYFILE MESSAGE.
*                SRC = IF PRESENT, A SOURCE ABORT WILL BE DONE.
*
*         EXIT   CONTROL POINT ERROR FLAG SET (CPU ABORT).
*
*         USES   A1, X1, A6, X6.
*
*         CALLS  DFM=, SYS=.
*****
 ^?ABT"DM MICRO  1,, ;A
 ^?ABT"DM MICCNT ^?ABT"DM
          IFNE   ^?ABT"DM,,1
          DAYFILE (;A),,R
          SX6    4RABTP/16D
          IFC    NE,  ;B ,3
          R=  X1,1
          LX1    -4
          BX6    X1+X6
          LX6    40D
          RJ     =XSYS=
          ENDM
 ATIME    SPACE  4,12D
*****     ATIME - REQUEST ACCUMULATED CPU TIME IN MILLISECONDS.
*
 ATIME    MACRO  REPLY

*         RETURN (X6) = 30/TIME-LIMIT,30/TIME-USED.
*
*         USES   A1, X1, A6.
*
*         NEEDS  CPAREA.
*****
          CPAREA 23B,;A
          ENDM
 CLOCK    SPACE  4,15D
*****     CLOCK - RETURN TIME OF DAY.
*
 CLOCK    MACRO  REPLY

*         EXIT   (X6) = 10H*HH.MM.SS.
*
*         USES   A1, X1, A6.
*
*         CALLS  TIM=.
*****
          R=  X6,2
          RJ     =XTIM=
          IFC    NE,  ;A ,1
          R=  A6,;A
          ENDM
 CORE     SPACE  4,26D
*****     CORE - REQUEST CENTRAL MEMORY OR EXTENDED CORE STORAGE.
*
 CORE     MACRO  FL,TT

*                FL = WORD COUNT.  IF NULL, CURRENT FIELD LENGTH IS
*                     RETURNED.  *X6* MUST BE SPECIFIED IF WORD COUNT
*                     WILL BE > 377777B.
*                TT = REQUEST TYPE, CM OR ECS.  IF NULL, CM IS ASSUMED.
*
*         EXIT   (X6) = CM OR ECS FIELD LENGTH.
*
*         USES   A1, X1, A6.
*
*         CALLS  MEM=.
*****
          R=  X6,;A
          IFC    NE,  ;B ,1
          IFC    EQ, CM ;B ,2
          BX1    X1-X1
          SKIP   4
          IFC    EQ, ECS ;B ,2
          R=  X1,1
          SKIP   1
 CORE     ERR    INVALID REQUEST TYPE - ;B
          RJ     =XMEM=
          ENDM
 COST     SPACE  4,12D
*****     COST - REQUEST ACCUMULATED JOB COST.
*
 COST     MACRO  REPLY

*         RETURN (X6) = JOB COST, IN FLOATING POINT CENTS.
*
*         USES   A1, X1, A6.
*
*         NEEDS  CPAREA.
*****
          CPAREA 173B,;A
          ENDM
 CPAREA   SPACE  4,21D
*****     CPAREA - READ WORD FROM CONTROL POINT AREA.
*
*                CPAREA USES THE MODIFIED NUCC -TIM- FUNCTION FOR
*         ACCESS TO THE USERS CONTROL POINT AREA.  LOCATIONS 20B
*         THRU 177B ARE DEFINED.
*
 CPAREA   MACRO  CPLOC,REPLY

*                CPLOC = LOCATION IN CONTROL POINT TO SAMPLE.
*
*         EXIT   (X6) = WORD FROM CONTROL POINT AREA.
*
*         USES   A1, X1, A6.
*
*         CALLS  TIM=.
*****
          R=  X6,;A
          RJ     =XTIM=
          IFC    NE,  ;B ,1
          R=  A6,;B
          ENDM
 DATE     SPACE  4,15D
*****     DATE - RETURN TODAYS DATE.
*
 DATE     MACRO  REPLY

*         EXIT   (X6) = 10H MM/DD/YY.
*
*         USES   A1, X1, A6.
*
*         CALLS  TIM=.
*****
          R=  X6,1
          RJ     =XTIM=
          IFC    NE,  ;A ,1
          R=  A6,;A
          ENDM
 DAYFILE  SPACE  4,27D
*****     DAYFILE - DISPLAY MESSAGE.
*
 DAYFILE  MACRO  MESSAGE,OPTION,RECALL

*         MESSAGE = ADDRESS OF MESSAGE IN -C- FORMAT.
*         OPTION = DISPLAY OPTION:
*                0 = MESSAGE TO CONTROL POINT AND SYSTEM
*                    DAYFILES, AND TO LINE 1 OF B DISPLAY.
*                1 = MESSAGE TO CONTROL POINT DAYFILE ONLY.
*               10B= MESSAGE TO SYSTEM DAYFILE ONLY.
*              100B= MESSAGE TO B DISPLAY ONLY.
*
*         USES   A1, X1, A6, X6.
*
*         CALLS  DFM=.
*****
          R=  X6,;A
          IFC    EQ,  ;C ,5
          IFEQ   ;B,,2
          BX1    X1-X1
          SKIP   6
          R=  X1,;B
          SKIP   4
          IFEQ   ;B,,2
          BX1    -X1-X1
          SKIP   1
          R=  X1,-;B
          RJ     =XDFM=
          ENDM
 ENDRUN   SPACE  4,14D
*****     ENDRUN - TERMINATE PROGRAM RUN.
*
 ENDRUN   MACRO  MSG

*                MSG = IF PRESENT, ADDRESS OF DAYFILE MESSAGE.
*
*         EXIT   NORMAL CPU TERMINATION.
*
*         USES   A1, X1, A6, X6.
*
*         CALLS  DFM=, SYS=.
*****
 ^?END"DM MICRO  1,, ;A
 ^?END"DM MICCNT ^?END"DM
          IFNE   ^?END"DM,,1
          DAYFILE (;A),,R
          SX6    4RENDP/16D
          LX6    40D
          RJ     =XSYS=
          ENDM
 JDATE    SPACE  4,15D
*****     JDATE - REQUEST TODAYS JULIAN DATE.
*
 JDATE    MACRO  REPLY

*         RETURN (X6) = 5RYYDDD.
*
*         USES   A1, X1, A6.
*
*         CALLS  TIM=.
*****
          SX6    3
          RJ     =XTIM=
          IFC    NE,  ;A ,1
          R=  A6,;A
          ENDM
 JNAME    SPACE  4,12D
*****     JNAME - REQUEST JOB NAME.
*
 JNAME    MACRO  REPLY

*         RETURN (X6) = 42/7HJOBNAME,18/UNDEFINED.
*
*         USES   A1, X1, A6.
*
*         NEEDS  CPAREA.
*****
          CPAREA 21B,;A
          ENDM
 MESSAGE  SPACE  4,10
*****     MESSAGE - DISPLAY MESSAGE.
*
 MESSAGE  MACRO  ADDRESS,OPTION,RECALL

*         ENTRY  *ADDRESS* = ADDRESS OF C-FORMAT MESSAGE.
```

```
*                *OPTION* = DISPLAY OPTION AS FOLLOWS.
*
*                (ABSENT)    CONSOLE AND ALL DAYFILES.
*                C           CONSOLE ONLY.
*                L           LOCAL DAYFILE ONLY.
*                S           SYSTEM DAYFILE ONLY.
*
*         USES   A1, A6, X1, X6.
*
*         CALLS  DFM=.
*****
 ^?DFM"AA SET    0
          R=  X6,;A
*                            (X1) = OPTION.
          ECHO   3,A=(,1,2,3,C,L,S),B=(,1S6,1S6,1,1S6,1,1S3)
          IFC    EQ,,;B,A,,2
 ^?DFM"AA SET    B
          STOPDUP
          R=  X1,^?DFM"AA
*                            RECALL OPTION.
          IFC    NE,,;C,,,1
          BX1    -X1
*                            PROCESS REQUEST.
          RJ     =XDFM=
          ENDM
 PDATE    SPACE  4,10
*****     PDATE - PACK DATE AND TIME.
*
*
 PDATE    MACRO  REPLY
*
*
*         RETURN (X6) = 26/0, 16/EJD, 18/ETM.
*                       EJD = ENCODED JULIAN DATE (SEE COMCEJD)
*                       ETM = ENCODED WALL-CLOCK TIME (SEE COMCETM)
*
*         USES   X - 1, 2, 3, 4, 7.
*                B - NONE.
*                A - 1, 2, 6, 7.
*
*         CALLS  PDT.
*****
          RJ     =XPDT
          IFC    NE,  ;A ,1
          R=  A6,;A
          ENDM
 PPTIME   SPACE  4,12D
*****     PPTIME - REQUEST ACCUMULATED PP TIME USED.
*
 PPTIME   MACRO  REPLY

*         RETURN (X6) = 24/UNDEFINED,36/PPTIME(MSEC).
*
*         USES   A1, X1, A6.
*
*         NEEDS  CPAREA.
*****
          CPAREA 25B,;A
          ENDM
 RECALL   SPACE  4,18D
*****     RECALL - PLACE PROGRAM ON RECALL.
*
 RECALL   MACRO  STATUS

*                STATUS - IF PRESENT, PROGRAM WILL BE RECALLED
*                         WHEN BIT 0 OF (STATUS) IS SET.
*                       - IF NULL, PERIODIC RECALL IS REQUESTED.
*
*         USES   A1, X1, X2, A6, X6.
*
*         CALLS  RCL=, WNB=.
*****
          IFC    EQ,  ;A ,2
          RJ     =XRCL=
          SKIP   2
          R=  X2,;A
          RJ     =XWNB=
          ENDM
 RTIME    SPACE  4,18D
*****     RTIME - REQUEST REAL TIME CLOCK READING.
*
*                READING IS FROM THE INTERNAL SYSTEM CLOCK KEPT BY
*         MONITOR AND IS ELAPSED TIME SINCE DEADSTART.
*
 RTIME    MACRO  REPLY

*         RETURN (X6) = 12/UNDEFINED,12/SEC,12/MSC,24/MSEC.
*
*                SEC = SECONDS.
*                MSC = MILLISECONDS MOD 4096.
*                MSEC = REAL TIME MILLISECONDS.
*
*         USES   A1, X1, A6.
*
*         CALLS  TIM=.
*****
          SX6    4
          RJ     =XTIM=
          IFC    NE,  ;A ,1
          R=  A6,;A
          ENDM
 SYSTEM   SPACE  4,25D
*****     SYSTEM - REQUEST SYSTEM FUNCTION.
*
 SYSTEM   MACRO  REQUEST,RECALL,P1,P2

*                REQUEST = 3 CHARACTER SYSTEM REQUEST.
*                P1 = PARAMETER FOR BITS  0-17 OF REQUEST.
*                P2 = PARAMETER FOR BITS 18-35 OF REQUEST.
*
*         USES   A1, X1, X2, A6, X6.
*
*         CALLS  SYS=.
*****
          SX6    3R;A
          IFC    NE,  ;B ,1
          PX6    X6
          LX6    42D
          IFC    NE,  ;C ,2
          R=  X1,;C
          BX6    X1+X6
          IFC    NE,  ;D ,3
          R=  X2,;D
          LX2    18D
          BX6    X2+X6
          RJ     =XSYS=
          ENDM
 TIME     SPACE  4,15D
*****     TIME - REQUEST ACCUMULATED CPU TIME.
*
 TIME     MACRO  REPLY

*         RETURN (X6) = 24/**,24/SEC,12/MSEC.
*
*         USES   A1, X1, A6.
*
*         CALLS  TIM=.
*****
          BX6    X6-X6
          RJ     =XTIM=
          IFC    NE,  ;A ,1
          R=  A6,;A
          ENDM
 TODAY    SPACE  4,10
*****     TODAY - REQUEST TODAYS DAY-OF-THE-WEEK.
*
 TODAY    MACRO  REPLY,TYPE
*
*         ENTRY  *TYPE* = NULL, SET REPLY TO DAY-OF-WEEK.
*                       = NONNULL, SET REPLY TO FIRST 3 CHAR.
*
*         RETURN (X6) = DAY-OF-WEEK, L-FORMAT.
*                (X7) = FIRST 3 CHAR, L-FORMAT.
*
*         USES   A - 1, 2.
*                B - 2.
*                X - 1, 2, 3, 4.
*
*         CALLS  DOW=, EJD, TIM=.
*****
          JDATE
          RJ     =XDOW=
          IFC    NE,,;A,,,4
          IFC    EQ,,;B,,,2
          SA6    ;A
          SKIP   1
          SA7    ;A
          ENDM
 COMMSYS  SPACE  4
 COMMSYS  ENDX
COMMTXT
COMMON
 COMMTXT  CTEXT  COMMTXT - GENERAL MACRO SUPPLEMENT.
 COMMTXT  SPACE  4
***       COMMTXT - GENERAL MACRO SUPPLEMENT.
*         ADAPTED FROM G. R. MANSFIELDS CPCOM.
 ABS      SPACE  4,10
***       ABS - DEFINE ABSOLUTE ASSEMBLY.
*         GENERATES LOWEST ORIGIN LEAVING SPACE FOR ENTRY LIST.
*
*
*         ABS    ENTR
*
*         ENTRY  *ENTR* = IF SYMBOLIC NAME:
*                            *ENTRY* DECLARATIONS PRECEDE FIRST *ORG*.
*                            REQUIRED TO BEGIN ASSEMBLY:
*                                      IDENT  NAME,ENTR
*                                      ABS    ENTR
*                                      SYSCOM B1
*                            REQUIRED TO SET PROGRAM ORIGIN:
*                                      ORG    ENTR
*                          IF ASTERISK (*):
*                            *ENTRY* DECLARATIONS MAY APPEAR ANYWHERE.
*                            REQUIRED TO BEGIN ASSEMBLY:
*                                      IDENT  NAME,NAME
*                                      ABS
*                                      SYSCOM B1
*                            REQUIRED TO FINISH ASSEMBLY:
*                                      USE    0
*                             NAME     BSS    0
*                                      END
*                          IF NOT SPECIFIED:
*                            *ABS* AND *ENTRY* OPERATE AS DESCRIBED
*                            IN *COMPASS* REFERENCE MANUAL.


 ^?ABS=1  MACRO  S
          ^?ABS=2
 ABS1     IFC    NE,,S,,
 ABS2     IFC    EQ,,S,*,
*                            *ENTRY* ANYWHERE FORMAT.
 ^?SYSCOM RMT
          ORG    ORGR
          USE    'DECKNAME'
          RMT
 ^?ETY=1  MACRO  E
          ^?ETY=2 E
          USE
          BSS    1
          USE    *
 ^?ETY=1  ENDM
 ABS2     ELSE
*                            *ENTRY* BEFORE FIRST *ORG* FORMAT.
 ^?ETY=1  MACRO  E
          ^?ETY=2 E
          IF     -DEF,S,1
 S!!!     SET    ORGR
 S!!!     SET    S+1
 ^?ETY=1  ENDM
 ABS2     ENDIF
*
 ^?ETY=2  OPSYN  ENTRY
 ENTRY    OPSYN  ^?ETY=1
 ABS1     ENDIF
          ENDM

 ^?ABS=2  OPSYN  ABS
 ABS      OPSYN  ^?ABS=1
 IOFCOM   SPACE  4,33
*****     IOFCOM - DEFINE INPUT/OUTPUT FUNCTIONS.
*
 IOFCOM   MACRO
          BASE   M
*
*
*         /CIO/ - BASIC INPUT/OUTPUT FUNCTIONS.
*
          QUAL   CIO
 BKSP     =      40          BACKSPACE 1 LOGICAL RECORD.
 BKSPRU   =      44          BACKSPACE PHYSICAL RECORDS.
 EVICT    =      114         RELEASE FILE SPACE.
 POSMF    =      110         POSITION MULTI-FILE TAPE.
 READ     =      10          READ FILE TO CIRCULAR BUFFER.
 READAL   =      510         READ TO END OF INFORMATION (65 WORD PRUS).
```

```
 READER   =      200         READ TO END OF RECORD (NUCC MUX FILE ONLY).
 READIF   =      204         READ IF DATA READY    (NUCC MUX FILE ONLY).
 READN    =      260         READ NON-STOP (S OR L TAPES ONLY).
 READNS   =      250         READ NON-STOP (MASS STORAGE ONLY).
 READSK   =      20          READ BUFFER AND SKIP TO END OF RECORD.
 RETURN   =      174         RETURN FILE TO SYSTEM.
 REWIND   =      50          SKIP TO BEGINNING OF INFORMATION.
 REWRT    =      214         REWRITE DATA IN PLACE.
 REWRTF   =      234         REWRITE FILE IN PLACE.
 REWRTR   =      224         REWRITE RECORD IN PLACE.
 RPHR     =      0           READ PHYSICAL RECORD TO CIRCULAR BUFFER.
 SKIPB    =      640         SKIP RECORDS BACKWARDS.
 SKIPF    =      240         SKIP RECORDS FORWARDS.
 UNLOAD   =      60          UNLOAD FILE FROM JOB.
 WPHR     =      4           WRITE PHYSICAL RECORD FROM CIRCULAR BUFFER.
 WRITAL   =      514         WRITE DIRECT (65 WORD PRUS).
 WRITE    =      14          WRITE DATA FROM CIRCULAR BUFFER.
 WRITEF   =      34          WRITE END OF FILE.
 WRITEN   =      264         WRITE NON-STOP (S OR L TAPES ONLY).
 WRITER   =      24          WRITE END OF RECORD.
*
*         COMPOSITE FUNCTIONS:
*
*         SKIPEI - SKIP TO END OF INFORMATION.
*
*         USE SKIPF WITH 777777B RECORD COUNT, E.G.:
*
*         SX2    FET
*         MX1    18
*         LX1    18-42
*         BX2    X1+X2
*         SX7    /CIO/SKIPF
*         RJ     =XCIO=
*
*
 SKIPFB   =      740640      SKIP FILES BACKWARDS.
 SKIPFF   =      740240      SKIP FILES FORWARDS.
*
*         (THESE ARE SKIPB/SKIPF WITH 17B RECORD LEVEL)  TO USE:
*
*         SX2    FET
*         SX7    /CIO/SKIPB/2  [OR SKIPF/2]
*         LX7    1
*         RJ     =XCIO=
*
          QUAL   *
*
*
*         /CLOSE/ - FILE TERMINATION FUNCTIONS.
*
          QUAL   CLOSE
 NR       =      130         CLOSE AND POSITION BEFORE WRITTEN EOF/EOI:
*                            SETS FST R BIT IF SET IN FET.
 REEL     =      330         CLOSE REEL, REWIND.
 REELNR   =      350         CLOSE REEL, NO REWIND.
 REELUNL  =      370         CLOSE REEL, UNLOAD.
 RETURN   =      174         CLOSE FILE AND RETURN TO SYSTEM:
*                            DECREMENTS JOB TAPE COUNT.
 REWIND   =      150         CLOSE AND POSITION TO BOI.
 UNLOAD   =      170         CLOSE FILE AND UNLOAD FROM JOB:
*                            DOES NOT REDUCE JOB TAPE COUNT.
          QUAL   *
*
*
*         /OPEN/ - FILE INITIALIZATION FUNCTIONS.
*
          QUAL   OPEN
 ALTER    =      160         OPEN FILE FOR READ/WRITE, POSITION TO BOI.
 ALTERNR  =      120         OPEN FILE FOR READ/WRITE, NO REPOSITIONING.
 READ     =      140         OPEN FILE FOR READ, POSITION TO BOI.
 READNR   =      100         OPEN FILE FOR READ, NO REPOSITIONING.
 REEL     =      340         OPEN REEL, POSITION TO BOI.
 REELNR   =      300         OPEN REEL, NO REPOSITIONING.
 WRITE    =      144         OPEN FILE FOR WRITE, POSITION TO BOI.
 WRITENR  =      104         OPEN FILE FOR WRITE, NO REPOSITIONING.
          QUAL   *
*
          BASE   *
*****
 IOFCOM   ENDM
 SRTCOM   SPACE  4,10
*****     SRTCOM - DEFINE RECORD TYPE SYMBOLS.
*
 SRTCOM   MACRO

*         ENTRY  NONE.
*         EXIT   SYMBOLS OF THE FORM *XXRT*.
*****
 TXRT     =      0           TEXT OR UNRECOGNIZED
 P6RT     =      1           6000-TYPE PPU OVERLAY
 USRT     =      2           USER-DEFINED RECORD TYPE
 RLRT     =      3           RELOCATABLE CPU PROGRAM
 OVRT     =      4           SINGLE-ENTRY CPU OVERLAY
 ULRT     =      5           USER-LIBRARY DIRECTORY
 MDRT     =      6           *MODIFY* DECK
 MCRT     =      7           *MODIFY* COMMON DECK
 PLRT     =      8D          PROGRAM LIBRARY DIRECTORY
 OMRT     =      9D          MULTIPLE-ENTRY CPU OVERLAY
 P7RT     =      10D         7600-TYPE PPU OVERLAY
 OERT     =      11D         MULTIPLE-ENTRY ECS/CPU OVERLAY
          ENDM
 SYSCOM   SPACE  4
*****     SYSCOM - DEFINE SYSTEM COMMUNICATION SYMBOLS.
*
 SYSCOM   MACRO  B1,SL

*                *B1* = *B1* IF (B1) = 1.
*                *SL* = IF NON-BLANK,  A LIST OF SYMBOLS WHICH ARE NOT
*                       TO BE DEFINED.
*****
 .1       IFC    NE,**;B*
*
          MACRO  ^?SYS"EQ,S,V
          ECHO   3,X=(SL)
          IFC    EQ,*S*X*,2
          STOPDUP
          SKIP   1
 S        EQU    V
 ^?SYS"EQ ENDM
*
 =        OPSYN  ^?SYS"EQ
*
 .1       ENDIF
*****

          IFC    EQ, B!1 ;A ,1
          B1=1                USER DEFINES (B1) = 1.
*
*
*         SYSTEM COMMUNICATION SYMBOLS:
*
 ARGR     =      2           FIRST CONTROLCARD ARGUMENT.
 SPPR     =      27B         SPECIAL PROGRAM PARAMETER AREA (27B - 47B)
 ACTR     =      64B         ARGUMENT COUNT.(BITS  0-17)
 PGNR     =      64B         PROGRAM NAME.  (BITS 18-59)
 LWPR     =      65B         LWA+1 ASSIGNED PROGRAM SPACE.  (BITS 0-17)
 CMUR     =      65B         LOCATION OF CMU FLAG.  (BIT 59)
 FWPR     =      66B         FWA ASSIGNED PROGRAM SPACE.  (BITS 0-17)
 XJPR     =      66B         LOCATION OF CEJ FLAG.  (BIT 59)
 LDRR     =      67B         -LDR- COMPLETION.  (BIT 29)
 CCDR     =      70B         CONTROLCARD BUFFER.(8 LOCATIONS)
 ORGR     =      101B        PROGRAM ORIGIN.
 CHAR     =      6           NUMBER OF BITS PER CHARACTER.
 LINO     =      23D         LINES PER TERMINAL PAGE (ONLINE CRT SCREEN)
 LINP     =      60D         LINES PER PRINTER PAGE AT 6 LPI.
 LINQ     =      82D         LINES PER PRINTER PAGE AT 8 LPI.
 WIDO     =      79D               CHARACTER WIDTH PER LINE, ONLINE CRT.
 WIDT     =      72D               CHARACTER WIDTH PER LINE, ONLINE TTY.
 WIDP     =      137D              CHARACTER WIDTH PER LINE, PRINTER.
*
*****


          PURGMAC ^?SYS"EQ
 ^?SYSCOM HERE
          ENDM
 MOVE     SPACE  4,20D
*****     MOVE - MOVE BLOCK OF DATA.
*
 MOVE     MACRO  WC,FR,TO

*                WC = WORD COUNT OF BLOCK TO BE MOVED.
*                FR = FWA OF SOURCE BLOCK.
*                TO = FWA OF DESTINATION.
*                (B1) = 1.
*
*         USES   X - 1, 2, 3, 4, 6, 7.
*                B - 7.
*                A - 2, 4, 6, 7.
*
*         CALLS  MVE=.
*****
          R=  X1,;A
          R=  X2,;B
          R=  X3,;C
          RJ     =XMVE=
          ENDM
 IXX/X,B  SPACE  4
*****     IXI    XJ/XK,BL    DEFINE INTEGER DIVISION PSUEDO-OP.
*
*                DIVIDE XJ BY XK USING BL.  IF BL IS NOT
*         SPECIFIED, B7 WILL BE USED.
*
 IXX/X,B  OPDEF  I,J,K,L

*         USES   XJ, XK, BL.
*****
          PX.;B
          NX.;B
          PX.;C
          NX.;C
          FX.;A   X.;B/X.;C
          UX.;A,B.;D
          LX.;A   B.;D
          ENDM

 IXX/X    OPDEF  I,J,K
          IX.;A   X.;B/X.;C,B7
          ENDM
 OVERLAY  SPACE  4,10
*****     OVERLAY - REQUEST OVERLAY LOAD.
*
 OVERLAY  MACRO  NAM,LEV,SYS,FWA

*                NAM = OVERLAY NAME.
*                LEV = LEVELS.
*                SYS = IF SPECIFIED, LOAD FROM SYSTEM.
*                FWA = IF SPECIFIED, ADDRESS TO LOAD OVERLAY.
*
*         EXIT   (X1) = TRANSFER ADDRESS.
*         CALLS  OVL=.
*****
          IFC    EQ, ;B  ,2
          BX6    X6-X6
          SKIP   1
          R=  X6,;B*1S6
          IFC    NE, ;C  ,3  SET U+V BITS
          MX7    2
          LX7    1
          BX6    X6+X7
          R=  X2,;D
          IFC    NE, ;A X1 ,1
          R=  A1,;A
          RJ     =XOVL=
 OVERLAY  ENDM
 FET      SPACE  4,10
***       FET CREATION MACROS.
*
*                FET CREATION MACROS ALLOW FOR THE GENERATION AND
*         ASSEMBLY TIME INITIALIZATION OF FILE ENVIRONMENT TABLES.
*
*
*         THE GENERAL FORMAT OF THE CALL IS AS FOLLOWS:
* LFN     TYP    BUF,LEN,P1,P2,...,PN
*
*         LFN    FILE NAME.  A COMPASS SYMBOL OF THIS NAME IS DEFINED
*                FOR PROGRAM REFERENCE TO THE FET AND FOR USE WITH
*                THE FILE ACTION AND DATA TRANSFER MACROS.
*
*         TYP    IDENTIFIES THE TYPE AND MODE OF THE FILE:
*
*                FILEB - SEQUENTIAL BINARY.
*                FILEC - SEQUENTIAL CODED.
*                RFILEB - RANDOM BINARY.
*                RFILEC - RANDOM CODED.
*
*         BUF    FWA OF THE CIO BUFFER FOR THIS FILE.
*
*         LEN    LENGTH IN WORDS OF THE CIO BUFFER.
*
*         P1-PN  UP TO 8 OPTIONAL PARAMETERS IN ANY ORDER AS FOLLOWS:
*
*                (EPR)       SETS THE ERROR PROCESSING (EP) BIT.
*
*                (UPR)       SETS THE USER PROCESSING (UP) BIT.
```

```
*
*               (LBL)       SPECIFIES THAT THE FET IS TO BE LARGE
*                           ENOUGH TO HOLD TAPE LABEL INFORMATION.
*
*               (VSN)       SPECIFIES THAT THE FET IS TO BE LARGE
*                           ENOUGH TO HOLD TAPE LABEL INFORMATION
*                           INCLUDING A VSN PARAMETER.  (-LBL- SHOULD
*                           NOT BE SPECIFIED IF -VSN- IS USED.)  THE
*                           LABEL MACRO SHOULD BE USED IN ADDITION TO
*                           THE -LBL- OR -VSN- PARAMETERS ONLY IF THE
*                           FET LABEL FIELDS ARE TO BE INITIALIZED TO
*                           OTHER THAN BINARY ZERO.
*
*                           THE -VSN- SPECIFICATION ALSO MEANS THAT NO
*                           EOR WILL BE WRITTEN ON LEVEL 17 WRITER
*                           REQUESTS.
*
*               (DMP)       SETS THE CHECK-POINT DUMP BIT.
*
*               (EOR)       SPECIFIES THAT NO EOR WILL BE WRITTEN
*                           ON LEVEL 17 WRITER FILE ACTION REQUESTS.
*
*               (FET=N)     SPECIFIES MINIMUM FET LENGTH IN WORDS.
*                           OTHER SPECIFIED OPTIONS AUTOMATICALLY
*                           ADJUST FET LENGTH AS NECESSARY.
*
*               (IND=FWA,LEN)     DEFINES FWA AND LENGTH OF USERS
*                           INDEX BUFFER.  NOTE THIS SPECIFICATION
*                           WILL NOT SET THE RANDOM BIT.
*
*               (WOF=EXT,FET)  DEFINES PRINT/READI CONTROL INFORMATION
*                           AND INTERFACE LINKAGE FOR USE BY WOF/RDI
*                           ROUTINES.  FOR INPUT FILES, *EXT* IS THE
*                           ADDRESS OF THE INTERACTIVE PAGING ROUTINE,
*                           AND *FET* IS THE ADDRESS OF THE OUTPUT FET.
*                           FOR OUTPUT FILES, *EXT* IS THE ADDRESS OF 3
*                           WORD FET EXTENTION CONTROLING ROUTINE WOF
*                           AND *FET* IS THE ADDRESS OF THE INPUT FET.
 FILEB    SPACE  4,3
          MACRO  FILEB,FN,BF,WC,P1,P2,P3,P4,P5,P6,P7,P8
          =1     1,,5,;A,;B,;C,((;D),(;E),(;F),(;G),(;H),(;I),(;J),(;K))
          ENDM
 FILEC    SPACE  4,3
          MACRO  FILEC,FN,BF,WC,P1,P2,P3,P4,P5,P6,P7,P8
          =1     ,,5,;A,;B,;C,((;D),(;E),(;F),(;G),(;H),(;I),(;J),(;K))
          ENDM
 RFILEB   SPACE  4,3
          MACRO  RFILEB,FN,BF,WC,P1,P2,P3,P4,P5,P6,P7,P8
          =1  1,1,8D,;A,;B,;C,((;D),(;E),(;F),(;G),(;H),(;I),(;J),(;K))
          ENDM
 RFILEC   SPACE  4,3
          MACRO  RFILEC,FN,BF,WC,P1,P2,P3,P4,P5,P6,P7,P8
          =1  ,1,8D,;A,;B,;C,((;D),(;E),(;F),(;G),(;H),(;I),(;J),(;K))
          ENDM
 LABEL    SPACE  4,12D
***       LABEL - DEFINE TAPE LABEL INFORMATION.
*
*
*                 THE LABEL MACRO PROVIDES INITIALIZING INFORMATION FOR
*         FOR THE FET LABEL FIELDS.  IT MAY BE USED EITHER TO PRESET A
*         FET DURING ASSEMBLY OR TO GENERATE A 3 OR 4 WORD BLOCK OF
*         LABEL INFORMATION TO BE USED DURING EXECUTION.  IF THE LABEL
*         MACRO IS USED TO PRESET A FET, IT SHOULD BE NOTED THAT SPACE
*         FOR THE LABEL FIELDS MUST BE PRE-ALLOCATED WHEN THE FET IS
*         CREATED.  THUS THE *FILEC* OR *FILEB* MACRO USED TO CREATE THE
*         FET SHOULD INCLUDE EITHER THE *LBL* OR *VSN* PARAMETER, AS
*         APPROPRIATE.
*
*         THE CALL TO THE LABEL MACRO IS AS FOLLOWS:
*
*     FILE  LABEL  L=FNM,VSN=VSN,T=RP,E=ED,V=REEL,C=CR
*
*         WHERE:
*                 FILE = IF ABSENT, A 3 OR 4 WORD LABEL INFORMATION BLOCK
*                        IS GENERATED STARTING AT THE CURRENT VALUE OF
*                        THE ORIGIN COUNTER.
*                      = IF PRESENT AND NOT DEFINED, A LOCATION SYMBOL.
*                        IT IS DEFINED AS THE CURRENT VALUE OF THE
*                        LOCATION COUNTER.  A LABEL INFORMATION BLOCK IS
*                        GENERATED STARTING AT THE CURRENT VALUE OF THE
*                        ORIGIN COUNTER.
*                      = IF PRESENT AND DEFINED, A LABEL INFORMATION
*                        BLOCK IS GENERATED AT ORIGIN *FILE+9*, THUS
*                        PRESETING A FET.  -FILE- IS PRESUMED TO BE THE
*                        FWA OF A FET.
*                 FNM  = FILE LABEL, 1 TO 17 CHARACTERS.  IF
*                        OMITTED, THE RESULT IS BINARY ZERO,
*                        OTHERWISE, THE CHARACTER STRING IS LEFT
*                        JUSTIFIED WITH BLANK FILL.
*                 VSN  = VOLUMN SERIAL NUMBER OF THE VOLUMN.  1 TO 6
*                        CHARACTERS.  POSITIONING IS RIGHT JUSTIFIED
*                        WITH DISPLAY CODE ZERO FILL.
*                 RP   = RETENTION PERIOD.  1 TO 3 NUMERIC CHARACTERS.
*                        POSITIONING IS RIGHT JUSTIFIED WITH DISPLAY
*                        CODE ZERO FILL.
*                 ED   = EDITION NUMBER.  1 OR 2 NUMERIC CHARACTERS.
*                        POSITIONING IS RIGHT JUSTIFIED WITH DISPLAY
*                        CODE ZERO FILL.
*                 REEL = REEL NUMBER.  1 TO 4 NUMERIC CHARACTERS.
*                        POSITIONING IS RIGHT JUSTIFIED WITH DISPLAY
*                        CODE ZERO FILL.
*                 CR   = CREATION DATE.  FIVE NUMERIC CHARACTERS
*                        AS A JULIAN DATE (YYDDD).
*
*
*         ANY OMMITTED PARAMETER HAS THE DEFAULT OF BINARY ZERO.
*
*         PARAMETERS MAY APPEAR IN ANY ORDER.  PARAMETERS MAY BE
*         OMITTED SIMPLY BY LEAVING THEM OUT OF THE CALL LIST.
*
*
*
 LABEL    SPACE  4,10
**        =DSF MACRO - DISPLAY CODE ZERO FILL
*

 =DZF     MACRO  P,LENGTH,DZFMIC,N
 ;C  MICRO  1,, 0
 ^?DZF"AA MICRO  1,,⌐A ^?DZF"AA MICCNT ^?DZF"AA
          IFGT   ^?DZF"AA,;B,2
          ERR    LABEL PARAMETER ;D  (;A)  TOO LONG
          SKIP   10D
          IFNE   ^?DZF"AA,0,9D
          IFEQ   ^?DZF"AA,;B,2
 ;C  MICRO  1,,⌐A
          SKIP   3
 ^?DZF"AA SET    ;B-^?DZF"AA
 ^?DZF"AA MICRO  1,^?DZF"AA, 0000000000
 ;C  MICRO  1,,⌐^?DZF"AA';A
 ^?DZF"AA MICCNT ;C
 ^?DZF"AA DECMIC ^?DZF"AA,2
 ;C  MICRO  1,,⌐^?DZF"AA'L';C' =DZF     ENDM
 LABEL    SPACE  4,10
          MACROE LABEL,FILE,L,VSN,T,E,V,C
          BSS    0
 ^?LAB"EE SET    *O
 ^?LAB"GG SET    *
 ^?LAB"FF SET    0
          IFC    NE, ;A  ,1
          IF     -DEF,;A,2
 ;A  BSS  0
          SKIP   2
          ORG    ;A+9D
 ^?LAB"FF SET    1
          IFC    NE,;B
```

```
2
          DATA   17L;B
          SKIP   1
          BSSZ   2
          =DZF   ;E,2,^?LAB"AA,(EDITION NUMBER)
          =DZF   ;D,3,^?LAB"BB,(RETENTION CYCLE)
          =DZF   ;G,5,^?LAB"CC,(CREATION DATE)
          VFD    12D/'^?LAB"AA',18D/'^?LAB"BB',30D/'^?LAB"CC'
          =DZF   ;F,4,^?LAB"AA,(REEL NUMBER)
          VFD    36D/0,24D/'^?LAB"AA'
          IFC    NE,  ;C ,2
          =DZF   ;C,6,^?LAB"DD,(VOLUMN-SERIAL-NUMBER)
          VFD    36/'^?LAB"DD',24/0
          IFEQ   ^?LAB"FF,1,2
          ORG    ^?LAB"EE
          LOC    ^?LAB"GG
 LABEL    ENDM
 =1       SPACE  4,10
****      =1 - INTERNAL FET CREATION MACRO.
*
 =1       MACRO  MOD,RAN,SIZ,LFN,BUF,LEN,(P)

*                MOD = MODE BIT.                (;A)
*                RAN = RANDOM BIT.              (;B)
*                SIZ = MINIMUM FET SIZE.        (;C)
*                LFN = FILE NAME.               (;D)
*                BUF = FWA CIO BUFFER.          (;E)
*                LEN = LENGTH OF BUFFER.        (;F)
*                (P) = PARAMETER LIST.          (;G)
****

*         SYMBOL             MICRO
*^?FET"AA SCRATCH            SCRATCH
*^?FET"BB 1/EP,1/UP BITS     IND/FWA
*^?FET"CC FET LENGTH         IND/LEN
*^?FET"DD 1/DMP,1/VSN+EOR    WOF/EXT
*^?FET"EE                    WOF/FET
*
 ^?FET"BB SET
 ^?FET"CC SET    ;C
 ^?FET"DD SET
 ^?FET"BB MICRO
 ^?FET"CC MICRO
 ^?FET"DD MICRO
 ^?FET"EE MICRO
*
          IRP    P
          IFC    NE,  ;G ,35D
          IFC    NE, EOR ;G ,2
          IFC    EQ, VSN ;G ,3
 ^?FET"CC MAX    ^?FET"CC,14D
 ^?FET"DD SIOR   ^?FET"DD,1
          SKIP   30D
          IFC    EQ, DMP ;G ,2
 ^?FET"DD SIOR   ^?FET"DD,2
          SKIP   27D
          IFC    EQ, EPR ;G ,2
 ^?FET"BB SIOR   ^?FET"BB,1
          SKIP   24D
          IFC    EQ, UPR ;G ,2
 ^?FET"BB SIOR   ^?FET"BB,2
          SKIP   21D
          IFC    EQ, LBL ;G ,2
 ^?FET"CC MAX    ^?FET"CC,13D
          SKIP   18D
 ^?FET"AA MICRO  1,4, ;G
          IFC    EQ, FET= '^?FET"AA' ,3
 ^?FET"AA MICRO  5,, ;G
 ^?FET"CC MAX    ^?FET"CC,'^?FET"AA'
          SKIP   13D
          IFC    EQ, IND= '^?FET"AA' ,5
 ^?FET"BB MICRO  5,,,;G,
 ^?FET"AA MICCNT ^?FET"BB
 ^?FET"CC MICRO  6+^?FET"AA,, ;G
 ^?FET"CC MAX    ^?FET"CC,8D
          SKIP   7
          IFC    EQ, WOF= '^?FET"AA' ,5
 ^?FET"DD MICRO  5,,,;G,
 ^?FET"AA MICCNT ^?FET"DD
 ^?FET"EE MICRO  6+^?FET"AA,, ;G
 ^?FET"CC MAX    ^?FET"CC,6
          SKIP   1
 FET      ERR    UNRECOGNIZABLE PARAMETER - (;G)
          IRP
*
 ;D       VFD    42D/0L;D,17D/;A,1/1
          VFD    13/;B,3/^?FET"BB,8D/^?FET"DD,18D/^?FET"CC-5,18D/;E
          CON    ;E
          CON    ;E
          CON    ;E+;F
          IFGE   ^?FET"CC,6,6
          VFD    30D/'^?FET"EE',30D/'^?FET"DD'
          IFGE   ^?FET"CC,7,4
          CON
          IFGE   ^?FET"CC,8D,2
          VFD    42D/'^?FET"CC',18D/'^?FET"BB'
          BSSZ   ^?FET"CC-8D
          ENDM
 REQUEST  EJECT

*EJ

***       REQUEST STATUS BLOCK.
*
*         THE REQUEST MACRO GENERATES A 2 OR 3 WORD STATUS BLOCK FOR
*         INTERFACE TO THE PP PROGRAM -REQ- OR ITS CP STANDARD CALLING
*         ROUTINE -REQ=- (COMCREQ).
*
*         THE STATUS BLOCK HAS THE FOLLOWING FORM:
*
*T        STATUS 42/LOGICAL FILE NAME,4/,5/STS,9/COMPLETE
*T        FLAGS  12/,6/,6/EQ,12/FLAG1,12/FLAG2,12/DT
*T        VSN    36/TAPE-VSN,24/
*
*         THE BYTES -FLAG1-, -FLAG2-, AND -DT- HAVE THE
*         FOLLOWING FORMATS WITHIN THE -FLAGS- WORD.
*
*T        FLAG1  24/,1/ ,1/ ,1/ ,1/H,1/P,1/F,1/9,1/A,1/M,1/B,1/V,1/C,24/
*T        FLAG2  36/,1/ ,1/ ,1/ ,1/ ,1/ ,1/ ,1/ ,1/ ,1/D,1/S,1/I,1/E,12/
*T        DT     48/,6/DEV,6/ALLOC
*
*
*         THESE FIELDS AND BITS HAVE THE FOLLOWING USES:
*
*         STS    STATUS IS RETURNED TO THIS FIELD FOLLOWING A CALL TO
*                REQ.  THE RETURNED STATUSES (IN OCTAL) ARE:
*                =  0 REQUEST SATISFIED.
*                = 22 REQ CALLED WITHOUT AUTO=RECALL.
*                = 23 DEVICE DOES NOT MATCH DECLARED EQUIPMENT (EQ).
*                = 24 FNT IS FULL.
*                = 26 EQUIPMENT DOES NOT EXIST OR IS UNAVAILABLE TO THIS
*                     JOB.  (EG. CONTROL POINT TAPE POOL EXHAUSTED.)
*                = 30 FILE ALREADY ASSIGNED TO DEVICE.  THE ACTUAL
*                     DEVICE TYPE IS RETURNED TO THE DT FIELD.
*         EQ     EST ORDINAL OF DESIRED DEVICE.  THIS FIELD IS NORMALLY
*                SET ZERO, AND THE DT FIELD CONTROLS DEVICE ASSIGNMENT.
*                - NUCC ONLY:  THE EQ FIELD IS IGNORED UNLESS EITHER THE
*                              PROGRAM HAS SYSTEM PERMISSION OR IS THE
*                              EST ORDINAL OF AN ALLOCATABLE DEVICE.
*         VSN    THE VSN WORD IS PRESENT ONLY IF THE -V- BIT IN FLAG1
*                IS SET.  IF PRESENT, THE TAPE VSN SHOULD BE IN BITS
*                59-24, IN DISPLAY CODE, WITH RIGHT DISPLAY CODE ZERO
*                FILL.  THIS VSN IS USED TO AUTO-ASSIGN EXISTING LABELED
*                TAPES.
*
*         BITS IN FLAG1 AND FLAG2:
*         A      = 1 FOR AUTO ASSIGNMENT.
*                = 0 FOR FORCED OPERATOR ASSIGNMENT.
*         B      = 1 FOR TWO DEVICES.
*                = 0 FOR SINGLE DEVICE ASSIGNMENT.
*         C      = 1 FOR THE FILE TO HAVE EXISTING STATUS.
*                = 0 FOR THE FILE TO HAVE NEW STATUS.
*         D      = 1 FOR SYSTEM DEFAULT DENSITY ON A MAGNETIC TAPE.
*                = 0 IF DENSITY IS SPECIFIED IN THE ALLOC FIELD.
*                - MAGNETIC TAPE DEVICES ONLY.
*         E      = 1 TO FLAG FILE AS A CHECKPOINT DUMP FILE.
*                = 0 FILE NOT FLAGED.
*         F9     = 00 EXCEPT WHEN REQUESTING A 9-TRACK MAGNETIC TAPE.
*                = 01 FOR EBCDIC CHARACTER SET ON 9-TRACK TAPE.
*                = 10 FOR ASCII CHARACTER SET ON 9-TRACK TAPE.
*                = 11 FOR AN INTERESTING ABORT.
*         H      = 1 IF FILE MAY NOT OVERFLOW ASSIGNED DEVICE.
*                = 0 IF OVERFLOW IS ALLOWED.
*         I      = 1 FOR INHIBIT UNLOAD DISPOSITION.
*                = 0 FOR NORMAL DEVICE UNLOAD ACTION.
*         M      = 1 IF OPERATOR ASSIGNMENT NEEDED, RA+70 CONTAINS A
*                    1 TO 5 WORD C FORMAT MESSAGE TO BE DISPLAYED.
*                = 0 REQ WILL CONSTRUCT ITS OWN MESSAGE IF NEEDED.
*         P      = 1 IF FILE IS TO BE ASSIGNED TO A PERMANENT FILE
*                    DEVICE.
*                = 0 IF A PERMANENT FILE DEVICE IS NOT REQUIRED.
*                    - THIS BIT SHOULD BE SET ONLY IF AUTO ASSIGNMENT
*                      TO ANY ALLOCATABLE DEVICE HAS BEEN REQUESTED.
*         S      = 1 FOR SAVE DISPOSITION.
*                = 0 FOR NORMAL DISPOSITION.
*         V      = 1 IF THE REQUEST BLOCK IS 3 WORDS LONG AND
*                    THE VSN WORD IS PRESENT.
*                = 0 FOR A 2 WORD REQUEST STATUS BLOCK.
*
*
*         BITS B, D, F9, I, S, AND V ARE RELEVANT ONLY IF A MAGNETIC
*         TAPE UNIT IS BEING REQUESTED.  IF NOT, THEY SHOULD BE ZERO.
*
*
*         OTHER FIELDS. (DT, DEV, ALLOC)
*
*         DEV    = 0 INDICATES OPERATOR OR SYSTEM CHOICE OF DEVICE.
*                ' 0 DEVICE TYPE CODE OF REQUESTED DEVICE.  A FAIRLY
*                    COMPLETE LIST OF DEVICE TYPES CAN BE FOUND ON
*                    PAGE 3-10 OF THE SCOPE 3.3 REFERENCE MANUAL.
*                    THE NUCC IMPORTANT CODES ARE BELOW, IN OCTAL.
*                04 = 6603 DISK.
*                13 = 844 DISK.
*                20 = ECS.
*                40 = 7-TRACK MAGNETIC TAPE.
*                44 = PAPER TAPE READER.
*                45 = PAPER TAPE PUNCH.
*
*         ALLOC  ALLOCATION STYLE OR FORMAT FOR REQUESTED FILE.
*                IF A 7-TRACK MAGNETIC TAPE UNIT IS SPECIFIED BY -DEV-
*                ALLOC IS FORMATED AS XXYYZZ (BINARY) WHERE
*                XX = 00 => SCOPE STANDARD DATA FORMAT.
*                   = 01 => CDC RESERVED.
*                   = 10 => S DATA FORMAT.
*                   = 11 => L DATA FORMAT.
*                YY = 00 => UNLABELED.
*                   = 01 => SCOPE STANDARD (USASI U TYPE) LABELS.
*                   = 10 => ALTERNATE LABELS (3000 SERIES).
*                   = 11 => CDC RESERVED.
*                ZZ = 00 => HI DENSITY.  (556 BPI)
*                   = 01 => LO DENSITY.  (200 BPI)
*                   = 10 => HY DENSITY.  (800 BPI)
*                   = 11 => CDC RESERVED.
*
 REQUEST  SPACE  4,10
***       INTERACTIONS BETWEEN THE REQUEST STATUS BLOCK FIELDS.
*
*         IN GENERAL, SET THE BITS AND FIELDS TO REFLECT THE DESIRED
*         EQUIPMENT.  DO NOT COUNT ON THE SYSTEM TO IGNORE IRRELEVANT
*         BITS OR FIELDS.  IT IS KNOWN THAT
*                THE D BIT WILL OVERRIDE ANY DENSITY SPECIFIED WITH THE
*                ALLOC FIELD.
*                THE H BIT IS PROCESSED FOR ALL DEVICES.  ACTION IS
*                UNCLEAR WITH RESPECT TO SERIAL DEVICES.  (EG. TAPES.)
*                WITH RESPECT TO ALLOCATABLE DEVICES, CIO WILL ABORT A
*                JOB WHEN A FILE ASSIGNED WITH THE H BIT TO
*                AN ALLOCATABLE DEVICE OVERFLOWS THE
*                DEVICE.
*                THE P BIT MUST NOT BE SET UNLESS AUTO-ASSIGN TO
*                AN ALLOCATABLE DEVICE IS USED.  FOR NON AUTO-ASSIGN
*                THE POSSIBILITY OF AN OPERATOR ERROR EXISTS.
*                THE V BIT SHOULD BE SET FOR AUTO-ASSIGNMENT OF A
*                LABELED TAPE.  THE V BIT MAY PRODUCE PROBLEMS IF
*                THE DT FIELD SPECIFIES AN ALLOCATABLE DEVICE.
*                THE V BIT CAN BE SET FOR UNLABELED TAPES.
 REQUEST  SPACE  4,10
***       REQUEST MACRO.
*
*         THE REQUEST MACRO CAN BE USED TO GENERATE A REQUEST STATUS
*         BLOCK.  THE CALL IS:
*
*         LABEL  REQUEST  PARAM-LIST
*
*         LABEL IS DEFINED AS THE ADDRESS OF THE FIRST WORD OF THE
*                STATUS BLOCK.
*
*         PARAM-LIST IS ZERO OR MORE OF THE FOLLOWING PARAMETERS,
*         IN ANY ORDER, SEPARATED BY COMMAS.
*
*         LFN=X  SPECIFIES THE LOGICAL FILE NAME TO BE INSERTED
*                IN THE FIRST WORD OF THE BLOCK.  X IS ANY LEGAL FILE
*                NAME.  (EG. LFN=OUTPUT )
*         VSN=X  SPECIFIES THE TAPE-VSN, SETS THE V BIT AND GENERATES
*                THE VSN WORD.  (EG. VSN=OWNPL )
```

```
*         EQ=XX  SPECIFIES AN EST ORDINAL FOR THE EQ FIELD.
*                XX IS ANY TWO OCTAL DIGITS.  (EG. EQ=12 )
*         DT=X   SPECIFIES THE CONTENTS OF EITHER THE DT OR DEV FIELD.
*                IF X IS ONE OR TWO OCTAL DIGITS, SETS THE DEV FIELD.
*                IF X IS THREE OR FOUR OCTAL DIGITS, SETS THE ENTIRE
*                     DT FIELD.
*                IF X IS A TWO CHARACTER MNEMONIC, THE PROPER DEVICE
*                     TYPE CODE IS SET IN THE DEV FIELD.
*                MNEMONICS RECOGNIZED ARE:
*                MN  DT      UNIT
*                --  --  ------------------------------
*
*                AA  01  6603-I DISK.
*                AB  02  6638 DISK.
*                AC  04  6603-II DISK.
*                AL  05  821 DATA FILE.
*                AM  06  841 MULTIPLE DISK DRIVE.
*                AP  07  3234/854 DISK PACK DRIVE.
*                AF  10  814 DISK FILE.
*                AE  11  3637/863 DRUM.
*                AD  12  3637/865 DRUM.
*                AY  13  844 DISK FILE.
*                AX  20  ECS FILE.
*                MT  40  7-TRACK MAGNETIC TAPE UNIT.
*                NT  41  9-TRACK MAGNETIC TAPE UNIT.
*                TR  44  PAPER TAPE READER.
*                TP  45  PAPER TAPE PUNCH.
*                LP  50  501, 512, OR 505 LINE PRINTER.
*                L1  51  501 OR 505 LINE PRINTER.
*                L2  52  512 LINE PRINTER.
*                CR  60  405 CARD READER.
*                GF  66  PDP/8 - IMLAC GRAPHICS FILE.
*                CP  70  415 CARD PUNCH.
*                GC  72  GRAPHICS CONSOL.
*                HC  73  253-2 HARD COPY RECORDER.
*                FM  74  254-2 MICROFILM RECORDER.
*                DS  71  6612 KEYBOARD/DISPLAY CONSOL.
*
*         D=X    SETS THE DENSITY BITS IN ALLOC.  X MAY BE ONE OF
*                -  HI  FOR 556 BPI,
*                -  LO  FOR 200 BPI,
*                -  HY  FOR 800 BPI.
*         F=X    SETS THE RECORDING FORMAT.  X MAY BE ONE OF
*                -  SCOPE  FOR SCOPE FORMAT,
*                -  S      FOR STRANGER FORMAT,
*                -  L      FOR LONG RECORD FORMAT.
*         LABEL  SETS ALLOC FOR USASI -U- LABELS.
*         YLABEL SETS ALLOC FOR ALTERNATE (Y TYPE 3000) LABELS.
*         OV     SETS THE H BIT.  (NO DEVICE OVERFLOW)
*         EBCDIC SETS THE F9 FIELD TO 01.  (EBCDIC CHARACTER SET)
*         ASCII  SETS THE F9 FIELD TO 10.  (ASCII CHARACTER SET)
*                - NOTE: IF BOTH ASCII AND EBCDIC APPEAR, F9=11.
*         AUTO   SETS THE A BIT.  (AUTO-ASSIGN REQUESTED)
*         MSG    SETS THE M BIT.  (RA+70 CONTAINS OPERATOR MESSAGE)
*         TWO    SETS THE B BIT.  (TWO DEVICES REQUESTED)
*         VSN    SETS THE V BIT, FORCES 3 WORD STATUS BLOCK.
*         PF     SETS THE P BIT.  (PERMANENT FILE DEVICE)
*         EXIST  SETS THE C BIT.  (EXISTING STATUS ON FILE)
*         NEW    DOCUMENTS NEW STATUS.
*                - NOTE: IF BOTH NEW AND EXIST APPEAR, NEW IS
*                        IGNORED.
*         SDD    SETS THE D BIT.  (SYSTEM DEFAULT DENSITY)
*         IU     SETS THE I BIT.  (INHIBIT UNLOAD DISPOSITION)
*         SAVE   SETS THE S BIT.  (SAVE DISPOSITION)
*         CPD    SETS THE E BIT.  (FILE FLAGED AS CHECKPOINT DUMP FILE)
*
*
*         IF PARAM-LIST IS NULL, THE EFFECT IS
*
*         LABEL  BSSZ        2


 REQUEST  MACRO  P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12,P13,P14,P15,P16,
,P17,P18,P19,P20,P21,P22,P23,P24

          REQ=PL (;A,;B,;C,;D,;E,;F,;G,;H,;I,;J,;K,;L,;M,;N,;O,;P,;Q,;R,
,;S,;T,;U,;V,;W,;X)

          VFD    42/0L'^?REQ"LF',18/0
     VFD   12/^?REQ"B0,12/^?REQ"B1,12/^?REQ"B2,12/^?REQ"B3,12/^?REQ"B4
          IFNE   ^?REQ"VS,0,5
 ^?REQ"VS MICCNT ^?REQ"VS
          IFNE   ^?REQ"VS,9,2
          VFD    60/0
          SKIP   1
          VFD    36/'^?REQ"VS',24/0
          ENDM
 REQ=PL   SPACE  4,10
**        REQ=PL - PROCESS REQUEST/ASSIGN PARAMETER LIST.
*


 REQ=PL   MACRO  P
 ^?REQ"B0 SET    0
 ^?REQ"B1 SET    0
 ^?REQ"B2 SET    0
 ^?REQ"B3 SET    0
 ^?REQ"B4 SET    0
 ^?REQ"VS SET    0
 ^?REQ"LF SET    0
 ^?REQ"DT SET    0
 ^?REQ"EQ SET    0
 ^?REQ"LF MICRO
 ^?REQ"VS MICRO
          BASE   OCTAL
          IRP    P
          IFC    NE, ;A  ,57D
 ^?REQ"AA MICRO  1,2, ;A
 ^?REQ"AB MICRO  3,, ;A
 ^?REQ"AC MICRO  1,3, ;A
          IFC    EQ, '^?REQ"AC' VSN ,1
 ^?REQ"VS SET    1
          IFC    EQ, '^?REQ"AA' D= ,3
          REQ=ML ^?REQ"B4,'^?REQ"AB',(HY,HI,LO),(2,0,1)
 ^?REQ"DT SET    1
          SKIP   48D
          IFC    EQ, '^?REQ"AA' F= ,3
          REQ=ML ^?REQ"B4,'^?REQ"AB',(SCOPE,S,L),(0,40,60)
 ^?REQ"DT SET    1
          SKIP   44D
 ^?REQ"AA MICRO  1,3, ;A
          IFC    EQ, '^?REQ"AA' EQ= ,4
 ^?REQ"AA MICRO  4,, ;A
 ^?REQ"B1 SET    '^?REQ"AA' 0
 ^?REQ"EQ SET    1
          SKIP   38D
          IFC    EQ, '^?REQ"AA' DT= ,17D
 ^?REQ"AA MICRO  4,1, ;A
 ^?REQ"AA SET    1R'^?REQ"AA'
 ^?REQ"DT SET    1
          IFLE   ^?REQ"AA,1RZ,3
 ^?REQ"AA MICRO  4,2, ;A
          REQ=ML ^?REQ"B4,'^?REQ"AA',(AA,AB,AC,AL,AM,AP,AF,AE,AD,AY,AX,A
,*,PF,MT,NT,TR,TP,LP,L1,L2,CR,GF,CP,GC,HC,FM,DS),(100,200,400,500,600,70
,0,1000,1100,1200,1300,2000,0003,0077,4000,4100,4400,4500,5000,5100,5200
,,6000,6600,7000,7200,7300,7400,7100)
          SKIP   30D
          IFLE   ^?REQ"AA,1R7,7
 ^?REQ"AA MICRO  4,, ;A
 ^?REQ"AB MICCNT ^?REQ"AA
 ^?REQ"AA SET    '^?REQ"AA' 0
          IFLE   ^?REQ"AB,2,1
 ^?REQ"AA SET    ^?REQ"AA*100
 ^?REQ"B4 SIOR   ^?REQ"B4,^?REQ"AA
          SKIP   22D
          ERR    ;A  IS AN INVALID DEVICE TYPE
          SKIP   20D
 ^?REQ"AA MICRO  1,4, ;A
          IFC    EQ, '^?REQ"AA' VSN= ,5
 ^?REQ"B2 SIOR   ^?REQ"B2,2
 ^?REQ"AA MICRO  5,,←A          =DZF   ('^?REQ"AA'),6,^?REQ"VS,(VSN)
 ^?REQ"VS SET    1
          SKIP   13D
          IFC    EQ, '^?REQ"AA' LFN= ,3
 ^?REQ"LF SET    1
 ^?REQ"LF MICRO  5,, ;A
          SKIP   9D
 ^?REQ"AA SET    0
          REQ=ML ^?REQ"B2,;A,(EXIST,NEW,VSN,TWO,MSG,AUTO,ASCII,EBCDIC,PF
,,OV),(1,0,2,3,10,20,100,40,200,400),^?REQ"AA
          IFEQ   ^?REQ"AA,0,6
          REQ=ML ^?REQ"B3,;A,(CPD,SAVE,IU,SDD),(1,4,2,10),^?REQ"AA
          IFEQ   ^?REQ"AA,0,4
          REQ=ML ^?REQ"B4,;A,(LABEL,YLABEL,NOLABEL),(4,10,0),^?REQ"AA
          IFEQ   ^?REQ"AA,0,1
 U        ERR    UNRECOGNIZED PARAMETER (;A)
 ^?REQ"DT SET    1
          IRP
          BASE   *
          ENDM
 REQ=ML   SPACE  4,10
**        REQ=ML - PROCESS REQUEST/ASSIGN MNEMONIC LIST.
*
*         REQ=ML S,P,(FLIST),(VLIST),ERR
*
*         CALL   S = A SET SYMBOL.
*                P = PARAMETER TO BE MATCHED.
*                FLIST = A LIST OF CHARACTER STRING OPTIONS, SEPARATED
*                        BY COMMAS.
*                VLIST = A LIST OF VALUES, CORRESPONDING TO THE FLIST
*                        OPTIONS.
*                ERR = IF OMITTED, A -U- ERROR WILL BE GENERATED WHEN P
*                      IS NOT FOUND IN FLIST.
*                      IF PRESENT, A SET SYMBOL.
*         RETURN S = SIOR-ED WITH THE VLIST VALUE CORRESPONDING
*                    TO THE FLIST ITEM MATCHING P.
*                ERR = IF PRESENT,
*                    = 1 IF P MATCHED IN FLIST.
*                    = PREVIOUS VALUE IF NOT.
*


 REQ=ML   MACRO  S,P,FL,VL,ER
          LOCAL  AAAAAAAA
 AAAAAAAA SET    0
          ECHO   4,X=(FL),Y=(VL)
          IFC    EQ, X P ,3
 AAAAAAAA SET    1
 S        SIOR   S,Y
          STOPDUP
          IFC    EQ, ER  ,2
 U        ERRZR  AAAAAAAA  ARGUMENT (P) NOT IN (FL).
          SKIP   1
 ER       SIOR   AAAAAAAA,ER
          ENDM
 ASSIGN   SPACE  4,10
***       ASSIGN MACRO.
*
*         THE ASSIGN MACRO PROVIDES AN INTERFACE TO REQ= (COMCREQ).
*
*
*         THE MACRO CALL IS
*
*         LABEL  ASSIGN  STS,MSG,(SET),(CLEAR),V,L,D,E
*
*         WHERE  STS   = ADDRESS IF THE REQUEST STATUS BLOCK.
*                MSG   = IF PRESENT, ADDRESS OF A MESSAGE TO BE
*                        DISPLAYED TO THE OPERATOR, IF NEEDED.
*                SET   = A LIST OF BITS AND FIELDS TO BE SET IN THE
*                        REQUEST STATUS BLOCK.  ANY OF THE REQUEST
*                        MACRO PARAMETERS MAY BE USED.
*                CLEAR = A LIST OF BITS (IN FLAG1 AND FLAG2) TO BE
*                        CLEARED IN THE REQUEST STATUS BLOCK.  ANY OF
*                        THE REQUEST MACRO PARAMETERS MAY APPEAR, BUT
*                        ONLY THOSE PERTAINING TO THE FLAG1 AND FLAG2
*                        BYTES WILL BE PROCESSED.  SPECIFICLY, ONLY THE
*                        BIT FLAGS MAY BE CLEARED: LFN, VSN, DT AND EQ
*                        FIELDS MUST BE SET.
*                V     = ADDRESS OF A WORD CONTAINING A PROPER VSN WORD.
*                        OVERRIDES ANY -VSN=- LITERAL IN SET.
*                L     = ADDRESS OF A WORD CONTAINING A PROPER LOGICAL
*                        FILE NAME AS 42/0LLFN,18/**.  OVERRIDES ANY
*                        -LFN=- LITERAL IN SET.
*                D     = DEVICE TYPE EXPRESSION.
*                        IF THE FIRST CHARACTER OF THE EXPRESSION IS
*                        THE CHARACTER ^ THE REMAINDER OF THE EXPRESSION
*                        IS ASSUMED TO BE A REGISTER EXPRESSION YIELDING
*                        THE ADDRESS OF A DT WORD.
*                        OTHERWISE, A GENERAL REGISTER EXPRESSION
*                        YIELDING A DT WORD.  OVERRIDES ANY DT=, D=, F=,
*                        LABEL, YLABEL, AND NOLABEL LITERALS IN SET.
*                E     = EST ORDINAL EXPRESSION.
*                        IF THE FIRST CHARACTER OF THE EXPRESSION IS
*                        THE CHARACTER ^ THE REMAINDER OF THE EXPRESSION
*                        IS ASSUMED TO BE A REGISTER EXPRESSION YIELDING
*                        THE ADDRESS OF A WORD FORMATED 54/**,6/EST-ORD.
*                        OTHERWISE, A GENERAL REGISTER EXPRESSION
*                        YIELDING AN EST ORDINAL.  OVERRIDES ANY -EQ=-
*                        LITERAL IN SET.
*
*
*         ONLY THE STS PARAMETER IS REQUIRED.  IF THE MSG PARAMETER IS
*         PRESENT, THE PARAMETER -MSG- WILL BE ADDED TO ANY SET LIST.
*         IF THE -V- PARAMETER IS PRESENT, THE PARAMETER -VSN- WILL BE
*         INCLUDED IN ANY SET LIST.
```

```
*
*         ENTRIES IN THE SET LIST GENERATE ENTRIES IN THE LITERALS
*         BLOCK.


 ASSIGN   MACRO  S,M,SETLIST,CLR,V,L,D,E
 ^?ASN"MX SET    0
 ^?ASN"AA MICRO
          IFC    NE,⌐B,3
          R=     B2,;B
 ^?ASN"AA MICRO  1,, MSG,
          SKIP   1
          SB2    B0
 ^?ASN"VS SET    0
          IFC    NE,⌐E,2
 ^?ASN"VS SET    1
 ^?ASN"AA MICRO  1,, '^?ASN"AA'VSN,
          REQ=PL (;D)
 ^?ASN"C2 SET    ^?REQ"B2
 ^?ASN"C3 SET    ^?REQ"B3
          REQ=PL ('^?ASN"AA';C)
 ^?ASN"S2 SET    ^?REQ"B2
 ^?ASN"S3 SET    ^?REQ"B3
 ^?ASN"S2 OCTMIC ^?ASN"S2,4
 ^?ASN"S3 OCTMIC ^?ASN"S3,4
 ^?ASN"C2 OCTMIC ^?ASN"C2,4
 ^?ASN"C3 OCTMIC ^?ASN"C3,4
          IFEQ   ^?REQ"EQ,0,1
 ^?REQ"B1 SET    7777B
 ^?ASN"EQ OCTMIC ^?REQ"B1,4
          IFC    NE, ;H  ,11D
 ^?ASN"AA MICRO  1,1, ;H
          IFC    EQ, '^?ASN"AA' ^ ,3
 ^?ASN"AA MICRO  2,, ;H
          R=     A3,'^?ASN"AA'
          SKIP   1
          R=     X3,;H
          MX4    -12D
 ^?ASN"MX SET    1
          BX1    -X4*X3
 ^?ASN"EQ MICRO  1,, 0000
          LX1    -12D
 ^?ASN"AA MICRO  1,, '^?ASN"EQ''^?ASN"S2''^?ASN"S3''^?ASN"C2''^?ASN"C3'
          SA3    ='^?ASN"AA'B
          IFC    NE, ;H  ,1
          BX3    X3+X1
          IFC    NE, ;E  ,2
          R=     B4,;E
          SKIP   4
          IFNE   ^?REQ"VS,0,2
          SB4    ='^?REQ"VS'
          SKIP   1
          SB4    B0
          IFC    NE, ;F  ,2
          R=     B3,;F
          SKIP   4
          IFNE   ^?REQ"LF,0,2
          SB3    =C⌐^?REQ"LF'
          SKIP   1
          SB3    B0
          IFC    NE, ;G  ,10D
 ^?ASN"AA MICRO  1,1, ;G
          IFC    EQ, '^?ASN"AA' ^ ,3
 ^?ASN"AA MICRO  2,, ;G
          R=     A1,'^?ASN"AA'
          SKIP   1
          R=     X1,;G
          IFEQ   ^?ASN"MX,0,1
          MX4    -12D
          BX4    -X4*X1
          SKIP   5
          IFNE   ^?REQ"DT,0,2
          R=     X4,^?REQ"B4
          SKIP   2
          IFEQ   ^?ASN"MX,0,1
          MX4    60D
          R=     X2,;A
          RJ     =XREQ=
          ENDM
 SAVE     SPACE  4,13D
*****     SAVE - SAVE REGISTER CONTENTS.
*
 SAVE     MACRO  RSA

*         ENTRY  RSA = ADDRESS OF 24 WORD REGISTER SAVE AREA.
*
*         EXIT   (B1) = 1.
*
*         CALLS  SVR=.
*****
 +        RJ     =XSVR=
 -        VFD    30D/;A
          ENDM
 RESTORE  SPACE  4,15D
*****     RESTORE - RESTORE REGISTER CONTENTS.
*
*                RESTORE SETS ALL REGISTERS AS THEY WERE PRECEDING A
*         PREVIOUS REGISTER SAVE CALL.
*
 RESTORE  MACRO  RSA

*         ENTRY  RSA = ADDRESS OF 24 WORD REGISTER SAVE AREA.
*                (B1) = 1.
*
*         CALLS  RSR=.
*****
          R=  X1,;A
          RJ     =XRSR=
          ENDM
 SNAP     SPACE  4,40
*****     SNAP - PRINT REGISTERS AND/OR MEMORY.
*
 SNAP     MACROE FWA,LWA,NR,NH,LNG,M1,M2,M3,NM

*                FWA = FIRST WORD ADDRESS OF MEMORY TO DUMP.
*                      DEFAULT = 0.
*                LWA = LAST WORD ADDRESS+1 OF MEMORY TO DUMP.
*                      DEFAULT = 0.
*                NR  < 0, IF NO REGISTER DUMP DESIRED.
*                      DEFAULT = 0.
*                NH  ' 0, IF NO HEADINGS DESIRED.
*                      DEFAULT = 0.
*                LNG = LENGTH OF MEMORY TO DUMP.  USED ONLY IF LWA = 0.
*                      DEFAULT = 0.
*                M1  = SNAP CALL NUMBER OF FIRST DUMP.
*                      DEFAULT = 1.
*                M2  = SNAP CALL NUMBER OF LAST DUMP.
*                      DEFAULT = 377777B.
*                M3  = NUMBER OF SNAP CALLS BETWEEN DUMPS.
*                      DEFAULT = 1.
*                NM  = NAME OF THIS SNAP CALL.  DISPLAY CODE CONSTANT.
*                      DEFAULT = 0.
*****
 +        RJ     =XSNP=
 -        VFD    30D/0
          CON    ;A
          CON    ;B
          VFD    10D/;C,10D/;D,40D/;E
          CON    ;F 1
          CON    ;G 377777B
          CON    ;H 1
          VFD    60D/;I
          CON    0
          ENDM
 TRACE    SPACE  4,16
*****     TRACE - 6000 INTERPRETIVE INSTRUCTION TRACE.
*
 TRACE    MACRO  ONOFF,(FLAGS)

*         TRACE  ON,(FLAGS)  START TRACE
*         TRACE  OFF         STOP TRACE
*
*                (FLAGS) = TRACE ON PARAMETER LIST AS FOLLOWS:
*                XJ = ENABLE XJ EXECUTION.
*                7600 = ENABLE 7600 INSTRUCTION SET.
*                NOMEM = ASSUME FIELD LENGTH WILL NOT CHANGE DURING
*                        EXECUTION, (DEFAULT CHECKING ON ENTRY AND
*                        AFTER EACH XJ.)
*                NOREG = NO REGISTER DUMP ON ENTRY.
*
*         CALLS  TRC=, TRX=.
*****
 TRACE    IFC    EQ, ON ;A
 ^?TRC"AA SET    0
          IRP    FLAGS
          IFC    EQ, XJ ;B ,2
 ^?TRC"AA SIOR   ^?TRC"AA,1
          SKIP   10D
          IFC    EQ, 7600 ;B ,2
 ^?TRC"AA SIOR   ^?TRC"AA,2
          SKIP   7
          IFC    EQ, NOMEM ;B ,2
 ^?TRC"AA SIOR   ^?TRC"AA,4
          SKIP   4
          IFC    EQ, NOREG ;B ,2
 ^?TRC"AA SIOR   ^?TRC"AA,10B
          SKIP   1
          ERR    ILLEGAL TRACE PARAMETER - (;B)
          IRP
 +        RJ     =XTRC=
 -        VFD    30/^?TRC"AA
 TRACE    ELSE   4
          IFC    EQ, OFF ;A ,2
          RJ     =XTRX=
          SKIP   1
 TRACE    ERR    ILLEGAL TRACE OPTION - (;A)
          ENDM
 CRACK    SPACE  4,17
*****     CRACK - CRACK CONTROL CARD PARAMETERS.
*
 CRACK    MACRO  ARGTABL

*                ARGTABL = ARGUMENT TABLE FOR ARG ROUTINE.
*
*         USES   ALL REGISTERS EXCEPT A0 AND A5.
*
*         CALLS  SES AND ARG.
*****
          RJ     =XSES
          SA3    ACTR
          SB4    X3
          R=  A4,ARGR
          R=  B5,;A
          RJ     =XARG
          ENDM
 COMMTXT  SPACE  4
 COMMTXT  ENDX
COMCSYS
COMMON
 SYS=     CTEXT  COMCSYS - PROCESS SYSTEM REQUEST.
          IF     -DEF,QUAL$,1
          QUAL   COMCSYS
 SYS=     SPACE  4
***       COMCSYS - PROCESS SYSTEM REQUEST.
*         G. R. MANSFIELD.  12/08/69.
 SYS=     SPACE  4
***              COMCSYS CONTAINS ROUTINES FOR PROCESSING CERTAIN
*         SYSTEM REQUESTS.  ENTRY POINTS ARE DERIVED BY APPENDING
*         AN EQUAL SIGN TO THE ROUTINE NAME.
 SYS=     SPACE  4
***       SYS - ISSUE SYSTEM REQUEST.
*         S. H. KEYSER.   08/02/73.
 SYS.     SPACE  4
***              SYS PROCESSES SYSTEM REQUESTS.
*
*         ENTRY  (X6) = SYSTEM REQUEST.
*
*         EXIT   REQUEST PROCESSED.
*
*         USES   A1, X1, A6.
*

          IF     DEF,XJPR,33D
 SYS1     PX1    X1          SET/CLEAR BIT 58
          AX1    59D-1       =+&-1
          SA1    X1+SYS.     FETCH XJ OR WAIT LOOP
          BX6    X1-X6       REGISTER SWAP
          BX1    X1-X6
          BX6    X1-X6
          LX6    30D
          SA6    X6          =SYS2
          BX6    X1          RESTORE X6
          IF     -DEF,B1=1,1
          SA1    1           RESTORE A1
          RJ     *           VOID STACK

 SYS2     SA1    XJPR        FIRST ENTRY GO CHECK FOR CEJ OPTION
          EQ     SYS1        SUBSEQUENT ENTRY WILL HAVE XJ OR WAIT LOOP

 SYS=     VFD    42D/4LSYS=,18D/*
          R=     A1,1
          NZ     X1,*        WAIT (RA+1) CLEAR
          SA6    A1
          EQ     SYS2
```

```
          IF     -DEF,B1=1,1
 +        VFD    30D/SYS2    FORCE UPPER
 -        XJ

 SYS.                       SCRATCH CELL FOR RECALL REQUESTS
 +        NG     X1,SYS2     WAIT LOOP FOR NO XJ
          IF     -DEF,B1=1,2
          SA1    A1
          SKIP   1
          SA1    B1
          LX1    59D-40D
          SKIP   11D
 SYS1     SA1    A1
          LX1    59D-40D
          NG     X1,SYS1     WAIT (RA+1) CLEAR IF AUTO-RECALL

 SYS=     VFD    42D/4LSYS=,18D/*
          R=     A1,1
          NZ     X1,*
          SA6    A1
          EQ     SYS1

 SYS.                       SCRATCH CELL FOR RECALL REQUESTS
 RCL=     SPACE  4,17D
***       RCL - PLACE PROGRAM ON RECALL.
*
*         EXIT   REQUEST PROCESSED IF RA+1 CLEAR.
*
*         USES   A1, X1, A6, X6.
*
*         CALLS  SYS=.
*

 RCL1     RJ     =XSYS=

 RCL=     VFD    42D/4LRCL=,18D/*
          R=     A1,1
          SX6    3RRCL
          LX6    42D
          ZR     X1,RCL1
          EQ     RCL=        RETURN IF RA+1 NOT CLEAR
 WNB=     SPACE  4,26D
***       WNB - WAIT NOT BUSY.
*
*         ENTRY  (X2) = ADDRESS OF FET STATUS WORD.
*
*         EXIT   IF STATUS WORD ZERO OR WHEN BIT 0 IS SET.
*
*         USES   A1, X1, A6, X6.
*
*         CALLS  SYS=.
*

 WNB2     R=     A1,1        WAIT RA+1 CLEAR
          NZ     X1,WNB1
          SX1    X2
          BX6    X6+X1
          RJ     =XSYS=

 WNB=     VFD    42D/4LWNB=,18D/*
          SX1    3RRCL       FORM RECALL REQUEST
          PX6    X1
          LX6    42D

 WNB1     SA1    X2
          LX1    -1
          NG     X1,WNB=     RETURN IF STATUS BIT SET
          NZ     X1,WNB2
          EQ     WNB=        RETURN IS STATUS WORD ZERO
 DFM=     SPACE  4,38D
***       DFM - SEND DAYFILE MESSAGE.
*         S. H. KEYSER.     08/20/73.
          SPACE  4
***                DFM PROCESSES REQUESTS FOR SCOPE MSG.
*
*         ENTRY  (X6) = ADDRESS OF MESSAGE IN -C- FORMAT.
*                (X1) = MSG OPTIONS, IN COMPLIMENT FOR AUTO-RECALL.
*                       0 = MESSAGE TO SYSTEM DAYFILE, CONTROL POINT
*                           DAYFILE AND SEND TO B DISPLAY.
*                       1 = MESSAGE TO CONTROL POINT DAYFILE ONLY.
*                     10B= MESSAGE TO SYSTEM DAYFILE ONLY.
*                    100B= MESSAGE TO B DISPLAY ONLY.
*
*         USES   A1, X1, A6, X6.
*
*         CALLS  SYS=.
*

 DFM1     LX1    18D         POSITION MSG OPTIONS
          BX6    X1+X6
          SX1    3RMSG
          LX1    42D
          BX6    X1+X6
          RJ     =XSYS=

 DFM=     VFD    42D/4LDFM=,18D/*
          PL     X1,DFM1     NO RECALL
          LX6    30D
          BX1    -X1
          SA6    SYS.        SET MSG STATUS WORD
          LX1    -24D
          PX1    X1          SET RECALL
          SX6    A6          SET INDIRECT ADDRESS
          LX1    24D
          EQ     DFM1
 TIM=     SPACE  4,27
***       TIM - PROCESS TIM FUNCTION.
*         S. H. KEYSER.     08/08/73.
 TIM=     SPACE  4
***                TIM PROCESSES REQUESTS FOR THE MONITOR FUNCTION TIM.
*
*         ENTRY  (X6) = TIM FUNCTION CODE.
*
*         EXIT   (X6) = RETURNED TIM RESPONSE.
*
*         USES   A1, X1, A6.
*
*         CALLS  SYS=.
*

 TIM=     VFD    42D/4LTIM=,18D/*
          SA1    TIMA
          LX6    24D         POSITION FUNCTION CODE
          BX6    X1+X6       FORM REQUEST
          RJ     =XSYS=      ISSUE
          SA1    X6          RETRIEVE RESPONSE
          BX6    X1
          EQ     TIM=

 TIMA     VFD    24D/4LTIMP,18D/,18D/SYS.  SKELETON TIM CALL
 MEM=     SPACE  4,33D
***       MEM - PROCESS MEMORY REQUEST.
*         S. H. KEYSER.     08/10/73.
 MEM=     SPACE  4
***                MEM PROCESSES REQUESTS FOR CENTRAL OR EXTENDED CORE.
*
*         ENTRY  (X1) = TYPE OF REQUEST, (0 FOR CM, 1 FOR ECS).
*                (X6) = WORD COUNT REQUEST, (0 IF NO CHANGE).
*
*         EXIT   (X6) = CURRENT CM OR ECS FIELD LENGTH.
*
*         USES   A1, X1, A6.
*
*         CALLS  SYS=.
*

 MEM1     LX6    18D
          BX6    X1+X6
          RJ     =XSYS=
          SA1    SYS.
          BX6    X1
          AX6    30D

 MEM=     VFD    42D/4LMEM=,18D/*
          LX6    30D         POSITION WORD COUNT
          SA6    SYS.
          BX6    X1
          SA1    MEMA
          EQ     MEM1

 MEMA     VFD    24D/4LMEMP,18D/,18D/SYS.
 COMCSYS  SPACE  4,9D
          IF     -DEF,QUAL$,8D
          QUAL   *
 SYS.     =      /COMCSYS/SYS.
 SYS=     =      /COMCSYS/SYS=
 RCL=     =      /COMCSYS/RCL=
 WNB=     =      /COMCSYS/WNB=
 DFM=     =      /COMCSYS/DFM=
 TIM=     =      /COMCSYS/TIM=
 MEM=     =      /COMCSYS/MEM=
 SYS=     ENDX
COMCCIO
COMMON
 CIO=     CTEXT  COMCCIO - I/O FUNCTION PROCESSOR.
          IF     -DEF,QUAL$,1
          QUAL   COMCCIO
          BASE   MIXED
 CIO=     SPACE  3
***       CIO - I/O FUNCTION PROCESSOR.
*         G. R. MANSFIELD.  12/08/69.
*         S. H. KEYSER.     07/18/73.  (6400)
*
*         ENTRY  (X2) = 24/0,18/P,18/FET ADDRESS.  P CAN BE EITHER A
*                       SKIP COUNT OR AN -OPEN- PARAMETER.
*                (X7) = FUNCTION CODE, IN COMPLIMENT FOR AUTO-RECALL.
*                (B1) = 1.
*
*         EXIT   (X2) = 42/0, 18/FET ADDRESS.
*
*                FUNCTION WILL BE ISSUED WHEN FET IS NOT BUSY.
*                IF FET STATUS WORD IS 0, OPERATION WILL NOT BE
*                PROCESSED AND IN AND OUT WILL BE SET TO FIRST.
*
*         USES   X - 1, 6, 7.
*                B - NONE.
*                A - 1, 6.
*
*         CALLS  WNB=, SYS=.
*

 CIO1     RJ     =XWNB=      WAIT FET NOT BUSY

 CIO2     SA1    X2          MASK OFF FILE NAME AND MODE FROM FET
          SX6    177777
          PX6    X6
          LX6    2
          BX6    -X6*X1
          SX1    3RCIO
          PL     X7,CIO3     AUTO-RECALL NOT REQUESTED.
          BX7    -X7
          PX1    X1

 CIO3     IX6    X6+X7
          LX1    52
          SA6    X2          POST FUNCTION CODE TO FET
          BX6    X1+X2       FORM CIO CALL
          SX2    X2          CLEAR BITS 18-35
          RJ     =XSYS=

 CIO=     VFD    42/4LCIO=,18/*
          SA1    X2          CHECK BUFFER STATUS
          LX1    -1
          NG     X1,CIO2     FET NOT BUSY
          NZ     X1,CIO1     STATUS WORD NON-ZERO AND BUSY
          IF     -DEF,B1=1,5 SET IN = OUT = FIRST
          SA1    X2+1
          SX6    X1
          SA6    X2+2
          SA6    A6+1
          SKIP   4
          SA1    X2+B1
          SX6    X1
          SA6    A1+B1
          SA6    A6+B1
          EQ     CIO=        RETURN
 CIO=     SPACE  3
          BASE   *
          IF     -DEF,QUAL$,2
          QUAL   *
 CIO=     =      /COMCCIO/CIO=
 CIO=     ENDX
COMCRDC
COMMON
 RDC=     CTEXT  COMCRDC - READ CODED LINE, -C- FORMAT.
          IF     -DEF,QUAL$,1
          QUAL   COMCRDC
          BASE   DECIMAL
 RDC      SPACE  4
***       RDC - READ CODED LINE, -C- FORMAT.
*         1 CODED LINE IS TRANSFERRED TO THE WORKING BUFFER.
*
*         ENTRY  (X2) = ADDRESS OF FET FOR FILE.
*                (B6) = FWA WORKING BUFFER.
```

```
*               (B7) = WORD COUNT OF WORKING BUFFER.
*         EXIT  (X2) = ADDRESS OF FET FOR FILE.
*               (B6) = LWA+1 TRANSFERED INTO WORKING BUFFER.
*               (X1) = 0, IF TRANSFER COMPLETE.
*                    = -2, IF EOI.
*                    = -1, IF EOF.
*                    = (B6) = LWA+1 OF DATA TRANSFERRED, IF EOR.
*
*         USES  ALL REGISTERS EXCEPT A0, X0, A5, X5.
*         CALLS LCB=, RDX=.


 +        EQ    RDC1

 RDC=     EQ    *+1S17      ENTRY/EXIT
          SA4   *-1         SET RETURN ADDRESS
          IF    -DEF,B1=1,1
          SB1   1
          SA1   X2+4        (B5) = LIMIT
          SA3   X2+B1       (X3) = FIRST
          SB7   B6+B7       (B7) = LWA+1 WORKING BUFFER
          MX4   60-12       (X4) = BYTE MASK
          SB5   X1

*         INITIALIZE REGISTERS FOR TRANSFER.

 RDC1     SA1   A3+B1       (B3) = IN
          SA2   A1+B1       (B4) = OUT
          SB3   X1
          SB4   X2

*         TRANSFER DATA FROM CIRCULAR BUFFER TO WORKING BUFFER.

 RDC2     EQ    B4,B3,=XLCB= LOAD CIRCULAR BUFFER IF OUT = IN
          SB4   B4+B1       (OUT+1)
          SA1   B4-B1       READ WORD
          BX7   -X4*X1      CHECK LOWER BYTE
          NE    B4,B5,RDC3  IF (OUT+1) ' LIMIT
          SB4   X3          (OUT+1) = FIRST

 RDC3     EQ    B6,B7,RDC4  WORKING BUFFER FULL
          BX6   X1
          SA6   B6          STORE WORD
          SB6   B6+1        ADVANCE WORKING BUFFER
          ZR    X7,=XRDX=   EXIT IF END OF LINE
          NE    B6,B7,RDC2  LOOP TO FILL WORKING BUFFER
          BX6   X4*X6       CLEAR LAST BYTE
          SA6   A6

 RDC4     ZR    X7,=XRDX=   EXIT IF END OF LINE
          EQ    RDC2        SKIP TO END OF CODED LINE
          SPACE 4
          BASE  *
          IF    -DEF,QUAL$,2
          QUAL  *
 RDC=     =     /COMCRDC/RDC=
 RDC=     ENDX
COMCWTW
COMMON
 WTW=     CTEXT COMCWTW - WRITE WORDS.
          IF    -DEF,QUAL$,1
          QUAL  COMCWTW
          BASE  DECIMAL
 WTW      SPACE 4
***       WTW - WRITE WORDS FROM WORKING BUFFER.
*
*         ENTRY (X2) = ADDRESS OF FET FOR FILE.
*               (B6) = FWA WORKING BUFFER.
*               (B7) = WORD COUNT OF WORKING BUFFER.
*               IF (B7) = 0, NO TRANSFER WILL BE PERFORMED.
*         EXIT  (X2) = ADDRESS OF FET FOR FILE.
*
*         USES  ALL REGISTERS EXCEPT A0, X0, A5, X5.
*         CALLS DCB=, WTX=.


 +        EQ    WTW1

 WTW=     EQ    *+400000B   ENTRY/EXIT
          SA4   *-1
          ZR    B7,WTW=     IF WORKING BUFFER EMPTY
          IF    -DEF,B1=1,1
          SB1   1
          SA1   X2+4        (B5) = LIMIT
          SA3   X2+B1       (X3) = FIRST
          SB7   B6+B7       (B7) = LWA+1 WORKING BUFFER
          SB5   X1

*         INITIALIZE REGISTERS FOR TRANSFER.

 WTW1     SA1   A3+2        (B4) = OUT
          SA2   A3+B1       (X2) = IN
          SB4   X1

*         TRANSFER DATA FROM WORKING BUFFER TO CIRCULAR BUFFER.

 WTW2     SB3   X2+B1       (IN+1)
          NE    B3,B5,WTW3  IF (IN+1) ' LIMIT
          SB3   X3          (IN+1) = FIRST
 WTW3     SA1   B6          NEXT WORD
          EQ    B3,B4,DCB=  DUMP CIRCULAR BUFFER IF (IN+1) = OUT
          SB6   B6+B1       ADVANCE WORKING BUFFER
          BX6   X1
          NO
          SA6   X2          STORE WORD
          SX2   B3          IN = IN+1
          NE    B6,B7,WTW2  LOOP TO END OF WORKING BUFFER
*         EQ    WTX         EXIT
 WTX      SPACE 4
***       WTX - WRITE EXIT.
*         IF BUFFER IS BUSY, RETURN.
*         OTHERWISE, WORD COUNT OF BUFFER IS CHECKED, AND A WRITE
*         FUNCTION IS REQUESTED IF NECESSARY.
*
*         ENTRY (A2) = ADDRESS OF IN.
*               (A3) = ADDRESS OF FIRST.
*               (A4) = RETURN ADDRESS.
*               (B3) = IN+1.
*               (B4) = OUT.
*               (B5) = LIMIT.
*               (X2) = IN
*         EXIT  TO RETURN ADDRESS.
*
*         USES  X - 1, 2, 3, 4, 6, 7.
*               B - 2.
*               A - 1, 6.
*
*         CALLS CIO=.


 WTX=     SA1   A3-B1       CHECK BUFFER STATUS
          SX6   X2          STORE IN
          LX1   59-0
          SA6   A2
          PL    X1,WTX1     IF BUFFER BUSY

**        IF BUFFER IS NOT BUSY, CHECK SIZE OF BUFFER.
*         ISSUE WRITE IF THRESHOLD IS REACHED.

          SA3   A3          FIRST
          SX6   B4-B3       (OUT-IN+1)
          SB2   X3          (LIMIT-FIRST)
          LX3   X6,B1       2*(OUT-IN+1)
          SX7   B5-B2
          AX6   60          SIGN OF (OUT-IN+1)
          BX4   X6-X7       INVERT BUFFER IF IN+1 # OUT
          IX6   X4-X3       BUFFER SIZE - 2*(OUT-IN+1)
          NG    X6,WTX1     IF BUFFER THRESHOLD NOT REACHED
          SX7   14B
          SX2   A3-B1
          RJ    =XCIO=

 WTX1     SB2   A4          SET RETURN ADDRESS
          SX2   A3-B1       RESET (X2)
          JP    B2          RETURN
 DCB      SPACE 4
***       DCB - DUMP CIRCULAR BUFFER.
*         IF BUFFER IS BUSY, RECALL AND RETURN.
*         IF BUFFER IS NOT BUSY, REQUEST WRITE FUNCTION AND RETURN.
*
*         ENTRY (A2) = ADDRESS OF IN.
*               (A3) = ADDRESS OF FIRST.
*               (A4) = RETURN ADDRESS.
*               (X2) = IN.
*         EXIT  TO RETURN ADDRESS - 1.
*
*         USES  X - 1, 2, 6, 7.
*               B - 2.
*               A - 1, 6.
*
*         CALLS CIO=, RCL=.


 DCB=     SA1   A3-B1       CHECK BUFFER STATUS
          SX6   X2          STORE IN
          LX1   59
          SA6   A2
          SX2   A3-B1
          SB2   A4-B1       CONTINUE WRITE
          NG    X1,DCB1     IF NOT BUSY
          ZR    X1,DCB1     OR BLANK FET
          RJ    =XRCL=
          JP    B2
 DCB1     SX7   14B
          RJ    =XCIO=
          JP    B2
          SPACE 4
          BASE  *
          IF    -DEF,QUAL$,4
          QUAL  *
 WTW=     =     /COMCWTW/WTW=
 WTX=     =     /COMCWTW/WTX=
 DCB=     =     /COMCWTW/DCB=

 WTW=     ENDX
COMCWTC
COMMON
 WTC=     CTEXT COMCWTC - WRITE CODED LINE, -C- FORMAT.
          IF    -DEF,QUAL$,1
          QUAL  COMCWTC
          BASE  DECIMAL
 WTC      SPACE 4
***       WTC - WRITE CODED LINE, -C- FORMAT.
*         1 CODED LINE IS TRANSFERRED FROM THE WORKING BUFFER.
*
*         ENTRY (X2) = ADDRESS OF FET FOR FILE.
*               (B6) = FWA WORKING BUFFER.
*         EXIT  (X2) = ADDRESS OF FET FOR FILE.
*
*         USES  ALL REGISTERS EXCEPT A0, X0, A5, X5, B7.
*         CALLS DCB=, WTX=.
 WTC=     SPACE 4
**        ASSEMBLY CONSTANT.
*
*         WTC[C64] SHOULD BE DEFINED IF COMCWTC IS TO BLANK FILL LINES
*               TO THE FINAL ZERO BYTE.  WTC[C64] WILL BE DEFINED IF
*               IP.CSET IS EQUAL TO 64.

          IF    -DEF,WTC[C64],3
          IF    DEF,IP.CSET,2
          IFEQ  IP.CSET,64,1
 WTC[C64] SET   1


 +        EQ    WTC1

 WTC=     EQ    *+1S17      ENTRY/EXIT
          SA4   *-1
          IF    -DEF,B1=1,1
          SB1   1
          SA1   X2+4        (B5) = LIMIT
          SA3   X2+B1       (X3) = FIRST
          IF    DEF,WTC[C64],2
          MX4   60-6
          SKIP  1
          MX4   60-12       (X4) = BYTE MASK
          SB5   X1

*         INITIALIZE REGISTERS FOR TRANSFER.

 WTC1     SA1   A3+2        (B4) = OUT
          SA2   A3+B1       (X2) = IN
          SB4   X1

*         TRANSFER DATA FROM WORKING BUFFER TO CIRCULAR BUFFER.

 WTC2     SB3   X2+B1       (IN+1)
          NE    B3,B5,WTC3  IF (IN+1) ' LIMIT
          SB3   X3          (IN+1) = FIRST
 WTC3     SA1   B6          NEXT WORD
          EQ    B3,B4,=XDCB= DUMP CIRCULAR BUFFER IF (IN+1) = OUT
          LX6   X1
          SB6   B6+B1       ADVANCE WORKING BUFFER
          BX7   -X4*X1
```

```
          SA6    X2          STORE WORD
          SX2    B3          IN = IN+1
          NZ     X7,WTC2     LOOP TO END OF LINE
          IF     DEF,WTC[C64],14D
          MX7    60-12
          BX1    -X7*X6
          NZ     X1,WTC4     IF NO ZERO BYTE
          IX7    X6+X7
          BX7    -X6*X7
          SA1    =4040404040404040404040B  BLANK FILL TO ZERO BYTE
          BX7    X1*X7       40B WHERE BLANKS NEEDED
          BX1    X7
          LX7    -3
          BX1    X1+X7       44B WHERE BLANKS NEEDED
          BX6    X6+X1
          LX1    -2          11B WHERE BLANKS NEEDED
          BX6    X6+X1       X6 BLANK FILLED TO ZERO BYTE
          SA6    A6          REWRITE LAST WORD
          EQ     =XWTX=      EXIT
          IF     DEF,WTC[C64],5

 WTC4     SX1    1R          BLANK FILL LAST ZERO CHARACTER
          IX6    X6+X1
          SA6    A6          REWRITE LAST WORD
          EQ     WTC2
          SPACE  4
          BASE   *
          IF     -DEF,QUAL$,2
          QUAL   *
 WTC=     =      /COMCWTC/WTC=
 WTC=     ENDX
COMCPFM
COMMON
 PFM=     CTEXT  COMCPFM - CALL PERMANENT FILE MANAGER.
          IF     -DEF,QUAL$,1
          QUAL   COMCPFM
          BASE   DECIMAL
 PFM=     SPACE  4
***       PFM - PERMANENT FILE FUNCTION PROCESSOR.
*         L. R. ATKIN.             08/03/73.
 PFM=     SPACE  4
***               PFM PROCESSES REQUESTS FOR THE SCOPE PERMANENT
*         FILE MANAGER.
*
*         ENTRY  (X2) = FDB ADDRESS.
*                (X6) = PERMANENT FILE ROUTINE NAME.
*                (X7) = FUNCTION CODE.
*
*         EXIT   (X6) = PFM REPLY.
*
*         USES   A1, A6, A7, X1, X7.
*         CALLS  SYS=.
*


 PFM1     BX7    X1-X7            REMOVE OLD FUNCTION
          SA7    A1          UPDATE FDB
          RJ     =XSYS=           ISSUE REQUEST
          SA1    X2
          MX6    -9
          LX1    -9
          BX6    -X6*X1           EXTRACT REPLY

 PFM=     EQ     *+400000B        ENTRY/EXIT
          PX6    X6               SET RECALL BIT
          LX6    42
          BX6    X2+X6            FORMAT REQUEST
          SA1    X2
          BX7    X1-X7            INSERT NEW FUNCTION
          SX1    X1
          EQ     PFM1
 PFM=     SPACE  4
          BASE   *
          IF     -DEF,QUAL$,2
          QUAL   *
 PFM=     EQU    /COMCPFM/PFM=
 PFM=     ENDX
COMCCDD
COMMON
 CDD      CTEXT  COMCCDD - CONSTANT TO DECIMAL DISPLAY CODE CONVERSION.
          BASE   DECIMAL
          IF     -DEF,QUAL$,1
          QUAL   COMCCDD
 CDD      SPACE  4,10
***       COMCCDD - CONSTANT TO DECIMAL DISPLAY CODE CONVERSION.
*         G. R. MANSFIELD.  11/13/69.
*         S. H. KEYSER.  07/21/73.
*         C. G. FILSTEAD.  10/04/74.  REWRITE.
 CDD      SPACE  4,10
***       CDD CONVERTS A POSITIVE INTEGER TO DECIMAL DISPLAY CODE WITH
*         LEADING ZERO SUPPRESSION.   THE INTEGER MULTIPLY INSTRUCTION
*         IS REQUIRED.   THE MICRO *MODEL* DEFINES THE TYPE OF CPU FOR
*         OPTIMIZATION PURPOSES:  *73* = 6400 TYPE,  *74* = 6600 TYPE,
*         AND *76* = 7600 TYPE.

          IF     -MIC,MODEL,1
 MODEL    MICRO  1,, 73

 ^?CDD"CP EQU    'MODEL'
 CDD      SPACE  4,10
***       CDD - CONSTANT TO DECIMAL DISPLAY CODE CONVERSION.
*
*         ENTRY  (B1) = 1.
*                (X1) = INTEGER TO BE CONVERTED.
*         EXIT   (X4) = CONVERSION LEFT ADJUSTED.
*                (X6) = CONVERSION RIGHT ADJUSTED.
*                (X7) = CONVERSION RIGHT ADJUSTED WITH LEADING ZEROS.
*                (B2) = 6*COUNT OF DIGITS IN (X4), (X6).
*
*         USES   X - 1, 2, 3.
*                B - 3, 4.
*                A - 2.
*
*         CALLS  NONE.


 CDD[CPU] IFGT   ^?CDD"CP,73

 CDD1     IX4    X1-X4       COMPUTE REMAINDER
          LX6    -6          SHIFT ASSEMBLY
          BX1    X3          SAVE QUOTIENT
          FX3    X2*X3       N = N/10
          IX7    X4-X7
          SB2    B2+B3       ADVANCE DIGIT COUNT
          BX6    X6+X7       ASSEMBLE DIGIT
          LX4    X3,B4       (N/10)*8
          FX7    X3+X3       (N/10)*2
          NZ     X1,CDD1     LOOP IF QUOTIENT ' 0

          SA2    A2+B1       =10H0000000000
          LX6    -6          LEFT ADJUST DIGITS
          SB3    B2-B1       SHIFT OFF (B2) ZEROS AND CHANGE TO 66-S
          IX7    X6+X2       CONVERT DIGITS
          AX1    X2,B3       CHANGE LEADING ZEROS TO BLANKS
          BX4    X7-X1
          LX7    X7,B2
          NO
          LX6    X4,B2       RIGHT ADJUST DIGITS

 CDD[CPU] ENDIF

 CDD      PS     0           ENTRY/EXIT
          SA2    CDDA        =1S48/10+1
          SB4    3

 CDD[CPU] IFGT   ^?CDD"CP,73
          FX3    X1*X2       N / 10
          SX6    B0          CLEAR ASSEMBLY
          SB2    B0          CLEAR DIGIT COUNT
          LX4    X3,B4       (N/10)*8
          IX7    X3+X3       (N/10)*2
          SB3    B4+B4
          EQ     CDD1

 CDD[CPU] ELSE

          BX6    X6-X6       CLEAR INITIAL ASSEMBLY
          SB2    0           CLEAR DIGIT COUNT
          SB3    B4+B4

 CDD1     BX3    X1          SAVE OLD VALUE
          FX1    X1*X2       N = N/10
          LX4    X1,B1       (N/10)*2
          LX7    X1,B4       (N/10)*8
          IX3    X3-X4       N-(N/10)*2
          IX3    X3-X7       N-(N/10)*2-(N/10)*8
          BX6    X6+X3       ASSEMBLE DIGIT
          LX6    -6          SHIFT ASSEMBLY
          SB2    B2+B3       ADVANCE DIGIT COUNT
          NZ     X1,CDD1     LOOP IF QUOTIENT ' 0

          SA2    A2+B1       =10H0000000000
          IX7    X2+X6       CONVERT DIGITS
          SB3    B2-B1       SHIFT OFF (B2) ZEROS AND CHANGE TO 66-S
          AX6    X2,B3
          BX4    X6-X7       REMOVE LEADING ZEROES
          LX6    X4,B2       RIGHT ADJUST DIGITS
          LX7    X7,B2
          EQ     CDD         RETURN

 CDD[CPU] ENDIF

 CDDA     CON    1S48/10+1
          CON    10H0000000000
          SPACE  4,6
          BASE   *
          IF     -DEF,QUAL$,2
          QUAL   *
 CDD      =      /COMCCDD/CDD
 CDD      ENDX
COMCWOD
COMMON
 WOD      CTEXT  COMCWOD - CONVERT WORD TO OCTAL DISPLAY CODE.
          IF     -DEF,QUAL$,1
          QUAL   COMCWOD
          BASE   D
 COMCWOD  SPACE  4
***       WOD - CONVERT WORD TO OCTAL DISPLAY CODE.
*         G. R. MANSFIELD.  12/13/69.
*         METHOD BY CHRIS WILLIS, MINN.
*         MODIFIED BY T. E. BOARD  10/18/73, NUCC.
*         REVISED - 11/29/75.  C. G. FILSTEAD.
 COMCWOD  SPACE  4
***               WOD CONVERTS A WORD TO OCTAL DISPLAY CODE BY AN
*         IN-LINE SEQUENCE OF SHIFTS AND MASKS.
*
*         ENTRY  (X1) = WORD TO CONVERT.
*                (B1) = 1.
*
*         EXIT   (X6,X7) = CONVERSION.
*
*         USES   X - 2, 3.
*                B - NONE.
*                A - 2.
*
*         CALLS  NONE.


 WOD      EQ     *+1S17      ENTRY/EXIT

*         (X1)   =           ABCDE FGHIJ KLMNO PQRST

          SA2    WODA        7.... 7.... 7.... 7....
          BX3    X2*X1       A.... F.... K.... P....
          LX1    3           BCDEF GHIJK LMNOP QRSTA

          BX6    X2*X1       B.... G.... L.... Q....
          LX1    3           CDEFG HIJKL MNOPQ RSTAB
          LX3    27          .K... .P... .A... .F...
          BX7    X2*X1       C.... H.... M.... R....

          LX6    21          ...L. ...Q. ...B. ...G.
          LX1    3           DEFGH IJKLM NOPQR STABC
          IX3    X3+X6       .K.L. .P.Q. .A.B. .F.G.
          BX6    X2*X1       D.... I.... N.... S....

          LX6    9           ..I.. ..N.. ..S.. ..D..
          BX3    X3+X6       .KIL. .PNQ. .ASB. .FDG.
          LX1    3           EFGHI JKLMN OPQRS TABCD
          BX6    X2*X1       E.... J.... O.... T....

          LX7    15          H.... M.... R.... C....
          BX3    X3+X7       HKIL. MPNQ. RASB. CFDG.
          LX6    3           ....J ....O ....T ....E
          BX3    X3+X6       HKILJ MPNQO RASBT CFDGE

          SA2    A2+B1       .7.7. 7.7.7 ..... .....
          BX7    X2*X3       .K.L. M.N.O ..... .....
          LX2    -15         ..... .7.7. 7.7.7 .....
          BX6    X2*X3       ..... .P.Q. R.S.T .....

          LX6    -15         ..... ..... .P.Q. R.S.T
          IX7    X7+X6       .K.L. M.N.O .P.Q. R.S.T
          LX3    30          RASBT CFDGE HKILJ MPNQO
```

```
          BX6    X2*X3       ..... .F.G. H.I.J .....

          LX2    15          .7.7. 7.7.7 ..... .....
          BX3    X2*X3       .A.B. C.D.E ..... .....
          SA2    A2+B1       33333 33333 33333 33333
          IX7    X7+X2

          LX6    -15         ..... ..... .F.G. H.I.J
          BX3    X3+X6       .A.B. C.D.E .F.G. H.I.J
          IX6    X3+X2
          LX1    -12         ABCDE FGHIJ KLMNO PQRST

          EQ     WOD         RETURN

 WODA     DATA   7000070000070000070000B
          DATA   0707070707000000000B
          DATA   10H0000000000
 COMCWOD  SPACE  4
          BASE   *
          IF     -DEF,QUAL$,2
          QUAL   *
 WOD      EQU    /COMCWOD/WOD
 WOD      ENDX
COMCSFN
COMMON
 SFN      CTEXT  COMCSFN - SPACE FILL NAME.
          IF     -DEF,QUAL$,1
          QUAL   COMCSFN
 SFN      SPACE  4
***       SFN - SPACE FILL NAME.
*         G. R. MANSFIELD.  12/08/69.
*         ADAPTED FROM ROUTINE DEVELOPED AT PURDUE UNIVERSITY.
*         REVISED - 03/27/74.  C. G. FILSTEAD.
 SFN      SPACE  4
***              SFN APPENDS SPACE CODES (55) TO A WORD.
*
*         ENTRY  (X1) = NAME LEFT JUSTIFIED, ZERO FILL.
*                (B1) = 1.
*
*         EXIT   (X6) = NAME SPACE FILLED.
*
*         USES   A2, X2.
*
*         CALLS  NONE.
*

 SFN      EQ     *+1S17      ENTRY/EXIT
          SA2    =4040404040404040404040B

*         EXAMPLE:  LET X1 = 0102 0304 0500 0000 0000

          MX6    -1
          IX6    X1+X6       0102 0304 0477 7777 7777

          BX6    -X6+X1      7777 7777 7700 0000 0000
          BX2    -X6*X2      0000 0000 0040 4040 4040
          BX6    X2
          LX6    -3          0000 0000 0004 0404 0404

          BX2    X2+X6       0000 0000 0044 4444 4444
          BX6    X1+X2       0102 0304 0544 4444 4444
          LX2    -2          0000 0000 0011 1111 1111
          BX6    X6+X2       0102 0304 0555 5555 5555

          EQ     SFN         RETURN
 SFN      SPACE  4
          IF     -DEF,QUAL$,2
          QUAL   *
 SFN      =      /COMCSFN/SFN
 SFN      ENDX
COMCCOD
COMMON
 COD      CTEXT  COMCCOD - CONVERT OCTAL DIGITS.
          BASE   DECIMAL
          IF     -DEF,QUAL$,1
          QUAL   COMCCOD
 COD      SPACE  4
***       COD - CONVERT OCTAL DIGITS.
*         G. R. MANSFIELD.  12/05/69.
*         S. H. KEYSER.  01/20/74.  (6400)
*         ADAPTED FROM SUBROUTINE -COD- IN LIBEDIT.
*         REVISED - 11/29/75.  C. G. FILSTEAD.
 COD      SPACE  4
***              COD CONVERTS UP TO 10 DIGITS TO DISPLAY CODE WITH
*         LEADING ZERO SUPPRESSION.  CONVERSION CONTAINS SPACE FILL
*         WITH BOTH RIGHT AND LEFT JUSTIFICATION PROVIDED.
*
*         THE MICRO *MODEL* DEFINES THE TYPE OF CPU FOR OPTIMIZATION.
*         *73* = 6400, *74* = 6600, *76* = 7600.



          IF     -MIC,MODEL,1
 MODEL    MICRO  1,, 73

 CPU      EQU    'MODEL'
 COD      SPACE  4,10
***       COD - CONVERT OCTAL DIGITS.
*
*         ENTRY  (X1) = NUMBER TO BE CONVERTED.
*                (B1) = 1.
*
*         EXIT   (X4) = DPC CONVERSION, LEFT JUSTIFIED.
*                (X6) = DPC CONVERSION, RIGHT JUSTIFIED.
*                (B2) = 6*COUNT OF DIGITS IN X6.
*
*         USES   X - 1, 2, 3, 7.
*                B - 3, 4.
*                A - 2.
*
*         CALLS  NONE.
*

 COD[CPU] IFGT   CPU,73

 COD1     AX1    3           N = N/8
          IX6    X3+X7       ASSEMBLE PREVIOUS DIGIT
          BX7    -X4*X1      MASK NEXT DIGIT
          SB3    B3+B2       COUNT (6*DIGITS)-1
          LX3    X6,B4       SHIFT ASSEMBLY
          NZ     X1,COD1     LOOP IF QUOTIENT ' 0

 COD[CPU] ELSE

 COD1     BX7    -X4*X1      MASK NEXT DIGIT
          AX1    3           N = N/8
          BX3    X3+X7       ASSEMBLE DIGIT
          LX3    -6          SHIFT ASSEMBLY
          SB3    B3-B2       COUNT (6*DIGITS)-1
          NZ     X1,COD1     LOOP IF QUOTIENT ' 0

 COD[CPU] ENDIF

          AX7    X2,B3       FORM (10-DIGITS) LOW ORDER 66-S
          IX3    X3+X2       CONVERT TO DISPLAY CODE
          BX4    X3-X7       REMOVE LEADING ZEROS
          SB2    B3+B1       (B2) = 6*DIGITS IN (X4), (X6)
          LX6    X4,B2       RIGHT JUSTIFY

 COD      EQ     *+1S17      ENTRY/EXIT

          SA2    =10H0000000000
          BX3    X3-X3       CLEAR ASSEMBLY
          MX4    -3          1 DIGIT MASK

 COD[CPU] IFGT   CPU,73

          SB3    -B1         6*DIGITS-1
          SB4    60-6
          BX7    -X4*X1      MASK FIRST DIGIT
          SB2    6

 COD[CPU] ELSE

          SB3    -B1         6*DIGITS-1
          SB2    X4+B1       -6

 COD[CPU] ENDIF

          EQ     COD1        BEGIN
 COD      SPACE  4
          BASE   *
          IF     -DEF,QUAL$,2
          QUAL   *
 COD      =      /COMCCOD/COD
 COD      ENDX
COMCSTT
COMMON
 STT      CTEXT  COMCSTT - SORT TWO WORD ENTRIES ON SECOND WORD.
          IF     -DEF,QUAL$,1
          QUAL   COMCSTT
          SPACE  4
***       COMCSTT - SORT TWO WORD ENTRIES ON SECOND WORD OF ENTRY.
*         L. R. ATKIN.  07/21/73.
 STT      SPACE  4
***              STT USES THE QUICKSORT METHOD (BY HOARE) TO SORT A
*         TABLE OF TWO WORD ENTRIES SUCH THAT THE SECOND WORDS ARE
*         IN ASCENDING NUMERICAL SEQUENCE.
*
*         ENTRY  (X6) = FWA OF TABLE.
*                (X7) = LWA+1 OF TABLE.
*                (B4) = FWA OF SCRATCH AREA.
*                       SCRATCH AREA MUST BE AT LEAST 2*LOG2(N),
*                       WHERE N IS THE NUMBER OF ENTRIES IN THE TABLE.
*                (B1) = 1.
*
*         USES   ALL REGISTERS EXCEPT A0.
*



 STT      SPACE  4
**        ASSEMBLY CONSTANT.
*
*         STT[CCS] SELECTS THE COLLATION SEQUENCE TO BE USED:
*
*                IF STT[CCS] IS DEFINED, THE CHARACTER COLLATION
*         SEQUENCE WILL BE USED.  IF STT[CCS] IS NOT DEFINED, THE
*         COLLATION SEQUENCE WILL BE THAT OF SIGNED INTEGERS.
*         WHEN STT[CCS] IS DEFINED, STT[XPR] IS IRRELAVANT.
*         DEFINITION OF STT[CCS] WILL SOMEWHAT DEGRADE THE
*         SPEED OF STT.
*
*         STT[XPR] SHOULD BE DEFINED FOR 60 BIT COMPARISONS.
*                IF STT[XPR] IS NOT DEFINED, STT WILL FAIL IF ANY
*                COMPARISON OF TWO KEYS YIELDS A DIFFERENCE GREATER
*                THAN 2**59-1.  THIS WILL OCCUR IF THE TABLE CONTAINS
*                INTEGERS GREATER THAN 2**58-1 AND LESS THAN -2**58+1,
*                OR FLOATING POINT NUMBERS GREATER THAN 2**47-1 AND
*                LESS THAN -2**47+1.
*
*                IF STT[XPR] IS DEFINED, STT WILL ACCURATELY SORT
*                ANYTHING, BUT WILL EXECUTE ABOUT 25 PERCENT LONGER.
*



 STT      EQ     *+400000B

          SB5    B4                INITIALIZE STACK POINTER
          SB6    B1+B1             CONSTANT 2
          SA6    B4                INITIALIZE STACK
          SA7    B4+B1

 STT1     LT     B5,B4,STT         IF STACK EMPTY, RETURN
          SA1    B5                FWA OF PARTITION
          SA2    B5+B1             LWA+1 OF PARTITION
          SB7    X2-2              ADDRESS OF LAST ENTRY
          IX6    X1-X2             -LENGTH OF PARTITION
          SX6    X6+B6
          NG     X6,STT2           IF NOT MINIMAL PARTITION
          SB5    B5-2              POP THE STACK
          EQ     STT1

*         NON-MINIMAL PARTITION.  CHOSE LAST ENTRY AS BOUND.

 STT2     SA5    B7+B1             BOUND
          SB3    X1                PROBE
          SA4    A5-B1             OTHER WORD
          BX0    X4                SAVE OTHER WORD

*         MOVE BOTTOM PROBE UP.

 STT3     GE     B3,B7,STT5        IF PARTITION COMPLETE.
          SA3    B3+B1
          IX6    X5-X3
          IF     DEF,STO[CCS],5
          BX4    X5-X3       + IF SIGNS THE SAME, - IF SIGNS DIFFER
          BX6    -X4*X6      IF SAME SIGN TAKE SIGN OF DIFFERENCE
          BX4    -X5*X4      IF SIGNS DIFFER TAKE SIGN OF SUBTRAHAND
          BX6    X4+X6       USE PROPER SIGN
          SKIP   5
          IF     DEF,STT[XPR],4
          BX4    X3-X5             NEGATIVE IF SIGNS DIFFER
          BX6    -X4*X6            IGNORE SUBTRACT IF SIGNS DIFFER
          BX4    X4*X5             USE SIGN OF PROBE IF SIGNS DIFFER
          BX6    X4+X6
```

```
          SB3    B3+B6             ADVANCE PROBE
          PL     X6,STT3           IF PROBED ENTRY LOWER THAN BOUND
          BX6    X3                MOVE HIGH ENTRY UP
          SA4    A3-B1
          BX7    X4                MOVE OTHER WORD
          SA7    B7
          SA6    A7+B1

*         MOVE HIGHER PROBE DOWN.

 STT4     GE     B3,B7,STT5        IF PARTITION COMPLETE.
          SA3    B7-B1
          IX6    X3-X5
          IF     DEF,STT[CCS],5
          BX4    X3-X5       + IF SIGNS THE SAME, - IF SIGNS DIFFER
          BX6    -X4*X6      IF SAME SIGN TAKE SIGN OF DIFFERENCE
          BX4    -X3*X4      IF SIGNS DIFFER TAKE SIGN OF SUBTRAHAND
          BX6    X4+X6       USE PROPER SIGN
          SKIP   5
          IF     DEF,STT[XPR],4
          BX4    X3-X5
          BX6    -X4*X6
          BX4    X4*X3
          BX6    X4+X6
          SB7    B7-B6
          PL     X6,STT4           IF PROBED ENTRY HIGHER THAN BOUND
          BX6    X3
          SA4    A3-B1             MOVE OTHER WORD
          BX7    X4
          SA6    B3-B1
          SA7    A6-B1
          EQ     STT3

*         PARTITIONING COMPLETE.  STORE BOUND IN FINAL SLOT.

 STT5     BX6    X5                STORE BOUND
          BX7    X0
          SA6    A4+B1
          SA7    A4

*         STACK REMAINING PARTITIONS, LONGEST FIRST.

          SX6    A4                LWA+1 OF FIRST PARTITION
          IX3    X6-X1             LENGTH OF FIRST
          SX7    A4+B6             FWA OF SECOND PARTITION
          IX4    X2-X7             LENGTH OF SECOND
          IX5    X3-X4
          SB5    B5+B6             ADVANCE STACK POINTER
          PL     X5,STT6           IF FIRST LONGER
          SA7    A1
          BX7    X1
          SA7    A7+B6
          SA6    A7+B1
          EQ     STT1

 STT6     SA6    A2
          BX6    X2
          SA7    A6+B1
          SA6    A7+B1
          EQ     STT1
          SPACE  4
          IF     -DEF,QUAL$,1
          QUAL   *
 STT      =      /COMCSTT/STT
 STT      ENDX
COMCSVR
COMMON
 SVR=     CTEXT  COMCSVR - SAVE/RESTORE REGISTERS.
          IF     -DEF,QUAL$,1
          QUAL   COMCSVR
          BASE   D
 SVR=     SPACE  4
***       COMCSVR - SAVE / RESTORE REGISTERS.
*
*
*                -0 IS PRESERVED IN ALL REGISTERS.  ROUTINES BLENDED
*         FROM COMPASS, FTN EXT, *DDT* AND OTHER SOURCES.
*
*
*         THE REGISTER SAVE AREA IS IN THE FOLLOWING FORMAT:
*
* X.REG   BSS    8           SAVED X-REGISTER CONTENTS
* B.REG   BSS    8           SAVED B-REGISTER CONTENTS
* A.REG   BSS    8           SAVED A-REGISTER CONTENTS
*
*         THE *A* AND *B* REGISTERS HAVE SIGN EXTENDED TO 60 BITS.
*
 SVR=     SPACE  4,8
***       SVR - SAVE ALL REGISTERS A,B, AND X
*
*
*+        RJ     =XSVR=
*-        VFD    30/RSA
*
*         ENTRY  (RSA) = 24D WORD REGISTER SAVE AREA.
*
*         EXIT   (B1) = 1.
*

 SVR=     EQ     *+4S15      ENTRY/EXIT...

 B=1      MICRO  1,, B0
          ECHO   4,BN=(B1,B7,B5,B2,B3,B4,B6)
          IF     DEF,BN=1,3
 B=1      MICRO  1,, BN
          STOPDUP
 B=1      SKIP

 SVR1     BSS
          PL     B1,*+2
 +        RJ     *
 B.NE.1   DUP    17          SAVE (B1) THE HARD WAY
 +        SB1    B1+B1
          NO
          PL     B1,*+2
 +        RJ     *           MARK BIT AS SET
 B.NE.1   ENDD

 B=1      ELSE
          MI     'B=1',*+4S15      IF NEGATIVE (INCLUDING -0)
          S'B=1' 'B=1'-1
          ZR     'B=1',SVR1  IF ('B=1') STILL  = 1
          S'B=1' 'B=1'+1
          EQ     *+4S15      BLOWUP WITH ORIGINAL VALUE IN ('B=1')

 SVR1     BSS
          IFC    NE, 'B=1' B1 ,1
          S'B=1' B1-B0       REMEMBER (B1)
 B=1      ENDIF

          SB1    A7-B0       REMEMBER (A7)
          SA7    SVRX7       SAVE (X7) TEMP
          SX7    A5-B0
          SA7    SVRA5       SAVE (A5) TEMP
          BX7    X5
          SA7    SVRX5       SAVE (X5) TEMP

          SA5    SVR=
          LX5    30
          SA5    X5-1        FETCH RJ WORD
          SX7    B1-B0
          SB1    1           SET (B1) = CONSTANT 1
          SA7    X5+3*8-1    SAVE (A7)
          SX7    A6-B0
          SA7    A7-B1       SAVE (A6)
          SA6    A7-2*8      SAVE (X6)
          SA5    SVRA5
          BX7    X5
          SA5    A5+B1
          BX6    X5
          SA7    A7-B1       SAVE (A5) FINALLY
          SA6    A6-B1       SAVE (X5) FINALLY

 SV=AX    ECHO   ,R=(4,3,2,1,0)
          SX7    A.R-B0
          BX6    X.R
          SA7    A7-B1       SAVE (A.R)
          SA6    A6-B1       SAVE (X.R)
 SV=AX    ENDD

          SA5    SVRX7
          BX6    X6-X6
          LX7    X5
          SA6    A6+8        SAVE (B0), FOR CONSISTENCY
          SA7    A6-B1       SAVE (X7) FINALLY

 B=1      IFC    NE, 'B=1' B0
          SX7    'B=1'-B0
          S'B=1' B1          RESET ('B=1') = 1
 B=1      ELSE
          SX7    B1
 B=1      ENDIF
          SA7    A6+B1       SAVE (B1) TENATIVELY

 SV=B     ECHO   ,U=(2,4,6),L=(3,5,7)
          SX6    B.U-B0
          SX7    B.L-B0
          SA6    A7+B1       SAVE (B.U)
          SA7    A6+B1       SAVE (B.L)
 SV=B     ENDD

 B.NE.1   IFC    EQ, 'B=1' B0
          SX4    0100B       *RJ*
          SA1    SVR1+1
          SX2    B1
          LX4    18
          SB2    18
          BX7    X7-X7

 SVR2     LX1    3
          SX3    A1
          IX7    X7+X7
          BX6    X4+X3
          LX6    30
          PL     X1,SVR3     IF 2**(B2-1) OF ORIGINAL (B1) WAS CLEAR
          BX7    X7+X2
 SVR3     SB2    B2-B1
          SA6    A1          RESET FOR NEXT TIME
          SA1    A1+2
          NZ     B2,SVR2     IF MORE BITS TO GO

          LX7    42
          AX7    42          SIGN EXTEND
          SA7    A7-7+1      SAVE (B1) FINALLY
 B.NE.1   ENDIF

          EQ     SVR=        EXIT..


 RSRA1    BSS    0           TEMP SAVE (A1)
 SVRX7    BSS    1           TEMP SAVE (X7)
 SVRA5    BSS    1           TEMP SAVE (A5)
 SVRX5    BSS    1           TEMP SAVE (X5)
 RSR=     SPACE  4,8
***       RSR - RESTORE ALL REGISTERS A,B, AND X.
*
*
*         ENTRY  (X1) = ADDRESS OF REGISTER SAVE AREA.
*                (B1) = 1.
*
*         EXIT   ALL REGISTERS RESTORED.
*
*         CALLS  IDL=.
*

 RSR3     BSS    0
          ECHO   1,BN=(B1,B2,B3,B4,B5,B6,B7,A0)
          S;A    A0+**       RESTORE (BN)

 RSR=     EQ     *+4S15      ENTRY/EXIT...
          MX0    42          SET UP RESTORE OF (B1-B7,A0)
          SB6    B0
          SB7    -6
          SB5    X1
          SB3    X1+8+7      ! (B7)
          RJ     =XIDL=      IDLE THE CONTROL POINT

 RSR2     SX6    607B+B7
          SX7    X6+B1
          LX6    30
          SA1    B3+B7       FETCH (B(I))
          BX6    X6+X7
          LX6    21          (SB1 A0+0, SB2 A0+0), (B3,B4), (B5,B6)
          SA2    A1+B1
          BX3    -X0*X1
          BX4    -X0*X2
          LX3    30
          IX7    X3+X4
          BX6    X6+X7
          SB7    B7+2
          SB6    B6+B1
          SA6    B6+RSR3-1
          MI     B7,RSR2     (-4, -2, 0)
          SA4    B5+2*8      (A0)
```

```
          SA1    A4+B1       (A1)
          BX7    X1
          SA7    RSRA1       REMEMBER (A1)
          SA3    A2+B1       (B7)
          BX1    -X0*X3
          SA3    RSRA
          BX2    -X0*X4
          LX1    30
          IX7    X2+X3
          BX6    X7+X1       (SB7 A0+0, SA0 A0+0)
          SA6    A6+1

          SA1    B5+2*8+6    (A6)
          SA2    A1+B1       (A7)
          SA5    B5          (X0)
 +        SA3    X1
          BX6    X3          CONTENTS OF (A6)
          SA6    X1-0        RESTORE (A6)
 +        SA4    X2
          BX7    X4          CONTENTS OF (A7)
          SA7    X2-0        RESTORE (A7)

          SA2    B5+2*8+2    (A2)
          BX0    X5          RESTORE (X0)
          SA3    A2+B1
          SA4    A3+B1
          SA5    A4+B1
          ECHO   1,R=(2,3,4,5)
          SA.R   X.R-0       RESTORE (A.R)

          SA1    B5+2        (X2)
          BX2    X1          RESTORE (X2)
          ECHO   2,R=(3,4,5,6,7)
          SA1    A1+B1
          BX.R   X1          RESTORE (X.R)

          MX1    60
          SA0    X1-0        (A0) = 777 777 B
          SA1    B5+B1       (X1)

 UX1      ECHO   ,R=(2,3,4,5,6,7)
          UX1,B.R
          LX1    10
 UX1      ENDD

          SB1    A0-B0
          SB1    X1+B1       REMEMBER SIGN OF (X1)
          SA1    RSRA1
          SA1    X1-0        RESTORE (A1)

          SX1    B1-B0
          PX1    B7
 PX1      ECHO   ,R=(6,5,4,3,2)
          LX1    -10
          PX1    B.R
 PX1      ENDD
          EQ     RSR3        A 40 WORD STACK WOULD MAKE THIS DANGEROUS

 RSRA     SB7    A0+**
          SA0    A0+**
 SVR=     SPACE  4
          BASE   *
          IF     -DEF,QUAL$,3
          QUAL   *
 SVR=     EQU    /COMCSVR/SVR=
 RSR=     EQU    /COMCSVR/RSR=
 SVR=     ENDX
COMCIDL
COMMON
 IDL=     CTEXT  COMCIDL - IDLE CONTROL POINT.
          IF     -DEF,QUAL$,1
          QUAL   COMCIDL
 IDL=     SPACE  4
***       COMCIDL - IDLE CONTROL POINT.
*         S. H. KEYSER.  03/07/74.
*         METHOD BY L. R. ATKIN AND D. J. SLATE.
 IDL=     SPACE  4
***              IDL IDLES THE CONTROL POINT.  THIS ENSURES THAT ALL
*         PENDING REQUESTS ARE COMPLETED.
*
*         ENTRY  (B1) = 1.
*
*         EXIT   THE CONTROL POINT IS IDLE (PP ACTIVITY HAS CEASED).
*                (X6) = RPV RETURN CODE, NON-ZERO IF ERRORS OCCURED AS
*                       CONTROL POINT WAS IDLEING DOWN.
*                (X7) = ADDRESS OF EXCHANGE PACKAGE WHEN ERRORS OCCURED.
*
*         USES   X - 1.
*                B - NONE.
*                A - 1, 6.
*
*         CALLS  GRS=, RPV= AND RRS=.
*
*         NEEDS  ENDRUN.
*
*         METHOD:
*                1.  CURRENT RPV STATUS IS SAVED.
*                2.  RPV STATUS IS SET TO REPRIEVE ON TERMINATION.
*                3.  CPU ACCESS IS TERMINATED.
*                4.  (WHEN CONTROL POINT ACTIVITY CEASES, RPV IS CALLED
*                    TO RESTART THE CPU PROGRAM.)
*                5.  PREVIOUS RPV STATUS IS RESTORED.
*
*         NOTE   AS A NEW CHECKSUM OF THE USERS RECOVERY CODE WILL BE
*                TAKEN, THE CALLING PROGRAM MUST VERIFY ITS INTEGRITY
*                BEFORE CALLING IDL.
*

 IDLA     BSSZ   2           SAVED REPRIEVE STATUS AREA
 IDLB     BSSZ   17D         EXCHANGE PACKAGE AREA

*         RESTART CONTROL POINT.

          SA1    IDLA
          RJ     =XRRS=      RESTORE REPRIEVE STATUS
          SA1    IDLB        PICK UP RPV RETURN CODE IN B0
          SX6    X1
          SX7    A1

 IDL=     PS     0           ENTRY/EXIT
          SX7    IDLA
          RJ     =XGRS=      GET REPRIEVE STATUS
          BX6    X6-X6
          NO
          SA6    IDLB        CLEAR STATUS WORD
          SX7    17700B      REPRIEVE ON ALL CONDITIONS
          RJ     =XRPV=      ISSUE REPRIEVE REQUEST
          ENDRUN             DROP CPU
 IDL=     SPACE  4
          IF     -DEF,QUAL$,2
          QUAL   *
 IDL=     =      /COMCIDL/IDL=
 IDL=     ENDX
COMCRPV
COMMON
 RPV=     CTEXT  COMCRPV - PROCESS REPRIEVE REQUEST.
          IF     -DEF,QUAL$,1
          QUAL   COMCRPV
 RPV=     SPACE  4
***       COMCRPV - PROCESS REPRIEVE REQUEST.
*         S. H. KEYSER.  03/13/74.
 RPV=     SPACE  4
***              COMCRPV CONTAINS ROUTINES FOR PROCESSING CERTAIN
*         REPRIEVE REQUESTS.  AN NUCC MODIFIED RPV IS USED WHICH
*         PROVIDES FOR AN ALTERNATE RECOVERY SEQUENCE.
*
*         (1)    STATUS  CON  0    REPRIEVE, NO CHECKSUM TAKEN.
*                        BSS  15D  EXCHANGE PACKAGE SAVE AREA.
*                        BSS  1    (RA+1) SAVE AREA.
*                CODE    ...       RECOVERY ROUTINE ENTRY.
*
*         (2)    STATUS  VFD  12/,18/LWA,30/  REPRIEVE, CHECKSUM OF
*                        CON  **   AREA STATUS+17 ! LWA KEPT HERE.
*                        BSS  14D  REST OF EXCHANGE PACKAGE BUFFER.
*                        BSS  1    (RA+1) SAVE AREA.
*                CODE    ...       RECOVERY CODE.
*                LWA     BSS  0    LWA RECOVERY ROUTINE.
*
*        *(3)    STATUS  VFD  42/0LFN,18/     LIBRARY LOAD FROM FILE
*                                             -FN- ON REPRIEVE.
*
*        * DENOTES NUCC NON-STANDARD CALLING FORM.
*

 EQUATES  SPACE  4
**        ASSEMBLY CONSTANTS - RPV INFORMATION IN CONTROL POINT AREA.

 W.CPRPV  CEQU   43B         RPV WORD
 C.CPRPV  CEQU   1           RPV FLAG BYTE IN W.CPRPV
 S.LBRPV  CEQU   10D         ESP LOAD FLAG IN C.CPRPV
 C.CPRPA  CEQU   2           RPV RECOVERY ADDRESS BYTE IN W.CPRPV
 W.CPESP  CEQU   177B        ESP FILE NAME WORD
 GRS=     SPACE  4,21
***       GRS - GET REPRIEVE STATUS.
*
*         ENTRY  (X7) = ADDRESS OF 2 WORD REPRIEVE STATUS SAVE AREA.
*                (B1) = 1.
*
*         EXIT   CURRENT RPV STATUS INFORMATION STORED IN SAVE AREA FOR
*                SUBSEQUENT CALL TO RESTORE REPRIEVE STATUS ROUTINE.
*                ((X7)) = (W.CPESP) BITS 59-17 ONLY.
*                ((X7)+1) = (W.CPRPV) SHIFTED SO THAT S.LBRPV IN C.CPRPV
*                           (ESP LOAD FLAG) IS JUSTIFIED IN BIT 59.
*
*         USES   A1, X1, A6, X6.
*
*         NEEDS  CPAREA.
*

 GRS=     PS     0           ENTRY/EXIT
          CPAREA W.CPESP
          MX1    42D         CLEAR BITS 0-17
          BX6    X1*X6
          SA6    X7+
          CPAREA W.CPRPV
          LX6    59D+12D+12D*C.CPRPV-S.LBRPV-60D
          SA6    X7+B1
          EQ     GRS=        RETURN
 RPV=     SPACE  4,27
***       RPV - ISSUE REPRIEVE REQUEST.
*
*         ENTRY  (A6) = ADDRESS OF RPV STATUS WORD.
*                (X7) = RPV ERROR CLASS BITS.
*                (B1) = 1.
*
*         EXIT   REQUEST PROCESSED.
*
*         USES   A1, X1, A6, X6 AND X7.
*
*         CALLS  SYS=.
*

 RPV1     LX1    42D
          BX6    X1+X6
          RJ     =XSYS=      ISSUE REQUEST

 RPV=     PS     0           ENTRY/EXIT
          SX6    A6          FORM SYSTEM REQUEST
          LX7    18D
          SX1    3RRPV
          BX6    X6+X7
          PX1    X1
          EQ     RPV1
 RRS=     SPACE  4,31
***       RRS - RESTORE REPRIEVE STATUS.
*
*         ENTRY  (A1) = ADDRESS OF 2 WORD SAVED REPRIEVE STATUS AREA.
*                (X1) = ((A1)).
*                (B1) = 1.
*
*         EXIT   REPRIEVE STATUS RESTORED.
*
*         USES   A1, X1, A6, X6 AND X7.
*
*         CALLS  RPV=.
*

 RRS2     LX1    24D+12D*C.CPRPA-12D-12D*C.CPRPV-59D+S.LBRPV+60D
          MX6    -17D
          BX1    -X6*X1      EXTRACT OLD RECOVERY ADDRESS
          SA1    X1          PICK UP OLD STATUS WORD
          BX6    X6*X1       CLEAR COMPLETE BIT
          SA6    A1          RESET STATUS WORD
          RJ     =XRPV=

 RRS=     PS     0           ENTRY/EXIT
          MX7    7
          BX6    X1
          SA1    A1+B1       RETRIEVE W.CPRPV
          LX7    60D-12D+7+59D-S.LBRPV+12D-60D
          BX7    X7*X1       EXTRACT OLD ERROR CLASS BITS
          LX7    18D-59D-12D+S.LBRPV+60D
          ZR     X6,RRS1     NO ESP FILENAME TO RESTORE
          SA6    A1-B1
```

```
          RJ     =XRPV=      RESTORE ESP FILENAME
          SA1    X6+B1       RESTORE X1
          LX7    60D-18D     RESTORE X7
          NG     X1,RRS=     DONE IF ESP CALL

 RRS1     NZ     X7,RRS2     CONTINUE IF REPRIEVE TO RESTORE
          EQ     RRS=        RETURN
 RPV=     SPACE  4,6
          IF     -DEF,QUAL$,4
          QUAL   *
 RPV=     =      /COMCRPV/RPV=
 GRS=     =      /COMCRPV/GRS=
 RRS=     =      /COMCRPV/RRS=
 RPV=     ENDX
COMCDXB
COMMON
 DXB      CTEXT  COMCDXB - DISPLAY CODE TO BINARY CONVERSION.
          IF     -DEF,QUAL$,1
          QUAL   COMCDXB
          BASE   D
 COMCDXB  SPACE  4
***       DXB - DISPLAY CODE TO BINARY CONVERSION.
*         G. R. MANSFIELD.  12/03/69.
 COMCDXB  SPACE  4
***               DXB CONVERTS ONE WORD OF DISPLAY CODE DIGITS TO
*         A BINARY VALUE.  CONVERSION MAY BE IN OCTAL OR DECIMAL BY
*         THE FOLLOWING CONDITIONS.
*         (1.)  ASSUMED BASE SPECIFIED BY CALLER.
*         (2.)  POST RADIX SPECIFICATION.  (B = OCTAL, D = DECIMAL)
*         (3.)  PRESENCE OF 8 OR 9 WILL FORCE DECIMAL IF NO *B* POST
*                RADIX.
*         ERROR CONDITIONS SENSED ARE -
*         (1.)  PRESENCE OF NON-DIGIT IN WORD.
*         (2.)  PRESENCE OF CHARACTER AFTER POST RADIX.
*         (3.)  PRESENCE OF 8 OR 9 WITH POST RADIX = B.
*
*
*         ENTRY  (X5) = WORD TO CONVERT.
*                (B7) NON-ZERO IF DECIMAL BASE ASSUMED.
*                (B1) = 1.
*         EXIT   (X6) = CONVERTED DIGITS.
*                (X4) ' 0 IF ERROR IN ASSEMBLY.
*
*         USES   X - 0, 1, 2, 3, 4, 5, 6, 7.
*                B - 2, 3, 4, 5.
*                A - NONE.
*
*         CALLS  NONE.


 DXB1     LX2    X7,B2       DECIMAL * 10
          IX7    X2+X7
          LX6    3           OCTAL * 8
          BX2    -X3*X1      8/9 PRESENCE
          LX7    1
          BX6    X6+X1       OCTAL + NEW DIGIT
          IX7    X7+X1       DECIMAL + NEW DIGIT
          SB5    B5+X2       NOTE 8/9
 DXB2     LX5    6           NEXT CHARACTER
          BX1    -X0*X5
          SB4    X1          CHECK CHARACTER
          LX2    X4,B4
          SX1    X1+B3       CONVERT CHARACTER
          BX5    X0*X5       CLEAR CHARACTER
          NG     X2,DXB1     LOOP IF DIGIT

**        CHECK FOR POST RADIX SPECIFICATION.

          SB3    B5+B7       SET BASE (ASSUMED OR 8/9)
          ZR     B4,DXB3     IF END OF ASSEMBLY
          SB3    B2+B2       BASE = DECIMAL
          NZ     X5,DXB      RETURN IF NOT LAST CHARACTER
          EQ     B4,B3,DXB3  IF *D*
          SB5    B4-B5
          NE     B5,B2,DXB   RETURN IF NOT *B* OR *B* AND 8/9 PRESENT
          SB3    B0          SET OCTAL
 DXB3     MX4    0           CLEAR ERROR
          ZR     B3,DXB      RETURN IF BASE = OCTAL
          BX6    X7          SET DECIMAL


 DXB      PS                 ENTRY/EXIT
          SX4    7774B       MASK FOR #0123456789          MX0    -6         (X
          SX6    B0          CLEAR OCTAL
          SB2    B1+B1       (B2) = 2
          BX7    X7-X7       CLEAR DECIMAL
          SB3    -1R0        (B3) = CONVERSION CONSTANT
          LX4    21
          SB5    B0          CLEAR 8/9 PRESENCE
          SX3    7           MASK FOR 8/9
          EQ     DXB2        ENTER CONVERSION LOOP
          SPACE  4
          BASE   *
          IF     -DEF,QUAL$,2
          QUAL   *
 DXB      EQU    /COMCDXB/DXB
 DXB      ENDX
COMCWTH
COMMON
 WTH=     CTEXT  COMCWTH - WRITE CODED LINE, -H- FORMAT.
          BASE   D
          IF     -DEF,QUAL$,1
          QUAL   COMCWTH
 WTH      SPACE  4
***       WTH - WRITE CODED LINE, -H- FORMAT, WITH TRAILING
*         SPACE SUPPRESSION.
*
*         ENTRY  (X2) = ADDRESS OF FET FOR FILE.
*                (B6) = FWA WORKING BUFFER.
*                (B7) = WORD COUNT OF WORKING BUFFER.
*                IF (B7) = 0, NO TRANSFER WILL BE PERFORMED.
*         EXIT   (X2) = ADDRESS OF FET FOR FILE.
*
*         USES   ALL REGISTERS EXCEPT A0, X0, A5, X5.
*         CALLS  DCB=, WTX=.


 +        EQ     WTH3

 WTH=     EQ     *+400000B   ENTRY/EXIT
          SA4    *-1
          ZR     B7,WTH=     IF WORKING BUFFER EMPTY
          IF     -DEF,B1=1,1
          SB1    1

*         DELETE TRAILING BLANK WORDS.

          SA3    WTHA        =1H
          SA1    B6+B7       PRESET (A1)
          SB7    B7+B1
 WTH1     SA1    A1-B1
          IX6    X1-X3
          SB7    B7-B1
          EQ     B7,B1,WTH2
          ZR     X6,WTH1
 WTH2     SA1    X2+4        (B5) = LIMIT
          SA3    X2+B1       (X3) = FIRST
          SB5    X1
          MX4    0
          NE     B7,B1,WTH3  IF NOT FIRST WORD
          MX4    12

*         INITIALIZE REGISTERS FOR TRANSFER.

 WTH3     SA1    A3+2        (B4) = OUT
          SA2    A3+B1       (X2) = IN
          SB4    X1

*         TRANSFER DATA FROM WORKING BUFFER TO CIRCULAR BUFFER.

 WTH4     SB3    X2+B1       (IN+1)
          NE     B3,B5,WTH5  IF (IN+1) ' LIMIT
          SB3    X3          (IN+1) = FIRST
 WTH5     EQ     B3,B4,=XDCB= DUMP CIRCULAR BUFFER IF (IN+1) = OUT
          SA1    B6          READ WORD
          SB7    B7-B1       DECREMENT WORD COUNT
          BX6    X1
          SA6    X2          STORE WORD
          SB6    B6+B1       ADVANCE WORKING BUFFER
          SX2    B3          IN = (IN+1)
          GE     B7,B1,WTH4  LOOP TO LAST WORD

          MX1    48          CHECK LAST BYTE
          BX7    -X1*X6
          ZR     X7,=XWTX=   EXIT IF 0000 BYTE
          SB6    WTHB        PREPARE ZERO WORD
          SX7    X7-2R
          NZ     X7,WTH4     IF NOT *  * BYTE

*         DELETE TRAILING SPACE BYTES.

          SA1    WTHA        =1H
          BX2    X6-X1
          SX7    B1
          SA1    WTHC        =4000400040004004000B
          IX7    X2-X7
          BX2    -X7+X2
          SB2    60-11
          BX7    X1*X2
          LX2    X7,B2
          IX1    X7-X2
          BX7    X7+X1
          SX2    B3
          BX7    X7+X4
          BX6    X7*X6
          SA6    A6
          EQ     =XWTX=      EXIT

 WTHA     DATA   1H
 WTHB     DATA   0
 WTHC     DATA   4000400040004004000B
 WTH=     SPACE  4,5
          IF     -DEF,QUAL$,2
          QUAL   *
 WTH=     =      /COMCWTH/WTH=
          BASE   *
 WTH=     ENDX
COMCRDH
COMMON
 RDH=     CTEXT  COMCRDH - READ CODED LINE, -H- FORMAT.
          IF     -DEF,QUAL$,1
          QUAL   COMCRDH
          BASE   DECIMAL
 RDH      SPACE  4
***       RDH - READ CODED LINE, -H- FORMAT.
*
*         ENTRY  (X2) = ADDRESS OF FET FOR FILE.
*                (B6) = FWA WORKING BUFFER.
*                (B7) = WORD COUNT OF WORKING BUFFER.
*         EXIT   (X2) = ADDRESS OF FET FOR FILE.
*                (X1) = 0, IF TRANSFER COMPLETE.
*                     = -2, IF EOI.
*                     = -1, IF EOF.
*                     = (B6) = LWA+1 OF DATA TRANSFERRED, IF EOR.
*
*         USES   ALL REGISTERS EXCEPT A0, X0, A5, X5.
*         CALLS  LCB=, RDX=.


 RDH=     SUBR   X           RETURN EXIT
          SA4    RDH6        SET RETURN ADDRESS
          IF     -DEF,B1=1,1
          SB1    1
          SA1    X2+4        (B5) = LIMIT
          SA3    X2+B1       (X3) = FIRST
          SB7    B6+B7       (B7) = LWA+1 WORKING BUFFER
          SB5    X1+

*         INITIALIZE REGISTERS FOR TRANSFER.

 RDH1     SA1    A3+B1       (B3) = IN
          SA2    A1+B1       (B4) = OUT
          MX7    60-6        (X7) = CHARACTER MASK
          SB3    X1
          SB4    X2+
          SX4    1R

*         TRANSFER DATA FROM CIRCULAR BUFFER TO WORKING BUFFER.

 RDH2     EQ     B4,B3,=XLCB= LOAD CIRCULAR BUFFER IF OUT = IN
          SA1    B4          READ WORD
          SB4    B4+B1       (OUT+1)
          BX2    -X7*X1      CHECK LAST CHARACTER
          NE     B4,B5,RDH3  IF (OUT+1) ' LIMIT
          SB4    X3          (OUT+1) = FIRST
 RDH3     ZR     X2,RDH4     IF LAST CHARACTER = 00
          EQ     B6,B7,RDH2  LOOP IF WORKING BUFFER FILLED
          BX6    X1          STORE WORD
          SB6    B6+B1       ADVANCE WORKING BUFFER
          SA6    B6-1
          EQ     RDH2        LOOP

 RDH4     LX7    6
          BX2    -X7*X1
          ZR     X2,RDH5     IF 0000 BYTE
```

```
          IX1    X1+X4       ADD * *
          MX7    60-6        RESET MASK
          EQ     RDH3        STORE WORD

*         SPACE FILL LAST WORD.

 RDH5     EQ     B6,B7,=XRDX= EXIT IF WORKING BUFFEF FILLED
          BX6    X1
          SX7    B1
          SA1    RDHA        =4040404040404040404040B
          IX7    X6-X7
          BX4    -X7+X6
          SB2    60-5
          BX7    X1*X4
          LX4    X7,B2
          SA1    RDHB        =1H
          IX2    X7-X4
          BX7    X7+X2
          BX4    -X7*X1
          IX6    X6+X4
          SA6    B6
          SB6    B6+B1
          EQ     =XRDX=      EXIT

*         SPACE FILL REMAINDER OF WORKING BUFFER.

 +        EQ     RDH1

 RDH6     EQ     B6,B7,RDH=  RETURN IF WORKING BUFFER FULL
          SA4    RDHB        =1H
          BX6    X4
 RDH7     SA6    B6
          SB6    B6+B1
          NE     B6,B7,RDH7
          EQ     RDH=        RETURN

 RDHA     DATA   4040404040404040404040B
 RDHB     DATA   1H
          SPACE  4
          BASE   *
          IF     -DEF,QUAL$,1
          QUAL   *
 RDH=     ENDX
COMCESP
COMMON
 ESP=     CTEXT  COMCESP - EXECUTE SYSTEM PROGRAM.
          IF     -DEF,QUAL$,1
          QUAL   COMCESP
 ESP      SPACE  4
***       COMCESP - EXECUTE SYSTEM PROGRAM.
*         D. L. MAUSNER.  05/15/76.  NUCC.
 ESP      SPACE  4
***              *ESP=* EXECUTES A CONTROL STATEMENT AT A SPECIFIED
*         FIELD LENGTH, AND THEN RETURNS TO THE CALLER.
*
*         ENTRY  (A1) = FWA OF A LIST AS FOLLOWS:
*                       ADDRESS OF C-FORMAT CONTROL STATEMENT
*                       ADDRESS OF A WORD CONTAINING DESIRED FL
*                (X1) = ((A1)).
*
*         EXIT   (X6) = 0, IF NO ERROR.
*                     = ERROR CODE, OTHERWISE (SEE *COMCXER*).
*
*         USES   ALL REGISTERS.
*
*         CALLS  JCM=.


 ESP=     SUBR   X           RETURN EXIT
          SB1    1
          SA2    A1+B1       (X2) = FL REQUEST
          SA2    X2

*         EXECUTE CONTROL STATEMENT.

          JCEXST X1,X2

*         EXTRACT ERROR FLAG.

          AX2    48D
          SX6    X2
          EQ     ESP=        RETURN
 ESP      SPACE  4
          IF     -DEF,QUAL$,1
          QUAL   *
 ESP=     ENDX
COMCCPM
COMMON
 CPM=     CTEXT  COMCCPM - CALL CONTROL POINT MANAGER.
          IF     -DEF,QUAL$,1
          QUAL   COMCCPM
          BASE   DECIMAL
 CPM=     SPACE  4,10
***       COMCCPM - CALL CONTROL POINT MANAGER.
*         C. G. FILSTEAD.  03/18/74.
*         REVISED - 02/16/75.  CGF.
 CPM=     SPACE  4,10
***       CPM - CALL CONTROL POINT MANAGER.
*
*         ENTRY  (X2) = BITS 0 - 47, ARGUMENTS.
*                (X7) = FUNCTION CODE.  IF (X7) < 0,  THEN (X7) IS THE
*                       COMPLEMENT OF THE FUNCTION CODE,  AND AUTO RE-
*                       CALL WILL BE REQUESTED.
*         EXIT   (A7) = ADDRESS OF *CPM.*.
*                (CPM.) = STATUS WORD.
*
*         USES   X - 1, 2, 6, 7.
*                B - NONE.
*                A - 1.
*
*         CALLS  SYS=, WNB=.


 CPM2     RJ     =XWNB=      PLACE PROGRAM ON RECALL
 CPM3     SA7    X2          SET STATUS WORD
          BX6    X2          CALL *CPM*
          RJ     =XSYS=

 CPM=     EQ     *+1S17      ENTRY/EXIT
          SA1    CPM.        CHECK STATUS WORD
          SX6    3RCPM       FORM *CPM* CALL
          PL     X7,CPM1     IF NO AUTO RECALL
          PX6    X6          ELSE, SET AUTO RECALL
          BX7    -X7         COMPLEMENT FUNCTION CODE
 CPM1     LX2    12          FORM STATUS WORD
          LX1    59          CHECK PREVIOUS STATUS
          BX7    X7+X2
          SX2    A1
          LX6    59-17
          BX2    X2+X6
          NG     X1,CPM3     IF STATUS WORD NOT BUSY
          EQ     CPM2        ELSE, RECALL

 CPM.     CON    1           STATUS WORD
          SPACE  4
          BASE   *
          IF     -DEF,QUAL$,3
          QUAL   *
 CPM=     =      /COMCCPM/CPM=
 CPM.     =      /COMCCPM/CPM.
 CPM=     ENDX
COMCJCM
COMMON
 JCM=     CTEXT  COMCJCM - JOB CONTROL MANAGER INTERFACE.
          IF     -DEF,QUAL$,1
          QUAL   COMCJCM
 JCM      SPACE  4
***       COMCJCM - JOB CONTROL MANAGER INTERFACE.
*         D. L. MAUSNER.  05/15/76.  NUCC.
 JCM      SPACE  4
***              *JCM=* PROVIDES A CPU INTERFACE WITH THE JOB CONTROL
*         MANAGER PACKAGE.
*
*         ENTRY  (X1) = STATUS WORD 1 BITS 42-59.
*                (X2) = STATUS WORD 2 BITS 00-59.
*                (X7) = FUNCTION CODE.
*                (B1) = 1.
*
*         EXIT   (X1) = STATUS WORD 1.
*                (X2) = STATUS WORD 2.
*                (X6) = 0, IF NO ERROR.
*                     = ADDRESS OF ERROR MESSAGE, OTHERWISE.
*                (B1) = 1.
*
*         USES   A - 1, 2, 6, 7.
*                X - 7.
*
*         NOTE   FUNCTIONS WHICH SUSPEND PROGRAM EXECUTION
*                USE *ALL* REGISTERS.
*
*         NOTE   THE SYSTEM REQUEST IS POSTED HERE, INSTEAD
*                OF IN *SYS=*, TO AVOID PROGRAM/MONITOR TIMING PROBLEMS.


 JCM1     SA1    B1          WAIT (RA+1)
          NZ     X1,*
          SA6    A1
          XJ     B0          ENTER MONITOR

*         SET EXIT CONDITIONS.

          SB1    1
          SA1    JCMB        RETURN STATUS WORDS
          SX6    X1          CHECK ERROR CODE
          AX6    9D
          SA2    A1+1
          ZR     X6,JCM=     IF NO ERROR
          SX6    A2+1        (X6) = ADDRESS OF MESSAGE

 JCM=     SUBR   X           RETURN EXIT
          MX6    42D         ISOLATE STATUS WORD 1 [42:59]
          BX6    X6*X1
          BX6    X6+X7       MERGE FUNCTION CODE
          BX7    X2
          SA1    JCMA
          SA6    A1+B1       SET STATUS WORDS
          SA7    A6+B1
          BX6    X1
          JP     JCM1        ENTER REQUEST

*         *ACE* STATUS WORDS.

 JCMA     CON    4LACEP+JCMB
 JCMB     BSS    5
 JCM      SPACE  4
          IF     -DEF,QUAL$,1
          QUAL   *
 JCM=     ENDX
COMCPPC
COMMON
 PPC=     CTEXT  COMCPPC - POINTER-PUSH COPY.
          IF     -DEF,QUAL$,1
          QUAL   COMCPPC
          BASE   DECIMAL
 COMCPPC  SPACE  4,10
***       COMCPPC - POINTER-PUSH COPY.
*         J. M. SCHNEIDER.  10/10/75.
*         WITH THE AID OF S. H. KEYSER AND L. R. ATKIN.
 COMCPPC  SPACE  4,10
***          PPC PERFORMS A POINTER PUSH COPY.  THE COPY STARTS
*         FROM THE CURRENT POSITION ON THE FILE, WITH DATA POSSIBLY
*         IN THE BUFFER.  THE COPY TERMINATES WHEN THE -EOR- BIT
*         IS FOUND TO BE SET IN THE R FET.
*
*         NOTATION                 IN THE DOCUMENTATION, THE FETS ARE
*                                  REFERED TO AS R, THE FET FOR THE FILE BEING
*                                  READ, AND W, THE FET FOR THE FILE BEING
*                                  WRITTEN.  THE NOTATION R(IN), FOR EXAMPLE,
*                                  MEANS THE VALUE OF THE R FETS -IN-
*                                  POINTER.
*
*         ENTRY  (B3) = FWA R FET.
*                (B4) = FWA W FET.
*                (X3) = CIO CODE FOR READING.
*                (X4) = CIO CODE FOR WRITING.
*                (B1) = 1.
*                R(FIRST) = W(FIRST).
*                R(LIMIT) = W(LIMIT).
*                R(OUT) = W(OUT).  (SEE NOTE)
*
*         EXIT   R(EOR) = 1.
*                R(COMPLETE) = 1.
*                W(IN) = R(IN).
*                (B2) = 2.
*
*         USES   A1, X1, A2, X2, A6, X6, X7.
*
*         NEEDS  RECALL.
*
*         CALLS  CIO=,RCL=.
*
*         NOTE   THE LENGTH OF THE BUFFER MUST BE AT LEAST
*                TWO PRU-S LONG.
*
```

```
*                UPON ENTRY, THE REQUIRMENT THAT R(OUT) = W(OUT)
*                MAY BE RELAXED TO THE EXTENT THAT IF THE R FET IS
*                INACTIVE AND THE BUFFER FROM W(OUT) TO R(IN) CONTAINS
*                VALID DATA TO BE WRITTEN, THE COPY WILL BE PERFORMED
*                CORRECTLY.
*
*                AT THE CONCLUSION OF THE COPY, THE W FET MAY INDICATE
*                DATA REMAINING IN THE BUFFER.  IN ANY CASE, IT IS THE
*                CALLERS RESPONSIBILITY TO PERFORM ANY BUFFER FLUSHING
*                OR CLOSING PROCEDURES NECESSARY FOR THE FILE BEING
*                WRITTEN.
*



 PPC1     BX7    X3          RE-ISSUE THE READ
          SX2    B3
          RJ     =XCIO=

 PPC2     RECALL             WAIT A BIT
          SA1    B4          CHECK W(COMPLETE)
          LX1    59-0
          PL     X1,PPC3     WRITE STILL RUNNING
          BX7    X4          RE-ISSUE THE WRITE
          SX2    B4
          RJ     =XCIO=

 PPC3     SA1    B3          CHECK R(COMPLETE)
          SA2    B3+B2       R(IN)
          LX1    59-0
          BX6    X2
          LX7    X1
          SA6    B4+B2       W(IN) <= R(IN)
          SA1    A6+B1       W(OUT)
          BX6    X1
          SA6    A2+B1       R(OUT) <= W(OUT)
          PL     X7,PPC2     READ STILL RUNNING
          LX7    -4+0        CHECK R(EOR)
          PL     X7,PPC1     NOT EOR

 PPC=     EQ     *+1S17      ENTRY/EXIT

          SB2    2           (B2) = 2
          EQ     PPC3        PUSH POINTERS FIRST
 COMCPPC  SPACE  4,10
          IF     -DEF,QUAL$,2
          QUAL   *
 PPC=     EQU    /COMCPPC/PPC=
          BASE   *
 PPC=     ENDX
COMCAIM
COMMON
 AIM=     CTEXT  COMCAIM - ATTACH IF MISSING.
          IF     -DEF,QUAL$,1
          QUAL   COMCAIM
 AIM=     SPACE  4
***       COMCAIM - ATTACH IF MISSING.
*         S. H. KEYSER.  12/19/73.
 AIM=     SPACE  4
***               AIM CHECKS WHETHER THE SPECIFIED FILE IS LOCAL TO THIS
*         CONTROL POINT.  IF NOT, THE SCOPE PERMANENT FILE MANAGER IS
*         CALLED TO ATTACH THE FILE.
*
*         ENTRY  (X1) = 42/0LFN,18/**.
*                (B1) = 1.
*
*         EXIT   (X6) > 0, FILE IS LOCAL.
*                (X6) = 0, FILE WAS ATTACHED.
*                (X6) =-0, FILE NOT FOUND.
*
*         USES   X - 1, 2, 7.
*                B - NONE.
*                A - 1, 6.
*
*         CALLS  SYS=.
 AIM=     SPACE  4,10
***       ASSEMBLY CONSTANT.
*
*
*  AIM[FDB]      IF DEFINED, THEN THE VALUE OF THIS SYMBOL IS THE
*                ADDRESS OF THE FILE DEFINITION BLOCK (FDB) TO BE
*                USED FOR THE ATTACH, IF NECESSARY.  IF NOT DEFINED,
*                A MINIMUM LENGTH INTERNAL FDB IS USED.
 AIM=     SPACE  4,10
          IF     DEF,KRONOS,3
 AIMA     CON    0                FET FIRST WORD
          CON    1S44             EP BIT
          SKIP   5
          IF     DEF,AIM[FDB],2
 AIMA     EQU    AIM[FDB]
          SKIP   2
          BSSZ   4           SPACE FOR FDB
 AIMA

 AIM=     PS     0           ENTRY/EXIT
          MX7    42D
          BX6    X7*X1       CLEAR BITS 0 - 17
          SA6    AIMA        SET PRM STATUS WORD
          SX1    3RPRM       FORM PRM SYSTEM CALL
          SX2    A6
          PX1    X1
          LX1    42D
          BX6    X1+X2
          RJ     =XSYS=
          SA1    X6
          BX2    X6
          SX6    X1-1
          NZ     X6,AIM=     IF ANY PERM BITS RETURNED
          BX6    X7*X1
          SA6    X2          ELSE SET UP AN FDB
          IF     DEF,KRONOS,2
          SX1    3RPFM&3RPRM
          SKIP   1
          SX1    3RPFA&3RPRM
          LX1    42D
          BX6    X1-X2       AND SET CALL TO PFA
          IF     DEF,KRONOS,3
          SX2    11B
          LX2    24D
          BX6    X2+X6
          RJ     =XSYS=
          SA1    X6          CHECK PFA RETURN STATUS
          SX2    X1-1
          IF     DEF,KRONOS,1
          AX2    10D
          BX6    X6-X6
          IX1    X6-X2
          BX6    X1-X2       -0 IF ANYTHING FUNNY
          EQ     AIM=        RETURN
 AIM=     SPACE  4
          IF     -DEF,QUAL$,2
          QUAL   *
 AIM=     =      /COMCAIM/AIM=
 AIM=     ENDX
COMCSET
COMMON
 SET      CTEXT  COMCSET - PRESET AREA OF STORAGE.
          IF     -DEF,QUAL$,1
          QUAL   COMCSET
 SET      SPACE  4
***       COMCSET - PRESET AREA OF STORAGE.
*         S. H. KEYSER.  02/02/74.
*         ADAPTED FROM ROUTINE PRESET IN COMPASS 2.0
 SET      SPACE  4
***               SET PRESETS AN AREA OF CENTRAL MEMORY TO A SPECIFIED
*         VALUE.  DISASTER IF FWA IS GREATER THAN LWA.
*
*         ENTRY  (X1) = DATA.
*                (X2) = FWA.
*                (X3) = LWA+1.
*                (B1) = 1.
*
*         USES   X - 2, 3, 6, 7.
*                B - NONE.
*                A - 6, 7.
*
*         CALLS  NONE.
*

 SET      PS     0           ENTRY/EXIT
          BX6    X1
          IX3    X3-X2       WORD COUNT
          SA6    X2          SET FIRST WORD
          SX2    B1
          LX3    -1
          BX7    X6
          PL     X3,SET1     IF EVEN
          SX3    X3
          ZR     X3,SET      IF 1 WORD ONLY
          SA6    A6+B1

 SET1     SA6    A6+B1
          IX3    X3-X2
          ZR     X3,SET      IF 2 OR 3 WORDS ONLY

 SET2     SA7    A6+B1
          IX3    X3-X2
          SA6    A7+1
          NZ     X3,SET2     LOOP TILL WORD COUNT SATISFIED
          EQ     SET         RETURN
 SET      SPACE  4
          IF     -DEF,QUAL$,2
          QUAL   *
 SET      =      /COMCSET/SET
 SET      ENDX
COMCWFW
COMMON
 WFW=     CTEXT  COMCWFW - WRITE FILL WORDS.
          IF     -DEF,QUAL$,1
          QUAL   COMCWFW
          BASE   D
          SPACE  4
***       COMCWFW - WRITE FILL WORDS.
*         S. H. KEYSER.  12/03/75.
          SPACE  4
***               WFW WRITES A SPECIFIED NUMBER OF FILL WORDS ON A FILE
*         THE CONTENTS OF WHICH ARE UNDEFINED.  THIS CAN ACTUALLY BE A
*         USEFULL FUNCTION FOR SITUATIONS SUCH AS GENERATING PADDING
*         SPACE ON A BLOCKED RECORD.
*
*         ENTRY  (X2) = ADDRESS OF FET FOR FILE.
*                (B6) = NUMBER OF WORDS TO FILL.
*                (B1) = 1.
*
*         EXIT   (X2) = ADDRESS OF FET FOR FILE.
*
*         USES   ALL REGISTERS EXCEPT B7, A0, X0, A5, AND X5.
*
*         CALLS  DCB=, WTX=.


 +        JP     WFW1        CONTINUE WRITE

 WFW=     PS     0           ENTRY/EXIT
          SA1    X2+4        (B5) = LIMIT
          SB5    X1
          SA3    X2+B1       (A3) = FIRST
          SA4    WFW=        SET RETURN ADDRESS
          SB2    X3
          SX4    B2-B5       (X4) = -BUFFER SIZE

 WFW1     SA2    A3+B1       READ IN
          SB3    X2+B1       (B3) = IN+B1
          SA1    A2+B1       READ OUT
          SB4    X1          (B4) = OUT
          SX3    B4-B3       OUT-(IN+1)
          SB2    B4-B3
          AX3    59
          BX6    -X4*X3      0 OR LIMIT-FIRST
          SB2    X6+B2       AMOUNT OF SPACE LEFT IN BUFFER
          SX6    B6-B2       WC-AVAIL
          BX7    X6
          AX6    59          -0 IF AVAIL > WC, ELSE +0
          BX7    X6*X7
          SB2    X7+B2       MIN (WORD COUNT, SPACE LEFT)
          SB3    B3+B2       ADVANCE IN
          SX3    B5-B3       TEST AGAINST LIMIT
          AX3    59          -0 IF LIMIT < IN
          BX6    X3*X4       -BUFFER SIZE IF WRAP
          SB3    X6+B3       ADJUST IN
          SX2    B3-B1
          GE     B2,B6,=XWTX=  WRITE EXIT IF ALL WORDS WRITTEN
          SB6    B6-B2       ELSE DECREMENT WORDS REMAINING
          JP     =XDCB=      AND DUMP CIRCULAR BUFFER
 WFW=     SPACE  4,5
          BASE   *
          IF     -DEF,QUAL$,2
          QUAL   *
 WFW=     =      /COMCWFW/WFW=
 WFW=     ENDX
COMCRWA
COMMON
 RWA=     CTEXT  COMCRWA - READ WORD ADDRESSABLE FILE.
          IF     -DEF,QUAL$,1
          QUAL   COMCRWA
```

```
          BASE   D
 RWA=     SPACE  4
***       COMCRWA - READ WORD ADDRESSABLE FILE.
*         L. R. ATKIN.  07/20/75.
 RWA=     SPACE  4
***             RWA READS A FILE STARTING AT A SPECIFIED DISK WORD
*         ADDRESS.  WORDS ON THE FILE ARE NUMBERED STARTING AT 100B.
*         IF THE WORKING BUFFER IS LONGER THAN THE CIRCULAR BUFFER,
*         RESULTS ARE UNPREDICTABLE.
*
*         ENTRY  (X1) = ADDRESS OF DATA TRANSFER ROUTINE.
*                (X2) = ADDRESS OF FET.
*                (X3) = DISK WORD ADDRESS.
*                (B6) = FWA CM WORKING BUFFER.
*                (B7) = LENGTH OF CM WORKING BUFFER.
*                ((X2)+7) = DISK WORD ADDRESS OF WORD AT ((X2)+3).
*                (B1) = 1.
*         EXIT   (X1) = 0 FOR TRANSFER COMPLETE.
*                (X1) = -1 IF EOF DETECTED ON FILE.
*                (X1) = ADDRESS OF LAST WORD TRANSFERRED INTO WORKING
*                            BUFFER IF EOR DETECTED ON FILE BEFORE
*                            TRANSFER WAS COMPLETED.
*                (B6) = ADDRESS OF LAST WORD TRANSFERRED TO WOPKING
*                            BUFFER.
*                (X2) = ADDRESS OF FET FOR FILE.
*                ((X2)+7) = DISK WORD ADDRESS OF WORD AT ((X2)+3).
*
*         USES   ALL REGISTERS EXCEPT A0, X0, A5, AND X5.
*         CALLS  (X1), CIO=, RCL=, WNB=.
*



 RWA1     NG     X2,RWA3     IF BUFFER QUIET
          RECALL             WAIT
          SA2    A2
          SA1    A4-B1       (X1) = IN
          EQ     RWA4

 RWA2     RECALL X2          WAIT UNTIL BUFFER QUIET

 RWA3     SX6    X4
          SA6    A4-B1       (FET+2) = OUT
          BX6    X3
          AX6    6           PRU NUMBER
          SA6    A4-3+6      (FET+6) = PRU NUMBER
          READ   A4-3
          SA2    X2
          MX6    -6
          BX7    -X6*X3      OFFSET INTO PRU
          SA1    A4-B1       (X1) = IN
 RWA4     IX6    X1-X4       IN - OUT
          PL     X6,RWA5     IF NO WRAPAROUND
          SX6    X6+B3       ADD BUFFER LENGTH
 RWA5     IX6    X7-X6       NUMBER OF WORDS TO SKIP - DATA LENGTH
          LX2    59-0        CHECK COMPLETE BIT
          PL     X6,RWA1     IF FIRST WORD NOT IN BUFFER
          SB4    B4+X7       ADVANCE OUT
          LT     B4,B5,RWA6  IF OUT NOT PAST LIMIT
          SB4    B4-B3       SUBTRACT BUFFER LENGTH
 RWA6     SX6    B4
          BX7    X3
          SA6    A4+
          SA7    A4-3+7      (FET+7) = DISK WORD ADDRESS
          SA6    RWAB        SAVE OLD OUT
 RWAA     SX2    A2+0
          RJ     *+400000B   CALL DATA TRANSFER ROUTINE
          SA3    RWAB        OLD OUT
          SA4    X2+3        NEW OUT
          IX6    X4-X3       TRANSFER LENGTH
          PL     X6,RWA7     IF NO WRAP AROUND
          SA4    A4+B1       LIMIT
          SX7    X4+
          SA4    X2+1        FIRST
          IX6    X6+X7
          SX4    X4+
          IX6    X6-X4
 RWA7     SA3    X2+7        ADVANCE DISK WORD ADDRESS
          IX6    X6+X3
          SA6    A3

 RWA=     EQ     *+400000B   ENTRY/EXIT
          SA4    RWAA        FORM RJ TO DATA TRANSFER ROUTINE
          MX7    -18
          BX4    X7*X4
          BX6    X1+X4
          SA6    A4
          SA4    X2+7        (X7) = NUMBER OF WORDS TO SKIP
          SA1    X2+B1       (B2) = FIRST
          IX7    X3-X4
          SA4    X2+4        (B5) = LIMIT
          SB2    X1
          SB5    X4
          SB3    B5-B2       (B3) = BUFFER LENGTH
          SA4    A4-B1       (X4) = OUT
          SB4    X4          (B4) = OUT
          NG     X7,RWA2     IF BACKWARD MOTION REQUIRED
          SA2    X2          FOR COMPLETE BIT TEST
          SA1    A4-B1       (X1) = IN
          EQ     RWA4

 RWAB     CON    0           OUT BEFORE DATA TRANSFER
 RWA=     SPACE  4,6
          BASE   *
          IF     -DEF,QUAL$,2
          QUAL   *
 RWA=     EQU    /COMCRWA/RWA=
 RWA=     ENDX
COMCSFA
COMMON
 SFA      CTEXT  COMCSFA - SPACE FILL ANYTHING.
          IF     -DEF,QUAL$,1
          QUAL   COMCSFA
 SFA      SPACE  4
***       SFA - SPACE FILL ANYTHING.
*         S. H. KEYSER.  08/26/73.
*         METHOD BY L. R. ATKIN.
*         REVISED - 03/27/74.  C. G. FILSTEAD.
 SFA      SPACE  4
***             SFA MAPS IMBEDDED ZERO CHARACTERS INTO BLANKS.
*
*         ENTRY  (X1) = CHARACTER STRING.
*
*         EXIT   (X6) = STRING SPACE FILLED.
*
*         USES   A2, X2.
*
*         CALLS  NONE.
*


 SFA      PS     0           ENTRY/EXIT

          SA2    =4040404040404040404040B
          BX6    -X2*X1
          IX6    X6-X2       # 40 IF CHARACTER = 01-37, 41-77

          BX6    X6+X1       # 40 IF CHARACTER = 01-77
          BX2    -X6*X2      = 40 IF CHARACTER = 00
          BX6    X2
          LX2    -3

          BX2    X2+X6       40 --> 44
          BX6    X1+X2       00 --> 44
          LX2    -2
          BX6    X6+X2       00 --> 55
          JP     SFA         RETURN
 SFA      SPACE  4,4
          IF     -DEF,QUAL$,2
          QUAL   *
 SFA      =      /COMCSFA/SFA
 SFA      ENDX
COMCVFN
COMMON
 VFN      CTEXT  COMCVFN - VERIFY FILE NAME.
          IF     -DEF,QUAL$,1
          QUAL   COMCVFN
 VFN      SPACE  4
***       COMCVFN - VERIFY FILE NAME.
*         L. R. ATKIN.  12/07/73.
*         DECIPHERED BY S. H. KEYSER.  01/03/74.
 VFN      SPACE  4
***             VFN CHECKS THE SPECIFIED FILE NAME FOR LEGALITY.
*
*         RULES  (1)  THE NAME MUST BE 7 CHARACTERS OR LESS.
*                (2)  THE FIRST CHARACTER IS ALPHABETIC.
*                (3)  ALL OTHER CHARACTERS ARE ALPHANUMERIC.
*                (4)  IMBEDDED ZERO CHARACTERS (00) ARE ILLEGAL.
*
*         ENTRY  (X1) = 60/0LFN.
*                (B1) = 1.
*
*         EXIT   (X6) = MASK OF ILLEGAL CHARACTER POSITIONS.
*
*         USES   X - 2, 3, 4, 7.
*                B - NONE.
*                A - 2, 3.
*
*         CALLS  NONE.
*

 VFN      PS     0           ENTRY/EXIT
          SA2    =4040404040404040404040B
          MX3    42D
          BX6    X3*X1       EXTRACT FIRST 7 CHARS
          SX4    B1
          IX3    X1-X4       SET LOW ORDER ZEROS TO 40S
          BX3    -X1*X3
          BX7    X3+X6       0X FOR 8TH - 10TH CHARACTERS PRESENT
          SA3    =10HE000000000
          BX4    -X2*X6      NAME UNDER 37S
          IX6    X3+X4       4X IF INVALID UNDER 37
          BX3    X2*X7       NAME UNDER 40S
          IX4    X4-X2       4X IF NON-ZERO UNDER 37
          BX7    X3+X4       4X IF NON-ZERO UNDER 77
          BX4    X3*X6       4X IF INVALID UNDER 77
          BX6    X6+X3       4X IF FIRST > 0
          MX3    1
          BX6    X3*X6       EXTRACT FIRST CHARACTER VALIDITY
          BX6    X4+X6
          BX7    -X7+X6      4X IF INVALID
          BX3    X2*X7
          BX4    X3
          LX3    -5
          IX7    X4-X3
          BX6    X4+X7
          EQ     VFN         RETURN
 VFN      SPACE  4
          IF     -DEF,QUAL$,2
          QUAL   *
 VFN      =      /COMCVFN/VFN
 VFN      ENDX
COMCRDW
COMMON
 RDW=     CTEXT  COMCRDW - READ WORDS.
          IF     -DEF,QUAL$,1
          QUAL   COMCRDW
          BASE   DECIMAL
 RDW      SPACE  4
***       RDW - READ WORDS TO WORKING BUFFER.
*
*         ENTRY  (X2) = ADDRESS OF FET FOR FILE.
*                (B6) = FWA WORKING BUFFER.
*                (B7) = WORD COUNT OF WORKING BUFFER.
*         EXIT   (X2) = ADDRESS OF FET FOR FILE.
*                (X1) = 0, IF TRANSFER COMPLETE.
*                     = -2, IF EOI.
*                     = -1, IF EOF.
*                     = (B6) = LWA+1 OF DATA TRANSFERRED, IF EOR.
*
*         USES   ALL REGISTERS EXCEPT A0, X0, A5, X5.
*         CALLS  LCB=, RDX=.


 +        EQ     RDW1

 RDW=     SUBR   X           RETURN EXIT
          SA4    *-1         SET RETURN ADDRESS
          IF     -DEF,B1=1,1
          SB1    1
          SA1    X2+4        (B5) = LIMIT
          SA3    X2+B1       (X3) = FIRST
          SB7    B6+B7       (B7) = LWA+1 WORKING BUFFER
          SB5    X1+

*         INITIALIZE REGISTERS FOR TRANSFER.

 RDW1     SA1    A3+B1       (B3) = IN
          SA2    A1+B1       (B4) = OUT
          SB3    X1
          SB4    X2

*         TRANSFER DATA FROM CIRCULAR BUFFER TO WORKING BUFFER.

 RDW2     EQ     B4,B3,LCB=  LOAD CIRCULAR BUFFER IF OUT = IN
```

```
          SA1    B4          READ WORD
          BX6    X1
          SB4    B4+B1       (OUT+1)
          NE     B4,B5,RDW3  IF (OUT+1) ' LIMIT
          SB4    X3          (OUT+1) = FIRST
 RDW3     SA6    B6          STORE WORD
          SB6    B6+B1       ADVANCE WORKING BUFFER
          NE     B6,B7,RDW2  LOOP TO FILL WORKING BUFFER
*         EQ     RDX=        EXIT
 RDX      SPACE  4
***       RDX - READ EXIT.
*         EXIT FROM READ SUBROUTINE TO CALLER.
*         IF CIRCULAR BUFFER IS BUSY, OR EOR/EOF IS SENSED, NO ACTION
*         IS TAKEN.
*         OTHERWISE, THE WORD COUNT REMAINING IN THE BUFFER IS CHECKED
*         AND A READ FUNCTION ISSUED IF NECESSARY.
*
*         ENTRY  (A2) = ADDRESS OF OUT.
*                (A3) = ADDRESS OF FIRST.
*                (A4) = RETURN ADDRESS.
*                (X3) = FIRST.
*                (B3) = IN.
*                (B4) = OUT.
*                (B5) = LIMIT.
*
*         EXIT   TO RETURN ADDRESS.
*
*         USES   X - 1, 2, 3, 4, 6, 7.
*                B - 2.
*                A - 1, 6.
*
*         CALLS  CIO=, DEVTYP.


 RDX=     SUBR   D,X         DIRECT ENTTRY
          SA1    A3-B1       CHECK READ STATUS
          SX6    B4          STORE OUT
          LX1    59-0
          SA6    A2
          SX2    A3-B1       RESET (X2)
          PL     X1,RDX1     IF BUFFER BUSY
          LX1    60-4
          NG     X1,RDX1     IF EOR/EOF SET
          SA1    X2+B1       CHECK DEVICE TYPE
          RJ     =XDEVTYP    CHECK DEVICE TYPE
          NG     X6,RDX1     IF NON-ALLOCATABLE DONT READ AHEAD

**        IF BUFFER IS NOT BUSY, CHECK BUFFER SIZE.
*         ISSUE READ IF BUFFER THRESHOLD IS REACHED.

          SX6    B3-B4       (IN-OUT)
          SB2    X3          (LIMIT-FIRST)
          LX3    X6,B1       2*(IN-OUT)
          SX7    B5-B2
          AX6    60          SIGN OF (IN-OUT)
          BX4    X6-X7       INVERT BUFFER IF OUT # IN
          IX6    X4-X3       BUFFER SIZE - 2*(IN-OUT)
          NG     X6,RDX1     IF BUFFER THRESHOLD NOT REACHED
          READ   X2
 RDX1     SX1    B0          RESPONSE = 0
          SB2    A4          SET RETURN ADDRESS
          JP     B2          RETURN
 LCB      SPACE  4
***       LCB - LOAD CIRCULAR BUFFER.
*         REQUEST READ IF BUFFER IS EMPTY, NOT BUSY, AND NO EOR/EOF.
*         IF BUFFER IS BUSY, RECALL AND RETURN.
*
*         ENTRY  (A2) = ADDRESS OF OUT.
*                (A3) = ADDRESS OF FIRST.
*                (A4) = RETURN ADDRESS.
*                (B4) = OUT.
*         EXIT   TO RETURN ADDRESS - 1 IF CONTINUATION READ.
*                TO RETURN ADDRESS IF EOR/EOF.
*                (X1) = LAST WORD ADDRESS OF WORKING BUFFER.
*                (X1) = -1 IF EOF.
*                (X1) = -2 IF EOI.
*
*         USES   X - 1, 6, 7.
*                B - 2, 3.
*                A - 1, 6, 7.
*
*         CALLS  CIO=, RCL=.


 LCB=     SUBR   D,X         DIRECT ENTRY
          SA1    A3-B1       CHECK READ STATUS
          SX6    B4          STORE OUT
          LX1    59-0
          SA6    A2
          NG     X1,LCB2     IF BUFFER NOT BUSY
          RECALL
 LCB1     SB2    A4-B1       CONTINUE READ
          JP     B2

 LCB2     SA1    A2-B1       RE-READ IN
          SB3    X1
          NE     B3,B4,LCB1  IF BUFFER NOT EMPTY
          SA1    A3-B1       CHECK BUFFER STATUS
          LX1    59-4
          NG     X1,LCB3     IF EOR SET
          READ   A3-B1
          SB2    A4-B1       CONTINUE READ
          JP     B2

 LCB3     SA1    A3          SET (IN) = (OUT) = (FIRST)
          SX7    X1
          SA7    A3+B1
          SA7    A7+B1
          SX2    A3-B1       (X2) = FILE ADDRESS
          SX1    -2          SET RETURN RESPONSE
          SA3    X2
          LX3    59-9
          MI     X3,LCB4     IF EOI
          LX3    9-3
          SX1    -B1
          MI     X3,LCB4     IF EOF
          SX1    B6          ELSE SET EOR
 LCB4     SB2    A4          RETURN
          JP     B2
          SPACE  4
          BASE   *
          IF     -DEF,QUAL$,1
          QUAL   *
 RDW=     ENDX
COMCNOM
COMMON
 COMCNOM  CTEXT  COMCNOM - NON-OVERLAPPED MOVE.
          IF     -DEF,QUAL$,1
          QUAL   COMCNOM
          BASE   DECIMAL
 COMCNOM  SPACE  4
***       NOM - NON-OVERLAPPED MOVE.
*         L. R. ATKIN.              08/09/73.
*
*         ENTRY  (X1) = FWA OF SOURCE.
*                (X2) = FWA OF DESTINATION.
*                (X3) = LENGTH OF DATA.
*                (B1) = 1.
*
*         USES   A1, A2, A6, A7, X1, X2, X3, X6, X7, AND B2.
*
*                IF SOURCE AND DESTINATION AREAS OVERLAP, RESULTS ARE
*                UNPREDICABLE.
*


 NOM1     SA2    A1+B1             LOAD 1
          SA1    A2+B1             LOAD 2
          BX7    X2                MOVE 1
          LX6    X1                MOVE 2
          SX3    X3-1              REDUCE BLOCK COUNT
          SA7    A6+B1             STORE 1
          SA6    A7+B1             STORE 2
          SA2    A1+B1             LOAD 3
          SA1    A2+B1             LOAD 4
          BX7    X2                MOVE 3
          LX6    X1                MOVE 4
          SA7    A6+B1             STORE 3
          SA6    A7+B1             STORE 4
          SA2    A1+B1             LOAD 5
          SA1    A2+B1             LOAD 6
          BX7    X2                MOVE 5
          LX6    X1                MOVE 6
          SA7    A6+B1             STORE 5
          SA6    A7+B1             STORE 6
          SA2    A1+B1             LOAD 7
          SA1    A2+B1             LOAD 8
          BX7    X2                MOVE 7
          LX6    X1                MOVE 8
          SA7    A6+B1             STORE 7
          SA6    A7+B1             STORE 8
          NZ     X3,NOM1           IF MORE TO MOVE

 NOM=     EQ     *+400000B
          ZR     X3,NOM=           IF NOTHING TO MOVE
          SA1    X1                MOVE FIRST WORD
          BX6    X1
          SA6    X2
          SX3    X3-1              DECREMENT COUNT
          MX6    -3
          BX7    -X6*X3            REMAINDER AFTER DIVISION BY 8
          BX6    -X6-X7            COMPLEMENT
          SB2    X6
          AX3    3                 DIVIDE BY 8
          JP     NOM2+B2           MOVE REMAINDER

 NOM2     SA1    A1+B1
          BX6    X1
          SA6    A6+1
          DUP    6,3
          SA1    A1+B1
          BX6    X1
          SA6    A6+1

          ZR     X3,NOM=           IF DONE, RETURN
          EQ     NOM1              MOVE BLOCKS OF 8
 COMCNOM  SPACE  4,10
          BASE   *
          IF     -DEF,QUAL$,2
          QUAL   *
 NOM=     EQU    /COMCNOM/NOM=

          ENDX
COMCWSC
COMMON
 WSC=     CTEXT  COMCWSC - WRITE SHIFTED CODED LINE, -C- FORMAT.
          IF     -DEF,QUAL$,1
          QUAL   COMCWSC
          BASE   DECIMAL
 WSC      SPACE  4
***       WSC - WRITE SHIFTED CODED LINE, -C- FORMAT.
*         D. J. SLATE.  05/03/77.  ADAPTED FROM COMCWTC.
          SPACE  4
***               WSC TRANSFERS ONE CODED LINE FROM THE WORKING BUFFER.
*         THE LINE IS SHIFTED RIGHT ONE CHARACTER AND A BLANK IS
*         INSERTED IN COLUMN 1.  IF NECESSARY, AN EXTRA ZERO WORD
*         IS APPENDED TO THE END.
*
*         ENTRY  (X2) = ADDRESS OF FET FOR FILE.
*                (B6) = FWA WORKING BUFFER.
*         EXIT   (X2) = ADDRESS OF FET FOR FILE.
*                (B6) = LWA+1 TRANSFERRED FROM WORKING BUFFER.
*
*         USES   ALL REGISTERS EXCEPT A0, X0, A5, X5, B7.
*         CALLS  DCB=, WTX=.
 WSC=     SPACE  4
**        ASSEMBLY CONSTANT.
*
*         WSC[C64] SHOULD BE DEFINED IF COMCWSC IS TO BLANK FILL LINES
*                TO THE FINAL ZERO BYTE.  WSC[C64] WILL BE DEFINED IF
*                IP.CSET IS EQUAL TO 64.

          IF     -DEF,WSC[C64],3
          IF     DEF,IP.CSET,2
          IFEQ   IP.CSET,64,1
 WSC[C64] SET    1


 +        EQ     WSC1

 WSC=     EQ     *+1S17      ENTRY/EXIT
          SA4    *-1
          IF     -DEF,B1=1,1
          SB1    1
          SA1    X2+4        (B5) = LIMIT
          SA3    X2+B1       (X3) = FIRST
          MX4    60-12       (X4) = BYTE MASK
          SB5    X1+
          SX6    1R          RESET CARRY-OVER CHARACTER TO BLANK
          SA6    WSCA
          SX6    0           CLEAR B6 SAVE FLAG
          SA6    WSCB

*         INITIALIZE REGISTERS FOR TRANSFER.
```

```
 WSC1     SA1    A3+2        (B4) = OUT
          SA2    A3+B1       (X2) = IN
          SB4    X1

*         TRANSFER DATA FROM WORKING BUFFER TO CIRCULAR BUFFER.

 WSC2     SB3    X2+B1       (IN+1)
          NE     B3,B5,WSC3  IF (IN+1) ' LIMIT
          SB3    X3          (IN+1) = FIRST
 WSC3     SA1    B6          NEXT WORD
          EQ     B3,B4,=XDCB=  DUMP CIRCULAR BUFFER IF (IN+1) = OUT
          BX7    X1
          SB6    B6+B1       ADVANCE WORKING BUFFER
          SA1    WSCA        CARRY-OVER CHARACTER
          MX6    60-6        MASK FOR BOTTOM CHARACTER
          BX6    -X6*X7
          SA6    A1          SAVE NEW CARRY-OVER
          MX6    60-6
          BX6    X6*X7       INSERT OLD CARRY-OVER INTO CURRENT WORD
          BX6    X6+X1
          LX6    60-6        SHIFT RIGHT ONE CHARACTER
          BX7    -X4*X7
          SA6    X2          STORE WORD
          SX2    B3          IN = IN+1
          NZ     X7,WSC2     LOOP TO END OF LINE
          BX1    -X4*X6
          NZ     X1,WSC4     IF NO ZERO BYTE
          IF     DEF,WSC[C64],11D
          IX7    X6+X4
          BX7    -X6*X7
          SA1    =4040404040404040404040B  BLANK FILL TO ZERO BYTE
          BX7    X1*X7       40B WHERE BLANKS NEEDED
          BX1    X7
          LX7    -3
          BX1    X1+X7       44B WHERE BLANKS NEEDED
          BX6    X6+X1
          LX1    -2          11B WHERE BLANKS NEEDED
          BX6    X6+X1       X6 BLANK FILLED TO ZERO BYTE
          SA6    A6          REWRITE LAST WORD
          SA1    WSCB        CHECK B6 SAVE FLAG
          ZR     X1,=XWTX=   EXIT IF CLEAR
          SB6    X1+         ELSE RESTORE B6
          EQ     =XWTX=      EXIT

 WSC4     SX1    1R          BLANK FILL LAST ZERO CHARACTER
          IX6    X6+X1
          SA6    A6          REWRITE LAST WORD
          SX6    B6
          SB6    A1          (A1) = WSCA, (WSCA) = 0 NOW
          SA6    WSCB        SAVE B6
          EQ     WSC2        TRANSFER ONE FINAL ZERO WORD

 WSCA     CON    1R          CARRY-OVER CHARACTER
 WSCB     CON    0           SAVE B6
          SPACE  4
          BASE   *
          IF     -DEF,QUAL$,2
          QUAL   *
 WSC=     =      /COMCWSC/WSC=
 WSC=     ENDX
TLET
COMMON
          CTEXT  TLET
 TLET     SPACE  4,10
***       TEMPORARY LETS.

          USE    /TLET/
 TLET     BSS    0


          USE    *


**        JIGGLE - TABLE OF LETS TO JIGGLE.

          USE    /JIGGLE/
 JIGGLE   BSS    0

          JIGGLE FBBKRK
          JIGGLE FBMOBL
          JIGGLE FKCTRP
          JIGGLE FKPTRP
          JIGGLE FKSATK
          JIGGLE FKSCNQ
          JIGGLE FKSCOF
          JIGGLE FKSFRD
          JIGGLE FKSISO
          JIGGLE FKS2BS
          JIGGLE FNATKD
          JIGGLE FNATNP
          JIGGLE FNBKRK
          JIGGLE FNCTRP
          JIGGLE FNKTRP
          JIGGLE FORKBN
          JIGGLE FPDCQR
          JIGGLE FPDCQN
          JIGGLE FPDCQB
          JIGGLE FPDCQC
          JIGGLE FPDCKC
          JIGGLE FPDCKB
          JIGGLE FPDCKN
          JIGGLE FPDCKR
          JIGGLE FPNBSC
          JIGGLE FPNDBL
          JIGGLE FPNISC
          JIGGLE FPNISO
          JIGGLE FPNLBK
          JIGGLE FPNMAJ
          JIGGLE FPNPSA
          JIGGLE FPNPSB
          JIGGLE FPNPSD
          JIGGLE FPNVBK
          JIGGLE FQKTRP
          JIGGLE FQMOBL
          JIGGLE FRHUNG
          JIGGLE FRKDBL
          JIGGLE FRKOPF
          JIGGLE FRKTRP
          JIGGLE FRKVNP
          JIGGLE FRK7TH
          JIGGLE FRMOBL
          JIGGLE FTMBON

          JIGGLE 0                END OF TABLE
          USE    *
          ENDX
RELY
COMMON
 RELY     SPACE  4
**        COMMON DECK RELY
*
*         CALLED BY
*                CHESS30
*                MASSIO
*                GAMSCR
*                READER
*                INITSWI
*


          USE    /RELY/

 RELFILE  BSS    6                RELIABILITY FILE FET

          BSS    4
 RELFDB   BSS    6                RELFILE FDB

          BSS    4
 LIBFDB   BSS    6                LIBRARY FDB

          BSS    4
 GAMFDB   BSS    6                GAMFILE FDB

 RELSW    BSS    1                RELIABILITY SWITCH
 GAMOUT   BSS    1                GAMFILE OUT POINTER AT LAST EXTEND

          USE    *
LET
COMMON
          CTEXT  LET
 LET      SPACE  4
**        COMMON DECK LET
*
*         CALLED BY
*                READER
*                CHESS30
*                CRECMD
*                YRMOVE
*                INITAL
*                GAMSCR
*                INITSWI
*                EVALU8
*                SWICMD
*                GLOBLET
*                DEBUGR
*                MASSIO
*                MYMOVE
*                CHESS20
*


          MACRO  LET,NAME,MODE
 NAME     BSS    1
          IF     MIC,LETMIC,1
          'LETMIC' NAME,MODE
          ENDM


          USE    /LET/
 LET      BSS    0

 ASKFCT   LET    D                PERCENT TO TIME CONTROL TO ASK TQ
 COPIES   LET    D                NO. OF EXTRA GAME SCORES AT CENTRAL
 EXTRAS   LET    D                SACRED MINIMUM EXTRAT
 EXTRAT   LET    D                MINUTES OF TIME CONTROL BUFFER
 FBBKRK   LET    B                PENALTY FOR BISHOP ON BACK RANK
 FBMOBL   LET    B                BISHOP MOBILITY FACTOR
 FCHKOP   LET    B                MATING KING 2-FROM-EDGE BONUS
 FDEPTH   LET    D                DEPTH OF MAX ITERATION IF TIMING OFF
 FDRAWA   LET    D                DRAW ADJUST MAX MOVE FACTOR
 FDRAWG   LET    B                DRAW ADJUST MAX INCREASE
 FDRAWI   LET    D                DRAW ADJUST IRREVERSIBLE MOVE FACTOR
 FDRAWM   LET    D                DRAW ADJUST MAX MOVES COUNT
 FDRAWT   LET    D                DRAW ADJUST FRACTIONAL TOLERANCE
 FEZTOL   LET    B                THAT-WAS-EASY SCORE TOLERANCE
 FKCORN   LET    B                MATED KING CORNER HATRED
 FKCTRP   LET    B                ENDGAME KING CENTER TROPISM
 FKKNMT   LET    B                MATING KING AND KNIGHT TROPISM
 FKPTRP   LET    B                ENDGAME KING TO PAWN ATTRACTION
 FKSANQ   LET    B                KING SAFETY SANCTUARY FACTOR
 FKSATK   LET    B                KING SECTOR ATTACK FACTOR
 FKSCNQ   LET    B                KING SANCTUARY PENALTY IF CAN CASTLE
 FKSCOF   LET    B                KING OPEN FILE PENALTY IF CAN CASTLE
 FKSCRM   LET    B                KING SAFETY CRAMP FACTOR
 FKSFRD   LET    B                KING SAFETY CLOSE FRIENDS FACTOR
 FKSISO   LET    B                KING WITH ISOLATED PAWN FACTOR
 FKSMAT   LET    B                KING SAFETY ATTACK COUNT THRESHOLD
 FKSMNF   LET    B                KING SAFETY MIN FRIEND COUNT
 FKSOPF   LET    B                KING ON OPEN FILE FACTOR
 FKSQBN   LET    B                KING SAFETY QUEEN PRESENCE FACTOR
 FKS2BS   LET    B                KING FILE WITH 2 BARE SQS PENALTY
 FLXTOL   LET    B                ALPHA-BETA WINDOW MARGIN
 FMAXMT   LET    B                MAX MATERIAL EDGE SANS TRADE BONUS
 FMTEGE   LET    B                MIN EDGE FOR USING MATING ALGORITHM
 FNATKD   LET    B                ATTACKED KNIGHT PENALTY
 FNATNP   LET    B                ATT. KNIGHT UNDEF. BY PAWN PENALTY
 FNBKRK   LET    B                PENALTY FOR KNIGHT ON BACK RANK
 FNCTRP   LET    B                KNIGHT CENTER TROPISM
 FNKTRP   LET    B                KNIGHT KING TROPISM
 FORKBN   LET    B                PENALTY FOR MULTIPLE HUNG PIECES
 FPDCOL   BSS    0                PAWN ADVANCE CREDITS BY COL (8 WDS)
 FPDCQR   LET    B
 FPDCQN   LET    B
 FPDCQB   LET    B
 FPDCQC   LET    B
 FPDCKC   LET    B
 FPDCKB   LET    B
 FPDCKN   LET    B
 FPDCKR   LET    B
 FPNBSC   LET    B                BLOCKED 2ND RANK CENTER PAWN PENALTY
 FPNDBL   LET    B                DOUBLED PAWN PENALTY
 FPNISC   LET    B                LESS-EXPOSED ISOLATED PAWN PENALTY
 FPNISO   LET    B                ISOLATED PAWN PENALTY
 FPNLBK   LET    B                SEMI-FLUID PAWN PENALTY
 FPNMAJ   LET    B                PAWN MAJORITY ADVANCE CREDIT
 FPNPAS   LET    B                PASSED PAWN ROW**2 BONUS
 FPNPSA   LET    B                PASSED PAWN ATTACK NEXT SQ CREDIT
 FPNPSB   LET    B                PASSED PAWN BLOCK CREDIT
 FPNPSD   LET    B                PASSED PAWN DEFEND NEXT SQ CREDIT
 FPNVBK   LET    B                SUPER-OPPOSED PAWN PENALTY
 FQKTRP   LET    B                QUEEN KING TROPISM
 FQMOBL   LET    B                QUEEN MOBILITY FACTOR
 FRHUNG   LET    B                PENALTY FOR HAVING HUNG PIECE
```

```
 FRKDBL   LET    B               DOUBLED ROOKS BONUS
 FRKOPF   LET    B               ROOK ON OPEN FILE BONUS
 FRKTRP   LET    B               ROOK-KING TROPISM
 FRKVNP   LET    B               ROOK ATTACK VULNERABLE PAWN BONUS
 FRK7TH   LET    B               ROOK ON 7TH RANK BONUS
 FRMOBL   LET    B               ROOK MOBILITY FACTOR
 FTMBON   LET    B               TEMPO BONUS FOR SIDE ON MOVE
 FTRADE   LET    B               TRADE-DOWN BONUS FACTOR
 FTRDSL   LET    B               TRADE-DOWN TUNING FACTOR
 FTRPOK   LET    B               PAWN TRADE-DOWN RELAXATION
 FTRPWN   LET    B               PAWN TRADE-DOWN FACTOR
 GCLOCK   LET    D               GAME CLOCK (MINUTES)
 JIGMVS   LET    D               NO. OF NON-TRIVIAL MOVES TO JIGGLE
 JIGSIZ   LET    B               100B * MAX FRACTION TO JIGGLE
 LINEPP   LET    D               LINES TO PRINT PER PAGE
 LOGTRA   LET    D               LOG OF NO. OF TRANS. TABLE SLOTS
 MDEPTH   LET    D               MINIMUM TIME CONTROL SEARCH DEPTH
 MOVNUM   LET    D               CURRENT MOVE NUMBER
 MSCODE   LET    B               LIBRARY MOVE SELECTION CODE
 MSMASK   LET    B               LIBRARY MOVE SELECTION MASK
 MVRL50   LET    D               50 MOVE RULE
 OVRHED   LET    D               SECONDS TO TYPE IN MOVES, ETC
 PERCNT   LET    D               ESTIMATED PERCENTAGE OF CPU
 QADPTH   LET    D               EXTENDED DEPTH FOR SAFE CHECKS
 QDEPTH   LET    D               MAX QUIESCENCE EXTENSION DEPTH
 RANDOM   LET    B               15-BIT PSEUDO-RANDOM NUMBER
 RATEMY   LET    D               CHESS 4.0 ESTIMATED USCF RATING
 RATEPN   LET    D               RATING DIFF. = 100B SCORE
 RATEYR   LET    D               OPPONENTS ESTIMATED USCF RATING.
 RULMV1   LET    D               NUMBER OF MOVES IN FIRST TIME CONTROL
 RULMV2   LET    D               NUMBER OF MOVES IN EXTRA TIME CONTROL
 RULTM1   LET    D               MINUTES IN FIRST TIME CONTROL
 RULTM2   LET    D               MINUTES IN SUBSEQUENT TIME CONTROLS
 STEPHI   LET    D               UPPER PLY DIAGNOSTIC STEPPING LIMIT
 STEPLO   LET    D               LOWER PLY DIAGNOSTIC STEPPING LIMIT
 TGRACE   LET    D               TIME CONTROL GRACE PERIOD IN MS
 TMAXOM   LET    B               100B*MAX MULTIPLE OF TIMPM TO USE
 TPMOKR   LET    B               100B * (RATIO USED TO COMPUTE OTMPM)
 TPMRAT   LET    B               MINIMUM OK TIME-PER-MOVE REDUCTION
 TPMRMX   LET    B               100B * MAX MULT OF MTMPM TO USE
 TPONBN   LET    B               100B * PONDER TIME SHARING BONUS
 TQFRST   LET    D               FIRST TIME QUESTION MOVE NUMBER
 TQNEXT   LET    D               SUBSEQUENT TIME QUESTION MOVE NUMBER
 TRATIC   LET    B               NTRLO ADJUSTMENT INCREMENT
 TRATLO   LET    B               100B * (MIN MS THIS MOVE)/TIMPM
 TRATMN   LET    B               NTRLO MINIMUM
 TRATMX   LET    B               NTRLO MAXIMUM
 TRATOL   LET    B               TIMPM FLUCTUATION TOLERANCE
 TRCLVL   LET    D               LAST TRACING DEPTH + 1
 VALUEX   LET    B               VALUES OF PIECES (EMPTY SQ = 0)
 VALUEP   LET    B
 VALUER   LET    B
 VALUEN   LET    B
 VALUEB   LET    B
 VALUEQ   LET    B
 VALUEK   LET    B
 WODREL   LET    B               WOOD SEARCH RELIABILITY

          USE    *

          ENDX
KEYCOM
COMMON
 KEYCOM   SPACE  4
**        COMMON DECK KEYCOM
*
*         CALLED BY
*                CHESS11
*                DUDKBP
*                DUDVAL

          USE    /KEYCOM/
 RESET    BSS    1
 GLOBESW  BSS    1
          USE    *
TABLE
COMMON
 TABLE    SPACE  4,12
**        COMMON DECK TABLE
*
*         CALLED BY
*                CHESS
*                CHESS20
*

          USE    /TABLE/
 TABLE    BSS    9
          USE    *
STACKS
COMMON
          CTEXT  STACKS
 STACKS   SPACE  4
***       STACKS - COMMON BLOCK OF SCRATCH STORAGE.
*
*
*         CALLED BY
*                CREATE
*                EVALU8

          USE    /STACKS/
 STACK1   BSS    64
          USE    *
          ENDX
SQRLST
COMMON
          CTEXT  SQRLST
 SQRLST   SPACE  4,88
***       SQRLST - COMMON BLOCK OF SQUARE LISTS, ETC.
*
*         CALLED BY
*                CODEDIO
*                ENGGEN
*                YRMOVE
*                CREATE
*                LGLENG
*                CHESS11
*                INITAL
*                CHESS
*                LOADR
*                CRECMD
*                CHESS20
*                MINENG
*                MYMOVE
*                INITSWI
*                EVALU8
*                DEBUGR
*                GAMSCR
*

          USE    /SQRLST/
 SQRLSTB  BSS    0               SQUARE LIST BLOCK

 GLOBNK   BSS    0               GLOBAL TABLES
 NSRCH    BSS    NPLY            NO. OF MOVES SEARCHED FROM EACH NODE
 LIMBL    BSS    NPLY            LENGTHS OF LIMB ARRAY ROWS
 LIMBA    BSS    NPLY            FWA OF LIMB ARRAY ROWS
 VARIE    BSS    NPLY            CURRENT VARIATION BEING STUDIED
 SQRLT    BSS    1               FWA SQUARE LIST STACK
 CNTMV    BSS    1               TOTAL NUMBER OF COMPUTED MOVES
 TIMMV    BSS    1               TIME TO COMPUTE LAST MOVE
 TIMTO    BSS    1               TOTAL TIME
 TIMPS    BSS    1               PSEUDO-TOTAL TIME
 TIMPM    BSS    1               REMAINING MSEC/MOVE
 TILTC    BSS    1               MOVES UNTIL TIME CONTROL
 ITERS    BSS    1               NUMBER OF SEARCH ITERATIONS
 ATMPM    BSS    1               AVERAGE TIME PER MOVE
 NTRLO    BSS    1               NOMINAL MIN TIME RATIO
 MTMPM    BSS    1               MAXIMUM TIMPM TO USE
 OTMPM    BSS    1               OK (ACCEPTABLE MIN) TIME-PER-MOVE
 BSTMV    BSS    1               BEST MOVE (PLY ZERO)
 MBHSH    BSS    128             MATERIAL BALANCE HASH TABLE
 REPPB    BSS    LE.PAC*NHIS+LE.PAC*NPLY PACKED BOARDS FOR REPEAT CHECK
 REPPZ    BSS    1               FWA OF PLY ZERO PACKED BOARD
 WIERD    BSS    1               SPECIAL CONDITION FLAGS
 CSIDE    BSS    1               LAST COMPUTER SSIDE
 MAINV    BSS    1               NON-ZERO IF STILL DOING MAINVAR FULLS
 KILLS    BSS    NPLY+1          RECENT REFUTATIONS OR MAINVAR
 DRAWS    BSS    1               VALUE OF DRAW
 OBVIO    BSS    1               NON-ZERO IF MOVE IS OBVIOUS
 SAVES    BSS    1               SAVED SCORE FROM LAST MOVE
 MVNTR    BSS    1               NO. OF NON-TRIVIAL MOVES SO FAR
 PONDR    BSS    1               PREDICTED OPPONENTS MOVE
 SCORI    BSS    1               SCORE FROM LAST ITERATION (OR MOVE)
 NXTMV    BSS    1               NEXT PLAYERS MOVE
 PBSMV    BSS    1               MACHINE RESPONSE TO PONDERED MOVE.
 PSCOR    BSS    1               SCORE OF PBSMV
 PPNDR    BSS    1               TEMP SAVED NEXT PONDR
 PKILL    BSS    1               TEMP SAVED NEXT MACHINE REJOINDER
 PNDPAU   BSS    2               PNDPAU ENTRY POINT
 PNDSAV   BSS    4               PONDER SAVE AREA
 MYMVTM   BSS    2               MYMOVE START, STOP TIMES
 TIMPO    BSS    1               PONDER FREE TIME
 PVMOV    BSS    1               SAVE PONDER VMOVE WORD
 TIMIN    BSS    1               MIN MS TO SPEND ON MOVE
 TIMAX    BSS    1               MAX MS TO SPEND ON MOVE
 TIMLI    BSS    1               START TIME OF CURRENT ITERATION
 PWIRD    BSS    1               SAVED PONDR WIERD WORD
 PTMMV    BSS    1               SAVED PONDERED MOVE TIME

 GLOBNKE  BSS    0
 GLOBNKL  EQU    GLOBNKE-GLOBNK


 SQRLST   BSS    0
 ABLOC    BSS    2               ALL BLACK PIECES
 ALLOC    BSS    2               ALL PIECES
 AWLOC    BSS    2               ALL WHITE PIECES
 STOPS    BSS    2               ALL PIECES AND EDGE OF BOARD
 ABATK    BSS    2               ALL BLACK ATTACKS
 ALATK    BSS    0               BASE ADDRESS OF ATK LISTS
 CSTAT    BSS    2               CASTLE STATUS BITS
 AWATK    BSS    2               ALL WHITE ATTACKS

          ECHO   1,T=(K,Q,B,N,R,P)
 T!BLOC    BSS    2                ALL BLACK T LOCATIONS

 TPLOC    BSS    0               BASE ADDRESS FOR LOC SQUARE LISTS
 ENPAS    BSS    2               EN PASSANT SQUARES

          ECHO   1,T=(P,R,N,B,Q,K)
 T!WLOC    BSS    2                ALL WHITE T LOCATIONS

          ECHO   1,T=(WM,WR,WQ,BM,BR,BQ)
 T!DNI    BSS    2               SQUARES DENIED TO T

 ATKFR    BSS    2*64            ALL ATTACKS FROM EACH SQUARE
 ATKTO    BSS    2*64            ALL ATTACKS TO EACH SQUARE

          BSS    1               DUMMY
 NBOARD   BSS    64              THE LOOK-AHEAD BOARD

 WPSIG    BSS    1               WHITE PAWN SIGNATURE
 BPSIG    BSS    1               BLACK PAWN SIGNATURE

 WKSQR    BSS    1               WHITE KING SQUARE
 BKSQR    BSS    1               BLACK KING SQUARE

 PSMOD    BSS    1               PAWN STRUCTURE MODIFICATIONS FILES.

          ECHO   1,T=(QR,QN,QB,QC,KC,KB,KN,KR)
 PS!T!F    BSS    1                PAWN STRUCTURE FOR T FILE

 LPECS    BSS    1               LENGTH OF PREVIOUS PLY ECS
 PRECS    BSS    1               ECS ADDRESS OF PREVIOUS PLY
 SLECS    BSS    1               ECS ADDRESS OF CURRENT SQUARE LISTS
 VAECS    BSS    1               ECS ADDRESS OF VARIABLE AREA
 MSIDE    BSS    1               +2 IF WHITE TO MOVE, -2 IF BLACK
 OSIDE    BSS    1               0 IF WHITE TO MOVE, 1 IF BLACK
 SSIDE    BSS    1               +0 IF WHITE TO MOVE, -0 IF BLACK
 NPLYC    BSS    1               CURRENT PLY NUMBER
 MBVAL    BSS    1               MATERIAL BALANCE SIGNATURE AND VALUES
 MBSCR    BSS    1               MATERIAL BALANCE SCORE
 CNT50    BSS    1               MOVE COUNTS SINCE LAST IRREVERSIBLE
 REPPN    BSS    1               FWA OF PLY NPLYC PACKED BOARD
 REPST    BSS    1               FWA OF LAST IRREVERSABLE PACKED BOARD
 TRASH    BSS    1               TRANSPOSITIONS HASH VALUE
 TRAWD    BSS    1               TRANSPOSITIONS INFORMATION WORD
 SALPH    BSS    1               SAVED STARTING ALPHA
 SBETA    BSS    1               SAVED STARTING BETAR

 SQRLSTE  BSS    0
 SQRLSTL  EQU    SQRLSTE-SQRLST

 NODEBK   BSS    0

 VMOVE    BSS    1               MOVE LEADING TO THIS NODE
 SCORE    BSS    1               EVALU8 SCORE FOR THAT MOVE
 ALPHA    BSS    1
 BETAR    BSS    1               MUST CONTIGUOUSLY FOLLOW ALPHA
 INDEX    BSS    1               ADDRESS OF CURRENT MOVE
 GENFR    BSS    2               SQUARES TO GENERATE MOVES FROM
 GENTO    BSS    2               SQUARES TO GENERATE MOVES TO
 TRSRCH   BSS    2               TREE SEARCH ENTRY POINT
 EVALU8   BSS    2               EVALUATOR ENTRY POINT
```

```
 DEBUGR   BSS    2                DEBUGGER ENTRY POINT
 LINDX    BSS    1                CURRENT LWA+1 OF MOVES ARRAY
 NPLYD    BSS    1                NUMBER OF PLIES DEEPER THAN BASE
 NDTMP    BSS    10               TEMPORARY STORAGE PRIVATE TO NODE
 SAVA0    BSS    1                SAVE STACK DEPTH FOR EVALU8
 ESTAK    BSS    3                REGISTER SAVE STACK FOR EVALU8

 NODEBKE  BSS    0
 NODEBKL  EQU    NODEBKE-NODEBK

 MOVES    BSS    260*LE.MOV

 SQRLSTBE BSS    0                END OF SQUARE LIST BLOCK
 SQRLSTBL EQU    SQRLSTBE-SQRLSTB  LENGTH OF SQUARE LIST BLOCK
          ERRPL  SQRLSTBE-SQRLST-2000B  7600 CANT HACK BIG TRANSFER
          USE    *
          ENDX
CONST
COMMON
          CTEXT  CONST
 CONST    SPACE  4
***       CONST - COMMON BLOCK OF CONSTANT DATA.
*
*         SEE INITCON FOR VALUES.
*
*         CALLED BY
*                CREATE
*                CHESS11
*                INITCON
*                EVALU8
*                YRMOVE

          USE    /CONST/
          BSS    77
 DIRTB    BSS    78               DIRECTIONS BETWEEN SQUARES

 SQ2PB    BSS    0                SQUARE LOCATIONS IN PACKED BOARDS
 EXPND    BSS    64               8 X 8 TO 10 X 12 TRANSLATION

 ONBRD    BSS    2                THE SQUARES ON THE BOARD

 EIGHT    BSS    2                PROMOTION SQUARES

          BSS    6
 MVDIR    BSS    7                MOVE DIRECTIONS

          BSS    6
 MVMSK    BSS    7                MOVE MASKS

          BSS    6
 CNCHK    BSS    7                CHECK DIRECTION MASKS

 SQ2PS    BSS    2*64             SQUARE TO PAWN STRUCTURE TRANSLATION

 SQ2BT    BSS    64               SQUARE TO BIT TRANSLATION

 BT2SQ    BSS    120              BIT TO SQUARE TRANSLATION

 PSMSK    BSS    1                PAWN STRUCTURE VALUES MASK
 MBMSK    BSS    1                MATERIAL BALANCE VALUE MASK
          USE    *

          ENDX
BOARD
COMMON
          CTEXT  BOARD
 BOARD    SPACE  4
***       BOARD - COMMON BLOCK WITH THE ACTUAL POSITION.
*
*
*         CALLED BY
*                CREATE
*                CHESS11
*                LGLENG
*                INITSWI
*                INITAL
*                MASSIO
*                CRECMD
*                CHESS20
*                MYMOVE
*                YRMOVE
*                EVALU8
*                MINENG

          USE    /BOARD/
 BOARD    BSS    64               THE ACTUAL POSITION
 STATUS   BSS    1                STATUSES:
*                                 1/WHITE ON MOVE
*                                 1/UNUSED
*                                 1/WHITE IN CHECK
*                                 9/WHITE EP SQUARE (RANK 3)
*                                 9/WHITE CASTLE SQUARES (RANK 1)
*                                 9/50-MOVE RULE COUNT
*                                 1/BLACK ON MOVE
*                                 1/UNUSED
*                                 1/BLACK IN CHECK
*                                 9/BLACK EP SQUARE (RANK 6)
*                                 9/BLACK CASTLE SQUARES (RANK 8)
*                                 9/50-MOVE RULE COUNT
*

          USE    *
          ENDX
VERSION
COMMON
          CTEXT  VERSION
 VERSION  SPACE  4,10
**        COMMON DECK VERSION
*
*         CALLED BY
*                CODEDIO
*                CHESS11
*                EVALU8
*                CHESS40
*                GAMSCR

 V        MICRO  1,, 4.6          VERSION NUMBER

          ENDX
DISLINK
COMMON
 DISLINK  SPACE  4
**        COMMON DECK DISLINK
*
*         CALLED BY
*                CHESS11

 LABLS    DISLINK
 BOARD    DISLINK
 INFO     DISLINK
 PIECES   DISLINK
 TWINK    DISLINK
 SQRL     DISLINK
 MOVS     DISLINK
LINK
COMMON
 LINK     SPACE  4
**        COMMON DECK LINK
*
*         CALLED BY
*                CHESS11
*                DUDLINK

 TWINKLE  LINK
 DUDREAD  LINK
 DUDCLR   LINK
 DUDMSG   LINK
 DUDERR   LINK
 DUDDROP  LINK
 DUDKILL  LINK
 DUDSET   LINK
 DUDINFO  LINK
 DUDSQRL  LINK
 DUDDISP  LINK
 DUDMOVE  LINK
 DUDECS   LINK
MEMORY
COMMON
 MEMORY   SPACE  4,10
**        COMMON DECK MEMORY
*
*         CALLED BY
*                CODEDIO
*                DEBUGR
*                CHESS11
*                CHESS20
*                READER
*                MASSIO
*                MTR*
*                LOADR
*                CHESS30

          USE    /MEMORY/
 MEMORY   BSS    1                MEM CM STATUS WORD
          BSS    1                MEM ECS STATUS WORD
          BSS    1                FAKE ECS FWA
          BSS    1                FAKE ECS LWA+1
          USE    *
DISPMAC
COMMON
**        COMMON DECK DISPMAC
*
*         CALLED BY
*                CHESS11
          TITLE                   DISPLA - MACRO TO GENERATE DISPLAY BU
,FFERS POINTERS
**        DISPLA - MACRO TO GENERATE DISPLAY BUFFER POINTERS
          SPACE  2
          MACRO  DISPLA,LABL,SCREEN,MODE,SIZE,INTENS,FWA,LWA,ONOFF
 L        IFC    NE,*LABL**
          IF     -DEF,LABL,1
 LABL     BSS    1
          ORG    LABL
 L        ENDIF
 ONOFF    IFC    EQ,*ONOFF*OFF*
 +        VFD    1/1
 ONOFF    ELSE
 +        VFD    1/0
 ONOFF    ENDIF
          VFD    1/0
          VFD    4/INTENS
          VFD    2/0
 XXX      MICRO  1,1,*!SCREEN!*
 SCREEN   IFC    EQ,*'XXX'*R*
          VFD    1/1
 SCREEN   ELSE
          VFD    1/0
 SCREEN   ENDIF
 XXX      MICRO  1,1,*!MODE!*
 MODE     IFC    EQ,*'XXX'*D*
          VFD    1/1
          VFD    2/1
 MODE     ELSE
          VFD    1/0
 XXX      MICRO  1,1,*!SIZE!*
 NNN      SET    0
          IFC    EQ,*'XXX'*S*,1
 NNN      SET    1
          IFC    EQ,*'XXX'*M*,1
 NNN      SET    2
          IFC    EQ,*'XXX'*L*,1
 NNN      SET    3
          VFD    2/NNN
 MODE     ENDIF
          VFD    18/FWA
          VFD    30/LWA
          ENDM
          SPACE  4
          MACRO  DISPL,LABL,MODE,SIZE,INTENS,DYNAMIC

 FWA      MICRO  1,,*DYNAMIC!LABL$*
 LWA      MICRO  1,,*DYNAMIC!LABL.*

 L=LABL   DISPLA LEFT,MODE,SIZE,INTENS,'FWA','LWA',OFF
 R=LABL   DISPLA RIGHT,MODE,SIZE,INTENS,'FWA','LWA',OFF

          ENDM
DUDCOM
COMMON
 DUDCOM   SPACE  4
**        COMMON DECK DUDCOM
*
*         CALLED BY
*                CHESS11
*                DUDKBP
*                DUDVAL
*                RCP
          SPACE  2
**        DUD INTERFACE FUNCTION CODES
          SPACE  2
 F=NOP    EQU    0B               NO OPERATION
 F=INT    EQU    2B               INTERRUPT
 F=ION    EQU    4B               ENABLE INTERRUPT
 F=IOF    EQU    6B               DISABLE INTERRUPT
```

```
 F=RCP    EQU    10B              RESTORE CENTRAL PROCESSOR
 F=RPR    EQU    12B              REQUEST PRIORITY
 F=REM    EQU    14B              REQUEST EXIT MODE
 F=HOLD   EQU    16B              HOLD DISPLAY
 F=DROP   EQU    20B              DROP PP
 F=KILL   EQU    22B              KILL JOB
CHARMAC
COMMON
 CHARMAC  SPACE  4
**        COMMON DECK CHARMAC
*
*         CALLED BY
*                CHESS11
*                CHSBRD

          TITLE                   MACROS
**        CHAR - MACRO TO GENERATE CHARACTER DISPLAY DATA
          SPACE  2
          PURGMAC CHAR
 CHAR     MACRO  X,Y,STRING
          IFC    NE,*!X!**,1
          VFD    12/6000B+X
          IFC    NE,*!Y!**,1
          VFD    12/7000B+Y
 NNN      SET    1
          DUP    1000
 XXX      MICRO  NNN,2,"!STRING!"
 XXX      IFC    NE,"'XXX'""
          VFD    12/2R'XXX'
 NNN      SET    NNN+2
 XXX      ELSE
          STOPDUP
 XXX      ENDIF
          ENDD
          ENDM
          SPACE  2
**        FILL - MACRO TO ZERO FILL LAST WORD IN BUFFER
          SPACE  2
 FILL     MACRO
 NNN      SET    $+1
          IFNE   NNN,60,1
          VFD    NNN/0
          ENDM
KEYMACS
COMMON
 KEYMACS  SPACE  4
**        COMMON DECK KEYMACS
*
*         CALLED BY
*                GLOBCON
*                GLOBLET
*                GLOBSWI
*                QUEST0
*                QUEST1
*                GLOBAL

          MACRO  MEMBER,LABL,CHAR,OFF,SIB
          LOCAL  A,B,C,D
 INST     SET    INST+1
 A        EQU    INST
          IFC    NE,$$LABL$,1
 LABL     EQU    INST
 SIBL     IFC    EQ,"!SIB!"*"
 SIBL     MICRO  1,,"!A!+1"
 SIBL     ELSE
 SIBL     MICRO  1,,"!SIB!"
 SIBL     ENDIF
 OFFS     IFC    EQ,"!OFF!"*"
 OFFS     MICRO  1,,"!A!+1"
 OFFS     ELSE
 OFFS     MICRO  1,,"!OFF!"
 OFFS     ENDIF
 B        MICRO  1,,"'SIBL'"
 C        MICRO  1,,"'OFFS'"
 X        MICRO  2,1,"!CHAR!"
 D        IFC    EQ,"'X'""
 D        MICRO  1,,"1R!CHAR!"
 D        ELSE
 D        MICRO  1,,"!CHAR!B"
 D        ENDIF
          RMT
          IF     -DEF,'B',1
 'B'      EQU    0
          VFD    6/'D',12/'B',12/'C' LABL !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
          RMT
          ENDM

 HEREM    MACRO
          USE
          HERE
          ENDM

          MACRO  LINE,LABEL,STRING,MATCH

 LABEL    IFC    NE,"!LABEL!""
 LABL     MICRO  1,,"!LABEL!"
 I        SET    1
          DUP    MAXLINE
 MI       MICRO  I,2,*0102030405060708091011121314151617181920*
 A'MI'    MICRO  1,,**
 B'MI'    MICRO  1,,**
 I        SET    I+2
          ENDD
 LABEL    ELSE
 LABL     MICRO  1,,**
 LABEL    ENDIF

 I        SET    1
 J        SET    0

          DUP    MAXLINE
 MI       MICRO  I,2,*0102030405060708091011121314151617181920*
 KHAR     MICRO  'MI',1,"!STRING!"
 NEXT     MICRO  'MI'+1,1,"!STRING!"

 TRUE     IC     EQ,"'NEXT'""
 TRUE     MICRO  1,,"!MATCH!"
 TRUE     ELSE
 TRUE     MICRO  1,,"*"
 TRUE     ENDIF

 'LABL'   CHAR   (!STRING!),'MI',('KHAR'),('TRUE')
 LABL     MICRO  1,,**

          IFC    EQ,"'NEXT'"",1
          STOPDUP
 I        SET    I+2
          ENDD

          ENDM
          SPACE  4
          PURGMAC CHAR
          MACRO  CHAR,LABL,STRING,NUM,KHAR,TRUE
          LOCAL  FALSE

 C!NUM    MICRO  1,NUM,"!STRING!"

 1        IFC    NE,"'A!NUM'"'C!NUM'"
 2        IFEQ   J,0
 LAB      MICRO  1,,*'B!NUM'*
 J        SET    1
 2        ELSE
 LAB      MICRO  1,,**
 2        ENDIF

 A!NUM    MICRO  1,,"'C!NUM'"
 B!NUM    MICRO  1,,"!FALSE!"
 'LAB'    MEMBER (!KHAR!),TRUE,FALSE
          IFC    NE,"!LABL!"",1
 LABL     EQU    INST

 1        ENDIF
          ENDM
          SPACE  4
          MACRO  CHOICE,LABEL,STRING,TRUE,FALSE
          LOCAL  OK

 HIT      IFC    EQ,"!TRUE!"*"
 HIT      MICRO  1,,*OK*
 HIT      ELSE
 HIT      MICRO  1,,"!TRUE!"
 HIT      ENDIF

 LABL     IFC    NE,"!LABEL!""
 LABL     MICRO  1,,"!LABEL!"
 LABL     ELSE
 LABL     MICRO  1,,**
 LABL     ENDIF

 I        SET    1

          DUP    60
 THIS     MICRO  I,1,"!STRING!"
 NEXT     MICRO  I+1,1,"!STRING!"

 MISS     IFC    EQ,"'NEXT'""
 MISS     MICRO  1,,"!FALSE!"
 MISS     ELSE
 MISS     MICRO  1,,"*"
 MISS     ENDIF

 'LABL'   MEMBER ('THIS'),'HIT','MISS'

          IFC    EQ,"'NEXT'"",1
          STOPDUP

 LABL     MICRO  1,,**
 I        SET    I+1
          ENDD
 OK       EQU    INST+1
          ENDM
 INST     SET    0
          SPACE  4
 START.   BSS    0
 MAXLINE  SET    11
          LIST   A,G,-R
          SPACE  4
IOCOM
COMMON
          CTEXT  IOCOM
 IOCOM    SPACE  4
**        COMMON DECK IOCOM
*
*         CALLED BY
*                DUDLINK
*                CHESS30
*                EVALU8
*                MYMOVE
*                GAMSCR
*                READER
*                INITAL
*                CHESS20
*                DEBUGR
*                CODEDIO
*                INITSWI
*                CRECMD
*                YRMOVE
*                MINENG
*                CREATE
*                SWICMD
*                ENGGEN
*                LOADR

          USE    /IOCOM/
 ILINE    BSS    9                INPUT LINE
 OLINE    BSS    14               OUTPUT LINE (SCRATCH)
 INPFN    BSS    1                INPUT FILE NAME

 MOVMS    BSS    4                MOVE MESSAGE
 DBGMS    BSS    4                DEBUG MESSAGE
 ENDMS    BSS    4                END OF GAME MESSAGE
 HMTMS    BSS    4                HOW MUCH TIME MESSAGE
 GAMMS    BSS    7                GAME SCORE PARTIAL LINE
 GMCNT    BSS    1                NUMBER OF LINES ON GAMFILE
 SWICH    BSS    1                MISCELLANEOUS SWITCHES
 GOCNT    BSS    1                NUMBER OF MOVES TO MAKE
 MTIDS    BSS    1                MULTI-TERMINAL IDENTS
 MTLIM    BSS    1                MULTI-TERMINAL LIMIT
 OPNAM    BSS    1                OPPONENTS NAME
 ALTFN    BSS    1                ALTER FILE NAME DEFAULT
 LINES    BSS    1                LINES REMAINING ON PAGE
 CPRPV    BSS    1                RPV WORD FROM CTL PT
 CPESP    BSS    1                ESP WORD FROM CTL PT
                                  TELEX INTERRUPT WORD ON KRONOS
 RPVNO    BSS    1                REPRIEVE FAILURE FLAG
          USE    *
          ENDX
DEBUG
COMMON
          CTEXT  DEBUG
 DEBUG    SPACE  4,10
**        DEBUG - BLOCK OF DEBUGGING DATA.
*
```

```
*         CALLED BY
*                INITSWI
*                EVALU8
*                MYMOVE
*                YRMOVE
*

          USE    /DEBUG/

 NDTTL    BSS    1                TOTAL NODE COUNT FOR THIS MOVE
 NDCNT    BSS    NPLY             NODE COUNTS PER PLY
 NDDIT    BSS    NPLY/2+2         NODE COUNTS PER DEPTH ITERATION
 PVARY    BSS    NPLY+1           PRINCIPLE VARIATION
 NDSCR    BSS    1                NODES PASSED TO POSITIONAL EVALUATOR
 NDREG    BSS    1                NODES SCORED BY REGULAR EVALUATOR
 NDMOP    BSS    1                NODES SCORED BY CHECKMATE ALGORITHM
 POSINI   BSS    1                AVERAGE OF PREL POSITIONAL SCORES
 POSMIN   BSS    1                MIN POSITIONAL SCORE IN TREE
 POSMAX   BSS    1                MAX POSITIONAL SCORE IN TREE
 POSCNT   BSS    1                NO. OF POSITIONAL SCORES

          ERRNZ  POSMAX-POSMIN-1

          USE    *
          ENDX
USEIML
COMMON
          USE    /IML/
 SCRATCH  CON    0                SCRATCH FOR SAVING X5
 DF1      CON    -1               DISPLAY FILE 1 (FOR BOARD AND PIECES)
 ANITIM   CON    60               NO OF MSEC TO WAIT FOR ANIMATION
 NIN      BSS    1                NO OF INCREMENTS FOR ANIMATION
 OUT      BSS    1
 MSG      BSS    1                HOLDS ADDRESS OF MESSAGE
 MFLAG    BSS    1                FLAG
 IX       BSS    1                X-COORD
 IY       BSS    1                Y-COORD
 BNAME    BSS    1                HOLDS BLOCK NAME
 BCNAME   BSS    1                BLOCK OR FILE NAME
 LNAME    BSS    1                HOLDS LABEL NAME
 IHIT     BSS    1                ANIMATION LOOP CNTR & X-BUT INDICATOR
 XIN      BSS    1                TEMPORARY
 YIN      BSS    1                TEMPORARY
CALL
COMMON
***       CALL - MACRO TO CALL GYPSY ROUTINES USING THE RUN FORTRAN
*                CALLING SEQUENCE.  THIS CAN BE CHANGED WHENEVER GYPSY
*                IS FIXED TO ASSEMBLE WITHOUT AN F PARAMETER.  UP TO
*                SIX AND EIGHT PARAMETERS CAN BE PASSED:
*                     CALL   NAME,(P1, ... , P6),P7,P8
          SPACE  4
          NOREF  CNT
          PURGMAC CALL
 CALL     MACRO  NAME,PARAMS,P7,P8
 CNT      SET    0

          ECHO   ,I=(PARAMS)
 CNT      SET    CNT+1
          SB.CNT I
          ENDD

 CALL     IFC    NE, P7
          SX6    P7
          SA6    =X!NAME-4
          SX6    P8
          SA6    =X!NAME-3
 CALL     ENDIF

          RJ     =X!NAME
          ENDM
CHESTXT
*FILE TINP
          IDENT  CHESTXT
 CHESTXT  TITLE  CHESTXT - SYSTEMS TEXT FOR CHESS.
          STEXT
          ABS
          LIST   D,E,F,G,X
*
*         CHESS 4.5.
*
*         COPYRIGHT NORTHWESTERN UNIVERSITY 1976, ALL RIGHTS RESERVED.
*CALL     COMMCIO
*CALL     COMMDTR
*CALL     COMMJCM
*CALL     COMMCPM
*CALL     COMMSBR
*CALL     COMMSYS
*CALL     COMMTXT
 PURGMAC  SPACE  4
 NUCC     TITLE  INSTALLATION DEPENDANCIES.
***       INSTALLATION DEPENDENCIES.
*

          SKIP   1                REMOVE FOR NUCC
 NUCC     EQU    1                NUCC SYSTEM OPERATION

          SKIP   1                REMOVE FOR DUD DISPLAYS
 DISPLAY  EQU    1                DUD DISPLAY OPERATION

          SKIP   1                REMOVE FOR ONE-LINE
 ONELINE  EQU    1                ONE-LINE TIME-HOGGING SYSTEM

          IF     DEF,NUCC,1
 IMLAC    EQU    1                IMLAC PDS-1 OPERATION

          SKIP   1                REMOVE TO SUPPRESS ECS REQUESTS
 ECS      EQU    0                NO ECS REQUESTS
*         SKIP   1                REMOVE FOR AUTOMATIC ECS
 ECS      EQU    1                REQUEST ECS WITHOUT ASKING

*         SKIP   1                REMOVE FOR KRONOS OPERATION
 KRONOS   EQU    1                KRONOS/NOS OPERATION

*         SKIP   1                REMOVE FOR T-DISPLAY CODE
 TDISPLAY EQU    1                KRONOS DIS T-DISPLAY

*         SKIP   1                REMOVE FOR OVERLAY MERGE
 LIBGEN   EQU    1                MERGE 1,0 AND 2,0 OVERLAYS

*         SKIP   1                REMOVE FOR SNEAKY ECS INPUT PATH
 MAILBOX  EQU    1                GRAB INPUT VIA ECS

*         SKIP   1                REMOVE FOR ELECTRONIC CHESS BOARD
 CBD      EQU    1                ELECTRONIC CHESS BOARD INPUT


*         VALIDATE DEPENDENT OPTIONS.

          IF     DEF,LIBGEN,4      NO DISPLAYS PERMITTED WITH LIBGEN
          IF     DEF,DISPLAY,1
          ERR    LIBGEN - DISPLAY CONFLICT.
          IF     DEF,IMLAC,1
          ERR    LIBGEN - IMLAC CONFILICT.
          SPACE  4
 NUCC     IF     -DEF,NUCC
***       NORTHWESTERN UNIVERSITY SPECIAL SYSTEM MACROES.
*

          PURGMAC READAL
          PURGMAC READIF
          PURGMAC READER
          PURGMAC WRITAL

          PURGMAC ATIME
          PURGMAC COST
          PURGMAC CPAREA
          PURGMAC JNAME
          PURGMAC PPTIME

 ATIME    MACRO  ADDR
          RJ     =XCPTIME
          SA6    ADDR
          ENDM
 READCW   SPACE  4,20
***       READAL AND WRITAL FOR KRONOS.
*


 KRONOS   IF     DEF,KRONOS
 READAL   MACRO  F,L
          R=  X2,F
          IFC    EQ,  L ,2
          SX7    200B
          SKIP   1
          SX7    -200B
          RJ     =XCIO=
          ENDM


 WRITAL   MACRO  F,L
          R=  X2,F
          IFC    EQ,  L ,2
          SX7    204B
          SKIP   1
          SX7    -204B
          RJ     =XCIO=
          ENDM
 KRONOS   ENDIF
 NUCC     SPACE  4

          PURGMAC JCM=A
          PURGMAC JCM=B
          PURGMAC JCM=C
          PURGMAC ^?CPM"DE
          PURGMAC ^?CPM"ML
          PURGMAC ^?CPM"M1
          PURGMAC ^?CPM"SW
          PURGMAC ^?CPM"TL
          PURGMAC ^?CPM"IR
          PURGMAC REQUEST
          PURGMAC ASSIGN
***       COMPASS PSEUDO-OPS IN MACRO FORM.
*


**        CONDITIONAL EQUATE.

          MACRO  CEQU,LABL,VALUE
          IF     -DEF,LABL,1
 LABL     EQU    VALUE
          ENDM

**        SET INCLUSIVE OR.

          MACRO  SIOR,LABL,A,B
          LOCAL  I,J,K
 LABL     SET    0
 I        SET    1
 J        SET    A
 K        SET    B

          DUP    21
          IFEQ   J/2*2,J,1
          IFNE   K/2*2,K,1
 LABL     SET    LABL+I

 J        SET    J/2
 K        SET    K/2
 I        SET    I*2
          ENDD

          ENDM

**        CONVERT BIT NUMBER TO MASK.

          MACRO  BIT,LABL,NUM
          LOCAL  A

 A        DECMIC NUM
 LABL     EQU    1S'A'
          ENDM

 NUCC     ENDIF
 CHESTXT  TTL    CHESTXT - SYSTEMS TEXT FOR CHESS.
 CHESTXT  TITLE  MACROS FOR CHESS.

***       CHESTXT - MACROS FOR CHESS.
*
*         BASED  ON MLISP SYSTEM OF NYU.

***       BASIC BUILDING BLOCK MACROS.
*
*         STACK  SYMBOLS AND MICRO DEFINITIONS.
*
*         .SX    MICRO STRING OF X-REGISTERS, TOP FIRST.
*         .FX    MICRO STRING OF FREE X-REGISTERS.
*         .PX    INTER-STATEMENT VALUE OF .FX.
*         .FB    MICRO STRING OF FREE B-REGISTERS.
*         .PB    INTER-STATEMENT VALUE OF .FB.
*         .FM    FIRST FREE MEMORY LOCATION IN STACK.
*         .PDB   ADDRESS OF BASE OF STACK IN MEMORY.
*         .SFM   FIRST FREE MEMORY LOCATION AT STATEMENT START.
*         (A0) = BASE OF STACK MEMORY AT ROUTINE START.
```

```
**        .RPX - REMOVE X-REGISTER FROM POOL.

 .RPX     MACRO  R
 .LFX     MICCNT .FX
          ERRZR  .LFX             IF NO FREE REGISTERS
 R        MICRO  1,1,*'.FX'*
 .FX      MICRO  2,,*'.FX'*
 .RPX     ENDM

**        .APX - ADD X-REGISTER TO POOL.

 .APX     MACRO  R
 .FX      MICRO  1,,*'.FX''R'*
 .APX     ENDM

**        .RTX - REMOVE X-REGISTER FROM STACK TOP.

 .RTX     MACRO  R
 .LSX     MICCNT .SX
          ERRZR  .LSX
 R        MICRO  1,1,*'.SX'*
 .SX      MICRO  2,,*'.SX'*
 .RTX     ENDM

**        .RBX - REMOVE X-REGISTER FROM STACK BOTTOM.

 .RBX     MACRO  R
 .LSX     MICCNT .SX
          ERRZR  .LSX
 R        MICRO  .LSX,1,*'.SX'*
          IFEQ   .LSX,1,2
 .SX      MICRO  1,1,**
          DUP    0,1
 .SX      MICRO  1,.LSX-1,*'.SX'*
 .RBX     ENDM

**        .ATX - ADD X-REGISTER TO TOP OF STACK.

 .ATX     MACRO  R
 .SX      MICRO  1,,*'R''.SX'*
 .ATX     ENDM

**        .ABX - ADD X-REGISTER TO BOTTOM OF STACK.

 .ABX     MACRO  R
 .SX      MICRO  1,,*'.SX''R'*
 .ABX     ENDM

**        .GTX - GET TOP STACK X-REGISTER.

 .GTX     MACRO  R
 .LSX     MICCNT .SX
          ERRZR  .LSX
 R        MICRO  1,1,*'.SX'*
 .GTX     ENDM

**        .LBX - LOAD BOTTOM X-REGISTER FROM STACK.

 .LBX     MACRO
          ERRZR  .FM
          .RPX   .BR
          .ABX   .BR
 .FM      SET    .FM-1
          SA'.BR'    '.PDB'+.FM'.SUB'
 .LBX     ENDM

**        .SBX - STORE BOTTOM X-REGISTER INTO STACK.

 .SBX     MACRO
          .RBX   .BR
 .RSXA    MICMIC .RSX'.BR'
          IFC    NE, '.RSXA'  ,1
          ERR    CANT STORE RESERVED REGISTER
          .APX   .BR
          BX7    X'.BR'
          SA7    '.PDB'+.FM'.SUB'
 .FM      SET    .FM+1
 .SBX     ENDM

**        .CHK - CHECK THAT N X-REGISTERS ARE FULL.

 .CHK     MACRO  N
 .LSX     MICCNT .SX
          IFLT   .LSX,N,2
          DUP    N-.LSX,1
          .LBX
 .CHK     ENDM

**        .CHK1 - PREPARE FOR UNARY OPERATION.

 .CHK1    MACRO
          .CHK   1
          .GTX   .S1
 .SS      MICMIC .S1
 .RSXA    MICMIC .RSX'.S1'
          IFC    NE, '.RSXA'  ,2
          .RTX   .S1
          .GFX   .SS
 .CHK1    ENDM

**        .CHK2 - PREPARE FOR BINARY OPERATION.

 .CHK2    MACRO
          .CHK   2
          .RTX   .S1
          .GTX   .S2
 .RSXA    MICMIC .RSX'.S1'
 .RSXB    MICMIC .RSX'.S2'
 CHK2     IFC    EQ, '.RSXB'
          IFC    EQ, '.RSXA'  ,1
          .APX   .S1
 .SS      MICMIC .S2
 CHK2     ELSE
          .RTX   .S2
          IFC    EQ, '.RSXA'  ,3
          .ATX   .S1
 .SS      MICMIC .S1
          DUP    0,1
          .GFX   .SS
 CHK2     ENDIF
 .CHK2    ENDM

**        .GFX - GET FREE X-REGISTER FROM POOL.

 .GFX     MACRO  R
 .LFX     MICCNT .FX
          IFEQ   .LFX,0,1
          .SBX
          .RPX   R
          .ATX   R
 .GFX     ENDM

**        .NORMLS - SAVE THE STACK X-REGISTERS IN CORE.

 .NORMLS  MACRO
 .LSX     MICCNT .SX
          DUP    .LSX,1
          .SBX
 .NORMLS  ENDM

**        .RPB - REMOVE B-REGISTER FROM POOL.

 .RPB     MACRO  R
 .LFB     MICCNT .FB
          ERRZR  .LFB
 R        MICRO  1,1,*'.FB'*
 .FB      MICRO  2,,*'.FB'*
 .RPB     ENDM

**        .APB - ADD B-REGISTER TO POOL.

 .APB     MACRO  R
 .FB      MICRO  1,,*'.FB''R'*
 .APB     ENDM

**        .STST - START STATEMENT.

 .STST    MACRO
 .SLVL    SET    .SLVL+1
          IFEQ   .SLVL,1,4
 .SX      MICRO  1,,**
 .FX      MICMIC .PX
 .FB      MICMIC .PB
 .FM      SET    .SFM
 .STST    ENDM

**        .STND - END STATEMENT.

 .STND    MACRO
 .SLVL    SET    .SLVL-1
 .STND    ENDM

**        GXI    XJ,XK            MAX FUNCTION OPDEF.

 GXX,X    OPDEF  I,J,K
          IX6    X.J-X.K
          AX6    59
          BX7    X6*X.K
          BX6    -X6*X.J
          BX.I   X7+X6
          ENDM

**        TXI    XJ,XK            MIN FUNCTION OPDEF.

 TXX,X    OPDEF  I,J,K
          IX6    X.K-X.J
          AX6    59
          BX7    X6*X.K
          BX6    -X6*X.J
          BX.I   X7+X6
          ENDM

**        QXI    XJ/XK            DIVISION OPDEF.

 QXX/X    OPDEF  I,J,K
          PX6    X.J
          PX7    X.K
          NX6    X6
          NX7    X7
          FX.I   X6/X7
          UX.I   X.I,B3
          LX.I   X.I,B3
          ENDM

**        .DMOSF - DEFINE MULTIPLE OPERAND SYMMETRIC FUNCTIONS.

          MACRO  .DMOSF,FNC,CPOP,OPSYM
 FNC      MACRO  Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8,Y9
          .STST
          LOAD   Y1
          FNC!L  (Y2),(Y3),(Y4),(Y5),(Y6),(Y7),(Y8),(Y9)
          .STND
 FNC      ENDM

 FNC!L    MACRO  Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8
          IFC    NE,$Y1$$,4
          LOAD   Y1
          .CHK2
          CPOP!X'.SS'    X'.S1'OPSYM!X'.S2'
          FNC!L  (Y2),(Y3),(Y4),(Y5),(Y6),(Y7),(Y8)
 FNC!L    ENDM
 .DMOSF   ENDM

**        .DBF - DEFINE SOME OTHER BINARY FUNCTIONS.

          MACRO  .DBF,FNC,CPOP,OPSYM1,OPSYM2
 FNC      MACRO  Y1,Y2
          .STST
          LOAD   Y1
          LOAD   Y2
          .CHK2
          CPOP!X'.SS'    OPSYM1!X'.S2'OPSYM2!X'.S1'
          .STND
 FNC      ENDM
 .DBF     ENDM

**        .DRF - DEFINE BINARY FUNCTIONS WITH ARGUMENTS REVERSED.

          MACRO  .DRF,FNC,CPOP,OPSYM1,OPSYM2
 FNC      MACRO  Y1,Y2
          .STST
          LOAD   Y1
          LOAD   Y2
          .CHK2
          CPOP!X'.SS'    OPSYM1!X'.S1'OPSYM2!X'.S2'
          .STND
 FNC      ENDM
 .DRF     ENDM

**        .CEXTP - EXTRACT PARAMETERS FOR COND.

 .CEXTP   MACRO  P1,P2,P3
 PRED     MICRO  1,,$P1$
 OP1      MICRO  1,,$(P2)$
```

```
 OP2      MICRO  1,,$(P3)$
 .CEXTP   ENDM

**        .DUR - DEFINE UNARY PREDICATES FOR COND.

          MACRO  .DUR,TUN,OP
 TUN      MACRO  OP1,OP2,TRUE
          LOAD   OP1
          .CHK   1
          .GTX   .S1
          OP     X'.S1',TRUE
 TUN      ENDM
 .DUR     ENDM

 .TZR     .DUR   ZR
 .TNZ     .DUR   NZ
 .TPL     .DUR   PL
 .TNG     .DUR   NG
 .TFALSE  .DUR   PL
 .TTRUE   .DUR   NG

**        .DBR - DEFINE BINARY PREDICATES FOR COND.

          MACRO  .DBR,TBN,UN
 TBN      MACRO  OP1,OP2,TRUE
          .T!UN    (MINUS,(OP1),(OP2)),,TRUE
 TBN      ENDM
 .DBR     ENDM

 .TEQ     .DBR   ZR
 .TNE     .DBR   NZ
 .TGE     .DBR   PL
 .TLT     .DBR   NG

 .TLE     MACRO  OP1,OP2,TRUE
          .TGE   (OP2),(OP1),TRUE
 .TLE     ENDM

 .TGT     MACRO  OP1,OP2,TRUE
          .TLT   (OP2),(OP1),TRUE
 .TGT     ENDM

**        .CHKS1 - PREPARE FOR UNARY SQUARE LIST OPERATION.

 .CHKS1   MACRO
          .CHK   2
          .RTX   .S1
          .GTX   .S2
          .ATX   .S1
 .CHKS1   ENDM

**        .CHKS2 - PREPARE FOR BINARY SQUARE LIST OPERATION.

 .CHKS2   MACRO
          .CHK   4
          .RTX   .S1
          .APX   .S1
          .RTX   .S2
          .APX   .S2
          .RTX   .S3
          .GTX   .S4
          .ATX   .S3
 .CHKS2   ENDM

**        .TNULLS - TEST FOR EMPTY SQUARE LIST.

 .TNULLS  MACRO  OP1,OP2,TRUE
          LOADS  OP1
          .CHKS1
          BX'.S2'    X'.S2'+X'.S1'
          ZR       X'.S2',TRUE
 .TNULLS  ENDM

**        .TNNULS - TEST FOR NON-EMPTY SQUARE LIST.

 .TNNULS  MACRO  OP1,OP2,TRUE
          LOADS  OP1
          .CHKS1
          BX'.S2'    X'.S2'+X'.S1'
          NZ       X'.S2',TRUE
 .TNNULS  ENDM

**        .DMTST - DEFINE SQUARE LIST MEMBERSHIP TESTING PREDICATES.

          MACRO  .DMTST,NAME,OP
 .T!NAME  MACRO  LST,SQ,TRUE
          LOAD   LST
          LOAD   INDEX,SQ2BT,(SQ)
          .CHK1
          .RTX   .ST
          UX'.ST'    X'.S1',B3
          .CHK1
          .RTX   .SS
          .RPB   .BC
          SB'.BC'    X'.S1'
          SA'.ST'    X'.ST'+B'.BC'
          SX'.SS'    B1
          LX'.SS'    X'.SS',B3
          BX'.ST'    X'.ST'*X'.SS'
          OP       X'.ST',TRUE
          .APB   .BC
          .APX   .SS
          .APX   .ST
 .T!NAME  ENDM
 .DMTST   ENDM

 MEMBS    .DMTST NZ               TEST FOR MEMBERSHIP
 NONMS    .DMTST ZR               TEST FOR NON-MEMBERSHIP

**        .DMSFS - DEFINE MULTIPLE OPERAND SYMMETRIC SQUARE LIST FNCS.

          MACRO  .DMSFS,FNC,CPOP,OPSYM
 FNC      MACRO  Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8,Y9
          .STST
          LOADS  Y1
          FNC!L    (Y2),(Y3),(Y4),(Y5),(Y6),(Y7),(Y8),(Y9)
          .STND
 FNC      ENDM

 FNC!L    MACRO  Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8
          IFC    NE,$Y1$$,5
          LOADS  Y1
          .CHKS2
          CPOP!X'.S4'    X'.S4'OPSYM!X'.S2'
          CPOP!X'.S3'    X'.S3'OPSYM!X'.S1'
          FNC!L    (Y2),(Y3),(Y4),(Y5),(Y6),(Y7),(Y8)
 FNC!L    ENDM
 .DMSFS   ENDM

**        .DBFS - DEFINE SOME OTHER BINARY SQUARE LIST FUNCTIONS.

          MACRO  .DBFS,FNC,CPOP,OPSYM1,OPSYM2
 FNC      MACRO  Y1,Y2
          .STST
          LOADS  Y1
          LOADS  Y2
          .CHKS2
          CPOP!X'.S4'    OPSYM1!X'.S4'OPSYM2!X'.S2'
          CPOP!X'.S3'    OPSYM1!X'.S3'OPSYM2!X'.S1'
          .STND
 FNC      ENDM
 .DBFS    ENDM

**        .DRFS - DEFINE SQUARE LIST FUNCTIONS WITH ARGUMENTS REVERSED.

          MACRO  .DRFS,FNC,CPOP,OPSYM1,OPSYM2
 FNC      MACRO  Y1,Y2
          .STST
          LOADS  Y1
          LOADS  Y2
          .CHKS2
          CPOP!X'.S4'    OPSYM1!X'.S2'OPSYM2!X'.S4'
          CPOP!X'.S3'    OPSYM1!X'.S1'OPSYM2!X'.S3'
          .STND
 FNC      ENDM
 .DRFS    ENDM

**        MICMIC - DEFINE A MICRO FROM A CONSTRUCTED MICRO NAME.

          MACRO  MICMIC,NAME,EXP
 NAME     MICRO  1,, 'EXP'
 MICMIC   ENDM

**        REMC - REMOVE A CHARACTER (MICRO) FROM A MICRO.

 REMC     MACRO  KAR,MIC
 .MC      MICCNT MIC
 .IC      SET    0
 REM      DUP    .MC
 .IC      SET    .IC+1
 .KR      MICRO  .IC,1, 'MIC'
 REM      IFC    EQ, '.KR' 'KAR'
          STOPDUP
 .BS      MICRO  1,.IC-1, 'MIC'
 .ES      MICRO  .IC+1,, 'MIC'
          IFEQ   .IC,1,1
 .BS      MICRO  1,,**
 MIC      MICRO  1,, '.BS''.ES'
 FIND     SKIP
 REM      ENDIF
 REM      ENDD
          ERR    COUDNT FIND CHARACTER
 FIND     ENDIF
 REMC     ENDM

**        INIT - INITIALIZE MACRO SYSTEM.

 INIT     MACRO

          SST
          NOREF  .FM,.FMO,.IC,.LFB,.LFX,.LSX,.MFM,.R,.SFM,.SLVL,.S1
          NOREF  .MC,.RST,.RSX
          ECHO   1,R=(1,2,3,4,5)
 .RSX!R   MICRO  1,,**
          ECHO   1,R=(2,4,5,6,7)
 .RSB!R   MICRO  1,,**
 .PX      MICRO  1,,*12345*
 .PB      MICRO  1,,*24567*
 .SAVER   SET    0
 .SUB     MICRO  1,,**
 .RSX     MICRO  1,,**
 .RSB     MICRO  1,,**
 .MFM     EQU    64
 .SLVL    SET    0
 .FM      SET    0
 .LFM     SET    0
 .SFM     SET    0
 .PDB     MICRO  1,,*ESTAK*
 FORWARD  SET    LE.MOV
 REVERSE  SET    -LE.MOV
          ERRNZ  59-S.ILL
          ERRNZ  BETAR-ALPHA-1
          B1=1

 INIT     ENDM

***       VARIOUS GENERAL PURPOSE MACROS.

**        LOAD - LOAD VARIABLE OR EXPRESSION INTO REGISTER STACK.

 LOAD     MACRO  Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8,Y9
 LOAD     IFC    EQ,$Y2$$
 .RSX     MICCNT .RSX
 .R       SET    0
 LOAD     DUP    .RSX
 .R       SET    .R+1
 .S1      MICRO  .R,1, '.RSX'
 .RSXA    MICMIC .RSX'.S1'
          IFC    EQ, '.RSXA' Y1 ,3
 LOAD     STOPDUP
          .ATX   .S1
 FIND     SKIP
 LOAD     ENDD
          .GFX   .S1
          IF     ABS,Y1,2
          R=  X'.S1',Y1
          DUP    0,1
          SA'.S1'    Y1
 LOAD     ELSE
          Y1     (Y2),(Y3),(Y4),(Y5),(Y6),(Y7),(Y8),(Y9)
 LOAD     ENDIF
 FIND     ENDIF
 LOAD     ENDM

**        STORE - STORE TOP OF STACK INTO VARIABLE.

 STORE    MACRO  Y
          .CHK   1
          .GTX   .S1
          BX7    X'.S1'
          SA7    Y
 STORE    ENDM

**        STORX - STORE TOP OF STACK IN A RESERVED X-REGISTER.
```

```
 STORX    MACRO  Y
 .RSX     MICCNT .RSX
 .R       SET    0
 STORX    DUP    .RSX
 .R       SET    .R+1
 .SS      MICRO  .R,1, '.RSX'
 .RSXA    MICMIC .RSX'.SS'
          IFC    EQ, '.RSXA' Y ,4
          STOPDUP
          .RTX   .S1
          .ATX   .SS
 FIND     SKIP
 STORX    ENDD
          .CHK1
          RESERV X'.SS'
.RSX'.SS' MICRO 1,, Y
 FIND     ENDIF
          IFC    NE,*'.SS'*'.S1'*,1
          BX'.SS'    X'.S1'
 STORX    ENDM

**        RESERV - INTER-STATEMENT RESERVE REGISTERS (X OR B).

 RESERV   MACRO  REGS
          LOCAL  T,N
 RES      ECHO   ,REG=(REGS)
 T        MICRO  1,1, REG
 N        MICRO  2,1, REG
          REMC   N,.P'T'
 .RS!REG  MICRO  1,, REG
 .RST     MICMIC .RS'T'
 .RS'T'   MICRO  1,, '.RST''N'
 RES      ENDD
 RESERV   ENDM

**        RELEAS - RELEASE RESERVED REGISTERS (X OR B).

 RELEAS   MACRO  REGS
          LOCAL  T,N,Q,R
 REL      ECHO   ,RG=(REGS)
 R        MICRO  1,, RG
 REL      IF     -REG,RG
 .RSX     MICCNT .RSX
 .S1      SET    0
 REL1     DUP    .RSX
 .S1      SET    .S1+1
 .S1      MICRO  .S1,1, '.RSX'
 .RSXA    MICMIC .RSX'.S1'
          IFC    EQ, RG '.RSXA' ,3
 R        MICRO  1,, X'.S1'
 REL1     STOPDUP
 REL      SKIP
 REL1     ENDD
 GONE     SKIP
 REL      ENDIF
 T        MICRO  1,1, 'R'
 N        MICRO  2,1, 'R'
 Q        MICMIC .P'T'
 .RSR     MICMIC .RS'R'
 REL      IFC    NE, '.RSR'
 .P'T'    MICRO  1,,*'Q''N'*
          REMC   N,.RS'T'
 .RS'R'   MICRO  1,,**
 REL      ENDIF
 GONE     ENDIF
 REL      ENDD
 RELEAS   ENDM

**        SAVE - SAVE STACK AND RESERVED REGISTERS IN CORE.

 SAVER    OPSYN  SAVE
          PURGMAC SAVE
 SAVE     MACRO
 SAVE     IFEQ   .SAVER,0
          .STST
          .NORMLS
 .FMO     SET    .FM
 SAVT     ECHO   ,T=(X,B),C=(B,S)
 .RST     MICCNT .RS!T
 .RS      SET    0
 SAVE     DUP    .RST
 .RS      SET    .RS+1
 .RS      MICRO  .RS,1, '.RS!T'
          C!X6    T'.RS'
          IFEQ   .FM,.FMO,2
          SA6    .FM+'.PDB''.SUB'
          DUP    0,1
          SA6    A6+B1
 .FM      SET    .FM+1
 SAVE     ENDD
 SAVT     ENDD
 .SAVER   SET    1
 .SFM     SET    .SFM+.FM-.FMO
          IFC    EQ, '.SUB'  ,2
          SA0    .FM
          DUP    0,2
          IFNE   .FM,0,1
          SA0    .FM+A0
          .STND
 SAVE     ENDIF
 SAVE     ENDM

**        RESTOR - RESTORE SAVED RESERVED REGISTERS.

 RESTOR   MACRO
 RESTOR   IFNE   .SAVER,0
          .STST
 .X1S     MICRO  1,,**
          IFNE   .FM,0,2
          IFC    NE, '.SUB'  ,1
          SA0    A0-.FM
 .FMO     SET    .FM
 REST     ECHO   ,T=(B,X),C=(S,B)
 .RST     MICCNT .RS!T
 .RS      SET    .RST+1
 RESU     DUP    .RST
 .RS      SET    .RS-1
 .RS      MICRO  .RS,1, '.RS!T'
 .FM      SET    .FM-1
          IFEQ   .FM,.FMO-1,2
          SA1    .FM+'.PDB''.SUB'
          DUP    0,1
          SA1    A1-B1
          IFC    EQ, T'.RS' X1 ,3
          BX0    X1
 .X1S     MICRO  1,, S
          DUP    0,1
          C!T'.RS'    X1
 RESU     ENDD
          IFC    EQ, T'.X1S' XS ,1
          BX1    X0
 REST     ENDD
 .SAVER   SET    0
 .SFM     SET    .SFM-.FMO+.FM
          .STND
 RESTOR   ENDIF
 RESTOR   ENDM

**        DEFINE SOME MULTIPLE OPERAND SYMMETRIC FUNCTIONS.

          PURGMAC AND
 AND      .DMOSF B,*               BOOLEAN AND
 ORF      .DMOSF B,+               BOOLEAN OR
 XOR      .DMOSF B,-               BOOLEAN EXCLUSIVE OR
 TIMES    .DMOSF I,*               INTEGER MULTIPLY
 PLUS     .DMOSF I,+               INTEGER ADD
 MAXF     .DMOSF G,(,)
 MINF     .DMOSF T,(,)

**        DEFINE SOME BINARY FUNCTIONS.

 MINUS    .DBF   I,,-
 COR      .DBF   B,-,+
 CAND     .DBF   B,-,*
 CXOR     .DBF   B,-,-
 DIVIDE   .DBF   Q,,/
 ORC      .DRF   B,-,+
 ANDC     .DRF   B,-,*
 XORC     .DRF   B,-,-

**        LODREL - LOAD AND RELEASE AN X-REGISTER.

 LODREL   MACRO  Y
          .STST
          LOAD   Y
          RELEAS Y
          .STND
 LODREL   ENDM

**        SETQ - STORE A VALUE IN A NAMED VARIABLE.

 SETQ     MACRO  Y1,Y2
          .STST
          LOAD   Y2
          STORE  Y1
          .STND
 SETQ     ENDM

**        SETA - STORE A VALUE AT A COMPUTED ADDRESS.

 SETA     MACRO  Y1,Y2
          .STST
          LOAD   Y2
          LOAD   Y1
          .CHK   1
          .RTX   .SA
          STORE  X'.SA'
          .APX   .SA
          .STND
 SETA     ENDM

**        SETX - STORE A VALUE IN AN X-REGISTER AND RESERVE IT.

 SETX     MACRO  Y1,Y2
          .STST
          IFC    EQ, Y2  ,2
          .GFX   .S1
          DUP    0,1
          LOAD   Y2
          STORX  Y1
          .STND
 SETX     ENDM

**        INDEX - GET SUBSCRIPTED VECTOR ELEMENT.

 INDEX    MACRO  Y1,Y2
          .STST
          LOAD   Y2
          .CHK1
          SA'.SS'    Y1+X'.S1'
          .STND
 INDEX    ENDM

**        LSHIFT - LEFT SHIFT.

 LSHIFT   MACRO  Y1,Y2
          .STST
          LOAD   Y1
          IF     ABS,Y2,14
          .CHK   1
          .GTX   .S1
 .RSXA    MICMIC .RSX'.S1'
          IFC    EQ, '.RSXA'  ,6
          IFGT   Y2,0,1
          LX'.S1'    Y2
          IFLT   Y2,0,2
 .AY2     SET    Y2
          AX'.S1'    -.AY2
          DUP    0,3
          .CHK1
          SB3    Y2
          LX'.SS'    X'.S1',B3
          DUP    0,4
          LOAD   Y2
          .CHK2
          SB3    X'.S1'
          LX'.SS'    X'.S2',B3
          .STND
 LSHIFT   ENDM

**        NOT -  BOOLEAN COMPLEMENT.

 NOT      MACRO  Y
          .STST
          LOAD   Y
          .CHK1
          BX'.SS'    -X'.S1'
          .STND
 NOT      ENDM

**        ABSF - ABSOLUTE VALUE.

 ABSF     MACRO  Y
```

```
          .STST
          LOAD   Y
          .CHK1
          BX7    X'.S1'
          AX7    59
          BX'.SS'    X7-X'.S1'
          .STND
 ABSF     ENDM

**        MASK - CREATE OPTIONALLY SHIFTED BIT MASK.

 MASK     MACRO  Y1,Y2
          IFC    NE,$Y2$$,2
          LSHIFT (MASK,(Y1)),(Y2)
          DUP    0,4
          .STST
          .GFX   .S1
          MX'.S1'    Y1
          .STND
 MASK     ENDM

**        SETBIT - SET A GIVEN BIT NUMBER IN A WORD.

 SETBIT   MACRO  WD,BIT
          SETQ   WD,(ORF,WD,(LSHIFT,1,(BIT)))
 SETBIT   ENDM

**        CLRBIT - CLEAR A GIVEN BIT NUMBER IN A WORD.

 CLRBIT   MACRO  WD,BIT
          SETQ   WD,(ANDC,WD,(LSHIFT,1,(BIT)))
 CLRBIT   ENDM

**        TEST - SHIFT A BIT INTO SIGN POSITION FOR TESTING.

 TEST     MACRO  Y1,Y2
          .STST
          LOAD   Y1
          IF     ABS,Y2,12
 .TST     SET    Y2
          .CHK   1
          .GTX   .S1
 .RSXA    MICMIC .RSX'.S1'
          IFC    EQ, '.RSXA'  ,3
          IFNE   .TST,59,1
          LX'.S1'    59-.TST
          DUP    0,3
          .CHK1
          SB3    59-.TST
          LX'.SS'    X'.S1',B3
          DUP    0,4
          LOAD   Y2
          .CHK2
          SB3    X'.S1'-59
          AX'.SS'    X'.S2',B3
          .STND
 TEST     ENDM

**        LOCF - ADDRESS OF VARIABLE.

 LOCF     MACRO  Y
          .STST
          .GFX   .S1
          SX'.S1'    Y
          .STND
 LOCF     ENDM

**        IDXLOC - ADDRESS OF (INDEX,Y1,Y2).

 IDXLOC   MACRO  Y1,Y2
          .STST
          LOAD   Y2
          .CHK1
          SX'.SS'    Y1+X'.S1'
          .STND
 IDXLOC   ENDM

***       BRANCHING MACROS.

**        COND - BRANCH ON CONDITION TRUE.

 COND     MACRO  P,TRUE
          .STST
          .CEXTP P
          .T'PRED'    'OP1','OP2',TRUE
          .STND
 COND     ENDM

**        GOTO - UNCONDITIONAL BRANCH OR COMPUTED BRANCH.

 GOTO     MACRO  S,Y
          LOCAL  L
 GO       IFC    EQ,$Y$$
          EQ     S
 GO       ELSE
          .STST
          LOAD   Y
          .CHK   1
          .GTX   .S1
          SB3    X'.S1'
          JP     L+B3
 L BSS    0
          ECHO   1,D=(S)
          EQ     D
          .STND
 GO       ENDIF
 GOTO     ENDM

**        CALL - CALL SUBROUTINE.

 CALL     MACRO  ROUTINE
          SAVE
          RJ     =X!ROUTINE
          RESTOR
 CALL     ENDM

**        CALLE - CALL EXTERNAL SUBROUTINE.

 CALLE    MACRO  ROUTINE,TRACE
 .TRC     MICRO  1,5,$TRACE$
          SAVE
          IFC    NE, '.SUB'  ,2
          SX6    A0
          SA6    SAVA0
+         RJ     =X!ROUTINE
-         VFD    30/5L'.TRC'
          IFC    NE, '.SUB'  ,2
          SA1    SAVA0
          SA0    X1
          RESTOR
 CALLE    ENDM


***       MACROS FOR SQUARE LISTS.

**        LOADS - LOAD SQUARE LIST OPERAND.

 LOADS    MACRO  Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8,Y9
          IFC    EQ,$Y2$$,5
          .GFX   .S1
          SA'.S1'    Y1
          .GFX   .S2
          SA'.S2'    A'.S1'+B1
          DUP    0,1
          Y1     (Y2),(Y3),(Y4),(Y5),(Y6),(Y7),(Y8),(Y9)
 LOADS    ENDM

**        STORES - STORE SQUARE LIST.

 STORES   MACRO  Y
          .CHKS1
          BX6    X'.S2'
          BX7    X'.S1'
          SA6    Y
          SA7    A6+B1
 STORES   ENDM

**        DEFINE SOME MULTIPLE OPERAND SQUARE LIST SYMMETRIC FUNCTIONS.

 ANDS     .DMSFS B,*
 ORS      .DMSFS B,+
 XORS     .DMSFS B,-

**        DEFINE SOME BINARY FUNCTIONS.

 CANDS    .DBFS  B,-,*
 CORS     .DBFS  B,-,+
 CXORS    .DBFS  B,-,-
 ANDCS    .DRFS  B,-,*
 ORCS     .DRFS  B,-,+
 XORCS    .DRFS  B,-,-

**        SETQS - STORE NEW VALUE IN A SQUARE LIST.

 SETQS    MACRO  Y1,Y2
          .STST
          LOADS  Y2
          STORES Y1
          .STND
 SETQS    ENDM

**        SETAS - STORE A SQRLST AT A COMPUTED ADDRESS.

 SETAS    MACRO  Y1,Y2
          .STST
          LOADS  Y2
          LOAD   Y1
          .CHK   1
          .RTX   .SA
          STORES X'.SA'
          .APX   .SA
          .STND
 SETAS    ENDM

**        INDEXS - GET SUBSCRIPTED 2-WORD (SQRLST) VECTOR ELEMENT.

 INDEXS   MACRO  Y1,Y2
          .STST
          LOAD   Y2
          .CHK1
          .RTX   .SS
          .APX   .SS
          LOADS  Y1+X'.S1'
          .STND
 INDEXS   ENDM

**        ADDSQS - ADD A GIVEN (8X8) SQUARE NUMBER TO A SQUARE LIST.

 ADDSQS   MACRO  LST,SQ
          .STST
          LOAD   LST
          LOAD   INDEX,SQ2BT,(SQ)
          .CHK1
          .RTX   .ST
          UX'.ST'    X'.S1',B3
          .CHK1
          .RTX   .SS
          .RPB   .BC
          SB'.BC'    X'.S1'
          SA'.ST'    X'.ST'+B'.BC'
          SX'.SS'    B1
          LX'.SS'    X'.SS',B3
          BX6    X'.ST'+X'.SS'
          SA6    A'.ST'
          .APB   .BC
          .APX   .SS
          .APX   .ST
          .STND
 ADDSQS   ENDM

**        RAYSQS - FORM RAY BETWEEN 2 SQUARES EXCLUDING ENDPOINTS.

 RAYSQS   MACRO  SQ1,SQ2
          LOCAL RAYSQSEX,RAYSQSLP,RAYSQSL1
          LOCAL  RAYSQSEN
          .STST
          .RPB   .BQ1
          .RPB   .BQ2
          LOAD   SQ1
          .CHK1
          SA'.SS'    EXPND+X'.S1'          CONVERT IST SQ TO 10X12
          SB'.BQ1'    X'.SS'
          LOAD   SQ2
          .CHK1
          SA'.SS'    EXPND+X'.S1'          CONVERT 2ND SQ TO 10X12
          SB'.BQ2'    X'.SS'
          SX6    B'.BQ2'-B'.BQ1'             10X12 INTERVAL
          SA'.SS'    DIRTB+X6          GET DIRECTION OF RAY
          SB3    X'.SS'
          .GFX   .SS
          .CHK   3
          .RTX   .SS
          .CHKS1
          BX'.S2'    X0-X0             INITIALIZE RESULT LIST WD 1
```

```
          BX'.S1'    X0-X0             INITIALIZE RESULT LIST WD 2
          ZR     B3,RAYSQSEN       IF NO DIRECT RAY
          MX7    1
          .RPB   .BSC
 RAYSQSLP SB'.BQ1'    B'.BQ1'+B3             STEP ONE SQ ALONG RAY
          EQ     B'.BQ1',B'.BQ2',RAYSQSEX  IF DONE
          SX6    B'.BQ1'-60
          PL     X6,RAYSQSL1       IF SQ IN 2ND WORD
          SB'.BSC'    X6
          AX6    X7,B'.BSC'              SHIFT BIT INTO POSITION
          BX'.S2'    X'.S2'+X6             SET IN LIST WORD 1
          EQ     RAYSQSLP          LOOP FOR NEXT SQUARE

 RAYSQSL1 SB'.BSC'    X6-60
          AX6    X7,B'.BSC'
          BX'.S1'    X'.S1'+X6             SET BIT IN WORD 2
          EQ     RAYSQSLP          LOOP FOR NEXT SQUARE

 RAYSQSEX BSS    0
          SA'.SS'    ONBRD
          BX6    -X'.SS'*X'.S2'
          LX'.SS'    20
          BX7    -X'.SS'*X'.S1'
          BX6    X6+X7
          ZR     X6,RAYSQSEN       IF NOT PHONY RAY
          BX'.S1'    X0-X0
          BX'.S2'    X0-X0
 RAYSQSEN BSS    0
          .APX   .SS
          .APB   .BQ1
          .APB   .BQ2
          .APB   .BSC
          .STND
 RAYSQS   ENDM

**        COUNTS - COUNT MEMBERS OF A SQUARE LIST.

 COUNTS   MACRO  Y
          .STST
          LOADS  Y
          .CHK2
          CX'.S1'    X'.S1'
          CX'.S2'    X'.S2'
          IX'.SS'    X'.S2'+X'.S1'
          .STND
 COUNTS   ENDM

**        SQLOOP - BEGIN LOOP THRU SQUARE LIST.

 SQLOOP   MACRO  SQLA
          LOCAL  SQLL1,SQLL2,SQLL3
 .SQL1    MICRO  1,, SQLL1
 .SQL2    MICRO  1,, SQLL2
 .SQL3    MICRO  1,, SQLL3
          SETX   SQLN
          .STST
          LOAD   SQLA
          .CHK1
 .SQLXS   MICMIC .SS
          .RPB   .SQLWO
          .RPB   .SQLBS
          .RPB   .SQLAS
          SB'.SQLAS'    X'.S1'
          SA'.SQLXS'    B'.SQLAS'
          RESERV (X'.SQLXS',B'.SQLWO',B'.SQLBS',B'.SQLAS')
          SB'.SQLWO'    B0              INIT SQUARE LIST WORD NUMBER
          ZR     X'.SQLXS','.SQL3'              IF FIRST WORD EMPTY
          SB'.SQLBS'    12+BT2SQ          INIT BT2SQ CONVERT INDEX
 '.SQL1' MX0    1
          LX0    48
          NX6    X'.SQLXS',B3
          AX6    X0,B3
          BX'.SQLXS'    X6-X'.SQLXS'
          LOAD   SQLN
          .RTX   .SQN
          SA'.SQN'    B3+B'.SQLBS'             SQUARE NUMBER
          .STND
 SQLOOP   ENDM

**        SQLEND - END LOOP THRU SQUARE LIST.

 SQLEND   MACRO
          LOCAL  SQLL4
 '.SQL2' NZ     X'.SQLXS','.SQL1'       IF SOMETHING LEFT IN WORD
          EQ     B'.SQLWO',B1,SQLL4    IF SECOND WORD FINISHED
 '.SQL3' SA'.SQLXS'    B'.SQLAS'+B1
          SB'.SQLWO'    B1
          SB'.SQLBS'    60+BT2SQ
          LX'.SQLXS'    48
          NZ     X'.SQLXS','.SQL1'       IF 2ND WORD NON-EMPTY
 SQLL4 BSS    0
          RELEAS (SQLN,X'.SQLXS',B'.SQLWO',B'.SQLBS',B'.SQLAS')
 SQLEND   ENDM

***       MACROS FOR CHESS MOVE GENERATION AND SEARCHING.

**        GENMOV - GENERATE MOVES OF A GIVEN TYPE AND FROM A
*               GIVEN OPTIONAL SQRLST.

 GENMOV   MACRO  TYPE,SQRLST
          IFC    NE, SQRLST  ,8
          .STST
          LOAD   SQRLST
          .CHK   1
          .RTX   .S1
          .STND
          SAVE
          SA1    X'.S1'                LOAD SQUARE LIST
          SA2    A1+B1
 TYPE     MICRO  1,3, TYPE
          CALLE  GEN'TYPE'         CALL MOVE GENERATOR
 GENMOV   ENDM

**        SELOOP - BEGIN SELECT-MOVE-FOR-SEARCH LOOP.

 SELOOP   MACROE DIRECT,START,FIN,SELECT
          LOCAL  SELL1,SELL2,SELL3
          .STST
 .SEL1    MICRO  1,, SELL1
 .SEL2    MICRO  1,, SELL2
 .SEL3    MICRO  1,, SELL3
 .SEL4    MICRO  1,, SELECT
 .SDIR    SET    DIRECT FORWARD
          IFEQ   .SDIR,FORWARD,2
          LOAD   START LOCF,MOVES
          DUP    0,1
          LOAD   START MINUS,LINDX,LE.MOV
          .RPB   .SELST
          RESERV B'.SELST'
          .CHK   1
          .RTX   .S1
          SB'.SELST'    X'.S1'
          .APX   .S1
          IFEQ   .SDIR,FORWARD,2
          LOAD   FIN MINUS,LINDX,LE.MOV
          DUP    0,1
          LOAD   FIN LOCF,MOVES
          .RPB   .SELND
          RESERV B'.SELND'
          .CHK   1
          .RTX   .S1
          SB'.SELND'    X'.S1'
          .APX   .S1
          IFEQ   .SDIR,FORWARD,2
          GT     B'.SELST',B'.SELND','.SEL2'    IF NO MOVES
          DUP    0,1
          LT     B'.SELST',B'.SELND','.SEL2'    IF NO MOVES
 '.SEL1' BSS    0
          .STND
          SETX   SMOVE,(INDEX,B0,B'.SELST')
          COND   (NG,SMOVE),'.SEL3'
 SELOOP   ENDM

**        SELMOV - CURRENT MOVE PASSES SELECTION CRITERIA.

 SELMOV   MACRO
          SX6    B'.SELST'
          SA6    INDEX
          EQ     '.SEL4'
 SELMOV   ENDM

**        SELMAX - SELECT MOVE WITH MAXIMUM OF EXPRESSION.

 SELMAX   MACRO  EXP
          SETX   SVAL,(EXP)
          COND   (LE,SVAL,SMAX),'.SEL3'
          SETX   SMAX,SVAL
          RELEAS SVAL
          SETQ   INDEX,B'.SELST'
 SELMAX   ENDM

**        SELMIN - SELECT MOVE WITH MINIMUM OF EXPRESSION.

 SELMIN   MACRO  EXP
          SETX   SVAL,(EXP)
          COND   (GE,SVAL,SMIN),'.SEL3'
          SETX   SMIN,SVAL
          RELEAS SVAL
          SETQ   INDEX,B'.SELST'
 SELMIN   ENDM

**        SETMAX - SET MAXIMUM FOR SELMAX.

 SETMAX   MACRO  LOW
          IFC    NE, LOW  ,2
          SETX   SMAX,(LOW)
          DUP    0,1
          SETX   SMAX,-INFIN
 SETMAX   ENDM

**        SETMIN - SET MINIMUM FOR SELMIN.

 SETMIN   MACRO  HIGH
          IFC    NE, HIGH  ,2
          SETX   SMIN,(HIGH)
          DUP    0,1
          SETX   SMIN,INFIN
 SETMIN   ENDM

**        SELST - INDEX (+ OFFSET) OF CURRENT SELOOP MOVE.

 SELST    MACRO  Y
          .STST
          .GFX   .S1
          SX'.S1'    B'.SELST'+Y
          .STND
 SELST    ENDM

**        SELEND - END OF SELECT-MOVE-FOR-SEARCH LOOP.

 SELEND   MACRO
 '.SEL3' SB'.SELST'    B'.SELST'+.SDIR          POINT TO NEXT MOVE
          IFEQ   .SDIR,FORWARD,2
          LE     B'.SELST',B'.SELND','.SEL1'    LOOP IF MORE MOVES
          DUP    0,1
          GE     B'.SELST',B'.SELND','.SEL1'    LOOP IF MORE MOVES
          RELEAS (B'.SELST',B'.SELND',SMOVE)
 '.SEL2' BSS    0
 SELEND   ENDM

**        MODE - DEFINE SEARCH MODE.

          MACRO  MODE,NAME,ORD
 NAME     EQU    ORD
 S.NAME   EQU    S.MOD+L.MOD+ORD
 MODE     ENDM

**        SEARCH - SEARCH THE MOVE POINTED TO BY INDEX.

          MACROE SEARCH,TRACE,MODE,REFUTE,IGNORE,ILLEGAL,NON
 TRACE    BSS    0
 .MDM     SET    NON+MODE
          SETX   INDEX,INDEX
          SETA   INDEX,(ORF,(LSHIFT,1,S.SRC),(LSHIFT,.MDM,S.MOD),(LSHIFT
,,1,S.MODE),(CAND,(MASK,L.MOD,S.MOD+L.MOD),(INDEX,B0,INDEX)))
          RELEAS INDEX
          CALLE  TRSRCH,TRACE
          IFC    NE, ILLEGAL  ,1
          OR     X6,ILLEGAL
          IFC    NE, IGNORE  ,1
          PL     X6,IGNORE
          IFC    NE, REFUTE  ,1
          PL     X7,REFUTE
 SEARCH   ENDM

**        MINMAX - TEST AND FOLD IN SCORE (UNSEARCHED).

 MINMAX   MACROE NEWVAL,REFUTE,IGNORE,NON
          LOCAL  PPPPPPPP
          .STST
          IFC    NE, NEWVAL  ,4
          LOAD   NEWVAL
          .CHK1
          SX7    X'.S1'
```

```
          .APX   .SS
          .RPX   .SALF
          .RPX   .SBET
          .RPX   .SIDE
          SA'.SALF'    ALPHA             ALPHA
          SA'.SIDE'    OSIDE             SIDE TO MOVE
          SA'.SBET'    A'.SALF'+B1             BETAR
          IX'.SALF'    X'.SALF'-X7             ALPHA - SCORE
          IX'.SBET'    X7-X'.SBET'             SCORE - BETAR
          BX6    X'.SIDE'
          LX6    59
          AX6    59                +0 FOR WHITE, -0 FOR BLACK
          BX0    X'.SALF'-X'.SBET'             SWAP REGS IF BLACKS MOVE
          BX0    X6*X0
          BX'.SBET'    X0-X'.SBET'             PRUNING MARGIN
          BX'.SALF'    X0-X'.SALF'             MINIMAX DEFICIT
          IFC    EQ, NON  ,13
          IFC    EQ, IGNORE  ,2
          PL     X'.SALF',PPPPPPPP       IF NO NEW BEST SCORE
          DUP    0,1
          PL     X'.SALF',IGNORE
          SA7    ALPHA+X'.SIDE'          STORE NEW BEST SCORE
          SA'.SALF'    NPLYC             CURRENT PLY
          SX7    0                 NULL LIMB, ZERO LENGTH
          SA7    LIMBL+X'.SALF'
          IFC    NE, REFUTE  ,1
          PL     X'.SBET',REFUTE
          IFC    EQ, IGNORE  ,1
PPPPPPPP  BSS    0
          DUP    0,4
          IFC    NE, IGNORE  ,1
          PL     X'.SALF',IGNORE
          IFC    NE, REFUTE  ,1
          PL     X'.SBET',REFUTE
          .APX   .SALF
          .APX   .SBET
          .APX   .SIDE
          .STND
 MINMAX   ENDM

**        SETAB - SET ALPHA AND BETAR.

 SETAB    MACRO  A,B
          SETQ   ALPHA,(A)
          SETQ   BETAR,(B)
 SETAB    ENDM

***       MISCELLANEOUS MACROS FOR EVALU8.

**        LIMB - LOAD LIMB ARRAY ELEMENT.

 LIMB     MACRO  Y1,Y2
          .STST
          LOAD   INDEX,LIMBA,(Y1)
          IFC    EQ, Y2  ,3
          .CHK1
          BX0    X'.S1'
          DUP    0,3
          LOAD   Y2
          .CHK2
          IX0    X'.S1'+X'.S2'
          IFC    NE, '.SUB'  ,2
          .RPB   .BA0
          SB'.BA0'    A0
          SA0    STACK1
          SB3    B1
          RE     B3
          RJ     =XECRERR          IF ECS ERROR
          SA'.SS'    A0
          IFC    NE, '.SUB'  ,2
          SA0    B'.BA0'
          .APB   .BA0
          .STND
 LIMB     ENDM

**        HSQDST - ABSOLUTE HORIZONTAL DISTANCE BETWEEN 2 SQUARES.

 HSQDST   MACRO  Y1,Y2
          .STST
          LOAD   Y1
          LOAD   Y2
          .CHK2
          MX7    60-3
          BX6    -X7*X'.S1'
          BX7    -X7*X'.S2'
          IX6    X7-X6
          BX7    X6
          AX7    59
          BX'.SS'    X7-X6
          .STND
 HSQDST   ENDM

**        VSQDST - ABSOLUTE VERTICAL DISTANCE BETWEEN 2 SQUARES.

 VSQDST   MACRO  Y1,Y2
          .STST
          LOAD   Y1
          LOAD   Y2
          .CHK2
          BX6    X'.S1'
          BX7    X'.S2'
          AX6    3
          AX7    3
          IX6    X7-X6
          BX7    X6
          AX7    59
          BX'.SS'    X7-X6
          .STND
 VSQDST   ENDM

**        CENDST - DOUBLE RECTANGULAR DISTANCE FROM BOARD CENTER.

 CENDST   MACRO  Y
          .STST
          LOAD   Y                 SQUARE NUMBER
          .CHK1
          MX7    60-3
          BX7    -X7*X'.S1'             COLUMN
          LX7    1                 DOUBLE
          SX7    X7-7              DISTANCE FROM CENTER
          BX6    X7
          AX6    59
          BX7    X6-X7             ABSOLUTE VALUE
          BX0    X'.S1'
          AX0    3                 ROW
          LX0    1                 DOUBLE
          SX0    X0-7              DISTANCE FROM CENTER
          BX6    X0
          AX6    59
          BX0    X6-X0             ABSOLUTE VALUE
          IX'.SS'    X0+X7               SUM VERTICAL AND HORIZONTAL
          .STND
 CENDST   ENDM

**        ACCSID - ACCUMULATE SIDE-SIGN ADJUSTED FACTORS.

 ACCSID   MACRO  Y1,Y2,Y3
          SETQ   Y1,(PLUS,(XOR,(Y2),(Y3)),Y1)
 ACCSID   ENDM

**        PWNADV - FURTHEST PAWN IN STRUCTURE SIGNATURE COLUMN.

 PWNADV   MACRO  Y
          .STST
          LOAD   Y
          .CHK1
          SX7    X'.S1'-1
          BX7    X'.S1'-X7
          NX7    X7,B3
          SX'.SS'    B3-41
          .STND
 PWNADV   ENDM

**        SCORE - STORE FINAL SCORE AND RETURN FROM EVALU8.

 SCORE    MACRO  EVAL
          IFC    EQ, EVAL  ,2
          SCORE  (INDEX,ALPHA,OSIDE)
          DUP    0,2
          SETQ   SCORE,(EVAL)
          GOTO   EVALU8
 SCORE    ENDM

**        SETX3 - SETUP PARAMETER IN X3.

 SETX3    MACRO  Y
          .STST
          LOAD   Y
          .RTX   .S1
          IFC    NE, '.S1' 3 ,1
          BX3    X'.S1'
          .STND
 SETX3    ENDM

**        MBTAB - ASSEMBLE SPECIAL MATERIAL BALANCE TABLE ENTRY.

          MACRO  MBTAB,VAL,PIECES
 MBTAB    ECHO   ,S=(B,W)
          ECHO   1,T=(P,R,N,B,Q)
 .S!T     SET    0
 MBTAB    ENDD
          ECHO   1,M=(PIECES)
 .M       SET    .M+1
 .WVAL    SET    VAL
 .UVAL    SET    .WVAL/400B
 .LVAL    SET    .WVAL-.UVAL*400B
          VFD    8/.UVAL,2/0,4/.WQ,4/.WB,4/.WN,4/.WR,4/.WP!!!!!!!!!!!!!
,,!!!!!!!!!!!!!!8/.LVAL,2/0,4/.BQ,4/.BB,4/.BN,4/.BR,4/.BP
 MBTAB    ENDM

 OVERLAY  TITLE  OVERLAY STRUCTURE DEFINITIONS.
***       OVERLAY STRUCTURE DEFINITIONS.
*


 OVERLAY  EQU    1                 REMOVE FOR NON-OVERLAY FORM


**        OVER - DEFINE OVERLAY LEVEL.
*
*NN       OVER
*


          MACRO  OVER,NAME
 ENT      SET    0
 OVCR     MICRO  1,, NAME
 OV'OVCR' MICRO  1,,
 OVALL    MICRO  1,, 'OVALL',NAME
          ENDM
 OVALL    MICRO  1,,
          SPACE  4
**        ENTER - DEFINE OVERLAY ENTRY POINTS.
*
*ENTRY    ENTER
*


          MACRO  ENTER,NAME
 V.NAME   EQU    2R'OVCR'
 E.NAME   EQU    ENT
 ENT      SET    ENT+1
 OVOVCR   MICMIC OV'OVCR'
 OV'OVCR' MICRO  1,, 'OVOVCR',NAME
          ENDM
          SPACE  4
**        THIS - DECLARE THIS OVERLAY.
*
*NN       THIS
*


          MACRO  THIS,NAME

 X        MICRO  2,, 'OV!NAME'
 Y        MICRO  1,1, NAME
 Z        MICRO  2,1, NAME
          LCC    OVERLAY(CHESS!NAME,'Y','Z')
 END      MICRO  1,, CHE$$NAME

          IFC    NE,*'X'**
          ENTRY  CHE$$NAME
 CHE$$NAME  BSS    0
          ECHO   1,E=('X')
          EQ     =X!E
          ENDIF
          ENDM
          SPACE  4
**        JUMPUP - TRANSFER TO OVERLAY ENTRY POINT.
*
*         JUMPUP ENTRY
*
*         CALLS  JUMPUP.
*
```

```
 JUMPUP   MACRO  LABL
          LOCAL  XXXXXXXX
          SA5    XXXXXXXX
          RJ     =XJUMPUP

 XXXXXXXX VFD    42/V.LABL,18/E.LABL
          ENDM
          SPACE  4
**        OVERLAY STRUCTURE DEFINITION.
*



 10       OVER
 MYMOVE   ENTER
 LSTMOV   ENTER
 NOMOVE   ENTER
 MAKMOV   ENTER
 DBUGGR   ENTER
 MACMOV   ENTER
 NEWPOS   ENTER
 BKPCMD   ENTER
 SETEPT   ENTER
 PONDER   ENTER

          IF     DEF,DISPLAY,1
 11       OVER

          IF     DEF,IMLAC,1
 12       OVER

          IF     -DEF,LIBGEN,1
 20       OVER
 ENDCMD   ENTER
 SWICMD   ENTER
 PRICMD   ENTER
 CRECMD   ENTER
 PURCMD   ENTER
 BOACMD   ENTER
 LETCMD   ENTER
 PARCMD   ENTER
 KILCMD   ENTER
 YRMOVE   ENTER
 YUMOVE   ENTER
 YPMOVE   ENTER
 FINSHD   ENTER
 FINISH   ENTER
 INITAL   ENTER
 GONCMD   ENTER
 RECRDM   ENTER
 SAVCMD   ENTER
 SETCMD   ENTER
 PSLCMD   ENTER
 STACMD   ENTER
 PVACMD   ENTER
 TIMCTL   ENTER
 PINCMD   ENTER
 INPEOR   ENTER
 ECSCMD   ENTER
 NAMCMD   ENTER
 MSGCMD   ENTER
 COMCMD   ENTER
 CLOCMD   ENTER
          IF     DEF,NUCC,2
 MULCMD   ENTER
 PROSLV   ENTER
 BLICMD   ENTER

 30       OVER
 STRCMD   ENTER
 REWCMD   ENTER
 RETCMD   ENTER
 USECMD   ENTER
          IF     DEF,NUCC,1
 XEQCMD   ENTER
 RELCMD   ENTER
 REICMD   ENTER
 HDCCMD   ENTER
 ALTCMD   ENTER
 STDUSE   ENTER
 TOUCMD   ENTER
 PIDCMD   ENTER
 PPWCMD   ENTER
 AFNCMD   ENTER
 HELCMD   ENTER
 EXPCMD   ENTER
 SPECMD   ENTER
 INIMSI   ENTER
          IF     -DEF,KRONOS,1
          IF     DEF,NUCC,2
 DUMCMD   ENTER
 LOACMD   ENTER

 40       OVER
 SQRLST   TITLE  MOVE FORMAT.
**                FORMAT OF MOVE:
*
*                IMR STG FQ* ***
*                *** *** *** *SS
*                SSS SSS SSS SSS
*                SND DDD CAP EOK
*                FFF FFF TTT TTT
*
*                I = ILLEGAL MOVE.
*                M = NO LEGAL RESPONSES.
*                R = DRAW BY 2-TIME REPETITION.
*                S = SEARCHED IN SOME MODE.
*                T = PONDR MOVE NOW BEING SEARCHED
*                G = PONDR MOVE WAS ACTUALLY MADE.
*                F = PONDR MOVE SEARCHING FINISHED.
*                Q = MUST QUIT PONDR MOVE SEARCHING.
*                SS-S = SEARCHED IN EACH MODE.
*                N = NON-PERMANENT SEARCH MODE.
*                DDDD = SEARCH MODE.
*                C = CAPTURE.
*                A = AFFECTS CASTLE STATUS (ACS)
*                P = PROMOTION.
*                E = EN PASSANT OR CASTLE LONG.
*                O = CASTLE.
*                K = CHECK.
*
*                CAPEOK  SPECIAL CASES:
*                X0001X = UNUSED.
*                00010X = PAWN MOVE 2 SQUARES.
*                10010X = CAPTURE EN PASSANT.
*                X0011X = UNUSED.
*                X0100X = PROMOTE TO ROOK.
*                X0101X = PROMOTE TO KNIGHT.
*                X0110X = PROMOTE TO BISHOP.
*                X0111X = PROMOTE TO QUEEN.
*                X1000X = KING OR ROOK MOVE ACS.
*                01001X = O-O.
*                11001X = UNUSED.
*                X1010X = UNUSED.
*                01011X = O-O-O.
*                11011X = UNUSED.
*                011XXX = NULL MOVE.
*                11100X = P*R(R8)/R, ACS.
*                11101X = P*R(R8)/N, ACS.
*                11110X = P*R(R8)/B, ACS.
*                11111X = P*R(R8)/Q, ACS.
*
*                FFFFFF = FROM SQUARE.
*                TTTTTT = TO SQUARE.
*

 S.ILL    EQU    59              ILLEGAL
 S.MAT    EQU    58              MATE (CHECK- OR STALE-)
 S.REP    EQU    57              DRAW BY 2-TIME REPETITION
 S.SRC    EQU    56              SEARCHED IN SOME MODE
 S.THI    EQU    55              PONDR MOVE NOW BEING SEARCHED
 S.THG    EQU    54              PONDR MOVE WAS ACTUALLY MADE
 S.THF    EQU    53              PONDR MOVE SEARCHING FINISHED
 S.THQ    EQU    52              MUST QUIT PONDR MOVE SEARCHING
 S.MOD    EQU    18              SEARCH MODE
 S.CAP    EQU    17              CAPTURE
 S.ACS    EQU    16              AFFECTS CASTLE STATUS
 S.PRO    EQU    15              PROMOTION
 S.ENP    EQU    14              EN PASSANT CAPTURE
 S.CAS    EQU    13              CASTLE
 S.CHK    EQU    12              CHECK
 S.MFR    EQU    6               FROM SQUARE
 S.MTO    EQU    0               TO SQUARE
 L.MFR    EQU    6
 L.MTO    EQU    6
 L.MOD    EQU    5               NUMBER OF BITS IN SEARCH MODE
 S.NPM    EQU    S.MOD+L.MOD-1   NON-PERM = LAST BIT IN SEARCH MODE
 PERM     BIT    S.NPM-S.MOD
 M.CAP    BIT    S.CAP
 M.ACS    BIT    S.ACS
 M.PRO    BIT    S.PRO
 M.ENP    BIT    S.ENP
 M.CAS    BIT    S.CAS
 M.CHK    BIT    S.CHK
 M.MFR    EQU    7700B
 M.MTO    EQU    0077B
 LE.MOV   EQU    2               NUMBER OF WORDS/MOVE
 NPLY     EQU    30              MAXIMUM NUMBER OF PLY
 NHIS     EQU    50              NUMBER OF HISTORY POSITIONS TO KEEP
 LE.PAC   EQU    5               LENGTH OF PACKED BOARD ENTRY
 I        SET    NPLY*NPLY+NPLY

*         ECS ALLOCATION.

 L.LMB    EQU    I/2             LENGTH OF LIMBS
 L.PSH    EQU    1000B           LENGTH OF PAWN STRUCTURE HASH TABLE
 L.MSI    EQU    400B            LENGTH OF INDEX
 LE.TRA   EQU    5               LENGTH OF TRANSPOSITIONS TABLE ENTRY

          ORG    0

          IF     DEF,MAILBOX,1
 EC.MAIL  BSS    8               SNEAKY ECS BUFFER

 EC.LMB   BSS    L.LMB           LIMB ARRAY
 EC.PSH   BSS    L.PSH           PAWN STRUCTURE HASH TABLE
 EC.MSI   BSS    L.MSI           OPENING LIBRARY INDEX
 EC.TRA   BSS    0               TRANS/REF HASH TABLE

*         THE PIECE INDICES.

 PAWN     EQU    1
 ROOK     EQU    2
 NITE     EQU    3
 BISH     EQU    4
 QUEN     EQU    5
 KING     EQU    6

*         EVALUATION SCALING CONSTANTS.

 INFIN    EQU    377000B         INFINITE SCORE


 BASE     MODE   0               BASE NODE (PLY 0)
 MINI     MODE   1               MINIMUM GAME SCORE
 WOOD     MODE   2               CAPTURE SEARCH
 FULL     MODE   3               FULL SEARCHING
 PREL     MODE   4               PRELIMINARY SCORE (MINI SANS CAPS)
 LEGL     MODE   5               DETERMINE MOVE LEGALITY
 MATE     MODE   6               DETERMINE WHETHER MOVE IS MATE
 QUES     MODE   7               QUIESCENCE SEARCH
 POND     MODE   10B             PONDER SEARCH

**        FORMAT OF MATERIAL BALANCE WORD.
*
*         8/WUA,2/CWM,20/WS,8/WLA,2/CBM,20/BS, WHERE:
*         WUA = TOP 8 BITS OF SIGNED WHITE MATERIAL ADVANTAGE.
*         CWM = COARSE TOTAL WHITE MATERIAL / 512.
*         WS = WHITE MATERIAL SIGNATURE =
*         4/WQ,4/WB,4/WN,4/WR,4/WP, WHERE
*         WQ IS COUNT OF WHITE QUEENS, ETC.
*         WLA = LOWER 8 BITS OF SIGNED WHITE MATERIAL ADVANTAGE.
*         CBM = COARSE TOTAL BLACK MATERIAL / 512.
*         BS IS FORMATTED ANALOGOUSLY TO WS BUT FOR BLACK.

 S.WUA    EQU    52
 S.CWM    EQU    50
 S.WLA    EQU    22
 S.CBM    EQU    20
 L.WUA    EQU    8
 L.WLA    EQU    8
 L.CWM    EQU    2

**        DEFINITIONS OF FLAG BITS IN WIERD WORD.

 S.WRTC   EQU    0               PANIC TIME CONTROL CUTOFF
 S.WRA0   EQU    1               RE-ASPIRE AT PLY 0
 S.WRA1   EQU    2               RE-ASPIRE AT PLY 1
 S.WREZ   EQU    3               OBVIOUS MOVE FLAG
 KRONOS   IF     DEF,KRONOS
          TITLE  NOS MACROS.
 ATTACH   SPACE  4
***       ATTACH F
```

```
*
*               F = ADDRESS OF FET (WORKING FILE NAME).


          PURGMAC ATTACH

 ATTACH   MACRO  F
          SX6    3RPFM
          SX1    11B
          PX6    X6
          LX6    42
          LX1    24D
          BX6    X1+X6
          SX1    F
          BX6    X1+X6
          RJ     =XSYS=
          ENDM
 DEFINE   SPACE  4
***       DEFINE F
*
*               F = ADDRESS OF FET (WORKING FILE NAME).


          PURGMAC DEFINE

 DEFINE   MACRO  F
          SX6    3RPFM
          SX1    10B
          PX6    X6
          LX6    42
          LX1    24D
          BX6    X1+X6
          SX1    F
          BX6    X1+X6
          RJ     =XSYS=
          ENDM
 PURGE    SPACE  4
***       PURGE  F
*
*               F = ADDRESS OF FET (PERMANENT FILE NAME).


          PURGMAC PURGE

 PURGE    MACRO  F
          SX6    3RPFM
          SX1    3
          PX6    X6
          LX6    42
          LX1    24D
          BX6    X1+X6
          SX1    F
          BX6    X1+X6
          RJ     =XSYS=
          ENDM
 KRONOS   ENDIF
 ECERTB   TITLE  ECS ERROR RECOVERY MACROS.
***       ECS ERROR RECOVERY MACROS.
*

***       ECERTB - DEFINE ECS INSTRUCTION ADDRESS TABLE.
*
*LABL     ECERTB NEXT
*
*               LABL = NAME ON THIS SECTION OF THE TABLE.
*               NEXT = NAME ON NEXT SECTION OF THE TABLE. (OPTIONAL)
*



          MACRO  ECERTB,LABLXXX,NEXT
 ECERLC   MICRO  1,, 0
 ECERTY   MICRO  1,, 0-2000B
          IFC    NE, NEXT  ,2
 ECERLC   MICRO  1,, =X!NEXT
 ECERTY   MICRO  1,, 0

          RMT
          ENTRY  LABLXXX
 LABLXXX  BSS    0
          ECHO   1,A=('ECERLC'),B=('ECERTY')
          VFD    12/2000B+B,48/A
          RMT
          ENDM
 RE       SPACE  4
 RE       MACRO  ADDR
          LOCAL  XXXXXXXX
          IFC    NE,,ADDR,B3,,1
          ERR
 XXXXXXXX RL     ADDR
 ECERLC   MICRO  1,, XXXXXXXX,'ECERLC'
 ECERTY   MICRO  1,, 1,'ECERTY'
          ENDM
 WE       SPACE  4
 WE       MACRO  ADDR
          LOCAL  XXXXXXXX
          IFC    NE,,ADDR,B3,,1
          ERR
 XXXXXXXX WL     ADDR
 ECERLC   MICRO  1,, XXXXXXXX,'ECERLC'
 ECERTY   MICRO  1,, 2,'ECERTY'
          ENDM
 JIGGLE   SPACE  4,10
***       JIGGLE - DEFINE JIGGLE TABLE ENTRY.
*
*         JIGGLE VAR
*               VAR= NAME OF VARIABLE TO JIGGLE.


 JIGGLE   MACRO  VAR
          IF     DEF,.INITSWI,2    ASSEMBLE CODE ONLY IN INITSWI
          CON    VAR
          SKIP   1
          BSS    1
 JIGGLE   ENDM
 IOCOM    TITLE  INPUT/OUTPUT CONTROL MACROS.

*         MESSAGE NUMBERS.

 O.EMOV   EQU    0                ENTER MOVE
 O.EDBG   EQU    1                ENTER DEBUG COMMAND
 O.TGIO   EQU    2                THE GAME IS OVER
 O.HMTU   EQU    3                HOW MUCH TIME HAS BEEN USED
 CHARS    SPACE  4
 CHARS    MACRO  STRING,SYN,SEM
 STR      MICRO  1,, STRING
 N        MICCNT STR
 I        SET    1
          DUP    N
 C        MICRO  I,1, STRING
          POS    60-1R'C'
          VFD    1/1
 I        SET    I+1
          ENDD
          POS    0
 LOC      MICRO  1,1,*SYN*
 ADD      MICRO  1,,'LOC''LOC' +1
          VFD    30/SEM
          VFD    30/SYN'ADD'
          ENDM
 SWICH    SPACE  4
          MACRO  SWICH,NAM,DISP,TY,DEF
 S.NAM    EQU    SWBIT
          IFLT   SWBIT,18,1
 M.NAM    BIT    S.NAM
 SWBIT    SET    SWBIT+1
 X        MICRO  1,12,*DISP            *
 SWDISP   MICRO  1,,*'SWDISP''X'*
 SWTYPE   MICRO  1,, 'SWTYPE',TY
          IFC    EQ,*DEF*ON*,2
 SWDEFO   MICRO  1,, 1/1,'SWDEFO'
          ELSE   1
 SWDEFO   MICRO  1,, 1/0,'SWDEFO'
          ENDM
 SWBIT    SET    0
 SWDISP   MICRO  1,,
 SWTYPE   MICRO  1,,
 SWDEFO   MICRO  1,,
 SWICH    SPACE  4
**        SWITCH VALUES.


 DIS      SWICH  DISPLAY,DI,OFF
 ECH      SWICH  ECHO,EC,ON
 NOR      SWICH  (NO REPLY),NO,OFF
 M36      SWICH  (3.6 SYNTAX),36,OFF
 CAR      SWICH  CARDS,CA,ON
 DSD      SWICH  DSD,DS,OFF
 DBG      SWICH  DEBUG,DE,OFF
 TRC      SWICH  TRACE,TR,OFF
 LEW      SWICH  (LEARN WHITE),LW,OFF
 LEB      SWICH  (LEARN BLACK),LB,OFF
 EXP      SWICH  EXPERIENCE,EX,ON
 REC      SWICH  RECORD,RE,OFF
 WEI      SWICH  (WEIRD WORD),WE,ON
 SUM      SWICH  SUMMARY,SU,OFF
 NDC      SWICH  (NODE COUNTS),NC,OFF
 PRV      SWICH  VARIATION,VA,OFF
 LST      SWICH  LIST,LI,OFF
 TIM      SWICH  TIMING,TI,OFF
 ADI      SWICH  (A DISPLAY),AD,OFF
 BEL      SWICH  BELL,BE,ON
 DBA      SWICH  (DBG ALTER),DA,OFF
 MVA      SWICH  (MVE ALTER),MA,OFF
 ONL      SWICH  ONE-LINE,OL,OFF
 IML      SWICH  IMLAC,IM,OFF
 RAN      SWICH  RANDOM,RA,OFF
 PON      SWICH  PONDER,PO,OFF
 ALG      SWICH  ALGEBRAIC,AL,OFF
 CBD      SWICH  CBD,CB,OFF


 I        MICCNT SWDEFO
 SWDEFO   MICRO  1,I+6, SWCH/0,'SWDEFO'
 SWCH     EQU    60-SWBIT
 XLAT     SPACE  4,12
**        XLAT - BUILD TRANSLATION TABLE FOR RDRSXT.
*
*         XLAT   KEY,VALUE
*


 XLAT     MACRO  KEY,VALUE
 I        MICRO  1, KEY
 I        MICCNT I
          VFD    36/0L!KEY,6/I*6-1,18/VALUE
          ENDM
 CMND     SPACE  4,18
**        CMND - BUILD COMMAND TABLE.
*
*         CMND   KET,ROUTINE,PON
*
*         IF PON IS NON-NULL, THEN OK TO CONTINUE PONDERING OPP. MOVE.
*


 CMND     MACRO  MN,EPT,PON
 I        MICRO  1,, MN
 I        MICCNT I
 I        SET    I*6-1
          IF     -DEF,V.EPT,1
 V.EPT    EQU    0
          IF     DEF,E.EPT,1
 EPT      EQU    E.EPT
          IFC    NE, PON  ,1
 I        SET    I+40B
          VFD    24/0L!MN,6/I,12/V.EPT,18/EPT
          ENDM
 DISPOS   SPACE  4
 DISPOS   MACRO  LFN,TY,RECALL
          LOCAL  XXXXXXXX
          RMT
 XXXXXXXX DATA   0
          VFD    42/0L!LFN,18/2R!TY
          RMT
          BX6    X6-X6
          SA6    XXXXXXXX
          SYSTEM DSP,RECALL,A6
          ENDM
 KXI      SPACE  4,10
**        KXQ - LXQ IF PLUS, AXQ IF MINUS.
*
*         KXQ    VAL
*
*         WHERE  I = 0, 1, 2, 3, 4, 5, 6, OR 7.


 KXQ      OPDEF  R,V
 .KXVAL   SET    V
          IFMI   .KXVAL,2
          AX!R   -.KXVAL
          SKIP   1
          LX!R   .KXVAL
          ENDM
```

```
          END
*WEOR
*LEVEL *
CHESS
          IDENT CHESS
          SST
          LCC    OVERLAY(CHESS,0,0)
*
*         CHESS 4.5.
*
*         COPYRIGHT NORTHWESTERN UNIVERSITY 1976, ALL RIGHTS RESERVED.
*CALL     TABLE
          SPACE  4
*CALL     SQRLST
*CALL     MEMORY
*CALL     IOCOM
*CALL     BOARD
*CALL     LET
*CALL     RELY
*CALL     DEBUG
*CALL     TLET
          ORG    TABLE
          BSSZ   8
          DATA   10HW RNBQKBNR
          END
MTR*
          IDENT  MTR.
          TITLE                    MEM. - REQUEST STORAGE
**        MEM. - REQUEST STORAGE
*
*         ENTER: X1 = NEW FIELD LENGTH
*
*         USES:  A1,A6,X1,X6
*
*         CALLS  SYS=.
*



          ENTRY  MEM.
 MEM.     EQ     *+400000B
          SX6    X1+77B            ROUND UP TO NEAREST 100B
          AX6    6
          LX6    6+30
          SA6    MEMORY
          SYSTEM MEM,RECALL,A6
          EQ     MEM.

*CALL     MEMORY


          END
CODEDIO
          IDENT  CODEDIO
          SST
          LIST   F,X
          B1=1
 ECODEDI  ECERTB
 CODEDIO  TITLE  CODEDIO - CODED INPUT/OUTPUT.
*CALL SQRLST
*CALL     IOCOM

*         ASSEMBLY CONSTANTS

 LINBUF   EQU    102B              INPUT BUFFER LENGTH
 LOTBUF   EQU    102B              OUTPUT BUFFER LENGTH
          IF     DEF,TDISPLAY,1
 LDSBUF   EQU    14D*20D           DISPLAY BUFFER LENGTH
 LGMBUF   EQU    102B              GAME SCORE BUFFER LENGTH
 LHCBUF   EQU    102B              HARD COPY BUFFER LENGTH
 EOR      EQU    20B               EOR BIT
 W.CPRPV  CEQU   43B               RPV WORD IN CONTROL POINT AREA
 W.CPESP  CEQU   177B              ESP WORD IN CONTROL POINT AREA
*CALL     VERSION
*CALL     MEMORY
 FETS     SPACE  4
          ENTRY  INPUT
          ENTRY  OUTPUT
          ENTRY  GAMFILE
          ENTRY  HARDCPY
 INPUT    FILEC  BUFFIN,LINBUF,FET=6
 OUTPUT   FILEC  BUFFOT,LOTBUF,FET=6
          ORG    OUTPUT
          CON    0LOUTPUT+15B
          ORG    OUTPUT+6
 GAMFILE  FILEC  BUFFGM,LGMBUF,FET=6
 HARDCPY  FILEC  BUFFHC,LHCBUF,FET=6

 NUCC     IF     DEF,NUCC,4
          ORG    INPUT
          VFD    42/0LKEYBRD,18/1
          ORG    OUTPUT
          VFD    42/0LCONSOL,18/1
          ORG    HARDCPY
          DATA   0
          ORG    HARDCPY+6
          IF     DEF,TDISPLAY,6
 D        CON    0                 DISPLAY FET
          CON    BUFFDS
          CON    BUFFDS+20
          CON    BUFFDS
          CON    BUFFDS+LDSBUF
          CON    0

 BUFFOT   BSS    LOTBUF            OUTPUT BUFFER
          IF     DEF,TDISPLAY,1
 BUFFDS   BSSZ   LDSBUF            DISPLAY BUFFER
 BUFFHC   BSS    0                 HARDCOPY BUFFER
 BUFFGM   EQU    BUFFHC+LHCBUF     GAME SCORE BUFFER
 BUFFIN   EQU    BUFFGM+LGMBUF     INPUT BUFFER
 BUFFEND  EQU    BUFFIN+LINBUF     END OF BUFFERS

*         CHESS INITIALIZATION

          ENTRY  CHESS
 CHESS    RJ     =XPRELOD
          IF     DEF,TDISPLAY,2
          SX6    BUFFDS            POINTER TO DISPLAY BUFFER
          SA6    B0
          SX5    3
          SA4    64B
          MX2    42                MASK FOR FILE NAMES
          SB4    X4                NUMBER OF PARAMETERS
          ZR     B4,CHESS2         IF NO PARAMETERS
          SA1    B1+B1             (2) = IST PARAMETER
          BX1    X2*X1             EXTRACT TOP 7 CHARACTERS
          ZR     X1,CHESS1         IF FIRST PARAMETER NULL
          BX6    X5+X1
          SA6    INPUT             REPLACE INPUT FILE NAME
 CHESS1   SA1    A1+B1             (3) = 2ND PARAMETER
          BX1    X2*X1
          ZR     X1,CHESS2         IF SECOND PARAMETER NULL
          SB4    B4-1
          ZR     B4,CHESS2         IF NO SECOND PARAMETER
          BX6    X5+X1
          SA6    OUTPUT            REPLACE OUTPUT FILE NAME
 CHESS2   BSS    0
 KRONOS   IF     DEF,KRONOS
          SA1    INPUT
          MX2    42
          SX6    A1
          BX7    X2*X1
          BX6    X6+X7
          SA6    B1+B1
          SA1    OUTPUT
          SX6    A1
          BX7    X2*X1
          BX6    X6+X7
          SA6    A6+B1
 KRONOS   ENDIF
          OPEN   INPUT,ALTERNR,RECALL
          OPEN   OUTPUT,ALTERNR,RECALL
          RETURN GAMFILE,RECALL
          SA1    OUTPUT            SET COMPLETED WRITE IN FET
          MX2    42
          BX6    X2*X1
          SX1    15B
          BX6    X6+X1
          SA6    A1
 KRONOS   IF     -DEF,KRONOS
          SYSTEM REQ,RECALL,CHESSD
          SYSTEM REQ,RECALL,CHESSE
 KRONOS   ELSE
          SYSTEM CPM,RECALL,77777B,300B  SETTL,77777.
          SYSTEM CPM,RECALL,77777B,302B  SETASL,77777.
          SYSTEM CPM,RECALL,77777B,301B  SETJSL,77777.
 KRONOS   ENDIF
          SA1    INPUT
          SX2    EOR
          BX6    -X2*X1
          SA6    A1                CLEAR EOR BIT
          OPEN   GAMFILE,ALTERNR
          REWIND GAMFILE
          SA1    CHESSA            PAGE EJECT
          RJ     =XCARAGE
          SX5    CHESSB
          RJ     =XWRLINE

 NUCC     IF     DEF,NUCC
          SX2    W.CPRPV*100B
          SYSTEM TIM,RECALL,CPRPV,X2 GET RPV WORD
          SX2    W.CPESP*100B
          SYSTEM TIM,RECALL,CPESP,X2 GET ESP WORD
 NUCC     ENDIF

          BX7    X7-X7
          SA7    MEMORY+1          MEM ECS STATUS WORD
          IF     DEF,ECS,1
 ECS      IFEQ   ECS,1
          SYSTEM MEM,RECALL,A7,1   GET ECS SIZE
          SA2    A7
          AX2    30
          LX2    30
          BX6    X2
          SA6    A2
          NZ     X2,CHESS3         IF ECS AVAILABLE
 ECS      ENDIF

          IF     DEF,ECS,1
 ECS      IFEQ   ECS,1
          SX6    B1
          SA6    A6
 ECS      ENDIF
          IF     DEF,ECS,1
 ECS      IFEQ   ECS,0
          SX1    =XEREADER
          RJ     =XPLGECS          PLUG ECS INSTRUCTIONS
 ECS      ENDIF

          IF     -DEF,ECS,2
          SX5    CHESSC
          RJ     =XWRLINE

 CHESS3   SA1    SQRLT
          BX0    X1
          SB3    B0
          RJ     =XCHEKFE          GET MINIMUM REQUIRED FE
 KRONOS   IF     -DEF,KRONOS
          SX2    7700B
          RJ     =XERRSET          CALL RPV
 KRONOS   ELSE
          SA1    INPUT+1
          MX6    1
          BX1    -X6*X1
          AX1    48
          SX6    X1-2RTT
          NZ     X6,CHESS4         IF NOT INTERACTIVE JOB
          SA6    CPESP             CLEAR COMPLETE BIT
          SYSTEM TLX,RECALL,A6     DISABLE TERMINAL CONTROL
 CHESS4   BSS    0
 KRONOS   ENDIF

 MAILBOX  IF     DEF,MAILBOX
          SA0    CHESSA
          SX0    EC.MAIL
          SB3    1
          BX6    X6-X6
          SA6    A0                CLEAR MAIL BOX
          WE     B3
          RJ     =XECWERR
 MAILBOX  ENDIF

          JUMPUP SETEPT            INITIALIZE (1,0) OVERLAY

 CHESSA   DATA   1H1

 OVERLAY  IF     DEF,OVERLAY
 CHESSB   DATA   C+ HI.  THIS IS CHESS 'V'.  ENTER *HELP* FOR INSTRUCTIO
,NS.+
 OVERLAY  ELSE   1
 CHESSB   DATA   C* HI.  THIS IS CHESS 'V'.*

          IF     -DEF,ECS,1
 CHESSC   DATA   C+ ENTER *ECS ON* FOR BETTER RESPONSES.+
```

```
CHESSD   VFD    42/0LLIBRARY,18/0
         CON    1S31+1S28         *PF
         CON    0

CHESSE   VFD    42/0LGAMFILE,18/0
         CON    1S31+1S28

         BSS    BUFFEND-*
         ERRNG  BUFFEND-*
RDLINE   SPACE  4,40
***      RDLINE - INPUT LINE FROM STANDARD INPUT DEVICE.
*
*        ENTRY  (X1) = ADDRESS OF BUFFER.
*
*        EXIT   TO REREAD, IF FIRST TIME AND EOR/EOF.  SET S.DIS.
*               TO INPEOR, IF EOR/EOF AND NOT FIRST TIME.
*               TO CALLER, OTHERWISE.
*
*        USES   ALL REGISTERS EXCEPT A0, A5, X0, AND X5.
*        CALLS  JUMPUP, RDC=, DEVTYP.
*



         ENTRY  RDLINE
RDLINE   EQ     *+400000B
         BX4    X1
         SA1    INPUT+1
         RJ     =XDEVTYP          CHECK DEVICE TYPE
         PL     X6,RDLINE1        IF ALLOCATABLE (SO NOT INTERACTIVE)
         WRITER OUTPUT            ELSE FLUSH BUFFER
RDLINE1  READC  INPUT,X4,8
RDLINEA  ZR     X1,RDLINE2        IF NO EOR/EOF, TURN SWITCH
         SA1    SWICH
         SX6    B1
         SX7    B1
         LX7    S.CAR             CLEAR CARD MODE
         BX1    -X7*X1
DISPLAY  IF     DEF,DISPLAY
         BX6    X6+X1
         ERRNZ  S.DIS
         SA6    A1                SET DISPLAY SWITCH
         RJ     =XDUDLOAD
DISPLAY  ELSE
         SX6    B1
         LX6    S.DSD             SET DSD MODE
         BX6    X6+X1
         SA6    A1
DISPLAY  ENDIF
         RJ     =XREREAD

RDLINE2  SA1    RDLINEB
         BX6    X1
         SA6    RDLINEA           SET FIRST TIME SWITCH
         EQ     RDLINE            RETURN

RDLINE3  JUMPUP INPEOR

RDLINEB  ZR     X1,RDLINE         IF NO EOR/EOF
         EQ     RDLINE3           ELSE CHECK ALLOCATABILITY
WRLINE   SPACE  4,50
***      WRLINE - OUTPUT LINE ON STANDARD OUTPUT DEVICE.
*
*        FOR INTERCOM COMPATIBILITY, A LEADING BLANK IS ADDED TO
*        LINES WITHOUT ONE THAT ARE DESTINED FOR A NON-ALLOCATABLE
*        DEVICE.
*
*        ENTRY  (X5) = ADDRESS OF LINE.
*               (CARAGEA) = CARRIAGE CONTROL WORD.
*        EXIT   (CARAGEA) = 10H
*
*        USES   ALL REGISTERS EXCEPT X0, A0, AND A5.
*        CALLS  WTW=, WTC=, WSC=, CPLINE, DEVTYP.
*



         ENTRY  WRLINE
WRLINE   EQ     *+400000B
         SA1    OUTPUT+1
         RJ     =XDEVTYP          ELSE CHECK DEVICE TYPE
         SX2    OUTPUT
         NG     X6,WRLINE1        IF NON-ALLOCATABLE DEVICE
         WRITEW X2,CARAGEA,1
         EQ     WRLINE3

WRLINE1  BSS    0

NUCC     IF     DEF,NUCC
         SA1    MTIDS
         ZR     X1,WRLINE2        IF NOT MULTI-TERMINAL
         NG     X1,WRLINE2        IF MASTER UNKNOWN
         MX6    12
         BX6    X6*X1             EXTRACT MASTER ID
         SA1    WRLINEA           FILL
         BX6    X6+X1
         SA6    =XSYS.
         WRITEW X2,A6,1
NUCC     ENDIF

WRLINE2  SA1    X5                CHECK FIRST CHARACTER OF LINE
         MX6    6
         BX6    X6*X1
         LX6    6
         IF     -DEF,KRONOS,1
         IF     DEF,NUCC,1
         ZR     X6,WRLINE3        00 IS ESCAPE CODE
         SX6    X6-1R             COMPARE TO BLANK
         ZR     X6,WRLINE3        IF BLANK
         SB6    X5
         RJ     =XWSC=            ELSE INSERT BLANK
         EQ     WRLINE4

WRLINE3  WRITEC X2,X5             WRITE LINE
WRLINE4  RJ     =XCPLINE          PUT LINE ON HARDCPY
         SA1    CARAGEA+1
         BX6    X1
         SA6    A1-B1             UPDATE CARAGEA
         IF     DEF,ONELINE,1
         RJ     =XONLOUT          OUTPUT THROUGH EI2/ONE-LINE
         RJ     =XDUDINFO         DISPLAY LINE
         RJ     =XCNTLIN          COUNT LINE
         EQ     WRLINE            RETURN

NUCC     IF     DEF,NUCC
WRLINEA  VFD    12/0              ID
         VFD    12/0060B          ENTER ASCII MODE
         VFD    12/4000B          EXIT ASCII MODE
         VFD    12/0060B          ENTER ASCII MODE
         VFD    12/4000B          EXIT ASCII MODE
NUCC     ENDIF
PDB      SPACE  4
**       PDB - PROCESS DISPLAY BUFFER.
*        ENTRY  (X5) = ADDRESS OF OUTPUT LINE.



TDISPLAY IF     DEF,TDISPLAY
PDB      EQ     *+400000B   ENTRY/EXIT
         WRITEC D,X5        ADD LINE TO BUFFER

*        ROLL BUFFER.

         SA1    D+1         FIRST
         SA2    A1+B1       IN
         MX7    -12
         SB7    X2
         SA3    X1          SEARCH FOR END OF LINE
PDB1     BX6    -X7*X3
         SA3    A3+B1
         NZ     X6,PDB1     LOOP
         SB6    A3          COPY BUFFER
         BX6    X3
         SA6    X1
PDB2     SA3    A3+B1
         SB6    B6+B1
         BX6    X3
         SA6    A6+B1
         LT     B6,B7,PDB2  LOOP
         SX7    A6          RESET IN
         SA7    D+2
         SB6    A6
         MX6    0           CLEAR END OF BUFFER
PDB3     SA6    A6+B1
         SB6    B6+B1
         LT     B6,B7,PDB3  LOOP
         JP     PDB         RETURN
TDISPLAY ENDIF
CPLINE   SPACE  4,21
***      CPLINE - OUTPUT LINE ON COPY DEVICE.
*
*        ENTRY  (X5) = ADDRESS OF LINE.
*               (CARAGEA) = CARRIAGE CONTROL WORD.
*
*        USES   ALL REGISTERS EXCEPT X0, A0, AND A5.
*        CALLS  WTW=, WTC=.
*



         ENTRY  CPLINE
CPLINE   EQ     *+400000B
         IF     DEF,TDISPLAY,1
         RJ     PDB               PROCESS DISPLAY BUFFER
         SA1    HARDCPY
         ZR     X1,CPLINE2        IF NO HARD COPY DEVICE
         SA4    X5                CHECK FIRST BYTE
         AX4    48
         SX4    X4-0060B
         ZR     X4,CPLINE2        IF TTY ONLY LINE
         SX2    A1
         WRITEW X2,CARAGEA,1      ELSE WRITE CARAGE CONTROL
CPLINE1  WRITEC X2,X5

NUCC     IF     -DEF,NUCC
CPLINE2  EQ     CPLINE

NUCC     ELSE

CPLINE2  SA1    MTIDS
         MX6    12
         BX7    -X6*X1
CPLINE3  ZR     X7,CPLINE         IF NO MORE SLAVES
         SA3    WRLINEA           FILL
         LX1    12
         BX1    X6*X1
         BX6    X1+X3
         LX7    12
         SA6    =XSYS.
         SA7    CPLINEA
         WRITEW OUTPUT,A6,1
         WRITEC X2,X5
         SA1    CPLINEA
         MX6    12
         BX7    -X6*X1
         EQ     CPLINE3

CPLINEA  BSS    1                 REMAINING SLAVES
NUCC     ENDIF
ERRMSG   SPACE  4,16
***      ERRMSG - DISPLAY ERROR MESSAGE.
*
*        ENTRY  (X5) = ADDRESS OF MESSAGE.
*
*        CALLS  CARAGE, DUDERR, WRLINE.
*        USES   ALL REGISTERS.
*



         ENTRY  ERRMSG
ERRMSG   EQ     *+400000B
         SA1    =10H *********
         RJ     =XCARAGE
         RJ     =XWRLINE
         RJ     =XDUDERR
         SA1    SWICH
         SX5    ILINE
         LX1    59-S.ECH
         NG     X1,ERRMSG         IF ECHOING, DO NOT REPEAT LINE
         RJ     =XWRLINE          ECHO LINE
         EQ     ERRMSG
CARAGE   SPACE  4,18
***      CARAGE - SET CARAGE CONTROL WORDS.
*
*        ENTRY  (X1) = NEW CARRIAGE CONTROL WORD.
*
*        EXIT   (CARAGEA) = NEW CARRIAGE CONTROL WORD.
*
*        USES   X6, A6.
*



         ENTRY  CARAGE
CARAGE   EQ     *+400000B
         BX6    X1
         SA6    CARAGEA
         EQ     CARAGE
```

```
 CARAGEA  DATA   1H               CARRIAGE CONTROL WORD
          DATA   1H               DEFAULT CARRIAGE CONTROL

 CNTLIN   SPACE  4,40
***       CNTLIN - COUNT LINE.
*
*         ENTRY  (LINES) = LINES REMAINING ON PAGE.
*                        < 0, IF NO PAGING ALLOWED.
*                        = 0, IF PAGE AT FIRST OPPORTUNITY.
*                (INPFN) = 0, IF ON PRIMARY FILE.
*                (SWICH, S.CAR) = 1, IF INPUT FROM A FILE.
*
*         EXIT   (LINES) = LINES REMAINING ON PAGE.
*
*         CALLS  READER, DEVTYP.
*         USES   ALL REGISTERS.
*



 CNTLIN   EQ     *+400000B
          SA1    LINES
          NG     X1,CNTLIN         IF NO PAGING
          ZR     X1,CNTLIN1        IF ALREADY AT END OF PAGE
          SX6    B1
          IX6    X1-X6             COUNT LINE
          SA6    A1+
          NZ     X6,CNTLIN         IF NOT AT END OF PAGE
 CNTLIN1  SA1    SWICH
          LX1    59-S.CAR
          PL     X1,CNTLIN         IF NOT READING A FILE
          SA1    INPFN
          NZ     X1,CNTLIN         IF NOT ON PRIMARY FILE
          SA1    =XINPUT+1
          RJ     =XDEVTYP          CHECK INPUT DEVICE TYPE
          PL     X6,CNTLIN         IF ALLOCATABLE
          SA1    =XOUTPUT+1        CHECK OUTPUT DEVICE ALSO
          RJ     =XDEVTYP
          PL     X6,CNTLIN         IF ALLOCATABLE
          WRITEC =XOUTPUT,CNTLINA
          MX6    1                 PAGE WAIT READ
          RJ     =XREADER
          EQ     CNTLIN            RETURN

 CNTLINA  CON    10H (PAUSING)
          VFD    12/0057B          SUPPRESS CRLF
          VFD    12/0000B          CRLF
          VFD    36/0              IGNORED
 DEVTYP   SPACE  4,10
**        DEVTYP - CHECK FOR ALLOCATABLE DEVICE (EXCEPT TTY).
*
*         ENTRY  (X1) = 2ND WORD OF FET (FIRST).
*         EXIT   (X6) NEGATIVE IF NON-ALLOCATABLE.

 DEVTYP1  MX6    1                 SET NON-ALLOCATABLE

          ENTRY  DEVTYP
 DEVTYP   PS     0                 ENTRY-EXIT
          MX6    12
          BX6    X1*X6             EXTRACT DEVICE TYPE
          NG     X6,DEVTYP         IF NON-ALLOCATABLE
          LX6    12
          SX6    X6-2RTT
          ZR     X6,DEVTYP1        IF TTY (KRONOS DEVICE TYPE)
          BX6    X6-X6             SET ALLOCATABLE
          EQ     DEVTYP
*CALL     COMCSYS
          ENTRY  SYS=
          ENTRY  DFM=
          ENTRY  RCL=
          ENTRY  WNB=
          ENTRY  TIM=
          ENTRY  MEM=
          ENTRY  SYS.
*CALL     COMCCIO
          ENTRY  CIO=
 OPE=     EQU    CIO=
 CLO=     EQU    CIO=
          ENTRY  CLO=
          ENTRY  OPE=
*CALL     COMCRDW
          ENTRY  RDW=
          ENTRY  RDX=
          ENTRY  LCB=
*CALL     COMCRDC
          ENTRY  RDC=
*CALL     COMCWTW
          ENTRY  WTW=
          ENTRY  WTX=
          ENTRY  DCB=
*CALL     COMCWTC
          ENTRY  WTC=
*CALL COMCWSC
          ENTRY  WSC=
*CALL     COMCPFM
          ENTRY  PFM=
          END    CHESS
MASSIO
          IDENT  MASSIO
 MASSIO   TITLE  MASSIO - RANDOM ACCESS CHESS LIBRARY INPUT/OUTPUT.
          SST
          LIST   X
 EMASSIO  ECERTB ECODEDI
*CALL     BOARD
*CALL     MEMORY
*CALL     LET
*CALL     RELY
 MSDATA   SPACE  4,20
 MSIDW    BSS    1                 CURRENT INDEX WORD
          ENTRY  MSEOI             SET BY MSOPEN IN CHESS30
 MSEOI    DATA   6                 EOI POINTER
 MSBLK    BSSZ   127               BUFFER
 MSBLKE   BSS    0                 END OF BUFFER
 MSHSH    DATA   0                 CURRENT HASH CODE
 MSREC    BSS    6                 TEMPORARY BUFFER
 MSNXT    DATA   0                 NEXT PRU NUMBER
 MSPOS    DATA   0                 CURRENT PRU NUMBER
 MSMOD    DATA   0                 BUFFER MODIFICATION FLAG
 MSNAM    BSS    1                 NAME OF POSITION
 MSMSK    BSS    1                 MASK FOR WORD ONE
 MSWSA    BSS    1                 WORKING STORAGE ADDRESS
 MSIMD    CON    0                 INDEX MODIFIED FLAG

          ENTRY  LIBRARY
 LIBRARY  RFILEB MSBLK,MSBLKE-MSBLK+1,FET=7
 MSRWND   SPACE  4,14
***       MSRWND - REWIND RANDOM FILE.
*


          ENTRY  MSRWND
 MSRWND   EQ     *+400000B
          SA0    MSIDW
          SX0    EC.MSI
          SB3    1
          RE     B3
          RJ     =XECRERR
          SA1    A0
          BX6    X1
          BX7    X7-X7
          SA7    MSHSH
          SA6    MSNXT
          RJ     MSSETP            SET POSITION
          EQ     MSRWND            RETURN
 MSREAD   SPACE  4,20
***       MSREAD - READ NEXT BOARD POSITION.
*
*         ENTRY  (X6) = LOCATION TO RETURN NEXT PACKED POSITION.
*
*         EXIT   (X6) = 0, IF EOI.
*



          ENTRY  MSREAD
 MSREAD   EQ     *+400000B
          SA6    MSWSA             SAVE WSA ADDRESS
 MSREAD1  RJ     MSRREC            READ RECORD
          NZ     X1,MSREAD2        IF END OF BLOCK
          RJ     MSMOVE            RETURN PACKED BOARD
          SX6    B1
          EQ     MSREAD            RETURN

 MSREAD2  RJ     MSSETP            ADVANCE TO NEXT BLOCK
          NZ     X1,MSREAD1        IF NOT END OF SUBFILE
          RJ     MSADVC            ADVANCE SUBFILE
          ZR     X6,MSREAD         IF DONE
          EQ     MSREAD1           TRY NEXT SUBFILE
 MSCLOS   SPACE  4,28
***       MSCLOS - REWRITE INDEX.
*



          ENTRY  MSCLOS
 MSCLOS   EQ     *+400000B
          SA1    MSMOD
          ZR     X1,MSCLOS1        IF NO MODS
          RJ     MSRWRT            REWRITE LAST BLOCK
 MSCLOS1  SA1    MSIMD
          ZR     X1,MSCLOS         IF NO MOD TO INDEX
          BX6    X6-X1
          SA6    A1
          REWIND LIBRARY,RECALL
          SA1    MEMORY+1
          ZR     X1,MSCLOS2        IF FAKE ECS
          SX0    EC.MSI
          SB6    L.MSI
          RJ     =XWLW=            WRITE INDEX
          EQ     MSCLOS3

 MSCLOS2  SA1    A1+B1             FWA FAKE ECS
          SB6    X1+EC.MSI
          SB7    L.MSI
          SX2    LIBRARY
          RJ     =XRWW=            RE-WRITE WORDS

 MSCLOS3  SB6    MSEOI
          SX2    LIBRARY
          SB7    B1
          RJ     =XRWW=
          REWRTR LIBRARY,RECALL    REWRITE INDEX
          SX6    0
          SA6    MSBLK             CLEAR BUFFER
          EQ     MSCLOS            RETURN
 MSFLSH   SPACE  4,10
***       MSFLSH - FLUSH BUFFER.
*



          ENTRY  MSFLSH
 MSFLSH   EQ     *+400000B
          SA1    MSMOD
          ZR     X1,MSFLSH         IF NOTHING TO DO
          RJ     =XMSRWRT
          EQ     MSFLSH
 MSREPL   SPACE  4,88
***       MSREPL - REPLACE POSITION.
*
*         ENTRY  (A0) = ADDRESS OF POSITION TO WRITE.
*                (X7) = NAME, =0 IF SEARCH BY POSITION.
*
*         EXIT   (X6) = 0, IF POSITION REPLACED.
*                (X6) =-1, IF POSITION NOT REPLACED BECAUSE IDENTICAL.
*                (X6) =+1, IF POSITION ADDED.
*



          ENTRY  MSREPL
 MSREPL   EQ     *+400000B
          RJ     MSHASH            GET HASH CODE
 MSREPL1  RJ     MSSETP            SET POSITION
          ZR     X1,MSREPL5        IF END OF SUBFILE
 MSREPL2  RJ     MSRREC            READ RECORD
          NZ     X1,MSREPL1        IF END OF RECORD
          RJ     MSMACH            COMPARE
          NZ     X6,MSREPL2        IF NO MATCH
          SB2    4
          SA1    MSNAM
          SA2    MSREC
          IX6    X1-X2
          NZ     X6,MSREPL4        IF DIFFERENT NAMES
 MSREPL3  SA1    A0+B2
          SA2    MSREC+1+B2
          IX6    X1-X2
          NZ     X6,MSREPL4        IF DIFFERENT
          SB2    B2-B1
          PL     B2,MSREPL3        IF MORE TO COMPARE
          MX6    59
          EQ     MSREPL            RETURN NO CHANGE

 MSREPL4  RJ     MSINRT            INSERT RECORD
          BX6    X6-X6
          EQ     MSREPL            RETURN DONE

 MSREPL5  RJ     MSAPND            APPEND BOARD TO SUBFILE
```

```
         SX6    B1
         EQ     MSREPL
 MSSRCH  SPACE  4,88
***      MSSRCH - SEARCH FOR A POSITION UNDER MASK.
*
*        ENTRY  (A0) = ADDRESS OF PACKED BOARD POSITION.
*               (X7) = NAME. =0, IF POSITION SEARCH.
*               (X6) = LOCATION TO RETURN POSITION.
*
*        EXIT   (X6) = 0, IF NOT FOUND.
*               (X6) = 1, IF FOUND.
*               (X7) = NAME WORD.
*



         ENTRY  MSSRCH
 MSSRCH  EQ     *+400000B
         SA6    MSWSA             SAVE WSA ADDRESS
         RJ     MSHASH            GET HASH CODE
 MSSRCH1 RJ     MSSETP            SET POSITION
         ZR     X1,MSSRCH         IF END OF SUBFILE
 MSSRCH2 RJ     MSRREC            READ RECORD
         NZ     X1,MSSRCH1        IF END OF RECORD
         RJ     MSMACH            COMPARE
         NZ     X6,MSSRCH2        IF NO MATCH
         RJ     MSMOVE            RETURN PACKED BOARD
         SA1    MSREC
         BX7    X1                RETURN NAME WORD
         SX6    B1                RETURN SUCCESS
         EQ     MSSRCH            RETURN

 MSADVC  SPACE  4,22
**       MSADVC - ADVANCE TO NEXT SUBFILE.
*
*        ENTRY  (MSHSH) = OLD SUBFILE NUMBER.
*
*        EXIT   (MSHSH) = NEW SUBFILE NUMBER.
*               (MSNXT) = FIRST PRU NUMBER OF SUBFILE.
*               (X6) = 0, IF END OF LAST SUBFILE.
*



 MSADVC  EQ     *+400000B
         SA1    MSHSH
         MX7    -8
         SX6    X1+B1
         BX6    -X7*X6
         SA6    A1
         SX0    EC.MSI+X6
         SA0    MSNXT
         SB3    B1
         RE     B3
         RJ     =XECRERR
         SA1    MSHSH
         BX6    X1
         EQ     MSADVC
 MSAPND  SPACE  4,62
**       MSAPND - APPEND POSITION TO CURRENT SUBFILE.
*
*        ENTRY  (A0) = POSITION.
*               (MSNAM) = NAME.
*



 MSAPND  EQ     *+400000B
         SA1    MSHSH
         SX0    EC.MSI+X1
         SX5    A0
         SB3    B1
         SA0    MSIDW
         RE     B3
         RJ     =XECRERR
         SA1    A0
         ZR     X1,MSAPND1        IF NO FIRST BLOCK
         SA1    MSBLK             BLOCK HEADER
         LX1    -24
         MX4    48
         BX2    -X4*X1
         SX3    X2-21*6
         NZ     X3,MSAPND3        IF BLOCK NOT FULL
         SA4    MSEOI
         LX4    36
         LX1    24
         IX6    X1+X4
         SA6    A1                UPDATE HEADER
         RJ     MSRWRT            REWRITE
         EQ     MSAPND2

 MSAPND1 SA0    MSEOI
         WE     B3                UPDATE INDEX
         RJ     =XECWERR
         SA1    MSMOD
         SX6    1
         SA6    MSIMD             FLAG INDEX MOD
         ZR     X1,MSAPND2        IF NO MODS TO CURRENT BLOCK
         RJ     MSRWRT            REWRITE IT
 MSAPND2 SA4    MSEOI
         BX7    X4
         SX6    B1+B1
         SA7    MSBLK
         IX7    X7+X6
         SA7    A4                UPDATE EOI POINTER
         SX6    MSBLKE
         SX7    MSBLK
         SA6    LIBRARY+2         IN
         SA7    A6+B1             OUT
         BX6    X6-X6
         SA6    LIBRARY+6
         SKIPEI LIBRARY,RECALL
         WRITER X2,RECALL         RESERVE 2 PRUS AT EOI
 KRONOS  IF     -DEF,KRONOS
         SA1    RELSW
         ZR     X1,MSAPND3        IF NOT RELIABILITY MODE
         CLOSE  X2,NR,RECALL
         SX7    30B
         SX6    3RPFE
         SX2    LIBFDB
         RJ     =XPFM=            EXTEND LIBRARY
 KRONOS  ENDIF
 MSAPND3 SA1    MSBLK
         SX2    6
         LX2    24
         IX6    X2+X1
         SA6    A1                ADVANCE AMOUNT USED
         AX1    24
         SA6    MSMOD             INDICATE BLOCK MODIFIED
         SX1    X1+MSBLK+1        FWA IN BUFFER
         SA2    MSNAM
         BX6    X2
         SA6    X1                STORE NAME IN BUFFER
         SX1    X1+B1
         SB2    4                 MOVE 5 WORDS
 MSAPND4 SA2    X5+B2
         BX6    X2
         SA6    X1+B2
         SB2    B2-B1
         PL     B2,MSAPND4
         EQ     MSAPND            RETURN
 MSPACK  SPACE  4,88
***      MSPACK - PACK CURRENT BOARD POSITION.
*
*        ENTRY  (A0) = LOCATION TO STORE PACKED POSITION.
*               (B3) = POINTER TO THE BOARD POSITION TO PACK.
*               (X3) = STATUS WORD.
*               (MSCODE) = CODE TO PACK.
*
*        EXIT   POSITION PACKED INTO 5 WORDS:
*               WORD 1: 18/0,10/MSCODE,4/MCS,4/CST,4/WEP,4/BEP,16/SQRS
*               MCS = 1/W TO MOVE,1/W IN CHECK,1/B TO MOVE,1/B IN CHECK
*               CST = 1/W O-O-O OK,1/W O-O OK,1/B O-O-O OK,1/B O-O OK
*               WEP = WHITE EP FILE, 0 IF NONE, 1 IF KR FILE, ETC.
*               BEP = BLACK EP FILE, 0 IF NONE, 1 IF KR FILE, ETC.
*               SQRS = 4/QR1,4/QN1,4/QB1,4/Q1. EACH 7+PIECE ON SQUARE.
*               WORD 2: 4/K1, 4/KB1, ETC.
*               WORD 3: 4/Q3, 4/K3, ETC.
*               WORD 4: 4/QB5, 4/Q5, ETC.
*               WORD 5: 4/QN7, 4/QB7, ETC.
*
*               ALL SQUARES ARE NAMED FROM THE WHITE SIDE.
*               THE LOWEST 18 BITS OF THE BEST MOVE WILL BE PLACED
*               INTO THE UPPER 18 BITS OF WORD 1 BY THE CALLING
*               ROUTINE.
*



         ENTRY  MSPACK
 MSPACK  EQ     *+400000B
         SA1    MSCODE
         MX2    -10
         BX2    -X2*X1
         BX1    X3                STATUS WORD
         MX5    1
         BX3    X5*X1
         LX5    57-59
         BX4    X5*X1
         LX4    58-57
         BX3    X4+X3
         LX5    29-57
         BX4    X5*X1
         LX4    57-29
         BX3    X3+X4
         LX5    27-29
         BX4    X5*X1
         LX4    56-27
         BX3    X3+X4
         LX5    47-27
         BX4    X5*X1
         LX4    55-47
         BX3    X3+X4
         LX5    40-47
         BX4    X5*X1
         LX4    54-40
         BX3    X3+X4
         LX5    17-40
         BX4    X5*X1
         LX4    53-17
         BX3    X3+X4
         LX5    10-17
         BX4    X5*X1
         LX4    52-10
         BX3    X3+X4
         BX6    X3+X2
         LX6    32-16
         SX5    B1
         LX1    12
         MX0    -9
         BX2    -X0*X1            WHITE EP BITS
         BX2    X2+X5
         NX2    X2,B2
         SX2    B2-47
         LX2    20-16
         IX6    X6-X2
         LX1    30
         BX2    -X0*X1            BLACK EP BITS
         BX2    X2+X5
         NX2    X2,B2
         SX2    B2-47
         LX2    16-16
         IX6    X6-X2
         LX5    4*11-1
         MX4    -3                BIAS = -7
         SA1    B3                FWA OF THE BOARD TO PACK
         SB3    A0+5              LWA+1 TO STORE
         SB2    A0                FWA TO STORE
 MSPACK1 IX1    X1-X4             BIAS
         LX6    4
         LX5    4
         BX6    X1+X6
         SA1    A1+B1
         PL     X5,MSPACK1        IF NOT END OF WORD
         SA6    B2
         SB2    B2+B1
         BX6    X6-X6
         LT     B2,B3,MSPACK1     IF NOT END OF BOARD
         EQ     MSPACK            RETURN
 MSUNPK  SPACE  4,88
***      MSUNPK - UNPACK LAST POSITION READ INTO BOARD.
*
*        ENTRY  (MSREC) = POSITION.
*
*        EXIT   (BOARD) = POSITION.
*



         ENTRY  MSUNPK
 MSUNPK  EQ     *+400000B
         SA1    MSREC+1
         MX0    -4
         MX5    -2
         SX4    B1+B1
         LX1    28+2
         BX2    -X5*X1            WHITE ON MOVE/CHECK
         LX1    2
         BX3    -X5*X1            BLACK ON MOVE/CHECK
```

```
         IX2    X2+X4
         IX3    X3+X4
         BX2    -X4*X2
         BX3    -X4*X3
         LX2    30
         BX6    X2+X3
         LX6    30-3
         LX1    4
         BX2    -X0*X1            CASTLE STATUS
         SA3    X2+MSUNPKA
         BX6    X6+X3
         LX1    4
         BX2    -X0*X1            WHITE EP
         SB2    X2-1
         SX4    B1
         LX3    X4,B2
         LX3    48+1
         BX6    X3+X6
         LX1    4
         BX2    -X0*X1            BLACK EP
         SB2    X2-1
         LX3    X4,B2
         LX3    18+1
         BX6    X3+X6
         SA6    STATUS
         MX5    -3                -7 BIAS
         LX4    11*4-1
         SB2    BOARD             FWA
         SB3    A6                LWA+1
MSUNPK1  LX1    4
         LX4    4
         BX6    -X0*X1
         IX6    X6+X5             REMOVE BIAS
         SA6    B2
         SB2    B2+B1
         PL     X4,MSUNPK1        IF NOT DONE WITH WORD
         SA1    A1+B1
         LT     B2,B3,MSUNPK1     IF NOT DONE WITH BOARD
         EQ     MSUNPK            RETURN

MSUNPKA  BSS    0
         LIST   G
I        SET    0
1        DUP    2
J        SET    0
2        DUP    2
IJ       SET    I+J+1
IJ       SET    IJ/2
K        SET    0
3        DUP    2
L        SET    0
4        DUP    2
KL       SET    K+L+1
KL       SET    KL/2
         VFD    12/0,1/I,3/0,1/IJ,2/0,1/J,10/0
         VFD    12/0,1/K,3/0,1/KL,2/0,1/L,10/0
L        SET    1
4        ENDD
K        SET    1
3        ENDD
J        SET    1
2        ENDD
I        SET    1
1        ENDD
         LIST   -G
MSHASH   SPACE  4,50
**       MSHASH - COMPUTE HASH CODE.
*
*        ENTRY  (X7) = NAME, 0 IF POSITION SEARCH.
*               (A0) = LOCATION OF POSITION.
*
*        EXIT   (MSNAM) = NAME.
*               (MSMSK) = MASK FOR WORD ONE.
*               (MSHSH) = HASH CODE.
*               (MSNXT) = FIRST PRU TO READ.
*


MSHASH   EQ     *+400000B
         SA7    MSNAM
         SA1    MSMASK
         MX6    42
         BX7    X6*X7
         MX2    -10
         BX3    -X2*X1
         LX3    -28
         MX2    -32
         BX6    -X2+X3            BUILD WORD ONE MASK
         SA6    MSMSK
         NZ     X7,MSHASH1        IF NAME SEARCH
         SA1    A0                FIRST WORD OF POSITION
         BX1    -X2*X1
         SA2    A1+B1
         SA3    A2+B1
         SA4    A3+B1
         SA5    A4+B1
         BX7    X1-X2
         BX7    X3-X7
         BX7    X4-X7
         BX7    X5-X7
MSHASH1  BX6    X7
         LX6    30
         BX7    X6-X7
         BX6    X7
         LX6    15
         BX7    X6-X7
         BX6    X7
         LX6    7
         BX7    X6-X7
         MX6    -8
         BX7    -X6*X7
         SA7    MSHSH
         SX0    EC.MSI+X7
         SX5    A0
         SB3    B1
         SA0    MSNXT
         RE     B3
         RJ     =XECRERR
         SA0    X5
         EQ     MSHASH            RETURN
MSINRT   SPACE  4,25
**       MSINRT - INSERT BOARD POSITION.
*
*        ENTRY  (LIBRARY+3) = LWA+1 TO MOVE TO.
*               (A0) = FWA TO MOVE FROM.
*
*        EXIT   (MSMOD) ' 0.
*


MSINRT   EQ     *+400000B
         SA1    LIBRARY+3         OUT = LWA+1 TO MOVE TO
         SA2    MSNAM
         BX6    X2                MOVE NAME
         SA6    X1-6
         SB2    X1                LWA+1 TO MOVE TO
         SA2    A0                FIRST WORD TO MOVE
         SX7    1
MSINRT1  BX6    X2                MOVE POSITION
         SA6    A6+B1
         SB3    A6+B1
         SA2    A2+B1             NEXT WORD OF POSITION
         LT     B3,B2,MSINRT1     IF NOT DONE MOVING POSITION
         SA7    MSMOD             INDICATE BUFFER CHANGED
         EQ     MSINRT
MSMACH   SPACE  4,40
**       MSMACH - COMPARE RECORD.
*
*        ENTRY  (MSREC) = RECORD FROM DISK.
*               ((A0)) = RECORD TO BE FOUND.
*               (MSNAM) = 0, IF POSITION SEARCH.
*               (MSNAM) = NAME, IF NAME SEARCH.
*               (MSMSK) = MASK FOR FIRST POSITION WORD.
*
*        EXIT   (X6) = 0, IF MATCH.
*


MSMACH   EQ     *+400000B
         SA1    MSNAM
         ZR     X1,MSMACH1        IF POSITON SEARCH
         SA2    MSREC
         MX6    42
         BX1    X6*X1
         BX2    X6*X2
         IX6    X1-X2
         EQ     MSMACH

MSMACH1  SB2    5                 NUMBER OF WORD TO COMPARE
         SA1    MSREC
         SX6    B1
         NZ     X1,MSMACH         IF NAMED POSITION
         SA1    A1+B1             FIRST POSITION WORD
         SA2    A0
         SA3    MSMSK
         BX1    X3*X1
         BX2    X2*X3
MSMACH2  IX6    X1-X2
         NZ     X6,MSMACH         IF MISMATCH
         SB2    B2-B1
         ZR     B2,MSMACH         IF MATCH
         SA1    A1+B1
         SA2    A2+B1
         EQ     MSMACH2           COMPARE NEXT WORD
MSMOVE   SPACE  4,20
**       MSMOVE - MOVE RECORD TO CALLER.
*
*        ENTRY  (MSWSA) = ADDRESS TO MOVE RESULT.
*               (MSREC) = DATA TO MOVE.
*


MSMOVE   EQ     *+400000B
         SA1    MSWSA
         SB2    5
MSMOVE1  SA2    MSREC+B2
         BX6    X2
         SA6    X1+B2
         SB2    B2-B1
         PL     B2,MSMOVE1
         EQ     MSMOVE
MSRREC   SPACE  4,33
**       MSRREC - READ RECORD.
*
*        EXIT   (X1) ' 0, IF END OF BLOCK.
*


MSRREC   EQ     *+400000B
         SX1    B1
         MX4    36
         SA2    MSPOS             REQUESTED LOCATION
         ZR     X2,MSRREC         RETURN IF NO BLOCK
         SA3    MSBLK             ACTUAL LOCATION
         BX5    X2-X3
         BX5    -X4*X5
         ZR     X5,MSRREC2        IF BLOCK IN CORE
         SA1    MSMOD
         ZR     X1,MSRREC1        IF BLOCK NOT MODIFIED
         RJ     MSRWRT            REWRITE BLOCK
MSRREC1  SX6    MSBLK
         SA6    LIBRARY+2         SET IN
         SA6    A6+B1             SET OUT
         SA1    MSPOS
         BX6    X1
         SA6    LIBRARY+6
         READSK LIBRARY,RECALL
         SA1    LIBRARY+3
         SX6    X1+B1             SKIP HEADER WORD
         SA6    A1
         SA2    X1
         MX4    48
         AX2    24
         BX1    -X4*X2            EXTRACT DATA LENGTH
         IX6    X1+X6
         SA6    A1-B1             SET IN TO END OF DATA
         AX2    12
         BX6    X2
         SA6    MSNXT             NEXT BLOCK IN SUBFILE
MSRREC2  READW  LIBRARY,MSREC,6
         EQ     MSRREC
MSRWRT   SPACE  4,20
**       MSRWRT - REWRITE CURRENT BLOCK.
*
*        ENTRY  (MSBLK) = BLOCK TO REWRITE.
*


MSRWRT   EQ     *+400000B
         BX6    X6-X6
         SA6    MSMOD
         MX4    36
         SA3    MSBLK
         BX6    -X4*X3
```

```
          SA6    LIBRARY+6         SET PRU NUMBER
          SX7    A3
          SX6    MSBLKE
          SA6    LIBRARY+2         SET IN
          SA7    A6+B1             SET OUT
          REWRTR LIBRARY,RECALL    REWRITE BLOCK
          EQ     MSRWRT
 MSSETP   SPACE  4,18
**        MSSETP - SET PRU NUMBER.
*
*         ENTRY  (MSNXT) = PRU NUMBER.
*
*         EXIT   (MSPOS) = PRU NUMBER.
*                (X1) = 0, IF END OF SUBFILE.
*

 MSSETP   EQ     *+400000B
          SA1    MSNXT
          BX6    X1
          SA6    MSPOS
          SX7    MSBLK+1
          SA7    LIBRARY+3         SET OUT
          SA2    MSBLK
          MX4    48
          AX2    24
          BX4    -X4*X2            DATA LENGTH
          IX7    X7+X4
          SA7    A7-B1             SET IN
          AX2    12
          BX7    X2
          SA7    A1
          EQ     MSSETP
 ERE      SPACE  4,20
**        ERE - ECS READ ERROR.
*
*         ENTRY  (B7) = ECS WORD COUNT.
*



 ERE=     EQ     *+400000B
          SX5    B3
          SB3    B7
          RJ     =XECRERR
          SB3    X5
          EQ     ERE=
 DCB      SPACE  4,30
**        DCB - DUMP CIRCULAR BUFFER WITH REWRITE.
*
*         IF BUFFER IS BUSY, RECALL AND RETURN.
*         IF BUFFER IS NOT BUSY, REQUEST WRITE FUNCTION AND RETURN.
*
*         ENTRY  (A2) = ADDRESS OF IN.
*                (A3) = ADDRESS OF FIRST.
*                (A4) = RETURN ADDRESS.
*                (X2) = IN.
*         EXIT   TO RETURN ADDRESS - 1.
*
*         USES   X - 1, 2, 6, 7.
*                B - 2.
*                A - 1, 6.
*
*         CALLS  CIO=, RCL=.


 DCB=     SA1    A3-B1             CHECK BUFFER STATUS
          SX6    X2                STORE IN
          LX1    59
          SA6    A2
          SX2    A3-B1
          SB2    A4-B1             CONTINUE WRITE
          NG     X1,DCB1           IF NOT BUSY
          ZR     X1,DCB1           OR BLANK FET
          RJ     =XRCL=
          JP     B2
 DCB1     REWRT  X2
          JP     B2
 WTX      SPACE  4,30
***       WTX - WRITE EXIT.
*         IF BUFFER IS BUSY, RETURN.
*         OTHERWISE, WORD COUNT OF BUFFER IS CHECKED, AND A WRITE
*         FUNCTION IS REQUESTED IF NECESSARY.
*
*         ENTRY  (A2) = ADDRESS OF IN.
*                (A3) = ADDRESS OF FIRST.
*                (A4) = RETURN ADDRESS.
*                (B3) = IN+1.
*                (B4) = OUT.
*                (B5) = LIMIT.
*                (X2) = IN
*         EXIT   TO RETURN ADDRESS.
*
*         USES   X - 1, 2, 3, 4, 6, 7.
*                B - 2.
*                A - 1, 6.
*
*         CALLS  CIO=.


 WTX=     SA1    A3-B1             CHECK BUFFER STATUS
          SX6    X2                STORE IN
          LX1    59-0
          SA6    A2
          PL     X1,WTX1           IF BUFFER BUSY

**        IF BUFFER IS NOT BUSY, CHECK SIZE OF BUFFER.
*         ISSUE WRITE IF THRESHOLD IS REACHED.

          SA3    A3                FIRST
          SX6    B4-B3             (OUT-IN+1)
          SB2    X3                (LIMIT-FIRST)
          LX3    X6,B1             2*(OUT-IN+1)
          SX7    B5-B2
          AX6    60                SIGN OF (OUT-IN+1)
          BX4    X6-X7             INVERT BUFFER IF IN+1 # OUT
          IX6    X4-X3             BUFFER SIZE - 2*(OUT-IN+1)
          NG     X6,WTX1           IF BUFFER THRESHOLD NOT REACHED
          REWRT  A3-B1

 WTX1     SB2    A4                SET RETURN ADDRESS
          SX2    A3-B1             RESET (X2)
          JP     B2                RETURN
 RWW      SPACE  4,50
***       RWW - REWRITE WORDS FROM WORKING BUFFER.
*
*         ENTRY  (X2) = ADDRESS OF FET FOR FILE.
*                (B6) = FWA WORKING BUFFER.
*                (B7) = WORD COUNT OF WORKING BUFFER.
*                IF (B7) = 0, NO TRANSFER WILL BE PERFORMED.
*         EXIT   (X2) = ADDRESS OF FET FOR FILE.
*
*         USES   ALL REGISTERS EXCEPT A0, X0, A5, X5.
*         CALLS  DCB=, WTX=.


 +        EQ     RWW1

 RWW=     EQ     *+400000B   ENTRY/EXIT
          SA4    *-1
          ZR     B7,RWW=     IF WORKING BUFFER EMPTY
          IF     -DEF,B1=1,1
          SB1    1
          SA1    X2+4        (B5) = LIMIT
          SA3    X2+B1       (X3) = FIRST
          SB7    B6+B7       (B7) = LWA+1 WORKING BUFFER
          SB5    X1

*         INITIALIZE REGISTERS FOR TRANSFER.

 RWW1     SA1    A3+2        (B4) = OUT
          SA2    A3+B1       (X2) = IN
          SB4    X1

*         TRANSFER DATA FROM WORKING BUFFER TO CIRCULAR BUFFER.

 RWW2     SB3    X2+B1       (IN+1)
          NE     B3,B5,RWW3  IF (IN+1) ' LIMIT
          SB3    X3          (IN+1) = FIRST
 RWW3     SA1    B6          NEXT WORD
          EQ     B3,B4,DCB=  DUMP CIRCULAR BUFFER IF (IN+1) = OUT
          SB6    B6+B1       ADVANCE WORKING BUFFER
          BX6    X1
          NO
          SA6    X2          STORE WORD
          SX2    B3          IN = IN+1
          NE     B6,B7,RWW2  LOOP TO END OF WORKING BUFFER
          EQ     =XWTX=      EXIT
 WLW      SPACE  4,88,
***       WLW TRANSFERS WORDS FROM AN ECS WORKING BUFFER TO AN
*         CIRCULAR BUFFER.
*
*         ENTRY  (X2) = ADDRESS OF FET FOR FILE.
*                (X0) = FWA WORKING BUFFER.
*                (B6) = WORD COUNT OF WORKING BUFFER.
*                IF (B6) = 0, NO TRANSFER WILL BE PERFORMED.
*
*         EXIT   (X2) = ADDRESS OF FET FOR FILE.
*
*         USES   ALL REGISTERS EXCEPT A5 AND X5.
*         CALLS  ERE=, DCB=, WTX=.
*



 +        EQ     WLW1

 WLW=     EQ     *+400000B   ENTRY/EXIT
          SA4    WLW=        SET RETURN ADDRESS
          ZR     B6,WLW=     IF WORKING BUFFER EMPTY
          SA1    X2+4        (B5) = LIMIT
          SB5    X1
          SA3    X2+B1       (X3) = FIRST

*         INITIALIZE REGISTERS FOR TRANSFER.

 WLW1     SA2    A3+B1       (X2) = IN
          SB7    X2+B1       (B7) = IN+1
          SA1    A2+B1       (B4) = OUT
          SB4    X1

*         TRANSFER DATA TO CIRCULAR BUFFER.

 WLW2     SB3    B4-B7       LENGTH = OUT - IN+1
          SA0    X2          CM FWA = IN
          PL     B3,WLW3     IF NO WRAP AROUND
          SB3    X2+
          SB3    B5-B3       LENGTH = LIMIT - IN
          SB2    X3
          NE     B2,B4,WLW3  IF OUT ' FIRST
          SB3    B3-1
 WLW3     LE     B3,B6,WLW4  IF ENTIRE BUFFER TO BE USED
          SB3    B6+         TRANSFER ONLY WORD COUNT
 WLW4     ZR     B3,=XDCB=   IF BUFFER FULL
          SX2    3           RETRY COUNT
          RE     B3          FILL BUFFER
          RJ     =XERE=      IF ERROR
          SB7    B7+B3       ADVANCE IN+1
          LE     B7,B5,WLW5  IF IN ' LIMIT
          SB7    X3+B1       RESET IN+1 TO FIRST+1
 WLW5     SX2    B7-B1       RESET IN
          SX6    B3
          IX0    X0+X6       ADVANCE ECS ADDRESS
          SB6    B6-B3       COUNT TRANSFERRED DATA
          NZ     B6,WLW2     LOOP TO END OF DATA
          SB3    B7+
          EQ     =XWTX=      EXIT
          END
 LOADR
          IDENT  LOADR
          SST
          LIST   F,G,X
 LOADR    TITLE  LOADR/PRELOD - CHESS OVERLAY LOADER.
 LOADR    SPACE  4
**        PRELOD/LOADR - CHESS OVERLAY LOADER
*
*         AUTHOR:
*                K. E. GORLEN
*                VOGELBACK COMPUTING CENTER
*                NORTHWESTERN UNIVERSITY
*                04/15/70
*
          SPACE  4
**        ASSEMBLY CONSTANTS
          SPACE  2
 RA2      EQU    2                 FWA PARAMETERS
 RA65     EQU    65B               LWA+1 LOAD
 RA67     EQU    67B               LDR REPLY WORD
 LODBUF   EQU    103000B           PRELOADING BUFFER SIZE
 LODBFS   EQU    40000B            PRELOAD BUFFER IF NO CHE$$99
          SPACE  2
**        COMMON BLOCKS
*CALL     TLET
*CALL     IOCOM
*CALL     MEMORY
*CALL     SQRLST
          TITLE                    PRELOD - CHESS OVERLAY PRE-LOADER
```

```
 ECERTY   MICRO  1,,
 ECERLC   MICRO  1,,
 PRELOD   SPACE  4,88
**        PRELOD - CHESS OVERLAY PRE-LOADER
*
*         CALLING SEQUENCE:
*         CALL PRELOD
*
*                PRELOD LOCATES AND PRE-LOADS ALL OVERLAYS SPECIFIED
*         IN THE OVLY TABLE IN ORDER TO DETERMINE THEIR LENGTHS AND
*         LOAD ADDRESSES. THIS INFORMATION IS THEN STORED IN THE OVLY
*         TABLE FOR LATER USE BY LOADR. ANY OVERLAYS NOT FOUND ON LOCAL
*         FILES ARE LOADED FROM THE SYSTEM LIBRARY AND COPIED TO LOCAL
*         FILES. OVERLAYS INITIALLY FOUND ON LOCAL FILES ARE LOADED AND
*         REWRITTEN WITH THE 77 TABLES STRIPPED OFF. PRELOD IS RESIDENT
*         IN THE MOVES ARRAY AND IS THEREFORE DESTROYED.
*
*         53 AND 54 TABLES ARE FIXED UP TO LOOK LIKE 50 TABLES.  SINCE
*         THE (4,0) OVERLAY IS READ AS A WORD-ADDRESSABLE FILE, ITS
*         54 TABLE IS SHRUNK TO ONE WORD.
          SPACE  4
          ORG    MOVES             ASSEMBLE PRE-LOADER INTO MOVES ARRAY

          ENTRY  PRELOD
 PRELOD   PS                       ENTRY/EXIT
          IF     DEF,OVERLAY
          SB1    1                 CONSTANT 1
          SB3    FET               FET ADDRESS
          SB2    B1+B1             CONSTANT 2
          SB4    B2+B3             FET+IN
          SA1    RA65              REQUEST FIELD LENGTH FOR PRE-LOADING
          MX2    60-17
          BX1    -X2*X1            EXTRACT ADDRESS FIELD ONLY
          SA2    A1-B1             RA+64
          SB5    X2                COUNT OF PARAMETERS
          SX2    LODBFS            SHORT BUFFER BY DEFAULT
          LE     B5,B2,PRELOD1     IF NO 3RD PARAMETER
          SA4    RA2+2
          SA3    PRELODB           STAR
          BX4    X4-X3
          NZ     X4,PRELOD1        IF NOT STAR
          SB5    -1                FLAG SYSTEM LOAD NEEDED
          SX2    LODBUF            USE LONG BUFFER
 PRELOD1  BX7    X1
          IX1    X1+X2
          SA7    B3+B1             SET FIRST = (RA+65B) = LWA+1 LOAD
          SX7    X1-1
          SA7    B4+B2             SET LIMIT = (RA+65B)+LODBUF-1
          RJ     =XMEM.            ADJUST FL FOR OVERLAY PRELOADING
          SA5    OVLY              SET POINTER TO OVERLAY TABLES
          MX0    60-18
          NG     B5,PRELOD2        IF OPENING LIBRARY ON SYSTEM
          SX7    0                 ELSE DONT PROCESS OPENINGS LIBRARY
          SA7    OV99

**        INITIALIZE FET FOR OVERLAY FILE

 PRELOD2  SA1    B3+B1             SET FIRST = IN = OUT = LWA+1 LOAD
          BX6    X1
          SA6    A1+B1
          SA6    A6+B1
          BX7    X5*X0             SET FILE NAME
          SX1    B1                SET COMPLETE BIT
          BX7    X7+X1
          SA7    B3

**        OPEN FILE

          OPEN   B3,ALTER,RECALL

**        PRELOAD OVERLAY FROM LOCAL FILE

          REWIND X2,RECALL
          READ   X2,RECALL         READ OVERLAY
          SA3    X2
          LX3    55
          PL     X3,PRELOD9        IF NO END OF RECORD
          SA3    B4                SAVE IN POINTER
          REWIND X2,RECALL
          BX6    X3                RESTORE IN POINTER
          SA6    A3
          SA1    B4+B1             OUT POINTER
          IX3    X3-X1
          ZR     X3,PRELOD3        IF LOCAL FILE EMPTY
          MX3    6
          SA2    X1                LOAD FIRST WORD IN BUFFER
          BX4    X3*X2
          BX4    X3-X4
          NZ     X4,PRELOD5        IF NOT 77-TABLE
          LX2    60-36             ELSE, SET OUT TO POINT TO FWA OF
          SX6    X2+B1             50 TABLE
          IX6    X6+X1
          SA6    B4+B1             FET+OUT
          SA1    X6                FETCH FIRST WORD OF 50-TABLE
          BX6    X1
          BX2    X3*X1             TOP 6 BITS
          LX2    6
          SX2    X2-53B
          NZ     X2,PRELOD5        IF NOT ACTUALLY A 53-TABLE
          SA6    A1+2              SQUEEZE OUT UNWANTED HEADER WORDS
          SX6    A6
          SA6    B4+B1             STORE NEW VALUE OF OUT
          EQ     PRELOD5

**        PRELOAD OVERLAY FROM SYSTEM LIBRARY

 PRELOD3  SX6    X5                EXTRACT OVERLAY LEVEL
          SA1    A5+B1             LOAD OVERLAY NAME
          SX6    X6-2R00           CONVERT LEVEL TO OCTAL
          BX7    X0*X1
          SA7    PRELODA           STORE OVERLAY NAME
          MX1    2
          LX1    43                U AND V BITS
          LX6    48
          BX6    X6+X1
          SA1    B4                FET+IN
          BX6    X6+X1             SET LOAD ADDRESS = IN
          SA6    A7+B1
          MX7    0                 CLEAR RA+67B
          SA7    RA67
          SYSTEM LDV,,PRELODA      CALL LDV
 PRELOD4  RECALL                   WAIT
          SA1    A7
          ZR     X1,PRELOD4        IF LDR NOT FINISHED
          SA1    RA65              SET IN= (RA+65B) = LWA+1 OF OVERLAY
          SX6    X1
          SA6    B4

**        SET FWA AND LWA+2 OF OVERLAY IN TABLE

 PRELOD5  SA1    B4                FET + IN
          SA2    B4+B1             FET+OUT
          IX3    X1-X2             X3 = OVERLAY LENGTH
          SA4    X2                X4 = FIRST WORD OF OVERLAY
          ZR     X3,PRELOD9        ERROR IF LENGTH ZERO
          LX4    12
          MX1    60-12
          BX5    -X1*X4            EXTRACT TABLE CODE
          LX4    60-12
          MX6    3
          LX6    60-3
          BX4    -X6*X4            PRETEND ITS A 50-TABLE
          BX6    X4
          SA6    A4                RE-WRITE TABLE HEADER FIRST WORD
          SX5    X5-5000B          VERIFY 50 TABLE
          ZR     X5,PRELOD7        IF 50 TABLE
          SX5    X5+5000B-5300B
          ZR     X5,PRELOD7        53 TABLE IS ALSO OK
          SX5    X5+5300B-5400B    CAN ALSO HANDLE 54 TABLE
          NZ     X5,PRELOD9        ERROR IF UNKNOWN TABLE
          LX4    12+12
          BX5    -X1*X4            EXTRACT LEVEL NUMBERS
          MX7    60-17
          MX6    60-18
          SA1    A4+4              FETCH ENTRY POINT WORD
          NZ     X5,PRELOD6        IF NOT (0,0) OVERLAY
          SA1    A4+10B            ENTRY ADDRESS IS 4 WORDS LATER
 PRELOD6  BX7    -X7*X1            EXTRACT ENTRY ADDRESS
          LX4    60-12-12          RE-ADJUST FIRST WORD OF TABLE
          BX4    X6*X4
          BX4    X4+X7             INSERT ENTRY ADDRESS
          BX6    X4
          SA6    A4
          SX5    X5-400B           CHECK LEVEL NUMBER
          NZ     X5,PRELOD7        IF NOT (4,0) OVERLAY
          SA6    A1                ELSE MOVE FIRST WORD TO END OF HEADER
          SX6    A6                AND ADVANCE OUT TO SHRINK TABLE
          SA6    A2
 PRELOD7  LX4    30+12
          SX5    X4                EXTRACT FWA LOAD
          IX6    X3+X5             COMPUTE LWA+2 LOAD
          SX6    X6+B1
          LX5    30
          BX6    X5+X6             STORE IN TABLE
          SA6    A5+B1

**        WRITE OVERLAY OUT ON LOCAL FILE

          WRITEF B3,RECALL
 PRELOD8  SA5    A5+B2             ADVANCE TABLE POINTER
          NZ     X5,PRELOD2        LOOP TO END OF TABLE

**        TRUNCATE FIELD LENGTH

          SA1    B3+B1             RESTORE ORIGINAL FIELD LENGTH
          SX1    X1
          RJ     MEM.
          SA1    OV10+1
          SX6    X1-1              LWA+1 FIRST OVERLAY
          SA6    MEMORY+2          FAKE ECS FWA
          SA6    MEMORY+3          FAKE ECS LWA+1
          EQ     PRELOD

**        ERROR PROCESSING

 PRELOD9  SA1    A5+B1
          SX1    X1
          NZ     X1,PRELOD8        IF ERRORS OK
          DAYFILE ERMSG,0,RECALL   *ERRORS IN PRE-LOADING.*
          ABORT

 PRELODA  BSSZ   3                 LDR PARAMETERS

 PRELODB  CON    1L*
 ERMSG    DIS    ,*ERROR IN PRE-LOADING.*
          ENDIF
          IF     -DEF,OVERLAY
          SB1    1
          SA1    RA65
          RJ     =XMEM.
          EQ     PRELOD
          ENDIF

          USE
          TITLE                    LOADR - CHESS OVERLAY LOADER
**        LOADR - CHESS OVERLAY LOADER
*
*         ENTRY  (X5) = OVERLAY LEVELS, DISPLAY CODE.
*
*         EXIT   (X6) = TRANSFER ADDRESS.
*
*         USES   A1, A2, A3, A6, A7, X0, X1, X2, X3, X6, X7, B2, AND B3.
*         CALLS  MEM., CIO=.
*



          ENTRY  LOADR
 LOADR    EQ     *+400000B
          IF     DEF,OVERLAY
          SB2    B1+B1             CONSTANT 2
          SA2    LOADRA            LAST OVERLAY LOADED
          BX6    X5-X2
          MX0    48
          ZR     X6,LOADR3         IF ALREADY LOADED
          BX6    X5
          SA6    A2                LOADRA
          SB3    OVLY              SET POINTER TO OVERLAY TABLE
          SX7    B1

**        LOCATE DESIRED OVERLAY ENTRY IN TABLE

 LOADR1   SA2    B3                LOAD TABLE ENTRY
          SB3    B2+B3             ADVANCE TABLE POINTER
          BX3    -X0*X2            EXTRACT OVERLAY LEVEL
          BX3    X5-X3             COMPARE TO DESIRED LEVEL
          NZ     X3,LOADR1         LOOP IF NO MATCH

**        INITIALIZE FET

          BX7    X7+X2             FILE NAME AND COMPLETE BIT
          SA2    A2+B1             PICK UP FWA AND LWA+2 OF OVERLAY
          SA7    FET               STORE FILE NAME
          SA1    MEMORY            LOAD LAST STORAGE REQUEST
          SX6    X2                LWA+2
```

```
          LX2    30
          SX7    X2                FWA
          SA7    A7+B1             FIRST
          SA7    A7+B1             IN
          SA7    A7+B1             OUT
          SA6    A7+B1             LIMIT
          SX7    X6-1              LWA+1
          SA7    RA65              LWA+1 LOAD

**        REQUEST STORAGE IF NECESSARY

          AX1    30                COMPARE LIMIT TO FL

 NUCC     IF     DEF,NUCC
          IX1    X1-X6
          PL     X1,LOADR2         IF OK
          BX1    X6                ELSE, REQUEST STORAGE
 NUCC     ELSE
          SX7    X6+B1
          IX1    X1-X7
          PL     X1,LOADR2         IF OK
          BX1    X7                ELSE REQUEST STORAGE
 NUCC     ENDIF

          RJ     MEM.

**        LOAD OVERLAY

 LOADR2   REWIND FET,RECALL
          READSK X2,RECALL
          SA1    X2+B1             FIRST
          SA1    X1                50 TABLE
          SX6    X1                RETURN ENTRY ADDRESS
          SA6    LOADRA+1          SAVE TRANSFER ADDRESS
          SA1    X1-1
          RJ     =XPLGECS
          SA1    LOADRA+1
          BX6    X1
          EQ     LOADR

 LOADR3   SA1    A2+B1             LAST TRANSFER ADDRESS
          BX6    X1
          EQ     LOADR             RETURN

          ENTRY  LOADRA
 LOADRA   DATA   0                 LAST OVERLAY LOADED
          DATA   0                 LAST TRANSFER ADDRESS
 FET      BSSZ   5
          ELSE
          SA1    X1                OVERLAY LEVEL
          SA2    OVLY
          MX0    48
 LOADR1   BX6    X1-X2
          BX3    -X0*X3
          ZR     X3,LOADR2
          SA2    A2+1
          EQ     LOADR1

 LOADR2   AX2    30
          EQ     LOADR

          ENTRY  LOADRA
 LOADRA   BSSZ   1
          ENDIF

          SPACE  4
          IF     DEF,OVERLAY
 OVLY     MACRO  NN,ERR
 OV!NN     VFD    48/7LCHESS!NN      FILE NAME
          VFD    12/2L!NN           LEVELS
          VFD    42/7LCHE$$NN       OVERLAY NAME
          VFD    18/ERR               = 0 IF LOADING ERRORS ARE FATAL
          ENDM
          ELSE
 OVLY     MACRO  NN
          VFD    30/=XCHE$$!NN,30/2R!NN
          ENDM
          ENDIF
          SPACE  4
 OVLY     BSS    0                 OVERLAY TABLE

 OVALL    MICRO  2,, 'OVALL'

          ECHO   1,OV=('OVALL')
          OVLY   OV,0
          OVLY   99,1
          VFD    60/0              END OF TABLE
 ERRSET   TITLE  ERRSET/ERRPRO - RPV PROCESSING.
***       ERRSET - INTERCEPT EXECUTION ERRORS.
*
*         ENTRY  (X2) = ERROR CLASS MASK:
*                     = XXX1XX = MODE ERRORS.
*                     = XXX2XX = RA+1 ERRORS.
*                     = XXX4XX = LIMIT ERRORS.
*                     = XX1XXX = OPERATOR DROPS.
*                     = XX2XXX = ECS PARITY ERRORS.
*                     = XX4XXX = CPU ABORTS.
*                     = X1XXXX = NORMAL TERMINATION.
*
*         CALLS  SYS=.
*         USES   A1, A6, X1, X2, X6.
*



          ENTRY  ERRSET
 ERRSET   EQ     *+400000B
          BX6    X6-X6
          SA6    ERRPRO
          SYSTEM RPV,RECALL,A6,X2
          EQ     ERRSET            RETURN
 ERRPRO   SPACE  4,37
***       ERRPRO - PROCESS ERRORS.
*
*         EXIT   TO LSTMOV.
*



 ERRPRO   BSSZ   17
          SB1    1

          IF     -DEF,NUCC,1
          SYSTEM DMP,RECALL,ERRPROA DUMP FL

          SA2    ERRPROA
          SA1    ERRPROB           ADDRESS OF FIRST FET
          SX7    B1
          BX6    -X7*X2            CLEAR COMPLETE BIT FOR DMP CALL
          SA6    A2
 ERRPRO1  ZR     X1,ERRPRO2        IF DONE SETTING COMPLETE BITS
          SA2    X1
          SA1    A1+B1
          ZR     X2,ERRPRO1        IF NO FILE NAME
          BX6    X7+X2
          SA6    A2                SET FET COMPLETE
          EQ     ERRPRO1

 ERRPRO2  SX5    =C* EXECUTION ERROR.*
          RJ     =XERRMSG
          SA1    RPVNO             CHECK FOR REPEATED ABORT
          NZ     X1,ERRPRO3        GIVE UP IF REPEATED ABORT
          SX6    B1                ELSE SET ABORT FLAG, WHICH WILL BE
          SA6    A1                CLEARED WHEN A COMMAND IS READ.
          SX2    7700B
          RJ     =XERRSET          ENABLE RPV AGAIN
          SA1    =XINPUT           FAKE EOR STATUS IN INPUT FILE
          SX6    20B
          BX6    X1+X6
          SA6    A1
          SA1    A1+2              SET OUT = IN TO SHOW
          BX6    X1                EMPTY INPUT BUFFER.
          SA6    A1+B1
          JUMPUP LSTMOV

 ERRPRO3  DAYFILE (=C* REPRIEVE FAILURE.*)
          JUMPUP ENDCMD            END THE PROGRAM

          IF     DEF,NUCC,2
 ERRPROA  CON    0LDUMP000         WFL STATUS WORD
          ELSE   1
 ERRPROA  VFD    12/0,18/0,18/377777B,12/0 DMP STATUS WORD

 ERRPROB  CON    =XINPUT           FETS TO SET COMPLETE
          CON    =XOUTPUT
          CON    =XGAMFILE
          CON    =XLIBRARY
          CON    =XHARDCPY
          CON    =XRELFILE
          CON    0                 END OF LIST
 CHEKFE   TITLE  CHEKFE - CHECK FE.
***       CHEKFE - CHECK FE.
*
*         ENTRY  (X0) = ECS ADDRESS.
*                (B3) = LENGTH.
*                (MEMORY+1) = FE * 1S30.
*
*         EXIT   ENOUGH ECS IS AVAILABLE.
*
*         CALLS  SYS=.
*         USES   A1, A2, A6, X2, X6, X7.
*



          ENTRY  CHEKFE
 CHEKFE   EQ     *+400000B
          SA2    MEMORY+1          ECS FIELD LENGTH
          ZR     X2,CHEKFE         IF NO ECS
          AX2    30                FE
          SX6    B3                LENGTH
          IX7    X6+X0             LWA+1
          IX6    X7-X2
          NG     X6,CHEKFE         IF ADEQUATE ECS
          SX6    4000B             ROUND UP AND ADD 1 FOR CYBER 170
          IX7    X7+X6             SINCE CANT USE LAST ECS WORD.
          AX7    11
          LX7    30+11
          SA7    A2                MEMORY+1
          BX7    X1                SAVE X1
          SYSTEM MEM,RECALL,A7,1   REQUEST ECS
          BX1    X7
          EQ     CHEKFE            RETURN
 PLGECS   TITLE  PLGECS - PLUG ECS INSTRUCTIONS.
***       PLGECS - PLUG ECS INSTRUCTIONS.
*
*         ENTRY  (X1) = ADDRESS OF FIRST ECS INSTRUCTION ADDRESS TABLE.
*                     = 0, IF NOTHING TO PLUG.
*                (MEMORY+1) ' 0, IF ECS AVAILABLE.
*                           = 0, OTHERWISE.
*                TABLE FORMAT:
*                12/CODE,48/ADDRESS
*                CODE = 2000B = LINK TO NEXT SECTION OF TABLE.
*                     = 2001B = ECS READ INSTRUCTION.
*                     = 2002B = ECS WRITE INSTRUCTION.
*                     = 0000B = END OF TABLE.
*
*         USES   X1, X2, X6, X7, A1, A2, A6, A7, AND B2.
*



          ENTRY  PLGECS
 PLGECS   EQ     *+400000B
          ZR     X1,PLGECS         IF NOTHING TO PLUG
          SA2    MEMORY+1          ECS FL
          NZ     X2,PLGECS3        IF ECS AVAILABLE
          SA2    PLGECSA
          SA1    X1                FIRST TABLE ENTRY
          BX6    X2
          SA2    A2+B1
          BX7    X2

 PLGECS1  ZR     X1,PLGECS         IF DONE
          UX1    X1,B2
          JP     PLGECS2+B2

 PLGECS2  SA1    X1
          EQ     PLGECS1

 +        SA6    X1
          SA1    A1+B1
          EQ     PLGECS1

 +        SA7    X1
          SA1    A1+B1
          EQ     PLGECS1

 PLGECS3  SA1    X1
          SA2    PLGECSB
          BX6    X2
          SA2    A2+B1
          BX7    X2
          EQ     PLGECS1

 PLGECSA  RJ     =XFAKERE
          RJ     =XFAKEWE
```

```
 PLGECSB  RE     B3
          RJ     =XECRERR
          WE     B3
          RJ     =XECWERR
 FECSUP   TITLE  FECSUP - MOVE FAKE ECS UP.
***       FECSUP - MOVE FAKE ECS UP.
*
*         ENTRY  (MEMORY+2) = CURRENT FAKE ECS FWA.
*                (MEMORY+3) = CURRENT FAKE ECS LWA+1.
*                (OV11+1) = FWA+1 NEW FAKE ECS.
*
*         EXIT   (MEMORY+2) = NEW FWA FAKE ECS.
*                (MEMORY+3) = NEW LWA+1 FAKE ECS.
*
*         USES   A1, A2, A3, A6, A7, X1, X2, X3, X4, X6, X7, B2.
*         CALLS  MEM..
*



          ENTRY  FECSUP
 FECSUP   EQ     *+400000B
          IF     DEF,DISPLAY
          IF     DEF,OVERLAY
          SA1    MEMORY+2           FWA FAKE ECS
          SA2    A1+B1              LWA+1 FAKE ECS
          SA3    OV11+1             FWA+1 NEW FAKE ECS
          SX3    X3-1               FWA NEW FAKE ECS
          IX4    X2-X1              LENGTH OF FAKE ECS
          SX6    X3
          IX7    X6+X4
          SA6    A1                 NEW FWA FAKE ECS
          SA7    A2                 NEW LWA+1 FAKE ECS
          ZR     X4,FECSUP          IF NO FAKE ECS
          SB2    -B1
          BX1    X7
          RJ     =XMEM.             REQUEST CM
 FECSUP1  SA1    X2+B2              MOVE FAKE ECS
          BX6    X1
          SA6    X7+B2
          SX4    X4+B2              DECREMENT COUNT
          SX2    X2+B2              DECREMENT ADDRESSES
          SX7    X7+B2
          NZ     X4,FECSUP1         IF NOT ALL MOVED
          ENDIF
          EQ     FECSUP             RETURN
 FECSDN   SPACE  4
***       FECSDN - MOVE FAKE ECS DOWN.
*
*         ENTRY  (MEMORY+2) = FWA FAKE ECS.
*                (MEMORY+3) = LWA+1 FAKE ECS.
*                (OV10+1) = FWA+1 NEW FAKE ECS.
*
*         EXIT   (MEMORY+2) = NEW FWA FAKE ECS.
*                (MEMORY+3) = NEW LWA+1 FAKE ECS.
*                (X1) = NEW LWA+1 FAKE ECS.
*
*         USES   A2, A3, A5, A6, X2, X3, X4, X5, X6, X7.
*



          ENTRY  FECSDN
 FECSDN   EQ     *+400000B
          IF     DEF,DISPLAY
          IF     DEF,OVERLAY
          SA5    MEMORY+2           FWA FAKE ECS
          SA3    OV10+1             NEW FWA+1 FAKE ECS
          SA2    A5+B1              LWA+1 FAKE ECS
          SX3    X3-1               NEW FWA FAKE ECS
          IX4    X2-X5              LENGTH OF FAKE ECS
          BX6    X3
          IX7    X3+X4              NEW LWA+1 FAKE ECS
          SA6    A5
          SA7    A2
          SX1    X7                 RETURN NEW LWA+1
          ZR     X4,FECSDN          IF NO FAKE ECS
          MX7    59                 -1
 FECSDN1  SA2    X5                 MOVE ECS
          BX6    X2
          SA6    X3
          SX5    X5+B1
          SX3    X3+B1
          IX4    X4+X7
          NZ     X4,FECSDN1
          ELSE   1
          SA1    MEMORY+3
          EQ     FECSDN             RETURN
 ECSOFF   SPACE  4,30
***       ECSOFF - DISABLE ECS INSTRUCTIONS.
*
*         CALLS  PLGECS, ALOCFE.
*         USES   A1, A2, A3, A6, A7, X1, X2, X3, X4, X6, X7, B2.
*



          ENTRY  ECSOFF
 ECSOFF   EQ     *+400000B
          SX6    0
          SA6    MEMORY+1           TURN OFF ECS
          SA6    PONDR              STOP PONDERING ON OPPONENTS TIME
          SX1    =XEREADER
          RJ     =XPLGECS           PLUG PRIMARY OVERLAY
          SA1    OVLY+1
          LX1    30
          SA1    X1+B1
          RJ     =XPLGECS           PLUG (X,0) OVERLAY
          SA1    MEMORY+2           FAKE ECS ADDRESS
          SX1    X1+EC.PSH+L.PSH    END OF PAWN STRUCTURE HASH TABLE
          SA2    MEMORY+3           LWA+1 FAKE ECS
          SX4    X1-L.PSH           FWA PAWN STRUCTURE HASH TABLE
          SX1    X1+L.MSI           LWA+1 LIBRARY INDEX
          IX6    X1-X2
          NG     X6,ECSOFF1         IF ENOUGH AVAILABLE
          RJ     =XALOCFE           ALLOCATE FAKE ECS
 ECSOFF1  SB2    L.PSH-1
          BX7    -X7+X7             -0
 ECSOFF2  SA7    X4+B2              CLEAR PAWN TABLE
          SB2    B2-B1
          PL     B2,ECSOFF2         IF NOT DONE WITH TABLE

 MAILBOX  IF     DEF,MAILBOX
          SA1    MEMORY+2
          BX6    X6-X6
          SA6    X1                 CLEAR MAILBOX
          ERRNZ  EC.MAIL
 MAILBOX  ENDIF

          REWIND =XLIBRARY,RECALL
          SA1    MEMORY+2
          READW  X2,X1+EC.MSI,L.MSI  READ INDEX
          EQ     ECSOFF             RETURN

 FAKERE   TITLE  FAKERE - FAKE ECS READ.
***       FAKERE - FAKE ECS READ.
*
*         ENTRY  (A0) = ADDRESS TO PUT DATA.
*                (X0) + (MEMORY+2) = ADDRESS OF DATA.
*                (B3) = NUMBER OF WORDS TO TRANSFER.
*                (MEMORY+2) = FAKE ECS ADDRESS.
*
*         USES   A1, A2, A3, A6, A7, X6, X7.
*         CALLS  NOM=.
*



 FAKERE1  BX6    X1
          BX7    X2
          SA6    FAKEREA            SAVE REGISTERS
          SA7    A6+B1
          SX6    B2
          SA6    A7+B1
          BX6    X3
          SA6    A6+B1
          SA1    MEMORY+2           FAKE ECS ADDRESS
          IX1    X1+X0              SOURCE
          SX2    A0                 DESTINATION
          SX3    B3                 WORD COUNT
          RJ     =XNOM=             NON-OVERLAPPED MOVE
          SA1    FAKEREA            RESTORE REGISTERS
          SA2    FAKEREA+1
          SA3    FAKEREA+2
          SB2    X3
          SA3    A3+B1

 FAKERE   EQ     *+400000B
          NE     B3,B1,FAKERE1      IF NOT ONE WORD

*         MOVE ONE WORD.

          BX7    X1                 SAVE X1
          SA1    MEMORY+2
          IX1    X1+X0              FORM FAKE ECS ADDRESS
          SA1    X1
          BX6    X1
          SA6    A0+0
          BX1    X7                 RESTORE X1
          EQ     FAKERE

 FAKEREA  BSS    4                  SAVED X1, X2, B2, AND X3.
 FAKEWE   SPACE  4,40
***       FAKEWE - FAKE ECS WRITE.
*
*         ENTRY  (A0) = ADDRESS OF DATA.
*                (X0) + (MEMORY+2) = ADDRESS TO PUT DATA.
*                (B3) = NUMBER OF WORDS TO TRANSFER.
*                (MEMORY+3) = FAKE ECS LWA+1.
*                (MEMORY+2) = FAKE ECS ADDRESS.
*
*         USES   A1, A2, A3, A6, A7, X6, X7.
*         CALLS  ALOCFE, NOM=.
*



 FAKEWE1  BX6    X1
          BX7    X2
          SA6    FAKEREA            SAVE REGISTERS
          SA7    A6+B1
          SX6    B2
          SA6    A7+B1
          BX6    X3
          SA3    MEMORY+3           CURRENT LWA+1 FAKE ECS
          SA6    A6+B1
          SA2    MEMORY+2           CURRENT FWA FAKE ECS
          IX2    X2+X0              FWA TO PUT DATA
          SX1    X2+B3              LWA+1 TO PUT DATA
          IX6    X1-X3
          NG     X6,FAKEWE2         IF ADEQUATE CM
          RJ     =XALOCFE           ALLOCATE FAKE ECS
 FAKEWE2  SX1    A0                 FWA DATA
          SX3    B3                 LENGTH
          RJ     =XNOM=             NON-OVERLAPPED MOVE.
          SA1    FAKEREA            RESTORE REGISTERS
          SA2    FAKEREA+1
          SA3    FAKEREA+2
          SB2    X3
          SA3    A3+B1

 FAKEWE   EQ     *+400000B
          NE     B3,B1,FAKEWE1      IF NOT ONE WORD

*         MOVE ONE WORD.

          BX7    X1                 SAVE X1
          SA1    A0                 WORD TO MOVE
          BX6    X1
          SA1    MEMORY+2
          IX1    X1+X0
          SA6    X1
          BX1    X7
          EQ     FAKEWE
 ECWERR   TITLE  ECWERR - ECS WRITE ERROR RECOVERY.
***       ECWERR - ECS WRITE ERROR RECOVERY.
*
*         ENTRY  (X0) = ECS ADDRESS.
*
*         EXIT   TO REREAD.
*
*         CALLS  RDRCOD, ERRMSG.
*



          ENTRY  ECWERR
 ECWERR   EQ     *+400000B
          BX1    X0
          RJ     =XRDRCOD
          SA6    ECWERRA+1
          SA1    ECWERR
          AX1    30
          SX1    X1
          RJ     =XRDRCOD
          SA6    A6+B1
          SX5    ECWERRA
          RJ     =XERRMSG
          SX5    ECWERRB
          RJ     =XERRMSG
```

```
          RJ     =XREREAD

 ECWERRA  DATA   30C WE ERROR
 ECWERRB  DATA   C* TYPE: EXPLAIN,WERR.*
 ALOCFE   TITLE  ALOCFE - ALLOCATE FAKE ECS.
***       ALOCFE - ALLOCATE FAKE ECS.
*
*         ENTRY  (X1) = NEW LWA+1 FAKE ECS.
*                (SWICH,S.DIS) = DISPLAY MODE SWITCH.
*
*         EXIT   (MEMORY+3) = FAKE ECS LWA+1.
*
*         CALLS  DUDECS, MEM..
*         USES   A1, A2, A3, A6, X1, X3, X6, B2.
*


 ALOCFE1  RJ     =XMEM.            GET CM

          ENTRY  ALOCFE
 ALOCFE   EQ     *+400000B
          SA3    SWICH
          MX6    -6
          IX6    X1-X6             ROUND UP TO NEXT 100
          AX6    6
          LX6    6
          LX3    59-S.DIS
          SA6    MEMORY+3          SET NEW LWA+1
          PL     X3,ALOCFE1        IF NO DISPLAYS
          BX6    X2                SAVE REGISTERS
          SA6    ALOCFEA
          SX6    B4
          SA6    A6+B1
          SX6    B7
          SA6    A6+1
          RJ     =XDUDECS          GET CM
          SA2    ALOCFEA           RESTORE REGISTERS
          SA3    A2+B1
          SB4    X3
          SA3    A3+B1
          SB7    X3
          EQ     ALOCFE            RETURN

 ALOCFEA  BSS    3                 SAVED X2, B4, B7.
 ECRERR   TITLE  ECRERR - ECS READ ERROR RECOVERY.
***       ECRERR - ECS READ ERROR RECOVERY.
*
*         ENTRY  (X0) = ECS ADDRESS.
*                (A0) = CM ADDRESS.
*                (B3) = WORD COUNT.
*
*         EXIT   TO CALLER IF RETRY SUCCESSFUL.
*                TO REREAD, OTHERWISE.
*         CALLS  RDRCOD, ERRMSG.
*


          ENTRY  ECRERR
 ECRERR   DATA   0                 RECOVERED ERROR FLAG
          RE     B3
          EQ     ECRERR1
          EQ     ECRERR
 ECRERR1  RE     B3
          EQ     ECRERR2
          EQ     ECRERR
 ECRERR2  RE     B3
          EQ     ECRERR3
          EQ     ECRERR
 ECRERR3  BX1    X0
          RJ     =XRDRCOD
          SA6    ECRERRA+1
          SA1    ECRERR
          AX1    30
          SX1    X1
          RJ     =XRDRCOD
          SA6    A6+B1
          SX5    ECRERRA
          RJ     =XERRMSG
          SX5    ECRERRB
          RJ     =XERRMSG
          RJ     =XREREAD          READ NEW COMMAND

 ECRERRA  DATA   30C RE ERROR
 ECRERRB  DATA   C* TYPE: EXPLAIN,RERR.*
          SPACE  4
          LIST   G
*CALL     COMCNOM
          END
DUDLINK
          IDENT  DUDLINK
          SST
          TITLE  DUDLINK - LINKAGE TO (1,1) OVERLAY
          SPACE  4
**        DUDLINK - LINKAGE TO (1,1) OVERLAY
*
*                DUDLINK CONTAINS THE ENTRY POINTS TO ALL ROUTINES IN
*         THE (1,1) OVERLAY.  WHEN THIS OVERLAY IS LOADED BY A CALL TO
*         DUDLOAD, A ROUTINE IN THE (1,1) OVERLAY IS CALLED TO LINK
*         THE REMAINING ENTRY POINTS TO THEIR RESPECTIVE BLOCKS OF CODE.
*         A CALL TO DUDDROP DE-LINKS THE ENTRY POINTS AND TRUNCATES THE
*         (1,1) OVERLAY OUT OF THE FIELD LENGTH. A DE-LINKED ENTRY POINT
*         SIMPLY RETURNS TO THE CALLING PROGRAM. ADDITIONAL ENTRY POINTS
*         ARE ADDED BY INSERTING A CALL TO THE LINK MACRO IN THE LINK
*         COMMON DECK.
          EJECT
*CALL     IOCOM
**        LINK MACRO FOR (0,0) OVERLAY
          SPACE  2
          MACRO  LINK,LABL
          ENTRY  LABL
 LABL     PS
          EQ     LABL
          ENDM
          SPACE  4
**        LINKAGE TABLE FOR (0,0) OVERLAY
          SPACE  2
*CALL     LINK
          SPACE  4
**        DUDLOAD - LOAD (1,1) OVERLAY


          ENTRY  DUDLOAD
 DUDLOAD  PS
          SA1    DUDLOADA
          NZ     X1,DUDLOAD1       IF OVERLAY PRESENT
          RJ     =XFECSUP          MOVE FAKE ECS UP
          SX5    2R11

 IMLAC    IF     DEF,IMLAC
          SA1    SWICH
          LX1    0-S.IML
          MX6    59
          BX6    -X6*X1
          IX5    X6+X5
 IMLAC    ENDIF
          RJ     =XLOADR
          SA6    DUDLOADA          SAVE ENTRY ADDRESS
          BX1    X6
 DUDLOAD1 SB2    X1
          JP     B2                TRANSFER TO ENTRY POINT

 DUDLOADA VFD    60/0              (1,1) OVERLAY ENTRY ADDRESS
          SPACE  4
**        DUDDRP$ - COMPLETE DROP OF DISPLAY


          ENTRY  DUDDRP$
 DUDDRP$  RJ     =XFECSDN          MOVE FAKE ECS DOWN
          RJ     =XMEM.            TRUNCATE FIELD LENGTH
          MX6    0
          SA6    DUDLOADA          CLEAR ENTRY ADDRESS
          SA6    =XLOADRA
          EQ     DUDDROP
          END
JUMPUP
          IDENT  JUMPUP
***       JUMPUP - LOAD OVERLAY AND CALL SUBROUTINE.
*
*         ENTRY  (X5) = 30/UNUSED,12/OVERLAY,18/ENTRY NUMBER
*
*         USES   ALL REGISTERS.
*         CALLS  LOADR.
*


          ENTRY  JUMPUP
 JUMPUP   EQ     *+400000B
          SX6    X5                EXTRACT ENTRY NUMBER
          SA6    JUMPUPA
          AX5    18
          RJ     =XLOADR           LOAD OVERLAY (PARAMETER IN X1)
          SA2    JUMPUPA           ENTRY NUMBER
          IX6    X2+X6             ENTRY ADDRESS
          SB2    X6
          JP     B2                TO TRANSFER TABLE

 JUMPUPA  BSS    1
          END
READER
          IDENT  READER
          SST
 READER   TITLE  READER - READ INPUT FROM ANYWHERE.
          LIST   F,X
*CALL     IOCOM
*CALL     TLET
*CALL     MEMORY
*CALL     RELY
*CALL     LET
*CALL     SQRLST
          SPACE  4
 EREADER  ECERTB EMASSIO
 RA37     EQU    37B               ONL REQUEST WORD
 RA40     EQU    40B               ONL DATA AREA FWA
 RA50     EQU    50B               ONL DATA AREA LWA+1
 RA70     EQU    70B               CFO AREA
 RA77     EQU    77B               END OF CFO AREA
 RA100    EQU    100B
 READER   SPACE  4,88
***       READER - READ INPUT FROM ANYWHERE.
*
*         ENTRY  (X6) = QUESTION ORDINAL.
*                O.EMOV = ENTER MOVE. (OR WHATEVER IS IN MOVMS)
*                O.EDBG = ENTER DEBUG COMMAND. (OR WHATEVER IS IN DBGMS)
*                O.TGIO = THE GAME IS OVER. (OR WHATEVER IS IN ENDMS)
*                O.HMTU = HOW MUCH TIME HAS BEEN USED.
*                -INFIN = PAGE WAIT.
*
*         EXIT   (ILINE) = INPUT LINE.
*                TO COMMAND PROCESSOR, IF COMMAND.
*
*         USES   ALL REGISTERS.
*         CALLS  MANY ROUTINES.
*


          ENTRY  READER
 READER   EQ     *+400000B
          RJ     =XINLNCT          INITIALIZE LINE COUNT
          NG     X6,READER7        IF PAGE WAIT
          RJ     =XPROAUT          PROCESS AUTO-ALTER
 READER1  SA1    RELSW
          ZR     X1,READER2        IF NOT RELIABILITY MODE
          RECALL =XINPUT
          RECALL =XGAMFILE
          RECALL =XOUTPUT
          RECALL =XLIBRARY
          RECALL =XHARDCPY
          REWIND RELFILE,RECALL
          SA1    RELFILE+4         LIMIT
          SX6    X1-1
          SA6    RELFILE+2         SET IN
          REWRTR X2,RECALL
 READER2  RJ     =XRDRMIL          MOVE INPUT LINE
          ZR     B6,READER3        IF NOTHING MOVED
          SA1    ILINE
          SB2    54
          MX0    54
          BX6    X6-X6             SET OK TO REPRIEVE AGAIN
          SA6    RPVNO
          IF     DEF,NUCC,1
          RJ     =XRDRSPC          PROCESS SPECIAL CASES
          RJ     =XRDRGNT          GET NEXT TOKEN
          ID     X6,READER3        IF END OF CARD
          SA2    READERB
          RJ     =XRDRSKT          SEARCH KEYWORD TABLE
                                   (RETURNS ONLY IF NOT FOUND)
          SA1    READERA
          SA5    READERD+X1
          MX6    1
          SA6    LINES             DISABLE PAGING
          RJ     =XJUMPUP          RETURN

 READER3  WRITE  =XOUTPUT          FLUSH BUFFER
          SA1    SWICH
          LX1    59-S.DIS
```

```
          PL     X1,READER4        IF NOT DISPLAY
          RJ     =XDUDCLR
          SA1    READERA
          LX1    2
          SX1    X1+MOVMS          MESSAGE
          RJ     =XDUDMSG          DISPLAY OUTPUT LINE
          SA1    READERA
          RJ     =XDUDSET          SET VALID SYNTAX TABLES
          SX1    READERC
          RJ     =XDUDREAD         READ DISPLAY
          EQ     READER8

READER4   LX1    S.DIS-S.DSD
          PL     X1,READER5        IF NOT DSD MODE
          SA1    READERA
          LX1    2
          DAYFILE X1+MOVMS,100B,RECALL
          RJ     =XRDRCFO
          ZR     X6,READER12       IF EXIT TO PONDER
          EQ     READER8

READER5   BSS    0

ONELINE   IF     DEF,ONELINE
          LX1    S.DSD-S.ONL
          PL     X1,READER6        IF NOT ONE-LINE MODE
          SX1    READERC
          RJ     =XONLINP
          EQ     READER8
ONELINE   ENDIF

READER6   BSS    0

NUCC      IF     DEF,NUCC
          SA1    MTIDS
          ZR     X1,READER7        IF NOT MULTI-TERMINAL
          NG     X1,READER7        IF MASTER UNKNOWN
          SA2    READERE           PROMPT
          MX6    12
          BX6    X6*X1             EXTRACT MASTER ID
          BX6    X6+X2             MERGE PROMPT
          SA6    READERC
          WRITEC =XOUTPUT,A6       PROMPT USER
NUCC      ENDIF

          ENTRY  RDMORE            ENTER HERE AFTER SLAVE TYPE-IN
RDMORE    BSS    0
READER7   SA1    READERA           MOVE ORDINAL
          RJ     =XRDRCHK          CHECK IF SHOULD READ
          ZR     X6,READER12       IF NOTHING TO READ
          SA1    READERF
          RJ     =XRDLINE          READ A LINE
          SA1    LINEPP            INITIALIZE LINE COUNT
          BX6    X1
          SA6    LINES

NUCC      IF     DEF,NUCC
          SA1    MTIDS
          ZR     X1,READER8        IF NOT MULTI-TERMINAL
          SA4    READERF
          SA4    X4
          MX3    12
          BX6    X1-X4
          BX6    X3*X6
          NZ     X6,READER11       IF NOT MASTER TERMINAL
          SX6    2R
          BX4    -X3*X4            CLEAR ID
          LX6    48
          BX6    X6+X4
          SA6    A4
          NZ     X4,READER8        IF NOT EMPTY LINE
          BX6    X6-X6             CLEAR INPUT LINE
          SA6    A6
NUCC      ENDIF

READER8   SA1    READERF
          SA3    READERA
          SX6    READERC
          SX7    54
          SA6    A1                READERF
          LX7    18
          BX7    X6+X7
          SA1    X1
          SA2    A1+7
          MX6    12
          BX6    X6*X2
          SA6    A2                CLEAR PAST COLUMN 72
          MX6    1
          BX6    -X6*X3            CLEAR PAGE WAIT FLAG
          SA6    A3
          NG     X3,READER9        IF PAGE WAIT
          SA7    READERG           SET INPUT LINE POINTER
          EQ     READER2           PROCESS LINE

READER9   ZR     X1,READER         IF EMPTY LINE, RETURN
          SA7    READERG           SET INPUT LINE POINTER
          SB3    A1-B1             LWA TO TRANSFER
READER10  BX6    X1                MOVE 8 WORDS
          SA6    A1-8              BACK 8
          SA1    A1+B1
          SB4    A6
          LT     B4,B3,READER10
          EQ     READER2           PROCESS COMMANDS

          IF     DEF,NUCC,1
READER11  JUMPUP PROSLV            PROCESS SLAVE

READER12  JUMPUP PONDER            BEGIN/RESUME PONDERING

READERA   BSS    1                 QUESTION ORDINAL

READERB   CMND   EN,ENDCMD
          CMND   DR,ENDCMD
          CMND   SW,SWICMD
          CMND   PR,PRICMD,PON
          CMND   CR,CRECMD
          CMND   PU,PURCMD
          CMND   BO,BOACMD
          CMND   LE,LETCMD
          CMND   PA,PARCMD,PON
          CMND   KI,KILCMD
          CMND   GO,GONCMD
          CMND   NU,NOMOVE
          CMND   IN,INITAL
          CMND   RES,INITAL
          CMND   SA,SAVCMD
          CMND   SE,SETCMD
          CMND   PS,PSLCMD,PON
          CMND   DI,DISCMD,PON
          CMND   CH,CHACMD,PON
          CMND   REW,REWCMD,PON
          CMND   RET,RETCMD,PON
          CMND   US,USECMD
          CMND   STR,STRCMD
          IF     DEF,NUCC,1
          CMND   X,XEQCMD
          CMND   PV,PVACMD,PON
          CMND   STA,STACMD
          CMND   WH,WHACMD,PON
          CMND   PI,PINCMD,PON
          CMND   REL,RELCMD
          CMND   REI,REICMD
          CMND   AL,ALTCMD
          CMND   HA,HDCCMD,PON
          CMND   EC,ECSCMD
          CMND   TO,TOUCMD
          IF     DEF,NUCC,1
          CMND   MU,MULCMD,PON
          CMND   M,MSGCMD,PON
          CMND   NA,NAMCMD,PON
          CMND   CO,COMCMD
          CMND   CL,CLOCMD
          CMND   ID,PIDCMD,PON
          CMND   PW,PPWCMD,PON
          CMND   BK,BKPCMD
          CMND   AF,AFNCMD,PON
          CMND   BL,BLICMD
          CMND   SP,SPECMD
          IF     DEF,OVERLAY,2
          CMND   HE,HELCMD,PON
          CMND   EX,EXPCMD,PON
          IF     -DEF,KRONOS,1
          IF     DEF,NUCC,2
          CMND   DUMP,DUMCMD
          CMND   LOAD,LOACMD
          DATA   0                 END OF TABLE

READERC   BSSZ   8                 INPUT LINE BUFFER
          BSSZ   8                 PAGE WAIT INPUT LINE BUFFER

READERD   BSS    0
          ECHO   1,RETURN=(YUMOVE,DBUGGR,FINSHD,TIMCTL)
          VFD    42/V.RETURN,18/E.RETURN

NUCC      IF     DEF,NUCC
READERE   VFD    12/0              ID
          VFD    12/2R?            PROMPT
          VFD    12/0057B          SUPPRESS CRLF
          VFD    12/0000B          CRLF
          VFD    12/0              IGNORED
NUCC      ENDIF

READERF   BSS    1                 INPUT LINE BUFFER POINTER

READERG   VFD    42/54,18/READERC  INPUT LINE POINTER
REREAD    SPACE  4,12
***       REREAD - RE-ENTER READER.
*
*         EXIT   TO READER.
*


          ENTRY  REREAD
REREAD    EQ     *+400000B
          EQ     READER1
PROAUT    SPACE  4,25
***       PROAUT - PROCESS AUTO ALTER.
*
*         ENTRY  (X6) = QUESTION ORDINAL.
*
*         EXIT   TO ALTCMD, IF AUTO ALTER SELECTED.
*


PROAUT    EQ     *+400000B
          SX7    X6-O.TGIO
          PL     X7,PROAUT         IF NOT O.EMOV OR O.EDBG
          SA1    SWICH
          SB2    X6
          LX1    59-S.MVA
          ERRNZ  S.MVA-S.DBA-1
          LX1    B2
          PL     X1,PROAUT         IF NOT SELECTED
          BX6    X6-X6
          SA6    ILINE             USE DEFAULT ALTER FILE NAME
          JUMPUP ALTCMD            FAKE ALTER COMMAND
INLNCT    SPACE  4,20
***       INLNCT - INITIALIZE LINE COUNT.
*
*         ENTRY  (X6) = QUESTION ORDINAL.
*                (LINEPP) = LINES PER PAGE.
*
*         EXIT   (X6) = QUESTION ORDINAL.
*                (READERA) = QUESTION ORDINAL.
*                (LINES) = (LINEPP).
*                (READERF) = READERC, IF ORDINARY QUESTION.
*                          = READERC+8, IF PAGE WAIT.
*


INLNCT1   SA6    READERA           SAVE QUESTION ORDINAL
          SX7    X7+READERC+8
          SA7    READERF
          SA1    LINEPP            LINES PER PAGE
          BX7    X1
          SA7    LINES

INLNCT    EQ     *+400000B
          SX7    -8
          PL     X6,INLNCT1        IF NOT PAGE WAIT
          SA1    READERA
          BX6    X1+X6
          BX7    X7-X7
          EQ     INLNCT1
RDRCFO    SPACE  4,88
***       RDRCFO - READ COMMENT FROM OPERATOR.
*
*         EXIT   (READERC) = INPUT LINE.
*                (X6) = 0, IF EXIT TO PONDER DESIRED.
*


RDRCFO4   SX6    B1                RETURN READY
          BX7    X7-X7             CLEAR ALREADY READING
```

```
          SA7    RDRCFOA

 RDRCFO   EQ     *+400000B
          SA1    RDRCFOA
          NZ     X1,RDRCFO1        IF ALREADY READING
          SA1    B0
          SX6    10000B
          BX6    X6+X1
          BX7    X7-X7
          SA7    RA70              CLEAR INPUT AREA
          SA6    B0                SET PAUSE BIT
          SA6    RDRCFOA           SET READING
 RDRCFO1  SA1    READERA
          RJ     =XRDRCHK
          ZR     X6,RDRCFO         IF NOTHING TO BE READ
 RDRCFO2  RECALL
          SA2    RA70
          ZR     X2,RDRCFO2        IF NOTHING ENTERED
          MX0    48
          BX6    X2
          SA6    READERC           STORE FIRST WORD
          SA7    RA77              CLEAR LWA+1
 RDRCFO3  BX3    -X0*X2
          ZR     X3,RDRCFO4        IF DONE
          SA2    A2+B1
          BX6    X2
          SA6    A6+B1
          EQ     RDRCFO3           MOVE NEXT WORD

 RDRCFOA  CON    0                 ALREADY READING FLAG
 RDRCHK   SPACE  4,20
***       RDRCHK - CHECK IF SHOULD READ.
*
*         ENTRY  (X1) = QUESTION ORDINAL.
*
*         EXIT   (X6) = 0, IF EXIT TO PONDER DESIRED.
*                     ' 0, IF DATA IN BUFFER TO BE PROCESSED, OR NOT
*                          PONDERING FOR SOME REASON.
*
*         CALLS  READIF.
*


 RDRCHK   EQ     *+400000B
          SX6    X1-O.EMOV
          NZ     X6,RDRCHK         IF NOT MOVE QUESTION
          SA1    SWICH
          SA2    PONDR
          LX1    59-S.PON
          SX6    B1
          PL     X1,RDRCHK         IF NOT PONDERING
          ZR     X2,RDRCHK         IF NO PONDER MOVE
          LX2    59-S.THF
          NG     X2,RDRCHK         IF ALREADY DONE
          RJ     =XREADIF          CHECK FOR DATA ANYWHERE IN SIGHT
          EQ     RDRCHK            RETURN
 READIF   SPACE  4,30
***       READIF - CHECK IF DATA AVAILABLE.
*
*         EXIT   (X6) = 0, IF NO DATA IN INPUT BUFFER.
*


 READIF3  SA1    RA70
          BX6    X1                RETURN ZERO IF NO DATA

          ENTRY  READIF
 READIF   EQ     *+400000B
          SA1    SWICH
          LX1    59-S.DSD
          NG     X1,READIF3        IF DSD MODE
 READIF1  SA1    =XINPUT
          SA2    A1+B1             FIRST
          SA3    A2+B1             IN
          SA4    A3+B1             OUT
          IX6    X3-X4
 NUCC     IF     DEF,NUCC
          NZ     X6,READIF         IF DATA AVAILABLE
          LX1    59-0
          SX6    B1
          NG     X1,READIF2        IF NOT BUSY
          RECALL A1
          EQ     READIF1

 READIF2  PL     X1,READIF         IF NOT ALLOCATABLE
          READIF A1,RECALL
          SA3    A3                RELOAD IN
          IX6    X3-X4
 NUCC     ENDIF
 KRONOS   IF     DEF,KRONOS
          NZ     X6,READIF
          SA1    CPESP             INTERRUPT FLAG
          BX6    X1
          BX7    X7-X7
          ZR     X1,READIF2        IF NOT TIME TO READ
          SA7    A1                CLEAR READ FLAG
          READ   =XINPUT
          SX6    1
 READIF2  BSS    0
 KRONOS   ENDIF
 MAILBOX  IF     DEF,MAILBOX
          NZ     X6,READIF
          SA0    OLINE
          SB3    8
          SX0    EC.MAIL
          RE     B3                READ MAILBOX
          RJ     =XECRERR          IF ERRORS
          SA1    A0
          BX6    X1
          ZR     X6,READIF         IF NOTHING READ
          RECALL =XINPUT
          BX6    X6-X6             SET END OF LINE
          SA6    OLINE+8
          WRITEC X2,A0             MOVE LINE TO INPUT BUFFER
          BX7    X7-X7
          SA7    A0
          SB3    B1
          WE     B3                ACKNOWLEDGE INPUT
          RJ     =XECWERR          IF ERRORS
          SX6    1                 RETURN DATA AVAILABLE
 MAILBOX  ENDIF
          EQ     READIF
 ONLINP   SPACE  4,80
***       ONLINP - READ REMOTE TERMINAL.
 ONELINE  IF     DEF,ONELINE
*
*         ENTRY  (X1) = ADDRESS TO PLACE MESSAGE.
*


 ONLINP   EQ     *+400000B
          SB2    RA40
          SB3    RA50
          SB4    RA70
          SB5    RA100
          BX7    X7-X7
          SX5    X1                SAVE PARAMETER
          SX6    10000B            PAUSE BIT
 ONLINP1  SA7    B2
          SB2    B2+B1
          LE     B2,B3,ONLINP1     CLEAR INPUT AREA
 ONLINP2  SA7    B4
          SB4    B4+B1
          LT     B4,B5,ONLINP2     CLEAR CFO INPUT AREA
          SB2    RA40
          SA2    B0
          BX6    X6+X2             SET PAUSE BIT
          SA6    B0
          MX6    1
          LX6    -11
          SA6    B2                OUTPUT NULL LINE
          SX6    6631B             SET REQUEST
          SA6    B2-B1
          DAYFILE ONLINPA,100B,RECALL
 ONLINP3  RECALL
          SA2    B2-B1             RA37
          ZR     X2,ONLINP5        IF REMOTE INPUT
          SA2    B0
          LX2    59-12
          NG     X2,ONLINP3        IF NO INPUT
          SA1    RA70
 ONLINP4  BX6    X1
          SA6    X5
          SX5    X5+1
          SA1    A1+B1
          SB2    A1
          LT     B2,B5,ONLINP4     MOVE CFO TYPE-IN
          EQ     ONLINP

 ONLINP5  SA2    B2
          MX0    48
          SX1    B1
          BX7    X7-X7
          MX3    1
 ONLINP6  LX2    12
          BX6    -X0*X2
          ZR     X6,ONLINP7        IF END OF LINE
          BX4    X6-X1
          ZR     X4,ONLINP7        IF END OF LINE
          LX7    12
          BX7    X7+X6
          LX3    12
          PL     X3,ONLINP6        LOOP THROUGH WORD
          SA7    X5
          SX5    X5+B1
          SA2    A2+B1
          BX7    X7-X7
          EQ     ONLINP6           PROCESS NEXT WORD

 ONLINP7  LX7    12
          LX3    12
          PL     X3,ONLINP7
          SA7    X5
          SA2    B0
          SX6    10000B            CLEAR PAUSE BIT
          BX6    -X6*X2
          SA6    B0
          EQ     ONLINP            RETURN

 ONLINPA  DATA   C* WAIT FOR REMOTE INPUT.*
 ONLOUT   SPACE  4,35
***       ONLOUT - WRITE TO REMOTE TERMINAL.
*
*         ENTRY  (X5) = ADDRESS OF MESSAGE.
*


          ENTRY  ONLOUT
 ONLOUT   EQ     *+400000B
          SA1    SWICH
          LX1    59-S.ONL
          PL     X1,ONLOUT         IF NOT ONL MODE
          SA1    X5                FIRST WORD OF MESSAGE
          SB5    RA40
          MX0    48
 ONLOUT1  BX4    -X0*X1
          ZR     X4,ONLOUT2        IF END OF LINE
          BX6    X1
          SA6    B5
          SB5    B5+B1
          SA1    A1+B1
          EQ     ONLOUT1

 ONLOUT2  RJ     =XRDRSFN          SPACE FILL
          BX6    X6*X0
          SX7    B1
          BX6    X6+X7
          SA6    B5
          SX7    6031B             WRITE REQUEST
          RJ     =XONLWAT          WAIT FOR COMPLETE
          EQ     ONLOUT            RETURN
 ONLWAT   SPACE  4,35
***       ONLWAT - WAIT FOR ONL RESPONSE.
*
*         ENTRY  (X7) = REQUEST CODE.
*


 ONLWAT   EQ     *+400000B
          SA1    B0
          SX2    10000B
          BX6    X1+X2             SET PAUSE BIT
          SA6    B0
          SA7    RA37              SET REQUEST
          DAYFILE ONLWATA,100B,RECALL
 ONLWAT1  RECALL                   WAIT
          SA1    RA37
          ZR     X1,ONLWAT2        IF DONE
          SA1    B0
          LX1    59-12
          NG     X1,ONLWAT1        IF NO OPERATOR GO
          SA1    SWICH
          SX7    B1
          LX7    S.ONL
```

```
          BX6    -X7*X1           TURN OFF ONL
          LX7    S.DSD-S.ONL
          BX6    X6+X7            SET DSD MODE
          SA6    SWICH
          EQ     ONLWAT           RETURN

 ONLWAT2  SA1    B0
          SX7    10000B           CLEAR PAUSE BIT
          BX6    -X7*X1
          SA6    B0               CLEAR PAUSE BIT
          EQ     ONLWAT           RETURN

 ONLWATA  DATA   C* WAIT FOR REMOTE OUTPUT.*

 ONLCLR   SPACE  4,20
***       ONLCLR - CLEAR 200 UT SCREEN.
*



          ENTRY  ONLCLR
 ONLCLR   EQ     *+400000B
          SA1    SWICH
          LX1    59-S.ONL
          PL     X1,ONLCLR        IF NOT IN ONL MODE
          SX7    6051B
          MX6    1
          LX6    -11
          SA6    RA40             NULL LINE
          RJ     =XONLWAT         ISSUE REQUEST
          EQ     ONLCLR           RETURN

 ONELINE  ENDIF
 RDRGNT   SPACE  4,88
***       RDRGNT - GET NEXT TOKEN.
*
*         ENTRY  (A1) = ADDRESS OF CURRENT WORD IN BUFFER.
*                (X1) = ((A1))
*                (B2) = SHIFT COUNT FOR NEXT CHARACTER.
*                (X0) = 7777 7777 7777 7777 7700
*
*         EXIT   (X6) = INDEFINATE, IF TERMINATOR.
*                (X6) = 0, OTHERWISE.
*                (X7) = TOKEN, LEFT JUSTIFIED.
*                A1, X1, AND B2 ARE UPDATED.
*
*         USES   B4.
*



          ENTRY  RDRGNT
 RDRGNT   EQ     *+400000B
          SB4    54
          BX7    X7-X7
 RDRGNT1  AX6    X1,B2
          BX6    -X0*X6           EXTRACT CHARACTER
          SB2    B2-6
          PL     B2,RDRGNT2       IF NOT NEW INPUT WORD
          SB2    54
          SA1    A1+B1
 RDRGNT2  ZR     X6,RDRGNT5       IF ZERO CHARACTER
          SX6    X6-1R;
          ZR     X6,RDRGNT5       IF SEMI-COLON
          SX6    X6-1R*+1R;
          ZR     X6,RDRGNT3       IF STAR
          SX6    X6-1R++1R*
          NG     X6,RDRGNT4       IF NOT SEPARATOR

*         SEPARATOR.

          ZR     X7,RDRGNT1       IF NOTHING ASSEMBLED YET
          EQ     RDRGNT6          RETURN

 RDRGNT3  SX6    1R*-1R+          RESTORE STAR
 RDRGNT4  SX6    X6+1R+           RESTORE CHARACTER
          LX6    B4
          BX7    X6+X7            ASSEMBLE TOKEN
          SB4    B4-6
          PL     B4,RDRGNT1       LOOP TO END OF TOKEN
          EQ     RDRGNT6          RETURN

*         TERMINATOR.

 RDRGNT5  NZ     X7,RDRGNT6       IF SOMETHING ASSEMBLED
          MX6    2                RETURN INDEFINATE
          EQ     RDRGNT           RETURN

 RDRGNT6  BX6    X6-X6            RETURN TOKEN
          EQ     RDRGNT           RETURN
 RDRSPC   SPACE  4,20
***       RDRSPC - PROCESS SPECIAL CASES.
*
*         ENTRY  (X1) = (LINE)
*                (X1) = ((A1))
*                (B2) = 54
*                (X0) = 7777 7777 7777 7777 7700
*
*         EXIT   TO XEQCMD, IF LEADING $.
*
*         USES   X6.
*



 NUCC     IF     DEF,NUCC
 RDRSPC   EQ     *+400000B
          AX6    X1,B2            EXTRACT FIRST CHARACTER
          BX6    -X0*X6
          SX6    X6-1R$
          NZ     X6,RDRSPC        IF NOT ESP CALL
          JUMPUP XEQCMD
 NUCC     ENDIF
 RDRSKT   SPACE  4,88
***       RDRSKT - SEARCH KEYWORD TABLE.
*
*         ENTRY  (A2) = FWA OF TABLE.
*                (X2) = ((A2))
*                (X7) = KEY.
*                TABLE FORMAT:
*                24/KEY
*                1/ = 1 IF OK TO CONTINUE PONDERING OPP. MOVE
*                5/LENGTH OF KEY - 1
*                12/OVERLAY CHARACTERS OF COMMAND PROCESSOR OR ZERO
*                18/ENTRY POINT NUMBER OR ADDRESS OF PROCESSOR
*                TERMINATED BY ZERO WORD.
*
*         EXIT   TO COMMAND PROCESSOR IF HIT.  (A2) = ADDRESS OF HIT.
*                (PONDR) = 0 UNLESS SPECIFIED IN TABLE.
*
*         USES   X2 THRU X7, A6, AND B4, IF NO HIT OR PROC. RES.,
*                ALL REGISTERS, IF PROCESSOR IN OVERLAY
*         CALLS  JUMPUP.
*



          ENTRY  RDRSKT
 RDRSKT   EQ     *+400000B
          MX6    1
          MX5    24+1
 RDRSKT1  ZR     X2,RDRSKT        EXIT IF END OF TABLE
          BX3    -X5*X2
          AX3    30               EXTRACT BIT COUNT
          SB4    X3
          AX3    X6,B4            FORM MASK
          BX4    X7-X2            COMPARE
          BX4    X3*X4            EXTRACT KEY
          ZR     X4,RDRSKT2       IF HIT
          SA2    A2+B1
          EQ     RDRSKT1          TRY NEXT ENTRY

*         HIT.

 RDRSKT2  MX7    30
          LX2    24
          NG     X2,RDRSKT3       IF OK-TO-PONDER FLAG SET
          SX6    0                ELSE FORGET PONDERING OPP. MOVE
          SA6    PONDR
 RDRSKT3  LX2    60-24            RE-ADJUST X2
          MX6    42
          BX5    -X7*X2
          SB4    X2
          BX6    X6*X5
          ZR     X6,RDRSKT4       IF RESIDENT
          RJ     =XJUMPUP         LOAD AND EXECUTE PROCESSOR

 RDRSKT4  JP     B4               EXECUTE PROCESSOR
 RDRMIL   SPACE  4,88
***       RDRMIL - MOVE INPUT LINE.
*
*         ENTRY  (READERG) = INPUT LINE POINTER.
*
*         EXIT   (READERG) = NEW INPUT LINE POINTER.
*                (ILINE) = INPUT LINE.
*                (B6) = NUMBER OF CHARACTERS MOVED.
*
*         USES   ALL REGISTERS EXCEPT X0, A0.
*         CALLS  WRLINE.
*



 RDRMIL   EQ     *+400000B
          SA1    READERG
          MX6    54
          SB6    B0               INDICATE NOTHING READ
          ZR     X1,RDRMIL        IF NOTHING IN BUFFER
          SB2    X1
          AX1    18
          SB3    X1
          SA1    B2
          BX7    X7-X7            CLEAR ASSEMBLY REGISTER
          SB2    54
          SB5    ILINE
 RDRMIL1  AX2    X1,B3
          BX2    -X6*X2           EXTRACT CHARACTER
          ZR     X2,RDRMIL3       IF TERMINATOR
          SX3    X2-1R;
          ZR     X3,RDRMIL3       IF TERMINATOR
          LX2    X2,B2
          BX7    X2+X7
          SB2    B2-6
          SB6    B6+B1            ADVANCE CHARACTER COUNT
          PL     B2,RDRMIL2       IF NOT END OF OUTPUT WORD
          SA7    B5
          SB5    B5+B1
          SB2    54
          BX7    X7-X7
 RDRMIL2  SB3    B3-6
          PL     B3,RDRMIL1       IF NOT END OF INPUT WORD
          SA1    A1+B1
          SB3    54
          EQ     RDRMIL1

*         END OF LINE.

 RDRMIL3  SA7    B5               STORE LAST PART OF WORD
          BX7    X7-X7
          SA7    B5+B1            INSURE ZERO WORD
          ZR     X2,RDRMIL5       IF NO MORE ON CARD
          SX7    A1
          SX2    B3-6             ADVANCE PAST TERMINATOR
          PL     X2,RDRMIL4       IF NOT NEW WORD
          SX2    54
          SX7    X7+B1
 RDRMIL4  LX2    18
          BX7    X2+X7
 RDRMIL5  SA7    READERG
          ZR     B6,RDRMIL        IF NOTHING MOVED, RETURN
          SA2    SWICH
          LX2    59-S.ECH
          SX5    ILINE
          PL     X2,RDRMIL6       IF NOT ECHOING, PUT ON HARDCPY
          RJ     =XWRLINE         ECHO LINE
          EQ     RDRMIL           RETURN
 RDRMIL6  RJ     =XCPLINE         COPY
          EQ     RDRMIL           RETURN
 DISCMD   SPACE  4,88
***       DISCMD - PROCESS DISPLAY COMMAND.
*



 DISCMD   SA1    ILINE
          SB2    54
          MX0    54
          RJ     =XRDRGNT         SKIP FIRST TOKEN
          RJ     =XRDRAOD         ASSEMBLE OCTAL DIGITS
          SX7    X6               SAVE ADDRESS
          RJ     =XRDRAOD
          SA0    X6               SAVE COUNT
          SX0    X7
 DISCMD1  SA2    MEMORY
          AX2    30               CURRENT FL
          IX3    X0-X2
          PL     X3,DISCMD2       IF OUTSIDE FL
          SA1    X0               THE WORD
          SX0    X0+B1            ADVANCE ADDRESS
          RJ     =XRDRWOD         CONVERT WORD TO OCTAL
```

```
          SA6    OLINE
          SA7    A6+B1
          SX5    A6
          BX7    X7-X7
          SA7    A7+B1
          RJ     =XWRLINE
          SA0    A0-B1
          SB2    A0
          GT     B2,B0,DISCMD1     IF MORE TO PRINT
          RJ     =XREREAD

 DISCMD2  SX5    =C* OUT OF RANGE.*
          RJ     =XERRMSG
          RJ     =XREREAD

 CHACMD   SPACE  4,22
***       CHACMD - PROCESS CHANGE COMMAND.
*


 CHACMD   SA1    ILINE
          SB2    54
          MX0    54
          RJ     =XRDRGNT          SKIP FIRST TOKEN
          RJ     =XRDRAOD          ASSEMBLE OCTAL DIGITS
          SA6    CHACMDA           SAVE ADDRESS
          SA2    MEMORY
          AX2    30
          IX3    X6-X2
          PL     X3,DISCMD2        IF OUTSIDE FL
          RJ     =XRDRAOD          ASSEMBLE NEXT OCTAL DIGITS
          SA2    A6                ADDRESS TO CHANGE
          SA6    X2                UPDATE CM
          RJ     =XREREAD

 CHACMDA  EQU    OLINE             SCRATCH
 WHACMD   SPACE  4,10
***       WHACMD - PROCESS WHAT COMMAND.
*


 WHACMD   SA1    READERA
          LX1    2
          SX5    X1+MOVMS
          RJ     =XWRLINE          OUTPUT QUESTION
          RJ     =XREREAD
 RDRAOD   SPACE  4,88
***       RDRAOD - ASSEMBLE OCTAL DIGITS.
*
*         ENTRY  (A1, X1, B2) = INPUT STREAM POINTER.
*
*         EXIT   (X6) = ASSEMBLED RESULT.
*                A1, X1, B2 UPDATED.
*
*                ASSEMBLY STOPS ON ANY NON-BLANK, NON-OCTAL CHARACTER,
*                OR 20 DIGITS.
*
*         USES   B3, B4, B5, B6, X2, X3.
*


          ENTRY  RDRAOD
 RDRAOD   EQ     *+400000B
          BX6    X6-X6             CLEAR ASSEMBLY
          MX3    20                COUNTER
          SB3    1R
          SB4    1R0
          SB5    1R8
 RDRAOD1  AX2    X1,B2
          BX2    -X0*X2            EXTRACT CHARACTER
          SB2    B2-6
          SB6    X2
          PL     B2,RDRAOD2        IF NOT NEW INPUT WORD
          SA1    A1+B1
          SB2    54
 RDRAOD2  EQ     B6,B3,RDRAOD1     IF BLANK, IGNORE
          LT     B6,B4,RDRAOD      IF NON-OCTAL, RETURN
          GE     B6,B5,RDRAOD      IF NON-OCTAL, RETURN
          LX6    3
          SX2    B6-B4             CONVERT DIGIT
          BX6    X6+X2
          LX3    1
          NG     X3,RDRAOD1        IF NOT 20 DIGITS
          EQ     RDRAOD            RETURN
*CALL     COMCCDD
 RDRCDD   SPACE  4,23
***       RDRCDD - CONVERT DECIMAL DIGITS.
*
*         ENTRY  (X1) = NUMBER TO BE CONVERTED.
*
*         EXIT   (X6) = CONVERTED RESULT.
*
*         USES   X1, X2, X3, X4, X6, X7, A2, A3, A4, B2, B3, B4, B5.
*         CALLS  CDD.
*


          ENTRY  RDRCDD
 RDRCDD1  RJ     CDD               CONVERT POSITIVE NUMBER

 RDRCDD   EQ     *+400000B
          UX1    X1
          PL     X1,RDRCDD1        IF POSITIVE
          BX1    -X1
          RJ     CDD               CONVERT ABSOLUTE VALUE
          SX7    1R-&1R
          LX7    X7,B2
          BX6    X6-X7             APPEND SIGN
          EQ     RDRCDD
*CALL     COMCWOD
 RDRWOD   EQU    WOD
          ENTRY  RDRWOD
*CALL     COMCSFN
 RDRSFN   EQU    SFN
          ENTRY  RDRSFN
*CALL     COMCCOD
 RDRCOD   EQU    COD
          ENTRY  RDRCOD
          END
INITSWI
          IDENT  INITSWI
          SST
          LIST   F
 RA100    EQU    100B
 .INITSWI EQU    1                 THIS IS INITSWI
*CALL     BOARD
*CALL     IOCOM
*CALL     SQRLST
*CALL DEBUG
*CALL     LET
*CALL     TLET
*CALL     RELY
          ENTRY  RELFILE
          ORG    ILINE
 ILINE    BSSZ   9
 OLINE    BSSZ   14
 INPFN    BSSZ   1
 MOVMS    DATA   38L ENTER MOVE OR TYPE GO.
 DBGMS    DATA   38L ENTER DEBUG COMMAND.
 ENDMS    DATA   38L THE GAME IS OVER.
 HMTMS    DATA   38L WHERE IS THE MINUTE HAND  (00)
 GAMMS    BSSZ   7
 GMCNT    DATA   0
 SWICH    VFD    'SWDEFO'
 GOCNT    DATA   0
 MTIDS    DATA   0
 MTLIM    DATA   0
 OPNAM    DATA   10HCHALLENGER
 ALTFN    VFD    42/0LALTER,18/3
 LINES    CON    -1
 CPRPV    DATA   0
 CPESP    DATA   0
 LET      SPACE  4
          ORG    LET
 ASKFCT   CON    40
 COPIES   CON    0                 NUMBER OF GAME SCORES
 EXTRAS   CON    5
 EXTRAT   CON    20
 FBBKRK   CON    700B
 FBMOBL   CON    200B
 FCHKOP   CON    120B
 FDEPTH   CON    2
 FDRAWA   CON    77D
 FDRAWG   CON    20B
 FDRAWI   CON    2
 FDRAWM   CON    55D
 FDRAWT   CON    3
 FEZTOL   CON    20B
 FKCORN   CON    300B              MATED KING CORNER HATRED
 FKCTRP   CON    140B
 FKKNMT   CON    100B              MATING KING AND KNIGHT TROPISM
 FKPTRP   CON    500B
 FKSANQ   CON    200B
 FKSATK   CON    24B
 FKSCNQ   CON    120B
 FKSCOF   CON    100B
 FKSCRM   CON    40B
 FKSFRD   CON    100B
 FKSISO   CON    300B
 FKSMAT   CON    2
 FKSMNF   CON    2
 FKSOPF   CON    340B
 FKSQBN   CON    2
 FKS2BS   CON    100B
 FLXTOL   CON    64
 FMAXMT   CON    127*64
 FMTEGE   CON    400B              MIN EDGE FOR USING MATING ALGORITHM
 FNATKD   CON    200B
 FNATNP   CON    400B
 FNBKRK   CON    600B
 FNCTRP   CON    100B
 FNKTRP   CON    60B
 FORKBN   CON    1700B
 FPDCOL   BSS    0
 FPDCQR   CON    -20B
 FPDCQN   CON    -20B
 FPDCQB   CON    200B
 FPDCQC   CON    300B
 FPDCKC   CON    360B
 FPDCKB   CON    120B
 FPDCKN   CON    -40B
 FPDCKR   CON    -40B
 FPNBSC   CON    600B
 FPNDBL   CON    500B
 FPNISC   CON    700B
 FPNISO   CON    1500B
 FPNLBK   CON    240B
 FPNMAJ   CON    101B
 FPNPAS   CON    140B
 FPNPSA   CON    20B
 FPNPSB   CON    36B
 FPNPSD   CON    14B
 FPNVBK   CON    500B
 FQKTRP   CON    40B
 FQMOBL   CON    40B
 FRHUNG   CON    1500B
 FRKDBL   CON    500B
 FRKOPF   CON    500B
 FRKTRP   CON    100B
 FRKVNP   CON    500B
 FRK7TH   CON    1600B
 FRMOBL   CON    100B
 FTMBON   CON    500B
 FTRADE   CON    36
 FTRDSL   CON    80*64+36
 FTRPOK   CON    2
 FTRPWN   CON    8
 GCLOCK   CON    0
 JIGMVS   CON    0
 JIGSIZ   CON    0
 LINEPP   CON    23D
 LOGTRA   CON    9D
 MDEPTH   CON    3
 MOVNUM   CON    0                 CURRENT MOVE NUMBER
 MSCODE   CON    0
 MSMASK   CON    0
 MVRL50   CON    50                50 MOVE RULE
 OVRHED   CON    10
 PERCNT   CON    96
 QADPTH   CON    4
 QDEPTH   CON    2
 RANDOM   CON    0
 RATEMY   CON    1550
 RATEPN   CON    200
 RATEYR   CON    1350
 RULMV1   CON    40
 RULMV2   CON    10
 RULTM1   CON    120
 RULTM2   CON    30
 STEPHI   CON    0
 STEPLO   CON    NPLY
 TGRACE   CON    60000D
 TMAXOM   CON    400B
 TPMOKR   CON    63B
```

```
 TPMRAT   CON    50B
 TPMRMX   CON    170B
 TPONBN   CON    40B
 TQFRST   CON    15
 TQNEXT   CON    15
 TRATIC   CON    1
 TRATLO   CON    32B
 TRATMN   CON    14B
 TRATMX   CON    37B
 TRATOL   CON    0
 TRCLVL   CON    99D
 VALUEX   CON    0
 VALUEP   CON    100B            VALUE OF PAWN
 VALUER   CON    500B            VALUE OF ROOK
 VALUEN   CON    321B            VALUE OF KNIGHT
 VALUEB   CON    334B            VALUE OF BISHOP
 VALUEQ   CON    1130B           VALUE OF QUEN
 VALUEK   CON    10000B
 WODREL   CON    48
 TLET     SPACE  4,10
          ORG    TLET

 LIMBA    SPACE  4
          ORG    RELFILE
 RELFILE  FILEB  RA100,1,FET=6

          DATA   0LCHESSRELFILE
          BSSZ   RELFDB-*
 RELFDB   DATA   0LRELFILE
          DATA   14B             ID
          DATA   20B             PW
          DATA   103B            CY=1
          DATA   115B            RN=1
          DATA   0

          DATA   0LCHESSLIBFILE
          BSSZ   LIBFDB-*
 LIBFDB   DATA   0LLIBRARY
          DATA   14B             ID
          DATA   20B             PW
          DATA   103B            CY=1
          DATA   115B            RN=1
          DATA   0

          DATA   0LCHESSGAMFILE
          BSSZ   GAMFDB-*
 GAMFDB   DATA   0LGAMFILE
          DATA   14B             ID
          DATA   20B             PW
          DATA   103B            CY=1
          DATA   115B            RN=1
          DATA   0

 RELSW    DATA   0               RELIABILITY MODE SWITCH
 GAMOUT   DATA   0               GAMFILE OUT POINTER AT LAST EXTEND

          ORG    SQRLSTB
 GLOBNK   BSSZ   GLOBNKL
 SQRLST   BSSZ   SQRLSTL
 NODEBK   BSSZ   NODEBKL

          ORG    LIMBA
 I        SET    EC.LMB
 J        SET    NPLY
          DUP    NPLY
          CON    I
 I        SET    I+J
 J        SET    J-1
          ENDD

          ORG    SQRLT
          CON    EC.TRA+400B
          SPACE  4
          END
ENGGEN
          IDENT  ENGGEN
          SST
 ENGGEN   TITLE  ENGGEN - FAST ENGLISH NOTATION GENERATOR.
*CALL     IOCOM
*CALL     SQRLST
 ENGGEN   SPACE  4,88
***       ENGGEN - ENGLISH NOTATION GENERATOR.
*
*         ENTRY  (X1) = MOVE.
*                (B2) = ADDRESS OF BOARD WITH POSITION BEFORE MOVE.
*
*         EXIT   (X6) = 9R MOVE
*
*         USES   X0, X1, X2, X3, X4, X5, X6, X7, A2, A3, A4, A5, AND B2.
*


 ENGGEN2  MX6    -3
          BX5    -X6*X1
          LX1    -3
          BX4    -X6*X1
          LX1    -3
          BX3    -X6*X1
          LX1    -3
          BX2    -X6*X1
          LX1    59-S.CAP+9
          MX6    1
          BX6    X1*X6           EXTRACT CAPTURE BIT
          LX2    36
          LX3    42
          LX4    18
          LX5    24
          BX2    X2+X3
          BX4    X4+X5
          BX2    X2+X4
          SA3    =9R A1-A1
          LX6    30-59
          IX6    X3+X6
          IX6    X2+X6

          ENTRY  ENGGEN
 ENGGEN   EQ     *+400000B
          SA2    SWICH
          LX2    59-S.ALG
          NG     X2,ENGGEN2      IF ALGEBRAIC
          SA2    MSIDE
          MX0    54
          BX3    -X0*X1          TO SQUARE
          LX1    54
          BX4    -X0*X1          FROM SQUARE
          AX2    59
          MX0    -3
          BX7    -X0*X2
          LX1    59-S.CAP+6
          SA2    B2+X3           TO PIECE
          SA5    B2+X4           FROM FIECE
          LX3    -3
          LX4    -3
          BX3    X7-X3           COMPLEMENT RANK IF BLACK TO MOVE
          BX4    X7-X4
          SB2    1R1
          SX6    X3+B2           TO RANK
          SX7    X4+B2           FROM RANK
          BX3    X0*X3           CLEAR RANKS
          BX4    X0*X4
          LX3    3
          LX4    3
          SA3    ENGGENA+X3      TO FILE
          SA4    ENGGENA+X4      FROM FILE
          IX6    X6+X3           TO SQUARE
          IX7    X7+X4           FROM SQUARE
          SA5    ENGGENB+X5      FROM PIECE
          LX5    48
          LX7    30              FROM SQUARE
          SX0    1R
          LX0    54
          BX6    X6+X0
          NG     X1,ENGGEN1      IF CAPTURE
          LX6    6               TO SQUARE
          BX6    X6+X7
          BX6    X6+X5
          BX6    X6+X0
          SX7    1R-
          LX7    24
          BX6    X6+X7
          EQ     ENGGEN

*         CAPTURE.

 ENGGEN1  SA2    ENGGENB+X2      TO PIECE
          LX2    18
          BX6    X6+X7
          BX6    X6+X5
          BX6    X6+X2
          SX7    1R*
          LX7    24
          BX6    X6+X7
          EQ     ENGGEN          RETURN

 ENGGENA  BSS    0
          ECHO   2,X=(QR,QN,QB,Q,K,KB,KN,KR)
          VFD    54/2R!X
          VFD    6/0
          ECHO   1,X=(K,Q,B,N,R,P)
          VFD    60/0R!X
 ENGGENB  BSS    0
          ECHO   1,X=(P,P,R,N,B,Q,K)
          VFD    60/0R!X
          END
CHESS10
          IDENT  CHE$$10
          SST

          CON    =XECREATE       ECS PLUG TABLE
 10       THIS

          END    'END'
CREATE
          IDENT  CREATE
          SST
 CREATE   TITLE  CREATE - MISCELLANEOUS CHESS ALGORITHMS.
          LIST   G,X
          B1=1
*CALL     SQRLST
*CALL     LET
*CALL     TLET
*CALL     BOARD
*CALL     CONST
*CALL     STACKS
*CALL     IOCOM
*CALL DEBUG
 ECREATE  ECERTB EEVALU8
 ASMATK   SPACE  4,60
**        ASMATK - ASSEMBLE ATTACK MAPS.
*
*         ENTRY  (X6,X7) = ASSEMBLY SQUARE LIST.
*                (X1,X2) = SQUARE LIST OF PIECES WITH ATTACKS TO ADD.
*                (X0) = 0000 4000 0000 0000 0000
*                (A0) = 12+BT2SQ
*                (B4) = 60+BT2SQ
*
*         EXIT   (X6,X7) = ASSEMBLED SQUARE LIST.
*
*         USES   X1, X2, X3, X4, X5, A3, A4, B2.
*


 ASMATK   MACRO
          LOCAL  AAAAAAAA,BBBBBBBB,CCCCCCCC,DDDDDDDD
          LX2    48
          ZR     X1,BBBBBBBB     IF NOTHING IN FIRST WORD
          NX5    X1,B2
          AX5    X0,B2
 AAAAAAAA SA3    A0+B2           SQUARE NUMBER
          LX3    1
          SA3    ATKFR+X3
          BX1    -X5*X1
          SA4    A3+B1
          BX6    X6+X3
          BX7    X7+X4
          NX5    X1,B2
          AX5    X0,B2
          NZ     X5,AAAAAAAA
 BBBBBBBB ZR     X2,DDDDDDDD     IF NONE IN SECOND WORD
          NX5    X2,B2
          AX5    X0,B2
 CCCCCCCC SA3    B4+B2
          LX3    1
          SA3    ATKFR+X3
          BX2    -X5*X2
          SA4    A3+B1
          BX6    X6+X3
          BX7    X7+X4
          NX5    X2,B2
          AX5    X0,B2
          NZ     X5,CCCCCCCC
 DDDDDDDD BSS    0
          ENDM
 SETEPT   SPACE  4,88
```

```
***       SETEPT - SETUP RECURSIVE ENTRY POINTS.
*


          ENTRY  SETEPT
 SETEPT   SA1    SETEPTA
          BX6    X1
 SETEPT1  SA6    X6+1
          SA1    A1+B1
          BX6    X1
          NZ     X6,SETEPT1
          JUMPUP INIMSI            INITIALIZE MASS STORAGE INDEX

 SETEPTA  EQ     =XTRSRCH.
-         VFD    30/TRSRCH
          EQ     =XEVALU8.
-         VFD    30/EVALU8
          EQ     =XDEBUGR.
-         VFD    30/DEBUGR
          EQ     =XPNDPAU.
-         VFD    30/PNDPAU
          CON    0                 END OF ENTRY POINT PLUG TABLE
 CREATE   SPACE  4,88
***       CREATE - CREATE INITIAL SQUARE LISTS.
*
*         ENTRY  (BOARD) = INITIAL POSITION.
*
*         EXIT   ALL SQUARE LISTS ARE SET UP.
*


          ENTRY  CREATE
 CREATE   EQ     *+400000B
          SB1    1
          SX6    B1
          LX6    S.THI
          SA1    PONDR
          BX1    -X6*X1            CLEAR CURRENTLY PONDERING FLAG
          LX6    S.THQ-S.THI
          BX6    -X6*X1            ALSO CLEAR PONDER QUIT FLAG
          SA6    A1
          BX6    X6-X6
          BX7    X7-X7
          SA6    SQRLST
          SA7    A6+B1
          SB2    SQRLSTL/2-2/2
          SB6    STACK1
 CREATE1  SA6    A7+B1
          SA7    A6+B1             CLEAR ALL SQUARE LISTS
          SB2    B2-1
          NZ     B2,CREATE1
          SA6    PSMOD             PAWN STRUCTURE MODIFICATIONS
          SA6    NPLYD             CLEAR PLIES SINCE BASE
          SA6    =XECRERR          ECS READ ERROR FLAG
          SA1    LOGTRA            LOG OF TRANSPOSITIONS TABLE CAPACITY
          SB2    X1                SAVE IN TOP OF HASH WORD
          PX6    X6,B2
          SA6    TRASH             INIT TRANSPOSITIONS HASH
          SX5    EC.TRA            CLEAR TRANSPOSITIONS HASH TABLE
          SX1    B1
          LX1    X1,B2             GET NUMBER OF SLOTS IN TABLE
          BX6    X1                MULTIPLY BY LE.TRA
          LX6    2
          IX6    X6+X1             LENGTH OF TABLE IN WORDS
          ERRNZ  LE.TRA-5
          IX7    X5+X6             GET FWA OF SQRLST IN ECS
          SA7    SQRLT
          AX6    8                 GET NO. OF 400B BLOCKS TO CLEAR
          SB2    X6
          SB3    400B
          SX0    SQRLSTBE-SQRLST+1
          IX0    X0+X7             ALSO GET ENOUGH ECS FOR A FULL PLY
          RJ     =XCHEKFE          CHECK FOR ENOUGH ECS
          BX0    X5                SET EC.TRA AS FWE TO CLEAR
          SA0    SQRLST
          ERRNG  SQRLSTL-400B
          SX1    400B
 CREATE1A WE     B3
          RJ     =XECWERR
          SB2    B2-B1
          IX0    X0+X1
          NZ     B2,CREATE1A       IF NOT DONE CLEARING TABLE
          SA1    ONBRD             INITIALIZE STOPS
          BX6    -X1
          SA2    A1+B1
          SA6    STOPS
          BX7    -X2
          SA7    A6+B1
          SA1    STATUS
          MX0    51
          BX6    -X0*X1            EXTRACT BLACK 50 MOVE COUNT
          LX0    30
          BX7    -X0*X1            EXTRACT WHITE 50 MOVE COUNT
          LX0    30+9
          BX5    X6+X7
          BX6    -X0*X1            EXTRACT BLACK CASTLE SQUARES
          LX6    20-9
          SA6    CSTAT+1
          LX0    9
          BX7    -X0*X1            EXTRACT BLACK EP SQUARES
          LX7    40-18
          LX0    3+9+9
          SA7    ENPAS+1
          BX6    -X0*X1            EXTRACT WHITE CASTLE SQUARES
          LX6    30-39
          SA6    A6-B1
          LX0    9
          BX7    -X0*X1            EXTRACT WHITE EP SQUARES
          LX7    10-48
          SA7    A7-B1
          AX1    59
          SX6    B1+B1
          BX6    -X6-X1            +2 IF WHITE ON MOVE, -2 IF BLACK
          SA6    MSIDE
          AX6    59                +0 IF WHITE ON MOVE, -0 IF BLACK
          SA6    SSIDE
          SX7    B1
          BX6    X6*X7             0 IF WHITE ON MOVE, 1 IF BLACK
          SA6    OSIDE
          BX7    X6
          LX7    4                 16 * OSIDE
          IX7    X7-X6             15 * OSIDE
          LX7    1                 30 * OSIDE
          MX4    1
          SA1    SQRLT
          SB3    X7+30
          LX6    X5,B3             PUT SIDE TO MOVE COUNT LOWER
          BX7    X4+X1
          SA6    CNT50             INITIALIZE 50 MOVE RULE COUNT
          SA7    SLECS             INDICATE SL NOT IN ECS
          SX6    MOVES
          SA6    LINDX
          SX2    SQRLSTL
          IX6    X2+X1
          SA6    VAECS
          SX7    73567B            INITIALIZE PACKED BOARD WORD 1
          SA1    OSIDE             0 IF WHITE TO MOVE, ELSE 1
          BX2    -X1
          LX1    24
          SA2    ENPAS+1+X2        ENPASSANT SQUARE LIST, IF ANY
          BX7    X7+X1             SET SIDE TO MOVE IN PACKED BOARD
          ZR     X2,CREATE1C       IF NO ENPASSANT SQUARE
          LX2    59-10B
          ZR     X1,CREATE1B       IF WHITE TO MOVE
          LX2    30                ELSE WHITE PAWN IS IN FIRST WORD
 CREATE1B NX2,B3 X2                GET COLUMN NUMBER OF PAWN + 10B
          SX2    B3
          LX2    16
          BX7    X7+X2             SET E.P. STATUS IN PACKED BOARD
 CREATE1C SA1    CSTAT             SET CASTLING STATUSES
          SX6    B1
          LX1    60+23-38
          LX6    23
          BX2    X6*X1
          BX7    X7+X2             WHITE QUEEN SIDE
          LX1    7-1
          AX6    1
          BX2    X6*X1
          BX7    X7+X2             WHITE KING SIDE
          SA1    A1+B1             BLACK CASTLE STATUSES
          AX6    1
          LX1    60+21-28
          BX2    X6*X1
          BX7    X7+X2             BLACK QUEEN SIDE
          AX6    1
          LX1    7-1
          BX2    X6*X1
          BX7    X7+X2             BLACK KING SIDE
          SA1    REPPZ
          SA2    CREATEA
          BX6    X1
          SA6    REPPN             INITIALIZE PLY N POINTER
          SA7    X6                INITIALIZE PACKED BOARD
          BX7    X2
          DUP    LE.PAC-1,1
          SA7    A7+B1
          SX6    REPPB
          SA6    REPST             INITIALIZE START POINTER
          SA6    NBOARD-1          INITIALIZE OFF BOARD REFERENCE
          SB3    64-1              SQUARE NUMBER

*         LOOP OVER ALL SQUARES.  BUILD LOCATION SQUARE LISTS.

 CREATE2  SA1    BOARD+B3
          ZR     X1,CREATE3        IF EMPTY SQUARE
          RJ     =XADDLOC          ADD TO ALL SQUARE LISTS
          SX6    B3
          SA6    B6                SAVE PIECE SQUARE NUMBER
          SB6    B6+B1
 CREATE3  SB3    B3-B1
          PL     B3,CREATE2        LOOP THROUGH ENTIRE BOARD

*         LOOP THROUGH STACK OF PIECES.  BUILD ATTACK MAPS.

 CREATE4  SB5    STACK1
          EQ     B5,B6,CREATE5     IF ALL ATTACKS GENERATED
          SA3    B6-B1             SQUARE NUMBER
          SB6    B6-B1
          RJ     =XADDATK          ADD ATTACKS
          EQ     CREATE4
 CREATE5  SA1    MSIDE             INITIALIZE VALID MOVE SQUARE LISTS
          SA1    X1+ALLOC
          SA2    A1+B1
          SA3    ONBRD
          SA4    A3+B1
          BX6    X1
          BX7    X2
          SA6    GENFR
          SA7    A6+B1
          BX6    -X1*X3
          BX7    -X2*X4
          SA6    A7+B1
          SA7    A6+B1

*         COMPUTE MATERIAL BALANCE SIGNATURE.

          BX1    X1-X1
          SB2    TPLOC
          SB3    QUEN*2
 CREATE6  SA5    B2+B3
          SA2    A5+B1
          SA3    B2-B3
          SA4    A3+B1
          CX5    X5
          CX2    X2
          CX3    X3
          CX4    X4
          IX5    X5+X2
          IX3    X3+X4
          LX5    30
          IX3    X3+X5
          LX1    4
          IX1    X1+X3
          SB3    B3-2
          NZ     B3,CREATE6        LOOP THROUGH ALL PIECES
          RJ     =XMBHASH          COMPUTE MATERIAL BALANCE.
          SA6    MBVAL             UPDATE MATERIAL BALANCE WORD
          SA7    MBSCR             AND SCORE

**        CLEAR NSRCH AND LIMBL.

          BX6    X6-X6
          BX7    X7-X7
          SB2    NSRCH
          SB3    NSRCH+NPLY
 CREATE7  SA6    B2
          SA7    B2+LIMBL-NSRCH
          SB2    B2+B1
          LT     B2,B3,CREATE7
          SA5    STATUS
          SX6    B1
          SX7    B1
          LX6    27                BLACK CHECK BIT
```

```
          LX7    57               WHITE CHECK BIT
          BX5    -X6*X5
          BX5    -X7*X5
          SA1    AWATK            ALL WHITE ATTACKS
          SA3    KBLOC            BLACK KING LOCATION
          SA2    A1+B1
          SA4    A3+B1
          BX1    X1*X3
          BX2    X2*X4
          BX2    X1+X2
          ZR     X2,CREATE8       IF BLACK NOT IN CHECK
          BX5    X6+X5            SET BLACK IN CHECK
 CREATE8  SA1    ABATK            ALL BLACK ATTACKS
          SA3    KWLOC            WHITE KING LOCATION
          SA2    A1+B1
          SA4    A3+B1
          BX1    X1*X3
          BX2    X2*X4
          BX2    X1+X2
          ZR     X2,CREATE9       IF WHITE NOT IN CHECK
          BX5    X7+X5            SET WHITE IN CHECK
 CREATE9  BX6    X5
          SA6    A5               UPDATE STATUS
          RJ     =XPSUPDT         INITIALIZE PAWN STRUCTURE TABLES
          EQ     CREATE           RETURN

 CREATEA  BSS    0
          DUP    15,1
          VFD    4/7
 UPDATE   SPACE  4,88
***       UPDATE - UPDATE ALL SQUARELISTS.
*
*         ENTRY  (X1) = THE MOVE.
*         USES   ALL REGISTERS, STACK1.
*
*         EXIT   TO MOVEIT OR LOSEIT.
*



          ENTRY  UPDATE
 UPDATE   EQ     *+400000B
          SA5    REPPN            PLY N PACKED BOARD ADDRESS
          SB2    LE.PAC
          SA4    X5               FIRST WORD OF PACKED BOARD
          MX6    4
          SX3    B1
          LX6    20
          BX4    -X6*X4           CLEAR EP SQUARE IN PACKED BOARD
          LX3    24
          BX4    X3-X4            CHANGE SIDE TO MOVE IN PACKED BOARD
          DUP    LE.PAC-1         MOVE PACKED BOARD
          BX6    X4
          SA6    A4+B2
          SA4    A4+B1
          ENDD
          BX6    X4
          SA6    A4+B2
          SX7    A4+B1
          SA7    A5               UPDATE REPPN
          SA4    OSIDE            0 IF WHITE TO MOVE, 1 IF BLACK
          MX6    -18
          BX1    -X6*X1           CLEAR JUNK
          BX6    X6-X6
          SX5    B1
          MX0    54
          SA6    PSMOD
          SA6    ENPAS            CLEAR ENPASSANT SQUARES
          SA6    A6+B1
          BX7    -X0*X1           TO SQUARE
          AX1    6
          BX6    -X0*X1           FROM SQUARE
          BX2    X0*X1
          LX2    -6-3             SPLIT FLAGS
          SA6    UPDATEA          SAVE FROM SQUARE
          SA7    A6+B1
          SB2    X2               TOP 3 BITS OF FLAGS (CAP)
          JP     UPDATE1+B2

 UPDATE1  PL     X2,MOVEIT        0000XX ORDINARY MOVE
          EQ     UPDATE2          0001XX PAWN MOVE 2 SQUARES
 +        EQ     UPDATE3          001YYX PROMOTE TO YY+2
 +        EQ     UPDATE4          010YZX MISCELLANEOUS ACS
 +        EQ     UPDATE           011XXX NULL MOVE
 +        PL     X2,LOSEIT        1000XX ORDINARY CAPTURE
          EQ     UPDATE8          1001XX CAPTURE EN PASSANT
 +        EQ     UPDATE9          101YYX PROMOTE TO YY+2, CAPTURE
 +        EQ     UPDATE10         110XXX CAPTURE ACS
 +        EQ     UPDATE11         111YYX P*R(R1)/YY+2, ACS

*         PAWN MOVE 2 SQUARES.

 UPDATE2  SA1    SQ2BT+X7         CHECK FOR ENPASSANT CAPTURES
          UX1    B2,X1
          LX3    X5,B2
          LX4    2                0 IF WHITE MOVES, 4 IF BLACK
          IX4    X4+X1            0 IF WHITE, 5 IF BLACK
          SA2    TPLOC-2+X4       OPPONENT PAWNS
          LX2    1
          BX1    X3*X2            CHECK FOR PAWN ON EITHER SIDE
          LX2    60-1-1
          BX2    X3*X2
          BX1    X2+X1
          ZR     X1,MOVEIT        IF NO POSSIBLE EP CAPTURES
          IX4    X6+X7            ELSE SET ENPASSANT STATUS
          AX4    X4,B1            AVERAGE TO AND FROM SQUARE
          SA4    SQ2BT+X4
          UX4    B2,X4
          LX6    X5,B2
          SA6    ENPAS+X4         SET EN PASSANT SQUARE
          MX3    60-3
          BX7    -X3*X7           EXTRACT PAWN COLUMN
          SX7    X7+10B           COLUMN NO. + EP FLAG
          LX7    16               ALIGN FOR INSERTION INTO PACKED BOARD
          SA4    REPPN
          SA4    X4               FIRST WORD OF PACKED BOARD
          BX6    X4+X7            SET EP SQUARE IN PACKED BOARD
          SA6    A4
          EQ     MOVEIT

*         PROMOTE TO YY+2

 UPDATE3  RJ     =XPROMOT         PROMOTE PAWN
          EQ     MOVEIT

*         MISCELLANEOUS MOVES AFFECTING CASTLE STATUS.

 UPDATE4  LX1    X2,B1
          NG     X1,UPDATE5       IF CASTLE

*         MOVE TO OR FROM CASTLE STATUS SQUARE.

          RJ     =XPROACS         PROCESS ACS
          EQ     MOVEIT

*         CASTLE.

 UPDATE5  PL     X2,UPDATE6       IF SHORT CASTLE

*         LONG CASTLE.

          SX2    X7-2             ROOK FROM-SQUARE
          IX3    X6+X7
          AX7    X3,B1            ROOK TO-SQUARE
          EQ     UPDATE7

*         SHORT CASTLE.

 UPDATE6  SX2    X7+1             ROOK FROM-SQUARE
          IX3    X6+X7
          AX7    X3,B1            ROOK TO-SQUARE
 UPDATE7  BX6    X6-X6
          SA6    CSTAT+X4         DISALLOW FURTHER CASTLING
          SX6    X2
          SA6    A7+B1            ROOK FROM
          SA7    A6+B1            ROOK TO
          SA1    X6+NBOARD
          SA3    REPPN
          BX6    X3
          ERRNZ  REPST-REPPN-1
          SA6    A3+B1            SET REPEAT CHECK STOP POINT
          SA2    X3               GET FIRST WORD OF PACKED BOARD
          MX6    2
          LX6    24
          SB3    X4               OSIDE
          SB3    B3+B3            0 FOR WHITE CASTLE, 2 FOR BLACK
          AX6    X6,B3
          BX6    -X6*X2           CLEAR CASTLE BITS IN PACKED BOARD
          SA6    A2
          SB3    X7
          RJ     =XADDLOC         PUT ROOK ON LOC SQUARE LISTS
          SX3    B3
          RJ     =XADDATK         PUT ON ROOK ATTACKS
          SA1    UPDATEA+3
          SB3    X1
          RJ     =XCUTATK         CUT ATTACKS THRU ROOK TO-SQUARE
          SA3    UPDATEA+2
          RJ     =XDELATK         DELETE ATTACKS FROM ROOK FROM-SQUARE
          EQ     MOVEIT           MOVE KING

*         CAPTURE EN PASSANT

 UPDATE8  MX0    57
          BX1    X0*X6            FROM RANK
          BX2    -X0*X7           TO FILE
          BX3    X1+X2            SQUARE OF CAPTURED PAWN
          RJ     =XMBCAPT         CHANGE MATERIAL BALANCE
          SA6    MBVAL            UPDATE MATERIAL BALANCE WORD
          SA7    MBSCR            AND SCORE
          RJ     =XDELATK         REMOVE CAPTURED PAWN
          RJ     =XPRPATK         PROPAGATE ATTACKS THROUGH CAPTURE SQ
          EQ     MOVEIT           MOVE PAWN

*         P*X(X8)/YY+2

 UPDATE9  RJ     =XPROMOT         PROMOTE PAWN
          EQ     LOSEIT

*         CAPTURE AFFECTING CASTLE STATUS.

 UPDATE10 RJ     =XPROACS         PROCESS ACS
          EQ     LOSEIT

*         P*R(R8)/YY+2, AFFECTING CASTLE STATUS.

 UPDATE11 RJ     =XPROMOT         PROMOTE PAWN
          SA1    UPDATEA
          BX6    X1
          SA2    A1+B1
          BX7    X2
          SX5    1
          RJ     =XPROACS         PROCESS ACS
          EQ     LOSEIT

 UPDATEA  BSS    1                FROM SQUARE
          BSS    1                TO SQUARE
          BSS    1                ROOK FROM SQUARE FOR CASTLE
          BSS    1                ROOK TO SQUARE

 MOVEIT   SPACE  4,88
***       MOVEIT - MOVE A PIECE ON ALL SQUARE LISTS.
*
*         ENTRY  (UPDATEA) = FROM SQUARE NUMBER.
*                (UPDATEA+1) = TO SQUARE NUMBER.
*
*         EXIT   TO EXIT LINE OF UPDATE.
*



 MOVEIT   SA1    UPDATEA          FROM SQUARE
          SA2    A1+B1            TO SQUARE
          SB3    X2
          SA3    CNT50
          SX6    B1
          IX6    X6+X3            ADVANCE 50 MOVE COUNT
          LX6    30
          SA6    A3
          SA1    NBOARD+X1        TYPE
          AX2    X1,B1
          NZ     X2,MOVEIT1       IF NOT PAWN MOVE
          BX6    X6-X6
          SA6    A6               CLEAR 50 MOVE COUNT
          ERRNZ  REPPN-CNT50-1
          SA2    A3+B1            REPPN
          BX6    X2
          ERRNZ  REPST-REPPN-1
          SA6    A2+B1            SET REPEAT SEARCH STOP POINT
 MOVEIT1  RJ     =XADDLOC
          RJ     =XCUTATK         STOP ATTACKS THROUGH TO SQUARE
          SA3    UPDATEA
          RJ     =XDELATK         DELETE PIECE AND ATTACKS FROM FROM SQ
          RJ     =XPRPATK         PROPAGATE ATTACKS THROUGH FROM SQUARE
          SA3    UPDATEA+1
          RJ     =XADDATK         ADD ATTACKS FROM TO SQUARE
          RJ     =XPSUPDT         UPDATE PAWN STRUCTURE
```

```
         EQ     UPDATE            RETURN FROM UPDATE
 LOSEIT  SPACE  4,23
***      LOSEIT - CAPTURE A PIECE ON ALL SQUARE LISTS.
*
*        ENTRY  (UPDATEA) = FROM SQUARE NUMBER.
*               (UPDATEA+1) = TO SQUARE NUMBER.
*
*        EXIT   TO EXIT LINE OF UPDATE.
*



 LOSEIT  SA3    UPDATEA+1         TO SQUARE
         RJ     =XMBCAPT          CHANGE MATERIAL BALANCE
         SA6    MBVAL             UPDATE MATERIAL BALANCE WORD
         SA7    MBSCR             AND SCORE
         RJ     =XDELATK          DELETE ATTACKS OF CAPTURED PIECE
         SA1    UPDATEA
         SA2    A1+B1
         SB3    X2                TO SQUARE
         SA1    NBOARD+X1         THE MOVING PIECE
         RJ     =XADDLOC          ADD PIECE ON TO SQUARE
         SA3    UPDATEA
         RJ     =XDELATK          DELETE ATTACKS FROM FROM SQUARE
         RJ     =XPRPATK          PROPAGATE ATTACKS THROUGH FROM SQ
         SA3    UPDATEA+1
         RJ     =XADDATK          ADD ATTACKS FROM TO SQUARE
         RJ     =XPSUPDT          UPDATE PAWN STRUCTURE
         SA1    REPPN
         BX6    X1
         ERRNZ  REPST-REPPN-1
         SA6    A1+B1             SET REPEAT CHECK STOP POINT
         BX6    X6-X6
         ERRNZ  REPPN-CNT50-1
         SA6    A1-B1             CNT50
         EQ     UPDATE
 PROACS  SPACE 4,41
***      PROACS - PROCESS CASTLE STATUS CHANGES.
*
*        ENTRY  (X6) = FROM SQUARE.
*               (X7) = TO SQUARE.
*               (X5) = 1.
*
*        USES   ALL REGISTERS EXCEPT X7.
*



 PROACS  EQ     *+400000B
         SA2    SQ2BT+X6
         SA3    SQ2BT+X7
         UX2    X2,B2
         UX3    X3,B3
         SA2    CSTAT+X2
         LX4    X5,B2
         BX4    X2*X4
         ZR     X4,PROACS1        IF NOT FROM SQUARE
         SX6    X6+4              CONVERT FROM SQ TO CASTLE CLEAR MASK
         MX0    60-4
         BX6    -X0*X6
         LX6    2
         SB4    X6
         SA1    PROACSB           PACKED BOARD CASTLE STATUSES
         LX1    X1,B4             POSITION PROPER MASK
         BX6    -X0*X1            EXTRACT MASK
         SA1    REPPN
         LX6    20                POSITION OVER PACKED BOARD STATUS
         SA1    X1                FIRST WORD OF PACKED BOARD
         BX6    -X6*X1            SET NEW STATUS
         SA6    A1
         SA1    PROACSA
         BX6    -X4*X2            CLEAR STATUS BIT
         SA6    A2                UPDATE CSTAT
         LX4    X1,B2
         BX6    X4*X6             CHECK OTHER ROOK BIT
         NZ     X6,PROACS1        IF OTHER ROOK NOT MOVED
         SA6    A2+0
 PROACS1 SA3    CSTAT+X3
         LX4    X5,B3
         BX4    X3*X4
         ZR     X4,PROACS2        IF NOT TO SQUARE
         SX6    X7+4              OPERATE ON PACKED BOARD STATUS
         MX0    60-4
         BX6    -X0*X6
         LX6    2
         SB4    X6
         SA1    PROACSB           PACKED BOARD CASTLE STATUSES
         LX1    X1,B4
         BX6    -X0*X1
         SA1    REPPN
         LX6    20
         SA1    X1
         BX6    -X6*X1
         SA6    A1
         SA1    PROACSA
         BX6    -X4*X3
         SA6    A3
         LX4    X1,B3
         BX6    X4*X6
         NZ     X6,PROACS2
         SA6    A3+0
         EQ     PROACS

 PROACS2 EQU    PROACS            RETURN

 PROACSA VFD    6/0,1/1,45/0,1/1,6/0,1/0  OTHER ROOK SQUARES
 PROACSB VFD    8/0,4/01B,4/10B,12/0,4/14B,8/0,4/04B,4/02B,8/0,4/03B
 PROMOT  SPACE  4,88
***      PROMOT - PROMOTE A PAWN.
*
*        ENTRY  (X6) = FROM-SQUARE NUMBER.
*               (X2) = Y*** **** **** **** ****
*               Y = TYPE OF PIECE TO PROMOTE TO * 2.
*
*        EXIT   (NBOARD+X6) = PROMOTED PIECE.
*               (TPLOC+(MSIDE)) = SQUARE LIST WITH PAWN REMOVED.
*               (MBVAL) = MATERIAL BALANCE SIGNATURE.
*               (MBSCR) = EXTRACTED MATERIAL SCORE.
*
*        USES   ALL REGISTERS.
*        CALLS  MBHASH.
*



 PROMOT  EQ     *+400000B
         SA3    MSIDE
         SA4    SQ2BT+X6
         UX4    X4,B2
         SX1    X6                SAVE FROM SQUARE NUMBER
         SB3    X3+TPLOC          LOCATION OF PAWN LOC SQUARE LIST
         SX0    B1
         LX6    X0,B2             POSITION PAWN BIT
         SA4    B3+X4
         BX6    -X6*X4            REMOVE PAWN
         SA6    A4                UPDATE PAWN LOC SQUARE LIST
         MX5    58
         LX2    2                 POSITION YY
         BX2    -X5*X2            EXTRACT YY
         SX7    X2+B1             PIECE TYPE - 1
         SX6    X7+B1             PIECE TYPE
         AX3    59                EXTEND SIGN OF SIDE ON MOVE
         BX6    X3-X6             PUT ON SIDE
         SA6    NBOARD+X1         PUT PROMOTED PIECE ON NBOARD
         SX4    30
         LX7    2                 TYPE*4-4
         BX4    -X3*X4            30 IF WHITE, 0 IF BLACK
         SB2    X4
         LX6    X0,B2
         SA2    MBVAL
         IX2    X2-X6             REMOVE PAWN
         SB2    X7
         LX6    X6,B2
         IX1    X2+X6             ADD PIECE
         RJ     =XMBHASH          GET MATERIAL BALANCE
         SA6    MBVAL             UPDATE MATERIAL BALANCE WORD
         SA7    MBSCR             AND SCORE
         EQ     PROMOT            RETURN
 TRSRCH  SPACE  4,88
***      TRSRCH - STEP OUT INTO THE TREE.
*
*        ENTRY  (INDEX) = POINTER TO MOVE.
*
*        EXIT   (X6) = MINIMAX DEFICIT, IF MOVE IS LEGAL.
*               (X7) = REFUTATION MARGIN.
*               (X6) = 3777777777777777777B IF MOVE IS ILLEGAL.
*
*        CALLS  UPDATE, EVALU8.
*        USES   ALL REGISTERS.
*



         ENTRY  TRSRCH.
 TRSRCH. SA3    SWICH
         SA1    PONDR
         LX1    59-S.THQ
         NG     X1,TRSRCH9        IF MUST QUIT PONDR SEARCH
         LX3    59-S.TRC
         PL     X3,TRSRCH1        IF NOT TRACING
         SA1    NPLYC             DEPTH
         SA2    TRCLVL            LAST TRACING DEPTH + 1
         IX1    X1-X2
         PL     X1,TRSRCH1        IF TOO DEEP TO TRACE
         RJ     TRTRAC            RECORD TREE STEP
 TRSRCH1 SA2    SLECS
         PL     X2,TRSRCH2        IF SQUARE LISTS ALREADY SAVED
         SB3    SQRLSTL           NUMBER OF WORDS
         SA0    SQRLST            CM FWA
         MX6    1
         BX6    -X6*X2            CLEAR SL WRITTEN FLAG
         SA6    A2
         BX0    X6                ECS ADDRESS
         RJ     =XCHEKFE
         WE     B3                SAVE SQUARE LISTS
         RJ     =XECWERR          IF ERROR

 TRSRCH2 SA1    INDEX             POINTER TO MOVE
         SA1    X1+0              MOVE
         RJ     =XUPDATE          UPDATE ALL SQUARE LISTS.
         SA1    MSIDE
         LX2    X1,B1             *2
         IX2    X1+X2             *3
         LX2    1                 *6
         SB3    X1
         SA1    INDEX
         SB2    X2
         SA1    X1                THE MOVE
         SA2    TPLOC+X2          KING OF MOVING SIDE
         SA3    A2+B1
         SB4    -B3
         SA4    ALATK+B4          ATTACKS OF OTHER SIDE
         SA5    A4+B1
         BX2    X2*X4
         BX3    X3*X5
         BX2    X3+X2
         ZR     X2,TRSRCH3        IF LEGAL MOVE
         SX6    B1
         LX6    S.ILL
         BX4    X6+X1
         SX7    -INFIN            SET SCORE TO TERRIBLE
         SX1    B3                MSIDE
         AX1    59
         BX5    X1-X7             -INFIN FOR WHITE, +INFIN FOR BLACK
         SA1    SLECS
         SX2    SQRLSTL
         EQ     TRSRCH6           DOWN-DATE AND RETURN

 TRSRCH3 SA4    ALATK+B3          ATTACKS OF MOVING SIDE
         SA5    A4+B1
         SB2    -B2
         SA2    TPLOC+B2          KING OF OTHER SIDE
         SA3    A2+B1
         BX2    X2*X4
         BX3    X3*X5
         BX2    X2+X3
         ZR     X2,TRSRCH4        IF NOT CHECK
         SX6    M.CHK
         BX1    X6+X1             SET CHECK BIT

 TRSRCH4 SA5    A1+B1             SCORE WORD
         SX6    NODEBK
         SA0    X6                CM FWA
         SA4    LINDX
         IX6    X4-X6             LENGTH OF VARIABLE AREA
         SB3    X6                NUMBER OF WORDS TO WRITE
         SA2    VAECS
         BX0    X2                ECS FWA
         ERRNZ  VAECS-SLECS-1
         SA3    A2-B1             SLECS
         SX2    SQRLSTL
         MX4    1
         BX7    X3
         ERRNZ  SLECS-PRECS-1
         SA7    A3-B1             PRECS
         IX7    X6+X2
         ERRNZ  PRECS-LPECS-1
         SA7    A7-B1             LPECS
```

```
          IX7    X3+X7
          BX7    X4+X7             FLAG SL NOT WRITTEN
          SA7    A3                SLECS
          IX7    X2+X7
          BX7    -X4*X7
          SA7    A2                VAECS
          RJ     =XCHEKFE          CHECK FE
          WE     B3                SAVE VARIABLE AREA
          RJ     =XECWERR          IF ERROR
          BX6    X1                THE MOVE
          SA6    VMOVE             LEAVE IT FOR EVALU8
          BX7    X5                SCORE WORD
          SA7    SCORE             LEAVE IT FOR EVALU8
          SA2    NPLYC
          SA6    VARIE+X2          SAVE VARIATION
          SX6    MOVES
          SA6    LINDX             INITIALIZE LINDX
          SA1    MSIDE
          BX6    -X1               CHANGE SIDE TO MOVE
          SX0    B1
          SA4    NSRCH+X2          ADVANCE SEARCHED MOVES COUNT
          SX7    X4+B1
          SA7    A4
          SA3    ALLOC+X6
          SA4    ONBRD
          IX7    X0+X2             ADVANCE PLY
          SA6    A1
          AX6    59
          SA7    A2                UPDATE NPLYC
          SA6    SSIDE
          BX6    X0*X6
          SA6    A6-B1             OSIDE
          ERRNZ  SSIDE-OSIDE-1
          SA2    NPLYD             UPDATE PLIES SINCE BASE
          IX7    X0+X2
          SA7    A2
          SA1    A3+B1
          SA2    A4+B1
          BX6    X3
          BX7    X1
          SA6    GENFR
          SA7    A6+B1
          BX6    -X3*X4
          BX7    -X1*X2
          ERRNZ  GENFR+2-GENTO
          SA6    A7+B1
          SA2    STEPHI
          SA7    A6+B1
          SA3    A2+B1             STEPLO
          ERRNZ  STEPLO-STEPHI-1
          SA4    NPLYC
          IX2    X2-X4
          IX3    X4-X3
          BX6    X2+X3
          SA1    ALPHA
          SA2    A1+B1             BETAR
          ERRNZ  BETAR-ALPHA-1
          BX7    X1
          SA7    SALPH             PRESERVE INITIAL VALUES
          BX7    X2                OF ALPHA, BETAR.
          SA7    A7+B1
          ERRNZ  SBETA-SALPH-1
          NG     X6,TRSRCH5        IF NOT STEPPING
          SA1    =5LTSRCH
          RJ     =XDEBUGR
TRSRCH5   RJ     =XEVALU8

**        SAVE POSITION IN TRANSPOSITIONS TABLE IF WARRANTED.
**        HERE THE MN TERM IN THE TRAPRI PRIORITY FUNCTION IS
**        FINALLY COMPUTED.  SEE ROUTINE TRAPRI IN EVALU8.

          SA3    TRAWD             SAVE POSITION IN TRANSPOSITIONS
          SX5    X3                TABLE IF PRIORITY WARRANTS IT.
          NG     X5,TRSRCH5B       IF PRIORITY TOO LOW
          RJ     =XTRASAV          SAVE POSITION IN TABLE
TRSRCH5B  BSS    0
          SA2    NPLYC
          BX6    X6-X6
          SA6    NSRCH+X2          CLEAR LOCAL SEARCHED MOVE COUNT
          SA1    NDTTL             TOTAL NODE COUNT
          MX6    60-5
          BX6    -X6*X1            CHECK LOWER 5 BITS
          NZ     X6,TRSRCH5C       DONT CHECK FOR TYPE-IN TOO OFTEN
          RJ     =XPNDPAU          CHECK FOR TYPE-INS IF PONDERING
TRSRCH5C  SA4    VMOVE             THE MOVE
          SA1    PRECS             ECS ADDRESS OF PREVIOUS PLY
          ERRNZ  SCORE-VMOVE-1
          SA5    A4+B1             SCORE
          ERRNZ  PRECS-LPECS-1
          SA2    A1-B1             LPECS
TRSRCH6   SB3    X2
          BX0    X1
          SA0    SQRLST
          RE     B3                RESTORE NODE BANK AND SQUARE LISTS
          RJ     =XECRERR          IF ERROR
          SA2    INDEX
          BX6    X4                THE MOVE
          BX7    X5                ITS SCORE
          SA6    X2                UPDATE MOVES ARRAY
          SA7    A6+B1
          ERRNZ  59-S.ILL
          NG     X6,TRSRCH9        IF MOVE IS ILLEGAL
          SA1    PONDR
          LX1    59-S.THQ          CHECK PONDR QUIT FLAG
          NG     X1,TRSRCH9        IF SET, PRETEND ILLEGAL MOVE

*         MINIMAX EVALU8 SCORE.

          SA1    ALPHA
          SA4    SSIDE             +0 FOR WHITE, -0 FOR BLACK
          ERRNZ  BETAR-ALPHA-1
          SA2    A1+B1             BETAR
          IX1    X1-X7             ALPHA - SCORE
          IX2    X7-X2             SCORE - BETAR
          BX0    X1-X2             SWAP DIFFERENCES IF BLACK TO MOVE
          BX0    X4*X0
          BX4    X0-X1             MINIMAX DEFICIT
          BX5    X0-X2             REFUTATION MARGIN
          ERRNZ  SSIDE-OSIDE-1
          SA3    A4-B1             OSIDE
          PL     X4,TRSRCH8        IF NO NEW BEST SCORE
          BX0    X6                CHECK FOR NON-PERM BIT IN SEARCH MODE
          LX0    59-S.NPM
          NG     X0,TRSRCH8        IF PERMANENT RESULTS NOT WANTED
          SA7    ALPHA+X3          ELSE STORE NEW BEST

*         PROCESS LIMB ARRAY.

          SA1    NPLYC             CURRENT PLY NUMBER
          SA6    STACK1            STORE MOVE IN NEW BACKED UP LIMB
          SA2    LIMBL+1+X1        LENGTH OF LIMB TO COPY BACK
          SX7    X2+B1             ADD ONE FOR SEARCHED MOVE
          SA7    A2-B1             STORE AS LENGTH OF NEW LIMB
          SB3    X2                WORD COUNT FOR ECS READ
          SA0    A6+B1             CM ADDRESS = STACK1+1
          ZR     X2,TRSRCH7        IF HIGHER LIMB EMPTY
          SA2    LIMBA+1+X1        ECS ADDRESS OF LIMB TO COPY
          BX0    X2
          RE     B3                COPY HIGHER LIMB TO CM
          RJ     =XECRERR          IF ECS ERROR
TRSRCH7   SB3    B3+B1             WORD COUNT OF NEW LIMB
          SA0    A0-B1             CM ADDRESS OF NEW LIMB
          SA2    LIMBA+X1          ECS ADDRESS TO WRITE NEW LIMB
          BX0    X2
          WE     B3                WRITE OUT NEW LIMB
          RJ     =XECWERR          IF ECS ERROR
TRSRCH8   BX6    X4                RETURN MINIMAX DEFICIT
          BX7    X5                RETURN REFUTATION MARGIN
          EQ     TRSRCH            RETURN

*         ILLEGAL MOVE.

TRSRCH9   MX7    1                 RETURN NEGATIVE REFUTATION MARGIN
          BX6    -X7               RETURN OUT OF RANGE MINIMAX DEFICIT
          EQ     TRSRCH            RETURN
TRTRAC    SPACE  4,88
***       TRTRAC - PRINTOUT TREE TRACE.
*
*         ENTRY  (NBOARD) = POSITION BEFORE MOVE.
*                ((INDEX)) = MOVE.
*
*         CALLS  WRLINE, ENGGEN, RDRCOD, RDRCDD.
*


TRTRAC    EQ     *+400000B
          SA1    NPLYC
          SB5    X1
          GT     B5,B1,TRTRAC2     IF PLY2 OR BEYOND
          RJ     =XTRTRFL          ELSE SHOW DEMARCATION
          SA2    NPLYC
          SA1    NSRCH+X2
          SB5    X2
          RJ     =XRDRCOD          CONVERT MOVE ORDINAL
          EQ     B5,B1,TRTRAC1     IF AT PLY 1
          LX6    3*6
TRTRAC1   SA6    TRTRACB
          SA1    ALPHA
          RJ     =XRDRCDD
          SA6    TRTRACB+1         ALSO WRITE ALPHA
          SA1    BETAR
          RJ     =XRDRCDD
          SA6    TRTRACB+2         AND BETAR
          SX6    TRTRACB+3
          SA6    TRTRACA           SET POINTER
          RJ     =XTRTRFL          WRITE MOVE ORDINAL
TRTRAC2   BSS    0
          SA5    TRTRACA
          SX2    X5-TRTRACC
          NZ     X2,TRTRAC3        IF NOT FULL LINE
          SX5    TRTRACB
          SX6    X5
          SA6    A5                RESET POINTER
          RJ     =XWRLINE          OUTPUT LINE
TRTRAC3   SA1    INDEX
          SB2    NBOARD
          SA1    X1                THE MOVE
          RJ     =XENGGEN
          SA5    TRTRACA           POINTER
          SA6    X5+B1             STORE SECOND WORD
          SX7    A6+B1
          SA7    A5                ADVANCE POINTER
          SA1    NPLYC
          RJ     =XRDRCOD          CONVERT PLY NUMBER
          SA1    TRSRCH
          LX1    30
          SB2    X1                CALLING ADDRESS + 1
          SA2    B2-B1             TRACE WORD
          MX7    30
          BX2    -X7*X2            TRACE NAME
          LX6    36
          BX6    X7*X6
          BX6    X6+X2             COMBINE PLY AND NAME
          SA6    X5
          EQ     TRTRAC

TRTRACA   CON    TRTRACB
TRTRACB   BSS    12
TRTRACC   DATA   0
TRTRFL    SPACE  4,16
***       TRTRFL - TREE TRACE FLUSH.
*
*         CALLS  WRLINE.
*


          ENTRY  TRTRFL
TRTRFL    EQ     *+400000B
          SA1    TRTRACA
          SX2    X1-TRTRACB
          ZR     X2,TRTRFL         IF NOTHING TO FLUSH
          BX6    X6-X6
          SA6    X1                STORE ZERO BYTE
          SX5    TRTRACB
          SX6    X5
          SA6    A1                RESET TRTRACA
          RJ     =XWRLINE          WRITE LAST LINE
          EQ     TRTRFL
MBHASH    SPACE  4,88
***       MBHASH - MAINTAIN MATERIAL BALANCE HASH TABLE.
*
*         ENTRY  (X1) = MATERIAL BALANCE SIGNATURE.
*
*         EXIT   (X6) = MATERIAL BALANCE VALUE AND SIGNATURE.
*                (X7) = EXTRACTED SCORE.
*                (MBHSH) UPDATED IF NECESSARY.
*
*         CALLS  MBEVAL.
*         USES   ALL REGISTERS EXCEPT X3.
*


MBHASH    EQ     *+400000B
          SA5    MBMSK             10/-0,20/0,10/-0,20/0
```

```
         BX1    -X5*X1           REMOVE VALUES, IF ANY
         MX7    30
         BX2    X7*X1            EXTRACT WHITE PIECES
         LX2    30
         BX7    -X7*X1           EXTRACT BLACK PIECES
         IX4    X7+X2            SUM
         IX0    X7-X2            DIFFERENCE
         BX6    X0
         AX6    59
         BX0    X0-X6            ABSOLUTE VALUE OF DIFFERENCE
         BX4    X4-X0
         MX7    -7
         SB2    7
         AX0    X4,B2
         AX6    X0,B2
         BX6    X6-X0
         BX6    X6-X4
         BX6    -X7*X6           HASH
         SB7    X6
         SA2    B7+MBHSH         PROBE HASH TABLE
         BX4    X2-X1
         BX4    -X5*X4
         BX6    X2
         ZR     X4,MBHASH1       IF ENTRY IN TABLE
         MX7    8
         LX2    30
         BX4    X7
         LX4    30
         BX7    X4+X7            MASK FOR UPPER AND LOWER SCORE FIELDS
         BX4    X2-X1
         BX6    -X6*X7           COMPLEMENTED SCORE FROM TABLE
         BX2    -X7*X2           CLEAR SCORE FIELDS
         BX4    -X5*X4
         BX6    X2+X6            INSERT COMPLEMENTED SCORE
         ZR     X4,MBHASH1       IF CONVERSE IN TABLE
         RJ     =XMBEVAL         ELSE EVALUATE MATERIAL BALANCE
         SA6    B7+MBHSH         UPDATE TABLE
MBHASH1  MX7    8                NOW EXTRACT SCORE
         BX2    X7*X6            UPPER SCORE BITS
         LX7    30
         BX7    X7*X6            GET LOWER SCORE BITS
         AX2    60-8-8           POSITION UPPER BITS
         AX7    30-8             POSITION LOWER BITS
         BX7    X2+X7            MERGE TO FORM SCORE
         EQ     MBHASH           RETURN
MBCAPT   SPACE  4,88
***      MBCAPT - MATERIAL BALANCE CHANGE FOR CAPTURE.
*
*        ENTRY  (X3) = SQUARE NUMBER OF PIECE BEING REMOVED.
*               (NBOARD+X3) = TYPE OF PIECE BEING REMOVED.
*               (MBVAL) = PREVIOUS SIGNATURE AND VALUE.
*
*        EXIT   (X6) = NEW SIGNATURE AND VALUE.
*               (X7) = EXTRACTED VALUE.
*
*        CALLS  MBHASH.
*        USES   ALL REGISTERS EXCEPT X3.
*



         ENTRY  MBCAPT
MBCAPT   EQ     *+400000B
         SA1    NBOARD+X3        TYPE OF PIECE BEING REMOVED
         SX5    30
         BX2    X1
         AX2    59               SIGN(TYPE)
         BX1    X2-X1            ABS(TYPE)
         SX6    B1
         BX5    -X2*X5           30 IF WHITE, 0 IF BLACK
         IX1    X1-X6            TYPE-1
         LX1    2                TYPE*4-4
         IX1    X1+X5            SHIFT COUNT
         SB2    X1
         LX6    X6,B2            POSITION TYPE COUNT CHANGE
         SA1    MBVAL            OLD SIGNATURE
         IX1    X1-X6            REMOVE PIECE
         RJ     =XMBHASH         COMPUTE NEW VALUES.
         EQ     MBCAPT           RETURN
MBEXCH   SPACE  4,46
***      MBEXCH - MATERIAL BALANCE CHANGE FOR EXCHANGE.
*
*        ENTRY  (X3) = A CAPTURE MOVE.
*               (NBOARD+(X3,MTO)) = TYPE OF CAPTURED PIECE.
*               (NBOARD+(X3,MFR)) = TYPE OF RECAPTURED PIECE.
*               (MBVAL) = PREVIOUS SIGNATURE AND VALUE.
*
*        EXIT   (X6) = NEW SIGNATURE AND VALUE, ASSUMING BOTH PIECES
*                      ARE REMOVED.
*               (X7) = EXTRACTED SCORE.
*
*        CALLS  MBHASH.
*        USES   ALL REGISTERS.
*



         ENTRY  MBEXCH
MBEXCH   EQ     *+400000B
         MX0    -6
         BX2    -X0*X3           TO SQUARE
         LX3    -6
         BX1    -X0*X3           FROM SQUARE
         SA1    NBOARD+X1        RECAPTURED TYPE
         SA2    NBOARD+X2        CAPTURED TYPE
         SX5    30
         BX3    X1
         AX3    59
         BX1    X3-X1            ABS(RECAPTURED TYPE)
         BX2    -X3-X2           ABS(CAPTURED TYPE)
         SX4    B1+0
         IX1    X1-X4            RECAP-1
         IX2    X2-X4            CAP-1
         LX1    2                4*RECAP-4
         LX2    2                4*CAP-4
         BX6    -X3*X5           30 IF RECAP WHITE, 0 IF BLACK
         BX7    X3*X5            30 IF CAP WHITE, 0 IF BLACK
         IX1    X1+X6            RECAP SHIFT COUND
         IX2    X2+X7            CAP SHIFT COUND
         SB2    X1
         LX6    X4,B2            RECAP BIT
         SB2    X2
         LX4    X4,B2            CAP BIT
         IX6    X4+X6
         SA1    MBVAL
         IX1    X1-X6            DECREMENT BOTH COUNTS
         RJ     =XMBHASH         COMPUTE NEW VALUES
         EQ     MBEXCH           RETURN
PSUPDT   SPACE  4,88
***      PSUPDT - UPDATE PAWN STRUCTURE.
*
*        ENTRY  (PSMOD) = FILES TO UPDATE.  BIT 47 = QR, 46 = QN, ETC.
*
*        CALLS  PSHASH.
*        USES   ALL REGISTERS
*



PSUPDT   EQ     *+400000B
PSUPDT1  SA1    PSMOD
         ZR     X1,PSUPDT        IF NO MORE CHANGES, RETURN
         SX0    B1
         LX0    47
         NX6    X1,B2            FILE NUMBER IN B2
         AX6    X0,B2
         BX6    -X6*X1           CLEAR FILE BIT
         SA6    A1
         SB3    B2+B2            2*FILE
         SB3    B2+B3            3*FILE
         SB3    B3+B3            6*FILE
         MX0    18
         SA1    WPSIG
         ERRNZ  BPSIG-WPSIG-1
         SA2    A1+B1
         LX1    X1,B3
         LX2    X2,B3
         BX1    X1*X0
         BX2    X2*X0
         LX2    30
         BX1    X1+X2
         LX1    -12
         RJ     =XPSHASH         GET VALUE OF THIS STRUCTURE
         SA6    PSQRF+B2         SAVE SIGNATURE
         EQ     PSUPDT1          PROCESS NEXT FILE
PSHASH   SPACE  4,88
***      PSHASH - MAINTAIN PAWN STRUCTURE HASH TABLE.
*
*        ENTRY  (X1) = PAWN STRUCTURE SIGNATURE.
*
*        EXIT   (X6) = PAWN STRUCTURE SIGNATURE AND CODES.
*               (PSHSH) UPDATED IF NECESSARY.
*
*        CALLS  PSEVAL.
*        USES   ALL REGISTERS EXCEPT B2.
*



PSHASH   EQ     *+400000B
         SA5    PSMSK            12/-0,18/0,12/-0,18/0
         MX0    30
         BX1    -X5*X1           REMOVE CODES, IF ANY
         BX2    -X0*X1           BLACK PAWNS
         BX3    X0*X1            WHITE PAWNS
         LX3    30
         BX6    X1+X2
         BX7    X6
         AX7    9
         BX6    X6+X7
         MX7    -9
         BX4    -X7*X6
         SX0    EC.PSH+X4        ECS ADDRESS
         SB3    B1
         SA0    STACK1
         RE     B3
         RJ     =XECRERR
         SB3    X4
         SA2    A0
         BX3    X1-X2
         BX3    -X5*X3
         BX6    X2
         ZR     X3,PSHASH        IF IN TABLE
         LX2    30
         LX6    30
         BX3    X1-X2
         BX3    -X5*X3
         ZR     X3,PSHASH        IF CONVERSE IN TABLE
         RJ     =XPSEVAL         ELSE EVALUATE PAWN STRUCTURE
         SA6    STACK1
         SX0    B3+EC.PSH
         SA0    A6
         SB3    B1
         WE     B3
         RJ     =XECWERR
         SA2    A0
         BX6    X2
         EQ     PSHASH           RETURN
TRACKT   SPACE  4,20
***      TRACKT - LOOK UP TRANSPOSITIONS HASH TABLE.
*
*        CALLED BY EVALU8.
*
*        ENTRY  (REPPN) = POINTER TO PACKED BOARD.
*               (TRASH) = HASH VALUE FOR TRANSPOSITIONS TABLE.
*
*        FORMAT OF TABLE ENTRY:
*               WORD 1: VFD 12/MOVE,12/SCORE,2/FLAG,4/IT,5/PLY,1/SIDE,
*               4/CAST,1/EPF,3/EPCOL,4/SQ00,4/SQ01,4/SQ02,4/SQ03.
*               WORDS 2-5: SQ04 THRU SQ77, AS IN OTHER PACKED BOARDS.
*               MOVE = LOWER 12 BITS OF BEST MOVE FROM THIS POSITION.
*               SCORE = SCORE OF POSITION CONFINED TO 12 BITS.
*               FLAG = 00 (BINARY) IF NO SCORE WAS STORED,
*                      01 IF SCORE IS UPPER BOUND ON TRUE SCORE,
*                      10 IF SCORE IS LOWER BOUND ON TRUE SCORE,
*               AND    11 IF SCORE IS TRUE SCORE.
*               IT = VALUE OF ITERS, PLY = NPLYC, SIDE = OSIDE.
*               CAST  IS 4 BITS: WQ,WK,BQ,BK.  EACH BIT IS ON IF
*               CORRESPONDING CASTLING IS LEGAL.
*               EPF = 1 FOR AN ENPASSANT PAWN, EPCOL = ITS COLUMN NO.
*
*        EXIT   (TRAWD) = FIRST WORD OF ENTRY POINTED TO BY TRASH,
*               EXCEPT LOWER 18 BITS ARE REPLACED BY ENTRY CLOBBER
*               PRIORITY.  ALSO, IF NO MATCH, UPPER 12 BITS (MOVE
*               FIELD) ARE CLEARED.
*               (X6) = (TRAWD) IF MATCH, ELSE (X6) = 0.
*
*        CALLS  TRAPRI.
*
*        USES   ALL A AND X REGISTERS, B3, STACK1 THRU STACK1+LE.TRA-1.

         ENTRY  TRACKT

TRACKT1  SX6    -B1              TELL TRAPRI ABOUT NO MATCH
         RJ     =XTRAPRI         CALCULATE CLOBBERING PRIORITY
         UX6    X6               CLEAR MOVE FIELD SINCE NO MATCH
         SA6    TRAWD            FIRST WORD OF ENTRY AND PRIORITY
         BX6    X6-X6            SHOW NO MATCH
```

```
 TRACKT   EQ     *+400000B         ENTRY/EXIT
          SA1    TRASH             HASH
          UX1,B3 X1                B3 = LOG OF NO. OF SLOTS IN TABLE
          SX5    B1
          LX6    X5,B3
          IX6    X6-X5             MASK FOR TABLE INDEX
          BX6    X6*X1             EXTRACT MEANINGFUL BITS
          SB3    LE.TRA
          SX0    EC.TRA+X6         COMPUTE ECS ADDRESS OF ENTRY
          SA0    STACK1            CM ADDRESS TO RECEIVE ENTRY
          LX6    2
          IX0    X0+X6
          ERRNZ  LE.TRA-5
          RE     B3                READ ENTRY
-         RJ     =XECRERR
          SA5    A0                FIRST WORD OF ENTRY
          SA1    REPPN
          SA2    A0+B1             SECOND WORD OF ENTRY
          SA3    X1+B1             SECOND WORD OF CURRENT
          IX3    X3-X2             COMPARE
          NZ     X3,TRACKT1        IF NO MATCH
          SA2    A2+B1             3RD WORD
          SA4    A3+B1
          IX4    X4-X2
          NZ     X4,TRACKT1        IF NO MATCH
          SA2    A2+B1             4TH WORD
          SA3    A4+B1
          IX3    X3-X2
          NZ     X3,TRACKT1        IF NO MATCH
          SA4    A3+B1             5TH WORD
          SA2    A2+B1
          IX4    X4-X2
          NZ     X4,TRACKT1        IF NO MATCH
          SA3    X1                FIRST WORD
          MX0    60-9-16
          BX2    -X0*X5            EXTRACT BOARD AND STATUS ONLY
          IX6    X3-X2
          NZ     X6,TRACKT1        IF NO MATCH
          RJ     =XTRAPRI          CALCULATE CLOBBERING PRIORITY
          SA6    TRAWD
          EQ     TRACKT            MATCH
 TRASAV   SPACE  4,20
***       TRASAV - SAVE ENTRY IN TRANSPOSITIONS TABLE.
*
*         CALLED BY TRSRCH.
*
*         ENTRY  (TRAWD) = TRANSPOSITIONS WORD:
*                VFD 12/MOVE,48/WHOCARES
*                (SCORE) = SCORE TO BE SAVED.
*                (SALPH) = SAVED STARTING ALPHA.
*                (SBETA) = SAVED STARTING BETAR.
*                (REPPN) = POINTER TO PACKED BOARD TO BE SAVED.
*                (TRASH) = TRANSPOSITIONS HASH WORD.
*
*         EXIT   PACKED BOARD WITH SCORE AND MOVE (OR ZERO IF NO MOVE)
*                IS SAVED, UNLESS SCORE IS MORE THAN 12 BITS LONG.

          ENTRY  TRASAV

 TRASAV   EQ     *+400000B         ENTRY/EXIT

*         COMPUTE SCORE BOUNDS FLAGS.

          SA1    SALPH
          SA2    SCORE
          SA3    A1+B1             SBETA
          BX7    X2
          MX0    1
          AX7    11                CHECK MAGNITUDE OF SCORE
          NZ     X7,TRASAV         IF TOO LARGE TO FIT, FORGET IT
          IX1    X1-X2             SALPH - SCORE
          IX3    X2-X3             SCORE - SBETA
          BX1    X0*X1             THIS IS .LE. FLAG
          BX3    X0*X3             THIS IS .GE. FLAG

*         SET UP REST OF ENTRY FIRST WORD.

          ERRPL  NPLY-40B          ERROR IF PLY TOO BIG FOR TABLE
          SA5    ITERS
          SA4    NPLYC
          LX1    36                POSITION .LE. FLAG
          LX3    35                POSITION .GE. FLAG
          LX5    30                POSITION ITERS
          LX4    25                POSITION NPLYC
          MX0    60-12             MASK FOR SCORE
          BX6    X1+X3             SCORE BOUNDS FLAGS
          BX5    X4+X5             NPLYC, ITERS
          MX3    12
          SA1    TRAWD             UPPER 12 BITS = MOVE
          BX6    X6+X5
          BX1    X3*X1             MOVE
          BX6    X6+X1             INSERT MOVE
          BX2    -X0*X2            LOWER 12 BITS OF SCORE
          SA3    REPPN             POINTER TO PACKED BOARD
          LX2    36                POSITION SCORE
          SA4    X3                LOAD FIRST WORD OF PACKED BOARD
          BX6    X6+X2             INSERT SCORE
          BX6    X4+X6             COMPLETE ENTRY FIRST WORD

*         WRITE PACKED BOARD TO ECS.

          SA0    X3                CM ADDRESS TO WRITE
          SA6    X3                STORE MODIFIED FIRST WORD
          SA1    TRASH             TRANSPOSITIONS HASH
          UX1,B3 X1                B3 = LOG OF NO. OF SLOTS IN TABLE
          SX0    B1
          LX7    X0,B3
          IX7    X7-X0             MASK FOR TABLE INDEX
          SB3    LE.TRA            NO. OF WORDS TO WRITE
          BX7    X7*X1             EXTRACT MEANINGFUL BITS
          SX0    EC.TRA+X7         COMPUTE ECS ADDRESS OF SLOT
          LX7    2
          IX0    X0+X7
          ERRNZ  LE.TRA-5
          WE     B3                WRITE ENTRY TO TABLE
-         RJ     =XECWERR
          BX6    X4                RESTORE FIRST WORD OF PACKED BOARD
          SA6    X3
          EQ     TRASAV
 ADDLOC   SPACE  4,88
***       ADDLOC - ADD PIECE TO ALL LOCATION LISTS.
*
*         ENTRY  (B3) = SQUARE NUMBER (0 - 77B)
*                (X1) = PIECE TYPE.
*
*         EXIT   (B3) = SQUARE NUMBER.
*                (X1) = PIECE TYPE.
*
*         USES   ALL REGISTERS EXCEPT A0, A2, A5, A7, X0, B5, B6, B7.
*

 ADDLOC   EQ     *+400000B
          SA3    SQ2BT+B3
          UX3    X3,B2
          LX2    X1,B1             TYPE*2
          SB4    X3+TPLOC
          SX5    B1
          SA4    B4+X2
          LX7    X5,B2
          BX6    X7+X4
          SA6    A4                UPDATE PIECE SQUARE LIST.
          AX2    59
          BX2    X2-X5             SIDE
          LX2    1                 2*SIDE
          SB2    X2+0
          SA4    STOPS+X3
          BX6    X7+X4
          SA6    A4                UPDATE STOPS
          SA4    A4+ALLOC-STOPS
          BX6    X7+X4
          SA6    A4                UPDATE ALLOC
          SA4    A4+B2
          BX6    X7+X4
          SA6    A4                UPDATE AXLOC
          SX4    B3                UPDATE TRANSPOSITIONS HASH, WHICH
          BX6    X4-X1             IS THE SUM, OVER ALL THE PIECES ON
          LX6    5                 THE BOARD, OF A WEIRD FUNCTION OF
          MX7    60-4              THE PIECE TYPE AND THE SQUARE
          IX6    X6+X4             IT IS ON.
          LX6    4
          IX6    X6+X1
          IX6    X6+X4
          BX7    -X7*X6
          SB2    X7
          LX6    X6,B2
          SA3    TRASH             TRANSPOSITIONS HASH WORD
          BX7    X6
          LX7    60-16
          IX6    X7+X6
          MX7    60-16
          BX6    -X7*X6
          UX3,B4 X3                B4 = LOG OF NO. OF SLOTS
          IX3    X3+X6             ADD TO PREVIOUS VALUE
          AX6    X6,B4
          IX6    X6+X3             ALSO MAKE UPPER BITS USEFUL
          PX6    X6,B4             RE-PACK IN LOGTRA
          SA6    A3
          BX6    X1
          SA6    NBOARD+B3
          SA3    SQ2PB+B3          SQUARE NUMBER TO PACKED BOARD XLATE
          LX3    30
          UX3    X3,B2             WORD AND BIT IN PACKED BOARD
          SA4    REPPN
          SB4    X3
          SA4    B4+X4             WORD IN PACKED BOARD
          LX6    X6,B2
          IX6    X6+X4
          SA6    A4                UPDATE PACKED BOARD
          AX2    X1,B1
          NZ     X2,ADDLOC1        IF NOT A PAWN
          BX3    X5*X2             0 IF WHITE, 1 IF BLACK
          SB2    B3+B3
          SX4    X3+SQ2PS
          SA4    X4+B2             PAWN STRUCTURE TRANSLATION
          UX4    X4,B2
          SA3    WPSIG+X3          PAWN SIGNATURE WORD(SIDE)
          LX6    X5,B2             PAWN BIT
          BX6    X6+X3             SET PAWN BIT
          SA2    PSMOD             PAWN STRUCTURE MODIFICATIONS
          SA6    A3
          BX6    X4+X2
          SA6    A2                INDICATE FILES CHANGED
          EQ     ADDLOC            RETURN
 ADDLOC1  MX6    -2
          BX6    -X6-X2
 KING     EQU    3*2               ASSUMED
          NZ     X6,ADDLOC         IF NOT KING
          BX3    X5*X6             0 IF WHITE, 1 IF BLACK
          SX6    B3                TO SQUARE
          SA6    WKSQR+X3          UPDATE KING SQUARE
          EQ     ADDLOC            RETURN
 ADDATK   SPACE  4,88
***       ADDATK - ADD ATTACKS ON ALL SQUARE LISTS.
*
*         ENTRY  (X3) = SQUARE NUMBER.
*                (NBOARD) = POSITION.
*
*         USES   ALL REGISTERS EXCEPT B6.
*


 ADDATK   EQ     *+400000B
          SA4    NBOARD+X3         TYPE
          SA5    MVMSK+X4          MOVES OF PIECE
          SB2    X3
          SA3    SQ2BT+X3
          UX3    X3,B3             B3 = BIT, X3 = WORD
          ZR     X5,ADDATK1        IF SWEEP PIECE

*         NON-SWEEP PIECE ATTACK GENERATION.

          MX0    30
          LX0    X0,B3
          LX4    X5,B3
          SB4    B3-59
          MX5    1
          LX5    X5,B4
          LX3    59
          AX3    59
          BX5    -X3-X5
          BX5    X5-X0
          BX6    -X5*X4
          BX7    X5*X4
          EQ     ADDATK8           STORE ATKFR AND BUILD ATKTO

*         SWEEP PIECE ATTACK GENERATION.

 ADDATK1  BX6    X6-X6
          BX7    X7-X7
          SA1    STOPS
          SA2    A1+B1
```

```
          SA4    MVDIR+X4          MOVE DIRECTIONS
          BX0    X4
          SB7    60
          ZR     X0,*+400000B

 ADDATK2  SB5    X3                WORD NUMBER
          LX4    54
          AX4    54
          SB4    X4+B3             BIT NUMBER
 ADDATK3  PL     B4,ADDATK4        IF NOT OFF RIGHT END
          SB5    B5+B1             ADVANCE WORD
          SB4    B4+B7             SET PROPER BIT NUMBER
 ADDATK4  LT     B4,B7,ADDATK5     IF NOT OFF LEFT END
          SB5    B5-B1
          SB4    B4-B7
 ADDATK5  SX5    B1
          LX5    X5,B4
          NZ     B5,ADDATK6        IF IN SECOND WORD
          BX6    X6+X5
          BX5    -X1*X5
          ZR     X5,ADDATK7        IF STOPPED IN THIS DIRECTION
          SB4    B4+X4
          EQ     ADDATK3           CONTINUE IN SAME DIRECTION

 ADDATK6  BX7    X5+X7
          BX5    -X2*X5
          ZR     X5,ADDATK7        IF STOPPED IN THIS DIRECTION
          SB4    B4+X4
          EQ     ADDATK3           CONTINUE IN SAME DIRECTION

 ADDATK7  AX0    6
          BX4    X0
          NZ     X0,ADDATK2        IF NOT DONE IN ALL DIRECTIONS

*         STORE ATKFR AND BUILD ATKTO.
*               X6,X7 = ATKFR
*               B2 = SQUARE NUMBER.
*               X3,B3 = WORD,BIT IN ATKTO

 ADDATK8  SA1    ONBRD
          BX6    X1*X6
          SA2    A1+B1
          BX7    X2*X7
          SB4    B2+B2
          SA6    ATKFR+B4
          SA7    A6+B1
          BX1    X6                SAVE FIRST WORD
          SA4    NBOARD+B2
          AX4    59
          SX5    B1+B1
          BX4    X5-X4
          SX5    B1
          SA4    ALATK+X4
          BX6    X6+X4
          SA6    A4
          SA4    A4+B1
          BX6    X4+X7
          SA6    A4
          BX3    X3*X5             WORD NUMBER
          LX5    X5,B3             BIT
          SB4    X3+ATKTO
          MX0    1
          LX0    48
          ZR     X1,ADDATK10       IF NOTHING IN FIRST WORD
          PX3    X1
          SB3    12+BT2SQ
          NX6    X3,B2
          AX6    X0,B2
 ADDATK9  SA4    B3+B2             SQUARE NUMBER
          BX3    -X6*X3            CLEAR BIT
          LX4    1
          SA4    B4+X4
          BX6    X4+X5
          NX4    X3,B2
          SA6    A4
          AX6    X0,B2
          NZ     X6,ADDATK9
          ZR     X7,ADDATK         IF DONE WITH BOTH WORDS, RETURN
 ADDATK10 LX7    48
          PX3    X7
          SX7    0
          SB3    60+BT2SQ
          NX6    X3,B2
          AX6    X0,B2
          NZ     X6,ADDATK9        PROCESS SECOND WORD
          EQ     ADDATK            IF NOTHING IN EITHER WORD, RETURN

 DELATK   SPACE  4,88
***       DELATK - DELETE ATTACKS FROM ALL SQUARE LISTS.
*
*         ENTRY  (X3) = SQUARE NUMBER (0 - 77B)
*
*         EXIT   (B3) = SQUARE NUMBER (0 - 77B)
*
*         USES   STACK1, ALL REGISTERS.
*


 DELATK   EQ     *+400000B
          SA1    NBOARD+X3         TYPE
          SA2    SQ2BT+X3
          BX6    X3-X1             UPDATE TRANSPOSITIONS HASH, WICH IS
          LX6    5                 THE SUM, OVER ALL THE SQUARES ON THE
          MX7    60-4              BOARD, OF A WEIRD FUNCTION OF THE
          IX6    X6+X3             PIECE TYPE AND THE SQUARE IT IS ON.
          LX6    4
          IX6    X6+X1
          IX6    X6+X3
          BX7    -X7*X6
          SB3    X7
          LX6    X6,B3
          SA5    TRASH             TRANSPOSITIONS HASH WORD
          BX7    X6
          LX7    60-16
          IX6    X7+X6
          MX7    60-16
          BX6    -X7*X6
          UX5,B4 X5                B4 = LOG OF NO. OF SLOTS
          IX5    X5-X6             SUBTRACT FROM PREVIOUS VALUE
          AX6    X6,B4
          IX6    X5-X6             ALSO MAKE UPPER BITS USEFUL
          PX6    X6,B4             RE-PACK IN LOGTRA
          SA6    A5
          SA4    SQ2PB+X3
          LX4    30
          UX4    X4,B3             WORD AND BIT IN PACKED BOARD
          SA5    REPPN
          SB4    X4
          SA5    B4+X5             WORD IN PACKED BOARD
          LX6    X1,B3
          IX6    X5-X6
          SA6    A5                UPDATE PACKED BOARD
          UX2    X2,B2             X2 = WORD, B2 = BIT
          SX7    B1
          LX0    X7,B2
          SB3    X3                SAVE SQUARE NUMBER
          SB7    B3+B3             2 * SQUARE NUMBER
          SB4    X2+TPLOC
          LX3    X1,B1             2*TYPE
          SA4    B4+X3
          BX6    -X0*X4
          SA6    A4                UPDATE TPLOC
          SA4    B4-TPLOC+STOPS
          BX6    -X0*X4
          SA6    A4                UPDATE STOPS
          SA4    A4-STOPS+ALLOC
          BX6    -X0*X4
          SA6    A4                UPDATE ALLOC
          AX3    59
          BX5    X7*X3             0 IF WHITE, 1 IF BLACK
          BX3    X3-X7             +1 IF WHITE, -1 IF BLACK
          LX3    1
          SB5    X3                2 * SIDE
          SA4    A4+B5
          BX6    -X0*X4
          SA6    A4                UPDATE AXLOC
          SB4    STACK1
          SB6    X5+B7             INDEX INTO SQ2PS
          SA4    SQ2PS+B6
          UX4    X4,B2
          SA5    WPSIG+X5
          LX6    X7,B2             PAWN STRUCTURE BIT
          BX3    X5*X6
          ZR     X3,DELATK0        IF NO PAWN STRUCTURE CHANGE
          BX6    -X6*X5
          SA6    A5                UPDATE PAWN STRUCTURE SIGNATURE
          SA5    PSMOD
          BX6    X4+X5
          SA6    A5                INDICATE PAWN CHANGES
 DELATK0  SA4    ATKFR+B7
          SA5    A4+B1
          BX6    X6-X6
          SA6    A1                CLEAR NBOARD
          SA6    A4                CLEAR ATKFR
          SA6    A5
          LX7    47
          SB6    ATKTO+X2
          ZR     X4,DELATK2        IF NO ATTACK IN FIRST WORD
          SB7    12+BT2SQ
          PX4    X4
          NX6    X4,B2
 DELATK1  AX6    X7,B2
          BX4    -X6*X4            CLEAR BIT
          SA3    B7+B2             SQUARE NUMBER
          LX6    X3,B1             2*SQUARE NUMBER
          SA6    B4                STACK 2*SQUARE NUMBER
          SB4    B4+B1
          SA3    B6+X6             ATKTO(ATTACKED SQUARE)
          BX6    -X0*X3
          SA6    A3
          NX6    X4,B2
          NZ     X6,DELATK1        IF THIS WORD ALL PROCESSED
 DELATK2  ZR     X5,DELATK3        IF NO ATTACKS IN SECOND WORD
          SB7    60+BT2SQ
          LX5    48
          PX4    X5
          BX5    X5-X5
          NX6    X4,B2
          EQ     DELATK1           PROCESS SECOND WORD

 DELATK3  SB7    STACK1
          EQ     B7,B4,DELATK7     IF NO ATTACKS ON SQUARE
          SA4    B5+ALATK
          BX6    X4
          SA5    A4+B1
          SA4    A4+ALLOC-ALATK
          BX7    X5
          SA5    A4+B1
          SX0    B1
          SB5    B5+ALATK
 DELATK4  SA1    B4-B1
          SB4    B4-B1
          SA2    X1+ATKTO
          SA3    A2+B1
          BX2    X2*X4
          BX3    X3*X5
          BX2    X2+X3
          NZ     X2,DELATK6        IF ANOTHER ATTACK BY THIS SIDE
          AX1    1
          SA2    SQ2BT+X1
          UX2    X2,B2
          LX3    X0,B2
          NZ     X2,DELATK5        IF IN SECOND WORD
          BX6    -X3*X6
          EQ     DELATK6

 DELATK5  BX7    -X3*X7
 DELATK6  NE     B4,B7,DELATK4     IF NOT DONE
          SA6    B5
          SA7    A6+B1
          EQ     DELATK            RETURN

 DELATK7  EQU    DELATK            RETURN
 PRPATK   SPACE  4,88
***       PRPATK - PROPAGATE ATTACKS THROUGH A SQUARE.
*
*         ENTRY  (B3) = SQUARE NUMBER (0 - 77B)
*
*         USES   ALL REGISTERS.
*


 PRPATK   EQ     *+400000B
          SB7    B3+B3             2*SQUARE NUMBER
          SA1    ATKTO+B7
          SX0    B1
          LX0    47
          SB7    B0                WORD NUMBER OF BIT
          SA2    EXPND+B3
          SB6    X2                10X12 SQUARE NUMBER
          ZR     X1,PRPATK4        IF NO ATTACKS IN FIRST WORD
          SA0    12+BT2SQ
          NX7    X1,B2
          AX7    X0,B2
```

```
PRPATK1  SA3    A0+B2           8X8 SQUARE NUMBER
         BX1    -X7*X1
         SA4    NBOARD+X3
         SA5    MVMSK+X4
         NZ     X5,PRPATK3      IF NOT SWEEP PIECE
         SA2    EXPND+X3        10X12 SQUARE NUMBER
         SB2    X2
         SB2    B6-B2
         SA2    DIRTB+B2        DIRECTION
         LX3    1
         SB3    X3+ATKFR
         AX4    59
         SX6    B1+B1
         BX4    X6-X4           SIDE*2
         SB4    X4+ALATK
         SB5    X2+B6           TEST SQUARE (10X12)
         ZR     B7,PRPATK2      IF NOT IN SECOND WORD
         LX7    12              ELSE ADJUST ATTACKING BIT
PRPATK2  SA5    BT2SQ+B5
         NG     X5,PRPATK3      IF OFF BOARD
         LX6    B1,X5
         SB2    X6+B7
         SA3    B2+ATKTO
         BX6    X7+X3
         SA6    A3
         SA3    SQ2BT+X5
         UX3    X3,B2
         LX4    X0,B2
         LX4    13
         SA5    X3+B4
         BX6    X4+X5
         SA6    A5              UPDATE ALATK
         SA5    B3+X3
         BX6    X4+X5
         SA6    A5              UPDATE ATKFR
         SA5    STOPS+X3
         BX6    X5*X4
         SB5    B5+X2
         ZR     X6,PRPATK2      IF NOT STOPPED
PRPATK3  NX7    X1,B2
         AX7    X0,B2
         NZ     X7,PRPATK1      IF NOT DONE WITH WORD
PRPATK4  NZ     B7,PRPATK5      IF DONE WITH SECOND WORD
         SA1    A1+B1
         SB7    B1
         ZR     X1,PRPATK5      IF NOTHING IN SECOND WORD
         SA0    60+BT2SQ
         LX1    48
         NX7    X1,B2
         AX7    X0,B2
         EQ     PRPATK1         PROCESS SECOND WORD

PRPATK5  EQU    PRPATK          RETURN

 CUTATK  SPACE  4,88
***      CUTATK - STOP PROPAGATION OF ATTACKS THROUGH A SQUARE.
*
*        ENTRY  (B3) = SQUARE NUMBER.
*
*        USES   ALL REGISTERS.
*


 CUTATK  EQ     *+400000B
         SB7    B3+B3           2*SQUARE NUMBER
         SA1    ATKTO+B7
         SX0    B1
         LX0    47
         SB7    B0              WORD NUMBER OF BIT
         SA2    EXPND+B3
         SB6    X2              10X12 SQUARE NUMBER
         ZR     X1,CUTATK4      IF NO ATTACKS IN FIRST WORD
         PX1    X1
         SA0    12+BT2SQ
         NX7    X1,B2
         AX7    X0,B2
CUTATK1  SA3    A0+B2           8X8 SQUARE NUMBER
         BX1    -X7*X1          CLEAR BIT
         SA4    NBOARD+X3
         SA5    MVMSK+X4
         NZ     X5,CUTATK3      IF NOT SWEEP PIECE
         SA2    EXPND+X3        10X12 SQUARE NUMBER
         SB2    X2
         SB2    B6-B2
         SA2    DIRTB+B2        DIRECTION
         LX3    1
         SB3    X3+ATKFR
         AX4    59
         SX6    B1+B1
         BX4    X6-X4           SIDE*2
         SB4    X4+ALLOC
         SB5    X2+B6           TEST SQUARE (10X12)
         ZR     B7,CUTATK2      IF NOT IN SECOND WORD
         LX7    12              ELSE ADJUST BIT
CUTATK2  SA5    BT2SQ+B5
         NG     X5,CUTATK3      IF OFF BOARD
         LX6    X5,B1
         SB2    X6+ATKTO
         SA4    B2+B7
         BX6    X4*X7
         ZR     X6,CUTATK3      IF NOT ATTACKED
         BX6    -X7*X4
         SA6    A4              UPDATE ATKTO
         SA5    SQ2BT+X5
         UX5    X5,B2
         LX4    X0,B2
         LX4    13
         SA3    B3+X5
         BX6    -X4*X3
         SA6    A3
         SB2    X5+B4
         SA3    A4-B7
         SA5    B4
         BX6    X3*X5
         SA3    A3+B1
         SA5    A5+B1
         BX5    X3*X5
         BX6    X5+X6
         SB5    B5+X2           ADVANCE SQUARE
         NZ     X6,CUTATK2      IF ANOTHER ATTACK BY SAME SIDE
         SA3    B2+ALATK-ALLOC
         BX6    -X4*X3
         SA6    A3              CLEAR ALATK BIT
         EQ     CUTATK2

CUTATK3  NX7    X1,B2
         AX7    X0,B2
         NZ     X7,CUTATK1      IF NOT DONE WITH WORD
CUTATK4  NZ     B7,CUTATK5      IF DONE WITH SECOND WORD
         SB7    B1
         SA1    A1+B1
         ZR     X1,CUTATK5      IF NOTHING IN SECOND WORD
         SA0    60+BT2SQ
         LX1    48
         PX1    X1
         NX7    X1,B2
         AX7    X0,B2
         EQ     CUTATK1         PROCESS SECOND WORD

CUTATK5  EQU    CUTATK
 GENALL  SPACE  4,88
***      GENALL - GENERATE ALL MOVES, CAPTURES, AND CASTLES.
*
*        ENTRY  (GENFR) = SL OF PIECES YET TO MOVE.
*
*        USES   ALL REGISTERS.
*        CALLS  GENFSL, GENCAS.
*


         ENTRY  GENALL
 GENALL  EQ     *+400000B
         SA1    GENFR
         SA2    A1+B1
         BX3    X1+X2
         ZR     X3,GENALL       IF ALREADY GENERATED
         RJ     =XGENFSL        GENERATE MOVES AND CAPTURES
         EQ     GENALL          RETURN
 GENCAP  SPACE  4,88
***      GENCAP - GENERATE ALL CAPTURES.
*
*        ENTRY  (MSIDE) = SIDE TO MOVE.
*
*        CALLS  GENTSL, GENFPN.
*


         ENTRY  GENCAP
 GENCAP  EQ     *+400000B
         SA1    MSIDE
         BX6    -X1
         SA1    ALLOC+X6        OPPONENTS PIECES
         SA2    A1+B1
         RJ     =XGENTSL        GENERATE SIMPLE CAPTURES
         SA1    ENPAS
         SA2    A1+B1
         BX3    X1+X2
         ZR     X3,GENCAP       IF NO EN PASSANT SQUARE
         RJ     =XGENFPN        GENERATE EN PASSANT CAPTURE
         EQ     GENCAP          RETURN
 GENFSL  SPACE  4,88
***      GENFSL - GENERATE ALL MOVES FROM A SET OF SQUARES.
*
*        ENTRY  (X1,X2) = SQUARE LIST.
*               (LINDX) = LOCATION TO STORE FIRST MOVE.
*
*        EXIT   (LINDX) = LOCATION BEYOND LAST STORED MOVE.
*
*        USES   ALL REGISTERS.
*


         ENTRY  GENFSL
 GENFSL  EQ     *+400000B
         MX0    1
         LX0    48
         SA4    GENFR
         SA5    A4+B1
         BX6    -X1*X4
         BX7    -X2*X5
         SA6    A4              REMAINING VALID FROM SQUARES
         SA7    A4+B1
         BX1    X1*X4
         BX2    X2*X5
         SA3    MSIDE
         LX4    X3,B1           *2
         IX4    X3+X4           *3
         LX4    1               *6
         ERRNZ  KING-6
         SA5    TPLOC+X4        KINGS
         BX6    X1*X5
         SA4    A5+1
         BX4    X2*X4
         BX6    X4+X6
         SA6    GENFSLA         NON-0 IF KING MOVING
         SA3    TPLOC+X3
         BX6    X1*X3           EXTRACT PAWNS
         SA4    A3+B1
         SA6    GENFPNA         SAVE PAWNS
         BX7    X2*X4
         SA7    A6+B1
         SA5    LINDX
         BX1    -X3*X1          REMOVE PAWNS
         BX2    -X4*X2
         ERRNZ  LE.MOV-2
         SB7    X5-1

*        BEGIN OUTER LOOP THROUGH REQUESTED SQUARES.

         ZR     X1,GENFSL6      IF NOTHING IN FIRST WORD
         PX1    X1
         SB4    B0
         NX6    X1,B2
         AX6    X0,B2
GENFSL1  SA0    12+BT2SQ
         SA5    CSTAT
GENFSL2  BX1    -X6*X1
         BX6    X6*X5
         SA3    B2+A0           SQUARE NUMBER
         LX6    X6,B2
         LX6    S.ACS-47-S.MFR
         BX7    X6+X3           ACS @ SQUARE FROM
         LX7    S.MFR
         LX3    X3,B1
         SA3    X3+ATKFR
         SA4    GENTO           VALID TO SQUARES
         BX3    X4*X3

*        BEGIN INNER LOOP THROUGH SQUARES ATTACKED BY EACH IN OUTER.

         SB6    B0
         ZR     X3,GENFSL4      IF NO ATTACKS IN FIRST TO WORD
         PX3    X3
         SB5    B0
```

```
          SB3    12+BT2SQ
          NX6    X3,B2
          AX6    X0,B2
 GENFSL3  SA5    CSTAT+B6
          BX3    -X6*X3
          LX5    X5,B5
          BX5    X5*X6            ACS BIT
          SA4    ALLOC+B6
          LX4    X4,B5
          BX4    X4*X6            CAPTURE BIT
          LX4    S.CAP-47
          LX5    S.ACS-47
          BX6    X4+X5
          SA4    B2+B3            TO SQUARE NUMBER
          LX6    X6,B2
          BX6    X6+X4
          BX6    X6+X7
          ERRNZ  LE.MOV-2
          SA6    B7+B1            STORE IN MOVES ARRAY
          SB7    A6+B1            ADVANCE POINTER
          NX6    X3,B2
          AX6    X0,B2
          NZ     X6,GENFSL3       IF NOT DONE WITH TO WORD
          NZ     B5,GENFSL5       IF DONE WITH BOTH WORDS
 GENFSL4  SA4    GENTO+1
          SA3    A3+B1
          BX3    X4*X3            VALID TO SQUARES
          ZR     X3,GENFSL5       IF NO ATTACKS IN SECOND TO WORD
          LX3    48
          SB5    48
          SB6    B1
          SB3    60+BT2SQ
          NX6    X3,B2
          AX6    X0,B2
          EQ     GENFSL3          PROCESS SECOND TO WORD

*         TERMINATE OUTER LOOP.

 GENFSL5  NX6    X1,B2
          AX6    X0,B2
          ZR     X6,GENFSL6       IF NOTHING LEFT IN FROM WORD
          ZR     B4,GENFSL1       IF STILL IN FIRST WORD
          EQ     GENFSL7          CONTINUE PROCESSING SECOND FROM WORD

 GENFSL6  ZR     X2,GENFSL8       IF NOTHING IN SECOND FROM WORD
          LX2    48
          PX1    X2
          BX2    X2-X2
          SB4    B1
          NX6    X1,B2
          AX6    X0,B2
 GENFSL7  SA0    60+BT2SQ
          SA5    CSTAT+1
          LX5    48
          EQ     GENFSL2

 GENFSL8  SA1    GENTO
          ERRNZ  LE.MOV-2
          SX6    B7+B1
          SA2    A1+B1
          SA6    LINDX
          RJ     =XGENFPN         GENERATE PAWN MOVES
          SA1    GENFSLA
          ZR     X1,GENFSL        IF NO KING MOVES
          RJ     =XGENCAS         GENERATE CASTLE MOVES
          EQ     GENFSL           RETURN

 GENFSLA  BSS    1                NON-ZERO IF KING MOVES
 GENTSL   SPACE  4,88
***       GENTSL - GENERATE ALL MOVES TO A SET OF SQUARES.
*
*         ENTRY  (X1,X2) = SQUARE LIST.
*                (LINDX) = LOCATION TO STORE FIRST MOVE.
*
*         EXIT   (LINDX) = LOCATION BEYOND LAST STORED MOVE.
*
*         USES   ALL REGISTERS.
*



          ENTRY  GENTSL
 GENTSL   EQ     *+400000B
          SA3    GENTO            VALID TO SQUARES
          SA4    A3+B1
          BX6    -X1*X3           REMOVE CURRENT SET
          BX7    -X2*X4
          SA6    A3               UPDATE VALID TO SQUARES
          SA7    A4
          BX1    X3*X1            REMAINING VALID TO SQUARES
          BX2    X4*X2
          BX6    X1
          BX7    X2
          MX0    1
          SA6    GENTSLA          SAVE FOR PAWN MOVE GENERATION
          LX0    48
          SA7    A6+B1
          SA3    LINDX
          SA4    MSIDE
          SA4    TPLOC+X4         PAWNS OF MOVING SIDE
          SA5    GENFR            VALID FROM SQUARES
          BX6    -X4*X5           REMOVE PAWNS
          BX7    X4*X5
          SA6    A7+B1
          SA4    A4+B1
          SA7    GENFPNA          REMAINING PAWNS
          SA5    A5+B1
          BX6    -X4*X5
          BX7    X4*X5
          SA6    A6+B1
          SA7    A7+B1
          SB7    X3
          SB6    LE.MOV

*         BEGIN OUTPUT LOOP THROUGH REQUESTED SQUARES.

          ZR     X1,GENTSL6       IF NOTHING IN FIRST WORD
          SB4    B0
          PX1    X1
          NX6    X1,B2
          AX6    X0,B2
          SA0    12+BT2SQ
 GENTSL1  SA4    CSTAT
          SA5    ALLOC
 GENTSL2  BX1    -X6*X1
          BX4    X6*X4
          BX5    X5*X6
          LX5    S.CAP-S.ACS
          BX5    X5+X4
          LX7    X5,B2
          LX7    S.ACS-47         ACS @ CAP
          SA4    B2+A0            TO SQUARE NUMBER
          LX5    X4,B1
          BX7    X7+X4
          SA3    ATKTO+X5         SQUARES ATTACKING THIS SQUARE
          SA4    GENTSLB          VALID FROM SQUARES
          BX3    X3*X4
          SB5    B0
          ZR     X3,GENTSL4       IF NO ATTACKS IN FIRST WORD
          SB3    12+BT2SQ
          PX3    X3
          SA4    CSTAT
          NX6    X3,B2
          AX6    X0,B2
 GENTSL3  SA5    B3+B2            FROM SQUARE NUMBER
          LX5    S.MFR
          BX3    -X6*X3
          BX6    X6*X4            ACS
          LX6    X6,B2
          LX6    S.ACS-47
          BX6    X6+X7
          BX6    X6+X5
          SA6    B7
          SB7    B7+B6            ADVANCE LINDX
          NX6    X3,B2
          AX6    X0,B2
          NZ     X6,GENTSL3       IF NOT DONE WITH FROM WORD
          NZ     B5,GENTSL5       IF DONE WITH BOTH FROM WORDS
 GENTSL4  SA3    A3+B1
          SA4    GENTSLB+1
          BX3    X3*X4
          ZR     X3,GENTSL5       IF NO ATTACKS IN SECOND WORD
          SB3    60+BT2SQ
          LX3    48
          SB5    B1
          SA4    CSTAT+1
          PX3    X3
          LX4    48
          NX6    X3,B2
          AX6    X0,B2
          EQ     GENTSL3          PROCESS SECOND FROM WORD

*         TERMINATE OUTPUT LOOP.

 GENTSL5  NX6    X1,B2
          AX6    X0,B2
          ZR     X6,GENTSL6       IF NOTHING LEFT IN WORD
          ZR     B4,GENTSL1       IF STILL IN FIRST WORD
          EQ     GENTSL7          CONTINUE PROCESSING SECOND WORD

 GENTSL6  ZR     X2,GENTSL8       IF DONE WITH SECOND WORD
          LX2    48
          PX1    X2
          SB4    B1
          BX2    X2-X2
          NX6    X1,B2
          AX6    X0,B2
          SA0    60+BT2SQ
 GENTSL7  SA4    CSTAT+1
          SA5    ALLOC+1
          LX4    48
          LX5    48
          EQ     GENTSL2          PROCESS SECOND WORD

 GENTSL8  SA1    GENTSLA          VALID TO SQUARES
          SX6    B7
          SA2    A1+B1
          SA6    LINDX
          RJ     =XGENFPN         GENERATE PAWN MOVES
          EQ     GENTSL           RETURN

 GENTSLA  BSS    2                TO SQUARES
 GENTSLB  BSS    2                VALID FROM SQUARES
 GENFPN   SPACE  4,88
***       GENFPN - GENERATE ALL PAWN MOVES FROM A SQUARE LIST.
*
*         ENTRY  (GENFPNA) = SQUARE LIST OF PAWNS.
*                (X1,X2) = VALID TO-SQUARES.
*                (LINDX) = LOCATION TO STORE FIRST MOVE.
*
*         EXIT   (LINDX) = LOCATION BEYOND LAST STORED MOVE.
*
*         USES   ALL REGISTERS.
*



          ENTRY  GENFPN,GENFPNA
 GENFPN   EQ     *+400000B
          SA3    MSIDE
          SB3    X3
          SB3    -B3
          SA3    ALLOC+B3         ALL OPPONENT PIECES
          SA4    A3+B1
          BX6    -X3*X1           VALID MOVE SQUARES
          BX7    -X4*X2
          SA6    GNFPNMB
          SA7    A6+B1
          BX6    X1*X3            VALID CAPTURE SQUARES
          BX7    X2*X4
          SA3    ENPAS
          SA4    A3+B1
          BX3    X3*X1
          BX4    X4*X2
          BX6    X6+X3
          BX7    X7+X4
          SA6    GNFPNCA
          SA7    A6+B1
          SA1    GENFPNA
          SA3    LINDX
          SA2    A1+B1
          BX6    X1+X2
          ZR     X6,GENFPN        IF NO PAWNS, RETURN
          SB7    X3
          MX0    10
          PL     B3,GENFPN2       IF BLACK TO MOVE

*         WHITE MOVES ONE SQUARE.

          SB3    60-10
          SB4    -8
          BX7    X7-X7
          RJ     =XGNFPNM         GENERATE PAWN MOVES

*         WHITE MOVES TWO SQUARES.
```

```
          SA1    GNFPNMA
          MX6    8
          LX6    2*10-1
          SA3    ALLOC
          BX1    -X3*X1            REMOVE ILLEGAL MOVES
          BX1    X6*X1
          ZR     X1,GENFPN1        IF NO MOVES TWO SQUARES
          SB4    B4+B4
          SB3    60-10
          SX7    M.ENP
          RJ     =XGNFPNM

*         WHITE  CAPTURES LEFT.

 GENFPN1  SA1    GENFPNA
          MX0    9
          SA2    A1+B1
          SB3    60-9
          SB4    -7
          RJ     =XGNFPNC          GENERATE PAWN CAPTURES
*         WHITE CAPTURES RIGHT.

          SA1    GENFPNA
          MX0    11
          SA2    A1+B1
          SB3    60-11
          SB4    -9
          RJ     =XGNFPNC
          SX6    B7+0
          SA6    LINDX
          EQ     GENFPN            RETURN

*         BLACK MOVES ONE SQUARE.

 GENFPN2  LX0    10
          BX7    X7-X7
          SB3    10
          SB4    8
          RJ     =XGNFPNM

*         BLACK MOVES TWO SQUARES.

          SA2    GNFPNMA+1
          MX6    8
          LX6    5*10-1
          SA3    ALLOC+1
          BX2    -X3*X2            REMOVE ILLEGAL MOVES
          BX2    X6*X2
          ZR     X2,GENFPN3        IF NO MOVES TWO SQUARES
          SB3    10
          SX7    M.ENP
          BX1    X1-X1
          SB4    B4+B4
          RJ     =XGNFPNM

*         BLACK CAPTURES LEFT.

 GENFPN3  SA1    GENFPNA
          SX0    3777B
          SB3    11
          SB4    9
          SA2    A1+B1
          RJ     =XGNFPNC

*         BLACK CAPTURES RIGHT.

          SA1    GENFPNA
          SB3    9
          SB4    7
          SX0    777B
          SA2    A1+B1
          RJ     =XGNFPNC
          SX6    B7+0
          SA6    LINDX
          EQ     GENFPN            RETURN

 GENFPNA  BSS    2                 PAWN LOCATIONS
 GNFPNM   SPACE  4,88
***       GNFPNM - GENERATE PAWN MOVES FROM.
*
*         ENTRY  (X0) = SHIFT MASK.
*                (X1,X2) = PAWN SQUARES.
*                (X7) = EN PASSANT BIT, IF 2 SQUARE PAWN MOVES.
*                (B3) = SHIFT COUNT.
*                (B4) = MOVE DISTANCE.
*                (B7) = (LINDX)
*                (GNFPNMB) = SQUARES TO MOVE TO.
*
*         EXIT   (GNFPNMA) = PAWN LOCATIONS AFTER MOVING.
*                (X0) = SHIFT MASK.
*                (X2) = 0.
*                (B4) = MOVE DISTANCE.
*                (B7) = LAST LOCATION USED + LE.MOV.
*


 GNFPNM   EQ     *+400000B
          LX1    X1,B3             SHIFT TO ADVANCE ONE RANK
          LX2    X2,B3
          BX3    X1*X0
          BX4    X2*X0
          BX1    -X0*X1
          BX2    -X0*X2
          BX6    X1+X4
          BX2    X2+X3
          SA6    GNFPNMA           SAVE RESULTING POSITION
          SA3    GNFPNMB
          SA4    A3+B1
          BX1    X3*X6
          BX6    X2
          SA6    A6+B1
          BX2    X4*X2
          SB5    LE.MOV
          SX5    B1
          LX5    47
          ZR     X1,GNFPNM3        IF NOTHING IN FIRST WORD
          PX1    X1
          SB3    0
          SB6    B0
          SA0    12+BT2SQ
          NX6    X1,B2
          AX6    X5,B2
 GNFPNM1  SA4    EIGHT+B3
          LX4    X4,B6
          BX4    X4*X6             EXTRACT PROMO BIT
          SA3    A0+B2             TO SQUARE NUMBER
          BX1    -X6*X1
          SX6    X3+B4             FROM SQUARE NUMBER
          LX6    S.MFR
          BX6    X6+X7             EN PASSANT BIT
          BX6    X6+X3
          ZR     X4,GNFPNM2        IF NOT PROMOTION
          SX4    M.PRO
          BX6    X6+X4             ROOK
          AX4    2
          SA6    B7+0
          IX6    X6+X4             KNIGHT
          SA6    A6+B5
          IX6    X6+X4             BISHOP
          SA6    A6+B5
          IX6    X6+X4             QUEEN
          SB7    A6+B5
 GNFPNM2  SA6    B7
          SB7    B7+B5
          NX6    X1,B2
          AX6    X5,B2
          NZ     X6,GNFPNM1        IF NOT DONE WITH WORD
 GNFPNM3  ZR     X2,GNFPNM         IF DONE WITH SECOND WORD, RETURN
          LX2    48
          PX1    X2
          BX2    X2-X2
          SB3    B1
          SB6    48
          SA0    60+BT2SQ
          NX6    X1,B2
          AX6    X5,B2
          EQ     GNFPNM1           PROCESS SECOND WORD

 GNFPNMA  BSS    2                 RESULTING PAWN POSITIONS
 GNFPNMB  BSS    2                 VALID MOVE TO SQUARES
 GNFPNC   SPACE  4,88
***       GNFPNC - GENERATE PAWN CAPTURES FROM.
*
*         ENTRY  (X0) = SHIFT MASK.
*                (X1,X2) = PAWN SQUARES.
*                (B3) = SHIFT COUNT.
*                (B4) = MOVE DISTANCE.
*                (B7) = (LINDX)
*                (GNFPNCA) = VALID CAPTURE SQUARES.
*


 GNFPNC   EQ     *+400000B
          LX1    X1,B3             SHIFT TO ADVANCE ONE RANK
          LX2    X2,B3
          BX3    X1*X0
          BX4    X2*X0
          BX1    -X0*X1
          BX2    -X0*X2
          BX1    X1+X4
          BX2    X2+X3
          SA3    GNFPNCA
          SA4    A3+B1
          BX1    X1*X3             VALID CAPTURES
          BX2    X2*X4
          SX0    B1
          SX7    1
          LX0    47
          LX7    S.CAP
          SB5    LE.MOV
          ZR     X1,GNFPNC3        IF NO CAPTURES IN FIRST WORD
          SA0    12+BT2SQ
          SB3    B0
          SB6    B0
          NX6    X1,B2
          AX6    X0,B2
 GNFPNC1  SA3    CSTAT+B3
          SA4    ENPAS+B3
          SA5    A0+B2             SQUARE NUMBER
          LX3    X3,B6
          BX3    X3*X6             GET ACS
          LX3    B2,X3
          LX3    S.ACS-47
          LX4    X4,B6
          BX4    X4*X6
          LX4    B2,X4
          LX4    S.ENP-47
          BX4    X4+X3
          BX4    X4+X7             CAPTURE BIT
          SX3    X5+B4             FROM SQUARE
          LX3    S.MFR
          BX3    X3+X5
          SA5    EIGHT+B3
          BX1    -X6*X1
          LX5    X5,B6
          BX5    X6*X5
          BX6    X3+X4
          ZR     X5,GNFPNC2        IF NOT PROMOTION
          SX5    M.PRO
          BX6    X6+X5
          SA6    B7
          AX5    2
          IX6    X6+X5
          SA6    A6+B5
          IX6    X6+X5
          SA6    A6+B5
          IX6    X6+X5
          SB7    A6+B5
 GNFPNC2  SA6    B7
          SB7    B7+B5
          NX6    X1,B2
          AX6    X0,B2
          NZ     X6,GNFPNC1        IF MORE IN SAME WORD
 GNFPNC3  ZR     X2,GNFPNC         IF DONE WITH BOTH WORDS
          LX2    48
          SB3    B1
          SB6    48
          PX1    X2
          BX2    X2-X2
          NX6    X1,B2
          AX6    X0,B2
          SA0    60+BT2SQ
          EQ     GNFPNC1           PROCESS SECOND WORD


 GNFPNCA  BSS    2                 VALID CAPTURE SQUARES
 GENCAS   SPACE  4,88
***       GENCAS - GENERATE CASTLE MOVES.
*
*         ENTRY  (LINDX) = POINTER TO LWA+1 OF MOVES ARRAY.
*
```

```
         ENTRY  GENCAS
GENCAS   EQ     *+400000B
         SA1    MSIDE
         PL     X1,GENCAS1        IF WHITE TO MOVE
         SA1    KBLOC+1
         SA2    CSTAT+1
         BX3    X1*X2
         LX2    -20
         ZR     X3,GENCAS         IF CASTLING ILLEGAL
         SA4    ALLOC+1
         SA5    AWATK+1
         LX4    -20
         LX5    -20
         SX7    7476B+M.CAS+M.ACS
         EQ     GENCAS2

GENCAS1  SA1    KWLOC
         SA2    CSTAT
         BX3    X1*X2
         LX2    -30
         ZR     X3,GENCAS         IF CASTLING ILLEGAL
         SA4    ALLOC
         SA5    ABATK
         LX4    -30
         LX5    -30
         SX7    0406B+M.CAS+M.ACS
GENCAS2  SX1    34B
         SX3    14B
         LX2    -2
         PL     X2,GENCAS3        IF SHORT CASTLE ILLEGAL
         BX3    X3*X4
         BX1    X1*X5
         BX1    X1+X3
         NZ     X1,GENCAS3        IF ATTACKED OR OCCUPIED
         SA1    LINDX
         SA7    X1
         SX6    X1+LE.MOV
         SA6    A1
GENCAS3  SX1    360B
         SX3    340B
         LX2    -7
         PL     X2,GENCAS4        IF LONG CASTLE ILLEGAL
         BX3    X3*X4
         BX1    X1*X5
         BX1    X1+X3
         NZ     X1,GENCAS4        IF ATTACKED OR OCCUPIED
         SX6    -4B+M.ENP
         SA1    LINDX
         IX7    X6+X7
         SA7    X1
         SX6    X1+LE.MOV
         SA6    A1
         EQ     GENCAS

GENCAS4  EQU    GENCAS
REPCHK   SPACE  4,88
***      REPCHK - REPEAT CHECK.
*
*        ENTRY  (REPPN) = ADDRESS OF CURRENT PACKED BOARD POSITION.
*               (VMOVE) = MOVE THAT LED TO CURRENT POSITION.
*               (REPST) = ADDRESS OF EARLIEST POSITION TO COMPARE.
*
*        EXIT   (X6) = -1 + (NO. OF TIMES POSITION HAS RECURRED).
*               ONLY RECENT RECURRENCES (THOSE THAT FIT WITHIN THE
*               TABLE) ARE DETECTED.
*
*        USES   ALL REGISTERS EXCEPT A0.
*



         ENTRY  REPCHK
REPCHK   EQ     *+400000B
         SA1    VMOVE
         MX6    54
         BX2    -X6*X1            TO SQUARE
         SA2    SQ2PB+X2          TRANSLATE TO PACKED BOARD WORD NUMBER
         SA3    REPPN             CURRENT PACKED BOARD ADDRESS
         ERRNZ  REPST-REPPN-1
         SA4    A3+B1             EARLIEST BOARD TO CHECK
         LX2    30
         SB3    X3
         SB5    X4
         SB4    B3                FIRST BOARD TO COMPARE + LE.PAC
         SA1    X2+B3             LIKELY MISMATCH WORD
         SX6    -1                INITIALIZE REPEAT COUNT
REPCHK1  SB4    B4-2*LE.PAC       ADVANCE TO PREVIOUS ENTRY
         LT     B4,B5,REPCHK      RETURN IF NO MORE ENTRIES
         SA3    B4+X2             LIKELY MISMATCH
         IX5    X1-X3
         NZ     X5,REPCHK1        IF MISMATCH
         SB7    LE.PAC-1
REPCHK2  SA4    B3+B7
         SA5    B4+B7
         IX5    X4-X5
         SB7    B7-B1
         NZ     X5,REPCHK1        IF MISMATCH
         PL     B7,REPCHK2        IF NOT DONE WITH COMPARE
         SX6    X6+B1             A MATCH, SO INCREMENT REPEAT COUNT
         EQ     REPCHK1           COMPARE TO ANOTHER ENTRY
CHKCHK   SPACE  4,88
***      CHKCHK - CHECK FOR CHECKING MOVE.
*
*        ENTRY  (INDEX) = ADDRESS OF MOVE TO CHECK.
*
*        EXIT   (X6) < 0, IF MOVE IS CHECK.
*                         DOES NOT WORK ON CASTLES OR EN PASSANT.
*
*        USES   ALL REGISTERS EXCEPT A0 AND X0.
*



         ENTRY  CHKCHK
CHKCHK   EQ     *+400000B
         SA1    INDEX
         MX6    -6
         SA1    X1                THE MOVE
         BX3    -X6*X1            8X8 TO SQUARE
         SA4    OSIDE
         AX1    6
         BX2    -X6*X1            8X8 FROM SQUARE
         BX4    -X4
         SA1    WKSQR+1+X4        8X8 ENEMY KING SQUARE
         SA4    EXPND+X1
         SA5    EXPND+X2
         SB2    X4                10X12 KING SQUARE
         SB3    X5                10X12 FROM SQUARE
         SA4    EXPND+X3
         SB4    X4                10X12 TO SQUARE
         SB6    B2-B4             10X12 DISTANCE (TO SQR, KING SQR)
         SA1    DIRTB+B6          10X12 DIRECTION (TO SQR, KING SQR)
         ZR     X1,CHKCHK2        IF KING SQR NOT IN LINE WITH TO SQR
         UX1,B7 X1
         SA1    NBOARD+X2         MOVING PIECE
         SA1    CNCHK+X1
         SB5    B7+29
         LX7    X1,B5
         SB5    BT2SQ
         PL     X7,CHKCHK2        IF PIECE CANNOT CHECK THIS WAY
         NG     X1,CHKCHK1        IF SWEEP PIECE
         EQ     B6,B7,CHKCHK      IF DIRECTION = DISTANCE, IT IS CHECK
         EQ     CHKCHK2           TRY FOR DISCOVERED CHECK

CHKCHK1  SB4    B4+B7             MOVE TOWARD KING SQUARE
         EQ     B4,B2,CHKCHK      IF RAY IS CLEAR
         SA1    B5+B4             8X8 SQUARE NUMBER
         SA1    NBOARD+X1         SQUARE
         ZR     X1,CHKCHK1        IF SQUARE IS CLEAR
*        EQ     CHKCHK2

CHKCHK2  SB6    B2-B3             10X12 DISTANCE (FROM SQ, KING SQ)
         SA1    DIRTB+B6          10X12 DIRECTION
         SB7    X1
         BX6    X6-X6
         ZR     B7,CHKCHK         IF FR SQ NOT IN LINE WITH K SQ
         SB4    X4                10X12 TO SQUARE
         SB5    BT2SQ
CHKCHK3  SB2    B2-B7             MOVE OUT RAY FROM K SQUARE
         EQ     B2,B3,CHKCHK3     IF FROM SQUARE, SKIP IT
         SA1    B5+B2             8X8 SCAN SQUARE
         NG     X1,CHKCHK         IF OFF BOARD, RETURN
         EQ     B2,B4,CHKCHK      IF TO SQUARE, RETURN
         SA1    NBOARD+X1
         ZR     X1,CHKCHK3        IF EMPTY SQUARE, LOOP
         SA2    SSIDE
         BX2    X2-X1
         NG     X2,CHKCHK         IF NOT FRIENDLY PIECE
         SA1    CNCHK+X1
         PL     X1,CHKCHK         IF NOT A SWEEP PIECE
         SB5    B7+29
         LX6    X1,B5             RETURN CHECK IF CAN CHECK THIS WAY
         EQ     CHKCHK
BLDDNI   SPACE  4,88
***      BLDDNI - BUILD DENY LISTS.
*
*
*        USES   ALL REGISTERS, STACK1.
*
*               A SQUARE IS DENIED IF IT IS:
*                - ATTACKED BY A SMALLER ENEMY PIECE, AND
*                - NOT OCCUPIED BY AN EQUAL OR LARGER ENEMY PIECE.
*



BLDDNI   IF     DEF,BLDDNI
         ENTRY  BLDDNI
BLDDNI   EQ     *+400000B

         ECHO   ,COLOR=(WHITE,BLACK),F=(W,B),E=(B,W),S=(+,-),Y=(-,),Z=(
,,-),M=(B,N),L=(K,Q)

         QUAL   COLOR

         SA1    P!E!LOC             ENEMY PAWNS
         MX0    S!10
         SA2    A1+B1
         BX3    Z!X0*X1
         BX4    Z!X0*X2
         BX1    Y!X0*X1
         BX2    Y!X0*X2
         BX1    X4+X1
         BX2    X3+X2
         LX1    S!9
         LX2    S!9
         SA3    ONBRD
         BX6    X1*X3
         SA4    A3+B1
         BX7    X2*X4
         LX1    S!2
         LX2    S!2
         BX3    X3*X1
         BX4    X4*X2
         BX6    X3+X6             ENEMY PAWN ATTACKS
         BX7    X4+X7
         SX0    B1
         SA6    F!MDNI             DENY LIST FOR FRIENDLY MINOR PIECES
         SA7    A6+B1
         LX0    47
         SA0    12+BT2SQ
         SB4    60+BT2SQ
         SA1    M!E!LOC             ENEMY MINOR PIECES
         SA2    A1+B1
         SA3    A2+B1
         SA4    A3+B1
         BX1    X1+X3             ENEMY MINOR PIECES
         BX2    X2+X4
         ASMATK                   ASSEMBLE ATTACKS
         SA6    A7+B1
         SA7    A6+B1
         SA1    R!E!LOC             ENEMY ROOKS
         SA2    A1+B1
         ASMATK                   ASSEMBLE ATTACKS
         SA1    L!E!LOC             ENEMY KINGS AND QUEENS
         SA2    A1+B1
         SA3    A2+B1
         SA4    A3+B1
         BX1    X1+X3
         BX2    X2+X4
         BX6    -X1*X6
         BX7    -X2*X7
         SA6    A7+B1
         SA7    A6+B1
         SA5    A6-B1
         SA4    A5-B1
         SA3    R!E!LOC             ENEMY ROOKS
         BX1    X1+X3
         BX6    -X1*X4
         SA3    A3+B1
         BX2    X2+X3
         BX7    -X2*X5
         SA6    A4
         SA7    A5
         SA5    A6-B1
         SA4    A5-B1
         SA3    M!E!LOC             ENEMY MINOR PIECES
```

```
          BX1    X1+X3
          SA3    A3+B1
          BX2    X2+X3
          SA3    A3+B1
          BX1    X1+X3
          SA3    A3+B1
          BX2    X2+X3
          BX6    -X1*X4
          BX7    -X2*X5
          SA6    A4
          SA7    A5

          QUAL   *
          ENDD
          EQ     BLDDNI            RETURN
 BLDDNI   ENDIF
*CALL     COMCSTT
 MOVSTT   EQU    STT
          ENTRY  MOVSTT
          END
EVALU8
*FILE LINP
          IDENT  EVALU8
          EXT    MINI0,WOOD0,FULL0,PREL0,LEGL0,MATE0,QUES0,REPD
          EXT    POND0
          INIT                     INITIALIZE MACRO SYSTEM

***       EVALU8 - CHESS 4.0 RECURSIVE EVALUATOR.
*
*         ENTRY  SQRLST BLOCK CONTAINS DATA SPECIFICALLY RELEVANT TO
*                CURRENT NODE.  VMOVE WORD IN NODE BANK CONTAINS MOVE
*                LEADING TO CURRENT NODE.
*
*         EXIT   SCORE WORD IN NODE BANK CONTAINS
*                THE SCORE FOR THIS NODE, POSITIVE IF WHITE IS WINNING,
*                NEGATIVE IF BLACK IS, ROUGHLY IN UNITS OF MATERIAL,
*                WHERE A PAWN IS WORTH 64 UNITS.
*                AT PLY 0, BSTMV CONTAINS THE BEST MOVE.
*
*         USES   ALL REGISTERS.

*CALL VERSION
 EVALU8   TITLE  EVALU8 - CHESS 'V' RECURSIVE EVALUATOR.
*CALL IOCOM
*CALL     LET
*CALL     TLET
*CALL SQRLST
*CALL BOARD
*CALL CONST
*CALL STACKS
*CALL DEBUG
 EEVALU8  ECERTB EEVMINI

 STACK2   EQU    STACK1+1
 MOVES1   EQU    MOVES+1
 MOVES3   EQU    MOVES+3
 CONST    SPACE  4,10
**        CONSTANT DATA.

 CHECKS   DATA   1000000000100000000B  MASK FOR CHECK STATUS
 MCOMP    CON    M.CAP+M.PRO+M.ENP+M.CAS+M.MFR+M.MTO
 DELAB    DATA   5L ITER
 DPLAB    DATA   5L PRES
 EVALU8   SPACE  4,10
          ENTRY  EVALU8.
 EVALU8.  BSS    0
          SA0    B0                INIT UNUSED STACK POINTER

**        INCREMENT NODE COUNT FOR THIS PLY.

          SA1    NPLYC             PLY NUMBER
          SA2    NDCNT+X1          INCREMENT NODE COUNT FOR THIS PLY
          SX3    B1
          IX6    X2+X3
          SA6    A2
          SA2    NDTTL             UPDATE TOTAL NODE COUNT
          IX6    X2+X3
          SA6    A2

**        CLEAR LIMB STEMMING FROM THIS NODE.

          BX6    X6-X6             ZERO LIMB LENGTH
          SA6    LIMBL+X1
          SA6    NSRCH+X1          ALSO CLEAR SEARCHED MOVES COUNT
          MX6    1                 INIT TRANSPOSITIONS INFO WORD
          LX6    18                TO NO SCORE, NO MOVE, NEGATIVE
          SA6    TRAWD             CLOBBERING PRIORITY.

**        BRANCH ON SEARCH MODE TO PROPER MODULE.

   GOTO (BASE0,MINI0,WOOD0,FULL0,PREL0,LEGL0,MATE0,QUES0,POND0),!!!!!!!!
,(CAND,(MASK,60-L.MOD+1),(LSHIFT,VMOVE,-S.MOD))
 BASE     TITLE  BASE - PROCESS BASE NODE (PLY ZERO).
**        BASE - PROCESS BASE NODE (PLY ZERO).

*         NODE PRIVATE TEMPORARY EQUIVALENCES.

 SCORL    EQU    NDTMP+2           SAVED LAST BEST SCORE
 TIMUS    EQU    NDTMP+4           TIME (MS) USED SO FAR THIS MOVE
 STIMUS   EQU    NDTMP+6           TIME USED THRU PART OF ITERATION
 TMREM    EQU    NDTMP+7           TIMPM * TILTC = TOTAL REMAINING TIME
 LBSTMV   EQU    NDTMP+8           SAVE PREVIOUS COMPUTED BEST MOVE
 LLIMBL   EQU    NDTMP+9           SAVE PREVIOUS LIMBL IN CASE NEEDED

 BASE0    BSS    0

*         INITIALIZE NODE COUNTS, ETC.

          SA1    SSIDE
          BX7    X1
          BX6    X6-X6             CLEAR NODE COUNTS
          SA7    CSIDE             SET COMPUTER SIDE
          SA6    POSCNT            CLEAR PREL POSITIONAL SCORE COUNT
          SA6    TIMLI             CLEAR START TIME OF THIS ITERATION
          SA6    NPLYD             CLEAR PLY DEPTH PAST BASE LEVEL
          SA6    NDSCR             TOTAL POSITIONAL SCORES
          SA6    A6+B1             REGULAR POSITIONAL SCORES
          ERRNZ  NDREG-NDSCR-1
          SA6    A6+B1             CHECKMATE POSITIONAL SCORES
          ERRNZ  NDMOP-NDREG-1
          SA6    A6+B1             AVERAGE POSITIONAL SCORES
          ERRNZ  POSINI-NDMOP-1
          SA6    A6+B1             MIN POSITIONAL SCORE
          ERRNZ  POSMIN-POSINI-1
          SA6    A6+B1             MAX POSITIONAL SCORE
          ERRNZ  POSMAX-POSMIN-1
          SB2    NDCNT
          SB3    NDCNT+NPLY+NPLY/2+2
          ERRNZ  NDCNT+NPLY-NDDIT
 BAS0A    SA6    B2
          SB2    B2+B1
          LT     B2,B3,BAS0A
          SA1    NPLYC             0 IF NORMAL, 1 IF PONDERING
          SX7    B1
          SA7    NDCNT             SET PLY 0 COUNT TO 1
          SA7    NDCNT+X1          SET BASE PLY COUNT TO 1
          IX7    X7+X1             1 IF NORMAL, 2 IF PONDERING
          SA7    NDTTL             SET TOTAL NODE COUNT

          SETX   PLIND,NPLY-1
 BASE1    SETA   (IDXLOC,KILLS,PLIND),0
          SETX   PLIND,(MINUS,PLIND,1)
          COND   (PL,PLIND),BASE1  LOOP FOR ALL PLIES
          RELEAS PLIND
          COND   (NZ,PONDR),BAS1A  IF PREDICTED OPPONENT MOVE
          SETQ   PKILL,0           ELSE CLEAR OUR REJOINDER
 BAS1A    BSS    0

*         SET UP DRAW SCORE FROM USCF RATING PARAMETERS.

          SETX   ODRAW,DRAWS       OLD DRAW SCORE
          SETQ   DRAWS,(XOR,(DIVIDE,(LSHIFT,(MINUS,RATEYR,RATEMY),6),RAT
,EPN),SSIDE)
          COND   (PL,(XOR,SSIDE,(MINUS,ODRAW,DRAWS))),BAS1B
          SETX   ODRAW,DRAWS       UP TO AT LEAST RATINGS DRAW
 BAS1B    SETX   GDRAW,(PLUS,(LODREL,ODRAW),(XOR,SSIDE,FDRAWG))

**        MAKE DRAW MORE DESIRABLE AS GAME CONTINUES.
*
*         NEW DRAW SCORE F(G,I,S,D,O) =
*         WORSE(D+BETTER(0,S-D)*MIN(MIN(G,P1)+P2*I,P3)/(P3+P4),O+BETTER(
*         P5,-P5))
*         WHERE  D = RATINGS DRAW SCORE.
*                S = SCORE OF LAST MOVE.
*                I = NUMBER OF MOVES SINCE LAST IRREVERSIBLE.
*                G = MOVE NUMBER IN THE GAME (MOVNUM).
*                O = DRAW SCORE FROM LAST MOVE
*         AND    P1, P2, P3, P4, AND P5 ARE THE LETS:
*                FDRAWM, FDRAWI, FDRAWA, FDRAWT, AND FDRAWG.

          SETX   SMARG,(MINUS,SAVES,DRAWS)
          COND   (NG,(XOR,SSIDE,SMARG)),BAS1C
          SETX   MWEAR,(PLUS,(TIMES,FDRAWI,(AND,CNT50,777B)),(MINF,MOVNU
,M,FDRAWM))
          SETX   MWEAR,(MINF,MWEAR,FDRAWA)
          SETX   DRAWI,(DIVIDE,(TIMES,(LODREL,SMARG),(LODREL,MWEAR)),(PL
,US,FDRAWA,FDRAWT))
          SETQ   DRAWS,(PLUS,DRAWS,(LODREL,DRAWI))
          COND   (PL,(XOR,SSIDE,(MINUS,GDRAW,DRAWS))),BAS1C
          SETQ   DRAWS,(LODREL,GDRAW)
 BAS1C    BSS    0

*         SET UP TIME CONTROL PARAMETERS.

          SETQ   TMREM,(TIMES,TIMPM,TILTC)
          SETQ   LBSTMV,0
          SETQ   LLIMBL,0
          SETQ   TIMUS,0
          COND   (TRUE,(TEST,PONDR,S.THI)),BASE3
          SETX   TRATDF,(MINUS,(DIVIDE,(LSHIFT,TIMPM,6),(MAXF,ATMPM,1)),
,100B)
          COND   (LE,ITERS,2),BAS1D  IF LAST MOVE QUICK
          COND   (GT,TRATDF,TRATOL),BASE2   IF TIME SURPLUS
 BAS1D    BSS    0
          COND   (GE,TRATDF,(NOT,TRATOL)),BASE3   IF NO DEFICIT
          RELEAS TRATDF
          SETQ   ATMPM,TIMPM       RE-ADJUST OUR EXPECTATIONS
          SETQ   NTRLO,(MAXF,(MINUS,NTRLO,TRATIC),TRATMN)
          GOTO   BASE3

 BASE2    SETQ   NTRLO,(MINF,(PLUS,NTRLO,TRATIC),TRATMX)
          SETQ   ATMPM,TIMPM       RE-ADJUST OUR EXPECTATIONS
 BASE3    SETQ   TIMIN,(LSHIFT,(TIMES,NTRLO,TIMPM),-6)
          SETQ   TIMAX,(LSHIFT,(TIMES,TIMPM,TMAXOM),-6)

          SETQ   ITERS,1
          SETQ   BSTMV,0
          SETQ   SCORL,SAVES       ALSO INIT LAST MINIMAX BEST
          SETQ   SCORI,SAVES       INIT ITERATION SCORE TO LAST MOVE
          GENMOV ALL MOVES
          SETAB  -INFIN,INFIN
          COND   (TRUE,(TEST,PONDR,S.THI)),BAS3A
          SETQ   MVNTR,(PLUS,MVNTR,1) ASSUME THIS MOVE NON-TRIVIAL

*         CHECK FOR DRAW BY 50 MOVE RULE.

 BAS3A    COND   (LT,(AND,CNT50,777B),(PLUS,MVRL50,1)),BAS3B
          SX6    B1
          GOTO   REPD              PREPARE TO CLAIM A DRAW

 BAS3B    BSS    0

*         CHECK FOR DRAW BY 2-TIME REPETITION.

          CALLE  REPCHK
          ZR     X6,BASE4          IF ONLY SINGLE REPETITION
          PL     X6,REPD           IF 2-TIME REPETITION

*         CHECK POSITIONS LIBRARY FOR ROTE-LEARNED MOVE.

 BASE4    COND   (FALSE,(TEST,SWICH,S.EXP)),BASE8
          SA0    STACK1+8          SET UP SCRATCH AREA FOR MSPACK
          RJ     =XEVNPCK          PACK NBOARD FOR SEARCHING
          SX7    0
          SX6    STACK1            ADDRESS OF SEARCH RESULT
          RJ     =XMSSRCH          SEARCH LIBRARY
          ZR     X6,BASE8          IF NO MATCH FOUND

*         SEARCH MOVES ARRAY FOR MATCH WITH LIBRARY MOVE.

          SETX   LMOVE,(LSHIFT,STACK2,18)
          SELOOP SELECT=BASE6
          COND   (NZ,(AND,(XOR,LMOVE,SMOVE),MCOMP)),BASE5
          SELMOV
 BASE5    SELEND
          RELEAS LMOVE
          GOTO   BASE8

*         CONFIRM LEGALITY OF LIBRARY MOVE.

 BASE6    SEARCH MODE=LEGL,ILLEGAL=BASE8
          SETQ   BSTMV,(INDEX,B0,INDEX)

*         *TRIVIAL* MOVE.
```

```
BASE7    COND   (TRUE,(TEST,PONDR,S.THI)),BAS40
         SETQ   MVNTR,(MINUS,MVNTR,1)  DISCOUNT TRIVIAL MOVE
         GOTO   BAS40             UN-JIGGLE AND RETURN

*        SORT MOVES ON PRELIMINARY SCORES.

BASE8    SETQ   INDEX,(LOCF,MOVES-LE.MOV)
         SETQ   POSINI,0          INIT BASIC POSITIONAL SCORE
         SETA   (IDXLOC,NSRCH,NPLYC),0  CLEAR SEARCHED MOVE NUMBER

*        JIGGLE LETS FOR VARIATIONS IN PLAY.

         SA1    JIGSIZ            MAX AMOUNT TO JIGGLE
         RJ     =XJIGLET
BASE9    SELOOP SELECT=BAS12,START=(PLUS,INDEX,LE.MOV)
         SELMOV
         SELEND
         COND   (EQ,(INDEX,NSRCH,NPLYC),0),BAS32  IF NO MOVES
         SETQ   POSINI,(DIVIDE,POSINI,(MAXF,POSCNT,1))
         SETQ   POSMIN,(MINUS,POSINI,WODREL)
         SETQ   POSMAX,(PLUS,POSINI,WODREL)
         COND   (NG,SSIDE),BAS10  IF BLACK TO MOVE
         SELOOP                   ELSE COMPLEMENT SCORES FOR SORT
         SETA   (SELST,1),(NOT,(INDEX,B0,(SELST,1)))
         SELEND
BAS10    BSS    0
         SX6    MOVES
         SB4    STACK1
         SA1    LINDX
         BX7    X1
         RJ     =XMOVSTT          SORT MOVES ON ASCENDING SCORES
         SETAB  (MAXF,-INFIN,(MINUS,SAVES,FLXTOL)),(MINF,INFIN,(PLUS,SA
,VES,FLXTOL))
         SETQ   OBVIO,0           CLEAR OBVIOUS MOVE FLAG
         COND   (FALSE,(TEST,SWICH,S.DBG)),BA10A
         SA1    DPLAB
         CALLE  DEBUGR,PRES       BKP WITH PREL SCORES INTACT
BA10A    BSS    0
         COND   (EQ,(INDEX,NSRCH,NPLYC),1),BAS11  IF ONLY ONE MOVE
         COND   (LT,(MINUS,MOVES3,MOVES1),FLXTOL),BAS11
         SETQ   OBVIO,1           IF LARGE PREL SCORE DIFF
BAS11    BSS    0
         SELOOP                   CLEAR OUT PRELIMINARY SCORES
         SETA   (SELST,1),0
         SELEND
         COND   (LE,MVNTR,JIGMVS),BAS13  IF OK TO JIGGLE FINAL SCORES
         BX1    X1-X1             ELSE RESET PARAMS TO INITIAL VALUES
         RJ     =XJIGLET
         GOTO   BAS13

BAS12    SEARCH MODE=PREL,NON=PERM,ILLEGAL=BASE9
         COND   (NZ,(AND,(XOR,PKILL,(INDEX,B0,INDEX)),MCOMP)),BASE9

*        SEARCH REJOINDER TO PREDICTED MOVE FIRST.

         SETA   (IDXLOC,B1,INDEX),(XOR,SSIDE,INFIN)
         GOTO   BASE9

*        START LOOP ON ITERS, SEARCHING ALL MOVES FIRST 1-PLY, THEN 2
*        PLY, ETC., UNTIL FDEPTH, USING THE SCORE AND LIMB FROM EACH
*        SEARCH TO GUIDE THE NEXT DEEPER ONE BY SETTING ALPHA-BETAR
*        ASPIRATION WINDOW AND FIRST MOVE CHOICES AT LOW PLY LEVELS.

BAS13    COND   (FALSE,(TEST,SWICH,S.DBG)),BAS14
         SA1    DELAB
         CALLE  DEBUGR,ITER
BAS14    SETA   (IDXLOC,NSRCH,NPLYC),0
         SETQ   MAINV,1           SET MAIN VARIATION FLAG

*        SEARCH MOVES IN ORDER OF PRELIMINARY SCORES.

BAS15    SETQ   INDEX,(LOCF,MOVES-LE.MOV)
BAS16    COND   (TRUE,(TEST,PONDR,S.THQ)),BAS24  IF MUST QUIT
         COND   (ZR,(INDEX,NSRCH,NPLYC)),BAS19
         SETQ   MAINV,0           CLEAR FLAG IF PAST MAIN VARIATION
         COND   (FALSE,(TEST,SWICH,S.TIM)),BAS19   IF TIME CONTROL OFF
         COND   (TRUE,(TEST,PONDR,S.THG)),BA16A
         COND   (TRUE,(TEST,PONDR,S.THI)),BAS19
BA16A    BSS    0
         COND   (LT,ITERS,MDEPTH),BAS19   GO AT LEAST MIN DEPTH
         COND   (ZR,(PLUS,BSTMV,LBSTMV)),BAS19  IF NO MOVE YET
         COND   (GT,ITERS,MDEPTH),BAS17   IF PAST CONFIRM DEPTH
         COND   (ZR,BSTMV),BAS19   ELSE MUST GET A MOVE
         COND   (TRUE,(TEST,WIERD,S.WRTC)),BAS40  DONE IF ONLY CONFIRM
BAS17    CALLE  TIMUSD            TIME USED SO FAR
         SA6    STIMUS
         COND   (ZR,OBVIO),BAS18  IF MOVE IS NOT OBVIOUS
         COND   (ZR,BSTMV),BAS18   IF NO MOVE YET
         COND   (LT,STIMUS,TIMIN),BAS18   IF HAVENT USED MUCH TIME
         COND   (GT,(XOR,(MINUS,SCORI,SCORL),SSIDE),FEZTOL),BAS18
         SETBIT WIERD,S.WREZ      ELSE MOVE IS QUOTE *EASY*
         GOTO   BASE7             SPREAD TIME BONUS

BAS18    BSS    0
         SETX   CTMREM,(MINUS,TMREM,STIMUS)
         COND   (NG,(PLUS,CTMREM,TGRACE)),BAS24   IF PAST LIMIT+GRACE
         SETX   TILTM1,(MINUS,TILTC,1)  MOVES TO GO AFTER THIS ONE
         COND   (ZR,TILTM1),BAS19   IF THIS MOVE MAKES TIME CONTROL
         COND   (GE,STIMUS,TIMAX),BAS24   IF USING TOO MUCH
         SETX   CTIMPM,(DIVIDE,(LODREL,CTMREM),(LODREL,TILTM1))
         COND   (LT,(DIVIDE,(LSHIFT,(LODREL,CTIMPM),6),(MAXF,TIMPM,1)),
,TPMRAT),BAS24   IF USING TOO MUCH

BAS19    SELOOP SELECT=BAS21,START=(PLUS,INDEX,LE.MOV)

*        NO NEED TO TEST *FULL* BIT HERE, SINCE THIS IS FIRST SELOOP.

         SELMOV
BAS20    SELEND
         GOTO   BAS25

BAS21    SEARCH MODE=FULL,IGNORE=BAS16
         SETQ   BSTMV,(INDEX,B0,INDEX)
         SETQ   SCORL,(INDEX,ALPHA,OSIDE)  SAVE SCORE
         COND   (EQ,INDEX,(LOCF,MOVES)),BAS22   IF FIRST MOVE
         SETX   MVSWD,(INDEX,B1,INDEX)
         SETA   (IDXLOC,B0,INDEX),(MASK,1) ELSE POP NEW BEST TO TOP
         SETQ   OBVIO,0           ALSO, MOVE IS NOT OBVIOUS
         SELOOP DIRECT=REVERSE,START=(MINUS,INDEX,LE.MOV)
         SETA   (SELST,2),(INDEX,B0,(SELST,0))
         SETA   (SELST,3),(INDEX,B0,(SELST,1))
         SETA   (SELST,0),(MASK,1)
         SELEND
         SETQ   MOVES,BSTMV
         SETQ   MOVES+1,(LODREL,MVSWD)
BAS22    COND   (GT,BETAR,ALPHA),BAS16
         SETA   (IDXLOC,ALPHA,(MINUS,1,OSIDE)),(PLUS,(INDEX,ALPHA,OSIDE
,),(XOR,SSIDE,1))
         GOTO   BAS16

*        RUNNING OVER TIME CONTROL, USE BEST SO FAR.

BAS24    SETBIT WIERD,S.WRTC      SET WIERD CONDITION FLAG
         COND   (NZ,BSTMV),BAS40  DONE IF HAVE A BSTMV
         SETQ   BSTMV,LBSTMV  ELSE USE PREVIOUS BEST
         MINMAX NEWVAL=SCORL      RESTORE SCORE OF LAST BEST
         SETA   (IDXLOC,LIMBL,NPLYC),LLIMBL  FROM EARLIER ITERATION
         GOTO   BAS40             DONE

*        ALL MOVES SEARCHED, SO CHECK FOR CHECKMATE OR STALEMATE.
*        ALSE RE-ASPIRE IF NECESSARY.

BAS25    COND   (ZR,BSTMV),BAS32
         SETQ   LBSTMV,BSTMV      SAVE IN CASE NEEDED
         SETQ   LLIMBL,(INDEX,LIMBL,NPLYC)  ALSO SAVE LIMB LENGTH

*        SEARCH COMPLETE.  DETERMINE WHETHER TO EXTEND DEPTH BY 1 AND
*        RE-SEARCH.  ALSO MAINTAIN DEPTH ITERATION NODE COUNTS.

         SETQ   ITERS,(PLUS,ITERS,1)
         SETX   NDPREV,0
         SETX   LDEPM1,2
BAS26    COND   (GE,LDEPM1,ITERS),BAS27
         SETX   NDPREV,(PLUS,(INDEX,NDDIT,LDEPM1),NDPREV)
         SETX   LDEPM1,(PLUS,LDEPM1,1)
         GOTO   BAS26

BAS27    SETA   (IDXLOC,NDDIT,(LODREL,LDEPM1)),(MINUS,NDTTL,(LODREL,NDP
,REV))
         COND   (EQ,(INDEX,NSRCH,NPLYC),1),BASE7  IF ONLY 1 LEGAL MOVE
         COND   (GE,ITERS,NPLY/2),BAS40  NPLY/2 IS FAR ENOUGH
         COND   (GT,(ABSF,(INDEX,ALPHA,OSIDE)),INFIN-NPLY*2000B),BAS40

*        USE AUTO-TIME CONTROL (IF ON) OR FDEPTH.

         COND   (TRUE,(TEST,SWICH,S.TIM)),BAS28
         COND   (GE,ITERS,FDEPTH),BAS40  IF PAST SPECIFIED DEPTH
         GOTO   BAS29             GO DEEPER

*        AUTO-TIME CONTROL.

BAS28    BSS    0
         CALLE  TIMUSD            TIME USED ON MOVE SO FAR
         SA6    TIMUS             SAVE
         SA6    TIMLI             SAVE START TIME OF NEXT ITERATION
         COND   (TRUE,(TEST,PONDR,S.THG)),BA28A
         COND   (TRUE,(TEST,PONDR,S.THI)),BAS29
BA28A    BSS    0
         COND   (LT,TIMUS,TIMIN),BAS29   IF OK TO GO DEEPER
         COND   (LT,ITERS,MDEPTH),BAS29   IF YET TO DO MIN DEPTH
         COND   (GT,ITERS,MDEPTH),BAS40  DONE IF AT LEAST CONFIRMED

*        CONFIRM THAT MDEPTH BSTMV IS OK BY SEARCHING IT MDEPTH+1.

         SETBIT WIERD,S.WRTC      FLAG PENDING CONFIRMATION

*        SETUP FOR RE-SEARCH (CLEAR SEARCH MODE BITS, SET A-B WINDOW,
*        ETC.)

BAS29    BSS    0
         SETX   PAR,(INDEX,ALPHA,OSIDE)
         SETAB  (MINUS,PAR,FLXTOL),(PLUS,PAR,FLXTOL)
         SETQ   SCORI,PAR         SAVE ITERATION SCORE
         RELEAS PAR
         SETX   IPLY,0            LOOP TO COPY MAIN LIMB TO KILLS
BAS30    SETA   (IDXLOC,KILLS,(PLUS,IPLY,NPLYC)),(AND,(LIMB,NPLYC,IPLY)
,,MCOMP)
         SETX   IPLY,(PLUS,IPLY,1)
         COND   (LT,IPLY,(INDEX,LIMBL,NPLYC)),BAS30
         RELEAS IPLY
         SETA   (IDXLOC,KILLS,(PLUS,NPLYC,(INDEX,LIMBL,NPLYC))),0
BAS31    SETA   (IDXLOC,LIMBL,NPLYC),0
         SETQ   BSTMV,0
         SETX   MMASK,(MASK,42)   MASK TO CLEAR SEARCHED BITS
         SELOOP
         SETA   (SELST,0),(CAND,MMASK,SMOVE)
         SELEND
         RELEAS MMASK
         GOTO   BAS13             RE-SEARCH EVERYTHING

*        NO MOVE FOUND.

BAS32    COND   (NZ,(INDEX,NSRCH,NPLYC)),BAS34  IF AIM WAS TOO HIGH

         BX1    X1-X1             UNJIGGLE LETS
         RJ     =XJIGLET
         SETQ   VMOVE,(ORF,VMOVE,(LSHIFT,1,S.MAT))
         COND   (TRUE,(TEST,VMOVE,S.CHK)),BAS33
         SCORE  DRAWS             STALEMATE

BAS33    SCORE  (XOR,SSIDE,-INFIN)  CHECKMATE

*        RE-ASPIRE.

BAS34    SETA   (IDXLOC,ALPHA,OSIDE),(XOR,SSIDE,-INFIN)
         SETBIT WIERD,S.WRA0      SET WIERD CONDITION FLAG
         CLRBIT WIERD,S.WRA1
         SETQ   OBVIO,0
         SETA   (IDXLOC,KILLS,(PLUS,2,NPLYC)),0  FLAG NO MAIN VARIATION
         GOTO   BAS31             SETUP TO RE-SEARCH

*        UNJIGGLE AND RETURN.

BAS40    SX1    0                 UNJIGGLE LETS
         RJ     =XJIGLET
         CLRBIT PONDR,S.THQ        CLEAR PONDER SEARCH QUIT FLAG
RETURN   SPACE  4,10
**       RETURN - SCORE FINAL BESTVAL (ALPHA OR BETAR) AND RETURN.

         ENTRY  RETURN
RETURN   SCORE

         END
*LEVEL *
EVMINI
*FILE LINP
         IDENT  EVMINI
         EXT    DRAW,REPD,RETURN,MIN70
         INIT                     INIT MACRO SYSTEM

*CALL VERSION
 EVALU8   TITLE  EVALU8 - CHESS 'V' RECURSIVE EVALUATOR.
*CALL IOCOM
```

```
*CALL     LET
*CALL     TLET
*CALL SQRLST
*CALL BOARD
*CALL CONST
*CALL STACKS
*CALL DEBUG

 EEVMINI  ECERTB EEVFULL

**        CONSTANT DATA.

 WMJPC    VFD    10/0,4/17B,8/0,4/17B,34/0  WHITE MAJOR PIECES
 M.ALP    EQU    M.CAP+M.PRO+M.CAS+M.CHK
 SVRNK    VFD    60/0,21/0,8/377B,31/0  WHITE SEVENTH RANK
          VFD    31/0,8/377B,21/0,60/0  BLACK SEVENTH RANK
 MBPWN    VFD    26/0,4/17B,26/0,4/17B

*         NODE PRIVATE TEMPORARY EQUIVALENCES.

 POSISH   EQU    NDTMP+2           POSITIONAL SCORE

 PROPS    EQU    NDTMP             PAWNS ON 7TH RANK (2 WORDS)
 MINI     TITLE  MINI - GET POSITIONAL SCORE + EXCHANGE ANALYSIS.
**        MINI - GET POSITIONAL SCORE + EXCHANGE ANALYSIS.
*
*         ENTERED DIRECTLY FROM EVALU8 AND EVFULL.

          ENTRY  MINI0
 MINI0    BSS    0

*         CHECK FOR POSITION REPETITION OR TRANSPOSITION.

          RJ     =XEVTRAP

*         COME HERE ALSO FOR LOWER PLY WOOD.

*         CHECK STATUS DETERMINES WHETHER ALL MOVES ARE TO BE SEARCHED.

 MIN0E    COND   (GE,NPLYC,NPLY-3),MINI1  NO SEARCHING IF PLY TOO HIGH
          COND   (TRUE,(TEST,VMOVE,S.CHK)),MIN70

*         KING IS NOT IN CHECK, LOOK AT NULL MOVE.

*         ALSO COME HERE FOR UPPER WOOD, REGARDLESS OF CHECK STATUS.

 MINI1    BSS    0

          SETQ   NDSCR,(PLUS,NDSCR,1)  INCREMENT DEBUG COUNT

*         REMOVE 50-MOVE RULE STUFF.

**        BRANCH TO APPROPRIATE POSITIONAL SCORER DEPENDING ON ROUGH
*         MATERIAL COUNTS FROM MATERIAL BALANCE WORD.

          SETX   MBVAL,MBVAL
          SETX   CMMASK,(MASK,60-L.CWM)
          GOTO   (MINI2,MINI3,MINI3,MINI3,MINI4,MIN13,MIN13,MIN13,MINI4,
,MIN13,MIN13,MIN13,MINI4,MIN13,MIN13,MIN13),(PLUS,(LSHIFT,(ANDC,(LSHIFT,
,MBVAL,-S.CWM),CMMASK),L.CWM),(ANDC,(LSHIFT,MBVAL,-S.CBM),CMMASK))
          RELEAS CMMASK

*         NEITHER SIDE HAS MORE THAN 700B MATERIAL POINTS.

 MINI2    COND   (ZR,MBVAL),DRAW   TWO BARE KINGS
          COND   (ZR,MBSCR),MIN13  IF EQUAL MATERIAL
          COND   (PL,MBVAL),MINI4  IF WHITE EDGE

*         BLACK EDGE, WHITE WEAK.

 MINI3    SETX   MBVAL,(LSHIFT,MBVAL,30)

*         ONE SIDE HAS .LT. 7*64, TRY CHECKMATING.  CRITERIA FOR USING
*         CHECKMATING ALGORITHM...
*         1. MUST HAVE MATERIAL EDGE OF AT LEAST FMTEGE UNLESS
*         NEITHER SIDE HAS PAWNS.
*         3.     DONT TRY TO MATE IF HAVE PAWNS BUT NO Q OR R.

 MINI4    COND   (ZR,(AND,MBVAL,MBPWN)),MINI5
          COND   (LT,(ABSF,MBSCR),FMTEGE),MIN13
          COND   (NZ,(AND,MBVAL,WMJPC)),MINI5  HAS QUEEN OR ROOK
          COND   (NZ,(ANDC,(LSHIFT,MBVAL,-30),(MASK,60-4))),MIN13
          RELEAS MBVAL

**        DO MOP-UP EVALUATION.

 MINI5    SETQ   NDMOP,(PLUS,NDMOP,1)  INCREMENT DEBUG COUNT
          RJ     =XEVMOPS          GET MOPUP POSITIONAL SCORE
          GOTO   MIN48             FOLD IN WITH MATERIAL
 REGEV    EJECT
**        REGULAR POSITIONAL EVALUATION.

 MIN13    BSS    0

*         FIRST TRY TO REFUTE OR IGNORE ON BASIS OF MATERIAL ALONE.

          COND   (EQ,NPLYD,1),MIN15  DONT IF DOING PRELS
          SETX   MATSCOR,MBSCR     NO MORE 50-MOVE RULE STUFF
          SETX   BEST,(PLUS,(INDEX,POSMIN,(MINUS,1,OSIDE)),MATSCOR)
          MINMAX NEWVAL=(LODREL,BEST),IGNORE=MIN51,NON=PERM
          SETX   WORST,(PLUS,(INDEX,POSMIN,OSIDE),MATSCOR)
          MINMAX NEWVAL=WORST,REFUTE=MIN14,NON=PERM
          GOTO   MIN15

          RELEAS MATSCOR
 MIN14    SCORE  (LODREL,WORST)

 MIN15    SETQ   NDREG,(PLUS,NDREG,1)  INCREMENT DEBUG COUNT
          RJ     =XEVPOSS          GET REGULAR POSITIONAL SCORE

*         UPDATE DYNAMIC MATERIAL SCORE RELIABILITY ESTIMATE.

 MIN48    SETX   POSCOR,(LSHIFT,POSISH,-6)
          COND   (EQ,NPLYD,1),MIN49  IF DOING PRELS
          SETQ   POSMIN,(MINF,POSMIN,POSCOR)
          SETQ   POSMAX,(MAXF,POSMAX,POSCOR)
          GOTO   MIN50

 MIN49    SETQ   POSINI,(PLUS,POSINI,POSCOR)
          RELEAS POSCOR
          SETQ   POSCNT,(PLUS,POSCNT,1)  ADVANCE POSITIONAL SCORE COUNT

 MIN50    BSS    0

*         COMBINE MATERIAL BALANCE AND POSITIONAL SCORE.
*         50-MOVE RULE STUFF REMOVED.
          SETX   FSCOR,(PLUS,(LSHIFT,POSISH,-6),MBSCR)
          MINMAX NEWVAL=(LODREL,FSCOR),REFUTE=RETURN
 XCHANG   EJECT
*         DO REAL EXCHANGE ANALYSIS IF MINI, ESTIMATED IF PREL OR
*         PLY TOO DEEP.

 MIN51    GENMOV CAPTURES
          COND   (GE,NPLYC,NPLY-3),MIN62  NO SEARCHING IF PLY TOO HIGH
          SETX   MODE,(CAND,(MASK,60-L.MOD+1),(LSHIFT,VMOVE,-S.MOD))
          COND   (EQ,MODE,PREL),MIN62  PRELIMINARY SCORES
          COND   (EQ,MODE,WOOD),MIN65  IF CAPTURE SEARCH ONLY
          RELEAS MODE

*         DECIDE WHETHER TO AUGMENT CAPTURES WITH CHECK-QUIESCENCE.

          SETX   NPLYC,NPLYC
          SETX   NPLYD,NPLYD
          COND   (GE,NPLYD,(PLUS,ITERS,QADPTH,1)),MIN65
          COND   (FALSE,(TEST,VMOVE,S.CAP)),MI51A  IF LAST MOVE NOT CAP
          COND   (GE,NPLYD,(PLUS,ITERS,QDEPTH,1)),MIN65
 MI51A    COND   (LT,NPLYD,4),MIN65
          COND   (FALSE,(TEST,(INDEX,VARIE-2,NPLYC),S.CHK)),MIN65
          COND   (FALSE,(TEST,(INDEX,VARIE-4,NPLYC),S.CHK)),MIN65
          COND   (GE,(XOR,SSIDE,(INDEX,ALPHA,OSIDE)),FMTEGE),MIN65
          COND   (LE,NPLYD,(PLUS,ITERS,QDEPTH)),MIN52  IF LOW ENOUGH
          RELEAS NPLYD
          SETX   PLIND,(MINUS,(LODREL,NPLYC),2)  CHECK FOR PROMOTIONS
 MI51B    COND   (TRUE,(TEST,(INDEX,VARIE,PLIND),S.PRO)),MIN65
          SETX   PLIND,(MINUS,PLIND,2)  IF NOT PROMOTION, LOOP
          COND   (GE,PLIND,(MINUS,NPLYC,NPLYD)),MI51B  LOOP THRU BRANCH
          RELEAS PLIND

*         QUIESCENCE SEARCH.  DO CAPTURES FIRST.

 MIN52    SETMAX -1
          SELOOP DIRECT=REVERSE
          COND   (TRUE,(TEST,SMOVE,S.QUES)),MIN54
          SETX   SMTO,(AND,SMOVE,M.MTO)
          SETX   SMFR,(LSHIFT,(AND,SMOVE,M.MFR),-L.MTO)
          RELEAS SMOVE
          COND   (MEMBS,(MINUS,(LOCF,ALATK),MSIDE),SMTO),MIN53
          SELMAX (INDEX,VALUEX,(ABSF,(INDEX,NBOARD,SMTO)))
          GOTO   MIN54

 MIN53    SELMAX (MINUS,(INDEX,VALUEX,(ABSF,(INDEX,NBOARD,SMTO))),(INDEX
,,VALUEX,(ABSF,(INDEX,NBOARD,SMFR))))
          RELEAS (SMTO,SMFR)
 MIN54    SELEND
          COND   (EQ,SMAX,-1),MIN56   IF NO MORE CAPTURES
          RELEAS SMAX
 MIN55    SEARCH MODE=QUES,REFUTE=RETURN
          GOTO   MIN52             LOOK FOR NEXT BEST CAPTURE

*         SEARCH CHECKS AND PAWN PROMOTIONS.

 MIN56    GENMOV ALL MOVES
          SETQ   INDEX,(LOCF,MOVES-LE.MOV)
 MIN57    SELOOP SELECT=MIN59,START=(PLUS,INDEX,LE.MOV)
          COND   (TRUE,(TEST,SMOVE,S.QUES)),MIN58
          SELMOV
 MIN58    SELEND
          SCORE

 MIN59    SETX   SMOVE,(INDEX,B0,INDEX)
          COND   (NZ,(AND,(LSHIFT,SMOVE,60-12),M.ALP/10000B)),MIN61
          RELEAS SMOVE
 MIN60    CALLE  CHKCHK
          PL     X6,MIN57
 MIN61    SEARCH MODE=QUES,REFUTE=RETURN
          GOTO   MIN57

*         ESTIMATE RESULT OF EXCHANGES BY FINDING CAPTURE WITH LARGEST
*         ESTIMATED GAIN.

 MIN62    SETMAX 0                 LOOK ONLY FOR PLUS RESULTS
          SELOOP
          SETX   SMTO,(AND,SMOVE,M.MTO)
          SETX   SMFR,(LSHIFT,(AND,SMOVE,M.MFR),-L.MTO)
          RELEAS SMOVE
          COND   (MEMBS,(MINUS,(LOCF,ALATK),MSIDE),SMTO),MIN63
          SELMAX (INDEX,VALUEX,(ABSF,(INDEX,NBOARD,SMTO)))
          GOTO   MIN64

 MIN63    SELMAX (MINUS,(INDEX,VALUEX,(ABSF,(INDEX,NBOARD,SMTO))),(INDEX
,,VALUEX,(ABSF,(INDEX,NBOARD,SMFR))))
          RELEAS (SMTO,SMFR)
 MIN64    SELEND
          COND   (EQ,NPLYD,1),MI64A  IF DOING PRELS
          COND   (ZR,SMAX),RETURN  IF NO GOOD CAPTURES
          SCORE  (PLUS,MBSCR,(XOR,SSIDE,SMAX))

 MI64A    BSS    0
          SCORE  (PLUS,(INDEX,ALPHA,OSIDE),(XOR,SSIDE,(LODREL,SMAX)))
          SPACE  4,10
**        DO REGULAR EXCHANGE ANALYSIS.
*
*         TRY CAPTURING PIECES IN ORDER OF DECREASING ESTIMATED GAIN,
*         WHERE ESTIMATED GAIN IS VALUE OF CAPTURED PIECE (UNDEFENDED),
*         OR VALUE OF CAPTURED PIECE MINUS VALUE OF CAPTURING PIECE
*         (DEFENDED).  SEARCH A GIVEN CAPTURE ONLY IF MATERIAL BALANCE
*         VALUE (UPDATED TO REFLECT CAPTURE) IS BETTER THAN CURRENT
*         BEST VALUE, OR WITHIN A TOLERANCE WHICH TAKES INTO
*         ACCOUNT POSSIBLE POSITIONAL SCORE CHANGE AT A DEEPER PLY.
*         ALSO TRY PAWN PROMOTIONS.

 MIN65    SETMAX
          SELOOP DIRECT=REVERSE
          COND   (TRUE,(TEST,SMOVE,S.WOOD)),MIN67
          SETX   SMTO,(AND,SMOVE,M.MTO)
          SETX   SMFR,(LSHIFT,(AND,SMOVE,M.MFR),-L.MTO)
          RELEAS SMOVE
          COND   (MEMBS,(MINUS,(LOCF,ALATK),MSIDE),SMTO),MIN66
          SELMAX (INDEX,VALUEX,(ABSF,(INDEX,NBOARD,SMTO)))
          GOTO   MIN67

 MIN66    SETX   TSMTO,(ABSF,(INDEX,NBOARD,(LODREL,SMTO)))
          SETX   TSMFR,(ABSF,(INDEX,NBOARD,(LODREL,SMFR)))
          COND   (EQ,TSMFR,KING),MIN67
          SELMAX (MINUS,(INDEX,VALUEX,TSMTO),(INDEX,VALUEX,TSMFR))
          RELEAS (TSMTO,TSMFR)
 MIN67    SELEND
          COND   (EQ,SMAX,-INFIN),MI68A  NO MORE CAPTURES
          RELEAS SMAX
          SETX   SMOVE,(INDEX,B0,INDEX)
          COND   (TRUE,(TEST,SMOVE,S.PRO)),MIN68  IF PROMOTION
          SETX   SMTO,(AND,(LODREL,SMOVE),M.MTO)
          COND   (ZR,(INDEX,NBOARD,SMTO)),MIN68  IF ENPASSANT
```

```
          SETX3  SMTO              SET UP PARAM FOR MBCAPT
          RELEAS SMTO
          CALLE  MBCAPT            COMPUTE POST-CAP MATERIAL BALANCE
          SETX   CMVAL,X7
          SETX   INDEX,INDEX       SET SO WONT TRY TO SEARCH AGAIN
          SETA   INDEX,(ORF,(LSHIFT,1,S.WOOD),(INDEX,B0,INDEX))
          RELEAS INDEX
          SETX   BEST,(PLUS,(INDEX,POSMIN,(MINUS,1,OSIDE)),CMVAL)
          MINMAX NEWVAL=(LODREL,BEST),IGNORE=MIN65,NON=PERM
          RELEAS CMVAL
 MIN68    SEARCH MODE=WOOD,REFUTE=RETURN
          GOTO   MIN65

*         GENERATE PAWN PROMOTIONS (QUEEN ONLY).

 MI68A    SETQS  PROPS,(ANDS,(INDEXS,SVRNK,(LSHIFT,OSIDE,1)),(INDEXS,TPL
,OC,(XOR,SSIDE,PAWN*2)))
          COND   (NULLS,PROPS),RETURN  IF NO PAWNS PROMOTING
          SETQ   LINDX,(LOCF,MOVES)  CLEAR MOVE LIST
          GENMOV FSLIST,(LOCF,PROPS)       GENERATE PAWN PROMOTIONS
          SETQ   INDEX,(LOCF,MOVES-LE.MOV)
 MI68B    SELOOP SELECT=MI68D,START=(PLUS,INDEX,LE.MOV)
          SETX   PQFLAG,M.ENP+M.CAS  MASK FOR QUEENS ONLY
          COND   (NE,(AND,SMOVE,PQFLAG),PQFLAG),MI68C
          RELEAS PQFLAG
          SELMOV
 MI68C    SELEND
          SCORE

 MI68D    SEARCH MODE=WOOD,REFUTE=RETURN
          GOTO   MI68B
 WOOD     SPACE  4,10
**        WOOD - MATERIAL BALANCE CAPTURE SEARCH.

          ENTRY  WOOD0
 WOOD0    BSS    0

          COND   (GE,NPLYD,(PLUS,ITERS,QADPTH,2)),MINI1
          GOTO   MIN0E             PLY NOT TOO HIGH GET OUT OF CHECK
 PREL     SPACE  4,10
**        PREL - GET PRELIMINARY POSITIONAL SCORE.
*
*         ENTERED DIRECTLY FROM EVALU8.
*
*         SAME AS HIGH PLY WOOD BUT ESTIMATE CAPTURES.

          ENTRY  PREL0
 PREL0    EQU    MINI1
 QUES     SPACE  4,10
**        QUES - DO QUIESCENCE SEARCH.
*
*         ENTERED DIRECTLY FROM EVALU8.
*
*         SAME AS MINI.

          ENTRY  QUES0
 QUES0    EQU    MINI0

          END
*LEVEL *
EVFULL
*FILE LINP
          IDENT  EVFULL
          EXT    MIN70,REPD,RETURN
          INIT                     INIT MACRO SYSTEM

*CALL VERSION
 EVALU8   TITLE  EVALU8 - CHESS 'V' RECURSIVE EVALUATOR.
*CALL IOCOM
*CALL     LET
*CALL     TLET
*CALL SQRLST
*CALL BOARD
*CALL CONST
*CALL STACKS
*CALL DEBUG

 EEVFULL  ECERTB EEVREST

**        CONSTANT DATA.

 MCOMP    CON    M.CAP+M.PRO+M.ENP+M.CAS+M.MFR+M.MTO
 CRFCT    DATA   0000777777000077777B  CLEAR KILLER COUNTS
 MSCPC    CON    M.CAP+M.PRO+M.CHK

*         NODE PRIVATE TEMPORARY EQUIVALENCES.

 ATKPCS   EQU    NDTMP            ATTACKED PIECES (2 WORDS)
 HNGPCS   EQU    NDTMP+2          HUNG PIECES (2 WORDS)
 SAFSQS   EQU    NDTMP+4          SAFE SQUARES (2 WORDS)

 REFWD    EQU    NDTMP            TEMP FOR REFUTATION WORD
 MVMSC    EQU    NDTMP+1          MOVE COMPARE MASK FOR EVREFM

 FULL     TITLE  FULL - DO FULL DEPTH/WIDTH SEARCH FOR FINAL SCORE.
**        FULL - DO FULL DEPTH/WIDTH SEARCH FOR FINAL SCORE.
*
*         ENTERED DIRECTLY FROM EVALU8.

          ENTRY  FULL0
 FULL0    BSS    0

*         USE MINI IF .GE. TERMINAL DEPTH.

          COND   (GT,NPLYD,ITERS),=XMINI0

*         CHECK FOR POSITION REPETITION OR TRANSPOSITION.

          RJ     =XEVTRAP

*         SCORE A DRAW IF NO PIECES.

          COND   (ZR,MBVAL),=XDRAW

*         RESET KILLER TABLE ENTRY 1 PLY UP.

          SETA (IDXLOC,KILLS+1,NPLYC),(AND,(INDEX,KILLS+1,NPLYC),CRFCT)


*         USE MINI CHECK ESCAPE CODE IF AT LAST FULL PLY (BUT NOT AT 1).

          COND   (LT,NPLYD,(MAXF,ITERS,2)),FULL1
          COND   (TRUE,(TEST,VMOVE,S.CHK)),FUL52 IF IN CHECK
 FULL1    BSS    0
          COND   (EQ,ITERS,1),FUL10  TRY TO REFUTE ALL WITH CAPTURES
          COND   (ZR,MAINV),FUL10  IF DONE WITH MAIN VARIATION

*         TRY MAIN VARIATION, IF APPLICABLE.

 FULL2    SETQ   REFWD,(INDEX,KILLS,NPLYC)
          COND   (NZ,REFWD),FULL3  IF A MOVE LEFT IN MAIN VAR
          GOTO   FUL10

 FULL3    SETQ   MVMSC,7777B       MUST COMPARE ONLY FROM-TO SQUARES
          RJ     =XEVREFM          GENERATE AND SELECT IF POSSIBLE
          NG     X6,FUL10          IF NOT FOUND
 FULL7    SEARCH MODE=FULL,REFUTE=REFUTE

*         TRY CAPTURES IN ORDER OF DECREASING ESTIMATED GAIN, DOWN TO
*         ESTIMATED GAIN OF ZERO.

 FUL10    GENMOV CAPTURES
          SETQ   MAINV,0           DONE WITH MAIN VARIATION
 FUL11    SETMAX -1
          SELOOP DIRECT=REVERSE
          COND   (TRUE,(TEST,SMOVE,S.FULL)),FUL13
          SETX   SMTO,(AND,SMOVE,M.MTO)
          SETX   SMFR,(LSHIFT,(AND,SMOVE,M.MFR),-L.MTO)
          RELEAS SMOVE
          COND   (ZR,(INDEX,NBOARD,SMTO)),FUL13
          COND   (MEMBS,(MINUS,(LOCF,ALATK),MSIDE),SMTO),FUL12
          SELMAX (INDEX,VALUEX,(ABSF,(INDEX,NBOARD,SMTO)))
          GOTO   FUL13

 FUL12    SELMAX (MINUS,(INDEX,VALUEX,(ABSF,(INDEX,NBOARD,SMTO))),(INDEX
,,VALUEX,(ABSF,(INDEX,NBOARD,SMFR))))
          RELEAS (SMTO,SMFR)
 FUL13    SELEND
          COND   (EQ,SMAX,-1),FUL17   IF NO MORE GOOD CAPTURES
          RELEAS SMAX
 FUL15    SEARCH MODE=FULL,REFUTE=REFUTE
          GOTO   FUL11

 FUL17    BSS    0

*         TRY A RECENT REFUTATION AT THIS PLY.

          SETQ   REFWD,(INDEX,KILLS,NPLYC)
          SETQ   MVMSC,MCOMP       SET MASK FOR EVREFM CALLS
          RJ     =XEVREFM          GENERATE AND SELECT BEST KILLER
          NG     X6,FU20A          IF NOT AVAILABLE
 FUL20    SEARCH MODE=FULL,REFUTE=REFUTE

*         TRY TRANSPOSITIONS TABLE MOVE BEFORE OTHER KILLERS.

 FU20A    SETQ   MVMSC,7777B       SET MASK FOR FROM, TO ONLY
          SETX   TMOVE,(CAND,(MASK,60-12),(LSHIFT,TRAWD,12))
          COND   (ZR,TMOVE),FUL21  IF NONE
          SETQ   REFWD,(LODREL,TMOVE)  ELSE TRY IT
          RJ     =XEVREFM          GENERATE AND SELECT
          NG     X6,FUL21          IF NOT FOUND
 FU20B    SEARCH MODE=FULL,REFUTE=REFUTE

*         NOW TRY 2ND BEST KILLER.

 FUL21    SETQ   MVMSC,MCOMP       RESET MOVE COMPARE MASK
          SETQ   REFWD,(LSHIFT,(INDEX,KILLS,NPLYC),-30)
          RJ     =XEVREFM          TRY 2ND BEST KILLER
          NG     X6,FUL22          IF NO GOOD
 FU21B    SEARCH MODE=FULL,REFUTE=REFUTE

*         TRY REFMOV OF 2 PLIES EARLIER.

 FUL22    COND   (LT,NPLYD,2),FUL27  IF AT BASE PLY + 1
          SETQ   REFWD,(INDEX,KILLS-2,NPLYC)
          RJ     =XEVREFM          GENERATE AND SELECT
          NG     X6,FUL27          IF NO GOOD
 FUL25    SEARCH MODE=FULL,REFUTE=REFUTE

*         GET ATTACKED PIECES OUT OF WAY FIRST.

 FUL27    SETQS  ATKPCS,(ANDS,(INDEXS,ALLOC,MSIDE),(INDEXS,ALATK,(NOT,MS
,IDE)))
          SETQS  HNGPCS,(ANDCS,ATKPCS,(INDEXS,ALATK,MSIDE))
          SETQS  SAFSQS,(ANDCS,ONBRD,(INDEXS,ALATK,(NOT,MSIDE)))
          GENMOV FSLIST,(LOCF,HNGPCS)
          GENMOV FSLIST,(LOCF,ATKPCS)
          GENMOV ALL MOVES

*         IN 2-PLY SEARCH, TRY CHECKS, PROMOTIONS, CAPTURES FIRST.

          COND   (NE,ITERS,1),FUL28  IF NOT 2-PLY ITERATION
          SETQ   INDEX,(LOCF,MOVES-LE.MOV)
 FU27A    SELOOP SELECT=FU27C,START=(PLUS,INDEX,LE.MOV)
          COND   (TRUE,(TEST,SMOVE,S.FULL)),FU27B
          SELMOV
 FU27B    SELEND
          GOTO   FUL28             ALL DONE WITH CHECKS, ETC.

 FU27C    COND   (NZ,(AND,(INDEX,B0,INDEX),MSCPC)),FU27D
          CALLE  CHKCHK            CHECK FOR CHECK
          PL     X6,FU27A          IF NOT CHECK
 FU27D    SEARCH MODE=FULL,REFUTE=REFUTE
          GOTO   FU27A             TRY ANOTHER

*         SEARCH MOVES IN GENERATED ORDER, BUT FIRST THOSE TO SAFE SQS.

 FUL28    SETQ   INDEX,(LOCF,MOVES-LE.MOV)
 FUL29    SELOOP SELECT=FUL31,START=(PLUS,INDEX,LE.MOV)
          COND   (TRUE,(TEST,SMOVE,S.FULL)),FUL30
          COND   (NONMS,(LOCF,SAFSQS),(AND,SMOVE,M.MTO)),FUL30
          SELMOV
 FUL30    SELEND
          GOTO   FUL37

 FUL31    COND   (LT,NPLYD,ITERS),FUL36  IF NOT AT HORIZON - 1
          RJ     =XEVPARS          IS THIS MOVE WORTH SEARCHING
          NG     X6,FUL36          IF WORTH SEARCHING
          SETX   INDEX,INDEX       SET SO WONT TRY AGAIN
          SETA   INDEX,(ORF,(LSHIFT,1,S.SRC),(LSHIFT,1,S.FULL),(INDEX,B0
,,INDEX))
          RELEAS INDEX
          GOTO   FUL29

 FUL36    SEARCH MODE=FULL,REFUTE=REFUTE
          GOTO   FUL29

*         SEARCH ALL OTHER MOVES.

 FUL37    SETQ   INDEX,(LOCF,MOVES-LE.MOV)
 FUL38    SELOOP SELECT=FUL40,START=(PLUS,INDEX,LE.MOV)
          COND   (TRUE,(TEST,SMOVE,S.FULL)),FUL39
          SELMOV
 FUL39    SELEND
```

```
          GOTO   FUL46

 FUL40    COND   (LT,NPLYD,ITERS),FUL45  IF NOT AT HORIZON - 1
          RJ     =XEVPARS          MIGHT THIS MOVE MAKE PAR
          PL     X6,FUL38          IF NOT
 FUL45    SEARCH MODE=FULL,REFUTE=REFUTE
          GOTO   FUL38

*         ALL MOVES SEARCHED, CHECK FOR CHECKMATE OR STALEMATE.
*         ALSO CHECK FOR RE-ASPIRE IF AT PLY 1.

 FUL46    SETX   NPLYC,NPLYC
          COND   (ZR,(INDEX,NSRCH,NPLYC)),FUL49
          SETQ   REFWD,(LIMB,NPLYC)
          SETX   LIMBL,(INDEX,LIMBL,NPLYC)
          RELEAS NPLYC
          COND   (NE,NPLYD,1),FUL47
          COND   (ZR,LIMBL),FUL51
 FUL47    COND   (ZR,(LODREL,LIMBL)),RETURN

*         SAVE BEST MOVE (BUT NOT FIRST ONE SEARCHED) FOR TRANS/REF
*         TABLE.  ALSO SAVE IT IN KILLER TABLE IF MOVE IS NOT
*         CAPTURE OF PIECE JUST MOVED.

 FU47A    SETX   BMOVE,REFWD
          SETQ   TRAWD,(ORF,(ANDC,TRAWD,(MASK,12)),(AND,(LSHIFT,BMOVE,48
,),(MASK,12)))
          COND   (ZR,(AND,(XOR,VMOVE,BMOVE),M.MTO)),RETURN
          SETX   KILLER,(INDEX,KILLS,NPLYC)
          SETX   KMOVE,(AND,(LODREL,BMOVE),MCOMP)
          COND   (NE,(AND,KILLER,MCOMP),KMOVE),FUL48  IF NOT BEST LIKED
          SETA   (IDXLOC,KILLS,NPLYC),(PLUS,KILLER,(LSHIFT,1,18))
          SCORE                     WE HAVE UPPED MAIN KILLER POPULARITY

 FUL48    COND   (EQ,(AND,(LSHIFT,KILLER,-30),MCOMP),KMOVE),FU48A 2ND
          SETX   KILLER,(ORF,(ANDC,KILLER,(MASK,30)),(LSHIFT,KMOVE,30))
          RELEAS KMOVE
          GOTO   FU48B

 FU48A    SETX   KILLER,(PLUS,KILLER,(LSHIFT,1,48))  UP 2ND POPULARITY
 FU48B    COND   (LT,(LSHIFT,KILLER,-48),(LSHIFT,(LSHIFT,KILLER,30),-48)
,),FU48C
          SETX   KILLER,(LSHIFT,KILLER,30)  2ND PROMOTED TO FIRST
 FU48C    SETA   (IDXLOC,KILLS,NPLYC),(LODREL,KILLER)
          SCORE

 FUL49    SETQ   VMOVE,(ORF,VMOVE,(LSHIFT,1,S.MAT))
          COND   (TRUE,(TEST,VMOVE,S.CHK)),FUL50

*         STALEMATE.

          SCORE  DRAWS              DRAW

*         CHECKMATE.

 FUL50    SCORE  (XOR,SSIDE,(PLUS,-INFIN,(LSHIFT,NPLYD,10)))

*         RE-ASPIRE, UNLESS PONDR QUIT FLAG IS SET.

 FUL51    COND   (TRUE,(TEST,PONDR,S.THQ)),RETURN
          SETA   (IDXLOC,NSRCH,NPLYC),0
          SETA   (IDXLOC,ALPHA,OSIDE),(XOR,SSIDE,-INFIN)
          SETBIT WIERD,S.WRA1       SET WIERD CONDITION FLAG
          CLRBIT WIERD,S.WRA0
          SETX   MMASK,(MASK,42)    MASK TO CLEAR SEARCHED BITS
          SELOOP
          SETA   (SELST,0),(CAND,MMASK,SMOVE)
          SELEND
          RELEAS MMASK
          GOTO   FUL28              RE-SEARCH EVERYTHING

**        IN CHECK AT LAST FULL PLY.

 FUL52    SETQ   MAINV,0            SHOULD BE CLEAR
          GOTO   MIN70              SPECIAL CODE TO GET OUT OF CHECK

*         ALPHA-BETA REFUTATION MOVE FOUND.

 REFUTE   SETQ   REFWD,(INDEX,B0,INDEX)
          GOTO   FU47A              TREAT AS BEST MOVE

          END
*LEVEL *
EVREST
*FILE LINP
          IDENT  EVREST
          INIT                      INIT MACRO SYSTEM

*CALL VERSION
 EVALU8   TITLE  EVALU8 - CHESS 'V' RECURSIVE EVALUATOR.
*CALL IOCOM
*CALL     LET
*CALL     TLET
*CALL SQRLST
*CALL BOARD
*CALL CONST
*CALL STACKS
*CALL DEBUG
 LEGL     TITLE  LEGL - CHECK MOVE LEGALITY.

 EEVREST  ECERTB EEVCODE
**        LEGL - CHECK MOVE LEGALITY.
*
*         ENTERED DIRECTLY FROM EVALU8.

          ENTRY  LEGL0
 LEGL0    BSS    0

          SCORE  0                  ILLEGAL BIT SET BY TRSRCH
 MATE     SPACE  4,10
**        MATE - CHECK WHETHER MOVE IS CHECKMATE OR STALEMATE.
*
*         ENTERED DIRECTLY FROM EVALU8.

          ENTRY  MATE0
 MATE0    BSS    0

*         SEARCH MOVES TO FIND AT LEAST ONE LEGAL RESPONSE.

          GENMOV ALL MOVES
          SETQ   INDEX,(LOCF,MOVES-LE.MOV)
 MATE1    SELOOP SELECT=MATE3,START=(PLUS,INDEX,LE.MOV)
          SELMOV
          SELEND

*         NO LEGAL RESPONSES.

          SETQ   VMOVE,(ORF,VMOVE,(LSHIFT,1,S.MAT))
          COND   (TRUE,(TEST,VMOVE,S.CHK)),MATE2

*         STALEMATE.

          SCORE  0

*         CHECKMATE.

 MATE2    SCORE  (XOR,SSIDE,(PLUS,-INFIN,(LSHIFT,NPLYC,10)))

*         SEARCH MOVE TO DETERMINE LEGALITY.

 MATE3    SEARCH MODE=LEGL,NON=PERM,ILLEGAL=MATE1

*         FOUND A LEGAL MOVE.

          SCORE  0
 POND     SPACE  4,10
***       POND - THINK (PONDER) ON OPPONENTS TIME.
*
*         CALLED FROM EVALU8 AT PLY 0.
*
*         ENTRY  (PONDR) = OPPONENTS PREDICTED MOVE.
*                ALL MOVES MUST HAVE BEEN GENERATED, AS BY LSTMOV.
*
*         THE PONDR MOVE IS SEARCHED IN BASE MODE, SO THE TREE SEARCH
*         RESEMBLES THE NORMAL ONE EXCEPT THAT IT BEGINS AT PLY 1
*         INSTEAD OF PLY 0.  TIME CONTROL IS ALSO DONE DIFFERENTLY,
*         SINCE ALL THE TIME UP TO THE MOMENT THE OPPONENT ACTUALLY
*         MAKES HIS MOVE IS FREE.

          ENTRY  POND0

 PONDA    CON    M.CAP+M.PRO+M.ENP+M.CAS+M.MFR+M.MTO

 POND0    BSS    0
          SETAB  -INFIN,INFIN
          SELOOP SELECT=POND3
          COND   (NZ,(AND,(XOR,SMOVE,PONDR),PONDA)),POND1
          SELMOV
 POND1    SELEND

*         ERROR - COULDNT FIND PONDR MOVE.

 POND2    SX5    =C* PONDERING ABANDONED.*
          RJ     =XERRMSG
          SETQ   PONDR,0            ABANDON PONDERING
          SCORE  0

*         FOUND PONDR MOVE, SO SEARCH IT.

 POND3    SEARCH MODE=BASE,IGNORE=POND2
          CLRBIT PONDR,S.THQ        CLEAR PONDER SEARCH QUIT FLAG
          COND   (ZR,BSTMV),POND2  IF NO MACHINE MOVE
          SETQ   PVMOV,(INDEX,B0,INDEX)  SAVE VMOVE WORD FOR MYMOVE
          SCORE

 EVPNTC   SPACE  4,10
***       EVPNTC - COMPUTE PONDER TIME CONTROL.
*
*         CALLED FROM PNDPAU AFTER YPMOVE HAS SIGNALLED THAT THE
*         OPPONENT HAS INDEED MADE THE EXPECTED MOVE.  EVPNTC
*         RE-ADJUSTS THE TIME CONTROL TO REFLECT THAT FREE TIME
*         IS NOW OVER.  IF CURRENT ITERATION IS ALREADY BEYOND
*         THAT WHICH WOULD BE REASONABLE HAD THE TIME NOT BEEN
*         FREE, THEN THE REST OF THE SEARCH IS SHORT-CIRCUITED
*         AND THE BEST MOVE SO FAR IS USED.

          ENTRY  EVPNTC

 EVPNTC   EQ     *+400000B         ENTRY/EXIT
          RJ     =XTIMUSD          GET CPTIME SO FAR
          SA6    TIMPO             SAVE FREE TIME
          SETX   TPMPM,(PLUS,(LSHIFT,(TIMES,TPONBN,TIMPO),-6),TIMPM)
          SETQ   TIMIN,(LSHIFT,(TIMES,TPMPM,NTRLO),-6)
          RELEAS TPMPM
          COND   (LE,TIMPO,TIMIN),EVPNTC  IF HAVENT USED MUCH TIME
          COND   (FALSE,(TEST,SWICH,S.TIM)),EVPNTC  IF NOT TIMING
          COND   (GE,TIMPO,TIMAX),EVPNTC1  IF CLEARLY TOO MUCH TIME
          COND   (LE,TIMLI,TIMIN),EVPNTC  IF OK TO FINISH THIS ITERATION

*         TOO MUCH TIME: MUST CURTAIL REST OF SEARCH.

 EVPNTC1  SETBIT PONDR,S.THQ  SET FLAG TO QUIT SEARCH
          EQ     EVPNTC            RETURN
 REPD     SPACE  4,6
**        REPD - DRAW BY REPETITION.
*
*         ENTRY  (X6) = 0 IF ONLY SINGLE REPETITION.

          ENTRY  REPD
 REPD     ZR     X6,DRAW           IF ONLY SINGLE REPETITION
          SETQ   VMOVE,(ORF,VMOVE,(LSHIFT,1,S.REP))  SET REPETITION FLAG
 DRAW     SPACE  4,10
**        DRAW - DRAW BY REPETITION OR STALEMATE, ETC.

          ENTRY  DRAW
 DRAW     SCORE  DRAWS

          END
*LEVEL *
EVCODE
          IDENT  EVCODE

***       EVCODE - PURE COMPASS ROUTINES FOR EVALU8.

          SST
          B1=1

*CALL IOCOM
*CALL     LET
*CALL     TLET
*CALL SQRLST
*CALL BOARD
*CALL CONST
*CALL STACKS
*CALL DEBUG

 EEVCODE  ECERTB EEVCHEK
 MBEVAL   TITLE  MBEVAL - MATERIAL BALANCE EVALUATOR.
***       MBEVAL - MATERIAL BALANCE EVALUATOR.
*
*         CALLED BY CREATE/UPDATE.
*
*         ENTRY  (X1) = MATERIAL SIGNATURE.
*
```

```
*         EXIT   (X6) = SIGNATURE WITH VALUES INSERTED.
*
*         FORMAT OF MATERIAL BALANCE WORD.
*
*         8/WUA,2/CWM,20/WS,8/WLA,2/CBM,20/BS, WHERE:
*         WUA = TOP 8 BITS OF SIGNED WHITE MATERIAL ADVANTAGE.
*         CWM = COARSE TOTAL WHITE MATERIAL / 512.
*         WS = WHITE MATERIAL SIGNATURE =
*         4/WQ,4/WB,4/WN,4/WR,4/WP, WHERE
*         WQ IS COUNT OF WHITE QUEENS, ETC.
*         WLA = LOWER 8 BITS OF SIGNED WHITE MATERIAL ADVANTAGE.
*         CBM = COARSE TOTAL BLACK MATERIAL / 512.
*         BS IS FORMATTED ANALOGOUSLY TO WS BUT FOR BLACK.
*
*         USES   A2,A4,A5,X0,X7,B4,B5,B6.

          ENTRY  MBEVAL

 MBEVAL   EQ     *+400000B         ENTRY/EXIT

*         COMPUTE WHITE MATERIAL ADVANTAGE FROM STANDARD PIECE VALUES.

          MX0    60-4              MASK FOR MATERIAL COUNTS
          SB6    B0                SIDE PASS COUNTER
 MBEVAL1  SA5    VALUEP            VALUE OF PAWN
          BX7    -X0*X1            COUNT OF PAWNS
          SB5    X7                SAVE
          IX6    X5*X7             MULTIPLY BY PAWN VALUE
          SA5    A5+B1             VALUE OF ROOK
          LX1    60-4              RIGHT ADJUST ROOK COUNT
          BX7    -X0*X1            COUNT OF ROOKS
          IX7    X5*X7             MULTIPLY BY ROOK VALUE
          IX6    X6+X7             ADD TO PAWN CREDIT
          SA5    A5+B1             VALUE OF KNIGHT
          LX1    60-4
          BX7    -X0*X1            COUNT OF KNIGHTS
          IX7    X5*X7
          IX6    X6+X7
          SA5    A5+B1             VALUE OF BISHOP
          LX1    60-4
          BX7    -X0*X1            COUNT OF BISHOPS
          IX7    X5*X7
          IX6    X6+X7
          SA5    A5+B1             VALUE OF QUEEN
          LX1    60-4
          BX7    -X0*X1            COUNT OF QUEENS
          IX7    X5*X7
          IX6    X6+X7
          LX1    60-10-4           COMPLETE A 30 BIT SHIFT
          NZ     B6,MBEVAL2        IF BOTH SIDES COUNTED
          BX4    X6                ELSE SAVE BLACK MATERIAL
          SB4    B5                SAVE BLACK PAWN COUNT
          SB6    B1                INCREASE SIDE PASS COUNTER
          EQ     MBEVAL1           LOOP FOR WHITE PIECES

*         USE SPECIAL TABLE, IF POSSIBLE, FOR LOW MATERIAL VALUES.

 MBEVAL2  IX7    X6+X4             WHITE PLUS BLACK MATERIAL
          AX7    10                GIVES 0 IF LESS THAN 16
          NZ     X7,MBEVAL4        IF NOT LOW MATERIAL SITUATION
          SA5    MBMSK             10/1777B,20/0,10/1777B,20/0
          SB6    LMBTAB-1          LENGTH MINUS ONE OF TABLE
 MBEVAL3  SA2    MBTAB+B6          NEXT TABLE ENTRY
          BX7    X1-X2             COMPARE TO SIGNATURE
          BX7    -X5*X7            CLEAR VALUE FIELDS
          ZR     X7,MBEVAL10       IF MATCH, USE TABLE ENTRY
          MX7    8
          BX0    X7
          LX0    30
          BX7    X0+X7
          BX0    X2
          LX2    30                ELSE TRY SWITCHING SIDES
          BX0    -X0*X7            COMPLEMENT SCORE FROM TABLE
          BX2    -X7*X2            CLEAR OUT BOGUS SHIFTED SCORE
          BX2    X2+X0             MERGE IN COMPLEMENTED SCORE
          BX7    X1-X2
          BX7    -X5*X7
          ZR     X7,MBEVAL10       IF MATCH WITH SWITCHED ENTRY
          SB6    B6-B1             ELSE LOOP FOR NEXT ENTRY
          PL     B6,MBEVAL3        IF MORE TABLE ENTRIES

**        COMPUTE BASIC MATERIAL SCORE = ISIGN(MIN0(255*64,MIN0(FMAXMT,
*               IABS(WE))+FTRADE*IABS(WE)*(FTRDSL-TM)*(4*EP+FTRPOK)
*               /(4*EP+FTRPWN)/64/64/64),WE)
*               WHERE WE = WHITE EDGE, TM = TOTAL MATERIAL,
*               EP = PAWN COUNT OF SIDE WITH + EDGE, AND
*               FMAXMT,FTRADE,FTRDSL, AND FTRPWN ARE LETS.
*               FTRPOK IS A LET (TLET).
*
*         PRINCIPLES OF TRADE DOWN ADJUSTMENTS.
*
*         WHEN AHEAD, TRADE PIECES BUT NOT PAWNS.
*         WHEN BEHIND, TRADE PAWNS BUT NOT PIECES.

 MBEVAL4  IX0    X6-X4             WHITE EDGE (ROUGHLY IN PAWNS * 2**6)
          BX7    X0                GET IABS(WE)
          AX7    59
          BX7    X7-X0
          SA5    FTRDSL
          IX5    X5-X6
          IX5    X5-X4             FTRDSL - TM
          IX5    X5*X7
          SA2    FTRADE
          IX5    X5*X2             FTRADE*IABS(WE)*(FTRDSL-TM)
          PL     X0,MBEVAL5        IF WHITE EDGE
          SB5    B4+0              ELSE USE BLACK PAWN COUNT
 MBEVAL5  SA2    FTRPOK            PAWN TRADE-DOWN RELAXATION
          SB5    B5+B5             MULTIPLY EDGY PAWN COUNT BY 4
          SB5    B5+B5
          SX2    X2+B5             4*EP+FTRPOK
          IX5    X5*X2
          SA2    FTRPWN            PAWN TRADE DOWN FACTOR
          SX2    X2+B5             FTRPWN + 4 * PAWN COUNT
          PX5    X5
          PX2    X2
          NX5    X5
          NX2    X2
          FX5    X5/X2
          UX5    X5,B5
          LX5    X5,B5
          AX5    18                /64/64/64
          SA2    FMAXMT
          IX2    X2-X7
          IX5    X5+X7             ...+IABS(WE)
          PL     X2,MBEVAL6        IF(FMAXMT.GE.IABS(WE))
          IX5    X5+X2             ...+MIN0(FMAXMT,IABS(WE))
 MBEVAL6  SB6    255*64            MAXIMUM SCORE
          SB5    X5
          AX0    59                GET SIGN OF WE
          GE     B6,B5,MBEVAL7
          SX5    B6+0              IF LARGER, USE MAXIMUM
 MBEVAL7  BX5    X0-X5             APPLY CORRECT SIGN

*         INSERT SCORE INTO SIGNATURE.

          MX7    60-8
          BX2    -X7*X5            WHITE LOWER ADVANTAGE BITS
          AX5    8
          BX5    -X7*X5            WHITE UPPER ADVANTAGE BITS
          LX5    52
          LX2    22
          BX5    X1+X5             INSERT UPPER SCORE BITS
          BX5    X5+X2             INSERT LOWER SCORE BITS

*         INSERT COARSE MATERIAL COUNTS INTO SIGNATURE.

          AX6    9                 TRUNCATE TO MULTIPLE OF 8 PAWNS
          SB6    X6
          SB5    3
          LE     B6,B5,MBEVAL8     IF WHITE . LT. 32 POINTS
          SX6    3                 ELSE DUMP INTO LAST PARTITION
 MBEVAL8  LX6    60-2-8            INSERT WHITE INTO SIGNATURE
          BX5    X5+X6
          AX4    9
          SB6    X4
          LE     B6,B5,MBEVAL9     IF BLACK .LT. 32
          SX4    3                 ELSE USE HIGHEST PARTITION
 MBEVAL9  LX4    30-2-8            INSERT BLACK INTO SIGNATURE
          BX6    X5+X4
          EQ     MBEVAL            RETURN

*         USE SPECIAL TABLE ENTRY.

 MBEVAL10 BX6    X2
          EQ     MBEVAL            RETURN


**        MBTAB - TABLE OF SPECIAL MATERIAL BALANCE VALUES.

 MBTAB    BSS    0

 0        MBTAB  (WN)
 0        MBTAB  (WB)
 60B      MBTAB  (WR,BN)
 20B      MBTAB  (WR,BB)
 20B      MBTAB  (WN,WN)
 100B     MBTAB  (WB,WN)
 40B      MBTAB  (WP,BN)
 20B      MBTAB  (WP,BB)
 100B     MBTAB  (WP,WP,BN)
 60B      MBTAB  (WP,WP,BB)
 160B     MBTAB  (WP,WP,WP,BN)
 120B     MBTAB  (WP,WP,WP,BB)

 LMBTAB   EQU    *-MBTAB           LENGTH OF MBTAB
 PSEVAL   TITLE  PSEVAL - PAWN STRUCTURE ANALYZER.
***       PSEVAL - PAWN STRUCTURE ANALYZER.
*
*         CALLED BY CREATE/UPDATE.
*
*         ENTRY  (X1) = PAWN STRUCTURE SIGNATURE.
*
*         EXIT   (X6) = SIGNATURE WITH DESCRIPTORS INSERTED.
*
*         FORMAT OF SIGNATURE WORD IS 30/WS,30/BS, WHERE
*                WS = VFD 12/WD,6/WQC,6/WMC,6/WKC.
*                WD = WHITE PAWN COLUMN DESCRIPTOR, THE MEANING OF
*                WHOSE BITS, FROM LEFT TO RIGHT, IS...
*                11.  DOUBLED  (OTHER BITS DESCRIBE MOST ADVANCED PAWN)
*                10.  MAJORITY. SET IF SIDE HAS PAWN MAJORITY IN 3 COLS
*                9.   PASSED. SET IF PAWN IS PASSED.
*                8.   ATTACKED. SET IF PAWN IS ATTACKED BY ENEMY PAWNS
*                7-6. DEFENSE. DESCRIBES ACTUAL AND POTENTIAL P DEFENSES
*                     00 = NO ACTUAL OR POTENTIAL (IF MOVED UP) DEFENSES
*                     01 = NO ACTUAL, BUT ONE OR MORE POTENTIAL DEFENSES
*                     10 = ONE ACTUAL PAWN DEFENSE.
*                     11 = TWO ACTUAL PAWN DEFENSES.
*                5.   SET FOR BLOCK OR CONFRONTATION NEXT ROW.
*                4-3. POTENTIAL CONFRONTATION FLUIDITY MEASURE.
*                     00 = PAWN WILL BE BLOCKED BY ENEMY PAWN.
*                     01 = P WILL BE SUPER-OPPOSED (2 ATTACKS, 0 DEFENS)
*                     10 = SEMI-FLUID (ONE MORE ATTACK THAN DEFENSE)
*                     11 = FLUID (AT LEAST AS MANY DEFENSES AS ATTACKS)
*                2-0. CODE FOR HOLES IN TARGET (MIDDLE) COLUMN.
*                     000 = NO HOLES (ENEMY PAWNS CAN HIT 5TH AND 6TH R)
*                     001 = 6TH RANK UNSUPPORTED HOLE.
*                     010 = 6TH RANK HOLE SUPPORTED BY A PAWN.
*                     011 = UNUSED.
*                     100 = 5TH AND 6TH RANK HOLES, BOTH UNSUPPORTED.
*                     101 = HOLES ON 5TH AND 6TH, 6TH SUPPORTED.
*                     110 = HOLES ON 5TH AND 6TH, 5TH SUPPORTED.
*                     111 = HOLES ON 5TH AND 6TH, BOTH SUPPORTED.
*                WQC,WMC,WKC ARE COLUMN PAWN LOCATORS, CONTAINING BITS
*                FOR EACH RANK FROM 2ND TO 7TH (LEFT TO RIGHT WITHIN
*                FIELD, AND SUCH THAT RANKS ARE COUNTED FROM THE POINT
*                OF VIEW OF THE SIDE WITH THE PAWN).  WQC IS THE
*                COLUMN ADJACENT TO THE TARGET (MIDDLE) COLUMN ON THE
*                QUEENS SIDE, AND WKC IS THE ADJACENT COLUMN ON THE
*                KINGS SIDE.  WMC IS THE TARGET COLUMN.
*                BS IS FORMATTED ANALOGOUSLY TO WS BUT FOR BLACK.
*
*         USES   ALL REGISTERS EXCEPT B2,B3.

          ENTRY  PSEVAL

 PSEVAL   EQ     *+400000B         ENTRY/EXIT

          SB4    B0                INITIALIZE SIDE LOOP COUNTER
          SB5    B0                INITIALIZE DESCRIPTOR
          SX0    373737B           MASK FOR ADVANCING PAWN WEDGES

*         LOOP FOR WHITE AND BLACK SIDES.

 PSEVAL1  BSS    0

*         COUNT PAWNS IN OUR TARGET COLUMN, FOR DOUBLED FLAG, ETC.

          MX5    6
          LX5    60-18
          BX6    X5*X1             EXTRACT OUR TARGET COLUMN
          CX7    X6                COUNT OUR PAWNS
          SB7    X7
          ZR     B7,PSEVAL10       IF NO PAWN IN COLUMN
          LX6    24                RIGHT ADJUST COLUMN
```

```
          EQ     B7,B1,PSEVAL2     IF ONLY ONE PAWN
          SB5    4000B             ELSE SET DOUBLED FLAG

*         FIND OUR MOST ADVANCED PAWN IN TARGET COLUMN.

 PSEVAL2  SX7    X6-1
          BX7    X6-X7
          NX7    X7,B7
          SB7    B7-42             0=2ND RANK,1=3RD,...,5=7TH

*         FIND ACTUAL PAWN DEFENSES FOR TARGET PAWN.

          ZR     B7,PSEVAL5        NO DEFENSE POSSIBLE FOR 2ND RANK
          SX4    100010B           SET UP MASK TO FIND DEFENSES
          LX4    33
          AX4    X4,B7             POSITION MASK TO DEFENSE ROW
          BX3    X4*X1             EXTRACT PAWN DEFENSES
          BX7    X0                PREPARE DEFENSE MASK SHIFT MASK
          LX7    30
          ZR     X3,PSEVAL4        IF NONE, LOOK FOR POTENTIALS
          CX3    X3                ELSE COUNT THEM
          SX3    X3+B1             10 = 1 DEFENSE, 11 = 2 DEFENSES
          LX3    6
          SB5    B5+X3             INSERT INTO DESCRIPTOR
          EQ     PSEVAL5

*         SET FLAG FOR PASSED PAWN.

 PSEVAL3  SB5    B5+1000B          SET PASSED FLAG IN DESCRIPTOR
          EQ     PSEVAL10          LOOK FOR HOLES

*         FIND POTENTIAL PAWN DEFENSES FOR TARGET PAWN.

 PSEVAL4  BX4    X7*X4             MOVE DEFENSE MASK BACK A ROW
          LX4    1
          ZR     X4,PSEVAL5        NO MORE ROWS, NO POTENTIAL DEFENSES
          BX3    X4*X1             EXTRACT POTENTIAL DEFENDERS
          ZR     X3,PSEVAL4        IF NONE, LOOP FOR NEXT ROW BACK
          SB5    B5+100B           SET POTENTIAL DEFENSE FLAG

*         CHECK FOR ATTACKS AND CONFRONTATIONS WITH ENEMY PAWNS.

 PSEVAL5  SX7    B7-5
          ZR     X7,PSEVAL3        IF PAWN ON 7TH RANK
          SX6    20102B            INITIAL WEDGE (ATTACK AND BLOCK) MASK
          LX6    X6,B7
          BX7    X6*X1             EXTRACT ATTACKING PAWNS
          ZR     X7,PSEVAL6        IF PAWN IS NOT ATTACKED
          SB5    B5+400B           ELSE SET ATTACKED FLAG
          EQ     PSEVAL10          DONT BOTHER ABOUT POTENTIAL CONFRONTS

*         CHECK FOR CONFRONTATION NEXT ROW.

 PSEVAL6  BX6    X0*X6             ADVANCE WEDGE MASK
          LX6    1
          ZR     X6,PSEVAL3        NO POTENTIAL CONFRONT, PAWN IS PASSED
          SB7    B7+B1             ADVANCE CONFRONTATION ROW NUMBER
          BX7    X6*X1             EXTRACT CONFRONTING PAWNS
          ZR     X7,PSEVAL7        NO CONFRONTATION, TRY NEXT ROW
          SB5    B5+40B            SET NEXT ROW CONFRONTATION FLAG
          EQ     PSEVAL8

*         LOOP FOR OTHER POTENTIAL CONFRONTATIONS.

 PSEVAL7  BX6    X0*X6             ADVANCE WEDGE MASK
          LX6    1
          ZR     X6,PSEVAL3        IF PAWN IS PASSED
          SB7    B7+B1
          BX7    X6*X1
          ZR     X7,PSEVAL7        LOOP IF NO CONFRONTATION

*         CHECK FOR PAWN BLOCK.

 PSEVAL8  LX5    30                MASK FOR ENEMY TARGET COLUMN
          BX4    X5*X7             EXTRACT BLOCKING PAWN, IF ANY
          NZ     X4,PSEVAL10       IF PAWN IS BLOCKED

*         GET POTENTIAL FLUIDITY BY COMPARING DEFENSE AND ATTACK COUNTS.

          CX7    X7                COUNT ENEMY PAWN ATTACKS
          SX4    100010B           SET UP MASK TO FIND DEFENSES
          LX4    33
          AX4    X4,B7             POSITION MASK TO DEFENSE ROW
          BX4    X4*X1             EXTRACT PAWN DEFENSES
          CX4    X4                COUNT THEM
          IX4    X4-X7             COMPARE TO ATTACKS
          NG     X4,PSEVAL9        IF FEWER DEFENSES THAN ATTACKS
          SX4    0                 ELSE SET FLUID
 PSEVAL9  SX4    X4+3              1=SUPER-OPPOSED,2=SEMI-FLUID,3=FLUID
          LX4    3
          SB5    B5+X4             INSERT FLUIDITY INTO DESCRIPTOR

*         COMPUTE 5TH AND 6TH RANK HOLE DATA.

 PSEVAL10 SX5    200020B           FORM MASKS TO DETECT HOLE GUARDS
          LX4    X5,B1
          BX6    X4*X1             EXTRACT ENEMY HOLE GUARDS
          NZ     X6,PSEVAL12       NO 6TH, AND HENCE, NO 5TH RANK HOLES
          SX3    040004B           PREPARE MASK TO TEST HOLE SUPPORT
          SB5    B5+B1             SET 6TH RANK HOLE FLAG
          LX3    30
          BX7    X3*X1             EXTRACT 6TH RANK HOLE SUPPORT
          BX5    X5*X1             EXTRACT ENEMY 5TH-RANK HOLE GUARDS
          ZR     X7,PSEVAL11       IF NO 6TH RANK HOLE SUPPORT
          SB5    B5+1              ELSE SET IN DESCRIPTOR
 PSEVAL11 NZ     X5,PSEVAL12       IF NO 5TH RANK HOLE
          SB5    B5+3              ELSE SET IN DESCRIPTOR
          LX3    1                 MASK FOR 5TH RANK HOLE SUPPORT
          BX7    X3*X1
          ZR     X7,PSEVAL12       IF NO 5TH RANK HOLE SUPPORT
          SB5    B5+2              ELSE SET IN DESCRIPTOR

*         DONE WITH ONE SIDE DESCRIPTOR EXCEPT FOR MAJORITY FLAG.

 PSEVAL12 SX7    B5                INSERT DESCRIPTOR INTO SIGNATURE
          LX7    48
          BX1    X1+X7
          LX1    30                SWITCH SIDES
          SB4    B4+B1             INCREASE SIDE LOOP COUNTER
          SB5    B0                INITIALIZE DESCRIPTOR FOR BLACK
          LE     B4,B1,PSEVAL1     IF MUST YET DO BLACK

*         COMPUTE MAJORITY FLAG.

          MX6    60-18             MASK FOR ALL 3 COLUMNS
          BX7    -X6*X1            BLACK PAWNS
          LX6    30
          BX6    -X6*X1            WHITE PAWNS
          CX7    X7                COUNT BLACK PAWNS
          CX6    X6                COUNT WHITE PAWNS
          SX5    B1                PREPARE MAJORITY FLAG
          LX5    48+10             TENTATIVE SET WHITE MAJORITY
          IX7    X6-X7             WHITE MINUS BLACK
          BX6    X1                DEFAULT NO MAJORITY FLAG
          ZR     X7,PSEVAL14       IF NEITHER SIDE HAS MAJORITY
          PL     X7,PSEVAL13       IF WHITE HAS MAJORITY
          LX5    30                SET BLACK MAJORITY
 PSEVAL13 BX6    X1+X5             ADD APPROPRIATE MAJORITY FLAG
          EQ     PSEVAL            RETURN

 PSEVAL14 EQU    PSEVAL            RETURN
 TRAPRI   SPACE  4,20
**        TRAPRI - CALCULATE TRANSPOSITION TABLE ENTRY PRIORITY.
*
*         CALLED BY TRACKT IN CREATE.
*
*         ENTRY  (X5) = FIRST WORD OF TABLE ENTRY.
*                (X6) = 0 IF MATCH WITH TABLE, ELSE -1.
*
*         EXIT   (X6) = FIRST WORD OF TABLE ENTRY, BUT WITH
*                LOWER 18 BITS REPLACED BY SIGNED PRIORITY
*                DIFFERENCE, WHICH, IF .GE.0, IMPLIES THAT THE CURRENT
*                BOARD IS ELIGIBLE TO REPLACE THE TABLE ENTRY.
*
*                THE COMPLETE FUNCTION P(IT,IN,NT,NN,MT,MN) =
*                128*(1-MT)*(IN-IT)-(NN-NT)
*                M IS FOR PRESENCE (1) OR ABSENCE (0) OF MOVE STORED.
*                I = ITERATION NUMBER (ITERS), AND N = PLY (NPLYC).
*                THE SUFFIXES T AND N DIFFERENTIATE BETWEEN THE TABLE
*                ENTRY AND THE NEW (CURRENT) BOARD.
*
*         USES   ALL A AND X REGISTERS.

          ENTRY  TRAPRI

 TRAPRI   EQ     *+400000B         ENTRY/EXIT
          MX0    12                MASK FOR MT
          BX3    X0*X5             EXTRACT ENTRY MOVE
          SX6    B0                INIT PRIORITY TO 0
          LX5    30                RIGHT ADJUST IT
          NZ     X3,TRAPRI1        IF THERE IS A MOVE (MT=1)
          SA4    ITERS             IN
          MX0    60-4              MASK FOR IT
          BX6    -X0*X5            IT
          IX6    X4-X6             IN-IT
          LX6    7                 128*(IN-IT)
 TRAPRI1  SA1    NPLYC             NN
          MX2    5                 MASK FOR NT
          BX2    X2*X5
          LX2    5                 NT
          LX5    30                SHIFT TO ORIGINAL POSITION
          IX1    X1-X2             NN-NT
          IX6    X6-X1
          MX3    60-18
          BX5    X3*X5             42 BITS OF ENTRY WORD
          BX6    -X3*X6            18 BITS OF PRIORITY
          BX6    X5+X6             COMBINE THEM
          EQ     TRAPRI            ALL DONE
 JIGLET   TITLE  JIGLET - JIGGLE LET VALUES FOR VARIETY.
***       JIGLET - JIGGLE LET VALUES FOR VARIETY.
*
*         CALLED FROM EVALU8.
*
*         ENTRY  (X1) = MAXIMUM FRACTIONAL JIGGLE * 100B.
*                (RANDOM) = RANDOM 15 BIT VALUE.
*                EACH JIGGLE TABLE ENTRY = 42/PJ,18/PTR , WHERE:
*                PJ = -(PREVIOUS JIGGLE).  PTR POINTS TO THE LET.
*
*         EXIT   LETS POINTED TO IN JIGGLE TABLE ARE JIGGLED.
*                EACH PJ IS UPDATED TO -(CURRENT JIGGLE).
*
*         USES   ALL A AND X REGISTERS EXCEPT A0, X0.

          ENTRY  JIGLET

 JIGLET   EQ     *+400000B         ENTRY/EXIT
          SA2    RANDOM            PICK UP RANDOM SEED
          SA3    JIGGLE-1          FWA -1 OF JIGGLE TABLE
 JIGLET1  SA3    A3+B1             PICK UP NEXT ENTRY
          ZR     X3,JIGLET         RETURN IF DONE
          SA4    X3                THE PARAMETER TO BE JIGGLED
          AX3    18                MINUS PREVIOUS JIGGLE VALUE
          IX4    X4+X3             UN-JIGGLE
          LX2    60-15             GET SIGNED RANDOM VALUE
          AX2    60-15             SIGNED VALUE WITH 14 SIG BITS
          IX5    X2*X1
          IX5    X5*X4             RAN * SIZE * PARAM
          AX5    14+6              MINUS ABSOLUTE JIGGLE VALUE
          IX6    X4-X5             JIGGLE THE PARAMETER
          SA6    A4                STORE JIGGLED PARAMETER
          SX7    A4
          LX5    18
          MX6    42
          BX5    X6*X5
          BX6    X7+X5             42/-NEW JIGGLE,18/POINTER
          SA6    A3                STORE UPDATED JIGGLE ENTRY
          IX2    X2*X2             GET NEW RANDOM VALUE FROM SEED
          AX2    2
          SX2    X2+65432B
          MX6    60-15
          BX2    -X6*X2            NEW 15 BIT SEED
          EQ     JIGLET1           LOOP FOR NEXT ENTRY
 EVNPCK   SPACE  4,10
***       EVNPCK - PACK NBOARD FOR MASSIO LIBRARY SEARCHING.
*
*         CALLED FROM EVALU8.
*
*         ENTRY  (A0) = FWA OF AREA IN WHICH TO PACK.
*
*         CALLS  MSPACK.
*
*         USES   ALL REGISTERS EXCEPT A0.

          ENTRY  EVNPCK

 EVNPCK   EQ     *+400000B         ENTRY/EXIT
          SA1    CSTAT
          SA3    ENPAS
          SA5    CNT50
          SA2    OSIDE
          MX7    1                 SET WHITE TO MOVE
          ZR     X2,EVNPCK1        IF WHITE TO MOVE
          LX5    30
          LX7    30                SET BLACK TO MOVE
```

```
 EVNPCK1  SA2    A1+B1
          SA4    A3+B1
          LX3    48-10
          LX1    39-30
          LX4    18-40
          LX2    9-20
          BX6    X1+X2
          BX6    X6+X3
          BX6    X6+X4
          BX6    X6+X5
          BX3    X6+X7            SET SIDE TO MOVE IN STATUS
          SB3    NBOARD
          RJ     =XMSPACK         PACK NBOARD AT (A0)
          EQ     EVNPCK           RETURN

          END
EVCHEK
*FILE LINP
          IDENT  EVCHEK
          EXT    RETURN
          INIT
          TITLE  EVCHEK - GET OUT OF CHECK.
*CALL IOCOM
*CALL     LET
*CALL     TLET
*CALL SQRLST
*CALL BOARD
*CALL CONST
*CALL STACKS
*CALL DEBUG

 EEVCHEK  ECERTB EEVPARS

**        NODE PRIVATE TEMPORARY EQUIVALENCES.

 KSQN     EQU    NDTMP+1           SOME-ONES KING SQUARE
 CHKPCS   EQU    NDTMP+2           CHECKING PIECES (2 WORDS)
 CHKRAY   EQU    NDTMP+4           CHECK INTERPOSITION SQUARES (2 WORDS)
 EVCHEK   SPACE  4,10
**        EVCHEK - GET OUT OF CHECK
*
*         ENTERED DIRECTLY FROM EVMINI, EVFULL.
*
*         EXITS  BACK TO PARENT NODE.

          ENTRY  MIN70

*         KING IS IN CHECK, LOCATE IT.

 MIN70    SQLOOP (IDXLOC,TPLOC,(XOR,SSIDE,KING*2))
          SETQ   KSQN,SQLN
          SQLEND

*         TRY TO CAPTURE CHECKING PIECE.

          SETQS  CHKPCS,(ANDS,(INDEXS,ATKTO,(LSHIFT,KSQN,1)),(INDEXS,ALL
,OC,(NOT,MSIDE)))
          GENMOV TSLIST,(LOCF,CHKPCS)

 MIN71    SETMAX
          SELOOP DIRECT=REVERSE
          COND   (TRUE,(TEST,SMOVE,S.MINI)),MIN73
          SETX   SMTO,(AND,SMOVE,M.MTO)
          SETX   SMFR,(LSHIFT,(AND,SMOVE,M.MFR),-L.MTO)
          RELEAS SMOVE
          COND   (MEMBS,(MINUS,(LOCF,ALATK),MSIDE),SMTO),MIN72
          SELMAX (INDEX,VALUEX,(ABSF,(INDEX,NBOARD,SMTO)))
          GOTO   MIN73

 MIN72    SELMAX (MINUS,(INDEX,VALUEX,(ABSF,(INDEX,NBOARD,SMTO))),(INDEX
,,VALUEX,(ABSF,(INDEX,NBOARD,SMFR))))
          RELEAS (SMTO,SMFR)
 MIN73    SELEND
          COND   (EQ,SMAX,-INFIN),MIN75
          RELEAS SMAX
 MIN74    SEARCH MODE=MINI,REFUTE=RETURN
          GOTO   MIN71

*         TRY KING MOVES TO UNATTACKED SQUARES.

 MIN75    SETQ   LINDX,(LOCF,MOVES)  CLEAR OUT MOVES ARRAY
          GENMOV FSLIST,(IDXLOC,TPLOC,(XOR,SSIDE,KING*2))
          SETQ   INDEX,(LOCF,MOVES-LE.MOV)
 MIN76    SELOOP SELECT=MIN78,START=(PLUS,INDEX,LE.MOV)
          COND   (MEMBS,(IDXLOC,ALATK,(NOT,MSIDE)),(AND,SMOVE,M.MTO)),MI
,N77
          SELMOV
 MIN77    SELEND
          GOTO   MIN79

 MIN78    RJ     =XEVPARS          CHECK AGAINST PAR
          PL     X6,MIN76          IF NOT WORTH SEARCHING
 MI78A    SEARCH MODE=MINI,REFUTE=RETURN
          GOTO   MIN76

*         TRY INTERPOSITIONS.

 MIN79    SETQ   LINDX,(LOCF,MOVES)    CLEAR MOVES ARRAY
          SQLOOP (LOCF,CHKPCS)
          SETX   CHPCSQ,SQLN
          SQLEND
          SETQS  CHKRAY,(RAYSQS,KSQN,(LODREL,CHPCSQ))
          GENMOV TSLIST,(LOCF,CHKRAY)
          SETQ   INDEX,(LOCF,MOVES-LE.MOV)
 MIN80    SELOOP SELECT=MIN81,START=(PLUS,INDEX,LE.MOV)
          SELMOV
          SELEND
          GOTO   MIN82

 MIN81    RJ     =XEVPARS
          PL     X6,MIN80          IF NOT WORTH SEARCHING
 MI81A    SEARCH MODE=MINI,REFUTE=RETURN

          GOTO   MIN80

*         TRY ENPASSANT CAPTURE.

 MIN82    COND   (NULLS,ENPAS),MIN85  IF NO ENPASSANT SQUARES
          SETQS  =XGENFPNA,(INDEXS,TPLOC,MSIDE)
          SETQ   LINDX,(LOCF,MOVES)  CLEAR MOVES ARRAY
          GENMOV FPN,(LOCF,ENPAS)
          SETQ   INDEX,(LOCF,MOVES-LE.MOV)
 MIN83    SELOOP SELECT=MIN84,START=(PLUS,INDEX,LE.MOV)
          SELMOV
          SELEND
          GOTO   MIN85

 MIN84    SEARCH MODE=MINI,REFUTE=RETURN
          GOTO   MIN83

*         ALL MOVES SEARCHED, CHECK FOR CHECKMATE.

 MIN85    COND   (NZ,(INDEX,NSRCH,NPLYC)),RETURN

*         CHECKMATE.

          SETQ   VMOVE,(ORF,VMOVE,(LSHIFT,1,S.MAT))
          SCORE  (XOR,SSIDE,(PLUS,-INFIN,(LSHIFT,NPLYD,10)))

          END
*LEVEL *
EVPARS
*FILE LINP
          IDENT  EVPARS
          TITLE - EVPARS - CHECK WHETHER MOVE MIGHT MAKE PAR VALUE.
          INIT                     INIT MACRO SYSTEM
*CALL IOCOM
*CALL     LET
*CALL     TLET
*CALL SQRLST
*CALL BOARD
*CALL CONST
*CALL STACKS
*CALL DEBUG

 EEVPARS  ECERTB EEVTRAP
 EVPARS   SPACE  4,10
**        EVPARS - CHECK WHETHER MOVE MIGHT MAKE PAR VALUE.
*
*         CALLED FROM EVFULL, EVCHEK.
*
*         ENTRY  (INDEX) = POINTER TO MOVE.
*
*         EXIT   (X6) .LT. 0  IFF MOVE IS CAPTURE OR CHECK OR PROMOTION
*                OR CASTLE, OR IF IT IS FIRST TO BE SEARCHED FROM THIS
*                NODE OR IS POSSIBLY A REPETITION OR IF MATERIAL SCORE
*                IS TOO LOW GIVEN THE BOUNDS ON THE POSITIONAL SCORE.
*
*         CALLS  CHKCHK.

 M.ALP    EQU    M.CAP+M.PRO+M.CAS+M.CHK

          ENTRY  EVPARS

 EVPARS1  MX6    1                MUST SEARCH MOVE
 EVPARS   EQ     *+400000B        ENTRY/EXIT

          COND   (ZR,(INDEX,NSRCH,NPLYC)),EVPARS1
          SETX   MAXVAL,(PLUS,(INDEX,POSMIN,(MINUS,1,OSIDE)),MBSCR)
          MINMAX NEWVAL=MAXVAL,NON=PERM,IGNORE=EVPARS2
          RELEAS MAXVAL
          GOTO   EVPARS1

*         AVOID SEARCHING MOVES THAT COULDNT MEET PAR.

 EVPARS2  SETX   SMOVE,(INDEX,B0,INDEX)
          COND   (NZ,(AND,(LSHIFT,SMOVE,60-12),M.ALP/10000B)),EVPARS1
          COND   (ZR,(LSHIFT,CNT50,-31)),EVPARS3
          COND   (NE,(ABSF,(INDEX,NBOARD,(LSHIFT,(AND,SMOVE,M.MFR),-L.MT
,O))),PAWN),EVPARS1
          RELEAS SMOVE
 EVPARS3  CALLE  CHKCHK            CHECK FOR POSSIBLE CHECK
          GOTO   EVPARS            (X6) .GE. 0 IF NOT CHECK

          END
*LEVEL *
EVTRAP
*FILE LINP
          IDENT  EVTRAP
          TITLE - EVTRAP - CHECK FOR REPETITION OR TRANSPOSITION.
          EXT    REPD,RETURN
          INIT                     INIT MACRO SYSTEM
*CALL IOCOM
*CALL     LET
*CALL     TLET
*CALL SQRLST
*CALL BOARD
*CALL CONST
*CALL STACKS
*CALL DEBUG

 EEVTRAP  ECERTB EEVREFM
 EVTRAP   SPACE  4,10
**        EVTRAP - CHECK FOR REPETITION OR TRANSPOSITION.
*
*         CALLED FROM EVMINI, EVFULL.
*
*         EXITS  TO CALLER OR REPD OR RETURN OR PARENT NODE.
*
*         CALLS  REPCHK, TRACKT.

          ENTRY  EVTRAP
 EVTRAP   EQ     *+400000B        ENTRY/EXIT

*         CHECK FOR POSITION REPETITION.

          CALLE  REPCHK
          PL     X6,REPD           IF REPETITION

*         LOOK UP TRANSPOSITIONS HASH TABLE.

          CALLE  TRACKT            LOOK UP POSITION
          ZR     X6,EVTRAP          IF NO MATCH
          SETX   TRAWD,TRAWD       ELSE CHECK FOR SAME ITER AND PLY
          COND   (NE,(CAND,(MASK,60-4),(LSHIFT,TRAWD,30)),ITERS),EVTRAP
          COND   (NE,(CAND,(MASK,60-5),(LSHIFT,TRAWD,35)),NPLYC),EVTRAP
          SETX   TVFLAG,(CAND,(MASK,60-2),(LSHIFT,TRAWD,26))
          SETX   TVALUE,(LSHIFT,(LSHIFT,(LODREL,TRAWD),12),-48)
          GOTO   (EVTRAP,FUL0A,FUL0B,FUL0D),(LODREL,TVFLAG)

*         TVFLAG = 1.  -INFIN.LT.TRUE VALUE.LE.TVALUE.

 FUL0A    COND   (LE,TVALUE,ALPHA),FUL0D  ACCEPT VALUE
          COND   (GE,TVALUE,BETAR),EVTRAP  REJECT
          SETQ   BETAR,TVALUE      ELSE ACCEPT AS UPPER BOUND
          GOTO   FUL0C             CHECK FOR REFUTATION

*         TVFLAG = 2.  TVALUE.LE.TRUE VALUE.LT.INFIN.

 FUL0B    COND   (GE,TVALUE,BETAR),FUL0D  ACCEPT
          COND   (LE,TVALUE,ALPHA),EVTRAP  REJECT
          SETQ   ALPHA,TVALUE      ELSE ACCEPT AS LOWER BOUND
 FUL0C    COND   (GE,ALPHA,BETAR),RETURN  IF REFUTE
          GOTO   EVTRAP             ELSE CONTINUE
```

```
*         TVFLAG = 3.  TVALUE.EQ.TRUE VALUE.

 FUL0D    SCORE  (LODREL,TVALUE)   USE IT

          END
*LEVEL *
EVREFM
*FILE LINP
          IDENT  EVREFM
          TITLE  EVREFM - GENERATE AND SELECT A *REFMOV*.
          INIT
*CALL IOCOM
*CALL LET
*CALL SQRLST
*CALL BOARD
*CALL CONST
*CALL STACKS
*CALL DEBUG

 EEVREFM  ECERTB EEVMOPS

*         NODE PRIVATE TEMPORARY EQUIVALENCES.

 REFWD    EQU    NDTMP             REFUTATION WORD
 MVMSC    EQU    NDTMP+1           MASK FOR MOVE COMPARISON
 EVREFM   SPACE  4,10
**        EVREFM - GENERATE AND SELECT A PARTICULAR *REFUTATION* MOVE.
*
*         CALLED FROM EVFULL.
*
*         ENTRY  (NDTMP) = REFUTATION MOVE WORD.
*                (NDTMP+1) = MASK TO USE IN MOVE COMPARISON.
*
*         EXIT   (X6) .GE. 0  IFF MOVE IS POSSIBLE AND NOT YET SEARCHED.
*                (INDEX) = POINTER TO MOVE IF (X6) .GE. 0.
*
*         CALLS  GENFSL.

          ENTRY  EVREFM

 EVREFM1  BX6    X6-X              EXIT WITH MOVE FOUND
 EVREFM   EQ     *+400000B         ENTRY/EXIT
          SETX   REFWD,REFWD
          COND   (ZR,REFWD),EVREFM4  IF NO MOVE
          SETX   REFMAN,(INDEX,NBOARD,(LSHIFT,(AND,REFWD,M.MFR),-L.MTO))
          RELEAS REFWD
          COND   (ZR,REFMAN),EVREFM4  NO MAN ON SQUARE
          COND   (NG,(XOR,MSIDE,REFMAN)),EVREFM4  WRONG SIDE
          GENMOV FSLIST,(IDXLOC,TPLOC,(LSHIFT,(LODREL,REFMAN),1))
          SETX   REFMOV,REFWD
          SELOOP SELECT=EVREFM1
          COND   (NZ,(AND,(XOR,REFMOV,SMOVE),MVMSC)),EVREFM3
          COND   (TRUE,(TEST,SMOVE,S.SRC)),EVREFM3
          COND   (FALSE,(TEST,SMOVE,S.PRO)),EVREFM2  IF NOT PROMOTION
          SETX   PQFLAG,M.ENP+M.CAS  USE ONLY QUEEN PROMOTION
          COND   (NE,(AND,SMOVE,PQFLAG),PQFLAG),EVREFM3
          RELEAS PQFLAG
 EVREFM2  SELMOV
 EVREFM3  SELEND
          RELEAS REFMOV
 EVREFM4  MX6    1                 WE DIDNT FIND THE MOVE
          EQ     EVREFM            RETURN

          END
*LEVEL *
EVMOPS
*FILE LINP
          IDENT  EVMOPS
          TITLE  EVMOPS - MOPUP EVALUATION FUNCTION.
          EXT    DRAW
          INIT                     INIT MACRO SYSTEM
*CALL IOCOM
*CALL     LET
*CALL     TLET
*CALL SQRLST
*CALL BOARD
*CALL CONST
*CALL STACKS
*CALL DEBUG

 EEVMOPS  ECERTB EEVPOSS

*         NODE PRIVATE TEMPORARY EQUIVALENCES.

 KSIDE    EQU    NDTMP             SIDE TO BE MATED = + OR - ZERO
 KCORN    EQU    NDTMP+1           KING CORNERING SCORE
 POSISH   EQU    NDTMP+2           POSITIONAL SCORE
 NSPTK    EQU    NDTMP+3           KING AND KNIGHT MATING TROPISM
 NSLOC    EQU    NDTMP+4           (2 WORDS) MATING K AND N SQUARES
 PCOL     EQU    NDTMP+6           PAWN COLUMN LOOP COUNTER
 PSIDE    EQU    NDTMP+7           SIDE WITH PASSED PAWN
          SPACE  4,10
**        EVMOPS - CHECKMATING ALGORITHM FOR ENDGAMES.
*
*         CALLED FROM EVMINI.
*
*         EXIT   (POSISH) = MOPUP-SCORE * 100B
*
*         SPECIAL EXIT TO DRAW IF STALEMATE.
*
*         PRINCIPLES.  1. DRIVE KING TO CORNER.
*                      2. APPROACH KING WITH KING AND KNIGHTS.
*                      3. DONT STALEMATE HIM.
*                      4. POSITION WINNING KING 2 SQUARES FROM EDGE.

          ENTRY  EVMOPS
 EVMOPS   EQ     *+400000B         ENTRY/EXIT
          SETQ   KSIDE,(NOT,(LSHIFT,MBSCR,-59))
 MINI5    SQLOOP (IDXLOC,TPLOC,(XOR,KSIDE,KING*2))
          SETX   KSQN,SQLN
          SQLEND

*         GUARD AGAINST STALEMATE.

          COND   (NG,(XOR,KSIDE,SSIDE)),MINI7  IF WINNER ON MOVE
          COND   (TRUE,(TEST,VMOVE,S.CHK)),MINI7  IF LOSER IN CHECK
          SETX   MBVAL,MBVAL
          COND   (PL,MBVAL),MINI6  IF BLACK GETTING MATED
          SETX   MBVAL,(LSHIFT,MBVAL,30)  ADJUST LOSER LOWER
 MINI6    COND   (NZ,(ANDC,(LODREL,MBVAL),(MASK,60-4*5))),MINI7
          COND   (NULLS,(CANDS,(INDEXS,ALATK,(XOR,(NOT,KSIDE),2)),(INDEX
,S,ATKFR,(LSHIFT,KSQN,1)))),DRAW  IF NO LEGAL MOVES, IS STALEMATE

*         DRIVE KING TO CORNER.

 MINI7    SETQ   KCORN,(CENDST,KSQN)

*         APPROACH KING WITH WINNING KING AND KNIGHTS.

          SETQ   NSPTK,0
          SETQS  NSLOC,(ORS,(INDEXS,TPLOC,(XOR,KSIDE,-KING*2)),(INDEXS,T
,PLOC,(XOR,KSIDE,-NITE*2)))
          SQLOOP (LOCF,NSLOC)
          SETQ   NSPTK,(PLUS,(HSQDST,SQLN,KSQN),(VSQDST,SQLN,KSQN),-14,N
,SPTK)
          SQLEND
          RELEAS KSQN

*         POSITION WINNING KING 2 SQUARES FROM EDGE.

          SETX   KOPP,(MASK,60)
          SQLOOP (IDXLOC,TPLOC,(XOR,KSIDE,-KING*2))
          SETX   ROW,(LSHIFT,SQLN,-3)
          COND   (EQ,ROW,2),MINI8
          COND   (EQ,ROW,5),MINI8
          RELEAS ROW
          SETX   COLUMN,(AND,SQLN,7)
          COND   (EQ,COLUMN,2),MINI8
          COND   (EQ,COLUMN,5),MINI8
          RELEAS COLUMN
          SETX   KOPP,0
 MINI8    SQLEND

*         ADD ALL FACTORS TOGETHER AND SCALE RESULT.

          SETX   CHMTSC,(MINUS,(PLUS,(AND,(LODREL,KOPP),FCHKOP),(TIMES,F
,KCORN,KCORN)),(TIMES,FKKNMT,NSPTK))
          SETQ   POSISH,(XOR,(NOT,KSIDE),(LODREL,CHMTSC))

*         AWARD  FOR PASSED PAWNS.

          SETQ   PCOL,0
 MINI9    SETX   PSWORD,(INDEX,PSQRF,PCOL)
          SETQ   PSIDE,0
 MIN10    COND   (FALSE,(TEST,PSWORD,57)),MIN11  IF PAWN NOT PASSED
          SETX   PPSTP2,(PLUS,(PWNADV,(ANDC,(LSHIFT,PSWORD,24),(MASK,54)
,)),1)
          ACCSID POSISH,(TIMES,PPSTP2,PPSTP2,FPNPAS),PSIDE
 MIN11    COND   (NG,PSIDE),MIN12   IF DONE WITH BOTH SIDES
          SETQ   PSIDE,(NOT,0)  ELSE FLIP SIDE
          SETX   PSWORD,(LSHIFT,PSWORD,30)
          GOTO   MIN10

 MIN12    SETQ   PCOL,(PLUS,PCOL,1)
          COND   (LT,PCOL,8),MINI9   LOOP FOR NEXT COLUMN
          RELEAS (PPSTP2,PSWORD)

          EQ     EVMOPS            RETURN

          END
*LEVEL *
EVPOSS
*FILE LINP
          IDENT  EVPOSS
          TITLE  EVPOSS - REGULAR POSITIONAL EVALUATION.
          INIT                     INIT MACRO SYSTEM
*CALL IOCOM
*CALL     LET
*CALL     TLET
*CALL SQRLST
*CALL BOARD
*CALL CONST
*CALL STACKS
*CALL DEBUG

 EEVPOSS  ECERTB EMYMOVE
 CONST    SPACE  4,10
**        CONSTANT DATA.

 PTCMSK   DATA   0000007700000000007700B MASK FOR TARGET PAWN COLUMNS
 PSCMSK   DATA   0000770077000000000B MASK FOR W PAWN SIDE COLUMNS
 KSANQ    VFD    20/0,10/606B,10/606B,20/0  ALL R1,R2,N1,N2
          VFD    20/0,10/606B,10/606B,20/0

**        NODE PRIVATE TEMPORARY EQUIVALENCES.

 POSISH   EQU    NDTMP+2           POSITIONAL SCORE
 PCOL     EQU    NDTMP+6           PAWN COLUMN LOOP COUNTER

 KSAFE    EQU    NDTMP+3           KING SAFETY FACTOR
 KSIMP    EQU    NDTMP+4           KING SAFETY IMPORTANCE
 KARSQS   EQU    NDTMP+5           KING SECTOR SQUARES (2 WORDS)

 LSIDE    EQU    NDTMP             SIDE IN MIDGAME SCORE LOOP
 KSQN     EQU    NDTMP+1           SOME-ONES KING SQUARE
 CTROP    EQU    NDTMP+3           ACCUMULATE CENTER TROPISM
 KTROP    EQU    NDTMP+4           ACCUMULATE KING TROPISM
 MOBIL    EQU    NDTMP+5           ACCUMULATE MOBILITY
 RLOC     EQU    NDTMP+6           ALL ROOK LOCATIONS (2 WORDS)

 APLOC    EQU    NDTMP+6           ALL PAWN LOCATIONS (BOTH SIDES)

 NSQ      EQU    NDTMP+3           NUMBER OF ATTACKED PIECES
 NBLAST   EQU    NDTMP+4           NUMBER OF HUNG PIECES
 ATKNPS   EQU    NDTMP+5           ATTACKED PIECES (2 WORDS)
 ATKERS   EQU    NDTMP+5           ATTACKING PIECES (2 WORDS)
 BLAST    EQU    NDTMP+7           VALUE OF ATTACKED PIECE
 EVPOSS   SPACE  4,10
**        EVPOSS - REGULAR POSITIONAL EVALUATION.
*
*         CALLED FROM EVMINI.
*
*         EXIT   (POSISH) = SCORE * 100B

          ENTRY  EVPOSS
 EVPOSS   EQ     *+400000B         ENTRY/EXIT
          SETQ   POSISH,0          INIT POSITIONAL SCORE

*         LOOP THRU BOTH SIDES FOR SYMMETRIC FACTORS.

          SETQ   LSIDE,0           INITIALIZE SIDE LOOP

*         LOCATE OPPONENTS KING.

 MIN16    SETQ   KSQN,(INDEX,WKSQR,(ORF,LSIDE,1))

*         PRIMITIVE SAFETY EVALUATION OF OPPONENT KING.

          SETQ   KSAFE,0           INITIALIZE

*         GET IMPORTANCE OF KING SAFETY FROM COUNTS OF OUR R,B,N S
*         PLUS EXTRA FOR QUEENS.
```

```
         SETX   KSIMP,(COUNTS,(ANDCS,(INDEXS,ALLOC,(XOR,LSIDE,2)),(INDE
,XS,TPLOC,(XOR,LSIDE,PAWN*2))))
         COND   (NULLS,(INDEXS,TPLOC,(XOR,LSIDE,QUEN*2))),MIN17
         SETX   KSIMP,(PLUS,KSIMP,FKSQBN)
 MIN17   SETX   KSIMP,(MAXF,0,(MINUS,(LODREL,KSIMP),2))
         COND   (ZR,KSIMP),MIN25  NOT ENOUGH PIECES TO JUSTIFY
         SETQ   KSIMP,(LODREL,KSIMP)

*        GET SQRLST OF SQUARES ATTACKED BY KING.

         SETQS  KARSQS,(INDEXS,ATKFR,(LSHIFT,KSQN,1))

*        PUNISH KING NOT ON R1,N1,R2, OR N2 (SANCTUARY).

         COND   (MEMBS,(LOCF,KSANQ),KSQN),MIN18
         COND   (NZ,(INDEX,CSTAT,(ORF,LSIDE,1))),MI17A
         SETQ   KSAFE,FKSANQ
         GOTO   MI19A

*        DONT PUNISH AS MUCH IF CAN STILL CASTLE.

 MI17A   SETQ   KSAFE,FKSCNQ
         GOTO   MI19A

*        PUNISH KING IN SANCTUARY IF TOO CRAMPED.

 MIN18   COND   (NONMS,(LOCF,EIGHT),KSQN),MIN19
         COND   (NNULS,(CANDS,(ORS,(INDEXS,TPLOC,(XOR,LSIDE,-PAWN*2)),(
,INDEXS,ALATK,(XOR,LSIDE,2)),EIGHT),KARSQS)),MIN19
         SETQ   KSAFE,(PLUS,KSAFE,FKSCRM)

*        PUNISH IF 2 BARE SQUARES IN FRONT OF CASTLED KING.

 MIN19   SETX   LSIDE,LSIDE
         COND   (MEMBS,(IDXLOC,ALLOC,(XORC,LSIDE,2)),(ANDC,(PLUS,KSQN,(
,XORC,LSIDE,10B)),(MASK,54))),MI19A  SQ IN ROW IN FRONT OF KING
         COND   (MEMBS,(IDXLOC,ALLOC,(XORC,LSIDE,2)),(ANDC,(PLUS,KSQN,(
,XORC,LSIDE,20B)),(MASK,54))),MI19A  SQ 2 ROWS AWAY FROM KING
         RELEAS LSIDE
         SETQ   KSAFE,(PLUS,KSAFE,FKS2BS)
 MI19A   BSS    0

*        PENALIZE IF KING HAS NOT AT LEAST N FRIENDLY PIECES NEXT TO K.

         SETX   KFREND,(COUNTS,(ANDS,KARSQS,(INDEXS,ALLOC,(XOR,LSIDE,-2
,))))
         SETX   FKSMNF,FKSMNF
         COND   (GE,KFREND,FKSMNF),MIN20
         SETQ   KSAFE,(PLUS,KSAFE,(TIMES,(MINUS,FKSMNF,(LODREL,KFREND))
,,FKSFRD))
         RELEAS FKSMNF

*        PENALIZE FOR SQUARES ATTACKED BY OUR PIECES.

 MIN20   SETX   KSNATK,(COUNTS,(ANDS,KARSQS,(INDEXS,ALATK,(XOR,LSIDE,2)
,)))
         SETX   KSNATK,(MINUS,(LSHIFT,KSNATK,2),FKSMAT)
         COND   (LE,KSNATK,0),MIN21   IF NOT ENOUGH ATTACKS
         SETQ   KSAFE,(PLUS,(TIMES,FKSATK,(LODREL,KSNATK)),KSAFE)
 MIN21   BSS    0

*        PENALIZE KING ON FILE WITHOUT HIS PAWN,OR IF PAWN IS ISOLATED.

         SETX   PSWORD,(INDEX,PSQRF,(ANDC,KSQN,(MASK,60-3)))
         COND   (PL,LSIDE),MIN22  IF WHITE TO MOVE
         SETX   PSWORD,(LSHIFT,PSWORD,30)
 MIN22   COND   (NZ,(AND,PSWORD,7700B)),MIN23
         COND   (NZ,(INDEX,CSTAT,(ORF,LSIDE,1))),MI22A  IF CAN CASTLE
         SETQ   KSAFE,(PLUS,KSAFE,FKSOPF)
         GOTO   MIN23

*        DONT PENALIZE SO MUCH FOR NO PAWN IF CAN CASTLE.

 MI22A   SETQ   KSAFE,(PLUS,KSAFE,FKSCOF)
 MIN23   COND   (NZ,(AND,PSWORD,(LSHIFT,PSCMSK,30))),MIN24
         RELEAS PSWORD
         SETQ   KSAFE,(PLUS,KSAFE,FKSISO)
 MIN24   ACCSID POSISH,(TIMES,KSAFE,KSIMP),LSIDE
 MIN25   BSS    0

*        IF LOW MATERIAL, MAKE OPPONENT KING ATTRACT CENTER.

         SETX   MBVAL,MBVAL
         COND   (PL,LSIDE),MIN26
         SETX   MBVAL,(LSHIFT,MBVAL,30)
 MIN26   SETX   CORSMAT,(ANDC,(LSHIFT,(LODREL,MBVAL),-S.CWM),(MASK,60-L
,.CWM))
         COND   (GT,(LODREL,CORSMAT),1),MIN27   IF MORE THAN 1700B WOOD
         ACCSID POSISH,(TIMES,(CENDST,KSQN),FKCTRP),LSIDE

*        ALSO ATTRACT IT TO PAWNS ON BOARD.

         SETQS APLOC,(ORS,(INDEXS,TPLOC,PAWN*2),(INDEXS,TPLOC,-PAWN*2))
         COND   (NULLS,APLOC),MIN27    IF NO PAWNS
         SETX   KSQN,KSQN
         SETX   KPD,0
         SQLOOP (LOCF,APLOC)
         SETX   KPD,(PLUS,KPD,(HSQDST,KSQN,SQLN))
         SETX   KPD,(PLUS,KPD,(VSQDST,KSQN,SQLN))
         SQLEND
         ACCSID POSISH,(TIMES,(MINUS,(DIVIDE,KPD,(COUNTS,APLOC)),6),FKP
,TRP),LSIDE
         RELEAS (KSQN,KPD)
 MIN27   BSS    0

*        EVALUATE KNIGHTS FOR KING AND CENTER TROPISM.
*        ALSO PENALIZE FOR BEING ON BACK RANK.
*        ALSO PENALIZE IF ATTACKED, ESPECIALLY IF UNDEFENDED BY PAWNS.

         SETQ   CTROP,0
         SETQ   KTROP,0
         SQLOOP (IDXLOC,TPLOC,(XOR,LSIDE,NITE*2))
         SETQ   CTROP,(MINUS,(PLUS,CTROP,6),(CENDST,SQLN))
         SETQ   KTROP,(PLUS,(NOT,(PLUS,(HSQDST,KSQN,SQLN),(VSQDST,KSQN,
,SQLN))),KTROP,5)
         COND   (NZ,(AND,(XOR,LSIDE,SQLN),70B)),MIN28
         ACCSID POSISH,(NOT,FNBKRK),LSIDE
 MIN28   COND   (NONMS,(IDXLOC,ALATK,(XOR,LSIDE,-2)),SQLN),MI28A
         ACCSID POSISH,(NOT,FNATKD),LSIDE  PENALIZE FOR ATTACKED KNIGHT
         COND   (NNULS,(ANDS,(INDEXS,ATKTO,(LSHIFT,(LODREL,SQLN),1)),(I
,NDEXS,TPLOC,(XOR,LSIDE,PAWN*2)))),MI28A  IF DEFENDED BY A PAWN
         ACCSID POSISH,(NOT,FNATNP),LSIDE
 MI28A   SQLEND
         ACCSID POSISH,(PLUS,(TIMES,FNCTRP,CTROP),(TIMES,FNKTRP,KTROP))
,,LSIDE

*        EVALUATE BISHOPS FOR MOBILITY.
*        ALSO PENALIZE FOR BEING ON BACK RANK.

         SETQ   MOBIL,0
         SQLOOP (IDXLOC,TPLOC,(XOR,LSIDE,BISH*2))
         COND   (NZ,(AND,(XOR,LSIDE,SQLN),70B)),MIN29
         ACCSID POSISH,(NOT,FBBKRK),LSIDE
 MIN29   BSS    0
         SETQ   MOBIL,(PLUS,(COUNTS,(ANDCS,(INDEXS,ATKFR,(LSHIFT,(LODRE
,L,SQLN),1)),(INDEXS,TPLOC,(XOR,LSIDE,PAWN*2)))),MOBIL,-7)
         SQLEND
         ACCSID POSISH,(TIMES,FBMOBL,MOBIL),LSIDE

*        EVALUATE ROOKS FOR MOBILITY, KING TROPISM,DOUBLING,OPEN AND
*        SEMI-OPEN FILES,7TH RANK.

         SETQ   MOBIL,0
         SETQ   KTROP,0
         SETQS  RLOC,(INDEXS,TPLOC,(XOR,LSIDE,ROOK*2))
         SQLOOP (LOCF,RLOC)
         SETQ   MOBIL,(PLUS,(COUNTS,(INDEXS,ATKFR,(LSHIFT,SQLN,1))),MOB
,IL)
         SETQ   KTROP,(PLUS,(NOT,(MINF,(HSQDST,KSQN,SQLN),(VSQDST,KSQN,
,SQLN))),KTROP)
         COND   (NE,(AND,(XOR,LSIDE,SQLN),70B),60B),MIN30
         ACCSID POSISH,FRK7TH,LSIDE
 MIN30   SETX   PSWORD,(INDEX,PSQRF,(AND,SQLN,7))
         COND   (NZ,(AND,PSWORD,PTCMSK)),MIN31
         ACCSID POSISH,FRKOPF,LSIDE
         GOTO   MIN33

 MIN31   COND   (PL,LSIDE),MIN32
         SETX   PSWORD,(LSHIFT,PSWORD,30)
 MIN32   COND   (NE,(AND,(LSHIFT,PSWORD,42),340B),40B),MIN33
         COND   (NZ,(AND,(MASK,6,42),(LODREL,PSWORD))),MIN33

*        AWARD BONUS FOR ROOK HITTING VULNERABLE PAWN.

         ACCSID POSISH,FRKVNP,LSIDE
 MIN33   COND   (NULLS,(ANDS,(INDEXS,ATKFR,(LSHIFT,(LODREL,SQLN),1)),RL
,OC)),MIN34
         ACCSID POSISH,FRKDBL,LSIDE
 MIN34   SQLEND
         ACCSID POSISH,(PLUS,(TIMES,FRKTRP,KTROP),(TIMES,FRMOBL,MOBIL))
,,LSIDE

*        EVALUATE QUEENS FOR MOBILITY, KING TROPISM.

         SQLOOP (IDXLOC,TPLOC,(XOR,LSIDE,QUEN*2))
         ACCSID POSISH,(TIMES,(PLUS,(HSQDST,KSQN,SQLN),(VSQDST,KSQN,SQL
,N)),FQKTRP),(NOT,LSIDE)
         SETX   QMOB,(COUNTS,(ANDCS,(INDEXS,ATKFR,(LSHIFT,(LODREL,SQLN)
,,1)),(INDEXS,ALATK,(XOR,(NOT,LSIDE),2))))
         ACCSID POSISH,(TIMES,(LODREL,QMOB),FQMOBL),LSIDE
         SQLEND

*        EVALUATE PAWN STRUCTURE FACTORS FOR DOUBLED,ISOLATED,
*        BACKWARD,PASSED,MAJORITY,ADVANCED CENTER,ETC.
*        ALSO PENALIZE FOR BLOCKED 2ND RANK CENTER PAWN.

         SETQ   PCOL,0            INIT LOOP THRU COLUMNS.
 MIN35   SETX   PSWORD,(INDEX,PSQRF,PCOL)
         COND   (PL,LSIDE),MIN36
         SETX   PSWORD,(LSHIFT,PSWORD,30)
 MIN36   COND   (ZR,(AND,(MASK,6,42),PSWORD)),MIN44    IF NO PAWN
         COND   (FALSE,(TEST,PSWORD,41)),MI36A  IF NOT 2ND RANK
         COND   (NZ,(LSHIFT,(MINUS,(LSHIFT,PCOL,1),7),-1)),MI36A
         COND   (ZR,(INDEX,NBOARD,(PLUS,34B,(XOR,LSIDE,-14B),PCOL))),!!
,MI36A   IF 3RD RANK SQUARE NOT BLOCKED
         ACCSID POSISH,FPNBSC,(NOT,LSIDE)  PENALIZE BLCKD 2ND RK CTR PN
 MI36A   BSS    0
         COND   (PL,PSWORD),MIN37   IF NOT DOUBLED
         ACCSID POSISH,FPNDBL,(NOT,LSIDE)
 MIN37   COND   (NZ,(AND,PSWORD,PSCMSK)),MIN38    IF NOT ISOLATED
         COND   (NZ,(AND,(MASK,6,12),PSWORD)),MI37A  IF ENEMY PAWN
         ACCSID POSISH,FPNISO,(NOT,LSIDE)
         GOTO   MIN38

 MI37A   ACCSID POSISH,FPNISC,(NOT,LSIDE)  SMALLER PENALTY
 MIN38   SETX   PDESC,(AND,(LSHIFT,PSWORD,12),370B)
         COND   (NE,PDESC,50B),MIN39
         ACCSID POSISH,FPNVBK,(NOT,LSIDE)
         GOTO   MIN40

 MIN39   COND   (NE,(LODREL,PDESC),60B),MIN40
         ACCSID POSISH,FPNLBK,(NOT,LSIDE)
 MIN40   SETX   PWNADV,(PWNADV,(ANDC,(LSHIFT,PSWORD,24),(MASK,54)))
         ACCSID POSISH,(TIMES,(INDEX,FPDCOL,PCOL),(MINUS,PWNADV,1)),LSI
,DE
         COND   (FALSE,(TEST,PSWORD,58)),MIN44    IF NOT MAJORITY
         ACCSID POSISH,(TIMES,(MINUS,PWNADV,1),FPNMAJ),LSIDE
         COND   (FALSE,(TEST,PSWORD,57)),MIN44    IF NOT PASSED
         RELEAS PSWORD
         SETX   PWNAD1,(PLUS,(LODREL,PWNADV),1)
         SETX   BLKSQ,(PLUS,(LSHIFT,(ANDC,(XOR,PWNAD1,LSIDE),(MASK,60-3
,)),3),PCOL)
         SETX   PASBON,FPNPAS
         COND   (NONMS,(IDXLOC,ALLOC,(XOR,LSIDE,-2)),BLKSQ),MIN41
         SETX   PASBON,(MINUS,PASBON,FPNPSB)
 MIN41   COND   (NONMS,(IDXLOC,ALATK,(XOR,LSIDE,-2)),BLKSQ),MIN42
         SETX   PASBON,(MINUS,PASBON,FPNPSA)
 MIN42   COND   (NONMS,(IDXLOC,ALATK,(XOR,LSIDE,2)),BLKSQ),MIN43
         RELEAS BLKSQ
         SETX   PASBON,(PLUS,PASBON,FPNPSD)
 MIN43   ACCSID POSISH,(TIMES,(LODREL,PASBON),PWNAD1,PWNAD1),LSIDE
         RELEAS (PWNAD1,PSWORD)
 MIN44   SETQ   PCOL,(PLUS,PCOL,1)
         COND   (LT,PCOL,8),MIN35

*        LOOP FOR BLACK.

         SETQ   LSIDE,(NOT,LSIDE)
         COND   (NG,LSIDE),MIN16

*        GIVE SIDE TO MOVE TEMPO BONUS.

         ACCSID POSISH,FTMBON,SSIDE

*        PENALIZE FOR IMPROPERLY DEFENDED PIECES (NOT PAWNS).

         SETQS  ATKNPS,(ANDS,(ANDCS,(INDEXS,ALLOC,MSIDE),(INDEXS,TPLOC,
,MSIDE)),(INDEXS,ALATK,(NOT,MSIDE)))
         SETX   NSQ,0
         SQLOOP (LOCF,ATKNPS)     FIND ATTACKED PIECES
         SETA   (IDXLOC,STACK1,NSQ),SQLN
         SETX   NSQ,(PLUS,NSQ,1)
         SQLEND
         COND   (ZR,NSQ),MIN48    IF NO ATTACKED PIECES
```

```
          SETQ   NSQ,(MINUS,(LODREL,NSQ),1)
          SETQ   NBLAST,0
 MIN45    SETX   SQN,(INDEX,STACK1,NSQ)
          SETQ   BLAST,(INDEX,VALUEX,(ABSF,(INDEX,NBOARD,SQN)))
          COND   (NONMS,(IDXLOC,ALATK,MSIDE),SQN),MIN46
          SETQS  ATKERS,(ANDS,(INDEXS,ATKTO,(LSHIFT,SQN,1)),(INDEXS,ALLO
,C,(NOT,MSIDE)))
          RELEAS SQN

*         FIND CHEAPEST ATTACKER.

          SETX   ATKR,INFIN
          SQLOOP (LOCF,ATKERS)
          SETX   ATKR,(MINF,(INDEX,VALUEX,(ABSF,(INDEX,NBOARD,SQLN))),AT
,KR)
          SQLEND
          COND   (GE,ATKR,BLAST),MIN47  IF ATTACKER .GE. ATTACKEE
          RELEAS ATKR
 MIN46    SETQ   NBLAST,(PLUS,NBLAST,1)
 MIN47    SETQ   NSQ,(MINUS,NSQ,1)  LOOP FOR NEXT ATTACKED PIECE
          COND   (PL,NSQ),MIN45
          COND   (ZR,NBLAST),MIN48  IF NO HUNG PIECES
          ACCSID POSISH,FRHUNG,(NOT,SSIDE)
          COND   (LE,NBLAST,1),MIN48  IF ONLY 1 HUNG PIECE
          ACCSID POSISH,FORKBN,(NOT,SSIDE)
          COND   (EQ,NPLYD,1),MIN48  IF DOING PRELS

*         PUNISH FORKED SIDE WITH MINIMUM POSITIONAL SCORE.

          COND   (ZR,OSIDE),MI47A  IF WHITE TO MOVE
          SETQ   POSISH,(MAXF,(LSHIFT,POSMAX,6),POSISH)
          GOTO   MIN48

 MI47A    SETQ   POSISH,(MINF,(LSHIFT,POSMIN,6),POSISH)

 MIN48    EQ     EVPOSS            RETURN

          END
*LEVEL *
MYMOVE
          IDENT  MYMOVE
 MYMOVE   TITLE  MYMOVE - MISCELLANEOUS CONTROL ROUTINES.
          SST
          LIST   F
*CALL     IOCOM
*CALL     SQRLST
*CALL     LET
*CALL TLET
*CALL     BOARD
*CALL DEBUG
 EMYMOVE  ECERTB
 MAKMOV   SPACE  4,88
***       MAKMOV - MAKE MOVE FOR OPPONENT.
*
*         ENTRY  (BSTMV) = OPPONENTS MOVE.
*
*         EXIT   TO LSTMOV, IF NO REPLY MODE.
*                TO MYMOVE, OTHERWISE.
*

          ENTRY  MAKMOV
 MAKMOV   RJ     =XTHEMOV          MAKE THE PLAYERS MOVE
          RJ     =XDUDMOVE         MAKE MOVE ON DISPLAYS
          SA1    SWICH
          LX1    59-S.NOR
          BX6    X6-X6
          PL     X1,MAKMOV1        IF NOT NO-REPLY MODE
          SA6    PONDR             ELSE CLEAR PREDICTED MOVE
          EQ     LSTMOV            LIST OPPONENTS MOVES

 MAKMOVA  CON    567777B           MASK TO COMPARE MOVES

 MAKMOV1  SA1    PONDR             PREDICTED OPPONENT MOVE
          SA2    BSTMV             ACTUAL OPPONENT MOVE
          SA3    MAKMOVA           MASK TO COMPARE MOVES
          BX2    X2-X1             COMPARE
          BX2    X3*X2
          ZR     X2,MYMOVE         IF ACTUAL AND PREDICTED ARE SAME
          SA6    PONDR             ELSE CLEAR PREDICTED MOVE
*         EQ     MYMOVE

 MYMOVE   SPACE  4,17
***       MYMOVE - COMPUTE MACHINES MOVE.
*
*         ENTRY  (BOARD) = CURRENT POSITION.
*
*         EXIT   TO RECRDM, IF LEGAL REPLY.
*                TO FINISH, OTHERWISE.
*
          ENTRY  MYMOVE
 MYMOVE   RJ     =XCREATE
          SA1    PONDR             CHECK WHETHER COMPUTED MOVE
          LX1    59-S.THF          ALREADY BY PONDERING.
          NG     X1,MYMOVE5        IF SO
          MX6    59                FLAG NO (-1 MSEC) PONDER FREE TIME
          SA6    TIMPO
          BX6    X6-X6
          SA6    WIERD             CLEAR WIERD MESSAGE WORD
          ATIME  MYMVTM            START TIME IN MILLISECONDS
          SX6    M.ACS+M.PRO       SET NULL MOVE
          R=     X7,BASE           SET BASE SEARCH MODE
          LX7    S.MOD
          BX6    X7+X6
          SA1    STATUS
          SA2    MYMOVEC
          BX1    X2*X1
          ZR     X1,MYMOVE1        IF NOT IN CHECK
          SX7    B1                ELSE SAY WE ARE IN CHECK
          LX7    S.CHK
          BX6    X6+X7
 MYMOVE1  BSS    0
          SA6    VMOVE
          SA1    SWICH
          LX1    59-S.RAN
          PL     X1,MYMOVE2        IF RANDOM SWITCH OFF
          RTIME  RANDOM            ELSE GET NEW SEED
 MYMOVE2  SA1    RANDOM
          MX6    60-15
          BX6    -X6*X1            EXTRACT LOW 15 BITS OF MSEC CLOCK
          SA6    A1
          RJ     =XEVALU8
          RJ     =XTIMUSD          GET THE TIME USED
          SA6    TIMMV             TIME TO COMPUTE MOVE
          RJ     =XXLATPV          TRANSLATE PRINCIPAL VARIATION
          SA6    PONDR             SAVE PREDICTED OPPONENT RESPONSE
          SA7    PKILL             AND MACHINES PREDICTED REJOINDER
 MYMOVE3  SA3    TIMTO             ADVANCE TOTAL CP TIME USED
          SA2    TIMMV
          IX6    X2+X3
          SA6    A3
          SA1    SCORE             SCORE FROM THE MOVE
          BX6    X1
          SA6    SAVES             SAVE IT
          SA1    CNTMV             COUNT MOVES COMPUTED
          SX6    X1+B1
          SA6    A1
          RJ     =XTRTRFL          TREE TRACE FLUSH
          SA1    BSTMV
          ZR     X1,MYMOVE4        IF NO MOVE WAS FOUND
          JUMPUP RECRDM            RECORD MOVE

 MYMOVE4  JUMPUP FINISH

*         MOVE WAS ALREADY COMPUTED WHILE PONDERING.

 MYMOVE5  SA1    PBSMV             SAVED MOVE
          SA2    PSCOR             SAVED SCORE
          BX6    X1
          BX7    X2
          SA6    BSTMV             SET MACHINE MOVE
          SA7    SCORE             SET MACHINE SCORE
          SA1    PPNDR             SAVED PREDICTED OPPONENT RESPONSE
          MX6    42
          BX6    -X6*X1
          SA6    PONDR
          SA1    PVMOV             RESTORE VMOVE WORD WITH FLAGS
          SA2    PWIRD             RESTORE WIERD WORD
          BX7    X2
          SA7    WIERD
          BX6    X1
          SA6    VMOVE
          SA1    PTMMV             SAVED TIME OF MOVE
          BX6    X1
          SA6    TIMMV
          RJ     =XLSTLGL
          EQ     MYMOVE3

 MYMOVEB  CON    4000000000040000000B
 MYMOVEC  DATA   1000000000010000000B MASK FOR CHECK STATUS
 MACMOV   SPACE  4,19
***       MACMOV - MAKE MOVE FOR MACHINE.
*
*         ENTRY  (BSTMV) = MACHINES MOVE.
*                (GOCNT) = NUMBER OF MOVES YET TO GO.
*
*         EXIT   TO MYMOVE, IF GOCNT ' 0, DECREMENT GOCNT.
*                TO LSTMOV, OTHERWISE.
*

          ENTRY  MACMOV
 MACMOV   RJ     =XTHEMOV
          RJ     =XDUDMOVE         MAKE MOVE ON DISPLAYS
          SA1    GOCNT
          ZR     X1,LSTMOV         IF NOT GOING
          SX6    X1-1
          SA6    A1
          BX6    X6-X6             CLEAR OPPONENTS PREDICTED RESPONSE
          SA6    PONDR
          EQ     MYMOVE
 NEWPOS   SPACE  4,13
***       NEWPOS - ENTER NEW POSITION.
*
*         EXIT   TO LSTMOV.
*                CLEARS REPEAT TABLE.
*

          ENTRY  NEWPOS
 NEWPOS   SX6    REPPB
          SA6    REPPZ
          SX6    -INFIN
          SA6    ALPHA
          BX6    -X6
          SA6    A6+B1
          ERRNZ  BETAR-ALPHA-1
          SA1    SWICH
          BX6    X6-X6
          SA6    SAVES             CLEAR SAVED SCORE FROM LAST MOVE
          SA6    DRAWS             CLEAR DRAW SCORE
          SA6    MVNTR             CLEAR NO. OF NON-TRIVIAL MOVES
          SA6    PONDR             CLEAR PONDERED MOVE
          LX1    59-S.DIS
          PL     X1,LSTMOV         IF NOT DISPLAY
          RJ     =XDUDLOAD
*         EQ     LSTMOV
 LSTMOV   SPACE  4,16
***       LSTMOV - LIST LEGAL MOVES.
*
*         ENTRY  (BOARD) = POSITION.
*
*         EXIT   TO YRMOVE.
*
*         USES   ALL REGISTERS, STACK1.
*         CALLS  LSTLGL.
*

          ENTRY  LSTMOV
 LSTMOV   RJ     =XLSTLGL          LIST LEGAL MOVES
          JUMPUP YRMOVE
 LSTLGL   SPACE  4,20
***       LSTLGL - LIST LEGAL MOVES.
*
*         ENTRY  (BOARD) = POSITION.
*
*         EXIT   (MOVES) = LIST OF LEGAL MOVES.
*
*         CALLS  CREATE, GENALL.
*

 LSTLGL   EQ     *+400000B
          RJ     =XCREATE
          RJ     =XGENALL
          SX6    MOVES
 LSTLGL1  SA6    INDEX
          SA2    LINDX
          SA1    X6                THE MOVE
          IX6    X6-X2
          PL     X6,LSTLGL         IF DONE WITH LEGAL SEARCH, RETURN
          SX6    MATE+PERM
          LX6    S.MOD             SET TO TEST FOR MATE
          BX6    X6+X1
```

```
          SA6    A1
 +        RJ     =XTRSRCH          SEARCH IT
 -        VFD    30/5LLSTMV        TRACEBACK
          SA1    INDEX
          SX6    X1+LE.MOV         ADVANCE
          EQ     LSTLGL1           TRY NEXT MOVE
 PONDER   SPACE  4,10
***       PONDER - THINK ON OPPONENTS TIME.
*
*         ENTRY  SQRLSTS AND OPPONENTS MOVES SET UP AS BY LSTMOV.
*                (PONDR) = PREDICTED OPPONENT MOVE.
*                S.THI = 0 MEANS FIRST CALL TO PONDER.  PONDER WILL
*                SET S.THI.
*                S.THI = 1 MEANS RESUME PONDERING. (PNDPAU1 IS ENTERED).
*                S.THG = 1 MEANS PONDR MOVE WAS ACTUALLY MADE.
*                THEREFORE, THE TREE SEARCH NEED NOT CONTINUE TO CHECK
*                FOR TYPE-INS, BUT MUST WORRY ABOUT TIME CONTROL AND
*                PREPARE TO TERMINATE THE SEARCH.
*                S.THF = 1 MEANS PONDR SEARCHING ALREADY DONE.  AFTER
*                PONDER HAS SET S.THF, YPMOVE WILL NOT CALL PONDER BACK
*                BUT WILL CALL MYMOVE AFTER IT HAS OFFICIALLY PROCESSED
*                THE OPPONENTS MOVE.  THEN MYMOVE WILL USE THE MACHINES
*                MOVE THAT HAD BEEN PREVIOUSLY COMPUTED BY PONDER.
*                NOTE: PONDER WILL NOT BE RECALLED IF THE OPPONENT MAKES
*                A MOVE THAT WAS NOT THE PREDICTED ONE.

          ENTRY  PONDER

 PONDER   SA1    PONDR
          MX6    1
          LX1    59-S.THF
          NG     X1,PONDER2        IF HAVE ALREADY FINISHED
          LX1    S.THF-S.THI
          NG     X1,PNDPAU1        IF NOT FIRST CALL

*         FIRST CALL: SET UP TO START PONDERING.

          BX6    X6+X1             SET PONDERING FLAG
          LX6    1+S.THI
          SA6    A1
          MX6    59                FLAG NO (-1 MSEC) PONDER FREE TIME
          SA6    TIMPO
          BX6    X6-X6
          SA6    WIERD             CLEAR WIERD MESSAGE WORD
          ATIME  MYMVTM            GET START TIME
          SX6    M.ACS+M.PRO       SET NULL MOVE
          R=     X7,POND           SET POND SEARCH MODE
          LX7    S.MOD
          BX6    X7+X6
          SA6    VMOVE
          RJ     =XEVALU8          COMPUTE THE MOVE
          SA1    BSTMV             SAVE MOVE
          SA2    SCORE             ALSO SAVE SCORE
          BX6    X1
          BX7    X2
          SA6    PBSMV
          SA7    PSCOR
          SA1    WIERD             SAVE WIERD WORD
          BX6    X1
          SA6    PWIRD
          RJ     =XTIMUSD          GET TIME TO COMPUTE MOVE
          SA6    PTMMV
          SA3    TIMPO             HAVE WE RECORDED FREE TIME
          PL     X3,PONDER1        IF YES
          SA6    A3                ELSE THE WHOLE MOVE WAS FREE
 PONDER1  SA1    PONDR
          ZR     X1,PONDER2        IF HAD TO ABORT PONDERING
          SX6    B1
          LX6    S.THF
          BX6    X1+X6             SET PONDER ALREADY DONE FLAG
          SA6    A1
          RJ     =XXLATPV          TRANSLATE PRINCIPAL VARIATION
          SA6    PPNDR             SAVE NEXT PONDR MOVE
          SA7    PKILL             AND MACHINES PREDICTED REJOINDER
 PONDER2  JUMPUP YPMOVE            WAIT FOR OPPONENTS MOVE
 PNDPAU   SPACE  4,10
***       PNDPAU - PAUSE WHILE PONDERING TO CHECK FOR TYPE-INS.
*
*         CALLED FROM TRSRCH BEFORE DOWNDATING.
*                ALSO RE-ENTERED FROM PONDER.
*
*         PNDPAU RETURNS IMMEDIATELY IF NOT PONDERING OR IF
*         PONDR MOVE HAS ALREADY BEEN MADE.  OTHERWISE, PNDPAU
*         CHECKS FOR A TYPE-IN.  IF NONE, PNDPAU RETURNS.  IF A
*         TYPE-IN, THEN PLY 0 SQRLSTS ARE RESTORED AND YPMOVE IS
*         ENTERED.  WHEN YPMOVE IS DONE AND PONDERING IS TO BE
*         CONTINUED, PONDER IS CALLED WITH S.THI SET AND PNDPAU IS
*         ENTERED AT PNDPAU1 TO RESUME PONDERING.
*
*         CALLS  READIF, EVPNTC, YPMOVE.

          ENTRY  PNDPAU.

 PNDPAU.  BSS    0                 ENTERED FROM PNDPAU
          SA1    PONDR
          LX1    59-S.THI
          PL     X1,PNDPAU         RETURN IF NOT PONDERING
          LX1    S.THI-S.THG
          NG     X1,PNDPAU         RETURN IF PONDR MOVE ALREADY MADE
          RJ     =XREADIF          CHECK FOR TYPE-IN
          ZR     X6,PNDPAU         RETURN IF NO TYPE-IN

*         SAVE VITAL CELLS.

          SA4    VMOVE             SAVE SOME CELLS
          SA1    PRECS
          SA5    SCORE
          SA2    LPECS
          BX6    X4
          BX7    X5
          SA6    PNDSAV            SAVE VMOVE
          SA7    A6+B1             SAVE SCORE
          BX6    X1
          SA6    A7+B1             SAVE PRECS
          BX7    X2
          SA7    A6+B1             SAVE LPECS

*         RESTORE PLY 0 SQRLSTS FOR YPMOVE.

          SA1    SQRLT             ADDRESS OF PLY 0 SQRLSTS
          SB3    SQRLSTBE-SQRLST   MAX LENGTH OF SQRLSTS
          BX0    X1
          SA0    SQRLST
          RE     B3                RESTORE PLY 0 NODE BANK AND SQRLSTS
          RJ     =XECRERR          IF ERROR
          JUMPUP YPMOVE            PROCESS THE TYPE-IN

*         RETURN FROM YPMOVE TO RESUME PONDERING.

 PNDPAU1  SA1    PONDR
          LX1    59-S.THG
          PL     X1,PNDPAU2        IF PONDR MOVE NOT YET MADE
          RJ     =XEVPNTC          CHECK AND START PONDER TIME CONTROL
 PNDPAU2  SA1    PNDSAV            RESTORE SAVED CELLS
          SA2    A1+B1
          BX6    X1
          SA6    VMOVE             RESTORE VMOVE
          BX7    X2
          SA7    SCORE             RESTORE SCORE
          SA1    A2+B1
          SA2    A1+B1
          BX6    X1
          BX7    X2
          SA6    PRECS             RESTORE PRECS
          SA7    LPECS             RESTORE LPECS
          EQ     PNDPAU            RETURN TO RESUME SEARCHING
 NOMOVE   SPACE  4,22
***       NOMOVE - MAKE THE NULL MOVE.
*
*         EXIT   REVERSES THE ON-THE-MOVE BITS.
*                CLEARS EP SQUARES.
*                TO NEWPOS.
*
*         USES   ALL REGISTERS, STACK1.
*


          ENTRY  NOMOVE
 NOMOVE   SA2    MYMOVEB
          SA1    STATUS
          BX6    X1-X2             TOGGLE ON-THE-MOVE BITS
          MX0    9
          LX0    27                CLEAR EP SQUARES
          BX6    -X0*X6
          LX0    30
          BX6    -X0*X6
          SA6    A1
          EQ     NEWPOS
 XLATPV   SPACE  4,60
***       XLATPV - TRANSLATE PRINCIPLE VARIATION INTO ENGLISH NOTATION.
*
*         ENTRY  (LIMBA) = ADDRESS OF PRINCIPLE VARIATION IN ECS.
*                (LIMBL) = LENGTH OF PRINCIPLE VARIATION IN ECS.
*                (BSTMV) = COMPUTERS MOVE.
*
*         EXIT   (PVARY) = ENGLISH NOTATION FOR PRINCIPLE VARIATION.
*                IF THE MOVE WAS PONDERED, THE FIRST WORD WILL BE
*                10H PONDR NNN , WHERE NNN = PONDER FREE TIME.
*                (X6) = OPPONENTS PREDICTED MOVE.
*                (X7) = MACHINES PREDICTED REJOINDER.
*
*         CALLS  UPDATE, ENGGEN, RDRCDD.
*         USES   ALL REGISTERS, STACK1.
*


 XLATPV   EQ     *+400000B
          BX6    X6-X6
          BX7    X7-X7
          SA6    PVARY             IN CASE OF NO VARIATION
          SA1    BSTMV
          ZR     X1,XLATPV         IF NO MOVE, NO VARIATION
          SA2    SLECS
          PL     X2,XLATPV1        IF SQUARE LISTS ALREADY SAVED
          SB3    SQRLSTL           NUMBER OF WORDS
          SA0    SQRLST            CM FWA
          MX6    1
          BX6    -X6*X2            CLEAR SL WRITTEN FLAG
          SA6    A2
          BX0    X6                ECS ADDRESS
          RJ     =XCHEKFE
          WE     B3                SAVE SQUARE LISTS
          RJ     =XECWERR          IF ERROR

 XLATPV1  SA1    LIMBA
          SA2    LIMBL
          SB3    X2
          BX0    X1
          SA0    PVARY
          BX6    X6-X6
          SA6    XLATPVA           CLEAR OPPONENTS PREDICTED MOVE
          SA6    A6+B1             CLEAR MACHINE REJOINDER
          SA6    INDEX
          ZR     B3,XLATPV2        IF NOTHING TO READ
          RE     B3
          RJ     =XECRERR
 XLATPV2  SA1    INDEX
          SA2    LIMBL
          IX6    X1-X2
          SX7    X1+B1
          PL     X6,XLATPV5        IF DONE
          SA7    A1                ADVANCE INDEX
          SA2    PONDR
          SX6    B1
          AX2    S.THF
          BX6    X6*X2             =1 IF PONDER JUST FINISHED, ELSE 0
          IX6    X1-X6             PLY NUMBER OR PLY NUMBER - 1
          SB2    X6
          SA1    PVARY+X1
          NE     B2,B1,XLATPV3     IF NOT AT OPPONENTS RESPONSE
          BX6    X1
          SA6    XLATPVA           SAVE OPPONENTS MOVE TO PONDER
          EQ     XLATPV4

 XLATPV3  SB2    B2-B1             PLY LEVEL - 1
          NE     B2,B1,XLATPV4     IF NOT AT OUR REJOINDER
          BX6    X1
          SA6    XLATPVA+1         ELSE SAVE OUR REJOINDER
 XLATPV4  SB2    NBOARD
          RJ     =XENGGEN
          SA1    A1                RE-FETCH MOVE
          SA6    A1                STORE TRANSLATION
          RJ     =XUPDATE          MODIFY NBOARD
          SA1    MSIDE
          BX6    -X1               COMPLEMENT MSIDE
          SA6    A1
          EQ     XLATPV2

 XLATPV5  BX6    X6-X6
          SA6    PVARY+X1          INDICATE END OF VARIATION
          SA1    SLECS
          SB3    SQRLSTL
          SA0    SQRLST
          BX0    X1
```

```
          RE     B3                RESTORE SQUARE LISTS
          RJ     =XECRERR
          SA1    PONDR
          LX1    59-S.THF
          PL     X1,XLATPV6        IF HAVENT JUST FINISHED PONDERING
          SA1    TIMPO             GET PONDR FREE TIME
          SX2    500               ROUND
          IX1    X1+X2
          RJ     =XRDRCDD          CONVERT TO DECIMAL
          AX6    3*6               TRUNCATE TO WHOLE SECONDS
          MX2    6*6
          SA1    =6L PONDR
          BX6    -X2*X6
          BX6    X1+X6
          SA6    PVARY
 XLATPV6  SA1    XLATPVA
          SA2    A1+B1
          BX6    X1                OPPONENTS PREDICTED MOVE
          BX7    X2                PREDICTED MACHINE REJOINDER
          EQ     XLATPV

 XLATPVA  BSS    2                 PREDICTED OPPONENT AND MACHINE MOVES
 TIMUSD   SPACE  4,19
***       TIMUSD - INDICATE TIME USED SINCE START OF MOVE.
*
*         EXIT   (X6) = CPU MILLISECONDS SINCE START OF MOVE.
*
*         USES   A1, A2, A6, X1, X2.
*         CALLS  SYS=.
*



          ENTRY  TIMUSD
 TIMUSD   EQ     *+400000B
          ATIME  MYMVTM+1          GET CURRENT CP MILLI-SEC
          SA1    MYMVTM            STARTING CP MILLI-SEC
          SA2    A1+B1
          MX6    30
          BX1    -X6*X1
          BX2    -X6*X2
          IX6    X2-X1
          EQ     TIMUSD
 CPTIME   SPACE  4,30
***       CPTIME - COMPUTE CPU TIME USED SINCE START OF JOB.
*
*         EXIT   (X6) = CPU MILLISECONDS SINCE START OF JOB.
*
*         USES   A1, A6, X1, X2.
*         CALLS  TIM=.
*



 NUCC     IF     -DEF,NUCC
          ENTRY  CPTIME
 CPTIME   EQ     *+400000B
          TIME                     24/JUNK,24/SEC,12/MSEC
          MX1    24
          BX1    -X1*X6            24/0,24/SEC,12/MSEC
          MX2    48
          BX1    X2*X1             SEC*10000B
          BX6    -X2*X6            MSEC
          BX2    X1                SEC*10000B
          LX1    1                 SEC*20000B
          IX1    X1+X2             SEC*30000B
          LX1    -7                SEC*140B
          IX1    X2-X1             SEC*7640B
          LX1    -2                SEC*1750B=SEC*1000D
          IX6    X1+X6             SEC*1000D+MSEC
          EQ     CPTIME
 NUCC     ENDIF
 THEMOV   SPACE  4,88
***       THEMOV - MAKE THE MOVE.
*
*         ENTRY  (BSTMV) = THE MOVE.
*
*         CALLS  UPDATE.
*         USES   ALL REGISTERS, STACK1.
*



 THEMOV   EQ     *+400000B
          SA1    BSTMV
          RJ     =XUPDATE
          SA1    BSTMV
          MX2    -6
          BX3    -X2*X1            TO SQUARE
          SA3    NBOARD+X3
          AX3    1
          AX1    13
          MX2    -5
          BX1    -X2*X1
          ZR     X3,THEMOV1        IF PAWN MOVE
          ZR     X1,THEMOV2        IF NOT IRREVERSIBLE
 THEMOV1  SX6    REPPB             RESET REPEAT POINTER
          SA6    REPPZ
 THEMOV2  SA1    REPPZ
          SX2    REPPB+NHIS*LE.PAC
          SB4    LE.PAC
          SX6    X1+B4
          IX7    X2-X6
          NG     X7,THEMOV3        IF TOO MANY POSITIONS
          SA6    A1+0              ADVANCE POINTER
          EQ     THEMOV5

 THEMOV3  SA2    REPPB+LE.PAC      FIRST WORD TO MOVE
          SB6    X6                LWA+1 TO MOVE FROM
 THEMOV4  BX6    X2                MOVE ONE WORD OF REPEAT TABLE
          SA6    A2-B4
          SA2    A2+B1
          SB7    A2
          LE     B7,B6,THEMOV4     MOVE ALL POSITIONS
 THEMOV5  SB2    63
 THEMOV6  SA1    NBOARD+B2
          BX6    X1
          SB2    B2-B1
          SA6    BOARD+1+B2
          PL     B2,THEMOV6
          SA1    CSTAT
          SA3    ENPAS
          SA5    CNT50
          SA2    OSIDE
          ZR     X2,THEMOV7        IF WHITE TO MOVE
          LX5    30
 THEMOV7  SA2    A1+B1
          SA4    A3+B1
          LX3    48-10
          LX1    39-30
          LX4    18-40
          LX2    9-20
          BX6    X1+X2
          BX6    X6+X3
          BX6    X6+X4
          BX6    X6+X5
          SA1    STATUS
          SA2    MYMOVEB
          BX1    X1*X2
          BX1    X1-X2
          BX6    X6+X1
          SA6    A1
          EQ     THEMOV
          END
 DEBUGR
          IDENT  DEBUGR
          SST
          LIST   X
*CALL     BOARD
*CALL     MEMORY
*CALL     IOCOM
*CALL     SQRLST
 DEBUGR   SPACE  4,42
          ENTRY  DEBUGR.
 DEBUGR.  BX6    X1
          SA6    DEBUGRA
          IF     DEF,OVERLAY
          SX2    =XLOADRA+2        OVERLAY LOADING FET
          SA3    X2+2              SAVE IN
          REWIND X2,RECALL
          BX6    X3
          SA6    A3                RESTORE IN
          WRITEF X2,RECALL         SAVE CURRENT (1,0) OVERLAY
          ENDIF
          SA1    NPLYC
          RJ     =XRDRCOD          CONVERT PLY NUMBER
          SA2    DEBUGRA
          MX0    30
          BX6    -X0*X6
          BX2    X0*X2
          BX6    X6+X2
          SA6    DBGMS
          SA1    TRSRCH
          LX1    30
          SA1    X1-1              CALL WORD
          BX1    -X0*X1            CALL NAME
          SA2    A2+B1
          BX6    X2+X1
          LX6    12
          SA6    A6+B1
          SA1    VMOVE
          BX1    -X0*X1
          RJ     =XRDRCOD
          SA6    A6+B1
          BX6    X6-X6
          SA6    A6+B1
          SX5    DBGMS
          RJ     =XWRLINE
          SX6    O.EDBG
          RJ     =XREADER
          ENTRY  DBUGGR
 DBUGGR   EQU    DEBUGR

 DEBUGRA  CON    6L PAUSE&6L
          CON    5L
 BKPCMD   SPACE  4,45
***       BKPCMD - PROCESS BREAKPOINT COMMAND.
*



          ENTRY  BKPCMD
 BKPCMD   SA2    BKPCMDA           BREAKPOINT ADDRESS
          ZR     X2,BKPCMD1        IF NO OTHER BREAKPOINT
          SA3    A2+B1             OLD CONTENTS
          BX6    X3
          SA6    X2+               CLEAR OLD BREAKPOINT
 BKPCMD1  SA1    ILINE
          SB2    54
          MX0    54
          RJ     =XRDRGNT          SKIP FIRST TOKEN
          RJ     =XRDRAOD          ASSEMBLE ADDRESS
          SX6    X6
          SA6    A2                SAVE ADDRESS
          ZR     X6,BKPCMD2        IF NO NEW BREAKPOINT
          SA3    MEMORY
          AX3    30
          IX3    X6-X3
          PL     X3,BKPCMD3        IF OUTSIDE FL
          SA4    X6                OLD CONTENTS
          BX7    X4
          SA7    A2+B1
          SX7    X6+B1
          LX7    30                FORM EQ TO RETURN
          MX4    1
          LX4    -3
          BX7    X4+X7
          SA7    A7+B1             EQ RETURN
          SA4    A7+B1             EQ PROBKP
          BX7    X4
          SA7    X6                STORE JUMP TO PROCESS BREAKPOINT
 BKPCMD2  BSS    0
          IF     DEF,OVERLAY
          SX2    =XLOADRA+2        OVERLAY LOADING FET
          SA3    X2+2              SAVE IN
          REWIND X2,RECALL
          BX6    X3
          SA6    A3                RESTORE IN
          WRITEF X2,RECALL         SAVE CURRENT (1,0) OVERLAY
          ENDIF
          RJ     =XREREAD          RETURN

 BKPCMD3  BX6    X6-X6             CLEAR BREAKPOINT ADDRESS
          SA6    A6
          SX5    =C* OUT OF RANGE.*
          RJ     =XERRMSG
          EQ     BKPCMD2

 BKPCMDA  DATA   0                 BREAKPOINT ADDRESS
 PROBKP1  DATA   0                 BREAKPOINT CONTENTS
          DATA   0                 EQ RETURN
          EQ     PROBKP            MODEL JUMP
 PROBKP   SPACE  4,15
***       PROBKP - PROCESS BREAKPOINT.
*



 PROBKP   SAVER  PROBKPA
```

```
          SA1    PROBKPB
          RJ     =XDEBUGR
          RESTORE PROBKPA
          EQ     PROBKP1           EXECUTE INSTRUCTION

 PROBKPA  BSS    24                REGISTER SAVE AREA
 PROBKPB  DATA   5L BKPT

*CALL     COMCSVR
 NUCC     IF     -DEF,NUCC
 IDL=     SPACE  4,88
***       IDL= - DUMMY IDLE CONTROL POINT FOR RSR=.
*
 IDL=     EQ     *+400000B
 IDL1     SA1    1                 WAIT FOR RA+1 TO CLEAR
          NZ     X1,IDL1
          EQ     IDL=
 NUCC     ELSE
*CALL     COMCIDL
*CALL     COMCRPV
 NUCC     ENDIF
          END
INITCON
          IDENT  INITCON
          SST
          LIST   G
          NOREF  BIT0,BIT1,BIT2,BIT3,BIT4,BIT5,BIT6,BIT7,B,T
          NOREF  I,J,K,L,M,N
*CALL     CONST
 ISSQL    MACRO  EXPR,IP,JP,KP
 I        SET    0
          DUP    8
 I        OCTMIC I
 BIT'I'   SET    0
 I        SET    I+1
          ENDD
 I        EXTRACT IP
 J        EXTRACT JP
 K        EXTRACT KP

 I        SET    IS
 1        DUP    IN
 J        SET    JS
 2        DUP    JN
 K        SET    KS
 3        DUP    KN
          SETBIT EXPR+28
 K        SET    K+KC
 3        ENDD
 J        SET    J+JC
 2        ENDD
 I        SET    I+IC
 1        ENDD

 BIT0     SET    BIT3/40B
 BIT3     SET    BIT3-BIT0*40B

 +        VFD    5/BIT3
          VFD    10/BIT4
          VFD    10/BIT5
          VFD    10/0
          VFD    10/BIT1
          VFD    10/BIT2
          VFD    5/BIT0
          ENDM
 FIELD    MACRO  SKIP,WIDTH,VALUES
 +        VFD    SKIP/0
          IFNE   SKIP,60
          IRP    VALUES
          VFD    WIDTH/VALUES
          IRP
          ENDIF
          ENDM
          MACRO  EXTRACT,I,IF,IM,II
 I!S      SET    IF 1
 I!N      SET    IM 1
 I!C      SET    II 1
          ENDM
          PURGMAC SETBIT
 SETBIT   MACRO  EXPR
          LOCAL  M
 T        SET    EXPR
 L        SET    T/8
 B        SET    T-L*8
 L        OCTMIC L
 M        BIT    B+1
 BIT'L'   SET    BIT'L'+M
          ENDM
 ISQRL    MACRO  EXPR,IP,JP,KP
 I        SET    0
          DUP    8
 I        OCTMIC I
 BIT'I'   SET    0
 I        SET    I+1
          ENDD
 I        EXTRACT IP
 J        EXTRACT JP
 K        EXTRACT KP

 I        SET    IS
 1        DUP    IN
 J        SET    JS
 2        DUP    JN
 K        SET    KS
 3        DUP    KN
          SETBIT EXPR
 K        SET    K+KC
 3        ENDD
 J        SET    J+JC
 2        ENDD
 I        SET    I+IC
 1        ENDD

 +        VFD    20/0
 I        SET    0
          DUP    8
 I        OCTMIC I
          VFD    10/BIT'I'
 I        SET    I+1
          ENDD
          VFD    20/0
          ENDM

 ISSSQL   MACRO  S,EXPR,IP,JP,KP
 I        SET    0
          DUP    8
 I        OCTMIC I
 BIT'I'   SET    0
 I        SET    I+1
          ENDD
 I        EXTRACT IP
 J        EXTRACT JP
 K        EXTRACT KP

 I        SET    IS
 1        DUP    IN
 J        SET    JS
 2        DUP    JN
 K        SET    KS
 3        DUP    KN
          SETBIT EXPR+28
 K        SET    K+KC
 3        ENDD
 J        SET    J+JC
 2        ENDD
 I        SET    I+IC
 1        ENDD

          VFD    1/S,4/0
          VFD    10/BIT1
          VFD    10/BIT2
          VFD    10/BIT3
          VFD    10/BIT4
          VFD    10/BIT5
          VFD    5/0
          ENDM

          ORG    ONBRD
 ONBRD    ISQRL  8*I+J,(0,8,1),(0,8,1)
          ORG    EIGHT
 EIGHT    ISQRL  I+J,(0,8,1),(0,2,56)
          ORG    MBMSK
          VFD    10/-0,20/0,10/-0,20/0
          ORG    PSMSK
          VFD    12/-0,18/0,12/-0,18/0
          ORG    SQ2PS

          DUP    8,2
          VFD    12/1776B,48/0
          VFD    12/1776B,48/0

 I        SET    48
 J        SET    53

 1        DUP    6
 K        SET    I
 L        SET    J

 2        ECHO   ,M=(300,340,160,070,034,016,007,003)
          VFD    12/2000B+L,8/M!B,40/0
          VFD    12/2000B+K,8/M!B,40/0

 L        SET    L-6
 K        SET    K-6
 2        ENDD

 I        SET    I+1
 J        SET    J-1
 1        ENDD

          DUP    8,2
          VFD    12/1776B,48/0
          VFD    12/1776B,48/0
          ORG    SQ2BT
 L        SET    39
 I        SET    0
 1        DUP    2
 2        DUP    4
 J        SET    1
 K        SET    L
          DUP    8,2
          VFD    3/2,9/K-J,48/I
 J        SET    J+1

 L        SET    L-10
 2        ENDD
 L        SET    59
 I        SET    1
 1        ENDD
          ORG    *
          SPACE  4
          ORG    BT2SQ
          DUP    20,1
          CON    -1

 I        SET    0
          DUP    8
          CON    -1
          DUP    8,2
          CON    I
 I        SET    I+1

          CON    -1
          ENDD

          DUP    20,1
          CON    -1
          ORG    MVDIR
          BSSZ   1
          ORG    MVMSK
          BSSZ   1
 S        MICRO  1,, +
          DUP    2
          ORG    MVDIR'S'KING
          FIELD  60
          ORG    MVDIR'S'QUEN
          FIELD  12,6,(-11,-10,-9,-1,+1,+9,+10,+11)
          ORG    MVDIR'S'BISH
          FIELD  36,6,(-11,-9,+9,+11)
          ORG    MVDIR'S'NITE
          FIELD  60
          ORG    MVDIR'S'ROOK
          FIELD  36,6,(-10,-1,+1,+10)
          ORG    MVDIR'S'PAWN
          FIELD  60
          ORG    MVMSK'S'PAWN
          ISSQL  'S'8+I,(-1,2,2)
          ORG    MVMSK'S'ROOK
          BSSZ   1
          ORG    MVMSK'S'NITE
          ISSQL  I*J*8+I*K*3-I*J*K,(-1,2,2),(1,2,1),(-1,2,2)
          ORG    MVMSK'S'BISH
          BSSZ   1
          ORG    MVMSK'S'QUEN
```

```
          BSSZ   1
          ORG    MVMSK'S'KING
          ISSQL  I*J-4*J+I/6*5*J,(5,4,1),(-1,2,2)
          ORG    CNCHK'S'KING
          BSSZ   1
          ORG    CNCHK'S'QUEN
          ISSSQL 1,I*J-4*J+I/6*5*J,(5,4,1),(-1,2,2)
          ORG    CNCHK'S'BISH
          ISSSQL 1,I+J,(-8,2,16),(-1,2,2)
          ORG    CNCHK'S'NITE
          ISSSQL 0,I*J*8+I*K*3-I*J*K,(-1,2,2),(1,2,1),(-1,2,2)
          ORG    CNCHK'S'ROOK
          ISSSQL 1,I*J,(-1,2,2),(1,2,7)
          ORG    CNCHK'S'PAWN
          ISSSQL 0,'S'8+I,(-1,2,2)

 S        MICRO  1,, -
          ENDD
          ORG    DIRTB-77
          BSSZ   77+78

          ECHO   ,I=(1,9,10,11)
 J        SET    1
          DUP    7
          ORG    DIRTB+J*I
          VFD    12/2000B+I,30/J,18/I
          ORG    DIRTB-J*I
          VFD    12/1777B-I,30/J,18/-I
 J        SET    J+1
          ENDD

          ECHO   ,I=(8,12,19,21)
          ORG    DIRTB+I
          VFD    12/2000B+I,48/0
          ORG    DIRTB-I
          VFD    12/1777B-I,48/0
          ENDD

          ORG    EXPND
 I        SET    21
 K        SET    0
 J        SET    12
 1        DUP    8
 2        DUP    8
          VFD    30/K,12/2000B+J,18/I
 J        SET    J-4
          IFLT   J,0,2
 J        SET    56
 K        SET    K+1
 I        SET    I+1
 2        ENDD
 I        SET    I+2
 1        ENDD
          END
CHESS11
          IDENT  CHE$$11
          SST
 DISPLAY  IF     DEF,DISPLAY
          TITLE  CHE$$11 - CHESS INTERFACE TO DYNAMIC USER DISPLAY
          SPACE  4
***       CHE$$11 - CHESS INTERFACE TO DYNAMIC USER DISPLAY
*
*         AUTHOR:
*               K. E. GORLEN
*               VOGELBACK COMPUTING CENTER
*               NORTHWESTERN UNIVERSITY
*               4/1/70
*
          TITLE                   ASSEMBLY CONSTANTS,COMMON BLOCKS, AND
, MACRO DEFINITIONS
          SPACE  4
 XO       EQU    54B              CHESS PIECE X OFFSET
 YO       EQU    120B             CHESS PIECE Y OFFSET
 RA65     EQU    65B              LWA+1 LOAD
          LIST   -G,-A,-D
 11       THIS
          EXT    DUDSET.
          EXT    DUDLOAD,DUDDRP$
          EXT    DUDKBP$,ALC$,CLOCK$,DUDCLK$,RCP$,MTR$
          EXT    BOARD$,BOARD.
          EXT    CHESMEN,CHESLEN
*CALL     VERSION
*CALL     DUDCOM
          SPACE  4
**        COMMON BLOCKS
*CALL     MEMORY
*CALL     BOARD
*CALL     CONST
*CALL     SQRLST
          SPACE  2
*CALL     KEYCOM
          USE
*CALL     DISPMAC
*CALL     CHARMAC
          SPACE  4
 DISDIS   SPACE  4
 DISDIS   MACRO  LABL
          MX0    1
          SA1    SCREEN
          SA2    L=LABL+X1
          BX6    -X0*X2
          SA6    A2
          ENDM
 DISBLD   MACRO  LABL
          SA1    SCREEN
          SB6    X1
          RJ     LABL
          SA1    SCREEN
          MX0    1
          SA1    L=LABL+X1
          BX6    -X0*X1
          SA6    A1
          ENDM
 DISUPD   SPACE  4
 DISUPD   MACRO  LABL
          SA1    DUD!LABL
          RJ     ENTER
          SA1    L=LABL
          NG     X1,DUD!LABL!1
          SB6    B0
          SX6    B0
          SA6    SCREEN
          RJ     LABL
 DUD!LABL!1 SA1    R=LABL
          NG     X1,DUD!LABL!2
          SB6    B1
          SX6    B1
          SA6    SCREEN
          RJ     LABL
 DUD!LABL!2 RJ     EXIT
          ENDM
 DISOFF   SPACE  4
 DISOFF   MACRO  LABL
          MX0    1
          SA1    SCREEN
          SA2    L=LABL+X1
          BX6    X0+X2
          SA6    A2
          ENDM
 DISREL   SPACE  4
 DISREL   MACRO  LABL
          SX1    0
          SA2    SCREEN
          SB2    L=LABL+X2
          RJ     =XALC$
          ENDM
 DISTRT   SPACE  4
          MACRO  DISTRT,LABL
 DISL     MICRO  1,, LABL
 LABL!A    BSS    2                RETURN ADDRESSES

 LABL     EQ     *+400000B
          SA1    LABL
          BX6    X1
          SA6    LABL!A+B6
          ENDM
 DISEXT   SPACE  4
 DISEXT   MACRO
          SA1    SCREEN
          SB6    X1
          JP     'DISL'A+B6
          ENDM
          SPACE  4
**        LINK MACRO FOR (1,1) OVERLAY
          SPACE  2
          MACRO  LINK,ENTRY
          EXT    ENTRY
+         EQ     ENTRY.
-         VFD    30/ENTRY+1
          ENDM
          SPACE  4
**        DISLINK MACRO FOR (1,1) OVERLAY
          SPACE  2
          MACRO  DISLINK,LABL
 X        MICRO  6,1,*LABL*
          IFC    EQ,*'X'**,1
          ENTRY  D=LABL
 D=LABL   BSS    1
          ENDM
          TITLE                   DUDLOAD -  OVERLAY INITIALIZATION
**        DUDLOAD - OVERLAY INITIALIZATION
*
*         ENTRY  (MEMORY+3) = LWA+1 FAKE ECS.
*


          CON    0                ECS PLUG TABLE
          ENTRY  CHE$$11
 CHE$$11  SB1    1
          SA1    DUDLOADA
          PL     X1,DUDLOAD4      IF DUD ALREADY LOADED
          MX7    0                ELSE, CLEAR FLAG
          SA7    A1
          SA7    W=STAT           CLEAR DUD STATUS WORDS
          SA7    W=ISTAT

**        DISPLAY BUFFER POINTER TABLE INITIALIZATION

          SA1    RA65             LWA+1 OF THIS OVERLAY
          SX2    X1               SET FWA AND LWA+1 = LWA+1 LOAD FOR
          SX1    X1               EACH UNSPECIFIED BUFFER
          LX1    30
          BX3    X1+X2
          MX0    60-18
          SA2    W=DPL            LWA+1 OF DISPLAY BUFFER POINTERS
          SA1    W=DPF            FWA OF DISPLAY BUFFER POINTERS
          SB2    X1
          SB3    X2
 DUDLOAD1 SA1    B2               PICK UP NEXT POINTER
          BX2    -X0*X1
          NZ     X2,DUDLOAD2      IF ADDRESS PRESENT
          BX6    X3+X1            INSERT ADDRESS
          SA6    B2
 DUDLOAD2 SB2    B2+B1            ADVANCE TABLE ADDRESS
          NE     B2,B3,DUDLOAD1   IF NOT END OF TABLE

**        (1,1) OVERLAY ENTRY POINT LINKAGE

          SA1    DUDLINK
 DUDLOAD3 BX6    X1
          SA6    X6               STORE LINKAGE JUMP
          SA1    A1+B1
          NZ     X1,DUDLOAD3      LOOP TO END OF TABLE

**        LOAD DYNAMIC USER DISPLAY DRIVER

          SA1    DUD
          RJ     MTR.

**        PRE-ALLOCATE FAKE ECS.

          SA1    MEMORY+3         LWA+1 FAKE ECS
          RJ     =XDUDECS

**        BUILD CHESS BOARD DISPLAY

 DUDLOAD4 SX1    DUDLOAD
          RJ     HOLDBG           HOLD BACKGROUND
          RJ     BLDIS            BUILD BOTH DISPLAYS
          SX6    B1
          SA6    HOLDBGA          LET BACKGROUND GO
          RJ     EXIT             EXIT DOES NOT RETURN

 DUD      VFD    18/3RDUD,3/2,39/W=STAT
 DUDLOADA VFD    60/-1
          SPACE  4
**        LINKAGE TABLE FOR (1,1) OVERLAY
          SPACE  2
 DUDLINK  BSS    0
*CALL     LINK
          DATA   0                END OF TABLE
          TITLE                   DISPLAY ENVIRONMENT TABLE
**        DISPLAY ENVIRONMENT TABLE - INTERFACE WITH DUD
          SPACE  4
```

```
          ENTRY  W=STAT,W=ISTAT,W=RFL,W=ARCL,W=MSC,W=KSW,W=KMAD,W=DPF
          ENTRY  W=DPL,W=XJP,W=A4B4,W=A5B5,W=IKEY,W=ICLK
          ENTRY  W=KSTAT,W=CNT
          EXT    DUDKBP$
          EXT    ALCHOP$

 W=STAT   VFD    60/0              PRIMARY STATUS WORD
 W=ISTAT  VFD    60/0              INTERRUPT STATUS WORD
 W=RFL    VFD    60/1              FIELD LENGTH REQUEST WORD
 W=ARCL   VFD    60/1              AUTO-RECALL CONTROL WORD
 W=MSC    VFD    60/0              REAL TIME MILLISECOND CLOCK
 W=KSW    VFD    36/0,24/W=KBF     KEYBOARD STATUS WORD
 W=KMAD   VFD    60/0              KEYBOARD MESSAGE ADDRESS
 W=DPF    VFD    60/DPF            BUFFER POINTER TABLE FWA
 W=DPL    VFD    60/DPL            BUFFER POINTER TABLE LWA+1
 W=XJP    BSSZ   21B               EXCHANGE JUMP PACKAGE BUFFER
 W=A4B4   EQU    W=XJP+4           REGISTERS A4,B4
 W=A5B5   EQU    W=XJP+5           REGISTERS A5,B5
 W=IKEY   VFD    30/DUDKBP$,30/0   KEYBOARD PROCESSOR ADDRESS
 W=ICLK   VFD    30/DUDCLK$,30/0   CLOCK INTERRUPT PROCESSOR ADDRESS
          SPACE  4
 W=KSTAT  VFD    60/0              KEYBOARD INPUT STATUS BIT
 W=CNT    VFD    30/48,30/0        CURRENT KEYBOARD CHARACTER POSITION
 W=KBF    BSS    12                KEYBOARD BUFFER
          TITLE                    DISPLAY BUFFER POINTER TABLE
          SPACE  4
 DPF      BSS    0
 LABLS    DISPL  CHAR,MED,3
 BOARD    DISPL  CHAR,MED,1
 INFO     DISPL  CHAR,SMALL,1
 D=ECS    DISPLA ,,,,,,,,OFF       FAKE ECS.  MUST BE FIRST.
 PIECES   DISPL  DOT,,1,*
 TWINK    DISPL  DOT,,1,*
 SQRL     DISPL  CHAR,LARGE,8,*
 MOVS     DISPL  CHAR,SMALL,1,*
 DPL      BSS    0
          TITLE  LINKAGE FROM 0,0 OVERLAY
 DUDLINK  SPACE  4
          EJECT
**        TWINKLE - FLASH CHESS PIECE
*
*         CALLING SEQUENCE:
*
*         CALL TWINKLE(INDEX)
*         INDEX -- INDEX OF SQUARE PIECE IS ON
          SPACE  4
 TWINKLE. SA1    X1
          BX7    X1                PICK UP INDEX
          SX1    TWINKLE
          RJ     HOLDBG
          SA7    TWINKLEA          SAVE SQUARE INDEX
          SA1    X7+BOARD
          BX7    X1
          SA7    A7+B1             SAVE PIECE CODE
          MX6    0                 REMOVE PIECE FROM BOARD
          SA6    A1
          RJ     BLDB              BUILD DISPLAY OF REMAINING PIECES
          SA1    TWINKLEA          RESTORE PIECE TO BOARD
          SA2    A1+B1
          BX6    X2
          SA6    X1+BOARD
          SX7    B1                ENABLE BACKGROUND PROCESSING
          SA7    HOLDBGA           LET BACKGROUND RUN
          SA1    WHITE             COMPUTE CORE REQUIRED FOR FLASHING
          AX1    59                PIECE BUFFER
          BX1    -X1-X2            COMPLEMENT IF WHITE .FALSE.
          SA2    X1+CHESLEN+6      X2 = PIECE LENGTH IN BYTES
          BX1    -X2
          SB2    L=TWINK+B6
          RJ     ALC$              ALLOCATE BUFFER STORAGE
          SA2    TWINKLEA
          SA1    A2+B1
          MX7    0
          RJ     BLDP              BUILD PIECE
          ZR     X7,TWINKLE2       IF LAST WORD STORED IN BUFFER
 TWINKLE1 LX7    12                ELSE, LEFT JUSTIFY
          PL     X7,TWINKLE1
          SA7    B2                AND STORE
 TWINKLE2 SX6    6                 SET FLASH COUNT
          SA6    TWINKLED
          SA1    L=TWINK+B6        INTENSIFY PIECE
          MX6    1
          LX6    60-4
          BX6    X1+X6
          SA6    A1
 TWINKLE3 SB2    100D              DELAY 100 MS
          RJ     CLOCK$
          SA1    L=TWINK+B6
          SA2    TWINKLED
          SX6    X2-1              DECREMENT COUNTER
          SA6    A2
          MX7    1                 TOGGLE SIGN BIT OF BUFFER POINTER
          BX7    X1-X7
          SA7    A1
          NZ     X6,TWINKLE3       LOOP
          MX6    1                 RESTORE NORMAL INTENSITY
          LX6    60-4
          BX7    X7-X6
          SA7    A7
          RJ     ALCHOP$
          RJ     EXIT

 TWINKLEA VFD    60/0              SQUARE INDEX
 TWINKLEB VFD    60/0              PIECE CODE
 TWINKLED VFD    60/0              FLASH COUNT
DUDREAD   SPACE  4
**        DUDREAD - READ KEYBOARD
*
*         CALLING SEQUENCE:
*
*         CALL DUDREAD(BUFFER)
*
          SPACE  4
 DUDREAD. SB1    1
          BX5    X1                SAVE ADDRESS OF BUFFER
 DUDREAD1 SX1    W=KSTAT
          RJ     RCL               WAIT FOR INPUT TO COMPLETE

**        PACK INTERNAL BUFFER INTO USERS BUFFER

          MX0    48
          SA1    W=KSW
          SA2    X1                FIRST WORD OF INTERNAL BUFFER
          SB2    X5                FIRST WORD OF USER BUFFER
          SB3    B2+6              LWA+1 OF USER BUFFER
          MX5    1                 CIRCULAR COUNTER
          MX6    0                 ASSEMBLY REGISTER

 DUDREAD2 LX2    12                POSITION INTERNAL BUFFER WORD
          BX3    -X0*X2            EXTRACT THE CHARACTER
          BX2    -X0+X2            FLAG BYTE AS PROCESSED
          LX6    6                 POSITION ASSEMBLY REGISTER
          LX5    6                 SHIFT CIRCULAR COUNTER
          BX6    X6+X3             ASSEMBLE IT
          PL     X5,DUDREAD3       IF NOT ONE WORD ASSEMBLED
          SA6    B2                INTO USER BUFFER
          SB2    B2+B1
          EQ     B2,B3,DUDREAD     IF END OF BUFFER
          MX6    0                 CLEAR ASSEMBLY REGISTER
 DUDREAD3 ZR     X3,DUDREAD4       IF NO MORE DATA
          NZ     X2,DUDREAD2       IF MORE CHARACTERS IN THIS WORD
          SA2    A2+B1             NEXT WORD OF INTERNAL BUFFER
          EQ     DUDREAD2

 DUDREAD4 LX6    6                 SHIFT ASSEMBLY REGISTER
          LX5    6                 SHIFT CIRCULAR COUNTER
          PL     X5,DUDREAD4       IF NOT LEFT JUSTIFIED
          SA6    B2                INTO USER BUFFER
          MX6    0
          SA6    DUDREADA          CLEAR TYPEIN ACCEPTED FLAG
          SA6    W=KSTAT           CLEAR COMPLETE BIT
 DUDREAD5 SB2    B2+B1
          EQ     B2,B3,DUDREAD6
          SA6    B2
          EQ     DUDREAD5

 DUDREAD6 SA1    DUDREADB          INTERRUPT TO KEYBOARD PROCESSOR
          RJ     FTN
          EQ     DUDREAD           RETURN
          SPACE  2
 DUDREADA BSSZ   1                 TYPEIN ACCEPTED FLAG
 DUDREADB VFD    30/=XDUDKBP$,30/F=INT
          SPACE  4
**        DUDCLR - CLEAR KEYBOARD INPUT LINE.
*


 DUDCLR.  SA1    W=KMAD
          NZ     X1,DUDCLR         IF ERROR MESSAGE IS UP
          SA5    DUDREADA
          NG     X5,DUDCLR         IF ALREADY CLEARED
          MX6    59
          SA6    A5
          SA1    DUDCLR
          RJ     ENTER
          SA1    W=KSW             RESET KEYBOARD STATUS WORD
          MX6    24
          BX6    -X6*X1            CLEAR CHARACTER COUNT AND READY FLAG
          SA6    A1
          SX7    48                RESET W=CNT
          LX7    30
 DUDCLR1  SA7    W=CNT
          MX6    0                 RESET RESET AND WAITING FLAGS
          SA6    RESET
          RJ     EXIT
          SPACE  4
**        DUDMSG - DISPLAY KEYBOARD MESSAGE
*
*         CALLING SEQUENCE:
*
*         CALL DUDMSG(MESSAGE)
          SPACE  4
 DUDMSG.  SB2    B0
          SB3    3                 NUMBER OF WORDS
          MX0    48
 DUDMSG1  SA2    X1+B2             MOVE THREE WORD MESSAGE INTO BUFFER
          BX6    X2
          SA6    B2+MSGBUF
          SB2    B2+B1
          BX6    -X0*X6
          ZR     X6,DUDMSG2        IF END OF MESSAGE
          LT     B2,B3,DUDMSG1     IF NOT DONE
          EQ     DUDMSG

 DUDMSG2  GE     B2,B3,DUDMSG      IF DONE, RETURN
          SA6    B2+MSGBUF         CLEAR NEXT WORD
          SB2    B2+B1
          EQ     DUDMSG2
 DUDERR   SPACE  4
***       DUDERR - DISPLAY ERROR MESSAGE.
*
*         ENTRY  (X5) = ADDRESS OF MESSAGE.
*


 DUDERR.  SA1    DUDERR
          RJ     ENTER
          SA5    X5                MOVE WORD 1
          BX7    X5
          SA7    DUDERRA
          SX7    A7
          MX6    59
          SA6    DUDREADA          SET TYPEIN ACCEPTED FLAG
          SA7    W=KMAD            SET MESSAGE ADDRESS
          SA5    A5+B1
          BX6    X5
          SA6    X7+B1             MOVE WORD 2
          SA5    A5+B1
          BX6    X5
          SA6    A6+B1             MOVE WORD 3
          SA1    W=KSW             CLEAR KEYBOARD READY FLAG
          MX0    48
          LX0    36
          BX6    X0*X1
          LX0    12
          BX5    -X0*X6
          ZR     X5,DUDERR2        IF COLUMN 1
          SX5    B1
          LX5    48
          IX6    X6-X5             DECREMENT CHARACTER COUNT
          SA6    A1
          SA1    W=CNT
          SX6    X1                WORD OFFSET
          AX1    30                BIT OFFSET
          SX1    X1+12
          SX2    X1-60
          NZ     X2,DUDERR1        IF SAME WORD
          SX1    0
          SX6    X6-1
 DUDERR1  LX1    30
          BX7    X1+X6
          EQ     DUDCLR1

 DUDERR2  SA6    A1                W=KSW
          MX6    0
```

```
          SA6    RESET
          RJ     EXIT
          SPACE  4

 DUDERRA  BSSZ   3                ERROR MESSAGE BUFFER
          SPACE  4
**        DUDDROP - DROP DUD DISPLAY DRIVER
*
*         CALLING SEQUENCE:
*
*         CALL DUDDROP
          SPACE  4
 DUDDROP. SX1    F=DROP
          RJ     FTN
          MX0    12
          SA1    DUDLINK          DE-LINK ENTRY POINTS
 DUDDROP1 SX6    X1-1
          LX6    30
          BX2    X0*X1
          BX6    X6+X2
          SA6    X1               STORE JUMP
          SA1    A1+B1
          NZ     X1,DUDDROP1      LOOP TO END OF TABLE
          MX6    59
          SA6    DUDLOADA         SET DUD NOT LOADED
          EQ     DUDDRP$          TO COMPLETE DROP
          SPACE  4
**        DUDKILL - KILL PROGRAM
*
*         CALLING SEQUENCE:
*
*         CALL DUDKILL
          SPACE  4
 DUDKILL. SX1    F=KILL
          RJ     FTN
          PS
 DUDDISP  SPACE  4
**        DUDDISP - SET DISPLAYS.
*
*         ENTRY  (X1) = DISPLAY NAMES, LEFT ADJUSTED.
*


 DUDDISP. BX5    X1
          SX1    DUDDISP          HOLD BACKGROUND
          RJ     HOLDBG
          MX0    54
          LX5    6
          SX2    1R.
          BX6    -X0*X5
          ZR     X6,DUDDISP1      IF NO LEFT SCREEN CHANGE.
          BX7    X6-X2
          ZR     X7,DUDDISP1      IF NO LEFT SCREEN CHANGE
          SA1    DUDDISPA-1+X6
          BX6    X1
          SA6    BLDISA           SET LEFT SCREEN ROUTINE ADDRESS
 DUDDISP1 LX5    6
          BX6    -X0*X5
          ZR     X6,DUDDISP2      IF NO RIGHT SCREEN CHANGE
          BX7    X6-X2
          ZR     X7,DUDDISP2      IF NO RIGHT SCREEN CHANGE
          SA1    DUDDISPA-1+X6
          BX6    X1
          SA6    BLDISB           SET LEFT SCREEN DISPLAY
 DUDDISP2 RJ     BLDIS            BUILD DISPLAYS
          SX6    B1
          SA6    HOLDBGA          LET BG RUN
          RJ     EXIT             RETURN

 DUDDISPA RJ     DISPLA
          RJ     DISPLB
          RJ     DISPLC
 DUDMOVE  SPACE  4,10
**        DUDMOVE - MAKE MOVE ON DISPLAY.
*


 DUDMOVE. SA1    DUDMOVE
          RJ     ENTER
          RJ     BLDIS            BUILD DISPLAY
          RJ     EXIT
          SPACE  4
**        DUDINFO - PUT MISC INFORMATION ON SCREENS.
*
 DUDINFO. RJ     INFO
          EQ     DUDINFO
          SPACE  4
**        DUDSQRL - DISPLAY SQUARE LIST.
*
*         ENTRY  (X1) = POINTER TO SQUARE LIST.
*
 DUDSQRL. BX6    X1
          SA6    SQRLD
          DISUPD SQRL
 DUDECS   SPACE  4
**        DUDECS - ALLOCATE FAKE ECS WITH DUD UP.
*
*         ENTRY  (MEMORY+3) = NEW LWA+1 OF FAKE ECS.
*
*         USES   A1, A2, A6, X1, X2, X6, B2, B4, B7.
*         CALLS  HOLDBG, ALC$, EXIT.
*


 DUDECS.  SX1    DUDECS
          RJ     HOLDBG           HOLD BACKGROUND
          SA1    MEMORY+3
          SB4    X1               NEW LWA+1
          SA1    D=ECS
          LX1    30
          SB2    X1               FWA FAKE ECS
          SX1    B4-B2            NEW LENGTH
          SB2    A1
          RJ     =XALC$           ALLOCATE CORE
          SX6    B1
          SA6    HOLDBGA          LET BACKGROUND RUN
          RJ     EXIT             EXIT INTERRUPT MODE
          SPACE  4
          TITLE                   SUBROUTINES
**        FTN$ - PASS FUNCTION TO DUD
*
*         MODE:  INTERRUPT
*
*         ENTER: X1 = FUNCTION WORD
*
*         USES:  X1,X6,A1,A6
*
*         CALLS: RCL$
          SPACE  4
 FTN$     PS
          BX6    X1
          SA6    W=ISTAT          POST REQUEST
          SX1    A6               ENTER AUTO-RECALL
          SB1    1
          RJ     RCL$
          EQ     FTN$
          SPACE  4
**        FTN - PASS FUNCTION TO DUD
*
*         MODE:  NORMAL
*
*         ENTER: X1 = FUNCTION WORD
*
*         USES:  X1,X6,A1,A6
*
*         CALLS: RCL
          SPACE  4
 FTN      PS
          BX6    X1
          SA6    W=STAT           POST REQUEST
          SX1    A6               ENTER AUTO-RECALL
          SB1    1
          RJ     RCL
          EQ     FTN
          SPACE  4
**        RCL$ - ENTER AUTO-RECALL ON STATUS WORD
*
*         MODE:  INTERRUPT
*
*         ENTER: X1 = STATUS WORD ADDRESS
*
*         USES:  X1,X2,A1,A2
*
*         CALLS  MTR$
          SPACE  4
 RCL$     PS
          SA2    X1
          LX2    59
          NG     X2,RCL$          IF COMPLETE BIT ALREADY SET
          SA2    RCLA             ELSE, ENTER AUTO-RECALL
          BX1    X1+X2            ADD ADDRESS TO REQUEST
          RJ     MTR$
          EQ     RCL$

 RCLA     VFD    18/3RRCL,3/2,39/0
          SPACE  4
**        RCL - ENTER AUTO-RECALL ON STATUS WORD
*
*         MODE:  NORMAL
*
*         ENTER: X1 = STATUS WORD ADDRESS
*
*         USES:  X1,X2,A1,A2
*
*         CALLS: MTR
          SPACE  4
 RCL      PS
          SA2    X1
          LX2    59
          NG     X2,RCL           EXIT IF COMPLETE BIT ALREADY SET
          SA2    RCLA             ELSE, ENTER AUTO-RECALL
          BX1    X1+X2            ADD ADDRESS TO REQUEST
          RJ     MTR.
          EQ     RCL
          SPACE  4
**        MTR - ISSUE MONITOR REQUEST
*
*         MODE:  NORMAL
*
*         ENTER  B1 = 1
*                X1 = MONITOR REQUEST
*
*         USES:  A1,A6,X1,X6

 MTR.     PS     0
          BX6    X1
 MTR1     SA1    B1
          NZ     X1,MTR1          WAIT FOR RA+1 TO CLEAR
          SA6    B1               ISSUE REQUEST
 MTR2     SA1    B1
          NZ     X1,MTR2          WAIT FOR RA+1 TO CLEAR
          EQ     MTR.             RETURN
          SPACE  4
**        ENTER - ENTER INTERRUPT MODE
*
*         MODE:  NORMAL
*
*         ENTER: X1 = BACKGROUND RETURN ADDRESS (BITS 30-47)
*
*         EXIT:  B1 = 1
*
*         USES   X1,X6,A1,A6,B1,B2
*
*         CALLS  RCL,FTN
          SPACE  4
 ENTER    PS
          SB1    1                CONSTANT 1
          LX1    30
          SB2    X1               SET RETURN ADDRESS
          SX1    ENTERA
          RJ     RCL              WAIT FOR INTERRUPT READY
          SA1    ENTERB           REQUEST INTERRUPT TO ENTER1
          RJ     FTN
          JP     B2               BACKGROUND RETURN

 ENTER1   MX6    0
          SA6    ENTERA           CLEAR COMPLETE BIT
          EQ     ENTER            INTERRUPT RETURN

 ENTERA   VFD    60/1
 ENTERB   VFD    30/ENTER1,30/F=INT
          SPACE  4
**        EXIT - EXIT FROM INTERRUPT ROUTINE
*
*         MODE:  INTERRUPT
*
*         EXIT DOES NOT RETURN
*
*         CALLS  CLOCK$
          SPACE  4
 EXIT     PS
          SX6    B1               SET INTERRUPT COMPLETE BIT
          SA6    ENTERA
          SB2    B0               EXIT FROM INTERRUPT MODE
 EXITA    RJ     CLOCK$
```

```
         SPACE  4
**       HOLDBG - HOLD UP BACKGROUND PROGRAM
*
*        MODE:  INTERRUPT
*
*        ENTER: X1 ' ADDRESS TO CONTINUE BACKGROUND EVENTUALLY
*
*        CALLS: ENTER
*
 HOLDBG  PS     0
         SB7    X1              BACKGROUND ADDRESS
         SX1    HOLDBG1         ADDRESS OF RECALL REQUEST
         LX1    30
         RJ     ENTER           ENTER INTERRUPT MODE
         MX6    0
         SA6    HOLDBGA         CLEAR COMPLETE BIT
         EQ     HOLDBG          RETURN IN INTERRUPT MODE

 HOLDBG1 SX1    HOLDBGA
         RJ     RCL             WAIT UNTIL FORGROUND SAYS TO GO
         JP     B7              CONTINUE BACKGROUND

 HOLDBGA VFD    60/0            STATUS WORD
 BLDIS   SPACE  4
**       BLDIS - BUILD BOTH DISPLAYS.
*
*        MODE:  INTERRUPT
*

 BLDIS   EQ     *+400000B
         SB6    B0
         SX6    B0
         SA6    SCREEN
 BLDISA  RJ     DISPLA          BUILD LEFT SCREEN
         SB6    B1
         SX6    B1
         SA6    SCREEN
 BLDISB  RJ     DISPLB          BUILD RIGHT SCREEN
         EQ     BLDIS           RETURN

 SCREEN  BSS    1
 DISPLA  SPACE  4
**       DISPLA - BUILD THE A DISPLAY.
*
*        BUILD NORMAL GAME BOARD DISPLAY.
*

 DISPLA  DISTRT
         DISOFF INFO
         DISREL SQRL
         DISREL MOVS
         DISDIS LABLS
         DISDIS PIECES
         DISDIS BOARD
         DISREL TWINK
         SA0    BOARD
         RJ     BLDB            BUILD BOARD
         DISEXT
 DISPLB  SPACE  4
**       DISPLB - BUILD B DISPLAY.
*
*        BUILD DEBUGGING DISPLAY.
*

 DISPLB  DISTRT
         DISREL PIECES
         DISREL TWINK
         DISREL SQRL
         DISOFF LABLS
         DISOFF BOARD
         DISDIS INFO
         DISDIS MOVS
         DISEXT
 DISPLC  SPACE  4
**       DISPLC - BUILD C DISPLAY.
*
*        BUILD DEBUG BOARD DISPLAY.
*

 DISPLC  DISTRT
         DISREL TWINK
         DISREL MOVS
         DISDIS LABLS
         DISDIS BOARD
         DISOFF INFO
         DISDIS PIECES
         SA0    NBOARD
         RJ     BLDB            BUILD BOARD DISPLAY
         DISBLD SQRL
         DISEXT
 SQRL    SPACE  4
**       SQRL - BUILD SQUARE LIST DISPLAY.
*

 SQRL    EQ     *+400000B
         SA3    SQRL
         BX6    X3
         SA6    SQRLE+B6        SAVE RETURN ADDRESS
         SA1    SQRLD           ADDRESS OF SQUARE LIST
         SA1    X1
         SA2    A1+B1
         BX6    X1
         BX7    X2
         SA6    SQRLA
         SA7    A6+B1
         CX6    X6
         CX7    X7
         IX6    X6+X7
         SA6    A7+B1
         SX1    X6
         SB2    L=SQRL+B6
         RJ     ALC$
         SA6    SCREEN
         MX0    1
         LX0    48
         MX7    57
         SX5    60B
         SA4    SQRLC
         SB3    B0
         SA1    SQRLA
         ZR     X1,SQRL2        IF NOTHING IN FIRST WORD
         NG     X1,SQRL2
         PX1    X1
         SA0    12+BT2SQ
         NX6    X1,B4
         AX6    X0,B4
 SQRL1   BX1    -X6*X1
         SA2    B4+A0
         BX3    -X7*X2
         AX2    3
         IX3    X5*X3
         IX2    X5*X2
         BX6    X3
         LX6    48
         LX3    12
         LX2    36
         IX6    X6+X4
         IX6    X6+X3
         IX6    X6+X2
         SA6    B2
         SB2    B2+B1
         NX6    X1,B4
         AX6    X0,B4
         NZ     X6,SQRL1
         NZ     B3,SQRL3        IF DONE
 SQRL2   SA1    A1+B1
         ZR     X1,SQRL3        IF DONE
         NG     X1,SQRL3
         LX1    48
         PX1    X1
         SA0    60+BT2SQ
         SB3    B1
         NX6    X1,B4
         AX6    X0,B4
         EQ     SQRL1

 SQRL3   SA1    SCREEN
         SB6    X1
         JP     SQRLE+B6

 SQRLA   BSS    2               SAVED SQUARE LIST
 SQRLB   BSS    1               NUMBER OF BITS
 SQRLC   VFD    12/6030B+XO
         VFD    12/7030B+YO
         VFD    12/2RX
         VFD    12/6030B+XO
         VFD    12/2R0
 SQRLD   CON    CSTAT           SQUARE LIST ADDRESS
 SQRLE   BSS    2               SAVED RETURN ADDRESSES
 INFO    SPACE  4
**       INFO - PUT INFORMATION ON SCREEN.
*

 INFO    EQ     *+400000B
         SX1    ROLLA
         RJ     RCL             WAIT TILL ROLL STOPS
         SX6    X5
         SA6    INFOD           SAVE PARAMETER
         MX0    48
         BX7    X7-X7           CLEAR ASSEMBLY REGISTER
         MX4    1               CIRCULAR SHIFT COUNTER
         SA2    INFOB           IN POINTER
         SB2    X2
         MX3    1
         SA5    X5              MESSAGE
 INFO1   SX1    6000B           X COORDINATE
         RJ     INSB            INSERT BYTE
         SA1    INFOC           Y COORDINATE
         SX6    X1-10B
         SA6    A1
         RJ     INSB            INSERT BYTE
         SB3    31              MAXIMUM NUMBER TO INSERT
 INFO2   LX5    12
         BX1    -X0*X5          EXTRACT BYTE
         ZR     X1,INFO4        IF DONE
         LX4    12
         SB3    B3-B1
         RJ     INSB            INSERT BYTE
         PL     X4,INFO3        IF NOT NEW INPUT WORD
         SA5    A5+1
 INFO3   PL     B3,INFO2        IF NOT FULL LINE
         SX1    6000B           FILL OUT WORD
         RJ     INSB            INSERT BYTE
         EQ     INFO1           ELSE START NEW LINE

 INFO4   LX7    12
         LX3    12
         PL     X3,INFO4        LEFT ADJUST
         SA7    B2              STORE LAST WORD
         SX7    B2+B1
         SA7    INFOB           UPDATE IN POINTER
         SA1    INFOC
         SX6    X1-2
         SA6    A1              UPDATE Y COORDINATE
         SX6    INFOA
         RJ     ROLL            ROLL DISPLAY IF NECESSARY
         SA5    INFOD           RESTORE PARAMETER
         EQ     INFO            RETURN

 INFOA   CON    INFO$
         CON    INFO.
         CON    7720B
         CON    7060B
 INFOB   CON    INFO$           IN POINTER
 INFOC   CON    7720B           Y COORDINATE
 INFOD   CON    0               SAVED PARAMETER
 INSB    SPACE  4,26
**       INSB - INSERT BYTE.
*
*        ENTRY  (X1) = BYTE.
*               (X3) = CIRCULAR SHIFT COUNTER.
*               (X7) = PARTIALLY ASSEMBLED WORD.
*               (B2) = ADDRESS TO STORE WORD.
*               (B1) = 1.
*
*        EXIT   EVERYTHING IS UPDATED.
*


 INSB    EQ     *+400000B
         LX7    12
         BX7    X1+X7           INSERT BYTE
         LX3    12
         PL     X3,INSB         IF NOT FULL WORD, RETURN
         SA7    B2              STORE IT
         SB2    B2+B1           ADVANCE ADDRESS
         BX7    X7-X7           CLEAR ASSEMBLY REGISTER
         EQ     INSB            RETURN
 ROLL    SPACE  4,88
**       ROLL - ROLL DISPLAY.
*
*        ENTRY  (ROLLA) = 1 IF NORMAL MODE, 0 IF INTERRUPT MODE.
*               (X6) = ADDRESS OF PARAMETERS:
*                      FWA BUFFER
```

```
*                               LWA+1 BUFFER
*                               TOP OF SCREEN
*                               BOTTOM OF SCREEN
*                               CURRENT IN POINTER
*                               NEXT Y COORDINATE
*
*               DISPLAY IS ROLLED IF LESS THAN 16 WORDS LEFT IN BUFFER
*               OR NEXT Y COORDINATE BELOW BOTTOM OF SCREEN.
*


 ROLL     EQ     *+400000B
          SA6    ROLLB            SAVE PARAMETER ADDRESS
          SB2    B1+B1
          SA2    X6+B1            LWA+1 BUFFER
          SA3    A2+B2            BOTTOM OF SCREEN
          SA4    A3+B1            CURRENT IN POINTER
          SA5    A4+B1            NEXT Y
          IX6    X5-X3
          IX7    X2-X4
          SX7    X7-16
          BX7    X7+X6
          PL     X7,ROLL          IF NO NEED TO ROLL, RETURN
          SA2    ROLLA
          NZ     X2,ROLL2         IF NOT INTERRUPT MODE
          SB2    B1               MINIMUM DELAY
          RJ     CLOCK$           DELAY
          EQ     ROLL3            CONTINUE

 ROLL2    BX6    X6-X6
          SA6    A2               SET ROLL BUSY
          SA1    ROLL
          LX1    30
          SB2    X1               SAVE RETURN ADDRESS
          SA1    ROLLC            INTERRUPT TO ROLL3
          RJ     FTN
          JP     B2               RETURN

 ROLL3    SA1    ROLLB            PARAMETERS
          SB2    B0               MOVE OFFSET
          MX5    2                CHECK FOR COORDINATES
          SX7    B1
          LX7    36               Y INCREMENT
          SA2    X1               FWA BUFFER
          SA2    X2               FIRST WORD OF BUFFER
          SA3    X1+4             IN POINTER
          SB3    X3               LAST WORD ADDRESS OF DATA
          MX0    48
          SA4    X1+2             TOP OF SCREEN
 ROLL4    BX6    X2*X5
          ID     X6,ROLL9         IF COORDINATE WORD
          BX6    X2
 ROLL5    SA6    A2+B2            MOVE IT
          SA2    A2+B1            NEXT WORD
          SB4    A2
          LT     B4,B3,ROLL4      IF NOT DONE
 ROLL6    SA3    X1+4
          SX7    X3+B2
          SA7    A3               UPDATE IN POINTER
          BX6    X6-X6
 ROLL7    ZR     B2,ROLL8         IF DONE CLEARING EXCESS
          SA6    A6+B1
          SB2    B2+B1
          EQ     ROLL7

 ROLL8    SA3    A3+B1
          SX6    X3+B1            ADVANCE Y
          SA6    A3
          BX6    X1
          RJ     ROLL             TRY AGAIN
          SX6    B1               (RETURNS ONLY WHEN DONE)
          SA6    ROLLA            SET NOT BUSY
          SB2    B0
          RJ     CLOCK$           EXIT

 ROLL9    IX6    X7+X2
          BX2    X6
          LX2    24
          BX2    -X0*X2
          IX2    X4-X2
          PL     X2,ROLL5         IF NOT ABOVE TOP OF SCREEN
 ROLL10   SB2    B2-B1            CHANGE MOVE OFFSET
          SA2    A2+B1
          SB4    A2
          BX6    X2*X5
          GE     B4,B3,ROLL6      IF END OF BUFFER
          ID     X6,ROLL9         IF COORDINATE
          EQ     ROLL10           CONTINUE SEARCH

 ROLLA    CON    1                COMPLETE BIT
 ROLLB    DATA   0                PARAMETER ADDRESS
 ROLLC    VFD    30/ROLL3,30/F=INT INTERRUPT WORD
          EQ     INFO
          TITLE                   BLDB - BUILD CHESS BOARD DISPLAY
          SPACE  4
***       BLDB -- BUILD CHESS BOARD DISPLAY
*
*         MODE:  INTERRUPT
*
*               BLDB SCANS THE BOARD ARRAY AND CALLS BLDP TO CONSTRUCT
*         A CHESS PIECE ON THE APPROPRIATE SQUARE.
*
*         CALLS  ALC$,BLDP
*         USES:  A1,A2,A3,A7,B2,B3,X1,X2,X3,X7
          SPACE  4
 BLDB     PS

**        COMPUTE STORAGE REQUIREMENT

          SB1    1                CONSTANT 1
          SB2    63               BOARD INDEX
          SA2    WHITE            COLOR SWITCH
          SA3    ORIENT           ORIENTATION SWITCH
          AX2    59               FORM MASKS
          MX1    59
          BX3    X3-X2            SET K AND Q FILE LABELS
          BX3    -X1*X3
          SA1    X3+BLDBA
          BX7    X1
          SA7    FILE+1
          MX3    0
 BLDB1    SA1    B2+A0            LOAD PIECE CODE
          ZR     X1,BLDB2         IF SQUARE EMPTY
          BX1    -X2-X1           COMPLEMENT IF WHITE .FALSE.
          SA1    X1+=XCHESLEN+6   GET LENGTH OF PIECE IN BYTES
          IX3    X1+X3            ADD TO TOTAL
 BLDB2    SB2    B2-B1            DECREMENT SQUARE INDEX
          PL     B2,BLDB1         LOOP THROUGH ENTIRE BOARD.
          BX1    -X3              SET FOR CALL TO ALC$
          SB2    L=PIECES+B6
          SX6    A0
          SA6    BLDBB+B6         SAVE BOARD ADDRESS
          SA3    BLDB
          BX6    X3
          SA6    BLDBC+B6         SAVE RETURN ADDRESS
          RJ     ALC$             ALLOCATE STORAGE

**        BUILD CHESS PIECES IN DISPLAY BUFFER

          SA1    BLDBB+B6
          SA0    X1               RESTORE BOARD ADDRESS
          MX7    0                CLEAR FOR BLDP ROUTINE
          SB3    64               INITIALIZE BOARD INDEX
 BLDB3    SB3    B3-B1            DECREMENT BOARD INDEX
          SA1    B3+A0            LOAD PIECE CODE
          SX2    B3               SET SQUARE INDEX
          RJ     BLDP             CONSTRUCT PIECE IN BUFFER
          NZ     B3,BLDB3         LOOP THROUGH BOARD
          ZR     X7,BLDB          IF LAST WORD STORED IN BUFFER
 BLDB4    LX7    12               ELSE, LEFT JUSTIFY
          PL     X7,BLDB4
          SA7    B2               AND STORE
          JP     BLDBC+B6         RETURN

 BLDBA    DATA   10LB  K  Q  B
          DATA   10LB  Q  K  B
 BLDBB    BSS    2                SAVED BOARD ADDRESS
 BLDBC    BSS    2                SAVED RETURN ADDRESSES
          SPACE  2
          TITLE                   BLDP - BUILD CHESS PIECE IN DISPLAY B
,UFFER
**        BLDP - BUILD CHESS PIECE IN DISPLAY BUFFER
*
*         ENTER:
*               B2 = CURRENT DISPLAY BUFFER ADDRESS
*               X1 = PIECE CODE
*               X2 = SQUARE INDEX
*               X7 = ANY PREVIOUSLY INCOMPLETED WORD
*
*         USES:  A1,A4,A7,B2,B7,X0,X1,X2,X3,X4,X5,X6,X7
          SPACE  4
 BLDP     PS
          ZR     X1,BLDP          IF NO PIECE
          SA4    WHITE            COLOR SWITCH
          AX4    59               FORM MASK
          BX1    -X4-X1           COMPLEMENT IF WHITE .FALSE.
          SA1    X1+=XCHESMEN+6   FETCH FWA OF PIECE COORDINATES
          SA4    ORIENT           BOARD ORIENTATION SWITCH
          BX2    X2-X4            COMPUTE SQUARE RANK AND FILE
          MX0    60-3             CALCULATE X AND Y TRANSLATIONS
          BX5    -X0*X2           SPLIT INDEX INTO RANK AND FILE
          AX2    3
          LX2    4                *20B
          LX4    B1,X2            *40B
          IX4    X2+X4            *60B
          SX4    X4+YO            +Y-OFFSET
          LX5    4                *20B
          LX2    B1,X5            *40B
          IX5    X2+X5            *60B
          SX5    X5+XO            +X-OFFSET
          MX0    60-12
          SB7    B1+B1
          SA1    X1               LOAD FIRST WORD OF COORDINATES

**        TRANSLATE PIECE TO APPROPRIATE SQUARE

 BLDP1    LX2    B7,X1            POSITION BIT 9 OF BYTE
          AX2    59               FORM MASK - 0 IF X, 1 IF Y
          LX1    12
          BX3    -X0*X1           ISOLATE COORDINATE BYTE
          ZR     X3,BLDP          STOP ON ZERO BYTE
          BX1    -X0+X1           FLAG BYTE AS PROCESSED
          BX6    X2*X4            MASK X TRANSLATION
          BX2    -X2*X5           OR Y TRANSLATION
          BX2    X2+X6            X2 = X OR Y TRANSLATION
          IX3    X3+X2            TRANSLATE COORDINATE
          LX7    12               ADD TO CURRENT BUFFER WORD
          BX7    X7+X3
          PL     X7,BLDP2         IF WORD NOT ASSEMBLED
          SA7    B2               ELSE, STORE IN DISPLAY BUFFER
          SB2    B2+B1            ADVANCE BUFFER ADDRESS
          MX7    0                CLEAR FOR NEXT WORD
 BLDP2    NZ     X1,BLDP1         IF NOT READY FOR NEXT WORD
          SA1    A1+B1            ELSE, LOAD NEXT WORD OF COORDINATES
          EQ     BLDP1            LOOP
 WHITE    DATA   -0
 ORIENT   DATA   0
          TITLE                   BDC - BLOCK DECIMAL CONVERSION
**        BDC - BLOCK DECIMAL CONVERSION FOR TREE LIMITS AND INDEX
*
*         MODE:  NORMAL
*
*         ENTER: B1 = 1
*               B2 = BLOCK FWA
*               A2 = BUFFER FWA
*               X2 = (BUFFER FWA)
          SPACE  4
 BDC      PS
          SB3    7B               SET INDEX
          SB4    1R0
          SA3    BDCA             .1P48+1
          SA4    A3+B1            10.P
          AX2    12               POSITION BASE COORDINATES

**        BEGIN DECIMAL CONVERSION

 BDC1     SA1    B2+B3            LOAD NEXT BINARY NUMBER
          MX6    0                CLEAR ASSEMBLY REGISTER
          ZR     X1,BDC3          SUPPRESS DISPLAY IF ZERO
          PX1    X1
          DX7    X1*X3
          FX1    X1*X3
          SB5    X1               ISOLATE TENS COEFFICIENT
          ZR     B5,BDC2          SUPPRESS LEADING ZERO
          SX6    B4+B5            CONVERT
          LX6    6                POSITION
 BDC2     FX7    X7*X4            COMPUTE REMAINDER
          SX7    X7+B4            CONVERT
          IX6    X6+X7            ASSEMBLE NUMBER
 BDC3     SX1    B3               COMPUTE COORDINATES
          LX1    4                BUFFER INDEX*20B
          LX7    X1,B1            BUFFER INDEX*40B
          IX7    X1+X7            BUFFER INDEX*60B
          IX7    X2+X7            ADD BASE COORDINATES
```

```
          LX7    12               POSITION
          BX6    X6+X7            ASSEMBLE WITH NUMBER
          SA6    A2+B3            STORE IN DISPLAY BUFFER
          SB3    B3-B1            DECREMENT INDEX
          PL     B3,BDC1          LOOP FOR 8 WORD BLOCK
          EQ     BDC              EXIT

 BDCA     VFD    60/1+0.1P48
 BDCB     VFD    60/10.P
          TITLE                   MISCELLANEOUS DISPLAYS
          SPACE  4
 LABLS$   BSS    0
          SPACE  2
**        LEFT SCREEN TITLE
          SPACE  2
          CHAR   270B,777B,(CHESS  'V')
          SPACE  2
**        RANK LABELS
          SPACE  2
 X        SET    60B
 Y        SET    150B
 RANK     SET    1

          DUP    8
 XXX      MICRO  RANK,1,*12345678*
          CHAR   X-24B,Y,'XXX'
 RANK     SET    RANK+1
 Y        SET    Y+60B
          ENDD
          FILL
          SPACE  2
**        FILE LABELS
          SPACE  2
 FILE     CHAR   X+24B,100B,(R  N  B  Q  K  B  N  R)
          FILL
          SPACE  2
**        MESSAGE BUFFER
          SPACE  2
          VFD    36/0,12/6000B,12/7044B
 MSGBUF   BSSZ   3
          SPACE  2
 LABLS.   BSS    0
 INFO$    BSSZ   300D
 INFO.    BSS    0
          SPACE  4
 DISPLAY  ELSE
 END      MICRO  1,,
 DISPLAY  ENDIF
          END    'END'
GLOBAL
          IDENT  GLOBAL
          SST
 DISPLAY  IF     DEF,DISPLAY
*CALL     KEYMACS
          ENTRY  GLOBAL
 GLOBAL   BSS    0
 X        LINE   (ALTER ),ANY
          LINE   (CHANGE ),CHA
          LINE   (COMPLEMENT),END
          LINE   (CLOCK),NNN
          LINE   (CREATE),CRE
          LINE   (DROP),END
          LINE   (DISPLAY ),DIS
          LINE   (ECS O),ECS
          LINE   (END),END
          LINE   (EXPLAIN ),ANY
          LINE   (GO),NNN
          LINE   (HARDCOPY),END
          LINE   (HELP),END
          LINE   (HOLD),END
          LINE   (ID ),ANY
          LINE   (INITIALIZE),END
          LINE   (MSG ),ANY
          LINE   (SAVE ),ANY
          LINE   (SETUP ),ANY
          LINE   (STRIP ),ANY
          LINE   (STATUS ),ANY
          LINE   (TOURNAMENT),END
          LINE   (USE ),ANY
          LINE   (WHAT),END
          LINE   (XEQ ),ANY

 CHA      CHOICE 01234567,CHA,*
          MEMBER (,),*
 DIS      CHOICE ( 01234567),DIS,END
 END      MEMBER (.)

 CRE      MEMBER (,),*,END
          CHOICE LD,*,*
          CHOICE XPRNBQK,*,*
          CHOICE KQ,*,*
          CHOICE RNB,*,*
          CHOICE 12345678,CRE

 NNN      MEMBER (,),NUM,END
 NUM      CHOICE 0123456789,NUM,END

 ECS      MEMBER (N),END,*
          MEMBER (F),*
          MEMBER (F),END

 ANY      MEMBER 00
          HEREM
 DISPLAY  ENDIF
          END
GLOBCON
          IDENT  GLOBCON
          SST
 DISPLAY  IF     DEF,DISPLAY
*CALL     KEYMACS
          ENTRY  GLOBCON
 GLOBCON  BSS    0
 X        LINE   (BOARD,),BOA
          LINE   (BLITZ),NNN
          LINE   (BKP ),OCT
          LINE   (KILL),END
          LINE   (NAME ),ANY
          LINE   (NULL MOVE),END
          LINE   (PARAMETERS),ANY
          LINE   (PINFO ),ANY
          LINE   (PRINT ),PRI
          LINE   (PSLIST ),ANY
          LINE   (PURGE),END
          LINE   (PVARIABLE ),ANY
          LINE   (PWRD ),ANY
          LINE   (REINCARNATE),END
          LINE   (RELIABILITY),END
          LINE   (RESIGN),END
          LINE   (REWIND ),ANY
          LINE   (RETURN ),ANY

 ANY      MEMBER 00
 BOA      CHOICE LD,*,*
          CHOICE XPRNBQK,BOA,*
          CHOICE 12345678,BOA,*
          CHOICE (,/),BOA,*
 END      MEMBER (.)
 PRI      MEMBER N,BRD,BRD
 BRD      LINE   BOARD,END
 NNN      MEMBER (,),NUM,END
 NUM      CHOICE 0123456789,NUM,END
 OCT      CHOICE 01234567,OCT,END
          HEREM
 DISPLAY  ENDIF
          END
GLOBLET
          IDENT  GLOBLET
          SST
 DISPLAY  IF     DEF,DISPLAY
*CALL     KEYMACS
          SPACE  4
 LINEC    MACRO  A,B
 'L'      LINE   A,LE!B
 L        MICRO  1,,*   *
          ENDM
          ENTRY  GLOBLET
 GLOBLET  BSS    0
 X        LINE   (LET ),LEN

 L        MICRO  1,, LEN
 LETMIC   MICRO  1,,*LINEC*
*CALL     LET

 LEB      MEMBER (=),BIN
 BIN      CHOICE 01234567,BIN,*
          CHOICE 89,NOB,*
 BOD      CHOICE B,END,DEE
 DEE      CHOICE D,END,END
 END      MEMBER (.)

 LED      MEMBER (=),DEC
 DEC      CHOICE 01234567,DEC,*
          CHOICE 89,NOB,BOD

 NOB      CHOICE 0123456789,NOB,DEE

          HEREM
 DISPLAY  ENDIF
          END
GLOBSWI
          IDENT  GLOBSWI
          SST
 DISPLAY  IF     DEF,DISPLAY
*CALL     KEYMACS
          ENTRY  GLOBSWI
 GLOBSWI  BSS    0

 X        LINE   (SWITCH ),SWI

 ALL      MICRO  1,,
 SWTYPE   MICRO  2,, 'SWTYPE'
          ECHO   ,A=('SWTYPE')
 BB       MICRO  1,1, A
          IF     -DEF,.'BB',3
 .'BB'    EQU    1
 'BB'     MICRO  1,,
 ALL      MICRO  1,, 'ALL','BB'

 CC       MICMIC 'BB'
 'BB'     MICRO  1,, 'CC',A
          ENDD

 ALL      MICRO  2,, 'ALL'
 FIN      MICRO  1,,
          ECHO   1,A=('ALL')
 FIN      MICRO  1,, 'FIN''A'

 FIN      MICRO  2,, 'FIN'

 LAB      MICRO  1,, SWI
          ECHO   2,A=('FIN')
 'LAB'    LINE   A,ONF
 LAB      MICRO  1,,*   *

 ONF      MEMBER (.),,FON
 FON      LINE   ( ON),END
          LINE   ( OFF),END

 END      MEMBER (.)

          HEREM
 DISPLAY  ENDIF
          END
QUEST0
          IDENT  QUEST0
          SST
 DISPLAY  IF     DEF,DISPLAY
*CALL     KEYMACS
          ENTRY  QUEST0
 QUEST0   BSS    0
          CHOICE PRNBQK,MOV,*
          CHOICE 0O,CAS

 MOV      CHOICE /,*,MOC
          CHOICE KQ,*,*
          CHOICE RNB,*,*
          CHOICE 12345678,*,*
 MOC      CHOICE -,TOS,*
          CHOICE *X,CAP

 CAP      CHOICE PRNBQ,*
          CHOICE /,*,PRO
 TOS      CHOICE KQ,*,*
          CHOICE RNB,*,*
          CHOICE 12345678,*,*
 PRO      CHOICE =,*,END
          CHOICE BRNQ,END

 CAS      MEMBER -,*
          MEMBER 0,*
          MEMBER -,*,END
          MEMBER 0,END

 END      MEMBER .
```

```
         HEREM
 DISPLAY ENDIF
         END
QUEST1
         IDENT  QUEST1
         SST
 DISPLAY IF     DEF,DISPLAY
*CALL    KEYMACS
         ENTRY  QUEST1
 QUEST1  BSS    0
 X       LINE   (ANY),END

 END     MEMBER .
         HEREM
 DISPLAY ENDIF
         END
QUEST3
         IDENT  QUEST3
         SST
 DISPLAY IF     DEF,DISPLAY
*CALL    KEYMACS
         ENTRY  QUEST3
 QUEST3  BSS    0

 NNN     CHOICE 0123456789,NNN,END

 END     MEMBER (.)

         HEREM
 DISPLAY ENDIF
         END
DUDVAL
         IDENT  DUDVAL
         SST
 DISPLAY IF     DEF,DISPLAY
         TITLE                  DUDVAL --- DYNAMIC USER DISPLAY TYPE-
,IN VALIDATION

***      DUDVAL --- DYNAMIC USER DISPLAY TYPE-IN VALIDATION
*
*        AUTHOR:
*              LAWRENCE R. ATKIN
*              CONTROL DATA CORP  (CGODSO)
*              12/02/69
*
*        ENTRY  (X1) = TABLEI.
*              EACH TABLE CONTAINS A SERIES OF LINKED 30 BIT ENTRIES:
*              6/CHARACTER,12/SIBLING,12/OFFSPRING
*
*              POSSIBLE VALUES:
*              0 QUEST0
*              1 QUEST1
*              2 0
*              3 QUEST3
*
*
*
*
*        LOGICAL = DUDVAL(CHAR,NEXT)
*
*        CHAR   THE CHARACTER JUST ENTERED ON THE KEYBOARD
*
*        NEXT   =0 IF THE NEXT CHARACTER IS NOT FORCED.
*               =-1 IF THE TYPE-IN IS COMPLETE.
*               >0 IF THE NEXT CHARACTER IS FORCED
*
*        LOGICAL=.TRUE. IF THE CHARACTER IS VALID
*               =.FALSE. OTHERWISE
*
*               IF RESET = 0 ON ENTRY, THE ENTIRE TYPE-IN IS RESCANNED.
*               RESET IS SET TO -1 BEFORE EXIT
*
*               IF GLOBESW = 0 ON ENTRY, TABLE1 IS NOT USED IN
*               VALIDATING THE TYPE-IN.
*
         TITLE -- DATA DIVISION
*CALL,DUDCOM

 VALUE   BSSZ   1                VALUE OF DUDVAL
 NEXT    BSSZ   1                DUMMY NEXT FOR RESCAN
 OLDNEXT BSSZ   1                PREVIOUS NEXT DURING RESCAN
 HOLD    BSSZ   1                HOLDS PARAMETER ADDRESS DURING RESCAN
 W.CNT   BSSZ   1                WORD AND BIT OFFSET DURING RESCAN
 W.KSW   BSSZ   1                CHARACTER COUNT DURING RESCAN

 TABLE   CON    =XGLOBCON
         CON    =XGLOBAL
         CON    =XGLOBLET
         CON    =XGLOBSWI
         CON    0
 TABLEMAX EQU   *-TABLE
         CON    0
 IAMAT   BSSZ   TABLEMAX         CURRENT POSITION IN EACH TABLE
 SAVE    BSSZ   TABLEMAX         SAVE AREA FOR IAMAT IN CASE INVALID

 RCL     VFD    18/3RRCL,3/2,39/W=STAT
 INT     VFD    30/DUDKBP$,30/F=INT

*CALL    KEYCOM

         EXT    W=STAT,DUDKBP$
         EXT    W=KSW,W=CNT
         EXT    W=KSTAT
         TITLE                  DUDSET -- SET TABLE BASE ADDRESSES
         ENTRY  DUDSET.
 DUDSET. SA3    DUDSETA+X1
         SX6    F=IOF            DISABLE INTERRUPTS
         SA6    =XW=STAT
         RECALL A6
         BX6    X3
         SA6    TABLE+TABLEMAX-1
         BX7    X7-X7
         SA7    RESET
         SX6    F=ION
         SA6    =XW=STAT         ENABLE INTERRUPTS
         RECALL A6
         SA1    INT
         BX6    X1
         SA6    =XW=STAT         RESCAN
         RECALL A6
         EQ     =XDUDSET         RETURN

 DUDSETA VFD    60/=XQUEST0
         VFD    60/=XQUEST1
         VFD    60/0
         VFD    60/=XQUEST3
         TITLE                  GETNTRY -- GET ENTRY FROM TABLE
**       GETNTRY GET ENTRY FROM TABLE
*
*        ENTRY:
*        X1     BASE ADDRESS OF TABLE
*        X2     ORDINAL WITHIN TABLE
*
*        EXIT:
*        X5     ENTRY (RIGHT JUSTIFIED)
*
*        DESTROYS X2,B4
*

         ENTRY  GETNTRY
 GETNTRY PS
         SB4    X1               BASE ADDRESS OF TABLE
         SX2    X2-1             REMOVE OFFSET
         LX2    59               /2, REMAINDER IN SIGN BIT
         SA5    B4+X2            LOAD WORD FROM TABLE
         NG     X2,GETNTRY       IF RIGHT ENTRY
         LX5    30
         EQ     GETNTRY          RETURN
         TITLE                  VALONE -- VALIDATE ONE CHARACTER
***      VALONE VALIDATE ONE CHARACTER
*
*        ENTRY/EXIT CONDITIONS:
*
*              SAME AS DUDVAL EXCEPT:
*
*              ADDRESS OF NEXT IS IN B7
*              CHARACTER IS IN B6
*              1 IS IN B1
*              RESET IS NOT CHECKED
*

 VALONEX  SA1   VALUE
          BX6   X1
          NG    X6,VALONE         IF LEGAL
          SA1   SAVE
 VALONEX1 BX7   X1
          SA7   A1-SAVE+IAMAT     RESTORE IAMAT
          SB5   A1-SAVE-TABLEMAX+1
          ZR    B5,VALONE         IF DONE
          SA1   A1+B1
          EQ    VALONEX1

 VALONE   PS
          MX7   0
          SA7   VALUE             INITIALIZE
          SA7   B7                NEXT
          SA1   TABLE
          ZR    B6,VALONE5        IF CHARACTER = 00B
          SA2   GLOBESW
          NZ    X2,VALONE1        IF CHECKING ALL TABLES
          SA1   A1+2              ELSE SKIP FIRST TWO TABLES
 VALONE1  ZR    X1,VALONEX        IF DONE
          SA2   A1+IAMAT-TABLE    IAMAT
          BX6   X2
          SA6   A1-TABLE+SAVE     SAVE IAMAT
 VALONE2  ZR    X2,VALONE3        IF NOT IN THIS TABLE
          RJ    GETNTRY           GET ENTRY
          MX0   54
          LX5   30+6
          BX0   -X0*X5            EXTRACT CHAR(I)
          ZR    X0,VALONE4        IF ANYTHING GOES
          SB4   X0
          MX0   48
          EQ    B4,B6,VALONE6     IF CHAR = CHAR(I)
          LX5   12
          BX2   -X0*X5            EXTRACT SIBLING(I)
          NZ    X2,VALONE2        IF ALTERNATIVES EXIST
          MX7   0
          SA7   A2                SET IAMAT(I)=0
 VALONE3  SA1   A1+B1             ADVANCE TO NEXT TABLE
          EQ    VALONE1

 VALONE4  MX6   59
          SA6   B7                NEXT
          EQ    VALONE            RETURN TRUE

 VALONE5  MX6   59
          EQ    VALONE            RETURN TRUE

 VALONE6  MX6   59
          SA6   VALUE
          LX5   12+12
          BX6   -X0*X5            EXTRACT OFFSPRING(I)
          ZR    X6,VALONE9        IF NO OFFSPRING
          SA6   A2                UPDATE IAMAT
          BX2   X6
          RJ    GETNTRY           GET ENTRY J
          LX5   30+6+12
          BX7   -X0*X5            EXTRACT  SIBLING(J)
          NZ    X7,VALONE7        IF ALTERNATIVES
          SA4   B7                NEXT
          MX7   54
          LX5   60-12
          BX7   -X7*X5            EXTRACT CHAR(J)
          ZR    X7,VALONE4        IF ANYTHING GOES
          ZR    X4,VALONE8        IF NEXT NOT DETERMINED
          BX7   X7-X4
          ZR    X7,VALONE3        IF NEXT = CHAR(J)
 VALONE7  MX7   60
          SA7   B7                SET NEXT = -0
          EQ    VALONE3

 VALONE8  NG    X4,VALONE3        IF MORE THAN ONE CHOICE ALREADY
          SA7   B7                SET NEXT = CHAR(J)
          EQ    VALONE3

 VALONE9  MX7   0
          SA7   IAMAT
          MX6   59                RETURN TRUE
 VALONE10 SB2   A7-IAMAT-TABLEMAX+1
          ZR    B2,VALONE4
          SA7   A7+B1             ZERO IAMAT
          EQ    VALONE10
          TITLE                  DUDVAL -- VALIDATE TYPE-INS FOR DUD
          ENTRY  DUDVAL

 DUDVAL   PS
          SB1   1
          SA2   RESET
          ZR    X2,DUDVAL2        IF RESET
 DUDVAL1  SA2   X1                CHAR
          SB6   X2
          SA1   A1+B1
```

```
          SB7    X1               ADDRESS(NEXT)
          RJ     VALONE           VALIDATE ONE CHARACTER
          EQ     DUDVAL           RETURN

 DUDVAL2  SX6    A1
          SA6    HOLD             SAVE PARAMETER ADDRESS
          SB7    NEXT
          SX7    B1
          SA7    IAMAT            INITIALIZE IAMAT
 DUDVAL3  SB6    A7-IAMAT-TABLEMAX+1
          ZR     B6,DUDVAL4
          SA7    A7+B1
          EQ     DUDVAL3

 DUDVAL4  MX7    59
          SA7    A2               RESET
          SX7    48
          LX7    30
          MX6    0
          SA6    NEXT
          SA6    OLDNEXT
 DUDVAL5  SA7    W.CNT
          SA2    W=CNT
          SA6    W.KSW
          IX6    X7-X2
          ZR     X6,DUDVAL8       IF DONE
          SA1    W=KSW
          SB2    X1
          SA3    B2+X7            WORD IN BUFFER
          MX0    48
          LX7    30
          SB2    X7
          AX3    X3,B2
          BX3    -X0*X3
          SB6    X3
          SA1    B7               NEXT
          BX6    X1
          SA6    OLDNEXT
          SA5    W=KSTAT
          MX4    59
          ZR     B6,DUDVAL7       IF CR
          RJ     VALONE           VALIDATE ONE CHARACTER
          MX5    0
          SX4    B0
          PL     X6,DUDVAL7       IF INVALID
          SA1    W.KSW
          SX6    X1+B1            INCREMENT CHAR COUNT
          SA2    W.CNT
          SB2    X2               WORD
          LX2    30
          SB3    X2-12            BIT
          PL     B3,DUDVAL6       IF NOT END OF WORD
          SB3    48
          SB2    B2+B1
 DUDVAL6  SX7    B2               WORD
          SX2    B3               BIT
          LX2    30
          BX7    X7+X2
          EQ     DUDVAL5

 DUDVAL7  SA1    W=KSW
          MX0    24
          BX7    -X0*X1
          SA1    W.KSW
          LX1    60-12
          BX7    X7+X1            FORM NEW W=KSW
          SA7    W=KSW
          SA1    W.CNT
          BX7    X1
          SA7    W=CNT
          SA1    HOLD
          SA1    X1+B1            ADDRESS OF NEXT
          SA2    OLDNEXT
          BX6    X2
          SA6    X1               NEXT
          BX6    X5
          SA6    W=KSTAT
          BX6    X4
          EQ     DUDVAL           RETURN FALSE

 DUDVAL8  SA1    HOLD
          SA1    X1
          SA2    X1               CHAR
          NZ     X2,DUDVAL1       IF NOT NEW TABLES
          SA3    NEXT
          SA4    A1+B1            ADDRESS OF NEXT
          BX6    X3
          SA6    X4               NEXT
          MX6    59
          EQ     DUDVAL           RETURN TRUE
 DISPLAY  ENDIF
          END
DUDKBP
          IDENT  DUDKBP$
          SST
 DISPLAY  IF     DEF,DISPLAY
          ENTRY  DUDKBP$
          TITLE  DUDKBP$ --- DYNAMIC USER DISPLAY KEYBOARD PROCESSING RO
,UTINE
***       DYNAMIC USER DISPLAY KEYBOARD PROCESSING ROUTINE
*
*         AUTHOR:
*                LAWRENCE R. ATKIN
*                CONTROL DATA CORP (CGODSO)
*                11/07/69
*
*         DUDKBP$ IS ENTERED AS AN INTERRUPT ROUTINE ANYTIME A CHARACTER
*         IS TYPED ON THE 6612 CONSOLE.
*
*         ENTRY INFORMATION:
*                A0        CONTAINS THE CHARACTER ENTERED
*
*         EXIT INFORMATION:
*                W=KSW, W=CNT, RESET, AND WAITING HAVE BEEN UPDATED
*
*         CALLS:
*                DUDVAL(CHAR,NEXT)
*
*         ACTION:
*                THE ACTION ACCOMPLISHED DEPENDS UPON THE CHARACTER
*                TYPED IN AND THE SETTING OF VARIOUS FLAGS.
*
*                CHAR = BKSP       IF COLUMN 1, THEN ILLEGAL
*                                  ELSE UPDATE W=KSW AND W=CNT, SET
*                                  RESET = .TRUE.
*                CHAR = LBLK       IF COLUMN 1, THEN ILLEGAL
*                                  ELSE UPDATE W=KSW AND W=CNT, SET
*                                  RESET = .TRUE.
*                CHAR = SPACE BAR  SET CHAR = BLANK
*                CHAR = RBLK       RIGHT SCREEN TOGGLE
*                CHAR = CAR RETURN IF W=STAT = F=NOP, THEN
*                                  SET W=STAT = 1
*                                  ELSE SET WAITING = .TRUE.
*                CHAR > 60B        ILLEGAL
*                CHAR = OTHER      CALL DUDVAL(CHAR,NEXT)
*                                  UPDATE W=KSW AND W=CNT
*                                  IF NEXT > 0, THEN RECYCLE
*

**        ASSEMBLY CONSTANTS

 K=PLUS   EQU    1R+              PLUS KEY
 K=MNUS   EQU    1R-              MINUS KEY
 K=LPRN   EQU    1R(
 K=RPRN   EQU    1R)
 K=LBLK   EQU    53B              LEFT BLANK
 K=RBLK   EQU    55B              RIGHT BLANK
 K=BKSP   EQU    61B              BACKSPACE
 K=CARR   EQU    60B              CARRIAGE RETURN
 K=SPCE   EQU    62B              SPACE BAR
*CALL     DUDCOM

 DUDKBP$  SB1    1                CONSTANT ONE
          SB2    A0               CHARACTER ENTERED
          MX0    48               CONVIENIENT MASK
          SA1    W=KSW            KEYBOARD STATUS WORD
          SA2    W=CNT            BYTE AND WORD COUNTS
          SA4    SAVE
          NZ     X4,ILLCHAR       IF STILL PROCESSING ANOTHER TYPE-IN
          SB3    X1
          SA3    X2+B3            CURRENT WORD IN BUFFER
          LX1    12               POSITION CHARACTER COUNT LOWER
          SB3    K=BKSP
          EQ     B2,B3,BKSP       IF BACKSPACE
          SB3    K=LBLK
          EQ     B2,B3,LBLK       IF LEFT BLANK
          SB3    K=RBLK
          EQ     B2,B3,RBLK       IF RIGHT BLANK
          SB3    K=SPCE
          EQ     B2,B3,SPCE       IF SPACE BAR
          SB3    K=CARR
          EQ     B2,B3,CARR       IF CARRIAGE RETURN
          GE     B2,B3,ILLCHAR    IF NOT DISPLAY CODE
          BX6    -X0*X1
          NZ     X6,DUDKBP0       IF NOT COLUMN 1
          SB3    K=PLUS
          EQ     B2,B3,PLUS       IF PLUS KEY
          SB3    K=MNUS
          EQ     B2,B3,MNUS       IF MINUS KEY
 DUDKBP0  SX6    A0
          SA6    CHAR             PUT CHARACTER IN PARAMETER
          SA1    DUDVALP
          RJ     DUDVAL
          SB1    1
          SA5    CHAR
          ZR     X5,DUDKBP4       IF RESCAN
          PL     X6,ILLCHAR       IF INVALID CHARACTER
 DUDKBP00 SA1    W=KSW
          MX0    48
          SA2    W=CNT
          SB3    X1               ADDRESS OF KEYBOARD BUFFER
          LX1    12
          BX6    -X0*X1           EXTRACT CHARACTER COUNT
          SB5    X6+B1            INCREMENT CHARACTER COUNT
          SB6    60D
          GE     B5,B6,ILLCHAR    IF MORE THAN 60 CHARACTERS
          SA3    X2+B3            PICK UP WORD IN KEYBOARD BUFFER
          SB5    X2               WORD ORDINAL IN BUFFER
          LX2    30D
          SB6    X2               BIT POSITION IN WORD
          AX3    X3,B6            POSITION WORD
          BX6    X0*X3            CLEAR OUT OLD CONTENTS
          BX7    -X0*X5           CLEAR ANY GARBAGE
          BX6    X7+X6            INSERT CHARACTER
          LX6    X6,B6            REPOSITON TO NORMAL
          SA6    A3               PUT IT BACK INTO THE BUFFER
          SB6    B6-12            UPDATE BIT POSITION
          SX5    B1               INDICATE ONE CHARACTER ADDED
          PL     B6,DUDKBP1       IF NOT END OF WORD
          SB5    B5+B1            INCREMENT WORD NUMBER
          SB6    48
 DUDKBP1  SX6    B6               PACK WORD AND BIT OFFSET TOGETHER
          SX7    B5
          LX6    30
          BX6    X6+X7
          SA6    A2               STORE W=CNT
          IX6    X5+X1            INCREMENT (OR DECREMENT) CHAR COUNT
          LX6    48               RESTORE TO ORIGINAL POSITION
          SA6    A1               STORE W=KSW
 DUDKBP11 SA1    NEXT
          ZR     X1,DUDKBP3       IF NO FORCED CHARACTERS
          NG     X1,DUDKBP2       IF TYPE-IN COMPLETE
          SX6    X1
          SA6    SAVE             SAVE NEXT CHARACTER
          SB2    30
          RJ     =XCLOCK$         DELAY 30 MILLISECONDS
          SA1    SAVE
          SA0    X1               PUT CHARACTER IN A0
          MX6    0
          SA6    A1               CLEAR SAVE
          EQ     DUDKBP0          TRY IT

 DUDKBP2  MX0    1
          LX0    1+36             POSITION TO KEYBOARD READY FLAG
          SA1    W=KSW
          BX6    X0+X1            SET READY BIT
          SA6    A1               W=KSW
 DUDKBP3  SA1    ILLCHARB
          NG     X1,ILLCHAR
          EQ     ILLCHAR1

 DUDKBP4  NG     X6,DUDKBP11      IF VALID
          MX6    59
          SA6    ILLCHARB
          EQ     DUDKBP11
**        BKSP   PROCESS BACKSPACE KEY

 BKSP     BX6    -X0*X1           EXTRACT CHARACTER COUNT
          ZR     X6,ILLCHAR       IF COLUMN 1
          LX0    48
          BX1    X1*X0            CLEAR KEYBOARD READY
          SB5    X2               WORD OFFSET
          LX2    30
          SB6    X2+12            INCREMENT BIT POINTER
```

```
         MX7    0
         SA7    NEXT
         SA7    RESET
         SA7    W=KSTAT
         SA7    W=KMAD
         SB7    60
         MX5    59              SET  CHARACTER INCREMENT = -1
         NE     B6,B7,DUDKBP1   IF NOT END OF WORD
         SB5    B5-B1           DECREMENT WORD OFFSET
         SB6    B0              SET BIT OFFSET
         EQ     DUDKBP1

**       LBLK   PROCESS LEFT BLANK KEY

 LBLK    LX1    48
         MX0    24
         BX6    -X0*X1          CLEAR CHAR COUNT AND KEYBOARD READY
         SA6    A1              STORE W=KSW
         SX6    48
         LX6    30
         SA6    A2              STORE W=CNT
         MX6    0
         SA6    RESET
         SA6    W=KSTAT
         SA6    W=KMAD
         EQ     DUDKBP3

**       RBLK   PROCESS RIGHT BLANK

 RBLK    SA3    A1              W=KSW
         LX0    36
         BX6    -X0*X3          EXTRACT TYPE-IN READY BYTE
         NZ     X6,RBLK1        IF TYPE-IN READY
         EQ     DUDKBP3

 RBLK1   LX0    24              RESTORE X0
         MX6    1
         SA5    W=KMAD          ERROR MESSAGE POINTER
         BX6    X6+X5           SET REPEAT BIT
         SA6    A5
         EQ     CARR1           CONTINUE AS IF CARRAGE RETURN

**       SPCE   PROCESS SPACE BAR

 SPCE    SA0    1R
         BX6    -X0*X1          EXTRACT CHARACTER COUNT
         NZ     X6,DUDKBP0      IF NOT COLUMN 1
         MX6    0
         SA6    B0              CLEAR PAUSE BIT
         EQ     DUDKBP3         RETURN

**       CARR   PROCESS CARRIAGE RETURN

 CARR    SA3    A1              W=KSW
         MX6    1
         SA5    W=KMAD
         BX6    -X6*X5
         SA6    A5              CLEAR REPEAT BIT
 CARR1   SA5    X3              FIRST WORD OF BUFFER
         SA4    HOLD
         BX4    X4-X5
         ZR     X4,CARR2        IF *HOLD.*
         LX1    11
         PL     X1,ILLCHAR      IF KEYBOARD NOT READY
         MX7    0
         SX6    B1
         SA7    RESET
         SA7    NEXT
         SA6    W=KSTAT         SET COMPLETE BIT
         LX0    36              7777 0000 7777 7777 7777
         BX6    X3*X0           CLEAR READY FLAG
         SA6    A3              W=KSW
         MX5    0
         EQ     DUDKBP00        STORE ZERO BYTE IN BUFFER

 CARR2   SX6    F=HOLD
         SA6    W=ISTAT
         SA1    RCL
         RJ     MTR$
 CARR3   SA5    W=KMAD
         NZ     X5,DUDKBP3      IF ERROR MESSAGE IS UP
         MX0    24
         SA1    W=KSW
         BX6    -X0*X1
         SA6    A1              W=KSW
         SX6    48
         LX6    30
         MX7    0
         SA6    W=CNT
         SA7    RESET
         EQ     DUDKBP3

**       PLUS   PROCESS PLUS KEY

 PLUS    BX6    -X0*X1          EXTRACT CHARACTER COUNT
         NZ     X6,DUDKBP0      IF NOT COLUMN 1
         MX6    59
 PLUS1   SA6    GLOBESW
         MX6    0
         SA6    RESET
         EQ     DUDKBP3

**       MNUS   PROCESS MINUS KEY

 MNUS    BX6    -X0*X1          EXTRACT CHARACTER COUNT
         NZ     X6,DUDKBP0      IF NOT COLUMN 1
         EQ     PLUS1

 ILLCHAR SA2    ILLCHARA
         NZ     X2,ILLCHAR1
         SA1    W=KMAD          SAVE ADDRESS OF CURRENT MESSAGE
         BX6    X1
         SA6    A2
         SX6    N=ILLEG         *ILLEGAL ENTRY.*
         SA6    A1
         SB2    30              DELAY 30 MSEC.
         RJ     CLOCK$
         SA1    ILLCHARA        RESTORE PREVIOUS MESSAGE
         BX6    X1
         SA6    W=KMAD
         MX6    0
         SA6    A1
         SA6    ILLCHARB
 ILLCHAR1 SB2   B0              TO EXIT FROM INTERRUPT MODE
         RJ     CLOCK$

 ILLCHARA BSSZ  1
 ILLCHARB BSSZ  1
 N=ILLEG  DATA  C*ILLEGAL ENTRY.*

*CALL     KEYCOM

         ORG    RESET
         DATA   0
         DATA   -1
         USE    *

         EXT    W=KMAD,CLOCK$
         EXT    RCP$
         EXT    MTR$
         EXT    W=KSW,W=CNT,DUDVAL
         EXT    W=ISTAT,W=A5B5,W=A4B4
         EXT    W=STAT
         EXT    W=KSTAT

 DUDVALP VFD    60/CHAR
         VFD    60/NEXT
         VFD    60/0

 CHAR    BSS    1
 NEXT    BSS    1
 SAVE    BSSZ   1

 RCL     VFD    18/3RRCL,3/2,39/W=ISTAT
 HOLD    VFD    12/1RH,12/1RO,12/1RL,12/1RD,12/1R.
 DISPLAY ENDIF
         END

ALC
         IDENT  ALC$
         SST
 DISPLAY IF     DEF,DISPLAY
         TITLE                  ALC$ - DYNAMIC USER DISPLAY CORE ALLO
,CATOR
         SPACE  2
         ENTRY  ALC$
         EXT    CLOCK$
         EXT    W=RFL,W=ARCL,W=DPL
         SPACE  2
         ENTRY  ALC$
**       ALC$ - DYNAMIC USER DISPLAY CORE ALLOCATION
*
*        AUTHOR:
*              LAWRENCE R. ATKIN
*              CONTROL DATA CORP (CGODSO)
*              01/25/70
*
*        ALC$ IS USED TO ALLOCATE CORE FOR DISPLAYS USED WITH DUD.
*
*        ENTRY INFORMATION:
*        B1     1
*        B2     DISPLAY BUFFER POINTER ADDRESS
*        X1     STORAGE REQUIRED, + IF WORDS, - IF BYTES
*        W=DPL  POINTER TO LAST DISPLAY BUFFER POINTER
*        W=DPF  POINTER TO FIRST DISPLAY BUFFER POINTER
*        W=ARCL CURRENT FIELD LENGTH IN BYTE 1
*
*        EXIT INFORMATION:
*        B2     FWA OF ALLOCATED BLOCK
*        ALL DISPLAY BUFFER POINTER WORDS HAVE BEEN UPDATED
*
 ALC$    PS     0               ENTRY/EXIT
         PL     X1,ALC1         IF REQUEST IN WORDS
         SA2    =D5.0           ELSE, CONVERT BYTE COUNT TO WORDS
         SA3    =D0.9
         BX1    -X1
         PX1    B0,X1           DIVIDE BY 5
         NX1    X1
         FX1    X1/X2
         FX1    X1+X3           ROUND UP TO NEAREST WORD
         UX1    B3,X1
         LX1    B3,X1
 ALC1    RJ     ALLOC
         SB3    X1
         SA1    W=DPL           POINTER TO LAST DBP
         SA5    X1-1            LAST DISPLAY BUFFER POINTER
         BX1    X5
         SA2    W=ARCL
         MX0    48
         LX2    24
         BX2    -X0*X2          EXTRACT CURRENT FIELD LENGTH/100B
         SA3    W=RFL           EXTRACT CURRENT REQUESTED FL/100B
         AX3    12
         BX3    -X0*X3
         IX7    X2-X3           COMPUTE B4 = MIN(X3,X2)
         AX7    59
         BX2    X2*X7
         BX3    -X7*X3
         BX2    X2+X3
         SB4    X2              SAVE IT
         LX2    6
         SX7    X1              EXTRACT FWA UNUSED
         IX7    X2-X7           UNUSED CORE
         SA2    B2              DESIRED DBP
         SX3    X2              LWA+1
         LX2    30
         SX4    X2              FWA
         IX5    X4-X3           - NO OF WORDS ASSIGNED TO THIS DBP
         SB7    X5+B3           NO OF ADDITIONAL WORDS REQUIRED
         SB6    X7              UNUSED STORAGE
         GT     B6,B7,ALC2      IF SUFFICIENT STORAGE IS AVAILABLE
         SX6    B7-B6           COMPUTE ADDITIONAL STORAGE NEEDED
         SX6    X6+100B         ROUND UP
         AX6    6               NO OF ADDITIONAL BLOCKS TO REQUEST
         SX6    X6+B4           NEW FIELD LENGTH/100B
         SX5    DUDALC$         INTERRUPT ADDRESS
         LX5    30
         LX6    12
         BX6    X6+X5           FORM STORAGE REQUEST
         SA6    W=RFL
         MX6    0
         SA6    ALC=            ALLOW CLOCK$ TO EXIT
         SB2    B0
         RJ     CLOCK$          RETURN FROM INTERRUPT MODE

 DUDALC$ SB1    1               RESTORE PARAMETERS
         SB2    0
         EQ     ALC1

 ALC2    LX2    30
         SB2    X4              RETURN B2 = FWA OF BUFFER
         MX7    0
         SA4    A0
         BX6    X4
         SA6    B5
```

```
         SA7    A0               DELETE REQUEST FROM STACK
         SA4    ALC=
         SX6    X4-1
         SA6    A4
         NG     B7,ALC3          IF CORE IS TO BE COMPRESSED
         SB5    B0-B1            DECREMENT
         SB6    X2-1             WORD AFTER LAST TO MOVE
         SB4    X1-1             FIRST WORD TO MOVE
         EQ     ALC4

ALC3     SB5    B1               INCREMENT
         SB6    X1               WORD AFTER LAST TO MOVE
         SB4    X2               FIRST WORD TO MOVE
ALC4     EQ     B4,B6,ALC6       IF NOTHING TO MOVE
         ZR     B7,ALC8          IF NOT MOVING ANYWHERE
ALC5     SA1    B4
         BX6    X1
         SA6    A1+B7            MOVE CORE
         SB4    B4+B5
         NE     B4,B6,ALC5       IF NOT DONE
ALC6     SX0    B7
         SX1    B7
         LX0    30
         IX0    X0+X1            DELTA IN BOTH ADDRESSES
         IX6    X2+X1            CHANGE LWA+1
         SA6    A2               PUT IT BACK
         SB6    A5+B1            LWA+1 DISPLAY BUFFER POINTER TABLE
ALC7     SB7    A6+B1            CURRENT DISPLAY BUFFER POINTER
         GE     B7,B6,ALC8       IF DONE
         SA1    B7
         IX6    X0+X1            MODIFY BOTH ADDRESSES
         SA6    A1
         EQ     ALC7

ALC8     SX6    B1
         LX2    9
         BX6    X6*X2
         SB6    X6               SET SCREEN IN B6
         LX2    -9
         EQ     ALC$             RETURN
ALLOC    TITLE  DETERMINE WHICH BUFFER TO ALLOCATE
**       ALLOC  DETERMINE WHICH BUFFER TO ALLOCATE
*
*        ENTRY INFORMATION:
*        B1     1
*        B2     0     IF NO NEW REQUEST
*               ELSE  DISPLAY BUFFER POINTER ADDRESS
*        X1     NUMBER OF WORDS REQUIRED
*        ALC$   RETURN ADDRESS
*
*        EXIT INFORMATION:
*        A0     ADDRESS OF LAST REQUEST IN STACK
*        B1     1
*        B2     DISPLAY BUFFER POINTER ADDRESS
*        B5     ADDRESS OF SELECTED REQUEST
*        X1     NUMBER OF WORDS REQUIRED
*        ALC$   RETURN
*        ALC=   NUMBER OF OUTSTANDING REQUESTS
*
*               IF THERE ARE TWO REQUESTS FOR THE SAME BUFFER,
*               THE OLDER REQUEST IS DISCARDED.
*
*        THE BUFFER THAT WANTS THE LEAST AMOUNT OF CORE IS ALLOCATED
*
ALLOC    PS     0
         ZR     B2,ALLOC3        IF NO NEW REQUEST
         SA2    ALLOCA
ALLOC1   SB3    X2               DBP ADDRESS
         EQ     B2,B3,ALLOC2     IF FOR SAME DBP
         ZR     B3,ALLOC2        IF NO MORE REQUESTS
         SA2    A2+B1            ADVANCE TO NEXT REQUEST WORD
         EQ     ALLOC1

ALLOC2   SA3    ALC$
         LX3    30
         SX6    X3               RETURN ADDRESS
         LX6    18
         BX6    X6+X1            NUMBER OF WORDS
         LX6    18
         SX3    B2
         BX6    X3+X6            DBP ADDRESS
         SA6    A2               INTO STACK
ALLOC3   SB3    377777B          INITIALIZE
         SA2    ALLOCA
ALLOC4   ZR     X2,ALLOC6        IF DONE
         LX2    60-18
         SB4    X2
         GE     B4,B3,ALLOC5     IF NOT BETTER
         SB3    B4
         SB5    A2               SAVE REQUEST ADDRESS
         BX6    X2               SAVE REQUEST WORD
ALLOC5   SA2    A2+B1
         EQ     ALLOC4

ALLOC6   SX7    A2-ALLOCA
         SA7    ALC=             NUMBER OF OUTSTANDING REQUESTS
         NZ     X7,ALLOC7        IF MORE TO DO
         SB2    B0
         RJ     CLOCK$           RETURN FROM INTERRUPT MODE

ALLOC7   SX1    X6               NUMBER OF WORDS
         LX6    18
         SB2    X6               DBP ADDRESS
         AX6    6
         MX7    1
         LX7    57               0400 0000 0000 0000 0000
         BX6    X6+X7            FORM EQ TO RETURN
         SA6    ALC$
         SA0    A2-B1            ADDRESS OF LAST REQUEST IN STACK
         EQ     ALLOC            RETURN

         BSSZ   1
ALLOCA   BSSZ   5
ALC=     BSSZ   1
         ENTRY  DUDALC$,ALC=
ALCHOP$  TITLE  CHOP FIELD LENGTH TO REQUIRED SIZE
         ENTRY  ALCHOP$
         SPACE  4
ALCHOP$  PS     0
         SA1    ALLOCA
         NZ     X1,ALCHOP$       IF REQUESTS PENDING, RETURN
         SA2    W=RFL
         AX2    12
         MX0    48
         BX2    -X0*X2
         SA1    W=DPL
         SA1    X1-1
         SX6    X1+100B          ROUND UP
         AX6    6
         BX7    X2-X6
         ZR     X7,ALCHOP$       IF FL IS RIGHT ALREADY
         LX6    12
         SA6    A2               W=RFL
         EQ     ALCHOP$


 DISPLAY ENDIF
         END
CLOCK
         IDENT  CLOCK$
         SST
 DISPLAY IF     DEF,DISPLAY
         SPACE  4
**       CLOCK$ - REAL TIME INTERVAL TIMER
*
*        AUTHOR:
*               K. E. GORLEN
*               VOGELBACK COMPUTING CENTER
*               NORTHWESTERN UNIVERSITY
*
*        MODE:  INTERRUPT
*
*        ENTER: B2 = DELAY TIME IN MILLISECONDS (377777B MAX.)
*
*        EXIT:  B2 = REQUESTED DELAY TIME MINUS ELAPSED REAL TIME
*
*        - CLOCK$ EXITS FROM INTERRUPT MODE, HENCE, ALL REGISTERS ARE
*        DESTROYED
*        - ANY INTERRUPT ROUTINE USING CLOCK$ MUST EXIT FROM INTERRUPT
*        MODE BY CALLING CLOCK$ WITH B2 = 0.
*
*        CALLS: RCP$

         ENTRY  CLOCK$,DUDCLK$
         EXT    W=MSC,RCP$
         EXT    ALC=,DUDALC$

 CLOCK$  PS
         EQ     B0,B2,DUDCLK$    IF NO NEW ENTRY TO ADD
         RJ     TIME             UPDATE STACK DELAYS
         SA1    CLOCK$           FORM NEW STACK ENTRY
         GT     B2,B0,CLOCK1     IF DELAY POSITIVE
         SB2    B0               ELSE, RESET TO ZERO
 CLOCK1  SX2    B2
         BX6    X1+X2
         SA6    B3+STACK         ADD TO END OF STACK
         SX7    B3+B1            INCREMENT STACK ENTRY COUNT
         SA7    COUNT
         SA5    NEXT
         EQ     B3,B0,CLOCK2     IF ONLY 1 ENTRY IN STACK
         SA4    X5
         SB4    X4
         GE     B2,B4,DUDCLK$    IF NEW DELAY NOT SHORTEST IN STACK
 CLOCK2  SX6    A6               ELSE, RESET NEXT
         SA6    A5

**       DUDCLK$ - INTERRUPT ENTRY POINT

 DUDCLK$ RJ     TIME             UPDATE STACK DELAY
         SB2    B0
         EQ     B3,B0,DUDCLK1    IF STACK EMPTY
         SA5    NEXT
         SA4    X5
         SB2    X4               B2 = SHORTEST DELAY IN STACK
         GE     B0,B2,DUDCLK2    IF TIME TO EXECUTE
         SB4    7777B
         GE     B4,B2,DUDCLK1    SENSE SHORTEST DELAY GT 7777B
         SB2    B4               YES, SO SET DELAY OF 7777B MSEC.
 DUDCLK1 SA1    ALC=             ALLOCATOR STACK COUNT
         NZ     X1,DUDALC$       IF ALLOCATOR NEEDS THE CP
         RJ     RCP$             ELSE EXIT INTERRUPT MODE

 DUDCLK2 LX4    30
         SB6    X4               B6 = RETURN ADDRESS
         SA1    B3+STACK-1       DELETE ENTRY FROM STACK
         BX6    X1               BY SUBSTITUTING LAST STACK ENTRY
         SA6    X5
         SX6    B3-B1            AND DECREMENT ENTRY COUNT
         SA6    COUNT
         EQ     B3,B1,DUDCLK5    IF STACK NOW EMPTY
         SB4    377777B          ELSE, SEARCH STACK FOR SHORTEST DELAY
 DUDCLK3 SA1    B3+STACK-2
         SB5    X1
         LT     B4,B5,DUDCLK4
         SX6    A1               SAVE ADDRESS OF ENTRY
         SB4    B5               AND NEW MINIMUM DELAY
 DUDCLK4 SB3    B3-B1
         NE     B3,B1,DUDCLK3    LOOP THROUGH STACK
         SA6    A5               UPDATE NEXT
 DUDCLK5 JP     B6               JUMP TO RETURN ADDRESS

*        TIME - ROUTINE TO COMPUTE ELAPSED TIME AND UPDATE DELAY STACK

 TIME    PS
         SB1    1                B1 = CONSTANT 1
         SB7    1000D            B7 = CONSTANT 1000D
         SA1    COUNT
         SA2    W=MSC            READ CLOCK
         SB3    X1               B3 = STACK ENTRY COUNT
         SB4    B3               B4 = STACK ENTRY COUNT
         MX0    48
         LX2    60-24
         SA1    START            X1 = START TIME
         BX6    X2               X2 = CURRENT TIME
         SA6    A1               SET START TIME = CURRENT TIME
         EQ     B0,B3,TIME       IF STACK EMPTY
         BX3    -X0*X1           X3 = START MSEC.
         BX4    -X0*X2           X4 = CURRENT MSEC.
         LX1    60-12
         BX1    -X0*X1           X1 = START SEC.
         LX2    60-12
         BX2    -X0*X2           X2 = CURRENT SEC.
         IX1    X2-X1            COMPUTE ELAPSED SECONDS
         PL     X1,TIME1         IF NO WRAP
         SX1    X1+B7            TO COMPENSATE FOR WRAP
 TIME1   IX4    X4-X3            COMPUTE ELAPSED MSEC.
         PL     X4,TIME2         IF NO WRAP
         SX1    X1-1             TO COMPENSATE FOR WRAP
         SX4    X4+B7
 TIME2   BX2    X1               CONVERT SECONDS TO MILLISECONDS
         LX1    10               *2000B
         LX2    3                *10B
         LX3    X2,B1            *20B
         IX1    X1-X2            *1770B
```

```
          IX1    X1-X3             *1750B = *1000D
          IX1    X1+X4             X1 = TOTAL ELAPSED MSEC.
          NG     X1,TIME           IF NEGATIVE ELAPSED TIME
          MX0    42
 TIME3    SA2    B4+STACK-1        SUBTRACT ELAPSED TIME FROM STACK
          BX6    X0*X2             SAVE JUMP ADDRESS
          BX2    -X0*X2            ISOLATE DELAY TIME
          IX2    X2-X1             SUBTRACT ELAPSED TIME
          BX2    -X0*X2            STIP OFF CARRY
          BX6    X6+X2             FORM NEW STACK ENTRY
          SA6    A2                RESTORE ENTRY
          SB4    B4-B1
          NE     B4,B0,TIME3       LOOP THROUGH STACK
          EQ     B0,B0,TIME        RETURN

 START    BSSZ   1
 COUNT    BSSZ   1
 NEXT     BSSZ   1
 STACK    BSSZ   10
 DISPLAY  ENDIF
          END
RCP
          IDENT  RCP$
          SST
 DISPLAY  IF     DEF,DISPLAY
          SPACE  4
**        RCP$ - EXIT FROM INTERRUPT ROUTINE
*
*         MODE:  INTERRUPT
*
*         ENTER: B2 = CLOCK INTERRUPT DELAY

          ENTRY  RCP$
          EXT    W=ISTAT,W=A4B4
*CALL     DUDCOM
          SPACE  4
 RCP$     PS
          SB1    1
          SA4    W=A4B4            RESTORE REGISTERS A4,B4,A5,B5
          SA5    A4+B1
          SB4    X4
          SB5    X5
          LX4    60-18
          SA4    X4
          LX5    60-18
          SA5    X5
          SX1    B2                FORM RCP REQUEST TO DUD
          LX1    12
          SX6    F=RCP
          BX6    X1+X6             DELAY TIME + FUNCTION CODE
          SA2    RCPA
          BX7    X2                SET AUTO-RECALL REQUEST
 RCP1     SA1    B1                WAIT FOR RA+1 TO CLEAR
          NZ     X1,RCP1
+         SA6    W=ISTAT           ISSUE RCP FUNCTION
          SA7    B1                ISSUE AUTO-RECALL REQUEST
 RCP2     EQ     RCP2

 RCPA     VFD    18/3RRCL,3/2,39/W=ISTAT
 DISPLAY  ENDIF
          END
CHESBRD
          IDENT  CHESBRD
          SST
 DISPLAY  IF     DEF,DISPLAY
          TITLE                    CHESS BOARD DISPLAY
          ENTRY  BOARD$,BOARD.
          LIST   L,-R
          SPACE  4
**        CHESBRD - CHESS BOARD DISPLAY
*
*         AUTHOR:
*               K. E. GORLEN
*               VOGELBACK COMPUTING CENTER
*               NORTHWESTERN UNIVERSITY
*CALL,CHARMAC
          TITLE                    CHESS BOARD
**        THE CHESS BOARD
          SPACE  2
 X        SET    60B
 Y        SET    110B
          SPACE  2
 BOARD$   BSS    0

 1        DUP    4
 Y        SET    Y+10B
 Y        IFEQ   Y,120B
          CHAR   X,Y,------------------------
 Y        ELSE
          CHAR   X-4B,Y,(I  I  I  I  I  I  I  I)
          CHAR   X,!,------------------------
 Y        ENDIF
          CHAR   X+10B,!,------------------------
 2        DUP    5
 Y        SET    Y+10B
          CHAR   X-4B,Y,(I  I//I  I//I  I//I  I//I)
 2        ENDD
 Y        SET    Y+10B
          CHAR   X-4B,Y,(I  I  I  I  I  I  I  I)
          CHAR   X,!,------------------------
          CHAR   X+10B,!,------------------------
 2        DUP    5
 Y        SET    Y+10B
          CHAR   X-4B,Y,(I//I  I//I  I//I  I//I  I)
 2        ENDD
 1        ENDD
 Y        SET    Y+10B
          CHAR   X,Y,------------------------
          CHAR   X+10B,!,------------------------
          FILL

 BOARD.   BSS    0

 DISPLAY  ENDIF
          END
CHESMEN
          IDENT CHESMEN
          SST
 DISPLAY  IF     DEF,DISPLAY
          SST
 DISPLAY  IF     DEF,DISPLAY
          TITLE CHESMEN --- CONSOLE DISPLAY CHESS MEN
          LIST   -R,-G
          ENTRY  CHESMEN,CHESLEN
          SPACE  4
***       CHESMEN --- CONSOLE DISPLAY CHESS MEN
*
*         ARTISTS:
*
*               L. R. ATKIN
*               C. G. FILSTEAD
*               K. E. GORLEN
*               12/27/68
*
          TITLE                    DATA GENERATION MACROS
**        COORD - X-Y COORDINATE GENERATION MACRO
          SPACE  2
 COORD    MACRO  X,P,Y
 III      SET    III+1
 JJJ      SET    1
          VFD    12/60!X!B
          DUP    100
 XXX      MICRO  JJJ,2,Y
 XXX      IFC    NE,*''XXX'**
 III      SET    III+1
 JJJ      SET    JJJ+2
          VFD    12/70'XXX'B
 XXX      ELSE
          STOPDUP
 XXX      ENDIF
          ENDD
 P        IFC    NE,$P$$
 III      SET    III+1
 JJJ      SET    1
          VFD    12/6060B-X!B
          DUP    100
 XXX      MICRO  JJJ,2,Y
 XXX      IFC    NE,*''XXX'**
 III      SET    III+1
 JJJ      SET    JJJ+2
          VFD    12/70'XXX'B
 XXX      ELSE
          STOPDUP
 XXX      ENDIF
 P        ENDIF
          ENDD
          ENDM
          SPACE  2
**        ENDP - END-OF-PIECE MACRO
          SPACE  2
 ENDP     MACRO
          LOCAL  KKK
 KKK      EQU    III
 BYTES    MICRO  1,,*'BYTES'KKK*
 III      SET    0
          VFD    12/0
 NNN      SET    $+1
          IFNE   NNN,60,1
          VFD    NNN/0
          ENDM
          SPACE  2
 III      SET    0
 BYTES    MICRO  1,,
          TITLE                    POINTERS TO PIECES
**        POINTERS TO PIECES
          SPACE  4
 CHESMEN  VFD    60/BKING
          VFD    60/BQUEEN
          VFD    60/BBSHP
          VFD    60/BNITE
          VFD    60/BROOK
          VFD    60/BPAWN
          VFD    60/*
          VFD    60/WPAWN
          VFD    60/WROOK
          VFD    60/WNITE
          VFD    60/WBSHP
          VFD    60/WQUEEN
          VFD    60/WKING
          TITLE                    DISPLAY COORDINATES OF CHESS MEN
**        DISPLAY COORDINATES OF CHESS MEN
          SPACE  4
 BKING    COORD  17,*,$12131427303132$
          COORD  20,*,$12152533$
          COORD  21,*,$122434$
          COORD  22,*,$121623$
          COORD  23,*,$1220212235$
          COORD  24,*,$1235$
          COORD  25,*,$1235374041$
          COORD  26,*,$1235374041$
          COORD  27,*,$12353637414243$
          COORD  30,!,$1243$
          ENDP
          SPACE  2
 BQUEEN   COORD  17,*,$124143$
          COORD  20,*,$12133537741$
          COORD  21,*,$1214313340$
          COORD  22,*,$1215252737$
          COORD  23,*,$1216212335$
          COORD  24,*,$121734$
          COORD  25,*,$1233$
          COORD  26,*,$1235$
          COORD  27,*,$123741$
          COORD  30,!,$1243$
          ENDP
          SPACE  2
 BBSHP    COORD  22,!,$12$
          COORD  23,!,$121327303132$
          COORD  24,!,$12142533$
          COORD  25,!,$12152434$
          COORD  26,!,$12161720212223335$
          COORD  27,!,$1237$
          COORD  30,!,$12313241$
          COORD  31,!,$1232333437$
          COORD  32,!,$121617202122233334$
          COORD  33,!,$12152434$
          COORD  34,!,$12142533$
          COORD  35,!,$121327303132$
          COORD  36,!,$12$
          ENDP
          SPACE  2
 BNITE    COORD  17,!,$25$
          COORD  20,!,$2326$
          COORD  21,!,$26$
          COORD  22,!,$122227$
          COORD  23,!,$12132230$
          COORD  24,!,$12142331$
          COORD  25,!,$1216172432$
          COORD  26,!,$122021222333$
          COORD  27,!,$1234$
          COORD  30,!,$123234$
          COORD  31,!,$1235$
          COORD  32,!,$1236$
          COORD  33,!,$1235$
```

```
         COORD  34,!,$121415161734$
         COORD  35,!,$12132033$
         COORD  36,!,$12222331$
         COORD  37,!,$252627$
         ENDP
         SPACE  2
 BROOK   COORD  20,*,$122630323334$
         COORD  21,*,$12132534$
         COORD  22,*,$121434$
         COORD  23,*,$1214243031323334$
         COORD  24,*,$1216172021222330$
         COORD  25,*,$1230$
         COORD  26,*,$123031323334$
         COORD  27,*,$1234$
         COORD  30,!,$1234$
         ENDP
         SPACE  2
 BPAWN   COORD  24,*,$12$
         COORD  25,*,$1214262730$
         COORD  26,*,$121625$
         COORD  27,*,$12202122232431$
         COORD  30,!,$1231$
         ENDP
         SPACE  2
 WPAWN   COORD  24,*,$12$
         COORD  25,*,$121314262730$
         COORD  26,*,$12131415162526273O$
         COORD  27,*,$1213141516172021222324252627303 1$
         COORD  30,!,$1213141516172021222324252627031$
         ENDP
         SPACE  2
 WROOK   COORD  20,*,$122630323334$
         COORD  21,*,$12132526273031323334$
         COORD  22,*,$1213142526273031323334$
         COORD  23,*,$121314152425262730313233334$
         COORD  24,*,$12131415161720212223242526273O$
         COORD  25,*,$121314151617202122232425262730$
         COORD  26,*,$12131415161720212223242526273031323334$
         COORD  27,*,$121314151617202122232425262730313233334$
         COORD  30,!,$12131415161720212223242526273031323334$
         ENDP
         SPACE  2
 WNITE   COORD  17,!,$25$
         COORD  20,!,$23242526$
         COORD  21,!,$2223242526$
         COORD  22,!,$12222324252627$
         COORD  23,!,$1213232425262730$
         COORD  24,!,$12131423242526273031$
         COORD  25,!,$121314151617242526273031323 2$
         COORD  26,!,$12131415161720212223242526273031323333$
         COORD  27,!,$1213141516172021222324252627303132333435$
         COORD  30,!,$121314151617202122232425262730313233334$
         COORD  31,!,$12131415161720212223242526273031323333435$
         COORD  32,!,$121314151617202122232425262730313233334536$
         COORD  33,!,$12131415161720212223242526273031323333435$
         COORD  34,!,$1213141516172021222324252627303132333435$
         COORD  35,!,$121320212223242526273031323 3$
         COORD  36,!,$1223222425262730 31$
         COORD  37,!,$252627$
         ENDP
         SPACE  2
 WBSHP   COORD  22,!,$12$
         COORD  23,!,$121327303132$
         COORD  24,!,$12131425262730313233$
         COORD  25,!,$121314152425262730313233334$
         COORD  26,!,$12131415161720212223242526273031323333435$
         COORD  27,!,$121314151617202122232425262730313233334353637$
         COORD  30,!,$12131415161720212223242526273033343536374041$
         COORD  31,!,$121314151620212223242526273031353637$
         COORD  32,!,$12131415161720212223242526273031323 5$
         COORD  33,!,$121314152425262730313233334$
         COORD  34,!,$12131425262730313233$
         COORD  35,!,$121327303132$
         COORD  36,!,$12$
         ENDP
         SPACE  2
 WQUEEN  COORD  17,*,$124143$
         COORD  20,*,$121335374041$
         COORD  21,*,$12131431333435363740$
         COORD  22,*,$121314152527303132333435363 7$
         COORD  23,*,$12131415162123242526273031323334 35$
         COORD  24,*,$121314151617202122232425262730313233334$
         COORD  25,*,$12131415161720212223242526273031323 3$
         COORD  26,*,$121314151617202122232425262730313233334 35$
         COORD  27,*,$12131415161720212223242526273031323333435363741$
         COORD  30,!,$12131415161720212223242526273031323333435363741424
,3$
         ENDP
         SPACE  2
 WKING   COORD  17,*,$12131427303132$
         COORD  20,*,$121314152526273031323 3$
         COORD  21,*,$121314152425262730313233334$
         COORD  22,*,$121314151623242526273031323334$
         COORD  23,*,$12131415161720212223242526273031323334 35$
         COORD  24,*,$121314151617202122232425262730313233334 35$
         COORD  25,*,$12131415161720212223242526273031323333435374041$
         COORD  26,*,$121314151617202122232425262730313233334353740 41$
         COORD  27,*,$12131415161720212223242526273031323333435363740414
,243$
         COORD  30,!,$12131415161720212223242526273031323333435363740414
,243$
         ENDP
         TITLE                    PIECE LENGTH TABLE
**       PIECE LENGTH TABLE
         LIST   G
         SPACE  4
 CHESLEN BSS    0

 NNN     SET    1
         DUP    6
 XXX     MICRO  NNN,8,*'BYTES'*
         VFD    60/'XXX'
 NNN     SET    NNN+8
         ENDD

         VFD    60/0

         DUP    6
 XXX     MICRO  NNN,8,*'BYTES'*
         VFD    60/'XXX'
 NNN     SET    NNN+8
         ENDD
 DISPLAY ENDIF
 DISPLAY ENDIF
         END
MTR
         IDENT  MTR$
         SST
 DISPLAY IF     DEF,DISPLAY
         SPACE  4
**       MTR$ - ISSUE MONITOR REQUEST
*
*        MODE:  INTERRUPT
*
*        ENTER: B1 = 1
*               X1 = MONITOR REQUEST
*
*        USES:  A1,A6,X1,X6

         ENTRY  MTR$

 MTR$    PS
         BX6    X1
 MTR1    SA1    B1               WAIT FOR RA+1 TO CLEAR
         NZ     X1,MTR1
         SA6    B1               ISSUE REQUEST
 MTR2    SA1    B1
         NZ     X1,MTR2          WAIT FOR RA+1 TO CLEAR
         EQ     MTR$
 DISPLAY ENDIF
         END
CHESS12
         IDENT  CHE$$12          DISPLAY INITIALIZATION
         TITLE  CHE$$12 - INITIALIZATION AND DISPLAY OF INITIAL BOARD P
         SST
 IMLAC   IF     DEF,IMLAC
***      DUDLOAD - INITIALIZATION AND DISPLAY OF INITIAL BOARD POSITION
*
         SPACE  4
 12      THIS                    DEFINE THIS OVERLAY
         ENTRY  CHE$$12
         ADDLIBS GYPSY
         NOREF  GIIV,GILINR,XX0,Y0,GISGR
         TITLE  DATA STORAGE
         USE    //
 A       BSS    1                FOR GYPSY

*CALL  USEIML

         USE    0
         TITLE  SYMBOL DEFINITIONS AND MICROS
***      SYMBOL DEFINITIONS
*
         SPACE  4
 BKING   EQU    40B-6
 BQUEEN  EQU    40B-5
 BBSHP   EQU    40B-4
 BNITE   EQU    40B-3
 BROOK   EQU    40B-2
 BPAWN   EQU    40B-1
 EMPTY   EQU    40B
 WPAWN   EQU    40B+1
 WROOK   EQU    40B+2
 WNITE   EQU    40B+3
 WBSHP   EQU    40B+4
 WQUEEN  EQU    40B+5
 WKING   EQU    40B+6
 BDX0    EQU    128              X-COORD OF BOTTOM LEFT SQUARE OF BRD
 BDY0    EQU    160              Y-COORD OF BOTTOM LEFT SQUARE OF BRD
 N       EQU    96               NO OF DOTS PER SQUARE
 M       EQU    16               NO OF DOTS BETWEEN LINES OF WHITE SQ
         SPACE  4
***      MICROS
*
         SPACE  4
         ECHO   ,I=(0,1,2,3,4,5,6,7,8)
 BDX!I   DECMIC BDX0+I*N
 BDY!I   DECMIC BDY0+I*N
 BDYY!I  DECMIC BDY0+I*N+48
         ECHO   ,J=(1,2,3,4,5)
 BDXX!I!J DECMIC BDX0+I*N+J*M
         ENDD
         SPACE  4
*CALL    VERSION
         TITLE  MACROS - CALL, COORD, ENDP, AND LINK
*CALL CALL
         SPACE  4
***      COORD - PIECE GENERATION MACRO
*                GENERATES THE GYPSY CALLS TO GENERATE A BLOCK THAT
*                DRAWS A PIECE.  A CALL IS OF THE FORM:
*                     PIECE   COORD   XCOORD,CHAR,YCOORDS
*                WHERE PIECE IS THE PIECE NAME AT THE FIRST CALL FOR
*                THIS PIECE, AND OTHERWISE NULL, XCOORD IS A POSITIVE
*                OCTAL X-COORDINATE60, CHAR IS EITHER * OR NULL, AND
*                YCOORDS IS A STRING OF POSITIVE TWO-DIGIT Y-COORDINATES
*                0 DELIMITED BY * SIGNS.  IF CHAR IS AN *, MIRROR
*                IMAGE COORDINATES ABOUT THE LINE X=30B ARE DRAWN.
         SPACE  4
         MACRO  COORD,LABEL,XS,P,YS
         NOREF  X,Y,JJJ,DX,DY,ADX,ADY,XY,YB
 X       SET    XS!B
 LAB     IFC    NE, LABEL
 LABELM  DECMIC LABEL
         CALL   GCNAME,(='LABELM')  GENERATE BLOCK FOR PIECE
         CALL   GISCAL,(=2,=0)
 XX0     SET    30B
 Y0      SET    30B              CURRENT BEAM POSITION (RELATIVE)
 LAB     ENDIF
 LOOP    DUP    2                FOR MIRROR-IMAGE POINTS
 JJJ     SET    1                POINTER TO Y-COORD
 XXX     MICRO  1,2,YS
 Y       SET    'XXX'B           GET FIRST Y-COORD
 LOOP1   DUP    100         FOR POSITIONING BEAM BETWEEN NONCONSEC PTS
 DX      SET    X-XX0
 DY      SET    Y-Y0
 INC     IFLT   DX,0
 ADX     SET    -DX              ABSOLUTE VALUE OF DX
 SDX     MICRO  1,1,*-*          SIGN
 INC     ELSE
 ADX     SET    DX
 SDX     MICRO                   SIGN
 INC     ENDIF
 INC     IFLT   DY,0
 ADY     SET    -DY              ABSOLUTE VALUE OF DY
 SDY     MICRO  1,1,*-*          SIGN
 INC     ELSE
 ADY     SET    DY
 SDY     MICRO                   SIGN
 INC     ENDIF
 XY      MAX    ADX,ADY
 ADX     DECMIC ADX,2
 ADY     DECMIC ADY,2
 INC     IFGT   XY,3
         CALL   GILINR,(='SDX''ADX',='SDY''ADY',=0,=0)   MOVE BEAM
```

```
 INC      ELSE
          CALL   GIIV,(='SDX''ADX',='SDY''ADY',=0)      MOVE BEAM
 INC      ENDIF
 XX0      SET    X
 Y0       SET    Y
 YB       SET    0                NO OF CONSECUTIVE PTS
 LOOP2    DUP    50               FOR DRAWING Y-VECTORS OF CONSEC PTS
 JJJ      SET    JJJ+2            NEXT Y-COORD
 XXX      MICRO  JJJ,2,YS
 YPOS2    IFC    NE, 'XXX'
 Y        SET    'XXX'B           PICK UP NEXT Y-COORD
 YPOS2    ELSE
 Y        SET    77B              END OF STRING FLAG
 YPOS2    ENDIF
 DELY     IFEQ   Y-Y0-YB,1        IF Y S ARE CONSECUTIVE
 YB       SET    YB+1
 DELY     ELSE
          STOPDUP
 YB       DECMIC YB,2
 YPOS3    IFGT   YB,3
          CALL   GILINR,(=0,='YB',=1,=0)   DRAW LONG VECTOR
 YPOS3    ELSE
          CALL   GIIV,(=0,='YB',=1)   DRAW SHORT VECTOR
 YPOS3    ENDIF
 Y0       SET    Y0+YB            UPDATE Y0
 DELY     ENDIF
 LOOP2    ENDD
 YPOS4    IFEQ   Y,77B
          STOPDUP                 END OF Y-COORD STRING
 YPOS4    ENDIF
 LOOP1    ENDD
 CHAR     IFC    EQ, P
          STOPDUP
 CHAR     ELSE
 X        SET    60B-X            REDO FOR MIRROR IMAGE POINTS
 CHAR     ENDIF
 LOOP     ENDD
          ENDM
          SPACE  4
***       ENDP - END OF PIECE MACRO
*
          SPACE  4
 ENDP     MACRO
          NOREF  SY
 XX0      DECMIC 110B-XX0,2
 Y0       SET    30B-Y0
 ENDP     IFLT   Y0,0
 Y0       SET    -Y0
 SY       MICRO  1,1,*-*
 ENDP     ELSE
 SY       MICRO
 ENDP     ENDIF
 Y0       DECMIC Y0,2
          CALL GILINR,(='XX0',='SY''Y0',=0,=0)  POSITION BEAM AT RIGHT
          CALL   GCENDI           END INSERTION MODE
          ENDM
          SPACE  4
***       LINK - MACRO FOR (1,2) OVERLAY
*
          SPACE  4
          MACRO  LINK,ENTRY
 ENTRIE   MICRO  1,6 ENTRY
+         EQ     =X'ENTRIE'.
-         VFD    30/=X!ENTRY+1
          ENDM
          TITLE
 LOADA    BSSZ   1                FLAG=0 IF NEVER BEEN CALLED
 CHE$$12  BX6    X5
          SA6    SCRATCH          SAVE X5
          SA1    LOADA            FLAG
          NZ     X1,LOAD7         CALLED BEFORE
          SX6    1                FIRST TIME CALLED
          SA6    A1               MAKE FLAG NON-ZERO
          SA1    =XOUTPUT
          BX6    X1
          SA6    OUT
          SX7    B0
          SA7    A1               ERASE CONSOL FROM CHESS MEMORY
          SA1    DUDLINK
 DUD      BX6    X1
          SA6    X6
          SA1    A1+B1
          NZ     X1,DUD
          CALL   GCINI,(A,=0)     INITIALIZE GYPSY

          CALL   GCNAME,(=40B)    CREATE BLOCK FOR EMPTY SQUARE
          CALL   GILINR,(=60B,=0,=0,=0)   MOVE BEAM TO RIGHT
          CALL   GCENDI           END BLOCK

          CALL   GCNAME,(=4RTPIN)   CREATE BLOCK FOR DISPLING TYPEIN BOX
          CALL   GIPOSA,(=876,=65,=0)
          CALL   GITEXT,(=6LTYPEIN,=0)   DISPLAY TYPEIN
          CALL   GIPOSY,(=59,=0)  DRAW BOX AROUND IT
          CALL   GISGR,(=1,=0)    PUT ON SUPPRESSED GRID MODE
          CALL   GIPOSA,(=872,=90,=0)
          CALL   GIPOSA,(=998,=59,=0)
          CALL   GISGR,(=0,=0)    TURN SUPPRESSED GRID OFF
          CALL   GCENDI

 BKING    COORD  17,!,$27303132$
          CALL   GIIV,(=2,=2,=1)
          CALL   GIIV,(=2,=1,=1)
          CALL   GILINR,(=4,=0,=1,=0)
          CALL   GIIV,(=0,=2,=1)
          CALL   GIIV,(=-2,=0,=1)
          CALL   GIIV,(=0,=2,=1)
          CALL   GIIV,(=2,=0,=1)
          CALL   GIIV,(=0,=2,=1)
          CALL   GIIV,(=2,=0,=1)
          CALL   GIIV,(=0,=-2,=1)
          CALL   GIIV,(=2,=0,=1)
          CALL   GIIV,(=0,=-2,=1)
          CALL   GIIV,(=-2,=0,=1)
          CALL   GIIV,(=0,=-2,=1)
          CALL   GILINR,(=4,=0,=1,=0)
          CALL   GIIV,(=2,=-1,=1)
          CALL   GIIV,(=2,=-2,=1)
          CALL   GIIV,(=0,=-3,=1)
          CALL   GIIV,(=-1,=-2,=1)
          CALL   GIIV,(=-3,=-3,=1)
          CALL   GIIV,(=0,=-3,=1)
          CALL   GIIV,(=1,=-1,=1)
          CALL   GIIV,(=2,=-1,=1)
          CALL   GIIV,(=1,=-3,=1)
          CALL   GILINR,(=-22B,=0,=1,=0)
          CALL   GIIV,(=1,=3,=1)
          CALL   GIIV,(=2,=1,=1)
          CALL   GIIV,(=1,=1,=1)
          CALL   GIIV,(=0,=3,=1)
          CALL   GIIV,(=-3,=3,=1)
          CALL   GIIV,(=-1,=2,=1)
          CALL   GILINR,(=71B,=1,=0,=0)
          CALL   GCENDI
          SPACE  2
 BQUEENM  DECMIC BQUEEN
          CALL   GCNAME,(='BQUEENM')  GENERATE BLOCK FOR QUEEN
          CALL   GISCAL,(=2,=0)
          CALL   GIIV,(=3,=3,=0)
          CALL   GILINR,(=-3,=10B,=1,=0)
          CALL   GILINR,(=-3,=-10B,=1,=0)
          CALL   GILINR,(=-6,=10B,=1,=0)
          CALL   GILINR,(=5,=-24B,=1,=0)
          CALL   GILINR,(=-5,=-5,=1,=0)
          CALL   GILINR,(=22B,=0,=1,=0)
          CALL   GILINR,(=-5,=5,=1,=0)
          CALL   GILINR,(=5,=24B,=1,=0)
          CALL   GILINR,(=-6,=-10B,=1,=0)
          CALL   GILINR,(=55B,=-3,=0,=0)
          CALL   GCENDI
          SPACE  2
 BBSHP    COORD  26,!,$161720212223$
          CALL   GIIV,(=-2,=2,=1)
          CALL   GIIV,(=-1,=2,=1)
          CALL   GIIV,(=0,=3,=1)
          CALL   GIIV,(=3,=3,=1)
          CALL   GILINR,(=2,=4,=1,=0)
          CALL   GILINR,(=2,=-4,=1,=0)
          CALL   GIIV,(=3,=-3,=1)
          CALL   GIIV,(=0,=-3,=1)
          CALL   GIIV,(=-1,=-2,=1)
          CALL   GIIV,(=-2,=-2,=1)
          CALL   GILINR,(=0,=-5,=1,=0)
          CALL   GILINR,(=4,=-4,=1,=0)
          CALL   GILINR,(=-14B,=0,=1,=0)
          CALL   GILINR,(=4,=4,=1,=0)
          CALL   GILINR,(=4,=17B,=0,=0)
          CALL   GIIV,(=0,=-2,=1)
          CALL   GIIV,(=-1,=1,=0)
          CALL   GIIV,(=0,=-2,=1)
          CALL   GIIV,(=-1,=0,=0)
          CALL   GIIV,(=0,=-1,=1)
          CALL   GILINR,(=60B,=-1,=0,=0)
          CALL   GCENDI
          SPACE  2
 BNITE    COORD  30,!,$32$
          CALL   GIIV,(=0,=2,=0)
          CALL   GIIV,(=2,=2,=1)
          CALL   GIIV,(=3,=-3,=1)
          CALL   GILINR,(=2,=-4,=1,=0)
          CALL   GIIV,(=0,=-2,=1)
          CALL   GILINR,(=-3,=-6,=1,=0)
          CALL   GIIV,(=0,=-3,=1)
          CALL   GIIV,(=2,=-2,=1)
          CALL   GILINR,(=-14B,=0,=1,=0)
          CALL   GIIV,(=2,=2,=1)
          CALL   GILINR,(=2,=4,=1,=0)
          CALL   GILINR,(=-1,=4,=1,=0)
          CALL   GIIV,(=-2,=-2,=1)
          CALL   GIIV,(=-1,=0,=1)
          CALL   GIIV,(=-2,=1,=1)
          CALL   GIIV,(=-1,=2,=1)
          CALL   GIIV,(=1,=1,=1)
          CALL   GIIV,(=1,=0,=1)
          CALL   GILINR,(=6,=6,=1,=0)
          CALL   GIIV,(=1,=0,=1)
          CALL   GILINR,(=60B,=-4,=0,=0)
          CALL   GCENDI
          SPACE  2
 BROOK    COORD  26,!,$3031323334$
          CALL   GILINR,(=4,=0,=1,=0)
          CALL   GILINR,(=0,=-4,=1,=0)
          CALL   GIIV,(=3,=0,=1)
          CALL   GILINR,(=0,=4,=1,=0)
          CALL   GIIV,(=3,=0,=1)
          CALL   GILINR,(=0,=-6,=1,=0)
          CALL   GIIV,(=-1,=-1,=1)
          CALL   GIIV,(=-2,=-1,=1)
          CALL   GIIV,(=-1,=0,=1)
          CALL   GILINR,(=0,=-6,=1,=0)
          CALL   GILINR,(=4,=-4,=1,=0)
          CALL   GILINR,(=-20B,=0,=1,=0)
          CALL   GILINR,(=4,=4,=1,=0)
          CALL   GILINR,(=0,=6,=1,=0)
          CALL   GIIV,(=-1,=0,=1)
          CALL   GIIV,(=-2,=1,=1)
          CALL   GIIV,(=-1,=1,=1)
          CALL   GILINR,(=0,=6,=1,=0)
          CALL   GIIV,(=3,=0,=1)
          CALL   GILINR,(=0,=-4,=1,=0)
          CALL   GIIV,(=3,=0,=1)
          CALL   GILINR,(=62B,=0,=0,=0)
          CALL   GCENDI
          SPACE  2
 BPAWN    COORD  33,!,$262730$
          CALL   GIIV,(=-2,=1,=1)
          CALL   GIIV,(=-2,=0,=1)
          CALL   GIIV,(=-2,=-1,=1)
          CALL   GIIV,(=0,=-2,=1)
          CALL   GIIV,(=2,=-2,=1)
          CALL   GILINR,(=0,=-4,=1,=0)
          CALL   GILINR,(=-3,=-6,=1,=0)
          CALL   GILINR,(=10B,=0,=1,=0)
          CALL   GILINR,(=-3,=6,=1,=0)
          CALL   GILINR,(=0,=4,=1,=0)
          CALL   GIIV,(=2,=2,=1)
          CALL   GILINR,(=55B,=2,=0,=0)
          CALL   GCENDI
          SPACE  2
 WPAWN    COORD  24,*,$12$
          COORD  25,*,$121314262730$
          COORD  26,*,$12131415162562730$
          COORD  27,*,$121314151617202122232425262730$1$
          COORD  30,!,$121314151617202122232425262730$1$
          ENDP
          SPACE  2
 WROOK    COORD  20,*,$1226273031323334$
          COORD  21,*,$12132526273031323334$
          COORD  22,*,$1213142526273031323334$
          COORD  23,*,$12131415242526273031323334$
          COORD  24,*,$121314151617202122232425262730$
          COORD  25,*,$121314151617202122232425262730$
          COORD  26,*,$121314151617202122232425262730273031323334$
          COORD  27,*,$121314151617202122232425262730273031323334$
          COORD  30,!,$121314151617202122232425262730273031323334$
```

```
         ENDP
         SPACE  2
WNITE    COORD  17,!,$25$
         COORD  20,!,$23242526$
         COORD  21,!,$2223242526$
         COORD  22,!,$1222232425262 7$
         COORD  23,!,$1213232425262730$
         COORD  24,!,$12131423242526273031$
         COORD  25,!,$121314151624252627303132$
         COORD  26,!,$12131415161720212223242526273031323 3$
         COORD  27,!,$121314151617202122232425262730313233334$
         COORD  30,!,$12131415161720212223242526273031323334$
         COORD  31,!,$121314151617202122232425262730313233343 5$
         COORD  32,!,$12131415161720212223242526273031323334353 6$
         COORD  33,!,$121314151617202122232425262730313233435$
         COORD  34,!,$12131415161720212223242526273031323334$
         COORD  35,!,$12132021222324252627303132 33$
         COORD  36,!,$122223242526273031$
         COORD  37,!,$252627$
         ENDP
         SPACE  2
WBSHP    COORD  22,!,$12$
         COORD  23,!,$121327303132$
         COORD  24,!,$12131425262730313233$
         COORD  25,!,$121314152425262730313233334$
         COORD  26,!,$12131415161720212223242526273031323334 35$
         COORD  27,!,$12131415161720212223242526273031323334353637$
         COORD  30,!,$12131415161720212223242526273033343536374041$
         COORD  31,!,$1213141516172021222324252627303135363 7$
         COORD  32,!,$121314151617202122232425262730313235$
         COORD  33,!,$121314152425262730313233334$
         COORD  34,!,$12131425262730313233$
         COORD  35,!,$121327303132$
         COORD  36,!,$12$
         ENDP
         SPACE  2
WQUEEN   COORD  17,*,$12414243$
         COORD  20,*,$12133536374041$
         COORD  21,*,$12131431323334353 63740$
         COORD  22,*,$121314152526273031323334353637$
         COORD  23,*,$12131415161212223242526273031323334 35$
         COORD  24,*,$12131415161720212223242526273031323334$
         COORD  25,*,$121314151617202122232425262730313233$
         COORD  26,*,$12131415161720212223242526273031323334 35$
         COORD  27,*,$121314151617202122232425262730313233343536374041$
         COORD  30,!,$121314151617202122232425262730313233343536374041 4
,243$
         ENDP
         SPACE  2
WKING    COORD  17,*,$12131427303132$
         COORD  20,*,$121314152526273031323 3$
         COORD  21,*,$12131415242526273031323334$
         COORD  22,*,$121314151623242526273031323334$
         COORD  23,*,$12131415161720212223242526273031323334 35$
         COORD  24,*,$12131415161720212223242526273031323334 35$
         COORD  25,*,$121314151617202122232425262730313233343536374041$
         COORD  26,*,$121314151617202122232425262730313233343536374041$
         COORD  27,*,$12131415161720212223242526273031323334353637404 14
,243$
         COORD  30,!,$12131415161720212223242526273031323334353637404 14
,243$
         ENDP
         CALL   GCIBL,(DF1,=0)    CREATE DISPLAY FILE DF1
         CALL   GIPOSA,(=407,=955,=0)
         CALL   GISCAL,(=3,=0)
         CALL   GITEXT,(=L*CHESS  'V'*,=0)   DISPLAY HEADING
         CALL   GISCAL,(=2,=5RSCALE)   SET HARDWARE SCALE TO 2
         CALL   GIPOSA,(='BDX0',='BDY0',=0)   POSITION BEAM
         CALL   GISGR,(=1,=0)     PUT ON SUPPRESSED GRID MODE
         CALL   GIPOSX,(='BDX8',=0) DRAW BOARD
LOAD1    ECHO   ,YI=(1,3,5,7),YIP1=(2,4,6,8)
         CALL   GISGR,(=0,=0)     TURN SUPPRESSED GRID OFF
         CALL   GIPOSY,(='BDY!YI',=0)
         CALL   GISGR,(=1,=0)     PUT ON SUPPRESSED GRID MODE
         CALL   GIPOSX,(='BDX0',=0)
         CALL   GISGR,(=0,=0)     TURN SUPPRESSED GRID OFF
         CALL   GIPOSY,(='BDY!YIP1',=0)
         CALL   GISGR,(=1,=0)     PUT ON SUPPRESSED GRID MODE
         CALL   GIPOSX,(='BDX8',=0)
LOAD1    ENDD
         CALL   GIPOSY,(='BDY0',=0)
LOAD2    ECHO   ,XI=(7,5,3,1),XIM1=(6,4,2,0)
         CALL   GISGR,(=0,=0)     TURN SUPPRESSED GRID OFF
         CALL   GIPOSX,(='BDX!XI',=0)
         CALL   GISGR,(=1,=0)     PUT ON SUPPRESSED GRID MODE
         CALL   GIPOSY,(='BDY8',=0)
         CALL   GISGR,(=0,=0)     TURN SUPPRESSED GRID OFF
         CALL   GIPOSX,(='BDX!XIM1',=0)
         CALL   GISGR,(=1,=0)     PUT ON SUPPRESSED GRID MODE
         CALL   GIPOSY,(='BDY0',=0)
LOAD2    ENDD
LOAD3    ECHO   ,IX=(1,3,5,7),IXP1=(0,2,4,6)  SHADE IN WHITE SQUARES
         CALL   GIPOSY,(='BDY0',=0)
LOAD4    ECHO   ,IY=(0,2,4,6),IYP1=(1,3,5,7),IYP2=(2,4,6,8)
LOAD5    ECHO   ,J=(1,3),K=(2,4)
         CALL   GISGR,(=0,=0)     TURN SUPPRESSED GRID OFF
         CALL   GIPOSX,(='BDXX!IX!J',=0)
         CALL   GISGR,(=1,=0)     PUT ON SUPPRESSED GRID MODE
         CALL   GIPOSY,(='BDY!IYP1',=0)
         CALL   GISGR,(=0,=0)     TURN SUPPRESSED GRID OFF
         CALL   GIPOSX,(='BDXX!IX!K',=0)
         CALL   GISGR,(=1,=0)     PUT ON SUPPRESSED GRID MODE
         CALL   GIPOSY,(='BDY!IY',=0)
LOAD5    ENDD
         CALL   GISGR,(=0,=0)     TURN SUPPRESSED GRID OFF
         CALL   GIPOSX,(='BDXX!IX!5',=0)
         CALL   GISGR,(=1,=0)     PUT ON SUPPRESSED GRID MODE
         CALL   GIPOSY,(='BDY!IYP1',=0)
LOAD6    ECHO   ,J=(1,3),K=(2,4)
         CALL   GISGR,(=0,=0)     TURN SUPPRESSED GRID OFF
         CALL   GIPOSX,(='BDXX!IXP1!J',=0)
         CALL   GISGR,(=1,=0)     PUT ON SUPPRESSED GRID MODE
         CALL   GIPOSY,(='BDY!IYP2',=0)
         CALL   GISGR,(=0,=0)     TURN SUPPRESSED GRID OFF
         CALL   GIPOSX,(='BDXX!IXP1!K',=0)
         CALL   GISGR,(=1,=0)     PUT ON SUPPRESSED GRID MODE
         CALL   GIPOSY,(='BDY!IYP1',=0)
LOAD6    ENDD
         CALL   GISGR,(=0,=0)     TURN SUPPRESSED GRID OFF
         CALL   GIPOSX,(='BDXX!IXP1!5',=0)
         CALL   GISGR,(=1,=0)     PUT ON SUPPRESSED GRID MODE
         CALL   GIPOSY,(='BDY!IYP2',=0)
LOAD4    ENDD
         CALL   GISGR,(=0,=0)     TURN SUPPRESSED GRID OFF
LOAD3    ENDD
         CALL   GINOP,(=4RLPEN)   LABEL FOR INSERTING CALL TO EN LPEN
         ECHO   3,I=(0,1,2,3,4,5,6,7)
         CALL   GIPOSA,(='BDXX03',='BDYY!I',=0)  PUT BEAM AT BEG RANK
         ECHO   1,J=(0,1,2,3,4,5,6,7)
         CALL   GINOP,(=1!I!J!B)  CALLS TO PIECES GO HERE, ONE PER SQ
         CALL   GISCAL,(=3,=0)    CHANGE SCALE TO 3
         CALL   GICALL,(=4RTPIN,=3RTPN)  SELECT BUTTON FOR TYPEINS
         CALL   GILP,(=0,=4RLPOF)   DISABLE LIGHT PEN
         CALL   GINOP,(=3RMSG)    BEGINNING OF MESSAGE
         CALL   GINOP,(=3RMSZ)    END OF MESSAGE
         CALL   GINOP,(=3RERR)    BEGINNING OF ERROR MESSAGE
         CALL   GINOP,(=3RERZ)    END OF ERROR MESSAGE
         CALL   GINOP,(=4RTYPE)
         CALL   GCENDI            END INSERTION MODE
LOAD7    RJ     =XSETUP           SET UP PIECES
         CALL   GCCBL,(DF1,=3RERR,=3RERZ)
         CALL   GINOP,(=3RERR)
         CALL   GINOP,(=3RERZ)
         CALL   GCENDI            CLEAR ERROR LINE
         CALL   GCSEND            EMPTY BUFFER
         SA5    SCRATCH           RESTORE X5
         SB1    1                 AND B1
         EQ     =XDUDLOAD         RETURN
DUDLINK  BSS    0
*CALL    LINK
         CON    0
         SPACE  4
IMLAC    ELSE   1
END      MICRO  1,,
         END    'END'
IML
         IDENT  IML               IMLAC DISPLAY ROUTINE
         TITLE  IML - IMLAC DISPLAY OVERLAY
         SST
IMLAC    IF     DEF,IMLAC
***      IML - IMLAC DISPLAY ROUTINE.
*        J. M. KYRIAKOPULOS 3/14/75.
         SPACE  4
***             IML CONTAINS ROUTINES FOR DISPLAYING THE CURRENT BOARD
*        POSITION, RELAYING MESSAGES TO THE IMLAC PDS-1 TERMINAL, AND
*        ACCEPTING LIGHT PEN AND KEYBOARD INPUT.
         SPACE  4
         ENTRY  DUDERR.           PASS ERROR MESSAGES
         ENTRY  DUDMSG.           PASS MESSAGES
         ENTRY  DUDSET.           SET FLAG FOR TYPE OF READ
         ENTRY  DUDCLR.           CLEAR TYPEIN LINE
         ENTRY  DUDREA.           INPUT A MOVE OR TYPEIN
         ENTRY  DUDMOV.           SEND CURRENT MOVE
         ENTRY  SYSTEM            CALLED BY GYPSY IF ERROR
         ENTRY  SETUP             SET UP PIECES ON BOARD
         ENTRY  DUDDIS.,DUDECS.,DUDDRO.,DUDINF.,DUDSQR.,DUDKIL.,TWINKL.
         TITLE  SYMBOL DEFINITIONS AND DATA STORAGE
***      SYMBOL DEFINITIONS
*
         SPACE  4
 N       EQU    96                NO OF DOTS PER SQUARE
 INC     EQU    8                 INCREMENT OF DOTS FOR ANIMATION
 NINSQ   EQU    N/INC             NO OF INCREMENTS PER SQUARE
 BDX0    EQU    128               X-COORD OF BOTTOM LEFT SQUARE
 BDY0    EQU    160               Y-COORD OF BOTTOM LEFT SQUARE
 BDX     EQU    BDX0+N/2
 BDY     EQU    BDY0+N/2
         SPACE  4
         USE    //
 A       BSS    1                 FOR GYPSY

*CALL  USEIML

         USE    /KCIO/
 KCIO    BSS    1                 I/O FLAG IN GYPSY

         USE    0
*CALL    BOARD
         TITLE  DATA STORAGE
*CALL    SQRLST
         TITLE  CALL MACRO AND MICROS
*CALL CALL
         SPACE  4
***      MICROS
*
         SPACE  4
 YTYP    DECMIC 20                Y-COORD OF TYPEIN LINE
 YERR    DECMIC 65                Y-COORD OF ERROR MESSAGE LINE
 YMES    DECMIC 110               Y-COORD OF MESSAGE LINE
 XMSG    DECMIC 48                X-COORD OF ALL THREE LINES
         TITLE  SYSTEM - ERROR ROUTINE CALLED BY GYPSY
***      SYSTEM - ERROR ROUTINE CALLED BY GYPSY
*
         SPACE  4
SYSTEM   BSSZ   1
         BX6    X2                ADDR OF ERROR MESSAGE
         SA6    MSG
         CALL   GCCBL,(DF1,=3RMSG,=3RMSZ)
         CALL   GIPOSA,(='XMSG','YMES',=3RMSG)  POSITION  BEAM
         SA1    MSG
         CALL   GITEXT,(X1,=3RMSZ)
         CALL   GCENDI
         CALL   GCSEND
         EQ     *+400000B         GET DUMP
         TITLE  SETUP - SET UP PIECES ON BOARD
***      SETUP - SET UP PIECES ON BOARD
*                CALLED BY DUDMOVE AND DUDLOAD
         SPACE  4
***             SETUP TAKES THE BOARD POSITION FOUND IN THE ARRAY BOARD
*        AND INSERTS CALLS TO THE APPROPRIATE DISPLAY BLOCK TO DISPLAY
*        THE PIECES ON THE BOARD
         SPACE  4
SETUP    BSSZ   1
         BX6    X6-X6             ZERO
         SA6    XIN               STORE COUNTER
SET1     SA1    X6+BOARD          PICK UP PIECE NO
         SX7    X1+40B            ADD OFFSET
         SA7    BNAME             STORE AS BLOCK NAME FOR PIECE
         SX7    X6+100B           GET SQUARE NO
         SA7    LNAME             STORE LABEL NAME OF SQUARE
         CALL   GCCBL,(DF1,LNAME,=0)   INSERT CALL TO THE PIECE DIS BLK
         CALL   GICALL,(BNAME,LNAME)
         CALL   GCENDI
         SB1    1
         SA2    XIN               PICK UP COUNTER
         SX6    X2+B1             INCREMENT IT
         SA6    A2                STORE COUNTER
         SX3    X6-64
         NG     X3,SET1           HAVE ALL SQUARES BEEN INSERTED
         EQ     SETUP             RETURN
         TITLE  DUDERR - PASS ERROR MESSAGES
***      DUDERR - PASS ERROR MESSAGES TO IMLAC
*
```

```
         SPACE  4
DUDERR.  BX6    X5
         SA6    MSG              ADDR OF MESSAGE
         CALL   GCCBL,(DF1,=3RERR,=3RERZ)
         CALL   GIPOSA,(='XMSG',='YERR',=3RERR)  POSITION BEAM
         SA1    MSG
         CALL   GITEXT,(X1,=3RERZ)
         CALL   GCENDI           END INSERTION MODE
         CALL   GCSEND
         SA5    MSG              RESTORE X5
         SB1    1                AND B1
         EQ     =XDUDERR
         TITLE  DUDMSG - PASS MESSAGES
***      DUDMSG - PASS MESSAGES TO IMLAC
*
         SPACE  4
DUDMSG.  BX6    X5
         SA6    SCRATCH          SAVE X5
         BX6    X1
         SA6    MSG              ADDR OF MESSAGE
         CALL   GCCBL,(DF1,=3RMSG,=3RMSZ)
         CALL   GIPOSA,(='XMSG',='YMES',=3RMSG)  POSITION BEAM
         SA1    MSG
         CALL   GITEXT,(X1,=3RMSZ)
         CALL   GCENDI           END INSERTION MODE
         CALL   GCSEND
         SA5    SCRATCH          RESTORE X5
         SB1    1                AND B1
         EQ     =XDUDMSG
         TITLE  USELESS ROUTINES
***      USELESS ROUTINES - ROUTINES CALLED BY CHESS FOR DUD DISPLAY
*               AND NOT NEEDED FOR THIS IMLAC DISPLAY.
         SPACE  4
DUDDRO.  EQ     =XDUDDROP
DUDKIL.  EQ     =XDUDKILL
DUDINF.  EQ     =XDUDINFO
DUDECS.  EQ     =XDUDECS
TWINKL.  EQ     =XTWINKLE
DUDSQR.  EQ     =XDUDSQRL
DUDDIS.  EQ     =XDUDDISP
         TITLE  DUDSET - SET FLAG FOR TYPE OF READ
***      DUDSET - SET FLAG FOR TYPE OF READ
*               THIS ROUTINE IS CALLED BEFORE A READ TO INDICATE WHAT
*               KIND OF REPLY CHESS IS EXPECTING.  IF X1 IS ZERO, A
*               MOVE IS EXPECTED.  OTHERWISE A TYPEIN IS EXPECTED.
*               X1 IS STORED INTO MFLAG FOR DUDREAD.
         SPACE  4
DUDSET.  BX6    X1
         SA6    MFLAG            STORE X1 FOR DUDREAD
         EQ     =XDUDSET
         TITLE  DUDCLR - CLEARS TYPEIN LINE
***      DUDCLR - CLEAR TYPEIN LINE AND INSERT TYPEIN SELECT SQUARE
*
         SPACE  4
DUDCLR.  BX6    X5
         SA6    SCRATCH          SAVE X5
         CALL   GCIBL,(DF1,=0)
         CALL   GINOP,(=3REND)
         CALL   GCENDI           APPEND A LABEL AT END OF DISPLAY FILE
         CALL   GCCBL,(DF1,=4RTYPE,=3REND)
         CALL   GINOP,(=4RTYPE)
         CALL   GCENDI           DELETE TYPEIN LINE
         CALL   GCCBL,(DF1,=3RTPN,=0)
         CALL   GICALL,(=4RTPIN,=3RTPN)
         CALL   GCENDI           INSERT TYPEIN SELECT SQUARE
         CALL   GCSEND           DUMP BUFFER
         SA5    SCRATCH          RESTORE X5
         SB1    1                AND B1
         EQ     =XDUDCLR         RETURN
         TITLE DUDREAD - INPUT A MOVE OR TYPEIN
***      DUDREAD - INPUT A MOVE OR TYPEIN
*               DUDREAD IS CALLED WHENEVER INPUT IS WANTED.  IF MFLAG
*               IS ZERO, A MOVE IS EXPECTED.  THE LIGHT PEN IS THEN
*               ENABLED FOR PIECES AND THE TYPEIN SELECT SQUARE.  IF
*               THE TYPEIN SELECT SQUARE IS SELECTED, IT IS DELETED
*               AND A READ ON KEYBRD IS INITIATED.  THE TYPEIN IS THEN
*               SENT TO CHESS VIA THE ADDRESS SPECIFIED BY X1.  IF A
*               PIECE IS SELECTED, THE TRACKING CROSS APPEARS OVER
*               THE PIECE AND BOTH ARE MOVED WITH THE LIGHT PEN.  THE
*               SELECTION OF THE X BUTTON DEFINES THE DESTINATION
*               SQUARE FOR THAT PIECE.  THE MOVE IS THEN CHECKED FOR
*               LEGALITY, TRANSLATED INTO DISPLAY CODE BY CALLING THE
*               CHESS SUBROUTINE LGLENG, AND SENT TO CHESS.
         SPACE 4
READA    DATA   10H              BLANKS
READB    CON   0.010416666666666P48+1-1777BS48+48S48  1/96
READD    DATA   8LP-R1.
DUDREA.  BX6    X5
         SA6    SCRATCH          SAVE X5
         BX6    X1
         SA6    MSG              SAVE ADDRESS TO PUT MOVE
         SA2    MFLAG
         NZ     X2,READ7         IF TYPEIN
READ0    CALL   GCCBL,(DF1,=4RLPEN,=0)   A MOVE
         CALL   GILP,(=1,=4RLPEN)   ENABLE LIGHTPEN
         CALL   GCENDI
         CALL   GLLPB,(IX,IY,BNAME,BCNAME,LNAME,IHIT)  GET HIT
         CALL   GCCBL,(DF1,=4RLPEN,=0)
         CALL   GINOP,(=4RLPEN)   DISABLE LIGHT PEN
         CALL   GCENDI
         SA2    IHIT
         ZR     X2,READ0         X BUTTON SELECTED
         SA2    BCNAME
         SA3    =4RTPIN          LABEL OF CALL TO TYPEIN SEL BOX
         IX6    X2-X3
         ZR     X6,READ7         IF TYPEIN BUTTON SELECTED
         SA2    LNAME            LABEL OF SQUARE
         SX7    X2-100B          SQUARE NUMBER
         SA3    X7+BOARD         GIVING PIECE NUMBER
         LX7    6
         SA7    YIN              SAVE SHIFTED FROM SQUARE
         SX7    X3+40B           LABEL OF PIECE BLOCK
         SA7    BNAME
         CALL   GCCBL,(DF1,=3RERR,=3RERZ)
         CALL   GINOP,(=3RERR)
         CALL   GINOP,(=3RERZ)
         CALL   GCENDI           CLEAR ERROR LINE
         CALL   GCSTC,(IX,IY)     POSITION TRACKING CROSS
         CALL   GCABT,(BNAME)     SPECIFY PIECE TO FOLLOW TRAC CROSS
READ1    CALL   GLLPT,(IX,IY)     GET DESTINATION OF TRACKING CROSS
         SA4    IX               GET X-COORD OF HIT
         SX4    X4-128
         NG     X4,READ1         IF LEFT OF BOARD
         SA5    IY               GET Y-COORD OF HIT
         SX5    X5-160
         NG     X5,READ1         IF BELOW BOARD
         SX1    X4-769
         PL     X1,READ1         IF RIGHT OF BOARD
         SX2    X5-769
         PL     X2,READ1         IF ABOVE BOARD
         SA3    READB            1/96
         FX1    X4*X3            / 96
         FX2    X5*X3            / 96
         LX2    3
         BX3    X1+X2            TO SQUARE
         SX7    X3+100B          LABEL NAME OF SQUARE
         SA7    BCNAME
         SA4    YIN              PICK UP FROM SQUARE
         BX4    X3+X4            ADD WITH TO SQUARE
         MX5    48               MASK
         SA1    LINDX            LWA+1 OF MOVES ARRAY
         SB2    2
         SB3    X1
         SB4    MOVES            FWA OF MOVES ARRAY
READ2    SA2    B4               GET MOVE
         NG     X2,READ3         IF ILLEGAL
         BX6    -X5*X2           MASK OFF TO AND FROM SQUARES
         BX6    X6-X4
         ZR     X6,READ4         IF EQUAL
READ3    SB4    B4+B2            NEXT MOVE
         LT     B4,B3,READ2
         SA3    READD
         BX6    X3
         SA2    MSG              FINISHED MOVE LIST W/OUT FINDING MOVE
         SA6    X2               RETURN P-R1    TO CHESS TO GET
         EQ     READ6              ILLEGAL MOVE MESSAGE BACK
READ4    BX6    X2               A MOVE WAS FOUND
         AX2    12               RIGHT ADJUST FLAGS
         SX5    10B              PROMOTION MASK
         BX5    X5*X2
         NZ     X5,READ8         IF PAWN PROMOTION
READ5    SA6    BSTMV            PUT MOVE IN BSTMV
         SA2    READA
         BX7    X2               PUT BLANKS IN X7
         SA5    SCRATCH          RESTORE X5
         SB1    1
         RJ     =XLGLENG         RETURNS DISPL CODE MOVE IN X6-X7
         SA1    MSG              ADDR TO PUT DISPLAY CODE MOVE
         SA6    X1
         SA7    A6+1             STORE MOVE
READ6    CALL   GCSEND           DUMP BUFFER
         SA5    SCRATCH          RESTORE X5
         SB1    1                AND B1
         EQ     =XDUDREAD        RETURN
READ7    CALL   GCCBL,(DF1,=3RTPN,=0)   TYPEIN
         CALL   GINOP,(=3RTPN)
         CALL   GCENDI           DELETE TYPEIN SELECT SQUARE
         CALL   GCCBL,(DF1,=3RERR,=3RERZ)
         CALL   GINOP,(=3RERR)
         CALL   GINOP,(=3RERZ)
         CALL   GCENDI           CLEAR ERROR LINE
         CALL   GCCBL,(DF1,=4RTYPE,=0)
         CALL   GIPOSA,(='XMSG',='YTYP',=4RTYPE)
         CALL   GCENDI           POSITION BEAM FOR TYPEIN
         CALL   GCSEND
         SB1    1
         SX6    B1
         SA6    KCIO             SET GYPSY FLAG = 1
         SA1    MSG
         SA5    SCRATCH          RESTORE X5
         RJ     =XRDLINE         READ MOVE FROM KEYBRD
         EQ     =XDUDREAD
MROOK    BSS    1
LROOK    BSS    1
MQUEEN   BSS    1
LQUEEN   BSS    1
MBSHP    BSS    1
LBSHP    BSS    1
MNITE    BSS    1
LNITE    BSS    1
READ8    SA1    BNAME            PAWN PROMOTION
         SX1    X1-40B           THE PAWN
         AX1    59               + OR - 0
         SB1    1
         SB5    40B              PIECE BLOCK NAME OFFSET
         SB6    200B             OFFSET FOR PROMOTION PIECE CALLS
         SX7    A2
         SA7    MFLAG            STORE ADDR MOVE
         SX2    B2               ROOK
         BX6    X2-X1
         SX6    X6+B5            BLOCK NAME OF ROOK
         SA6    LROOK
         SX6    X6+B6
         SA6    MROOK            LABEL OF CALL
         SX2    X2+B1            KNIGHT
         BX6    X2-X1
         SX6    X6+B5            BLOCK NAME OF KNIGHT
         SA6    LNITE
         SX6    X6+B6
         SA6    MNITE            LABEL OF CALL
         SX2    X2+B1            BISHOP
         BX6    X2-X1
         SX6    X6+B5            BLOCK NAME OF BISHOP
         SA6    LBSHP
         SX6    X6+B6
         SA6    MBSHP            LABEL OF CALL
         SX2    X2+B1            QUEEN
         BX6    X2-X1
         SX6    X6+B5            BLOCK NAME OF QUEEN
         SA6    LQUEEN
         SX6    X6+B6
         SA6    MQUEEN           LABEL OF CALL
         CALL   GCCBL,(DF1,=3RERR,=3RERZ)
         CALL   GINOP,(=3RERR)
         CALL   GINOP,(=3RERZ)
         CALL   GCENDI           DELETE ANY ERROR MESSAGE
         CALL   GCIBL,(DF1,=0,=0)
         CALL   GIPOSA,(=128,=7,=4RMENU)
         CALL   GISGR,(=1,=0)     TURN SUPPRESSED GRID MODE ON
         CALL   GIPOSA,(=862,=103,=0)
         CALL   GIPOSA,(=128,=7,=0)
         CALL   GISGR,(=0,=0)     TURN SUPPRESSED GRID MODE OFF
         CALL   GIPOSA,(=185,=40,=0)
         CALL   GITEXT,(=13LPICK A PIECE:,=0)
         CALL   GIPOSA,(=468,=55,=0)
         CALL   GISCAL,(=2,=0)    SET SCALE TO 2
         CALL   GILP,(=1,=0)      ENABLE LIGHT PEN
         CALL   GICALL,(LQUEEN,MQUEEN)
         CALL   GICALL,(LROOK,MROOK)
         CALL   GICALL,(LBSHP,MBSHP)
         CALL   GICALL,(LNITE,MNITE)
         CALL   GILP,(=0,=4RMENZ) DISABLE LIGHT PEN
         CALL   GCENDI
READ8A   CALL   GLLPB,(IX,IX,IX,IX,IY,IHIT)
```

```
          SA2    IHIT
          ZR     X2,READ8A          IF X BUTTON SELECTED
          CALL   GCDBL,(DF1,=4RMENU,=4RMENZ)   DELETE MENU
          SA2    IY                 LABEL OF PIECE CALL SELECTED
          SX6    X2-200B            BLOCK NAME OF PIECE
          SA6    BNAME              CHANGE PAWN TO CHOSEN PIECE
          SX6    X6-40B             PIECE
          BX7    X6
          AX7    59
          BX7    X6-X7              ABS VAL OF PIECE
          SX7    X7+2               P, E, AND O FLAGS RIGHT ADJUSTED
          LX7    13
          SA1    MFLAG              GET ADDR OF MOVE
          SA1    X1                 GET MOVE
          SX3    160000B            P, E, AND O FLAGS MASK
          BX2    X1*X3
          BX2    X2-X7
          ZR     X2,READ11          THE RIGHT PROMOTION MOVE
          MX5    48
          BX1    -X5*X1
          BX5    -X5+X3
          BX7    X1+X7              MOVE TO TEST AGAINST
          SA4    LINDX              LWA + 1 OF MOVES ARRAY
          SB2    2
          SB3    X4
          SB4    A1+B2
READ9     SA1    B4                 NEXT MOVE
          NG     X1,READ10          IF ILLEGAL
          BX4    X5*X1
          BX4    X4-X7
          ZR     X4,READ11          IF MOVE IS FOUND
READ10    SB4    B4+B2              ADDR NEXT MOVE
          LT     B4,B3,READ9
          EQ     350000B            COUGH AND CHOKE
READ11    BX6    X1
          EQ     READ5              STORE MOVE
          TITLE  DUDMOVE - SEND TO IMLAC CURRENT MOVE POSITION
***       DUDMOVE - SEND TO IMLAC CURRENT MOVE POSITION
*                THE MOVE IS CHECKED TO SEE WHETHER IT IS A CASTLE OR
*                A CAPTURE EN PASSANT.  IF SO, THE BOARD IS RECREATED
*                BY CALLING SETUP.  OTHERWISE THE MOVED PIECE IS DELETED
*                FROM ITS ORIGINAL SQUARE AND ANIMATED TO THE DES-
*                TINATION SQUARE.
          SPACE  4
DUDMOV.   BX6    X5
          SA6    SCRATCH            SAVE X5
          SA2    BSTMV              MOVE
          SX1    77B
          BX4    X1*X2              TO SQUARE
          SA3    X4+BOARD           ADDR OF PIECE NO
          SX7    X3+40B             BLOCK NAME OF PIECE
          SA7    BNAME
          SX6    X4+100B            LABEL NAME OF TO SQUARE
          SA6    BCNAME
          AX2    6                  SHIFT MASK
          BX5    X1*X2              FROM SQUARE
          SX6    X5+100B            BLOCK NAME OF FROM SQUARE
          SA6    LNAME
          AX2    6                  RIGHT ADJUST FLAGS
          SX6    76B                CAPTURE EN PASSANT MASK
          BX6    X6*X2
          SX6    X6-44B
          ZR     X6,MOVE10          IF CAPTURE EN PASSANT
          SX6    72B                CASTLING MASK
          BX6    X6*X2
          SX6    X6-22B
          ZR     X6,MOVE10          IF CASTLING
          SX1    7
          BX2    X4*X1              X COORD OF TO SQUARE NUMBER
          BX3    X5*X1              X COORD OF FROM SQ NUM
          AX4    3                  Y COORD OF TO SQ NUM
          AX5    3                  Y COORD OF FROM SQ NUM
          SX1    N                  NO OF DOTS PER SQUARE
          IX6    X1*X3
          IX7    X1*X5
          SX6    X6+BDX
          SX7    X7+BDY
          SA6    XIN                X DISPLACEMENT
          SA7    YIN                Y DISPLACEMENT
          IX6    X2-X3              DELTA X SQUARES
          IX7    X4-X5              D-Y SQUARES
          SB2    59
          AX4    X6,B2              SIGN X
          BX1    X6-X4              ABS D-X
          AX5    X7,B2              SIGN Y
          BX2    X7-X5              /D-Y/
          IX3    X1-X2
          ZR     X3,MOVE3           IF /D-X/ = /D-Y/ (BISHOP-TYPE MOVE)
          PL     X3,MOVE4           IF /D-X/ > /D-Y/
          SX3    INC                INCREMENT (NO OF POINTS)
          SX6    NINSQ              NO OF INCREMENTS PER SQ
          ZR     X1,MOVE1           IF D-X = 0  (ROOK-TYPE MOVE)
          BX7    X3-X4              SIGN X * INC (OTHERWISE KNIGHT MOVE)
          AX7    1                  / 2
          SA7    IX                 X INCREMENT
          EQ     MOVE2

MOVE1     BX7    X7-X7              ZERO
          SA7    IX                 X INCREMENT
MOVE2     BX7    X3-X5              SIGN Y * INC
          SA7    IY                 Y INCREMENT
          IX7    X6*X2              NINSQ * /D-Y/
          SA7    NIN                TOTAL NO OF INCREMENTS
          EQ     MOVE7

MOVE3     SX3    INC                INCREMENT
          SX6    NINSQ              NO OF INCREMENTS PER SQ
          BX7    X3-X5              SIGN Y * INC
          SA7    IY                 Y INCREMENT
          EQ     MOVE6

MOVE4     SX3    INC
          SX6    NINSQ              NO OF INCREMENTS PER SQ
          ZR     X2,MOVE5           IF D-Y = 0 (ROOK-TYPE MOVE)
          BX7    X3-X5              SIGN Y * INC (OTHERWISE KNIGHT MOVE)
          AX7    1                  / 2
          SA7    IY                 Y INCREMENT
          EQ     MOVE6

MOVE5     BX7    X7-X7              ZERO
          SA7    IY                 Y INCREMENT
MOVE6     BX7    X3-X4              SIGN X * INC
          SA7    IX                 Y INCREMENT
          IX7    X6*X1              NINSQ * /D-X/
          SA7    NIN                TOTAL NO OF INCREMENTS
MOVE7     CALL   GCCBL,(DF1,LNAME,=0)
          CALL   GICALL,(=40B,LNAME)   EMPTY FROM SQUARE
          RJ     MOVE11             FLASH PIECE
          CALL   GCDBL,(DF1,=4RANIM,=4RANIZ)
          CALL   GCIBL,(DF1,=0)
          CALL   GINOP,(=4RANIM)
          CALL   GINOP,(=4RANIZ)
          CALL   GCENDI
          SX7    1
MOVE8     SA7    IHIT               COUNTER
          CALL   GCCBL,(DF1,=4RANIM,=4RANIZ)
          CALL   GIPOSA,(XIN,YIN,=4RANIM)    MOVE BEAM
          CALL   GICALL,(BNAME,=4RANIZ)
          CALL   GCENDI             ANIMATE PIECE
          CALL   GCTIM,(ANITIM)
          SA1    IX                 X INC
          SA2    IY                 Y INC
          SA3    XIN                D DISPL
          SA4    YIN                Y DISPL
          IX6    X3+X1              INCREMENT X DISPL
          SA6    XIN
          IX6    X4+X2              INCREMENT Y DISPL
          SA6    YIN                Y DISPLACEMENT
          SA1    IHIT               COUNTER
          SA2    NIN                LIMIT
          SX7    X1+1               INCREMENT COUNTER
          IX3    X2-X7
          PL     X3,MOVE8
          CALL   GCDBL,(DF1,=4RANIM,=4RANIZ)   DELETE ANIMATION
          CALL   GCCBL,(DF1,BCNAME,=0)
          CALL   GICALL,(BNAME,BCNAME)   INSERT PIECE AT TO SQ
          RJ     MOVE11
          CALL   GCDBL,(DF1,=4RANIM,=4RANIZ)   DELETE FLASH
MOVE9     CALL   GCCBL,(DF1,=3RERR,=3RERZ)
          CALL   GINOP,(=3RERR)
          CALL   GINOP,(=3RERZ)
          CALL   GCENDI             CLEAR ERROR LINE
          CALL   GCSEND
          SA5    SCRATCH            RESTORE X5
          SB1    1                  AND B1
          EQ     =XDUDMOVE

MOVE10    RJ     =XSETUP            SET UP PIECES
          EQ     MOVE9

MOVE11    EQ     *+400000B
          CALL   GILINR,(=-60B,=0,=0,=4RANIM)
          CALL   GICALL,(BNAME,=0)
          CALL   GILINR,(=-60B,=0,=0,=0)
          CALL   GICALL,(BNAME,=0)
          CALL   GILINR,(=-60B,=0,=0,=0)
          CALL   GICALL,(BNAME,=4RANIZ)
          CALL   GCENDI             FLASH PIECE
          CALL   GCTIM,(=500)       PAUSE
          EQ     MOVE11
          SPACE 4
IMLAC     ENDIF
          END
LGLENG
          IDENT  LGLENG
          SST
IMLAC     IF     DEF,IMLAC
          B1=1
LGLENG    TITLE  LGLENG - GENERATE LEGAL ENGLISH NOTATION.
*CALL     BOARD
*CALL     SQRLST
LGLENG    SPACE  4,88
***       LGLENG - GENERATE LEGAL ENGLISH NOTATION.
*
*         ENTRY  (BSTMV) = THE MOVE.
*
*         EXIT   (X6, X7) = C-FORMAT MOVE.
*
*         USES   ALL REGISTERS.
*         CALLS  ENGGEN.
*


          ENTRY  LGLENG
LGLENG    EQ     *+400000B          ENTRY/EXIT
          SA1    BSTMV              THE MOVE
          SB1    1
          SB2    BOARD              ENGGEN PARAMETER
          BX2    X1
          LX2    59-S.PRO
          PL     X2,LGLENG1         IF NOT PROMOTION
          RJ     ENGOOD             DO ENGGEN PLUS SLASH FOR CLARITY
          SA1    BSTMV              THE MOVE AGAIN
          LX1    59-S.PRO+3         POSITION TYPE LOWER
          MX2    -2
          BX2    -X2*X1             EXTRACT PIECE INDEX
          BX7    X2
          SA3    LGLENGA            12/=R,12/=N,12/=B,12/=Q
          LX2    3                  8*INDEX
          LX7    2
          IX2    X7+X2              12*INDEX
          SB2    X2
          LX7    X3,B2
          MX1    12
          BX7    X1*X7              EXTRACT PIECE TYPE
          EQ     LGLENG             RETURN

LGLENG1   LX2    S.PRO-S.CAS
          PL     X2,LGLENG2         IF NOT CASTLE
          LX2    S.CAS-S.ENP        S.ENP IS SET FOR O-O-O
          AX2    30
          MX3    30
          SA1    LGLENGB            O-O-O
          BX3    X2*X3              CASTLE LONG MASK
          MX2    18
          BX3    X2+X3              OR CASTLE SHORT MASK
          BX6    X1*X3
          BX7    X7-X7
          EQ     LGLENG             RETURN

LGLENG2   RJ     ENGOOD
          BX7    X7-X7              CLEAR SECOND WORD
          EQ     LGLENG             RETURN


**        ENGOOD - GET SLIGHTLY CLEANED UP ENGGEN MOVE TRANSLATION.
*
*         CALLS  ENGGEN
*
*         INSERTS SLASH AFTER FROM PIECE SO NOT CONFUSED WITH BKP TYPIN.

ENGOOD    EQ     *+400000B          ENTRY/EXIT
          RJ     =XENGGEN           GET MOVE WITHOUT SLASH
          MX2    12
```

```
          BX3    X2*X6
          BX6    -X2*X6
          LX3    6                MAKE ROOM FOR SLASH
          BX3    X2*X3
          BX6    X6+X3            MOVE WITH PIECE SHIFTED
          SX2    1R/
          LX2    48               MOVE SLASH INTO POSITION
          BX6    X6+X2            INSERT INTO ENGLISH MOVE
          EQ     ENGOOD

 LGLENGA  VFD    12/2R=R,12/2R=N,12/2R=B,12/2R=Q,12/0
 LGLENGB  CON    5LO-O-O
 IMLAC    ENDIF
          END
CHESS20
          IDENT  CHE$$20
          SST
 CHESS20  TITLE  CHESS20 - MISCELLANEOUS COMMANDS AND SUBROUTINES.
          LIST   F,X
          B1=1
          CON    =XECHES20         ECS PLUG TABLE

 LIBGEN   IF     DEF,LIBGEN        IF 1,0 / 2,0 MERGE
 END      MICRO  1,,
          ENTRY  KILCMD
          ENTRY  INPEOR
          ENTRY  BLICMD
          ENTRY  CLOCMD
          ENTRY  COMCMD
          ENTRY  MSGCMD
          ENTRY  NAMCMD
          ENTRY  ECSCMD
          ENTRY  PINCMD
          ENTRY  PSLCMD
          ENTRY  GONCMD
          ENTRY  FINISH
          ENTRY  FINSHD
          ENTRY  PRICMD
          ENTRY  ENDCMD
 LIBGEN   ELSE   1
 20       THIS
*CALL     LET


 BSS.     OPSYN  BSS
          PURGMAC BSS

          MACRO  BSS,LABL,N
 LABL     BSS.   N
 X        MICRO  1,, LABL
 X        MICCNT X
          IFLE   X,6,5
 X        MICRO  1,,*N*
          IFC    EQ,*'X'*2*,2
 SQRLST   MICRO  1,, 'SQRLST',LABL
          SKIP   1
 OTHERS   MICRO  1,, 'OTHERS',LABL
          ENDM
 SQRLST   MICRO  1,,
 OTHERS   MICRO  1,,
*CALL     SQRLST
*CALL     TLET
*CALL     IOCOM
*CALL     MEMORY
 SQRLST   MICRO  2,, 'SQRLST'
 OTHERS   MICRO  2,, 'OTHERS'
 BSS      OPSYN  BSS.
*CALL     BOARD
 ECERTB   SPACE  4
 ECHES20  ECERTB EINITAL
*CALL     TABLE
          SPACE  4
 C.CPRPV  CEQU   1                 RPV BYTE IN W.CPRPV
 S.LBRPV  CEQU   10D               LIBRARY LOAD FLAG IN C.CPRPV
 END      SPACE  4,88
***       ENDCMD - END THE PROGRAM.
*



 ENDCMD   SX5    =C* THANK YOU FOR AN ENJOYABLE TIME.*
          RJ     =XWRLINE
          WRITER =XOUTPUT,RECALL
          WRITER =XHARDCPY,RECALL
          RJ     =XMSCLOS          CLOSE LIBRARY
          CLOSE  =XLIBRARY,,RECALL
          DAYFILE =C*GOODBYE*,0,RECALL
 NUCC     IF     DEF,NUCC
          SA2    CPRPV             OLD RPV WORD FROM CONTROL POINT
          LX2    11-S.LBRPV+12*C.CPRPV
          PL     X2,ENDCMD1        IF WE ARE NOT AN ESP CALL
          LX2    12-11+S.LBRPV
          MX3    -9
          BX3    -X3*X2
          LX3    6
          SYSTEM RPV,RECALL,CPESP,X3 RESTORE ESP FILE NAME
 NUCC     ENDIF
 ENDCMD1  ENDRUN
KIL       SPACE  4,10
***       KILCMD - KILL THE PROGRAM.
*
*         CALLS  DUDKILL.
*         EXIT   TO END.
*



 KILCMD   RJ     =XMSCLOS          CLOSE LIBRARY
          CLOSE  =XLIBRARY,,RECALL
          RJ     =XDUDKILL
          EQ     ENDCMD
 FINISH   SPACE  4,12
***       FINISH - END OF GAME.
*
*         CALLS  READER.
*



 FINISH   SA1    VMOVE
          RJ     =XCLMDRW          CLAIM DRAW IF APPROPRIATE
          SX5    ENDMS
          RJ     =XWRLINE
          SX6    O.TGIO
          SX7    0
          SA7    GOCNT
          RJ     =XREADER
 FINSHD   SX5    =C* ENTER INITIALIZE TO CONTINUE.*
          RJ     =XERRMSG
          EQ     FINISH

 PRI      SPACE  4,11
***       PRICMD - PRINT THE BOARD.
*
*         CALLS  PRIBRD, REREAD.
*         USES   OLINE, ALL REGISTERS.
*



 PRICMD   RJ     =XRDRSFT          SKIP FIRST TOKEN
          SA0    BOARD
          RJ     =XRDRGNT          GET NEXT TOKEN
          ZR     X7,PRICMD1        IF NONE
          SA2    PRICMDA
          RJ     =XRDRSXT          SEARCH TRANSLATION TABLE
          ZR     X6,PRICMD1        IF NOT FOUND
          SA0    X6+0
 PRICMD1  RJ     =XPRIBRD          PRINT THE BOARD
          RJ     =XREREAD

 PRICMDA  XLAT   N,NBOARD
          DATA   0                 END OF TABLE
 GONCMD   SPACE  4,22
***       GONCMD - GO NNN MOVES.
*
*         CALLS  RDRSFT, RDRGNT, RDRDXB.
*         USES   ALL REGISTERS.
*



 GONCMD   RJ     =XRDRSFT          SKIP FIRST TOKEN
          RJ     =XRDRGNT          GET NEXT TOKEN
          ID     X6,GONCMD1        IF NO PARAMETER
          BX5    X7
          SB7    B1                ASSUME DECIMAL
          RJ     =XRDRDXB          CONVERT
          NZ     X4,GONCMD1        IGNORE IF JUNK
          SX6    X6-1
          NG     X6,GONCMD1        IGNORE IF ZERO
          SA6    GOCNT
 GONCMD1  MX6    1                 DISABLE PAGING
          SA6    LINES
          BX6    X6-X6             CLEAR OPPONENTS PREDICTED MOVE
          SA6    PONDR
          JUMPUP MYMOVE
 INPEOR   SPACE  4,13
***       INPEOR - PROCESS END OF RECORD ON INPUT FILE.
*



 INPEOR   SA1    INPFN
          ZR     X1,INPEOR1        IF IN PRIMARY FILE
          BX6    X1
          SA6    =XINPUT
          BX7    X7-X7
          SA7    A1
          READ   A6
          RJ     =XREREAD

 INPEOR1  SA1    =XINPUT+1
          RJ     =XDEVTYP          CHECK DEVICE TYPE
          PL     X6,ENDCMD         IF ALLOCATABLE, STOP
          READ   =XINPUT
          RJ     =XREREAD
 NAMCMD   SPACE  4,12
***       NAMCMD - PROCESS NAME COMMAND.
*



 NAMCMD   RJ     =XRDRSFT          SKIP FIRST TOKEN
          BX7    X7-X7
          SB3    6
          SB4    54
 NAMCMD1  AX2    X1,B2
          BX2    -X0*X2
          ZR     X2,NAMCMD2        IF END OF INPUT
          LX6    X2,B4
          BX7    X6+X7
          SB4    B4-B3
          SB2    B2-B3
          PL     B2,NAMCMD1        IF NOT NEW WORD
          SA1    A1+B1
          SB2    54
          EQ     NAMCMD1

 NAMCMD2  BX1    X7
          RJ     =XRDRSFN
          SA6    OPNAM             SET OPPONENTS NAME
          RJ     =XREREAD
 NUCC     IF     DEF,NUCC
 MULCMD   SPACE  4,50
***       MULDMD - ALLOW NNN SLAVE TERMINALS.
*
*         CALLS  RDRSFT, RDRGNT, RDRDXB, WRLINE, ERRMSG.
*         USES   ALL REGISTERS.
*



 MULCMD   RJ     =XRDRSFT          SKIP FIRST TOKEN
          RJ     =XRDRGNT          GET NEXT TOKEN
          ZR     X7,MULCMD4        IF NOTHING ENTERED
          BX5    X7
          SB7    B1                ASSUME DECIMAL
          RJ     =XRDRDXB          CONVERT
          NZ     X4,MULCMD4        IF JUNK
          SA1    MTIDS             IDENTS
          SX7    X6-5
          PL     X7,MULCMD4        IF TOO MANY
          NZ     X6,MULCMD1        IF ALLOWING SOME
          SA6    A1                CLEAR IDS
          ERRNZ  MTLIM-MTIDS-1
          SA6    A6+B1             CLEAR COUNT
          ZR     X1,MULCMD4        IF NO CHANGE
          SX5    =C* ENTER sOLO*
          EQ     MULCMD2

 MULCMD1  SA2    =XOUTPUT+1
          PL     X2,MULCMD4        IF OUTPUT ALLOCATABLE
          SX7    12
          IX7    X6*X7
          SB2    X7                MASK LENGTH
          SA6    A1+B1             SET COUNTS
          MX6    12
          AX7    X6,B2             FORM MASK
          BX7    X1*X7             EXTRACT REMAINING IDS
          SA7    A1
          NZ     X7,MULCMD3        IF ALREADY MULTI
          MX6    12
```

```
          SX5    =C* ENTER mULTI*
          SA6    A1
 MULCMD2  RJ     =XWRLINE
 MULCMD3  RJ     =XREREAD

 MULCMD4  SX5    =C* MULTI COMMAND IGNORED.*
          RJ     =XERRMSG
          RJ     =XREREAD
 PROSLV   SPACE  4,50
**        PROSLV - PROCESS SLAVE ENTRY.
*
*         ENTRY  (A4) = ADDRESS OF LINE.
*                (X4) = ((A4)).
*


 PROSLV   SA1    MTIDS
          BX6    X6-X6
          SA6    A4               CLEAR LINE
          MX7    12
          BX6    X4*X7            EXTRACT ID
          PL     X1,PROSLV1       IF SLAVE
          SA6    A1               SET MASTER ID
          RJ     =XREREAD

 PROSLV1  SA2    A1+B1            LIMIT
          ERRNZ  MTLIM-MTIDS-1
          SB2    48
          BX2    -X2
          LX1    12
 PROSLV2  BX4    X1*X7
          ZR     X4,PROSLV3       IF NOT AN OLD SLAVE
          BX4    X6-X4
          ZR     X4,PROSLV4       IF AN OLD SLAVE
          SX2    X2+B1
          LX1    12
          SB2    B2-12
          NG     X2,PROSLV2       IF MORE ALLOWED
          EQ     =XRDMORE

 PROSLV3  BX6    X1+X6            ADD NEW SLAVE
          LX6    B2
          SA6    A1+
          EQ     =XRDMORE

 PROSLV4  BX1    -X7*X1           CLEAR OLD SLAVE
          SB3    B2-B1
          MX2    1
          AX2    B3
          BX6    X2*X1            LATER GUYS
          LX6    12               MOVE THEM UP
          BX1    -X2*X1           CLEAR LATER GUYS
          BX6    X1+X6            MERGE
          LX6    B2
          SA6    A1
          EQ     =XRDMORE
 NUCC     ENDIF
 PRIBRD   SPACE  4,32
***       PRIBRD - PRINT THE BOARD.
*
*         ENTRY  (A0) = ADDRESS OF BOARD.
*
*         CALLS  CARAGE, WRLINE.
*         USES   ALL REGISTERS, OLINE.
*



          ENTRY  PRIBRD
 PRIBRD   EQ     *+400000B
          RJ     =XBLDCMB
          SA1    PRIBRDA
          RJ     =XCARAGE
          SX5    =C* CURRENT BOARD POSITION.*
          RJ     =XWRLINE
          SX5    OLINE
          BX6    X6-X6
          SA6    X5+B1
          SA0    TABLE
 PRIBRD1  SA1    A0
          BX6    X1
          SA6    X5
          SA0    A0+B1
          RJ     =XWRLINE
          SB2    A0+0
          SB3    TABLE+9
          LT     B2,B3,PRIBRD1
          SA1    MSIDE
          SX5    PRIBRDB+2+X1
          RJ     =XWRLINE
          EQ     PRIBRD

 PRIBRDA  DATA   1H0

 PRIBRDB  DATA   38L BLACK (NUMBERS) TO MOVE.
          DATA   38L WHITE (LETTERS) TO MOVE.
 COMCMD   SPACE  4,30
***       COMCMD - PROCESS COMPLEMENT BOARD COMMAND.
*


 COMCMD   SA1    STATUS
          BX6    X1
          LX6    30
          SA6    A1               COMPLEMENT STATUS
          SB2    B0               INITIALIZE FIRST HALF POINTER
          SB3    70B              INITIALIZE SECOND HALF POINTER
          BX5    X5-X5
          SB4    10B              FIRST HALF END OF ROW POINTER
          SB6    B4               ROW INCREMENT
          SB7    B4+B4            OPPOSITE ROW DECREMENT
 COMCMD1  SA1    BOARD+B2
          SA2    BOARD+B3
          IX6    X5-X1            COMPLEMENT PIECE
          IX7    X5-X2
          SA6    A2               STORE IN REFLECTING LOCATION
          SA7    A1
          SB2    B2+B1            ADVANCE COLUMNS
          SB3    B3+B1
          LT     B2,B4,COMCMD1    IF NOT DONE WITH ROW
          SB4    B4+B6            ADVANCE END OF ROW POINTER
          SB3    B3-B7            PULL HIGH POINTER DOWN TWO ROWS
          LT     B2,B3,COMCMD1    CONTINUE IF HIGH POINTER STILL HI
 COMCMD2  JUMPUP NEWPOS
 CLOCMD   SPACE  4,11
***       CLOCMD - PROCESS CLOCK COMMAND.
*

 CLOCMD   RJ     =XRDRSFT         SKIP FIRST TOKEN
          RJ     =XRDRGNT         GET NEXT TOKEN
          RJ     =XPROCLO         PROCESS CLOCK CHANGES
          ZR     X5,CLOCMD1       IF NO ERROR
          RJ     =XERRMSG
 CLOCMD1  RJ     =XREREAD
 BLICMD   SPACE  4,50
***       BLICMD - PROCESS BLITZ COMMAND.
*
*         BLITZ,N  SETS TIMING PARAMETERS FOR A FAST GAME
*         AT A RATE OF N CPU SECONDS PER MOVE.  IF N IS
*         NOT SPECIFIED, IT IS ASSUMED TO BE 5.
*
*         CALLS  RDRSFT,RDRGNT,RDRDXB,CTIMPM,REREAD.


 BLICMD   RJ     =XRDRSFT         SKIP FIRST TOKEN
          RJ     =XRDRGNT         GET NEXT TOKEN
          ID     X6,BLICMD1       IF NO PARAMETER
          BX5    X7
          SB7    B1               ASSUME DECIMAL
          RJ     =XRDRDXB         CONVERT
          NZ     X4,BLICMD1       IGNORE IF JUNK
          ZR     X6,BLICMD1       IGNORE IF ZERO
          PL     X6,BLICMD2       OK IF POSITIVE AND NON-ZERO
 BLICMD1  SX6    5                SET DEFAULT RATE OF 5
 BLICMD2  SA6    RULTM1           SET THE VARIOUS LETS
          SA6    RULTM2
          BX6    X6-X6
          SA6    EXTRAT
          SA6    EXTRAS
          SA6    OVRHED
          SX6    100
          SA6    PERCNT
          SX6    1200
          SA6    RATEMY
          SA6    RATEYR
          SX6    60
          SA6    RULMV1
          SA6    RULMV2
          SX6    9999
          SA6    TQFRST
          SA6    TQNEXT
          SA1    SWICH            TURN ON TIMING SWITCH
          SX7    B1
          SA7    MDEPTH           MDEPTH = 1 = MINIMUM VALUE
          LX7    S.TIM
          BX6    X1+X7
          SA6    A1
          SX6    9999             SET JIGGLING FOR ENTIRE GAME
          SA6    JIGMVS
          SX6    16               SET 25 PERCENT MAXIMUM JIGGLE
          SA6    JIGSIZ
          RJ     =XINITTC         INITIALIZE TIME CONTROL
          RJ     =XREREAD
 PSLCMD   SPACE  4,88
***       PSLCMD - PROCESS PRINT SQUARE LIST COMMAND.
*
*         CALLS  RDRSFT, RDRGNT, PSLXPD, ERRMSG, WRLINE, REREAD.
*


 PSLCMD   RJ     =XRDRSFT         SKIP FIRST TOKEN
          RJ     =XRDRGNT         GET NEXT TOKEN
          ZR     X7,PSLCMD3       IF NONE
          SA2    PSLCMDA
          MX5    42
 PSLCMD1  ZR     X2,PSLCMD3       IF NOT FOUND
          BX3    X2-X7
          BX3    X3*X5
          ZR     X3,PSLCMD2       IF FOUND
          SA2    A2+1
          EQ     PSLCMD1

 PSLCMD2  SX4    X2               SAVE ADDRESS
          RJ     =XRDRAOD         ASSEMBLE OCTAL DIGITS
          LX6    1
          SX6    X6
          IX6    X6+X4            ADDRESS OF SQUARE LIST
          NG     X6,PSLCMD3       IF OUTSIDE FL
          SA2    MEMORY
          AX2    30
          IX2    X6-X2
          PL     X2,PSLCMD3       IF OUTSIDE FL
          SA6    PSLCMDB          SAVE ADDRESS
          SA2    X6+B1            SECOND WORD OF SQUARE LIST
          SA3    X6
          IX4    X2+X3
          ZR     X4,PSLCMD4       IF NO BITS SET
          LX2    2
          RJ     PSLXPD           EXPAND AND PRINT SECOND WORD
          SA2    PSLCMDB
          SA2    X2               FIRST WORD OF SQUARE LIST
          LX2    2+20
          RJ     PSLXPD           EXPAND AND PRINT FIRST WORD
          RJ     =XREREAD         RETURN

 PSLCMD3  SX5    =C* INVALID SQUARE LIST NAME.*
          RJ     =XERRMSG
          RJ     =XREREAD

 PSLCMD4  SX5    =C* NO BITS SET.*
          RJ     =XWRLINE
          RJ     =XREREAD

 PSLCMDA  BSS    0
          ECHO   1,X=(0,'SQRLST')
          VFD    42/0L!X,18/X
          DATA   0                END OF TABLE

 PSLCMDB  DATA   0                SQUARE LIST ADDRESS
 PSLXPD   SPACE  4,66
**        PSLXPD - EXPAND AND PRINT SQUARE LIST WORD.
*
*         ENTRY  (X2) = SQUARE LIST WORD SHIFTED 22 OR 2.
*
*         CALLS  WRLINE.
*         USES   OLINE.
*


 PSLXPD   EQ     *+400000B
          MX0    8
          LX0    1
          BX4    X2*X0            ROW 1
          LX2    10
```

```
          BX5    X2*X0            ROW 2
          LX2    10
          BX6    X2*X0            ROW 3
          LX2    10
          BX7    X2*X0            ROW 4
          MX0    -35
 PSLXPD1  IX4    X4-X0            MOVE ONE BIT
          IX5    X5-X0
          IX6    X6-X0
          IX7    X7-X0
          BX4    X4*X0
          BX5    X5*X0
          BX6    X6*X0
          BX7    X7*X0
          LX4    1
          LX5    1
          LX6    1
          LX7    1
          AX0    5
          NZ     X0,PSLXPD1       IF NOT DONE
          SA2    PSLXPDA
          IX7    X2+X7
          IX6    X2+X6
          SA7    OLINE
          BX7    X7-X7
          SA7    A7+B1
          SA6    A7+B1
          SA7    A6+B1
          IX6    X5+X2
          SA6    A7+B1
          SA7    A6+B1
          IX6    X4+X2
          SA6    A7+B1
          SA7    A6+B1
          SX5    OLINE
          RJ     =XWRLINE
          SX5    OLINE+2
          RJ     =XWRLINE
          SX5    OLINE+4
          RJ     =XWRLINE
          SX5    OLINE+6
          RJ     =XWRLINE
          EQ     PSLXPD           RETURN

 PSLXPDA  DATA   10H  --------
 PVACMD   SPACE  4,88
***       PVACMD - PROCESS PRINT VARIABLE COMMAND.
*


          ENTRY  PVACMD
 PVACMD   RJ     =XRDRSFT         SKIP FIRST TOKEN
          RJ     =XRDRGNT         GET NEXT TOKEN
          ZR     X7,PVACMD1       IF NONE
          SA2    PVACMDA
          RJ     =XRDRSXT         SEARCH TRANSLATION TABLE
          ZR     X6,PVACMD1       IF NOT FOUND
          SX4    X6               SAVE ADDRESS
          RJ     =XRDRAOD         ASSEMBLE OCTAL DIGITS
          SX6    X6
          IX6    X6+X4            ADDRESS OF VARIABLE
          NG     X6,PVACMD1       IF OUTSIDE FL
          SA2    MEMORY
          AX2    30
          IX2    X6-X2
          PL     X2,PVACMD1       IF OUTSIDE FL
          SA5    X6               WORD
          BX1    X6               ADDRESS
          RJ     =XRDRCOD         CONVERT ADDRESS
          LX6    12
          SA6    OLINE
          BX1    X5
          RJ     =XRDRWOD         CONVERT CONTENTS
          SA6    A6+B1
          SA7    A6+B1
          BX6    X6-X6
          SA6    A7+B1
          SX5    OLINE
          RJ     =XWRLINE
          RJ     =XREREAD

 PVACMD1  SX5    =C* INVALID VARIABLE NAME.*
          RJ     =XERRMSG
          RJ     =XREREAD

 PVACMDA  BSS    0
          ECHO   1,X=(0,'OTHERS')
          XLAT   X,X
          DATA   0                END OF TABLE
 PINCMD   SPACE  4,35
***       PINCMD - PROCESS PRINT INFORMATION COMMAND.
*


 PINCMD   RJ     =XRDRSFT         SKIP FIRST TOKEN
          RJ     =XRDRGNT         GET NEXT TOKEN
          ZR     X7,PINCMD1       IF NONE
          SA2    PINCMDA
          RJ     =XRDRSXT         SEARCH TRANSLATION TABLE
          ZR     X6,PINCMD1
          SA1    PINCMDB
          SB7    X6
          BX7    X1
          SA7    B7               FAKE RJ
          JP     B7+1

 PINCMD1  SX5    =C* INVALID INFORMATION KEY.*
          RJ     =XERRMSG
 PINCMD2  RJ     =XREREAD

 PINCMDA  XLAT   SU,=XPRISUM
          XLAT   NC,=XPRINDC
          XLAT   VA,=XPRIPRV
          XLAT   WE,=XPRIWEI
          XLAT   MO,=XPRIMOV
          DATA   0

 PINCMDB  EQ     PINCMD2
 ECSCMD   SPACE  4,32
***       ECSCMD - PROCESS ECS COMMAND.
*


 ECSCMD   RJ     =XRDRSFT         SKIP FIRST TOKEN
          RJ     =XRDRGNT         GET NEXT TOKEN
          SA2    ECSCMDA
          RJ     =XRDRSKT         SEARCH KEYWORD TABLE
          SA3    MEMORY+1
          ZR     X3,ECSCON        IF OFF, TURN ON
*         EQ     ECSCOF           ELSE TURN OFF
 ECSCOF   SPACE  4,10
**        ECSCOF - TURN ECS OFF.
*


 ECSCOF   SA3    MEMORY+1
          ZR     X3,ECSIGN        IF ALREADY OFF, IGNORE
          RJ     =XECSOFF         TURN OFF ECS
          SX5    =C* ECS TURNED OFF.*
          RJ     =XWRLINE
          EQ     COMCMD2          GENERATE LEGAL MOVES
 ECSCON   SPACE  4,10
**        ECSCON - TURN ECS ON.
*


 ECSCON   SA3    MEMORY+1
          NZ     X3,ECSIGN        IF ALREADY ON, IGNORE
          RJ     =XECSONR
          SX5    =C* ECS TURNED ON.*
          RJ     =XWRLINE
          RJ     =XREREAD
 ECSIGN   SPACE  4,8
**        ECSIGN - IGNORE ECS CHANGE.
*


 ECSIGN   SX5    =C* ECS COMMAND IGNORED.*
          RJ     =XERRMSG
          RJ     =XREREAD

 ECSCMDA  CMND   OF,ECSCOF
          CMND   ON,ECSCON
          DATA   0
 ECSONR   SPACE  4,35
***       ECSONR - ENABLE ECS INSTRUCTIONS.
*
*         USES   ALL REGISTERS.
*         CALLS  SYS=, MEM., PLGECS.
*


 ECSONR   EQ     *+400000B
          SX6    EC.TRA+401B      GET ENOUGH ECS TO START
          LX6    30
          SA6    MEMORY+1
          SYSTEM MEM,RECALL,A6,1
          SX1    =XEREADER
          RJ     =XPLGECS         PLUG (0,0) OVERLAY
          SX1    =XECHES20
          RJ     =XPLGECS         PLUG (2,0) OVERLAY
          SA2    MEMORY+1
          AX2    30               CURRENT FE
          SA1    A2+B1            FWA FAKE ECS
          SA4    A1+B1            LWA+1 FAKE ECS
          BX6    X1
          SA6    A4               DEALLOCATE FAKE ECS
          IX5    X4-X1            LENGTH OF FAKE ECS
          SX0    B1               DONT USE LAST ECS WORD ON CYBER 170
          IX2    X2-X0            TO PREVENT MODE ERROR.
          IX6    X2-X5
          AX6    59
          BX2    X6*X2
          BX5    -X6*X5
          BX6    X2+X5            MIN(FE, FAKE FE)
          SB2    X6               LENGTH
          SA0    X1               CM FWA
          BX0    X0-X0            ECS FWA = 0
          SB3    1777B            MAXIMIMUM 7600 TRANSFER
          GT     B3,B2,ECSONR2    IF ONLY SHORT BLOCK
 ECSONR1  WE     B3               WRITE LONG BLOCK
          RJ     =XECWERR         IF ERRORS
          SA0    A0+B3            ADVANCE ADDRESSES
          SX0    X0+B3
          SB2    B2-1777B         DECREMENT WORD COUNT
          LE     B3,B2,ECSONR1    IF MORE LONG BLOCKS
 ECSONR2  SB3    B2+
          WE     B3               WRITE SHORT BLOCK
          RJ     =XECWERR
          RJ     =XMEM.           RETURN EXCESS CM
          EQ     ECSONR           RETURN
 MSGCMD   SPACE  4,8
***       MSGCMD - PROCESS MESSAGE COMMAND.
*


 MSGCMD   BX1    X1-X1
          RJ     =XGAMSCR
          RJ     =XREREAD
 BLDCMB   SPACE  4,37
*CALL     COMCDXB
 RDRDXB   EQU    DXB
          ENTRY  RDRDXB

          EXT    DCB=
          EXT    WTX=
          EXT    LCB=
          EXT    RDX=
*CALL     COMCWTH
          ENTRY  WTH=
*CALL     COMCRDH
          ENTRY  RDH=
          TITLE
***       BLDCMB - BUILD CM DISPLAY BOARD.
*
*         ENTRY  (A0) = ADDRESS OF BOARD.
*
*         EXIT   (TABLE) = ABBREVIATED DISPLAY CODE POSITION.
*
*         USES   X1, X2, X6, X7, A1, B2, B3.
*


          ENTRY  BLDCMB
 BLDCMB   EQ     *+400000B
          SA1    A0+0
          SX2    1R0
          SB3    7                OUTER LOOP COUNT
          SX7    2R1              ROW NUMBER
 BLDCMB1  SB2    7                INNER LOOP COUNT
          BX6    X7
 BLDCMB2  ZR     X1,BLDCMB3       IF EMPTY SQUARE
          PL     X1,BLDCMB4       IF WHITE
          IX1    X2-X1            COMPUTE NUMBER
```

```
          EQ     BLDCMB4

 BLDCMB3  SX1    1R-
 BLDCMB4  LX6    6
          BX6    X6+X1
          SA1    A1+B1             GET NEXT SQUARE
          SB2    B2-B1             COLUMN
          PL     B2,BLDCMB2        IF NOT EIGHT COLUMNS PROCESSED
          SA6    TABLE+B3          PUT PIECES ON TABLE
          SX7    X7+100B           ADVANCE DISPLAY CODE RANK
          SB3    B3-B1             DECREMENT RANK
          PL     B3,BLDCMB1        IF NOT ENTIRE BOARD PROCESSED
          EQ     BLDCMB            ELSE RETURN
 RDRPNC   TITLE  RDRPNC - PROCESS NEXT CHARACTER.
***       RDRPNC - PROCESS NEXT CHARACTER.
*
*         ENTRY  (X0) = 7777 7777 7777 7777 7700
*                (X1) = CURRENT INPUT WORD.
*                (X1) = ((A1)).
*                (B2) = CURRENT INPUT CHARACTER SHIFT COUNT.
*                (B7) = SYNTAX TABLE ADDRESS.
*                SYNTAX TABLE FORMAT:
*                WORD 1:
*                1/0,1/A,1/B,1/C,...,1/>
*                WHERE X IS SET IF X IS A VALID CHARACTER.
*                WORD 2:
*                30/SEMANTICS ROUTINE ADDRESS, 30/NEXT SYNTAX TABLE
*                TABLE IS TERMINATED BY A ZERO WORD.
*
*         EXIT   TO SEMANTICS ROUTINE, IF THE NEXT CHARACTER IS VALID.
*                (B3) = THE CHARACTER.
*                (X6) = THE NUMBER OF LOWER VALUED VALID CHARACTERS
*                       IN THE SAME ENTRY.
*                A1, X1, B2, AND B7 ARE UPDATED.
*
*                TO CALLER, IF THE NEXT CHARACTER IS PERIOD, SEMI-COLON,
*                OR 6 ZERO BITS.
*                (X2) = THE CHARACTER.
*
*                INVALID CHARACTERS ARE IGNORED.
*
*         USES   X2, X3, A2, A3, AND B6.
*



          ENTRY  RDRPNC
 RDRPNC   EQ     *+400000B
 RDRPNC1  AX2    X1,B2             EXTRACT NEXT CHARACTER
          BX2    -X0*X2
          BX3    X0+X2
          SB3    X2                THE CHARACTER
          ZR     X2,RDRPNC         IF ZERO CHARACTER, RETURN
          ZR     X3,RDRPNC         IF SEMI-COLON, RETURN
          SX3    X2-1R.
          ZR     X3,RDRPNC         IF PERIOD, RETURN
          SB2    B2-6              ADVANCE TO NEXT CHARACTER
          PL     B2,RDRPNC2        IF NOT NEW WORD
          SB2    54
          SA1    A1+B1
 RDRPNC2  SB6    B7                INITIALIZE SYNTAX TABLE SCAN
 RDRPNC3  SA2    B6
          LX3    X2,B3
          ZR     X2,RDRPNC1        IF END OF TABLE, IGNORE
          NG     X3,RDRPNC4        IF VALID
          SB6    B6+2
          EQ     RDRPNC3           TRY NEXT ENTRY

*         VALID CHARACTER.

 RDRPNC4  SA3    A2+B1
          SB7    X3                NEXT TABLE ADDRESS
          AX3    30
          SB6    X3                SEMANTICS PROCESSOR
          MX6    1
          AX6    X6,B3             CONVERT BIT TO ORDINAL
          BX6    X6*X2
          CX6    X6
          SX6    X6-1
          JP     B6                PROCESS SEMANTICS
 RDRSFT   TITLE  RDRSFT - SKIP FIRST TOKEN.
***       RDRSFT - SKIP FIRST TOKEN.
*
*         ENTRY  (ILINE) = INPUT LINE.
*
*         EXIT   (A1) = ADDRESS OF INPUT WORD.
*                (X1) = ((A1))
*                (B2) = NEXT CHARACTER SHIFT COUNT.
*                (X0) = 7777 7777 7777 7777 7700
*
*         USES   X6, X7, B4.
*         CALLS  RDRGNT.
*



          ENTRY  RDRSFT
 RDRSFT   EQ     *+400000B
          SA1    ILINE
          SB2    54
          MX0    54
          RJ     =XRDRGNT          GET NEXT TOKEN
          EQ     RDRSFT            RETURN
 RDRSXT   SPACE  4,88
***       RDRSXT - SEARCH TRANSLATION TABLE.
*
*         ENTRY  (A2) = FWA OF TABLE.
*                (X2) = ((A2))
*                (X7) = KEY.
*                TABLE FORMAT:
*                36/KEY
*                6/LENGTH OF KEY-1
*                18/VALUE
*         TERMINATED BY ZERO WORD.
*
*         EXIT   (X6) = TRANSLATION.
*                (X6) = 0, IF NOT FOUND.
*
*         USES   A2, X2, X3, X4, X5, B4.
*



          ENTRY  RDRSXT
 RDRSXT   EQ     *+400000B
          MX4    1
          MX5    36
 RDRSXT1  SX6    X2
          ZR     X2,RDRSXT         IF END OF TABLE
          BX3    -X5*X2
          AX3    18
          SB4    X3                MASK SIZE
          BX3    X2-X7             COMPARE
          AX2    X4,B4             BUILD MASK
          BX2    X2*X3
          ZR     X2,RDRSXT         IF SAME
          SA2    A2+B1
          EQ     RDRSXT1           LOOP

          END    'END'
 INITAL
          IDENT  INITAL
          SST
 INITAL   TITLE  INITAL - INITIALIZE BOARD FOR NEW GAME.
*CALL     LET
*CALL     TLET
*CALL     IOCOM
*CALL     BOARD
*CALL     SQRLST
 EINITAL  ECERTB
 INITAL   SPACE  4,88
***       INITAL - INITIALIZE BOARD FOR NEW GAME.
*
*         EXIT   (BOARD) = INITIAL BOARD POSITION.
*                TO LSTMOV.
*



          ENTRY  INITAL
 INITAL   RJ     =XGMPRNT          PRINT GAME SCORE
          SA1    INITALA
          SB2    BOARD
 INITAL2  BX6    X1
          LX6    54
          AX6    54
          SA6    B2
          SB2    B2+B1
          AX1    6
          NZ     X1,INITAL2        IF NOT WHOLE BOARD INITIALIZED
          SA1    A1+B1
          NZ     X1,INITAL2
          SA1    A1+B1
          BX6    X1
          SA6    B2                STATUS
          ERRNZ  BOARD+64-STATUS
          BX7    -X7+X7
          SB2    128-1
 INITAL3  SA7    MBHSH+B2
          SB2    B2-1
          PL     B2,INITAL3
          SA1    INITALB
          BX6    X1
          SA6    MOVMS
          SB2    2
 INITAL4  SA1    A1+B1
          BX6    X1
          SA6    A6+B1
          SB2    B2-B1
          PL     B2,INITAL4
          SX0    EC.PSH            CLEAR PAWN STRUCTURE HASH TABLE
          SB2    L.PSH/128-1
          ERRNZ  L.PSH/128*128-L.PSH
          SB3    128
          SA0    MBHSH
 INITAL5  WE     B3
          RJ     =XECWERR
          SB2    B2-B1
          SX0    X0+B3
          ERRPL  EC.PSH+L.PSH-377777B
          PL     B2,INITAL5        IF NOT DONE CLEARING TABLE
          BX6    X6-X6
          SA6    MOVNUM
          SA6    GAMMS
          SA6    CNTMV
          SA6    TIMMV
          SA6    TIMTO
          SA6    GCLOCK
          SA6    TIMPS
          SA6    ITERS
          SA1    TQFRST
          BX7    X1
          SA7    TQNEXT
          RJ     =XINITTC          INITIALIZE TIME CONTROL
          SX5    MOVMS
          RJ     =XWRLINE
          RTIME  RANDOM            SET NEW RANDOM SEED
          SA1    RANDOM
          MX6    60-15
          BX6    -X6*X1            ONLY TAKE BOTTOM 15 BITS
          SA6    A1
          SA1    SWICH
          LX1    59-S.EXP
          PL     X1,INITAL6        IF LIBRARY CREATION
          RTIME  INITALC           GET A RANDOM NUMBER
          SA1    INITALC
          MX0    56
          BX2    -X0*X1            RANDOM NUMBER (0,15D)
          LX1    55
          AX1    59                RANDOM + OR - 0
          SX3    X2-10             RANDOM (-10,5)
          BX4    X3
          AX4    59
          BX6    X4*X2
          BX5    -X4*X3
          BX6    X6+X5             RANDOM (0,9)
          IX5    X6+X6             *2
          IX6    X5+X6             *3
          IX6    X6+X6             *6
          SB2    X6
          SA3    INITALD           MASKS
          BX1    X1-X3
          MX0    54
          LX6    X1,B2
          BX6    -X0*X6
          SA6    MSMASK            SET RANDOM MASK
 INITAL6  JUMPUP NEWPOS

 INITALA  VFD    12/-0,48/0203040605040302B
          VFD    12/-0,48/0101010101010101B
          VFD    12/-0,48/0
          VFD    12/-0,48/0
          VFD    12/-0,48/0
          VFD    12/-0,48/0
          VFD    12/-0,48/7676767676767676B
          VFD    12/-0,48/7574737172737475B
          VFD    60/0
          VFD    60/4000420000000422000B
```

```
 INITALB  DATA   38C ENTER MOVE OR TYPE GO.

 INITALC  BSS    1                RTIME STATUS WORD
 INITALD  CON    7064626154525146454 3B RANDOM 3 OUT OF 6 MASKS
 INITTC   SPACE  4,50
***       INITTC - INITIALIZE TIME CONTROL.
*
*         ENTRY  (RULMV1) = NUMBER OF MOVES IN FIRST PERIOD.
*                (RULMV2) = NUMBER OF MOVES IN SUBSEQUENT TIME PERIODS.
*                (RULTM1) = NUMBER OF MINUTES IN FIRST TIME PERIOD.
*                (RULTM2) = NUMBER OF MINUTES IN SUBSEQUENT TIME PERIODS
*                (EXTRAT) = TIME CONTROL BUFFER.
*                (EXTRAS) = SACRED MINIMUM EXTRAT.
*                (MOVNUM) = NUMBER OF MOVES ALREADY PLAYED.
*                (OVRHED) = NUMBER OF SECONDS OVERHEAD PER MOVE.
*                (TQFRST) = FIRST TIME QUESTION MOVE NUMBER.
*                (TPMRMX) = 100B * MAX MULT OF MTMPM TO USE.
*                (TRATLO) = 100B * (MIN MS THIS MOVE)/TIMPM.
*                (TPMOKR) = RATIO USED TO COMPUTE OTMPM.
*
*         EXIT   (TQNEXT) = (TQFRST).
*                (ATMPM) = AVERAGE TIME PER MOVE.
*                (NTRLO) = NOMINAL MINIMUM TIME RATIO.
*                (MTMPM) = MAXIMUM TIME PER MOVE (MSEC).
*                (OTMPM) = OK (ACCEPTABLE MIN) TIME-PER-MOVE.
*
*         CALLS  CTIMPM, GCLPST.
*         USES   A1, A2, A3, A6, A7, X1, X2, X3, X6, X7, B2.
*


          ENTRY  INITTC
 INITTC   EQ     *+400000B
          SA1    TQFRST
          BX6    X6-X6
          BX7    X1
          SA7    TQNEXT
          PX7    X6
          SA7    MTMPM             SET MTMPM INFINITE
          RJ     =XCTIMPM          COMPUTE TIME PER MOVE
          SA6    ATMPM             AVERAGE TIME PER MOVE
          SA2    TPMRMX
          IX7    X2*X6             COMPUTE MAXIMUM TIME PER MOVE
          AX7    6
          SA1    TRATLO
          BX6    X1
          SA6    A6+B1             NOMINAL MINIMUM TIME RATIO
          ERRNZ  NTRLO-ATMPM-1
          SA7    A6+B1             MTMPM
          ERRNZ  MTMPM-NTRLO-1

*         COMPUTE OTMPM FOR POSSIBLE EXTRAT RAIDING.

          SA1    RULMV2            COMPUTE A TIMPM FOR 2ND CONTROL
          SA3    RULTM2
          RJ     =XGCLPST
          SA1    TIMPM             THE JUST-COMPUTED TIME-PER-MOVE
          IX2    X6-X1             GET MINIMUM OF THAT AND 2ND CONTROL
          NG     X2,INITTC1        USE 2ND CONTROL IF FIRST IS BIGGER
          BX6    X1                ELSE USE FIRST CONTROL
 INITTC1  SA1    TPMOKR            NOW CUT A FRACTION OFF
          IX6    X1*X6
          AX6    6
          SA6    OTMPM             SET MINIMUM OK (ACCEPTABLE) TIMPM
          EQ     INITTC            RETURN

          END
CRECMD
          IDENT  CRECMD
          SST
 CRECMD   TITLE  CRECMD - PROCESS BOARD MODIFICATION COMMANDS.
*CALL     BOARD
*CALL     SQRLST
*CALL     IOCOM
*CALL      LET
 CRECMD   SPACE  4,88
***       CRECMD - PROCESS CREATE COMMAND.
*
          ENTRY  CRECMD
 CRECMD   RJ     =XRDRSFT          SKIP FIRST TOKEN
          SB7    CRECMDB
 CRECMD1  RJ     =XRDRPNC          PROCESS NEXT CHARACTER
*                                  (RETURNS ONLY ON END OF CARD)
 CRECMD2  BX1    X1-X1
          RJ     =XGAMSCR
          JUMPUP NEWPOS

 CRECMDA  DATA   0                 +0 IF WHITE PIECE, -0 IF BLACK
          DATA   0                 TYPE OF PIECE
          DATA   3                 3 IF QUEEN SIDE, 4 IF KING SIDE
          DATA   0                 FILE (0 IF CENTER, 1 IF BISHOP, ETC)

 CRECMDB  CHARS  LD,*,CRELOD
          CHARS  XPRNBQK,*,CREPCE
          CHARS  KQ,*,CREKOQ
          CHARS  RNB,*,CRERNB
          CHARS  12345678,CRECMDB,CRERNK
          DATA   0
 CRELOD   SPACE  4,9
**        CRELOD - LIGHT OR DARK PIECE.
*

 CRELOD   LX6    59
          AX6    59
          BX6    -X6               +0 IF L, -0 IF D
          SA6    CRECMDA
          EQ     CRECMD1
 CREPCE   SPACE  4,16
**        CREPCE - PIECE TYPE.
*

 CREPCE   SA3    CREPCEA+X6        TYPE
          BX6    X3
          SA6    CRECMDA+1
          EQ     CRECMD1

 CREPCEA  CON    BISH,KING,NITE,PAWN,QUEN,ROOK,0
 CREKOQ   SPACE  4,10
**        CREKOQ - KING OR QUEEN SIDE.
*

 CREKOQ   SX7    4
          IX6    X7-X6             4 IF K, 3 IF Q
          BX7    X7-X7
          SA6    CRECMDA+2         SIDE
          SA7    A6+B1             FILE
          EQ     CRECMD1
 CRERNB   SPACE  4,8
**        CRERNB - FILE.
*

 CRERNB   SX6    X6+1
          SA6    CRECMDA+3
          EQ     CRECMD1
 CRERNK   SPACE  4,20
**        CRERNK  - RANK.
*

 CRERNK   SA3    MSIDE
          SX5    70B
          BX5    X3*X5
          LX6    3
          BX5    X6-X5             COMPLEMENT IF BLACK TO MOVE
          SA2    CRECMDA           COLOR
          SA3    A2+B1             TYPE
          BX6    X2-X3             PIECE TO CREATE
          SX6    X6                REMOVE -0
          SA2    A3+B1             SIDE
          SA3    A2+B1             FILE
          BX2    X2-X3             FINAL FILE
          BX2    X2-X5             SQUARE
          SA6    BOARD+X2
          EQ     CRECMD1
 BOACMD   SPACE  4,88
***       BOACMD - PROCESS BOARD COMMAND.
*


          ENTRY  BOACMD
 BOACMD   RJ     =XRDRSFT          SKIP FIRST TOKEN
          SB7    BOACMDB
          SB5    B0                ROW NUMBER * 8
          SB4    B0                COLUMN NUMBER
 BOACMD1  RJ     =XRDRPNC          PROCESS NEXT CHARACTER
*                                  (RETURNS ONLY ON END OF CARD)
          EQ     CRECMD2           JUMPUP NEWPOS

 BOACMDA  DATA   0                 COLOR

 BOACMDB  CHARS  LD,*,BOALOD
          CHARS  XPRNBQK,BOACMDB,BOAPCE
          CHARS  12345678,BOACMDB,BOASKP
          CHARS  (/,),BOACMDB,BOANXT
          DATA   0
 BOALOD   SPACE  4,9
**        BOALOD - LIGHT OR DARK PIECE.
*

 BOALOD   LX6    59
          AX6    59
          BX6    -X6
          SA6    BOACMDA
          EQ     BOACMD1
 BOAPCE   SPACE  4,14
**        BOAPCE - PIECE TYPE.
*

 BOAPCE   SA3    CREPCEA+X6        TYPE
          SA4    BOACMDA           COLOR
          BX6    X3-X4
          SX6    X6                REMOVE -0
          SB6    B5+B4             SQUARE NUMBER
          SA6    BOARD+B6
          BX6    X6-X6
          EQ     BOASKP
 BOANXT   SPACE  4,6
**        BOANXT - ADVANCE TO NEXT ROW.
*

 BOANXT   SX6    8
*         EQ     BOASKP
 BOASKP   SPACE  4,12
**        BOASKP - SKIP N SQUARES.
*

 BOASKP   SB5    B5+X6
          SB5    B5+B1
          SB6    8
          LT     B5,B6,BOACMD1     IF NOT NEW ROW
          SB4    B4+B6
          SB5    B0
          SB6    64
          LT     B4,B6,BOACMD1     IF NOT DONE
          EQ     CRECMD2
 PURCMD   SPACE  4,88
***       PURCMD - PROCESS PURGE COMMAND.
*


          ENTRY  PURCMD
 PURCMD   SB2    63
          SX6    0
 PURCMD1  SA6    BOARD+B2
          SB2    B2-1
          PL     B2,PURCMD1        IF NOT ENTIRE BOARD CLEARED
          MX6    1
          SA6    STATUS            SET WHITE TO MOVE
          EQ     CRECMD2           JUMPUP NEWPOS
 SAVCMD   SPACE  4,24
***       SAVCMD - PROCESS SAVE COMMAND.
*


          ENTRY  SAVCMD
 SAVCMD   SA0    SAVCMDA
          SA3    STATUS
          SB3    BOARD
          RJ     =XMSPACK          PACK CURRENT POSITION
          RJ     =XRDRSFT          SKIP FIRST TOKEN
          RJ     =XRDRGNT          GET NEXT TOKEN
          ZR     X7,SAVCMD1        IF NONE
          SA1    MOVNUM
          MX6    42
          BX7    X6*X7
          BX7    X7+X1
          RJ     =XMSREPL          REPLACE POSITION
          BX1    X1-X1
          RJ     =XGAMSCR
          RJ     =XMSCLOS
          RJ     =XREREAD          RETURN

 SAVCMD1  SX5    =C* POSITION NAME MISSING.*
          RJ     =XERRMSG
```

```
          RJ     =XREREAD

 SAVCMDA  BSS    6                PACKED POSITION
 SETCMD   SPACE  4,23
***       SETCMD - PROCESS SETUP COMMAND.
*


          ENTRY  SETCMD
 SETCMD   RJ     =XRDRSFT         SKIP FIRST TOKEN
          RJ     =XRDRGNT         GET NAME
          ZR     X7,SAVCMD1       IF NOT SPECIFIED
          SX6    SAVCMDA
          RJ     =XMSSRCH         SEARCH FOR POSITION
          ZR     X6,SETCMD1       IF NOT FOUND
          MX6    42
          BX6    -X6*X7
          SA6    MOVNUM           STORE MOVE NUMBER
          RJ     =XMSUNPK         UNPACK POSITION
          EQ     CRECMD2          JUMPUP NEWPOS

 SETCMD1  SX5    =C* POSITION NOT FOUND.*
          RJ     =XERRMSG
          RJ     =XREREAD
 STACMD   SPACE  4,30
***       STACMD - PROCESS STATUS COMMAND.
*


          ENTRY  STACMD
 STACMD   RJ     =XRDRSFT         SKIP FIRST TOKEN
 STACMD1  RJ     =XRDRGNT         GET NEXT TOKEN
          ZR     X7,CRECMD2       IF NO MORE
          SA2    STACMDB
          RJ     =XRDRSKT         SEARCH KEYWORD TABLE
          SX5    =C* ILLEGAL STATUS TYPE.*
          RJ     =XERRMSG
          EQ     STACMD1

 STACMDA  DATA   0                0 IF BLACK, 30 IF WHITE

 STACMDB  CMND   CL,STACLR        CLEAR
          CMND   L,STALIT         SET WHITE
          CMND   D,STADRK         SET BLACK
          CMND   000,STACQS       CASTLE QUEEN SIDE
          CMND   OOO,STACQS       CASTLE QUEEN SIDE
          CMND   OO,STACKS        CASTLE KING SIDE
          CMND   00,STACKS        CASTLE KING SIDE
          CMND   M,STAMOV         TO MOVE
          CMND   EP,STAENP        EN PASSANT
          CMND   C,STACNT         50 MOVE COUNT
          CMND   N,STACNT         50 MOVE COUNT
          DATA   0                END OF TABLE
 STACLR   SPACE  4,8
**        STACLR - CLEAR STATUS.
*


 STACLR   MX6    1
          SA6    STATUS
          EQ     STACMD1
 STALIT   SPACE  4,8
**        STALIT - SET WHITE.
*


 STALIT   SX6    30
          SA6    STACMDA
          EQ     STACMD1
 STADRK   SPACE  4,8
**        STADRK - SET BLACK.
*


 STADRK   BX6    X6-X6
          SA6    STACMDA
          EQ     STACMD1
 STACQS   SPACE  4,7
**        STACQS - ALLOW CASTLE QUEEN SIDE.
*


 STACQS   SX6    420B
          EQ     STACAS           SET CASTLE STATUS
 STACKS   SPACE  4,7
**        STACKS - ALLOW CASTLE KING SIDE.
*


 STACKS   SX6    22B
*         EQ     STACAS           SET CASTLE STATUS
 STACAS   SPACE  4,14
**        STACAS - SET CASTLE STATUS.
*
*         ENTRY  (X6) = CASTLE STATUS BITS.
*


 STACAS   SA3    STACMDA
          SB3    X3+9
          SA3    STATUS
          LX6    X6,B3
          BX6    X6+X3
          SA6    A3
          EQ     STACMD1
 STAMOV   SPACE  4,16
**        STAMOV - SET SIDE TO MOVE.
*


 STAMOV   MX7    1
          SA3    STACMDA
          SB3    X3
          SA3    STATUS
          BX6    -X7*X3
          LX7    30
          BX6    -X7*X6
          LX7    X7,B3
          BX6    X6+X7
          SA6    A3
          EQ     STACMD1
 STAENP   SPACE  4,35
**        STAENP - SET EP BIT.
*


 STAENP   RJ     =XRDRGNT         GET NEXT TOKEN
          ZR     X7,STAENP1       IF NONE
          SA2    STAENPA
          RJ     =XRDRSXT         SEARCH TRANSLATION TABLE
          ZR     X6,STAENP1       IF INVALID
          SA3    STACMDA
          SB3    X3+18
          SA3    STATUS
          LX6    X6,B3
          MX7    -9
          LX7    X7,B3
          BX3    X7*X3
          BX6    X6+X3
          SA6    A3
          EQ     STACMD1

 STAENP1  SX5    =C* INVALID EP FILE.*
          RJ     =XERRMSG
          EQ     STACMD1

 STAENPA  XLAT   KR,002B
          XLAT   QR,400B
          XLAT   KN,004B
          XLAT   QN,200B
          XLAT   KB,010B
          XLAT   QB,100B
          XLAT   K,0020B
          XLAT   Q,0040B
          DATA   0
 STACNT   SPACE  4,27
**        STACNT - SET 50 MOVE RULE COUNT.
*


 STACNT   RJ     =XRDRGNT         GET NUMBER
          ZR     X7,STACNT1       IF NONE
          SB7    B1
          BX5    X7
          RJ     =XRDRDXB         CONVERT
          NZ     X4,STACNT1       IF INVALID
          SX7    X6-50D
          PL     X7,STACNT1       IF TOO HIGH
          NG     X6,STACNT1       IF TOO LOW
          SA3    STATUS
          MX7    -9
          BX3    X7*X3
          BX3    X3+X6
          LX6    30
          LX7    30
          BX3    X7*X3
          BX6    X6+X3
          SA6    A3
          EQ     STACMD1

 STACNT1  SX5    =C* INVALID 50 MOVE RULE COUNT.*
          RJ     =XERRMSG
          EQ     STACMD1
          END
YRMOVE
          IDENT  YRMOVE
          SST
          LIST   F,X
 YRMOVE   TITLE  YRMOVE - PROCESS PLAYERS MOVE.
*CALL     LET
*CALL TLET
*CALL     IOCOM
*CALL     SQRLST
*CALL DEBUG
*CALL     BOARD
 YRMOVE   TITLE  SYNTAX ROUTINES FOR PLAYERS MOVE PROCESSING.
***       YRMOVE - PROCESS PLAYERS MOVE.
*
*         ENTRY  (MOVES) = PLAYERS LEGAL MOVES.
*                (LINDX) = LWA+LE.MOV OF MOVES ARRAY.
*                (BOARD) = CURRENT POSITION.
*
*         EXIT   TO YPMOVE.
*                (NXTMV) = PLAYERS MOVE.
*
*         CALLS  READER.
*         USES   ALL REGISTERS.
*


          ENTRY  YRMOVE
 YRMOVE   SA1    SWICH
          LX1    59-S.REC
          PL     X1,YRMOVE1       IF NOT RECORDING
          SA0    YUMOVEA          BUFFER AREA
          SA3    STATUS
          SB3    BOARD
          RJ     =XMSPACK         PACK UP POSITION
          SA1    MOVNUM
          RJ     =XRDRCDD         CONVERT
          MX7    1
          SB2    B2-B1
          AX7    X7,B2
          BX7    X4*X7
          SA1    A1
          BX7    X1+X7
          RJ     =XMSREPL         REPLACE POSITION ON LIBRARY
 YRMOVE1  SA1    SWICH
          LX1    59-S.LST
          PL     X1,YRMOVE2       IF NOT LISTING
          SA0    BOARD
          RJ     =XPRIBRD         PRINT BOARD
          SX6    0
          SA6    NXTMV
 YRMOVE2  SA0    BOARD
          RJ     =XBLDCMB
          SX6    O.EMOV           *ENTER MOVE.*
          RJ     =XREADER


          ENTRY  YUMOVE
 YUMOVE   BX6    X6-X6
          SA6    YUMOVEA          MOVE MASK
          SA6    A6+B1            MOVE VALUE
          SA6    A6+B1
          SA6    A6+B1
          SA6    A6+B1
          SX7    60
          SA7    A6+B1            FROM SQUARE SHIFT COUNT
          SA6    A7+B1            FROM SQUARE VALUE
          SA7    A6+B1            TO SQUARE SHIFT COUNT
          SA6    A7+B1            TO SQUARE VALUE
          SA6    ALGBRAA          LEADING -B- FLAG
          SA1    ILINE
          SB2    54
```

```
          MX0    54
          SB7    YUMOVEB
 YUMOVE1  RJ     =XRDRPNC          PROCESS NEXT CHARACTER
*                                  (RETURNS ONLY ON END OF CARD)
          SA1    YUMOVEA           MOVE MASK
          SA5    LINDX             LWA+LE.MOV MOVES ARRAY
          SA2    A1+B1
          SB7    X5
          SB6    MOVES
          BX7    X7-X7             INDICATE NO MOVE FOUND YET

*         SEARCH MOVES ARRAY FOR EXACTLY ONE HIT.

 YUMOVE2  GE     B6,B7,YUMOVE5     IF END OF ARRAY
          SA5    B6                A MOVE
          BX6    X2-X5
          BX6    X6*X1
          ZR     X6,YUMOVE3        IF FORMAT MATCHES
          SA4    A2+B1
          BX6    X6-X4             CHECK REFLECTIONS
          ZR     X6,YUMOVE3
          SA4    A4+B1
          BX6    X6-X4
          ZR     X6,YUMOVE3
          SA4    A4+B1
          BX6    X6-X4
          NZ     X6,YUMOVE4
 YUMOVE3  SA4    YUMOVEA+5         FROM SQUARE SHIFT COUNT
          NG     X5,YUMOVE4        IF ILLEGAL
          BX6    -X0*X5            TO SQUARE
          AX5    6
          BX5    -X0*X5            FROM SQUARE
          SB3    X4
          SA5    BOARD+X5          PIECE ON FROM SQUARE
          SA4    A4+B1             REQUIRED PIECE TYPE
          BX5    X4-X5
          AX5    X5,B3
          SA4    A4+B1             TO SQUARE SHIFT COUNT
          NZ     X5,YUMOVE4        IF WRONG FROM PIECE TYPE
          SA5    BOARD+X6          PIECE ON TO SQUARE
          SB3    X4
          SA4    A4+B1             REQUIRED CAPTURED PIECE TYPE
          BX5    X5-X4
          AX5    X5,B3
          NZ     X5,YUMOVE4        IF INVALID TO PIECE

*         ACCEPTABLE MOVE.

          NZ     X7,YUMOVE6        IF AMBIGUOUS MOVE
          SA4    B6
          BX7    X4                SAVE MOVE
 YUMOVE4  SB6    B6+LE.MOV         ADVANCE TO NEXT MOVE
          EQ     YUMOVE2

*         END OF SCAN.

 YUMOVE5  NZ     X7,YUMOVE8        IF LEGAL MOVE
          SX5    =C* ILLEGAL MOVE.*
          EQ     YUMOVE7

*         AMBIGUOUS MOVE.

 YUMOVE6  SX5    =C* AMBIGUOUS MOVE.*

*         ILLEGAL MOVE.

 YUMOVE7  RJ     =XERRMSG
          EQ     YRMOVE2           TRY AGAIN

*         LEGAL MOVE.

 YUMOVE8  SA7    NXTMV             SET NEXT MOVE
          SA2    PONDR
          MX6    -18
          BX3    X2-X7             COMPARE TO PONDER MOVE
          BX3    -X6*X3
          SX6    B1
          ZR     X3,YUMOVE9        IF PONDERING CORRECT MOVE
          BX6    X6-X6
          SA6    A2                CLEAR PONDR
          EQ     YPMOVE            FINISH

 YUMOVE9  LX6    S.THG             SET GOOD MOVE
          BX6    X2+X6
          SA6    PONDR
*         EQ     YPMOVE
 YPMOVE   SPACE  4,20
***       YPMOVE - COMPLETE PLAYERS MOVE.
*
*         ENTRY  (NXTMV) = PLAYERS MOVE.
*                (PONDR) = PONDER MOVE.
*
*         EXIT   TO YRMOVE2, IF NO MOVE YET.
*                TO PONDER, IF PONDER MOVE CORRECT AND INCOMPLETE.
*                TO MAKMOV, OTHERWISE.
*



          ENTRY  YPMOVE
 YPMOVE   SA1    NXTMV
          ZR     X1,YRMOVE2        IF NO MOVE YET
          SA2    PONDR
          SA3    SWICH
          LX3    59-S.PON
          LX2    59-S.THF
          NG     X2,YPMOVE1        IF PONDER FINISHED
          ZR     X2,YPMOVE1        IF NOT PONDERING
          PL     X3,YPMOVE1        IF NOT PONDERING
          JUMPUP PONDER            FINISH PONDERING

 YPMOVE1  BX6    X1
          BX7    X7-X7
          SA7    TIMMV             CLEAR TIME
          SA6    BSTMV             SET PLAYERS MOVE
          SA7    A1                CLEAR NEXT MOVE
          IF     DEF,ONELINE,1
          RJ     =XONLCLR          CLEAR SCREEN
          SA1    =10H YOUR MOVE
          RJ     =XRECORD
          SX1    MOVMS
          RJ     =XDUDMSG
          RJ     =XDUDCLR
          RJ     =XLEARNR          LEARN MOVE IF NECESSARY.
          JUMPUP MAKMOV

 YUMOVEA  DATA   0                 MOVE MASK
          DATA   0                 MOVE VALUE
          DATA   0                 FIRST ALTERNATIVE
          DATA   0                 SECOND ALTERNATIVE
          DATA   0                 THIRD ALTERNATIVE
          DATA   0                 FROM SQUARE SHIFT COUNT
          DATA   0                 FROM SQUARE VALUE
          DATA   0                 TO SQUARE SHIFT COUNT
          DATA   0                 TO SQUARE VALUE

 YUMOVEB  CHARS  B,YUMOVEJ,LEADBE
          CHARS  ABCDEFGH,YUMOVEK,LHALFI
          CHARS  PRNBQK,YUMOVEC,LHSPCE
          CHARS  0O,YUMOVEG,CASTLE
          DATA   0

 YUMOVEJ  CHARS  /(,YUMOVEC,IGNORE
          CHARS  12345678,YUMOVEL,LHALRK
 YUMOVEC  CHARS  KQ,*,LHSKOQ
          CHARS  RNB,*,LHSRNB
          CHARS  12345678,*,LHSRNK
          CHARS  -,YUMOVEE,ORDMOV
          CHARS  *X,YUMOVED,CAPTUR
          DATA   0

 YUMOVED  CHARS  PRNBQ,YUMOVEE,RHSPCE
          DATA   0

 YUMOVEE  CHARS  /(,*,IGNORE
          CHARS  KQ,*,RHSKOQ
          CHARS  RNB,*,RHSRNB
          CHARS  12345678,*,RHSRNK
 YUMOVEO  CHARS  =/,YUMOVEF,IGNORE
          DATA   0

 YUMOVEF  CHARS  BRNQ,*,PROMOT
          DATA   0

 YUMOVEG  CHARS  -,YUMOVEH,IGNORE
          DATA   0

 YUMOVEH  CHARS  0O,YUMOVEI,IGNORE
          DATA   0

 YUMOVEI  CHARS  -,YUMOVEH,CASLNG
          DATA   0

 YUMOVEK  CHARS  12345678,YUMOVEL,LHALRK
          DATA   0

 YUMOVEL  CHARS  -,YUMOVEM,ORDMOV
          CHARS  *X,YUMOVEM,CAPTUR
          DATA   0

 YUMOVEM  CHARS  ABCDEFGH,YUMOVEN,RHALFI
          DATA   0

 YUMOVEN  CHARS  12345678,YUMOVEO,RHALRK
          DATA   0
 LHSPCE   TITLE  SEMANTICS ROUTINES FOR PLAYERS MOVE PROCESSING.
 LEADBE   SPACE  4
**        LEADBE - FIRST CHARACTER -B-.
*



 LEADBE   SX6    B1
          SA6    ALGBRAA           SET LEADING B FLAG
          BX6    X6-X6
*         EQ     LHSPCE
 LHSPCE   SPACE  4
**        LHSPCE - PIECE DEFINED ON LEFT HAND SIDE.
*

 LHSPCE   SA5    LHSPCEA+X6
          BX7    X7-X7
          BX6    X5
          SA7    YUMOVEA+5
          SA6    A7+B1
          EQ     YUMOVE1

 LHSPCEA  CON    BISH,KING,NITE,PAWN,QUEN,ROOK
 CASTLE   SPACE  4,10
**        CASTLE - MOVE IS CASTLE (ASSUME KING-SIDE)
*

 CASTLE   SX6    M.CAS+M.ENP+M.PRO+M.ACS
          SX7    M.CAS+M.ACS
          SA6    YUMOVEA
          SA7    A6+1
          EQ     YUMOVE1
 LHSKOP   SPACE  4,12
**        LHSKOQ - KING OR QUEEN SIDE SPECIFIED ON LHS.
*

 LHSKOQ   SX7    4
          IX6    X7-X6             4 IF K, 3 IF Q
          LX6    S.MFR
          SX7    700B
          SA7    YUMOVEA           MOVE MASK
          SA6    A7+B1             MOVE VALUE
          EQ     YUMOVE1
 LHSRNB   SPACE  4,10
**        LHSRNB - R/N/B SPECIFIED ON LHS.
*

 LHSRNB   SA3    YUMOVEA           MOVE MASK
          SX7    700B
          SA4    A3+B1             MOVE VALUE
          BX2    X7*X3
          SX6    X6+B1
          LX6    6
          BX6    X4-X6
          SA6    A4                UPDATE MOVE VALUE
          NZ     X2,YUMOVE1        IF KQ SPECIFIED
          SA7    A4+B1             ELSE SET ALTERNATES
          SX5    300B
          SA7    A7+2
          BX6    X5-X6
          SA6    A6                CORRECT FILE VALUE FOR QUEEN SIDE
          BX7    X3+X7
          SA7    A3                AND THE MASK
          EQ     YUMOVE1
 IGNORE   SPACE  4,6
***       IGNORE - IRRELEVANT CHARACTERS.
*

 IGNORE   EQU    YUMOVE1
 LHSRNK   SPACE  4,18
**        LHSRNK - RANK SPECIFIED ON LHS.
```

```
*
 LHSRNK   SA4    YUMOVEA           MASK
          SX6    B3-1R1            CONVERT CHARACTER TO RANK
          SA3    MSIDE
          SA5    A4+B1
          LX6    9
          SX7    7000B
          BX3    X3*X7
          BX6    X3-X6             COMPLEMENT IF BLACK
          BX7    X7+X4
          BX6    X6+X5
          SA7    A4
          SA6    A5
          EQ     YUMOVE1
 ORDMOV   SPACE  4,11
**        ORDMOV - ORDINARY MOVE.
*

 ORDMOV   SA4    YUMOVEA
          SX6    B1
          LX6    S.CAP
          BX6    X6+X4
          SA6    A4
          EQ     YUMOVE1
 CAPTUR   SPACE  4,15
**        CAPTUR - CAPTURE MOVE.
*

 CAPTUR   SA4    YUMOVEA
          SA5    A4+B1
          SX6    B1
          LX6    S.CAP
          BX7    X5+X6
          BX6    X4+X6
          SA6    A4
          SA7    A5
          EQ     YUMOVE1
 RHSPCE   SPACE  4,14
**        RHSPCE - PIECE SPECIFIED ON RHS.
*

 RHSPCE   SA4    RHSPCEB+X6        PIECE TYPE
          SA5    RHSPCEA+X6        SHIFT COUNT
          BX6    X4
          BX7    X5
          SA7    YUMOVEA+7
          SA6    A7+B1
          EQ     YUMOVE1

 RHSPCEA  CON    0,0,1,0,0

 RHSPCEB  CON    BISH,NITE,PAWN,QUEN,ROOK
 RHSKOQ   SPACE  4,14
**        RHSKOQ - KING OR QUEEN SIDE SPECIFIED ON RHS.
*

 RHSKOQ   SX7    4
          IX6    X7-X6             4 IF K, 3 IF Q
          SX7    7B
          SA4    YUMOVEA
          SA5    A4+B1
          BX7    X7+X4
          BX6    X6+X5
          SA7    A4
          SA6    A5
          EQ     YUMOVE1
 RHSRNB   SPACE  4,14
**        RHSRNB - R/N/B SPECIFIED ON RHS.
*

 RHSRNB   SA3    YUMOVEA           MOVE MASK
          SX7    7B
          SA4    A3+B1             MOVE VALUE
          BX2    X7*X3
          SX6    X6+B1
          BX6    X4-X6
          SA6    A4                UPDATE MOVE VALUE
          NZ     X2,YUMOVE1        IF KQ SPECIFIED
          SA7    A4+2              ELSE SET ALTERNATE
          AX5    X7,B1             3
          BX6    X6-X5             CORRECT FILE VALUE FOR QUEEN SIDE
          SA6    A6
          BX7    X3+X7
          SA7    A3                AND THE MASK
          EQ     YUMOVE1
 RHSRNK   SPACE  4,18
**        RHSRNK - RANK SPECIFIED ON RHS.
*

 RHSRNK   SX4    B3-1R1
          SA5    MSIDE
          SX7    70B
          BX6    X5*X7
          LX4    3
          BX6    X4-X6
          SA4    YUMOVEA
          SA5    A4+B1
          BX7    X7+X4
          BX6    X5+X6
          SA6    A5
          SA7    A4
          EQ     YUMOVE1
 PROMOT   SPACE  4,18
**        PROMOT - PROMOTE A PAWN.
*

 PROMOT   SA4    YUMOVEA
          SA5    A4+B1
          SA3    PROMOTA+X6
          SX6    M.PRO+M.ENP+M.CAS
          BX6    X6+X4
          BX7    X3+X5
          SA6    A4
          SA7    A5
          EQ     YUMOVE1

 PROMOTA  CON    M.PRO+M.ENP,M.PRO+M.CAS,M.PRO+M.ENP+M.CAS,M.PRO

 CASLNG   SPACE  4,7
**        CASLNG - CASTLE LONG.
*

 CASLNG   SX6    M.CAS+M.ACS+M.ENP
          SA6    YUMOVEA+1
          EQ     YUMOVE1
 LHALFI   SPACE  4
**        LHALFI - LEFT SIDE ALGEBRAIC FILE.
*

 LHALFI   SX7    B3-1RA
          SX6    700B
          LX7    6
          EQ     ALGBRA
 LHALRK   SPACE  4
**        LHALRK - LEFT SIDE ALGEBRAIC RANK.
*

 LHALRK   SX7    B3-1R1
          SX6    7000B
          LX7    9
          EQ     ALGBRA
 RHALRK   SPACE  4
**        RHALRK - RIGHT SIDE ALGEBRAIC RANK.
*

 RHALRK   SX7    B3-1R1
          SX6    70B
          LX7    3
          EQ     ALGBRA
 RHALFI   SPACE  4
**        RHALFI - RIGHT SIDE ALGEBRAIC FILE.
*

 RHALFI   SX7    B3-1RA
          SX6    7B
*         LX7    0
*         EQ     ALGBRA
 ALGBRA   SPACE  4,25
**        ALGBRA - FINISH ALGEBRAIC CHARACTER.
*

 ALGBRA   SA3    YUMOVEA
          SA2    A3+B1
          BX6    X3+X6
          BX7    X2+X7
          SA6    A3
          SA7    A2
          SA3    ALGBRAA
          ZR     X3,YUMOVE1        IF NOT SWITCH TO ALGEBRAIC
          BX6    X6-X6
          SA6    A3
          SA6    YUMOVEA+6         CLEAR FROM TYPE
          SX6    60
          SA6    A6-B1
          SX6    700B
          SX7    100B
          EQ     ALGBRA            SET ALGEBRAIC B

 ALGBRAA  CON    0                 LEADING -B- FLAG
 RECORD   TITLE  MISCELLANEOUS CONTROL ROUTINES.
***       RECORD - RECORD MOVE FOR EITHER SIDE.
*
*         ENTRY  (X1) = FIRST WORD OF MOVE MESSAGE.
*                (BSTMV) = THE MOVE.
*                (MOVES) = ALL LEGAL MOVES.
*                (BOARD) = POSITION BEFORE MOVE IS MADE.
*
*         CALLS  MINENG, WRLINE,GAMSCR.
*

 RECORD   EQ     *+400000B
          BX7    X1
          RJ     =XMINENG
          SX5    MOVMS
          RJ     =XWRLINE
          WRITE  =XOUTPUT          FLUSH THE BUFFER
          SX1    MOVMS
          RJ     =XDUDMSG          DISPLAY MOVE
          SA1    MSIDE
          RJ     =XGAMSCR
          SA1    SWICH
          LX1    59-S.ADI
          PL     X1,RECORD         IF NO DAYFILE
          DAYFILE MOVMS,10B
          EQ     RECORD
 LEARNR   SPACE  4,35
**        LEARNR - LEARN MOVES.
*

 LEARNR   EQ     *+400000B
          SA1    SWICH
          SA2    OSIDE
          SB2    X2
          LX1    59-S.LEW
          ERRNZ  S.LEB-S.LEW-1
          SB2    -B2
          SB2    60+B2
          LX1    X1,B2
          PL     X1,LEARNR         IF NOT LEARNING THIS SIDE
          SA0    YUMOVEA           BUFFER AREA
          SA3    STATUS
          SB3    BOARD
          RJ     =XMSPACK          PACK UP POSITION
          SA1    YUMOVEA
          SA2    BSTMV
          MX3    -18
          BX2    -X3*X2            EXTRACT MOVE
          LX2    42
          BX6    X1+X2
          SA6    A1                STORE THE MOVE
          BX7    X7-X7
          RJ     =XMSREPL          REPLACE MOVE IN LIBRARY
          NZ     X6,LEARNR         IF ADDED
          SA1    =10H *********
          RJ     =XCARAGE
          SX5    =C* DUPLICATE POSITION REPLACED.*
          RJ     =XWRLINE
          EQ     LEARNR
 RECRDM   SPACE  4,70
***       RECRDM - RECORD MOVE FOR MACHINE.
*
*         ENTRY  (BSTMV) = THE MOVE.
*                (MOVES) = ALL LEGAL MOVES.
*                (BOARD) = POSITION BEFORE MOVE IS MADE.
*
*         EXIT   TO MACMOV.
```

```
*
*         CALLS  RECORD, RDRCDD, WRLINE.
*


          ENTRY  RECRDM
RECRDM    RJ     =XRNGBEL          RING BELL
          SA1    BSTMV
          RJ     =XCLMDRW          CLAIM DRAW IF APPRORIATE
          SA1    =10H   MY MOVE
          SA2    TIMMV
          SA3    TIMPS
          IX7    X2+X3             ADVANCE PSEUDO-MILLISECONDS
          SA2    TIMPO             GET PONDER FREE TIME
          IX7    X7-X2             DONT CHARGE FOR IT
          SA7    A3
          RJ     =XRECORD
          SA1    SWICH
          LX1    59-S.WEI
          PL     X1,RECRDM1        IF NOT PRINTING WEIRDOS
          RJ     =XPRIWEI
RECRDM1   RJ     =XPSTGCL          UPDATE GAME CLOCK
          RJ     =XCTIMPM          COMPUTE TIME PER MOVE
          SA3    MOVNUM
          SA2    TILTC
          SA4    RULMV1
          SA5    RULMV2
          IX4    X3-X4             MOVNUM - RULMV1
          NG     X4,RECRDM2        IF HAVENT REACHED FIRST TC
          IX5    X2-X5             TILTC - RULMV2
          NZ     X5,RECRDM2        IF HAVENT JUST FINISHED A TC
          SA6    ATMPM             ELSE RESET TIMPM GOAL
RECRDM2   SA1    SWICH
          SA2    TQNEXT
          IX6    X3-X2
          LX1    59-S.TIM
          PL     X1,RECRDM4        IF NOT UNDER TIME CONSTRAINTS
          NG     X6,RECRDM4        IF NOT TIME TO ASK TIME USED
          SA5    TILTC             MOVES TIL TIME CONTROL
          SA4    ASKFCT
          IX6    X5*X4
          SA1    =0.01P48
          PX6    X6
          RX6    X6*X1
          UX6    X6
          SX7    X6-1
          AX7    59
          SX5    B1
          BX7    X5*X7
          BX6    X6+X7
          IX6    X6+X3
          SA6    A2                UPDATE NEXT TIME QUESTION
          RJ     AFT               ASK FOR TIME
RECRDM3   SX6    O.HMTU
          RJ     =XREADER
          ENTRY  TIMCTL
TIMCTL    SA1    ILINE
          SB2    54
          MX0    54
          RJ     =XRDRGNT
          ZR     X7,RECRDM4        IF CURRENT TIME CORRECT
          BX5    X7
          SB7    B1
          RJ     =XRDRDXB          CONVERT DECIMAL NUMBER
          NZ     X4,RECRDM8        IF CONVERSION ERROR
          SA1    AFTA              CHECK AGAINST COMPUTER TIME
          SA2    GCLOCK
          IX7    X6-X1
          SB6    X7-30
          SB7    X7+30
+         NG     B6,*+1            IF LESS THAN 30 MINUTES FAST
          SX7    X7-60
+         PL     B7,*+1            IF LESS THAN 30 MINUTES SLOW
          SX7    X7+60
          IX6    X7+X2             ADJUST TIME ERROR
          SA6    A2
          BX3    X6
          SA1    MOVNUM
          RJ     =XGCLPST          CONVERT TO PSEUDO-MILLISECONDS
          SA6    TIMPS
          RJ     =XCTIMPM          COMPUTE TIME PER MOVE
          SA6    ATMPM

RECRDM4   SA1    SWICH
          LX1    59-S.SUM
          PL     X1,RECRDM5        IF NO SUMMARY NEEDED
          RJ     =XPRISUM          PRINT SUMMARY
RECRDM5   SA1    SWICH
          LX1    59-S.NDC
          PL     X1,RECRDM6        IF NODE COUNTS NOT NEEDED
          RJ     =XPRINDC          PRINT NODE COUNTS
RECRDM6   SA1    SWICH
          LX1    59-S.PRV
          PL     X1,RECRDM7        IF PRINCIPLE VARIATION NOT NEEDED
          RJ     =XPRIPRV          PRINT PRINCIPLE VARIATION
RECRDM7   JUMPUP MACMOV            MAKE MACHINE MOVE

**        PROCESS ERROR.

RECRDM8   SX5    =C* SPECIFY NUMBER IN MINUTES.*
          RJ     =XERRMSG
          JP     RECRDM3
AFT       SPACE  4
**        AFT - ASK FOR TIME ON CHESS CLOCK.


AFT       PS                       ENTRY/EXIT
          SA1    GCLOCK            COMPUTE CURRENT TIME
          SA2    RULTM1
          IX6    X1-X2

*         DETERMINE MINUTE HAND POSITION.

+         SX6    X6-60             DISCARD HOURS
          PL     X6,*              LOOP
+         SX6    X6+60
          NG     X6,*
          SA6    AFTA
          SA1    =H* HAND  (00*
          BX7    X7-X7             CONVERT TO DECIMAL
          SB6    X6
          SB7    10
          LT     B6,B7,AFT2        IF 0 THRU 9 MINUTES
AFT1      SX7    X7+B1             ACCUMULATE TENS
          SB6    B6-B7
          GE     B6,B7,AFT1        IF MORE THAN 9 MINUTES
AFT2      SX6    B6                LX7    6
          IX6    X6+X1
          IX6    X6+X7
          SA6    HMTMS+2
          SX5    HMTMS
          RJ     =XWRLINE          WRITE LINE
          JP     AFT

AFTA      CON    0                 CURRENT MINUTE HAND POSITION
CLMDRW    SPACE  4,10
**        CLMDRW - CLAIM DRAW BY REPETITION IF APPROPRIATE.
*
*                ALSO CLAIMS IF 50 MOVE RULE LIMIT.
*         ENTRY  (X1) = MOVE WORD.
*
*         EXIT   DRAW IS CLAIMED IF S.REP IS SET.
*
*         CALLS  WRLINE.


          ENTRY  CLMDRW
CLMDRW    EQ     *+400000B
          LX1    59-S.REP
          PL     X1,CLMDRW         IF NOT REPETITION OR 50 MOVE LIMIT
          SX5    =C* I CLAIM DRAW BY REPETITION OR 50 MOVE RULE.*
          RJ     =XWRLINE          CLAIM THE DRAW
          EQ     CLMDRW            RETURN
PROCLO    SPACE  4,28
**        PROCLO - PROCESS CLOCK CHANGES.
*
*         ENTRY  (X7) = DISPLAY CODE MINUTES TO CHANGE TO.
*
*         EXIT   (X5) = ADDRESS OF ERROR MESSAGE.
*                     = 0, IF NO ERROR.
*


          ENTRY  PROCLO
PROCLO    EQ     *+400000B
          SX5    =C* SPECIFY NUMBER OF MINUTES.*
          ZR     X7,PROCLO         IF NOTHING SPECIFIED.
          BX5    X7
          SB7    B1
          RJ     =XRDRDXB          CONVERT
          SX5    =C* SPECIFY NUMBER OF MINUTES.*
          NZ     X4,PROCLO         IF CONVERSION ERROR
          SA6    GCLOCK
          BX3    X6
          SA1    MOVNUM
          RJ     =XGCLPST          CONVERT TO PSEUDO-MILLISECONDS
          SA6    TIMPS
          RJ     =XCTIMPM          COMPUTE TIME PER MOVE
          SA6    ATMPM
          BX5    X5-X5             CLEAR ERROR
          EQ     PROCLO
RNGBEL    SPACE  4,20
**        RNGBEL - RING BELL.
*
*         CALLS  WRLINE
*


RNGBEL    EQ     *+400000B
          SA1    SWICH
          LX1    59-S.BEL
          PL     X1,RNGBEL         IF NOT RINGING BELL
          SA1    =XOUTPUT+1
          RJ     =XDEVTYP
          PL     X6,RNGBEL         IF OUTPUT ALLOCATABLE
          SX5    RNGBELA           OUTPUT BELL
          RJ     =XWRLINE
          EQ     RNGBEL            RETURN

RNGBELA   BSS    0
NUCC      IF     DEF,NUCC
          VFD    12/0060B          ENTER ASCII MODE
          VFD    12/0207B          BELL
          VFD    12/4000B          EXIT ASCII MODE
          VFD    12/0057B          SUPPRESS CRLF
          VFD    12/0000B          CRLF
NUCC      ENDIF
KRONOS    IF     DEF,KRONOS
          VFD    12/0011B          ENTER ASCII MODE
          VFD    12/7647B          BELL
          VFD    12/0013B          END OF STRING
KRONOS    ENDIF
          IFNE   *P,60,1
          VFD    *P/0              FILL WORD
          ENDIF
GCLPST    SPACE  4,88
***       GCLPST - CONVERT GAME CLOCK TO PSEUDO-MILLISECONDS.
*
*         ENTRY  (X3) = REAL TIME MINUTES.
*                (X1) = NUMBER OF MOVES.
*                (PERCNT) = PERCENTAGE OF CPU ALLOCATED TO CHESS.
*                (OVRHED) = SECONDS TO ENTER MOVE, AND SYSTEM OVERHEAD.
*                (B1) = 1.
*
*         EXIT   (X6) = PSEUDO-MILLISECOND CLOCK.
*                     = PERCNT/100 * GCLOCK*60*1000 -
*                       PERCNT/100 * MOVNUM*OVRHED*1000
*
*         USES   A1, A2, X1, X2, X3.
*


          ENTRY  GCLPST
GCLPST    EQ     *+400000B
          SA2    OVRHED
          IX6    X1*X2             MOVNUM*OVRHED
          BX2    X3
          LX3    6                 64*GCLOCK
          LX2    2                 4*GCLOCK
          IX2    X3-X2             60*GCLOCK
          IX6    X2-X6             60*GCLOCK-MOVNUM*OVRHED
          SA1    PERCNT
          LX2    X1,B1             2*PERCNT
          LX1    3                 8*PERCNT
          IX2    X2+X1             10*PERCNT
          IX6    X6*X2             10*PERCNT*(60*GCLOCK-MOVNUM*OVRHED)
          EQ     GCLPST            RETURN
PSTGCL    SPACE  4,40
***       PSTGCL - CONVERT PSEUDO-MILLISECONDS TO GAME CLOCK.
*
*         ENTRY  (TIMPS) = PSEUDO-MILLISECONDS.
*                (MOVNUM) = NUMBER OF MOVES MADE IN GAME.
*                (PERCNT) = PERCENTAGE OF CPU ALLOCATED TO CHESS.
```

```
*                (OVRHED) = SECONDS TO ENTER MOVE, AND SYSTEM OVERHEAD.
*
*         EXIT   (GCLOCK) = ESTIMATED REAL TIME MINUTES SINCE FIRST MOVE
*                         = TIMPS/(1000*60*PERCNT/100) +
*                           MOVNUM*OVRHED/60
*                (X6) = (GCLOCK)
*
*         USES   A1, A2, A6, X1, X2, B2.
*



 PSTGCL   EQ     *+400000B
          SA1    MOVNUM
          SA2    OVRHED
          IX6    X1*X2             MOVNUM*OVRHED
          SX2    172574B
          LX2    42                60.
          PX6    X6
          NX6    X6
          SA1    PERCNT
          FX6    X6/X2             MOVNUM*OVRHED/60
          SX2    600
          IX2    X1*X2             PERCNT*600
          SA1    TIMPS
          PX1    X1
          PX2    X2
          NX1    X1
          NX2    X2
          FX2    X1/X2             TIMPS/(PERCNT*600)
          FX6    X6+X2             TIMPS/(PERCNT*600)+MOVNUM*OVRHED/60
          UX6    X6,B2
          LX6    X6,B2
          SA6    GCLOCK
          EQ     PSTGCL
 CTILTC   SPACE  4,38
***       CTILTC - COMPUTE NUMBER OF MOVES UNTIL TIME CONTROL.
*
*         ENTRY  (MOVNUM) = NUMBER OF MOVES MADE.
*                (RULMV1) = NUMBER OF MOVES IN FIRST PERIOD.
*                (RULMV2) = NUMBER OF MOVES IN SUBSEQUENT PERIODS.
*
*         EXIT   (TILTC) = NUMBER OF MOVES UNTIL TIME CONTROL.
*                        = RULMV1+N*RULMV2-MOVNUM
*                (X6) = (TILTC)
*                (X7) = N = MAX(0,(MOVNUM-RULMV1+RULMV2)/RULMV2)
*                     = NUMBER OF SUBSEQUENT TIME CONTROLS.
*
*         USES   A1, A2, A3, A6, X1, X2, X3, B2.
*



 CTILTC   EQ     *+400000B
          SA1    MOVNUM
          SA2    RULMV1
          SA3    RULMV2
          IX6    X1-X2             MOVNUM-RULMV1
          IX6    X6+X3             MOVNUM-RULMV1+RULMV2
          PX6    X6
          PX7    X3
          NX6    X6
          NX7    X7
          FX6    X6/X7             (MOVNUM-RULMV1+RULMV2)/RULMV2
          UX6    X6,B2
          LX6    X6,B2
          BX7    X6
          AX7    59
          BX7    -X7*X6            N = NUMBER OF SUBSEQUENT TIME CONTRLS
          IX6    X7*X3             N*RULMV2
          IX6    X2+X6             RULMV1+N*RULMV2 = NEXT TC
          IX6    X6-X1             RULMV1+N*RULMV2-MOVNUM
          SA6    TILTC             NUMBER OF MVOES TIL TC
          EQ     CTILTC            RETURN
 CTIMPM   SPACE  4,40
***       CTIMPM - COMPUTE TIME PER MOVE.
*
*         ENTRY  (RULMV1) = NUMBER OF MOVES IN FIRST PERIOD.
*                (RULMV2) = NUMBER OF MOVES IN SUBSEQUENT TIME PERIODS.
*                (RULTM1) = NUMBER OF MINUTES IN FIRST TIME PERIOD.
*                (RULTM2) = NUMBER OF MINUTES IN SUBSEQUENT TIME PERIODS
*                (EXTRAT) = TIME CONTROL BUFFER.
*                (EXTRAS) = ABS MINIMUM TIME CONTROL BUFFER.
*                (MOVNUM) = NUMBER OF MOVES ALREADY PLAYED.
*                (OVRHED) = NUMBER OF SECONDS OVERHEAD PER MOVE.
*                (MTMPM) = MAXIMUM MILLISECONDS ALLOWED PER MOVE.
*                (OTMPM) = OK (ACCEPTABLE MIN) TIME-PER-MOVE.
*
*         EXIT   (X6) = (TIMPM) = NUMBER OF MILLISECONDS ALLOWED/MOVE.
*                THE SAFETY MARGIN OF EXTRAT MAY BE REDUCED TO AS
*                LOW AS EXTRAS TO MAKE UP SOME OF THE TIME LOST DUE
*                TO SYSTEM CRASH, EXCESS OVERHEAD, TREE EXPLOSION,
*                ETC.
*
*         USES   A1, A2, A3, A6, X1, X2, X3, X6, X7, B2.
*         CALLS  CTILTC, APXTPM.
*



          ENTRY  CTIMPM
 CTIMPM   EQ     *+400000B
          RJ     =XCTILTC          COMPUTE NUMBER MOVES TIL TC
          SA3    EXTRAT
          RJ     APXTPM            COMPUTE FIRST APPROX TIME-PER-MOVE
          SA1    MTMPM             MAXIMUM TIME PER MOVE
          IX2    X1-X6
          AX2    59
          BX1    X2*X1
          BX6    -X2*X6
          BX6    X1+X6
          PL     X6,CTIMPM1        IF POSITIVE
          BX6    X6-X6             ELSE SET 0
 CTIMPM1  SA6    TIMPM
          SA1    OTMPM             OK TIME PER MOVE
          IX1    X6-X1             COMPARE TO WHAT WE HAVE
          PL     X1,CTIMPM         RETURN IF WE HAVE ENOUGH

*         RAID EXTRAT TO BOOST TIMPM.

          SA3    EXTRAS            SACRED MINIMUM EXTRAT
          RJ     APXTPM            GET MAX POSSIBLE TIMPM (M)
          PL     X6,CTIMPM2        IF POSITIVE
          BX6    X6-X6             ELSE SET ZERO
 CTIMPM2  SA1    OTMPM             OK TIME PER MOVE (N)
          SA2    TIMPM             FIRST APPROX TIME-PER-MOVE (T)
          IX3    X1-X2             (N-T)
          IX1    X3+X6             (N+M-T)
          IX6    X6-X2             (M-T)
          IX6    X3*X6             (N-T)*(M-T)
          NZ     X1,CTIMPM3        IF (N+M-T) IS NON-ZERO
          SX1    1000              ELSE PREVENT DIVISION BY 0
 CTIMPM3  PX6    X6
          PX1    X1
          NX6    X6                FLOAT((N-T)*(M-T))
          NX1    X1                FLOAT(N+M-T)
          FX6    X6/X1
          UX6    X6,B2
          LX6    X6,B2             (N-T)*(M-T)/(N+M-T)
          IX6    X6+X2             T+(N-T)*(M-T)/(N+M-T)
          SA1    OTMPM             NOW COMPARE TO OK TIME
          IX2    X1-X6
          PL     X2,CTIMPM4        WE SHOULDNT HAVE MORE
          BX6    X1                ELSE SET TO OK TIME
 CTIMPM4  NG     X6,CTIMPM5        IF TIME IS NEGATIVE
          NZ     X6,CTIMPM6        OK IF POSITIVE AND NON-ZERO
 CTIMPM5  SX6    1                 ELSE SET TO 1 MILLI-SECOND
 CTIMPM6  BSS    0
          SA6    TIMPM
          EQ     CTIMPM            RETURN
 APXTPM   SPACE  4,10
**        APXTPM - COMPUTE FIRST APPROX TO TIME-PER-MOVE.
*
*         ENTRY  (X3) = EXTRAT VALUE TO HIDE.
*                (X7) = AS LEFT BY CTILTC.
*
*         EXIT   (X6) = FIRST APPROX NO. MILLISECONDS ALLOWED PER MOVE.
*
*         USES   A1, A2, X1, X2, X3, X6, B2.
*         CALLS  GCLPST.



 APXTPM   EQ     *+400000B
          SA1    RULTM1
          SA2    RULTM2
          IX2    X2*X7             N*RULTM2
          IX2    X1+X2             RULTM1+N*RULTM2
          IX3    X2-X3             RULTM1+N*RULTM2-EXTRAT
          SA1    RULMV1
          SA2    RULMV2
          IX2    X2*X7             N*RULMV2
          IX1    X1+X2             RULMV1+N*RULMV2
          RJ     =XGCLPST          CONVERT TIME CONTROL TO PSEUDO-MSEC
          SA1    TILTC
          SA2    TIMPS
          IX6    X6-X2             PSEUDO-MSEC TIL TIME CONTROL
          PX6    X6
          PX1    X1
          NX6    X6
          NX1    X1
          FX6    X6/X1
          UX6    X6,B2
          LX6    X6,B2             PSEUDO-MSEC/MOVE UNTIL TC
          EQ     APXTPM            RETURN
 PRISUM   SPACE  4,88
***       PRISUM - PRINT SUMMARY INFORMATION.
*
*   MMM CL NNN S EEEE.EE NC CCCCCC T SSS I P SP UUUU RT RRRR RA AAAAA
*
*               MMM = MOVE NUMBER.
*               NNN = ESTIMATED REAL TIME CLOCK.
*               EEEE.EE = EVALUATION.  +1 = WHITE IS UP 1 PAWN.
*                         FRACTION IS IN OCTAL.
*               CCCCCC = TOTAL NODES SEARCHED.
*               SSS = SECONDS USED TO COMPUTE MOVE.
*               P = MAXIMUM PLY.
*               UUUU = MICROSECONDS PER NODE.
*               RRRR = REMAINING CPU SECONDS PER MOVE.
*               AAAAA = 15 BIT RANDOM NUMBER USED FOR JIGGLING LETS.
*
*         USES   ALL REGISTERS, OLINE.
*         CALLS  RDRCDD, WRLINE.
*



          ENTRY  PRISUM
 PRISUM   EQ     *+400000B
          SA1    MOVNUM
          RJ     =XRDRCDD          CONVERT MMM
          BX5    X6                SAVE MOVE NUMBER
          SA1    GCLOCK
          RJ     =XRDRCDD          CONVERT NNN
          MX0    -6*4
          BX6    -X0*X6            ZZZZZZBNNN
          LX5    6
          BX5    -X0*X5            ZZZZZZMMMB
          SX7    2RCL
          LX7    6*4               ZZZZCLZZZZ
          LX5    6*6               MMMBZZZZZZ
          BX6    X6+X5             MMMBZZBNNN
          BX6    X6+X7             MMMBCLBNNN
          SA6    OLINE
          SA1    SCORE
          AX1    6
          RJ     =XRDRCDD          CONVERT EEEE.
          SA1    SCORE
          SA2    PRISUMA           ZSZZZZZ.00
          BX7    X1
          LX6    6*3
          AX7    59
          BX6    X6-X2              S EEEE.00
          BX1    X1-X7
          MX7    -3
          BX2    -X7*X1
          AX1    3
          BX1    -X7*X1
          LX1    6
          IX6    X6+X2
          IX6    X6+X1              S EEEE.EE
          SA6    A6+B1
          SA1    NDTTL             TOTAL NODE COUNT
          RJ     =XRDRCDD
          SX7    2RNC&2R
          LX7    6*7
          BX6    X7-X6
          SA6    A6+B1
          SA1    TIMMV
          SX2    500
          IX1    X1+X2
          RJ     =XRDRCDD
          BX5    X6
          LX5    4*6-3*6
          SA1    ITERS             NUMBER OF ITERATIONS
          RJ     =XRDRCDD
          MX0    -6*4
          BX5    X0*X5
          BX6    -X0*X6
```

```
          BX6    X5+X6
          SX5    1RT&1R
          SX7    1RI&1R
          LX7    2*6
          LX5    8*6
          BX6    X6-X5
          BX6    X6-X7
          SA1    NDTTL            TOTAL NODES
          SA2    TIMMV            TIME (MSEC)
          SX3    1000D
          SA6    A6+B1
          IX2    X2*X3            TIME (USEC)
          PX1    X1
          PX2    X2
          NX1    X1
          NX2    X2
          FX1    X2/X1            USEC/NODE
          UX1    X1,B2
          LX1    X1,B2
          RJ     =XRDRCDD
          LX6    2*6
          SX7    2RSP&2R
          LX7    7*6
          BX6    X6-X7
          SX7    1RR&1R
          BX6    X6-X7
          SX2    500
          SA1    TIMPM
          SA6    A6+B1
          IX1    X1+X2
          SA2    =0.001P48
          PX1    X1
          FX1    X1*X2
          RJ     =XRDRCDD
          LX6    4*6
          SX7    1RT&1R
          LX7    9*6
          BX6    X6-X7
          SX7    2RRA&2R
          LX7    6
          BX6    X6-X7
          SA6    A6+B1
          SA1    RANDOM           CONVERT RANDOM JIGGLER
          RJ     =XRDRCOD
          LX6    5*6
          MX7    6*8
          BX6    X6*X7
          SA6    A6+B1
          SX5    OLINE
          RJ     =XWRLINE
          EQ     PRISUM           RETURN

 PRISUMA  VFD    12/1RS&1R ,30/,18/3R.00&3R
 PRINDC   SPACE  4,53
***       PRINT NODE COUNTS BY PLY AND ITERATION.
*

          ENTRY  PRINDC
 PRINDC   EQ     *+400000B
          SA1    =10H NODES/PLY
          SX5    OLINE
          BX6    X1
          SA6    X5
          SB6    A6+B1
          BX6    X6-X6
          SA6    OLINE+7
          SA0    NDCNT
          SB7    A6
 PRINDC1  SA1    A0               NEXT PLY NODE COUNT
          ZR     X1,PRINDC2       IF HIGHEST PLY REACHED
          RJ     =XRDRCDD         CONVERT
          SA6    B6
          SB6    B6+B1
          SA0    A0+B1
          LT     B6,B7,PRINDC1    IF OUTPUT LINE NOT FULL
          RJ     =XWRLINE
          SB6    OLINE
          SB7    OLINE+7
          EQ     PRINDC1

 PRINDC2  SB7    OLINE
          EQ     B6,B7,PRINDC3    IF NO PARTIAL LINE
          BX6    X6-X6
          SA6    B6
          RJ     =XWRLINE
 PRINDC3  SA1    =10H /PASS
          BX6    X1
          SA6    X5
          SB6    X5+B1
          SB7    OLINE+7
          SA0    NDDIT+2
 PRINDC4  SA1    A0               NEXT ITERATION NODE COUNT
          ZR     X1,PRINDC5
          RJ     =XRDRCDD
          SA6    B6
          SB6    B6+B1
          SA0    A0+B1
          LT     B6,B7,PRINDC4
          RJ     =XWRLINE
          SB6    OLINE
          SB7    OLINE+7
          EQ     PRINDC4

 PRINDC5  SB7    OLINE
          EQ     B6,B7,PRINDC6
          BX6    X6-X6
          SA6    B6
          RJ     =XWRLINE
 PRINDC6  SA1    =10H TIMING
          BX6    X1
          SA6    OLINE
          SA1    ATMPM
          SA2    PRINDCA
          SX3    500              SO WILL ROUND TO NEAREST SECOND
          IX1    X1+X3
          FX1    X1*X2
          RJ     =XRDRCDD
          SA6    A6+B1
          SA1    NTRLO
          RJ     =XRDRCOD
          SA6    A6+B1
          BX6    X6-X6
          SA6    A6+B1
          RJ     =XWRLINE
          SA1    =10H ALRGMP WD
          BX6    X1
          SA6    X5
          SA1    NDSCR
          SB6    5
 PRINDC7  RJ     =XRDRCDD
          SA6    A6+B1
          SB6    B6-B1
          SA1    A1+B1
          PL     B6,PRINDC7
          BX6    X6-X6
          SA6    A6+B1
          RJ     =XWRLINE
          EQ     PRINDC           RETURN

 PRINDCA  CON    1S48/1000+1
 PRIPRV   SPACE  4,32
***       PRIPRV - PRINT PRINCIPLE VARIATION.
*
*         ENTRY  (PVARY) = PRINCIPLE VARIATION.
*
*         CALLS  WRLINE.
*         USES   ALL REGISTERS, OLINE.
*

          ENTRY  PRIPRV
 PRIPRV   EQ     *+400000B
          SA0    PVARY
          SB6    OLINE
          SB7    OLINE+7
          SX5    B6
          BX6    X6-X6
          SA6    B7
 PRIPRV1  SA1    A0
          SA0    A0+B1
          ZR     X1,PRIPRV2       IF END OF VARIATION
          BX6    X1
          SA6    B6
          SB6    B6+B1
          LT     B6,B7,PRIPRV1
          RJ     =XWRLINE
          SB6    OLINE
          SB7    OLINE+7
          EQ     PRIPRV1
 PRIPRV2  SB7    X5
          EQ     B6,B7,PRIPRV     IF NO PARTIAL LINE, RETURN
          BX6    X6-X6
          SA6    B6               STORE END OF LINE
          RJ     =XWRLINE
          EQ     PRIPRV
 PRIWEI   SPACE  4,30
***       PRIWEI - PRINT WEIRD WORD.
*
*         CALLS  WRLINE.
*

          ENTRY  PRIWEI
 PRIWEI   EQ     *+400000B
          SA1    =XECRERR
          ZR     X1,PRIWEI1       IF NO ECS ERRORS
          SX5    =C* I AM LOSING MY MIND.*
          RJ     =XWRLINE
 PRIWEI1  SA1    WIERD
          BX0    X1
          LX0    59-S.WRTC
          PL     X0,PRIWEI2
          SX5    =C* TIME SURE FLIES.*
          RJ     =XWRLINE
 PRIWEI2  LX0    S.WRTC-S.WRA0
          PL     X0,PRIWEI3       IF NO PLY ZERO RE-ASPIRE
          SX5    =C* OH, YOU HAD THAT.*
          RJ     =XWRLINE
 PRIWEI3  LX0    S.WRA0-S.WRA1
          PL     X0,PRIWEI4       IF NO PLY ONE RE-ASPIRE
          SX5    =C* BE CAREFUL.*
          RJ     =XWRLINE
 PRIWEI4  SA1    SCORE
          SA2    CSIDE
          BX1    X2-X1
          SX4    INFIN-NPLY*2000B
          IX4    X4-X1
          PL     X4,PRIWEI5       IF NO MATE IN SIGHT
          SX3    INFIN
          IX1    X3-X1
          AX1    11
          ZR     X1,PRIWEI5       IF MATED ALREADY
          RJ     =XRDRCDD         CONVERT NUMBER OF MOVES
          SA1    PRIWEIA
          IX6    X1+X6
          BX7    X7-X7
          SA6    OLINE
          SA7    A6+B1
          SX5    A6
          RJ     =XWRLINE
 PRIWEI5  LX0    S.WRA1-S.WREZ
          PL     X0,PRIWEI6       IF NOT OBVIOUS
          SX5    =C* THAT WAS EASY.*
          RJ     =XWRLINE
 PRIWEI6  EQ     PRIWEI

 PRIWEIA  CON    8L MATE IN-8L
 PRIMOV   SPACE  4,30
***       PRIMOV - PRINT MOVES ARRAY.
*
*         CALLS  WRLINE, RDRCDD, ENGGEN.
*

          ENTRY  PRIMOV
 PRIMOV   EQ     *+400000B
          SX6    MOVES-LE.MOV
          SA6    PRIMOVA          INITIALIZE MOVES POINTER
 PRIMOV1  SA1    PRIMOVA
          SA2    LINDX
          SX6    X1+LE.MOV        ADVANCE MOVES POINTER
          SA6    A1
          IX7    X6-X2
          PL     X7,PRIMOV        IF DONE
          SA1    X6               THE MOVE
          SB2    NBOARD
          RJ     =XENGGEN         TRANSLATE TO ENGLISH
          SA6    OLINE
          SX5    A6
          SA1    A1+B1            ITS SCORE
          RJ     =XRDRCDD
          SA6    A6+B1
          BX6    X6-X6
```

```
          SA6    A6+B1
          RJ     =XWRLINE
          EQ     PRIMOV1

 PRIMOVA  DATA   0                MOVES ARRAY POINTER
          END
SWICMD
          IDENT  SWICMD
          SST
          LIST   F,X
 SWICMD   TITLE  SWICMD - CHANGE GLOBAL VARIABLES AND SWITCHES.
          SPACE  4
 LETMAC   MACRO  A,B
          USE
          VFD    42/0L!A,18/B
          USE    *
          ENDM
*CALL     IOCOM
 SWICMD   SPACE  4,88
***       SWICMD - CHANGE SWITCHES.
*



          ENTRY  SWICMD
 SWICMD   RJ     =XRDRSFT         SKIP FIRST TOKEN
          RJ     =XRDRGNT         GET TOKEN
          ID     X6,SWICMD1       IF NO TOKEN
          SA2    SWICMDA
          RJ     =XRDRSKT         SEARCH KEYWORD TABLE
          SX5    =C* UNDEFINED SWITCH NAME.*
          RJ     =XERRMSG
 SWICMD1  RJ     =XREREAD         RETURN

 SWICMDA  BSS    0
 SWTYPE   MICRO  2,, 'SWTYPE'
          ECHO   1,TY=('SWTYPE')
          CMND   TY,SWIC!TY
          DATA   0
 SWICIO   SPACE  4,30
 IMLAC    IF     DEF,IMLAC
**        SWICIM - SWITCH IMLAC.

 SWICIM   SB6    S.IML
          EQ     SWICIO
 IMLAC    ENDIF

 ONELINE  IF     DEF,ONELINE
**        SWICOL - SWITCH ONE-LINE.

 SWICOL   SB6    S.ONL
          EQ     SWICIO
 ONELINE  ENDIF

**        SWICDI - SWITCH DISPLAY.

 SWICDI   SB6    S.DIS
          EQ     SWICIO

**        SWICCB - SWITCH CHESS BOARD DRIVER.

 CBD      IF     DEF,CBD
 SWICCB   SYSTEM CBD
          SB6    S.DSD
          JP     SWICIO
 CBD      ENDIF

**        SWICDS - SWITCH DSD.

 SWICDS   SB6    S.DSD
          EQ     SWICIO

**        SWICCA - SWITCH CARDS.

 SWICCA   SB6    S.CAR
*         EQ     SWICIO

**        SWICIO - SWITCH I/O MODE.
*

 SWICIO   SA3    SWICH
          SX5    B1
          LX5    X5,B6
          SA2    SWICIOA
          BX6    -X2*X3
          BX7    X5+X6
          LX5    59-S.IML
          PL     X5,SWICIO1       IF NOT SWITCH TO IMLAC
          LX5    S.DIS-59
          BX7    X5+X7            SET DISPLAY ALSO
 SWICIO1  SA7    A3               UPDATE SWICH
          SA2    A2+B1
          BX6    X7-X3
          BX6    X2*X6
          ZR     X6,SWICIO2       IF NO DISPLAY MODE CHANGE
          RJ     =XDUDDROP        DROP DISPLAYS
          SA1    SWICH
          LX1    59-S.DIS
          PL     X1,SWICIO2       IF NOT DISPLAY ON
          RJ     =XDUDLOAD
 SWICIO2  RJ     =XREREAD

 SWICIOA  BSS    0

          IF     DEF,ONELINE,2
          POS    S.ONL+1
          VFD    1/1

          IF     DEF,IMLAC,2
          POS    S.IML+1
          VFD    1/1

          ECHO   ,B=(CAR,DSD)
          POS    S.B+1
          VFD    1/1
          ENDD

          POS    S.DIS+1
          IF     DEF,DISPLAY,2
          VFD    1/1
          SKIP   1
          VFD    1/0

          IF     DEF,IMLAC,2
          POS    S.IML+1
          VFD    1/1

          POS    S.DIS+1
          IF     DEF,DISPLAY,2
          VFD    1/1
          SKIP   1
          VFD    1/0

 SWIALL   SPACE  4,27
**        SWIALL - SWITCH ALL OTHER SWITCHES.
*
 SWIALL   SX2    A2
          SX3    SWICMDA
          IX6    X2-X3            BIT NUMBER
          SA6    SWIALLA
          RJ     =XRDRGNT         GET NEXT TOKEN
          ID     X6,SWIALL1       IF TERMINATOR
          SA2    SWIALLB
          RJ     =XRDRSKT         SEARCH KEYWORD TABLE
 SWIALL1  SA3    SWICH            IF NOT ON/OFF
          SA4    SWIALLA
          SX5    B1
          SB6    X4
          LX5    X5,B6
          BX6    X3-X5            COMPLEMENT BIT
          SA6    A3
          RJ     =XREREAD         RETURN TO READER

 SWIALLA  DATA   0                BIT NUMBER

 SWIALLB  CMND   ON,SWICON
          CMND   OF,SWICOF
          DATA   0
 SWICON   SPACE  4,13
**        SWICON - SWITCH ON.
*
 SWICON   SA3    SWICH
          SA4    SWIALLA
          SX5    B1
          SB6    X4
          LX5    X5,B6
          BX6    X3+X5
          SA6    A3
          RJ     =XREREAD
 SWICOF   SPACE  4,13
**        SWICOF - SWITCH OFF.
*

 SWICOF   SA3    SWICH
          SA4    SWIALLA
          SX5    B1
          SB6    X4
          LX5    X5,B6
          BX6    -X5*X3
          SA6    A3
          RJ     =XREREAD

          ECHO   2,TY=('SWTYPE')
          IF     -DEF,SWIC!TY,1
 SWIC!TY  EQU    SWIALL

 LETCMD   SPACE  4,88
***       LETCMD - CHANGE VARIABLES.
*



          ENTRY  LETCMD
 LETCMD   RJ     =XRDRSFT         SKIP FIRST TOKEN
          RJ     =XRDRGNT         GET NEXT TOKEN
          ID     X6,LETCMD3       IF NONE
          MX5    42
          SA2    LETCMDA
 LETCMD1  ZR     X2,LETCMD3       IF INVALID
          BX3    X2-X7
          BX3    X3*X5
          ZR     X3,LETCMD2       IF FOUND
          SA2    A2+B1
          EQ     LETCMD1

 LETCMD2  SB7    X2               DEFAULT RADIX
          RJ     =XRDRGNT
          ID     X6,LETCMD4       IF NO VALUE
          BX5    X7
          RJ     =XRDRDXB         CONVERT
          NZ     X4,LETCMD4       IF ASSEMBLY ERROR
          SB2    LET
          SB2    B2-LETCMDA
          SA6    A2+B2            UPDATE LET
          SA1    MOVNUM
          NZ     X1,LETCMD6       IF NOT START OF GAME
          RJ     =XINITTC         INITIALIZE TIME CONTROL
          EQ     LETCMD6          RETURN

 LETCMD3  SX5    =C* INVALID VARIABLE.*
          EQ     LETCMD5

 LETCMD4  SX5    =C* INVALID VALUE.*

 LETCMD5  RJ     =XERRMSG
 LETCMD6  RJ     =XREREAD

 B        EQU    0
 D        EQU    1
 LETCMDA  BSS    0
 LETMIC   MICRO  1,, LETMAC
*CALL     LET

          DATA   0                END OF TABLE
 PARCMD   SPACE  4,88
***       PARCMD - PARAMETER PRINTOUT.
*



          ENTRY  PARCMD
 PARCMD   RJ     =XRDRSFT         SKIP FIRST TOKEN
          RJ     =XRDRGNT         GET NEXT TOKEN
          SX6    1
          SA6    PARCMDA          FLAG NO PARAMETERS PRINTED
          SA7    PARCMDB          SAVE PARTIAL TARGET
          SA1    =20H*********5555555555
          SA2    A1+B1
          BX3    X7-X1            00 WHERE 47
          BX4    -X2*X3           00 WHERE 47 OR 07
          IX4    X4-X1            37 WHERE 47 OR 07, 4X OTHERWISE
          BX4    -X4*X7           40 WHERE 47, 3X OTHERWISE
          BX3    -X2*X7           00 WHERE 00 OR 40
          IX3    X3-X2            37 WHERE 00 OR 40, 4X OTHERWISE
          BX3    X3+X7            37 WHERE 00, 4X OTHERWISE
          BX3    -X3+X4           40 WHERE 00 OR 47, 3X OTHERWISE
          BX3    X2*X3            40 WHERE 00 OR 47, 00 OTHERWISE
          BX4    X3
```

```
          LX4    -5               01 WHERE 00 OR 47
          IX4    X3-X4            37 WHERE 00 OR 47
          BX6    X3+X4            77 WHERE 00 OR 47, 00 OTHERWISE
          SA6    A7+B1            SAVE MASK
 PARCMD1  SA0    SWICMDA
 PARCMD2  MX1    12
          SA2    PARCMDB          VALUE
          SA3    A2+B1            MASK
          BX3    -X3*X1           REAL MASK
 PARCMD3  SA4    A0
          ZR     X4,PARCMD4       IF END OF SWITCHES
          SA0    A0+B1            ADVANCE THROUGH SWITCHES
          BX6    X2-X4
          BX6    X6*X3
          NZ     X6,PARCMD3       IF NOT TO BE PRINTED
          SA6    PARCMDA          CLEAR PRINT FLAG
          RJ     =XPARSWI         PRINT SWITCH
          EQ     PARCMD2          LOOP THROUGH ALL SWITCHES

 PARCMD4  SA0    LETCMDA          PARAMETER TABLE
 PARCMD5  MX1    36
          SA2    PARCMDB
          SA3    A2+B1
          BX3    -X3*X1
 PARCMD6  SA4    A0               LET NAME
          ZR     X4,PARCMD7       IF END OF LETS
          SA0    A0+B1
          BX6    X2-X4            COMPARE
          BX6    X6*X3            EXTRACT VALUE UNDER MASK
          NZ     X6,PARCMD6       IF NOT TO BE PRINTED
          SA6    PARCMDA          CLEAR PRINT FLAG
          RJ     =XPARONE         PRINT PARAMETER
          EQ     PARCMD5          LOOP THROUGH REST OF LETS

 PARCMD7  SA1    PARCMDA          CHECK PRINT FLAG
          ZR     X1,PARCMD8       IF SOMETHING PRINTED
          BX6    -X6-X6
          SA6    PARCMDB+1        PRINT EVERYTHING
          EQ     PARCMD1          TRY AGAIN

 PARCMD8  RJ     =XREREAD         RETURN

 PARCMDA  CON    0                PRINTED FLAG

 PARCMDB  CON    0                VALUE
          CON    0                MASK
 PARSWI   SPACE  4,40
**        PARSWI - PRINT ONE SWITCH.
*
*         ENTRY  (A4) = BIT NUMBER OF SWITCH + SWICMDA.
*
*         USES   OLINE, ALL REGISTERS EXCEPT A0 AND X0.
*         CALLS  WRLINE.
*



 PARSWI   EQ     *+400000B
          SB2    A4-SWICMDA       BIT NUMBER
          SA2    SWICH
          SB3    B2+B2            2*BIT NUMBER
          SA1    PARSWIA+B3       SWITCH NAME
          BX6    X1
          SA6    OLINE
          SX7    B1
          AX6    X2,B2
          BX6    X6*X7            EXTRACT BIT
          SA2    PARSWIB+X6       ON / OFF
          SA1    A1+B1
          BX6    X1+X2
          SA6    A6+B1
          SX5    A6-B1
          RJ     =XWRLINE
          EQ     PARSWI           RETURN

 PARSWIA  BSS    0
 I        SET    1
          ECHO   ,TY=('SWTYPE')
 N        MICRO  I,10,*'SWDISP'*
          VFD    60/10L'N'
 N        MICRO  I+10,2,*'SWDISP'*
          VFD    12/2L'N',12/0H!TY,36/0
 I        SET    I+12
          ENDD

 PARSWIB  VFD    48/4R OFF,12/0
          VFD    48/4R ON ,12/0



 PARONE   SPACE  4,35
**        PARONE - PRINT ONE VARIABLE.
*
*         ENTRY  (X4) = 42/0L NAME, 18/MODE
*                (A4) = ADDRESS OF VARIABLE + LETCMDA - LET
*
*         USES   OLINE, ALL REGISTERS EXCEPT A0 AND X0.
*         CALLS  WRLINE, RDRSFN, RDRCOD, RDRCDD.
*



 PARONE   EQ     *+400000B
          MX5    42
          BX1    X4*X5            REMOVE JUNK
          RJ     =XRDRSFN         SPACE FILL NAME
          LX6    -12
          SA6    OLINE
          SB2    LET
          SB2    B2-LETCMDA
          SA1    A4+B2            VALUE
          SX4    X4               MODE
          ZR     X4,PARONE1       IF OCTAL
          RJ     =XRDRCDD         CONVERT DECIMAL DIGITS
          SX7    1RD
          EQ     PARONE2

 PARONE1  RJ     =XRDRCOD         CONVERT OCTAL DIGITS
          SX7    1RB
 PARONE2  SA6    OLINE+1
          LX7    54
          SA7    A6+B1
          SX5    A6-B1
          RJ     =XWRLINE
          EQ     PARONE           RETURN
          END
MINENG
          IDENT  MINENG
          SST
 MINENG   TITLE  MINENG - GENERATE MINIMUM ENGLISH NOTATION.
          LIST   G
          NOREF  I
*CALL     BOARD
*CALL     SQRLST
*CALL     IOCOM
 SNTX     SPACE  4
 SNTX     MACRO  SYN,M36

 LD       SET    0
 LS       SET    0
 LF       SET    0
 LN       SET    0
 RD       SET    0
 RS       SET    0
 RF       SET    0
 RN       SET    0

 SN       MICRO  1,, SYN
 N        MICCNT SN
 SIDE     MICRO  1,, L
 I        SET    1

          DUP    N
 C        MICRO  I,1, SYN
          IFC    EQ, 'C' / ,1
 C        MICRO  1,, D
          IFC    EQ, 'C' - ,2
 C        MICRO  1,,
 SIDE     MICRO  1,, R
          IFC    EQ, 'C' * ,2
 C        MICRO  1,,
 SIDE     MICRO  1,, R
 'SIDE''C'        SET    1
 I        SET    I+1
          ENDD

 RES      MICRO  1,,M36!1!M36!1
 LA       SET    LF*700B-LS*700B
 RA       SET    RF*7B-RS*7B

          VFD    1/'RES',3/0,1/LD,1/LS,1/LF,1/LN,1/RD,1/RS,1/RF,1/RN
          VFD    3/LN*7,3/LF*7,3/RN*7,3/RF*7
          VFD    12/LA,12/RA,12/LA

          ENDM
 MINENG   SPACE  4,88
***       MINENG - GENERATE MINIMUM ENGLISH NOTATION.
*
*         ENTRY  (BSTMV) = THE MOVE.
*                (MOVES) = ALL LEGAL MOVES.
*                (BOARD) = POSITION BEFORE MOVE IS MADE.
*                (X7) = FIRST WORD OF MOVE MESSAGE.
*
*         EXIT   (MOVMS+1 - MOVMS+4) = ENGLISH NOTATION.
*
*         USES   ALL REGISTERS.
*         CALLS  ADDCHR, ADDSQR, UNAMBG.
*



          ENTRY  MINENG
 MINENG   EQ     *+400000B
          SA7    MOVMS
          SX7    1R-
          LX7    54
          MX0    54
          SB7    48
          SA1    BSTMV
          BX2    -X0*X1           TO SQUARE
          LX1    -S.MFR
          BX3    -X0*X1           FROM SQUARE
          LX1    59-S.CAS+S.MFR
          PL     X1,MINENG1       IF NOT A CASTLE MOVE
          BX4    X1
          LX4    S.CAS-S.PRO
          NG     X4,MINENG1       IF PROMOTION

*         PROCESS CASTLE.

          SX2    3RO-O
          LX2    36
          BX7    X7+X2
          SB7    B7-18
          LX1    S.CAS-S.ENP
          PL     X1,SUFFIX        IF NOT LONG CASTLE
          LX2    -12
          BX7    X7+X2
          SB7    B7-12
          EQ     SUFFIX

*         NOT A CASTLE.

 MINENG1  SB2    MOVES-LE.MOV     FWA-LE.MOV MOVES ARRAY
          SA4    SWICH
          LX4    59-S.ALG
          NG     X4,MINENG9       IF ALGEBRAIC NOTATION
          SB3    B0               LIST POINTER
          SA4    LINDX
          SB5    X4               LWA+LE.MOV MOVES
          LX1    S.CAS-S.CAP
          SA3    BOARD+X3         FROM PIECE
          SA5    MINENGA+X3       FROM PIECE NAME
          LX5    48
          BX7    X7+X5
          SB7    B7-6
          PL     X1,MINENG4       IF NOT A CAPTURE

*         PROCESS CAPTURE.

          LX1    S.CAP-59         RESTORE MOVE
          SB6    B0
          SA2    BOARD+X2         TO PIECE
          AX4    X2,B1
          NZ     X4,MINENG2       IF NOT PAWN CAPTURE
          SB6    B1
 MINENG2  SB2    B2+LE.MOV
          GE     B2,B5,MINENG3    IF DONE WITH MOVES ARRAY SEARCH
          SA4    B2               A MOVE
          BX6    X4-X1
          MX5    -12
          BX6    -X5*X6
          ZR     X6,MINENG2       IF SAME TO AND FROM SQUARES
          LX4    59-S.ILL
          NG     X4,MINENG2       IF ILLEGAL MOVE
          BX6    -X0*X4           TO SQUARE
          LX4    S.ILL-59-S.MFR
          BX5    -X0*X4           FROM SQUARE
```

```
          LX4    59-S.CAP+S.MFR
          PL     X4,MINENG2        IF NOT A CAPTURE
          SA4    BOARD+X6          TO PIECE
          SA5    BOARD+X5          FROM PIECE
          BX4    X4-X2
          BX5    X5-X3
          AX4    X4,B6
          NZ     X4,MINENG2        IF NOT SAME TYPE ON FROM SQUARE
          NZ     X5,MINENG2        IF NOT SAME TYPE ON TO SQUARE

*         FOUND AN AMBIGUITY.

          SX6    B3
          SA6    B2+B1
          SB3    B2
          EQ     MINENG2

*         DONE WITH SEARCH FOR AMBIGUOUS CAPTURES.

 MINENG3  SA2    MINENGD
          RJ     =XUNAMBG          FIND MINIMUM SQUARE NAMES
          LX6    -4
          SX2    X6                FROM SQUARE FLAGS
          LX1    -S.MFR
          BX6    X6-X2
          BX3    -X0*X1
          LX1    S.MFR
          RJ     =XADDSQR
          SX5    1R*
          RJ     =XADDCHR
          BX3    -X0*X1
          LX6    4
          SX2    X6
          SA5    BOARD+X3
          SA5    MINENGA+X5        TO PIECE NAME
          RJ     =XADDCHR
          RJ     =XADDSQR
          EQ     SUFFIX

*         NOT A CAPTURE NOR CASTLE.  PROCESS NON-CAPTURE.

 MINENG4  SX6    7                 TO REFLECT SQUARE
          LX1    S.CAP-59          RESTORE MOVE
 MINENG5  SB2    B2+LE.MOV
          GE     B2,B5,MINENG8     IF DONE WITH MOVES ARRAY SEARCH
          SA4    B2                A MOVE
          MX5    -12
          BX4    X4-X1
          BX4    -X5*X4
          ZR     X4,MINENG5        IF SAME TO AND FROM SQUARES
          SA4    B2
          BX5    -X0*X4            TO SQUARE
          BX5    X5-X2
          LX4    59-S.ILL
          NG     X4,MINENG5        IF ILLEGAL MOVE
          LX4    S.ILL-59-S.MFR
          ZR     X5,MINENG6        IF SAME TO SQUARE
          BX5    X5-X6
 MINENG6  NZ     X5,MINENG5        IF NOT SAME NOR REFLECTION
          BX5    -X0*X4            FROM SQUARE
          SA5    BOARD+X5          FROM PIECE TYPE
          BX5    X5-X3
          LX4    59-S.CAP+S.MFR
          NZ     X5,MINENG5        IF NOT SAME TYPE MOVING
          NG     X4,MINENG5        IF CAPTURE
          LX4    S.CAP-S.PRO
          NG     X4,MINENG7        IF PROMOTION
          LX4    S.PRO-S.CAS
          NG     X4,MINENG5        IF CASTLE

*         AMBIGUOUS NON-CAPTURE.

 MINENG7  SX6    B3
          SA6    B2+B1
          SB3    B2
          SX6    7                 RESTORE
          EQ     MINENG5

*         DONE WITH SEARCH FOR AMBIGUOUS NON-CAPTURES.

 MINENG8  SA2    MINENGE
          RJ     =XUNAMBG          FIND MINIMUM SYNTAX
          LX6    -4
          SX2    X6
          LX1    -S.MFR
          BX6    X6-X2
          BX3    -X0*X1
          LX1    S.MFR
          RJ     =XADDSQR          ADD SQUARE NAME
          SX5    1R-
          RJ     =XADDCHR
          BX3    -X0*X1
          LX6    4
          SX2    X6
          RJ     =XADDSQR
          EQ     SUFFIX

*         ALGEBRAIC NOTATION.

 MINENG9  LX1    S.CAS-59          RESTORE MOVE
          MX6    -3
          BX5    -X6*X1
          LX1    -3
          BX4    -X6*X1
          LX1    -3
          BX3    -X6*X1
          LX1    -3
          BX2    -X6*X1
          LX2    42
          LX3    48
          LX4    24
          LX5    30
          BX6    X2+X3
          BX7    X4+X5
          BX6    X6+X7
          SA2    =6L-A1-A1
          LX1    59-S.CAP+9
          MX7    1
          BX7    X1*X7
          LX7    36-59
          IX2    X2+X7
          IX7    X2+X6
          SB7    54-36
          EQ     SUFFIX

          DATA   1RK,1RQ,1RB,1RN,1RR,1RP
 MINENGA  DATA   1RP,1RP           BASE FOR PIECE NAMES
 MINENGB  DATA   1RR,1RN,1RB,1RQ   BASE FOR FILE NAMES
 MINENGC  DATA   1RK,1RB,1RN,1RR   BASE FOR SIDE NAMES

 MINENGD  SNTX   P*Q,*
          SNTX   P*Q/N
          SNTX   P/N*Q
          SNTX   P*Q/F
          SNTX   P/F*Q
          SNTX   P*Q/FN,*
          SNTX   P/FN*Q,*
          SNTX   P/SF*Q
          SNTX   P*Q/SF
          SNTX   P*Q/SFN,*
          SNTX   P/SFN*Q,*
          SNTX   P/N*Q/N
          SNTX   P/N*Q/F
          SNTX   P/F*Q/N
          SNTX   P/F*Q/F
          SNTX   P/FN*Q/N
          SNTX   P/N*Q/FN
          SNTX   P/FN*Q/F
          SNTX   P/F*Q/FN
          SNTX   P/SF*Q/N
          SNTX   P/N*Q/SF
          SNTX   P/SF*Q/F
          SNTX   P/F*Q/SF
          SNTX   P/N*Q/SFN
          SNTX   P/SFN*Q/N
          SNTX   P/R*Q/SFN
          SNTX   P/SFN*Q/R
          SNTX   P/FN*Q/FN,*
          SNTX   P/SF*Q/FN
          SNTX   P/FN*Q/SF
          SNTX   P/SF*Q/SF
          SNTX   P/SFN*Q/FN,*
          SNTX   P/FN*Q/SFN,*
          SNTX   P/SFN*Q/SF
          SNTX   P/SF*Q/SFN
          SNTX   P/SFN*Q/SFN,*

 MINENGE  SNTX   P-FN,*
          SNTX   P-SFN,*
          SNTX   P/N-FN
          SNTX   P/F-FN
          SNTX   P/N-SFN
          SNTX   P/F-SFN
          SNTX   P/FN-FN,*
          SNTX   P/SF-FN
          SNTX   P/FN-SFN,*
          SNTX   P/SF-SFN
          SNTX   P/SFN-SFN,*
 ADDCHR   SPACE  4,22
**        ADDCHR - ADD CHARACTER TO OUTPUT STREAM.
*
*         ENTRY  (X5) = CHARACTER.
*                (X7) = OUTPUT WORD BEING ASSEMBLED.
*                (A7) = NEXT OUTPUT WORD ADDRESS -1.
*                (B7) = NEXT CHARACTER SHIFT COUNT.
*
*         EXIT   X7, A7, AND B7 ARE UPDATED.
*
*         USES   X5.
*


 ADDCHR   EQ     *+400000B
          LX5    X5,B7
          BX7    X7+X5
          SB7    B7-6
          PL     B7,ADDCHR         IF ROOM FOR MORE, RETURN
          SB7    54
          SA7    A7+B1
          BX7    X7-X7
          EQ     ADDCHR
 ADDSQR   SPACE  4,65
**        ADDSQR - ADD SQUARE NAME TO OUTPUT STREAM.
*
*         ENTRY  (X2) = 1/DELIMITER,1/SIDE,1/FILE,1/RANK
*                (X3) = SQUARE NUMBER.
*                (MSIDE) = SIDE TO MOVE.
*                (SWICH, S.M36) = CHESS 3.6 SYNTAX MODE SWITCH.
*                (X7, A7, B7) = OUTPUT STREAM POINTERS.
*                DELIMITER = 1, IF */* OR *()* REQUIRED.
*                SIDE = 1, IF *K* OR *Q* REQUIRED.
*                FILE = 1, IF *R*, *N*, OR *B* REQUIRED.
*                RANK = 1, IF RANK NUMBER REQUIRED.
*
*         EXIT   */SFN* ADDED TO OUTPUT STREAM.
*
*         USES   X2, X3, X4, X5, A4, AND A5.
*         CALLS  ADDCHR.
*


 ADDSQR   EQ     *+400000B
          ZR     X2,ADDSQR         IF NOTHING REQUIRED
          LX2    59-3
          PL     X2,ADDSQR1        IF NO DELIMITER
          SA4    SWICH
          LX4    -S.M36
          SX5    B1
          BX4    X4*X5
          SX5    X4+1R/            *(* IF 3.6 MODE
          RJ     =XADDCHR
 ADDSQR1  LX2    3-2
          PL     X2,ADDSQR2        IF SIDE NOT REQUIRED
          MX4    -3
          BX5    -X4*X3            FILE
          LX4    X5,B1             FILE*2
          SX4    X4-7              FILE*2-7
          AX4    1                 (FILE*2-7)/2
          ZR     X4,ADDSQR2        IF CENTRAL FILE
          MX5    -1
          AX4    59
          BX5    X5*X4             -1 IF Q, 0 IF K
          SA5    MINENGC+X5
          RJ     =XADDCHR
 ADDSQR2  LX2    2-1
          PL     X2,ADDSQR3        IF FILE NOT REQUIRED
          MX4    -3
          BX4    -X4*X3            FILE
          SA5    X4+MINENGB        FILE NAME
          RJ     =XADDCHR
 ADDSQR3  LX2    1-0
          PL     X2,ADDSQR4        IF RANK NOT REQUIRED
          AX3    3                 RANK
          SA4    MSIDE
```

```
         AX4   59
         MX5   -3
         BX4   -X5*X4            7 IF BLACK, 0 IF WHITE
         BX5   X3-X4             COMPLEMENT IF BLACK
         SX5   X5+1R1            RANK NAME
         RJ    =XADDCHR
ADDSQR4  LX2   0-3
         SA4   SWICH
         LX4   59-S.M36
         BX4   X4*X2
         PL    X4,ADDSQR         IF *)* NOT REQUIRED, RETURN
         SX5   1R)
         RJ    =XADDCHR
         EQ    ADDSQR
UNAMBG   SPACE 4,88
**       UNAMBG - FIND MINIMUM UNAMBIGUOUS SQUARE DESCRIPTORS.
*
*        ENTRY  (X1) = MOVE.
*               (X2) = FIRST WORD OF ALTERNATIVES TABLE.
*               (X2) = ((A2))
*               (B3) = ADDRESS OF FIRST AMBIGUITY.
*               ALTERNATIVES TABLE FORMAT:
*               12/FLAGS,12/MASK,12/REF1,12/REF2,12/REF3
*               FLAGS: 1/ALLOWED IN 3.6 MODE.
*                      3/UNUSED
*                      1/DELIMITER REQUIRED ON LHS
*                      1/SIDE REQUIRED ON LHS
*                      1/FILE REQUIRED ON LHS
*                      1/RANK REQUIRED ON LHS
*                      4/SIMILAR FOR RHS
*               MASK: BITS OVER FIELDS IN MOVE SPECIFIED IN SYNTAX.
*               REFI: REFLECTIONS FOR FILE BUT NOT SIDE SPECIFIED.
*
*        EXIT   (X6) = FLAGS FROM FIRST UNAMBIGUOUS ALTERNATIVE.
*
*        USES   X2, X3, X4, A2, A5, AND B2.
*


UNAMBG   EQ    *+400000B
         ZR    B3,UNAMBG5        IF NO AMBIGUITIES
UNAMBG1  SA5   SWICH
         LX5   59-S.M36
         MX6   48
         BX3   -X2*X5
         NG    X3,UNAMBG3        IF NOT ALLOWED BECAUSE 3.6 MODE
         LX2   24
         BX3   -X6*X2            MASK
         LX2   -24               RESTORE
         SB2   B3                HEAD OF LIST
UNAMBG2  SA5   B2                A CONFLICT
         BX4   X1-X5
         BX4   X4*X3
         BX5   X2                REFLECTIONS
         ZR    X4,UNAMBG3        IF STILL AMBIGUOUS
         BX4   X4-X5
         BX4   X4*X3
         ZR    X4,UNAMBG3        IF REFLECTION 1 AMBIGUOUS
         AX5   12
         BX4   X4-X5
         BX4   X4*X3
         ZR    X4,UNAMBG3        IF REFLECTION 2 AMBIGUOUS
         AX5   12
         BX4   X4-X5
         BX4   X4*X3
         NZ    X4,UNAMBG4        IF REFLECTION 3 NOT AMBIGUOUS
UNAMBG3  SA2   A2+B1             TRY NEXT SYNTAX
         EQ    UNAMBG1

UNAMBG4  SA5   B2+B1             NEXT CONFLICT
         SB2   X5
         NZ    X5,UNAMBG2        IF NOT END OF LIST
UNAMBG5  LX2   12
         MX6   -8
         BX6   -X6*X2            EXTRACT FLAGS
         EQ    UNAMBG            RETURN
SUFFIX   SPACE 4,88
**       SUFFIX - APPEND MISCELLANEOUS NONSENSE.
*
*        ENTRY  (BSTMV) = THE MOVE.
*               (X7, A7, B7) = OUTPUT STREAM POINTERS.
*
*        CALLS  ADDCHR.
*


SUFFIX   SA1   BSTMV
         LX1   59-S.PRO
         PL    X1,SUFFIX1        IF NOT PROMOTION
         SX5   4
         SA4   SWICH
         LX4   2-S.M36
         BX4   X4*X5             4 IF 3.6 MODE
         BX4   X4-X5             4 IF NOT 3.6 MODE
         SX5   X4+1R/            = IF NOT 3.6 MODE
         RJ    =XADDCHR
         LX1   3
         MX5   -2
         BX5   -X5*X1
         SA5   MINENGB+X5
         RJ    =XADDCHR

SUFFIX1  SX5   1R.
         RJ    =XADDCHR
         SX5   1R
         RJ    =XADDCHR
         SA1   BSTMV
         LX1   -12
         BX1   -X0*X1
         SX2   M.CAP/10000B+M.PRO/10000B+M.ENP/10000B+M.CAS/10000B
         SX3   M.CAP/10000B+M.ENP/10000B
         BX1   X1*X2
         BX1   X1-X3
         NZ    X1,SUFFIX2        IF NOT ENPASSANT CAPTURE
         SX5   1RE
         RJ    =XADDCHR
         SX5   1RP
         RJ    =XADDCHR
         SX5   1R.
         RJ    =XADDCHR
         SX5   1R
         RJ    =XADDCHR

SUFFIX2  SA1   BSTMV
         LX1   59-S.CHK
         PL    X1,SUFFIX3        IF NOT CHECK
         SX5   1RC
         RJ    =XADDCHR
         SX5   1RH
         RJ    =XADDCHR
         SX5   1RE
         RJ    =XADDCHR
         SX5   1RC
         RJ    =XADDCHR
         SX5   1RK
         RJ    =XADDCHR
         LX1   S.CHK-S.MAT
         NG    X1,SUFFIX4        IF CHECKMATE
         EQ    SUFFIX5

SUFFIX3  LX1   S.CHK-S.MAT
         PL    X1,SUFFIX6        IF NOT STALEMATE
         SX5   1RS
         RJ    =XADDCHR
         SX5   1RT
         RJ    =XADDCHR
         SX5   1RA
         RJ    =XADDCHR
         SX5   1RL
         RJ    =XADDCHR
         SX5   1RE
         RJ    =XADDCHR
SUFFIX4  SX5   1RM
         RJ    =XADDCHR
         SX5   1RA
         RJ    =XADDCHR
         SX5   1RT
         RJ    =XADDCHR
         SX5   1RE
         RJ    =XADDCHR
SUFFIX5  SX5   1R.
         RJ    =XADDCHR
SUFFIX6  SX5   0
         RJ    =XADDCHR
SUFFIX7  ZR    B7,SUFFIX8
         RJ    =XADDCHR
         EQ    SUFFIX7           FILE OUT WORD WITH ZEROS

SUFFIX8  RJ    =XADDCHR
         EQ    MINENG            RETURN

         END
GAMSCR
         IDENT  GAMSCR
         SST
 GAMSCR  TITLE  GAMSCR - PROCESS GAME SCORE.
         B1=1
         LIST   F
*CALL    LET
*CALL    VERSION
*CALL    IOCOM
*CALL    SQRLST
*CALL    RELY
 GAMSCR  TITLE  GAMSCR - RECORD MOVE FOR GAME SCORE.
***      GAMSCR - RECORD MOVE FOR GAME SCORE.
*
*        ENTRY  (X1) = 0, IF FULL LINE MESSAGE.
*               (X1) < 0, IF BLACK MOVE.
*               (X1) > 0, IF WHITE MOVE.
*               (MOVMS) = MOVE MESSAGE.
*               (ILINE) = FULL LINE MESSAGE.
*               (GAMMS) = PARTIAL LINE, IF ANY.
*


         ENTRY  GAMSCR
 GAMSCR  EQ    *+400000B
         SX2   =XGAMFILE
         NZ    X1,GAMSCR2        IF MOVE
         SA3   GAMMS
         ZR    X3,GAMSCR1        IF NO PARTIAL LINE
         WRITEC X2,A3
         RJ    =XEXTEND
         BX6   X6-X6
         SA6   GAMMS             INDICATE NO PARTIAL LINE
         SA6   GAMMS+5           CLEAR TIME
         SA1   GMCNT
         SX6   X1+B1             ADVANCE COUNT OF LINES ON GAMFILE
         SA6   A1
GAMSCR1  WRITEC X2,ILINE         WRITE LINE
         RJ    =XEXTEND
         SA1   GMCNT
         SX6   X1+B1
         SA6   A1
         EQ    GAMSCR            RETURN

*        MOVE.

GAMSCR2  NG    X1,GAMSCR6        IF BLACK MOVE

*        WHITE MOVE.

         SA3   GAMMS
         ZR    X3,GAMSCR3        IF NO WHITE MOVE THERE
         WRITEC X2,A3
         RJ    =XEXTEND
         BX6   X6-X6
         SA6   GAMMS+5           CLEAR TIME
         SA1   GMCNT
         SX6   X1+B1             ADVANCE COUNT
         SA6   A1
GAMSCR3  SA1   MOVNUM            MOVE NUMBER
         SX1   X1+B1
         BX6   X1
         SA6   A1                UPDATE MOVNUM
         RJ    =XRDRCDD          CONVERT DECIMAL DIGITS
         LX6   42                LEFT ADJUST
         SA3   MOVMS+1           -ABCDEFGHI
         MX0   24
         LX3   42                GHI-ABCDEF
         BX4   -X0*X3            0000ABCDEF
         BX6   X0*X6             NNNB000000
         BX1   X6+X4             NNNBABCDEF
         RJ    =XRDRSFN          SPACE FILL
         SA6   GAMMS
         MX0   18
         MX5   54
         LX5   42
         SB3   B1                LOOP COUNT
GAMSCR4  BX1   X0*X3              GHI0000000
         BX3   -X5*X3            00I0000000
         ZR    X1,GAMFIN         IF NOTHING LEFT
         ZR    X3,GAMSCR5        IF NOTHING IN NEXT WORD
         SA3   A3+B1             NEXT WORD
```

```
          LX3    42
          BX4    -X0*X3
          BX1    X4+X1
          RJ     =XRDRSFN
          SA6    A6+B1
          SB3    B3-B1
          PL     B3,GAMSCR4        IF NOT ALL DONE
          EQ     GAMFIN

 GAMSCR5  RJ     =XRDRSFN          SPACE FILL
          SA6    A6+B1
          EQ     GAMFIN

*         BLACK MOVE.

 GAMSCR6  SA3    GAMMS
          NZ     X3,GAMSCR7        IF WHITE MOVE PRESENT
          SA1    MOVNUM
          RJ     =XRDRCDD          CONVERT DECIMAL DIGITS
          LX6    -18
          SA6    GAMMS
          SA1    =10H
          BX6    X1
          SA6    A6+B1
          SA6    A6+B1             BLANK FILL
 GAMSCR7  MX0    6
          SA1    MOVMS+1
          BX1    -X0*X1
          SX6    1R
          LX6    54
          BX1    X6+X1
          MX5    48
          RJ     =XRDRSFN          SPACE FILL
          SA6    GAMMS+3
          BX1    -X5*X1
          ZR     X1,GAMFIN         IF ONE WORD LONG
          SA1    A1+B1
          RJ     =XRDRSFN          SPACE FILL
          SA6    A6+B1
          BX1    -X5*X1
          ZR     X1,GAMFIN         IF DONE
          SA1    A1+B1
          RJ     =XRDRSFN          SPACE FILL
          SA2    A6+B1
          MX0    30
          BX2    -X0*X2
          BX6    X0*X6
          BX6    X6+X2
          SA6    A6+B1
*         EQ     GAMFIN            FINISH UP
 GAMFIN   SPACE  4,40
**        GAMFIN - FINISH GAME SCORE LINE.
*
*         ENTRY  (A6) = ADDRESS OF LAST WORD STORED.
*                (TIMMV) = CPU TIME TO COMPUTE LAST MOVE (MSEC)
*
*         EXIT   GAMMS SPACE FILLED AND TIME APPENDED.
*


 GAMFIN   SA2    =10H
          BX6    X2
          SB2    A6
          SB3    GAMMS+4
          GT     B2,B3,GAMFIN3     IF LAST WORD ALREADY STORED
 GAMFIN1  EQ     B2,B3,GAMFIN2     IF SECOND FROM LAST STORED
          SA6    A6+B1
          SB2    A6
          EQ     GAMFIN1

 GAMFIN2  SA3    B3+B1
          MX0    30
          BX6    X0*X6
          BX3    -X0*X3
          BX6    X6+X3
          SA6    A6+B1
 GAMFIN3  SA1    TIMMV
          ZR     X1,GAMSCR         IF NO TIME, RETURN
          BX6    X6-X6
          SX2    500D
          IX1    X1+X2
          RJ     =XRDRCDD          CONVERT
          LX6    -18
          SA1    GAMMS+5
          MX0    30
          BX1    X0*X1
          BX6    -X0*X6
          BX6    X6+X1
          SA6    A1
          EQ     GAMSCR
 EXTEND   SPACE  4,21
**        EXTEND - EXTEND GAMFILE IF NECESSARY.
*
*         CALLS  PFM=, WNB=.
*


 EXTEND   EQ     *+400000B
          SA1    SWICH
          LX1    59-S.ADI
 KRONOS   IF     -DEF,KRONOS
          PL     X1,EXTEND1        IF NO DAYFILE
 KRONOS   ELSE
          PL     X1,EXTEND         IF NO DAYFILE
 KRONOS   ENDIF
          SX6    GAMMS
          SA1    X6
 +        NZ     X1,*+1
          SX6    =C*  *
          DAYFILE X6,10B
 KRONOS   IF     -DEF,KRONOS
 EXTEND1  SA1    RELSW
          ZR     X1,EXTEND         IF NO NEED TO EXTEND
          RECALL X2                WAIT UNTIL COMPLETE
          SA1    =XGAMFILE+3
          SA3    GAMOUT
          IX6    X1-X3
          ZR     X6,EXTEND         IF NOTHING WRITTEN OUT
          BX6    X1
          SA6    A3
          SX2    GAMFDB
          SX7    30B
          SX6    3RPFE
          RJ     =XPFM=            EXTEND GAMFILE
 KRONOS   ENDIF
          SX2    =XGAMFILE         RESTORE X2
          EQ     EXTEND            RETURN
```

```
 GMPRNT   TITLE  GMPRNT - PRINT THE GAME SCORE.
***       GMPRNT - PRINT THE GAME SCORE.
*
*         ACTION IF GAMFILE EMPTY, RETURN.
*                OTHERWISE, PRINT *COPIES* COPIES OF GAME SCORE AT
*                CENTRAL SITE, AND PRINT GAME SCORE ON OUTPUT.
*


          ENTRY  GMPRNT
 GMPRNT   EQ     *+400000B
          SA1    GMCNT
          ZR     X1,GMPRNT         IF NOTHING ON GAMFILE
          SX1    X1-3
          NG     X1,GMPRNT6        IF NOT MUCH ON GAMFILE
          SA1    CNTMV             NUMBER OF MOVES
          SA2    TIMTO             TOTAL TIME
          SX3    B1
          SA4    SWICH
          LX4    59-S.EXP
          PL     X4,GMPRNT         IF LIBRARY CREATION, RETURN
          IX4    X1-X3
          AX4    59
          BX3    X3*X4             1 IF CNTMV = 0, 0 OTHERWISE
          BX1    X1+X3             MIN(1,CNTMV)
          PX1    X1
          PX2    X2
          NX1    X1
          NX2    X2
          SX5    500D
          FX6    X2/X1             MSEC/MOVE
          PX5    X5
          NX5    X5
          FX6    X6+X5             ROUND
          FX6    X6/X5             2*SEC/MOVE
          UX6    X6,B2
          LX6    X6,B2
          AX1    X6,B1             SEC/MOVE
          RJ     =XRDRCDD          CONVERT
          SA6    GMFOOTB           AVERAGE TIME
          SA1    GAMMS
          ZR     X1,GMPRNT1        IF NO MESSAGE WAITING
          WRITEC =XGAMFILE,A1
          RJ     =XEXTEND
          SA1    GMCNT
          SX6    X1+B1
          SA6    A1
 GMPRNT1  WRITER =XGAMFILE,RECALL
          RJ     =XEXTEND
          SA1    =10H
          BX6    X1
          SA6    OLINE
          SA1    OPNAM             OPPONENTS NAME
          SA2    =10H CDC 176
          SA3    CSIDE
          BX4    X1-X2
          BX3    X3*X4             0 IF COMPUTER IS WHITE
          BX6    X3-X2
          BX7    X3-X1
          SA6    GMHDTWD           SET WHITE NAME
          SA6    GMHDTWF
          SA7    GMHDTWE           SET BLACK NAME
          SA7    GMHDTWG
          SA3    =XOUTPUT+1
          NG     X3,GMPRNT2        IF OUTPUT NOT ALLOCATABLE
          SA3    COPIES
          SX3    X3+B1             NUMBER OF COPIES
          SX5    =XOUTPUT
          RJ     GMPRTW            PRINT TWO WIDE
          EQ     GMPRNT4

 GMPRNT2  OPEN   PRINT,ALTERNR
          SX5    X2
          SA3    COPIES
          ZR     X3,GMPRNT3        IF NO COPIES
          RJ     =XGMPRTW          PRINT TWO WIDE
          WRITER PRINT,RECALL
 KRONOS   IF     -DEF,KRONOS
          DISPOS PRINT,PR,RECALL
 KRONOS   ELSE
          SA1    PRINT             SET PRINT FILE BUSY
          MX6    59
          BX6    X6*X1
          SA6    A1
          SYSTEM LFM,RECALL,A1,400B PRINT DISPOSITION
 KRONOS   ENDIF
 GMPRNT3  SX5    =XOUTPUT
          RJ     =XGMPROW          PRINT ONE WIDE
 GMPRNT4  SA1    =XHARDCPY
          ZR     X1,GMPRNT6        IF NO HARD COPY
          SX5    A1
          SA1    A1+B1
          NG     X1,GMPRNT5        IF NOT ALLOCATABLE
          SX3    B1
          RJ     =XGMPRTW          PRINT TWO WIDE
          EQ     GMPRNT6

 GMPRNT5  RJ     =XGMPROW          PRINT ONE WIDE

 GMPRNT6  SA1    RELSW
          ZR     X1,GMPRNT7        IF NOT RELIABILITY MODE
 KRONOS   IF     -DEF,KRONOS
          SX2    GAMFDB
          SX7    40B
          SX6    3RPFP
          RJ     =XPFM=            PURGE GAMFILE
 KRONOS   ELSE
          PURGE  =XGAMFILE
 KRONOS   ENDIF
 GMPRNT7  CLOSE  =XGAMFILE,UNLOAD,RECALL
 KRONOS   IF     -DEF,KRONOS
          SA1    GMPRNTA
          MX6    42
          BX6    X6*X1
          SA6    A1
          SYSTEM REQ,RECALL,A1     REQUEST(GAMFILE,*PF)
          OPEN   X2,ALTER,RECALL
 KRONOS   ENDIF
          BX6    X6-X6
          SA6    GMCNT
          SA1    RELSW
          ZR     X1,GMPRNT8        IF NOT RELIABILITY
 KRONOS   IF     -DEF,KRONOS
          SX2    GAMFDB
          SX7    20B
          SX6    3RPFC
          RJ     =XPFM=            CATALOG GAMFILE
```

```
 KRONOS   ELSE   1
          DEFINE =XGAMFILE
 GMPRNT8  SA1    =XGAMFILE+3
          BX6    X1
          SA6    GAMOUT
          SA6    A1-B1             SET BUFFER EMPTY
          EQ     GMPRNT

 GMPRNTA  VFD    42/0LGAMFILE,18/0
          CON    1S31+1S28
          CON    0
 GMPRTW   SPACE  4,62
**        GMPRTW - PRINT GAME SCORE TWO COLUMNS WIDE.
*
*         ENTRY  (X3) = NUMBER OF COPIES.
*                (X5) = FET ADDRESS OF OUTPUT FILE.
*                (GMCNT) = NUMBER OF LINES ON GAMFILE.
*



 GMPRTW   EQ     *+400000B
 GMPRTW1  ZR     X3,GMPRTW         IF DONE, RETURN
          SA4    GMCNT
          BX6    X3
          BX7    X4
          SA6    GMPRTWA           NUMBER OF COPIES
          SA7    A6+B1
          REWIND =XGAMFILE,RECALL
          READ   X2
 GMPRTW2  SA1    GMPRTWA+1         NUMBER OF LINES
          SX2    X1-50D
          SX3    X2-50D
          PL     X3,GMPRTW7        IF FULL PAGE
          PL     X2,GMPRTW4        IF MORE THAN ONE COLUMN
 GMPRTW3  RJ     GMHDOW            OUTPUT HEADING ONE WIDE
          RJ     GMBOOW            OUTPUT BODY ONE WIDE
          RJ     GMFOOT            OUTPUT FOOTING
          SA3    GMPRTWA
          SX3    X3-1
          EQ     GMPRTW1           NEXT COPY

*         MORE THAN ONE COLUMN, BUT LESS THAN TWO.

 GMPRTW4  ZR     X2,GMPRTW3        IF EXACTLY ONE COLUMN
          RJ     GMHDTW            OUTPUT HEADING TWO WIDE
          RJ     GMRDLS            READ LEFT SIDE
 GMPRTW5  RJ     GMPRBS            PRINT BOTH SIDES
          PL     X1,GMPRTW5        IF NOT DONE WITH RHS
 GMPRTW6  RJ     GMPRLS            PRINT LEFT SIDE
          PL     X1,GMPRTW6        IF NOT DONE WITH LEFT SIDE
          RJ     GMFOOT            OUTPUT FOOTING
          SA3    GMPRTWA
          SX3    X3-1
          EQ     GMPRTW1           NEXT COPY

*         TWO FULL COLUMNS.

 GMPRTW7  RJ     GMHDTW            OUTPUT HEADING TWO WIDE
          RJ     GMRDLS            READ LEFT SIDE
 GMPRTW8  RJ     GMPRBS            PRINT BOTH SIDES
          PL     X1,GMPRTW8        IF NOT DONE
          SA4    GMPRTWA+1         NUMBER OF LINES
          SX6    X4-100D           NUMBER OF LINES LEFT
          SA6    A4
          NZ     X6,GMPRTW2        IF NOT DONE
          RJ     GMFOOT            OUTPUT FOOTING
          SA3    GMPRTWA
          SX3    X3-1
          EQ     GMPRTW1           NEXT COPY

 GMPRTWA  BSS    1                 NUMBER OF COPIES LEFT
          BSS    1                 NUMBER OF LINES LEFT
 GMHDOW   SPACE  4,23
**        GMHDOW - PRINT ONE COLUMN HEADING.
*
*         ENTRY  (X5) = FET ADDRESS OF OUTPUT FILE.
*
*         CALLS  WTC=, SYS=.
*



 GMHDOW   EQ     *+400000B
          SA1    X5+B1
          NG     X1,GMHDOW1        IF NON-ALLOCATABLE
          WRITEC X5,=1C1
 GMHDOW1  DATE   GMHDOWB
          CLOCK  GMHDOWC
          WRITEC X5,GMHDOWA
          WRITEC X2,GMHDOWE        NAMES
          WRITEC X2,GMHDOWD
          EQ     GMHDOW            RETURN

 GMHDTWA  DATA   30H
 GMHDOWA  DATA   H*          CHESS 'V' GAME SCORE*
 GMHDOWB  BSS    1                 DATE
 GMHDOWC  BSS    1                 TIME
          DATA   0
 GMHDOWD  DATA   C*            WHITE                        BLACK
,            TIME*
 GMBOOW   SPACE  4,18
**        GMBOOW - PRINT BODY ONE WIDE.
*
*         ENTRY  (X5) = OUTPUT FET ADDRESS.
*
*         CALLS  RDC=, WTC=.
*



 GMBOOW   EQ     *+400000B
 GMBOOW1  READC  =XGAMFILE,OLINE+1,7
          NZ     X1,GMBOOW         IF DONE, RETURN
          WRITEC X5,OLINE
          EQ     GMBOOW1
 GMFOOT   SPACE  4,15
**        GMFOOT - PRINT FOOTING.
*
*         ENTRY  (X5) = FET ADDRESS.
*
 GMFOOT   EQ     *+400000B
          WRITEC X5,GMFOOTA
          EQ     GMFOOT

 GMFOOTA  DATA   H*          *
 GMFOOTB  DATA   H*        NN*     AVERAGE TIME
          DATA   C* SECONDS PER MOVE.*
 GMHDTW   SPACE  4,20
**        GMHDTW - PRINT TWO COLUMN HEADINGS.
*
*         ENTRY  (X5) = FET ADDRESS.
*
*         CALLS  WTC=, SYS=.
*



 GMHDTW   EQ     *+400000B
          SA1    X5+B1
          NG     X1,GMHDTW1
          WRITEC X5,=1C1
 GMHDTW1  DATE   GMHDOWB
          CLOCK  GMHDOWC
          WRITEC X5,GMHDTWA
          WRITEC X2,GMHDTWC        NAMES
          WRITEC X2,GMHDTWB
          EQ     GMHDTW

 GMHDTWB  DATA   C*            WHITE                        BLACK
,            TIME    WHITE                        BLACK
, TIME*
 GMHDTWC  DATA   10H
 GMHDTWD  DATA   30HWHITE NAME
 GMHDTWE  DATA   20HBLACK NAME
 GMHDOWE  DATA   10H
 GMHDTWF  DATA   30HWHITE NAME
 GMHDTWG  DATA   10CBLACK NAME
 GMRDLS   SPACE  4,17
**        GMRDLS - READ DATA FOR LEFT COLUMN.
*
*         EXIT   (MOVES) = 50 6-WORD ENTRIES.
*                (A0) = FWA MOVES.
*
*         CALLS  RDH=.
*
 GMRDLS   EQ     *+400000B
          SB6    MOVES
          SX2    =XGAMFILE
 GMRDLS1  READH  X2,B6,6
          SB7    MOVES+50*6
          LT     B6,B7,GMRDLS1
          SA0    MOVES
          EQ     GMRDLS            RETURN
 GMPRBS   SPACE  4,37
**        GMPRBS - PRINT ONE LINE ON BOTH SIDES.
*
*         ENTRY  (A0) = LEFT HAND SIDE POINTER.
*                (X5) = FET ADDRESS OF OUTPUT FILE.
*
*         EXIT   (A0) = NEW LEFT HAND SIDE POINTER.
*                (X1) < 0, IF END OF EITHER SIDE.
*
*         CALLS  RDC=, WTC=.
*



 GMPRBS1  SX1    -1                RETURN END

 GMPRBS   EQ     *+400000B
          SB4    A0
          SB2    5
          SB3    MOVES+50*6
          EQ     B3,B4,GMPRBS1     IF DONE WITH LHS
          READH  =XGAMFILE,OLINE+7,6 READ RIGHT SIDE
          NZ     X1,GMPRBS1        IF NO MORE RIGHT SIDE
          SB6    OLINE+1
          SB7    OLINE+7
 GMPRBS2  SA2    A0
          BX6    X2
          SA6    B6
          SA0    A0+B1
          SB6    B6+B1
          LT     B6,B7,GMPRBS2     MOVE LEFT SIDE
          WRITEC X5,OLINE
          BX1    X1-X1             RETURN SUCESS
          EQ     GMPRBS
 GMPRLS   SPACE  4,22
**        GMPRLS - PRINT LEFT COLUMN FROM MEMORY.
*
*         ENTRY  (A0) = CURRENT LEFT SIDE POINTER.
*                (X5) = OUTPUT FET ADDRESS.
*
*         EXIT   (X1) = 0, IF LINE PRINTED.
*                (A0) = NEW LEFT SIDE POINTER.
*
*         CALLS  WTH=.
*



 GMPRLS   EQ     *+400000B
          SB6    A0
          SB7    MOVES+50*6
          MX1    59
          GE     B6,B7,GMPRLS      IF DONE
          WRITEW X5,(=1H ),1
          WRITEH X2,A0,6
          SA0    A0+6
          SX1    0
          EQ     GMPRLS
 GMPROW   SPACE  4,22
**        GMPROW - PRINT GAME SCORE ONE COLUMN WIDE.
*
*         ENTRY  (X5) = OUTPUT FET ADDRESS.
*
*         CALLS  GMHDOW, GMBOOW, GMFOOT.
*



 GMPROW   EQ     *+400000B
          RJ     GMHDOW            OUTPUT HEADING
          REWIND =XGAMFILE,RECALL
          RJ     GMBOOW            OUTPUT BODY
          RJ     GMFOOT            OUTPUT FOOTING
          EQ     GMPROW

 MISC     SPACE  4,88
 PRINT    FILEC  SQRLST,NODEBK-SQRLST,FET=6

          HERE
          END
CHESS30
          IDENT  CHE$$30
 CHESS30  TITLE  CHESS30 - MORE MISCELLANEOUS COMMANDS.
          SST
          SYSCOM B1
          LIST   X,F
          CON    =XECHES30         ECS PLUG TABLE
```

```
 ECHES30  ECERTB
 30       THIS
          EXT    ECWERR
 EWE=     EQU    ECWERR
*CALL     RELY
*CALL     LET
*CALL     IOCOM
*CALL     MEMORY
 STRCMD   SPACE  4,88
***       STRCMD - PROCESS STRIP COMMAND.
*


 STRCMD   RJ     =XRDRIOC          INITIALIZE I/O COMMAND
          RJ     =XRDRGNT          GET NEXT TOKEN
          RJ     =XRDRVFN          VERIFY NEXT FILE NAME
          SA7    FET2
          DATE   STRCMDB,RECALL
          CLOCK  STRCMDC,RECALL
          OPEN   FET1,ALTER,RECALL
          WRITEW X2,STRCMDA,16     WRITE HEADER
          OPEN   FET2,ALTER,RECALL
          WRITEW X2,STRCMDA,16     WRITE HEADER
          RJ     =XMSRWND          REWIND LIBRARY
          BX6    X6-X6
          SA6    FET1+5            CLEAR COUNTS
          SA6    FET2+5
          SA6    INIEXPA           CLEAR INDEX LOADED FLAG
 STRCMD1  SX6    OLINE
          RJ     =XMSREAD
          ZR     X6,STRCMD3        IF EOI
          SA1    OLINE
          SX2    FET1
          ZR     X1,STRCMD2
          SX2    FET2
 STRCMD2  SA1    X2
          ZR     X1,STRCMD1        IF BLANK FILE NAME
          WRITEW X2,OLINE,6
          SA1    X2+5              COUNT POSITIONS
          SX6    B1
          IX6    X6+X1
          SA6    A1
          EQ     STRCMD1

 STRCMD3  WRITER FET1,RECALL
          WRITER FET2,RECALL
          SA2    FET1
          ZR     X2,STRCMD4        IF NO LEARN POSITIONS
          SA1    FET1+5
          ZR     X1,STRCMD4        IF NO LEARN POSITIONS
          MX6    42
          BX6    X6*X2             EXTRACT FILE NAME
          LX6    12
          MX3    48
          SA2    STRCMDD+3
          BX2    X3*X2
          BX7    -X3*X6
          BX7    X7+X2
          SA7    A2
          BX6    X3*X6
          SA6    A7+B1
          RJ     =XRDRCDD
          SA6    STRCMDD
          SX5    STRCMDD
          RJ     =XWRLINE
 STRCMD4  SA2    FET2
          ZR     X2,STRCMD5        IF NO SETUP POSITIONS
          SA1    FET2+5
          ZR     X2,STRCMD5        IF NO SETUP POSITIONS
          MX6    42
          BX6    X2*X6             EXTRACT FILE NAME
          LX6    12
          MX3    48
          SA2    STRCMDE+3
          BX2    X3*X2
          BX7    -X3*X6
          BX7    X7+X2
          SA7    A2
          BX6    X3*X6
          SA6    A7+B1
          RJ     =XRDRCDD
          SA6    STRCMDE
          SX5    STRCMDE
          RJ     =XWRLINE
 STRCMD5  RJ     =XREREAD          RETURN

 STRCMDA  VFD    6/77B,18/16B,36/0
          CON    7LCHE$$99
 STRCMDB  CON    7LDATE
 STRCMDC  CON    7LTIME
          BSSZ   13B
          VFD    6/50B,18/001111B,36/0

 STRCMDD  DATA   C*NNNNNNNNNN LEARN POSITIONS WRITTEN TO FILENAM*
 STRCMDE  DATA   C*NNNNNNNNNN SETUP POSITIONS WRITTEN TO FILENAM*
 REWCMD   SPACE  4,7
***       REWCMD - PROCESS REWIND COMMAND.
*


 REWCMD   RJ     =XRDRIOC          INITIALIZE I/O COMMAND
          REWIND A7,RECALL
          RJ     =XREREAD
 RETCMD   SPACE  4,8
***       RETCMD - PROCESS RETURN COMMAND.
*


 RETCMD   RJ     =XRDRIOC          INITIALIZE I/O COMMAND
          CLOSE  A7,RETURN,RECALL
          RJ     =XREREAD
 USECMD   SPACE  4,36
***       USECMD - PROCESS USE COMMAND.
*


 USECMD   RJ     =XRDRIOC          EXTRACT FILE NAME
          RJ     =XPROUSE          PROCESS USE
          ZR     X5,USECMD1        IF NO ERROR
          RJ     =XERRMSG
 USECMD1  RJ     =XREREAD
 LOACMD   SPACE  4,10
***       LOACMD - PROCESS LOAD COMMAND.
*

          IF     -DEF,KRONOS,1
 NUCC     IF     DEF,NUCC

 LOACMD   RJ     =XRDRIOC          INITIALIZE I/O COMMAND
          BX1    X7
          RJ     =XPROLOA          PROCESS LOAD
          RJ     =XREREAD
 DUMCMD   SPACE  4,30
***       DUMCMD - PROCESS DUMP COMMAND.
*


 DUMCMD   RJ     =XRDRIOC          INITIALIZE I/O COMMAND
          SA1    DUMCMDA           DEFAULT
          NZ     X7,DUMCMD1        IF NOT DEFAULT
          BX7    X1
 DUMCMD1  SA7    FET4
          RETURN A7,RECALL
          SA1    X2
          RJ     =XREQPFD          REQUEST PF DEVICE
          RJ     =XMSCLOS          CLOSE LIBRARY
          SA1    =XLIBRARY
          BX6    X1
          SA6    FET3
          REWIND A6,RECALL
          SB3    X2
          SB4    FET4
          SA1    B4+B1             SET IN = OUT = FIRST
          SX6    X1
          SA6    A1+B1
          SA6    A6+B1
          IF     DEF,KRONOS,2
          SX3    200B              READCW
          SKIP   1
          SX3    510B              READAL
          SX4    14B               WRITE
          RJ     =XPPC=            POINTER PUSH COPY
          WRITER B4,RECALL         FLUSH BUFFER
          SX6    0
          SA6    INIEXPA           CLEAR EXPLAIN FLAG
          RJ     =XREREAD          RETURN

 DUMCMDA  VFD    42/0LOPENING,18/3
 PROLOA   SPACE  4,30
***       PROLOA - PROCESS LIBRARY LOAD.
*
*         ENTRY  (X1) = LOAD FILE NAME.
*


 PROLOA   EQ     *+400000B
          NZ     X1,PROLOA1
          SA1    DUMCMDA           DEFAULT
 PROLOA1  BX6    X1
          SA6    FET3
          RJ     =XAIM=            ATTACH IF MISSING
          NG     X6,PROLOA2        IF NOT PRESENT
          SA6    PROLOAA           SAVE ATTACHED FLAG
          RETURN =XLIBRARY,RECALL
          SA1    X2
          BX6    X1
          SA6    FET4
          RJ     =XREQPFD          REQUEST PF DEVICE
          SB3    FET3
          SB4    FET4
          SA1    B4+B1             SET IN = OUT = FIRST
          SX6    X1
          SA6    A1+B1
          SA6    A6+B1
          SX3    10B               READ
          REWIND B3,RECALL
          IF     DEF,KRONOS,2
          SX4    204B              WRITECW
          SKIP   1
          SX4    514B              WRITAL
          RJ     =XPPC=            LOAD LIBRARY
          SX6    0
          SA6    INIEXPA           CLOBBER EXPLAIN FLAG
          WRITAL B4,RECALL         FLUSH BUFFER
          RJ     =XMSOPEN          OPEN LIBRARY
          SA1    PROLOAA
          NZ     X1,PROLOA3        IF NOT AUTO-ATTACHED
          RETURN FET3,RECALL
          EQ     PROLOA3

 PROLOA2  SX5    =C+ LOAD FILE MISSING.+
          RJ     =XERRMSG
 PROLOA3  EQ     PROLOA            RETURN

 PROLOAA  CON    0                 AUTO-ATTACH FLAG
 REQPFD   SPACE  4,15
**        REQPFD - REQUEST PF DEVICE.
*
*         ENTRY  (X1) = FILE NAME.
*


 REQPFD   EQ     *+400000B
 KRONOS   IF     -DEF,KRONOS
          MX6    42
          BX6    X1*X6
          SA6    REQPFDA           FORM REQUEST
          SYSTEM REQ,RECALL,A6     REQUEST PF DEVICE
 KRONOS   ENDIF
          EQ     REQPFD            RETURN

 REQPFDA  CON    0                 FILE NAME
          CON    1S28+1S31         *PF
 RDRSDS   SPACE  4,20
***       RDRSDS - SKIP DOLLAR SIGN.
*
*         ENTRY  (A1) = LINE.
*                (X1) = ((A1)).
*                (B2) = BIT OFFSET.
*
*         EXIT   (X6) = 0, IF DOLLAR SIGN.
*                (A1, X1, B2) ADVANCED PAST ALL SEPARATORS.
*
*         USES   X2.
*


 NUCC     IF     DEF,NUCC
 RDRSDS   EQ     *+400000B
          AX6    X1,B2
          BX6    -X0*X6
          SX6    X6-1R$
          NZ     X6,RDRSDS         IF NOT $
 RDRSDS1  SB2    B2-6
          PL     B2,RDRSDS2        IF NOT NEW WORD
```

```
          SA1    A1+1              ADVANCE
          SB2    54
 RDRSDS2  AX2    X1,B2
          BX2    -X0*X2            EXTRACT NEXT CHARACTER
          ZR     X2,RDRSDS         IF END OF LINE
          SX2    X2-1R+
          PL     X2,RDRSDS1        SKIP PAST SEPARATORS
          EQ     RDRSDS            RETURN
 NUCC     ENDIF
 MSOPEN   SPACE  4,40
**        MSOPEN - OPEN LIBRARY FILE.
*
*         EXIT   (MSEOI) = EOI POINTER.
*                (EC.MSI) = INDEX.
*


 MSOPEN6  SA1    LIBRARY+1         FIRST
          BX6    X6-X6
          SA6    X1                CLEAR FIRST WORD OF BUFFER

 MSOPEN   EQ     *+400000B
 MSOPEN1  OPEN   =XLIBRARY,ALTER,RECALL
          SA1    X2+B1
          MX6    1
          LX6    47-59
          BX6    X1+X6             SET RANDOM BIT
          SA6    A1
          REWIND X2,RECALL
          READ   X2,RECALL
          SA1    =XLIBRARY+2       IN
          SA3    A1+B1
          IX1    X3-X1
          NZ     X1,MSOPEN2        IF SOMETHING THERE
          REWIND LIBRARY,RECALL
          FILLW  X2,L.MSI+1
          WRITER X2,RECALL
          SX0    EC.MSI
          SB3    L.MSI
          SX4    X0+B3
          SA0    BUF1
          BX1    X1-X1             CLEAR INDEX
          RJ     =XSTE             PRESET ECS
          SX6    L.MSI/100B+2
          SA6    =XMSEOI           SET EOI POINTER
          EQ     MSOPEN6           RETURN

 MSOPEN2  SA1    MEMORY+1
          SX0    EC.MSI
          SB6    L.MSI
          ZR     X1,MSOPEN3        IF FAKE ECS
          RJ     =XRLW=            READ INDEX TO ECS
          EQ     MSOPEN4

 MSOPEN3  SA1    A1+B1             FWA FAKE ECS
          SX1    X1+EC.MSI+L.MSI+1
          RJ     =XALOCFE          ALLOCATE FAKE ECS
          SA1    MEMORY+2
          READW  =XLIBRARY,X1+EC.MSI,L.MSI  READ INDEX TO FAKE ECS

 MSOPEN4  NZ     X1,MSOPEN5
          READW  X2,=XMSEOI,1
          NZ     X1,MSOPEN5        IF BAD INDEX
          READW  X2,BUF1,1
          NZ     X1,MSOPEN6        IF NORMAL EOR
 MSOPEN5  RETURN X2,RECALL          GET RID OF IT
          SX5    =C+ INVALID LIBRARY RETURNED.+
          RJ     =XERRMSG
          SA1    =XLIBRARY
          RJ     =XREQPFD          REQUEST PF DEVICE
          EQ     MSOPEN1           MAKE A NEW LIBRARY
 PROUSE   SPACE  4,82
**        PROUSE - PROCESS USE.
*
*         ENTRY  (FET1) = FILENAME.
*
*         EXIT   (X5) = ADDRESS OF ERROR MESSAGE.
*                     = 0, IF NO ERRORS.
*


 PROUSE4  SA1    PROUSEA
          BX6    X1
          SA6    MSMASK
          SA1    PROUSEB
          RJ     =XRDRCDD
          SA6    PROUSEC
          SX5    PROUSEC
          RJ     =XWRLINE          *NNNNNNNNNN POSITIONS IGNORED.*
          SA1    PROUSEB+1
          RJ     =XRDRCDD
          SA6    PROUSED
          SX5    PROUSED
          RJ     =XWRLINE          *NNNNNNNNNN POSITIONS REPLACED.*
          SA1    PROUSEB+2
          RJ     =XRDRCDD
          SA6    PROUSEE
          SX5    PROUSEE
          RJ     =XWRLINE          *NNNNNNNNNN POSITIONS ADDED.*
          RJ     =XMSCLOS
          SX5    0                 RETURN NO ERROR

 PROUSE   EQ     *+400000B
          SX5    =C* FORMAT ERROR IN LIBRARY RECORD.* ERROR MESSAGE
          READ   FET1
 PROUSE1  READW  X2,ILINE,1
          NZ     X1,PROUSE         IF EOR, THEN ERROR
          SA1    ILINE
          MX3    6
          BX4    X3*X1
          BX6    X4-X3
          LX1    -36
          NZ     X6,PROUSE2        IF NOT 77 TABLE
          READW  X2,ILINE,X1       SKIP 77 TABLE
          NZ     X1,PROUSE         IF EOR, THEN ERROR
          EQ     PROUSE1           CHECK NEXT TABLE

 PROUSE2  LX4    6
          SX4    X4-50B
          NZ     X4,PROUSE         IF BAD HEADER
          SX1    X1-1111B
          NZ     X1,PROUSE         IF BAD HEADER
          SA0    ILINE+1
          SA1    MSMASK            SAVE MSMASK
          BX6    X1
          BX7    -X7+X7            REPLACE EVERYTHING
          SA7    A1
          SA6    PROUSEA
          BX6    X6-X6             CLEAR COUNTS
          SA6    PROUSEB
          SA6    A6+B1
          SA6    A6+B1
 PROUSE3  READW  FET1,A0-B1,6
          NZ     X1,PROUSE4        IF DONE, RESTORE MSMASK, PRINT COUNTS
          SA2    A0-B1
          BX7    X2
          RJ     =XMSREPL          REPLACE RECORD
          SA0    ILINE+1
          SA2    PROUSEB+1+X6      ADVANCE PROPER COUNT
          SX6    B1
          IX6    X6+X2
          SA6    A2
          EQ     PROUSE3

 PROUSEA  BSS    1                 SAVED MSMASK

 PROUSEB  BSS    1                 IGNORED COUNT
          BSS    1                 REPLACED COUNT
          BSS    1                 ADDED COUNT

 PROUSEC  DATA   C*NNNNNNNNNN POSITIONS IGNORED.*
 PROUSED  DATA   C*NNNNNNNNNN POSITIONS REPLACED.*
 PROUSEE  DATA   C*NNNNNNNNNN POSITIONS ADDED.*
 INIMSI   SPACE  4,10
***       INIMSI - INITIALIZE MASS STORAGE INDEX.
*


 INIMSI   RJ     =XMSOPEN          OPEN LIBRARY
          IF     DEF,NUCC,1
          SETWFL 0,E               DUMPS ON ERRORS
*         EQ     STDUSE
 STDUSE   SPACE  4,18
***       STDUSE - INITIALIZE LIBRARY FROM CHESS99.
*
*


 STDUSE   SA1    STDUSEB           *CHESS99*
          SA2    64B               NUMBER OF PARAMETERS
          SX2    X2-3
          SA4    =XMSEOI
          SX4    X4-L.MSI/100B-2
          NZ     X4,STDUSE3        IF LIBRARY COMPLETE
          SX5    STDUSEC

          IF     -DEF,KRONOS,1
 NUCC     IF     DEF,NUCC
          NG     X2,STDUSE1        IF AUTOMATIC LIBRARY
          SA4    4
          SA2    =1L*

          IF     DEF,KRONOS,2
          BX6    X6-X6
          SA6    A4

          BX3    X4-X2
          ZR     X3,STDUSE2        IF SYSTEM OPENING LOAD
          MX6    -2
          BX6    -X6+X4
          ZR     X4,STDUSE1        IF NO THIRD PARAMETER
          SA6    STDUSEA           CLOBBER DEFAULT
 STDUSE1  RJ     =XWRLINE
          WRITE  =XOUTPUT
          SA1    STDUSEA           FILE NAME
          RJ     =XPROLOA          PROCESS LOAD
          EQ     STDUSE3           RESUME

 STDUSE2  BSS    0
 NUCC     ELSE   1
          PL     X2,STDUSE3        IF NO AUTOMATIC LIBRARY

          BX6    X1
          SA6    FET1              SET FILE NAME
          REWIND A6
          RJ     =XWRLINE
          WRITE  =XOUTPUT
          RJ     =XPROUSE          USE CHESS99
 STDUSE3  JUMPUP INITAL            INITIALIZE FOR NEW GAME

          IF     -DEF,KRONOS,1
          IF     DEF,NUCC,1
 STDUSEA  VFD    42/0LOPENING,18/3 OPENING FILE NAME
 STDUSEB  VFD    42/0LCHESS99,18/3
 STDUSEC  DATA   C* OPENINGS LIBRARY INITIALIZATION.*
 ALTCMD   SPACE  4,18
***       ALTCMD - PROCESS ALTER COMMAND.
*


 ALTCMD   RJ     =XRDRIOC          INITIALIZE I/O COMMAND
          NZ     X7,ALTCMD1        IF FILE SPECIFIED
          SA4    ALTFN             USE DEFAULT
          BX7    X4
 ALTCMD1  SA2    =XINPUT
          SA3    INPFN
          BX6    X2
          NZ     X3,ALTCMD2        IF NOT ON PRIMARY FILE
          SA6    A3                SAVE PRIMARY FILE NAME
 ALTCMD2  SA7    A2                SET NEW INPUT FILE NAME
          REWIND A2
          READ   X2
          RJ     =XREREAD

 AFNCMD   SPACE  4,30
***       AFNCMD - PROCESS ALTER FILE NAME COMMAND.
*


 AFNCMD   RJ     =XRDRIOC          INITIALIZE I/O COMMAND
          ZR     X7,AFNCMD1        IF NO FILE NAME
          SA7    ALTFN             SAVE FILE NAME
          RJ     =XREREAD

 AFNCMD1  SX5    =C* ALTER FILE NAME MISSING.*
          RJ     =XERRMSG
          RJ     =XREREAD
 HDCCMD   SPACE  4,25
***       HDCCMD - PROCESS HARD COPY COMMAND.
*


 HDCCMD   RJ     =XRDRIOC
          RJ     =XPROHDC          PROCESS HARDCOPY
```

```
          ZR     X5,HDCCMD1        IF NO ERRORS
          RJ     =XERRMSG
 HDCCMD1  RJ     =XREREAD
 PROHDC   SPACE  4,27
**        PROHDC - PROCESS HARDCOPY.
*
*         ENTRY  (X7) = HARDCOPY FILE NAME.
*                (X7) = 0, IF DEFAULT SHOULD BE USED.
*
*         EXIT   (X5) = ADDRESS OF ERROR MESSAGE.
*                (X5) = 0, IF NO ERRORS.
*



 PROHDC   EQ     *+400000B
          SX5    =C* HARD COPY COMMAND IGNORED.*
          NZ     X7,PROHDC1        IF NAME SPECIFIED
          SA2    PROHDCA           ELSE USE *HARDCPY*
          BX7    X2
 PROHDC1  SA2    =XHARDCPY
          NZ     X2,PROHDC         IF ALREADY HAVE HARDCOPY
          BX5    X5-X5             CLEAR ERROR MESSAGE
          SA7    A2
          SX6    2RPR+110000B&3
          BX6    X6-X7
          SA6    OLINE+1
          BX7    X7-X7
          SA7    A6-B1
 KRONOS   IF     -DEF,KRONOS,1
          SYSTEM DSP,RECALL,A7     DISPOSE(HARDCPY,*PR)
          OPEN   A2,ALTERNR,RECALL
          EQ     PROHDC            RETURN

 PROHDCA  VFD    42/0LHARDCPY,18/3
 SPECMD   SPACE  4,21
***       SPECMD - PROCESS SPEED COMMAND.
*
*         SPEED,N SETS TIMING PARAMETERS FOR A FAST GAME
*         AT A RATE OF N CPU SECONDS PER MOVE.


 SPECMD   RJ     =XPRORELEL          PROCESS RELIABILITY
          ZR     X5,SPECMD1        IF NO ERRORS
          RJ     =XERRMSG
 SPECMD1  BX7    X7-X7
          RJ     =XPROHDC          PROCESS HARDCOPY
          ZR     X5,SPECMD2        IF NO ERRORS
          RJ     =XERRMSG
 SPECMD2  SA1    SWICH
          SX6    M.SUM+M.REC
          SX7    B1
          LX7    S.ADI
          BX6    X1+X6
          BX6    X6+X7
          LX7    S.TIM-S.ADI
          BX6    X6+X7
          LX7    S.ECH-S.TIM
          BX6    -X7*X6            SWITCH ECHO OFF
          SA6    A1
          SX6    2
          SA6    COPIES
          SX6    15                INCREASE SIZE OF TRANSPOSITIONS TABLE
          SA6    LOGTRA
          JUMPUP BLICMD
 TOUCMD   SPACE  4,21
***       TOUCMD - PROCESS TOURNEMENT COMMAND.
*



 TOUCMD   RJ     =XPRORELEL          PROCESS RELIABILITY
          ZR     X5,TOUCMD1        IF NO ERRORS
          RJ     =XERRMSG
 TOUCMD1  BX7    X7-X7
          RJ     =XPROHDC          PROCESS HARDCOPY
          ZR     X5,TOUCMD2        IF NO ERRORS
          RJ     =XERRMSG
 TOUCMD2  SA1    SWICH
          SX6    M.NDC+M.SUM+M.PRV+M.REC
          SX7    B1
          LX7    S.ADI
          BX6    X1+X6
          BX6    X6+X7
          LX7    S.TIM-S.ADI
          BX6    X6+X7
          LX7    S.ECH-S.TIM
          BX6    -X7*X6            SWITCH ECHO OFF
          LX7    S.PON-S.ECH
          BX6    X6+X7             TURN ON THINKING ON OPP. TIME
          SA6    A1
          SX6    10
          SA6    COPIES
          SX6    15                INCREASE SIZE OF TRANSPOSITIONS TABLE
          SA6    LOGTRA
          RJ     =XREREAD
 RDRVFN   SPACE  4,32
**        RDRVFN - VERIFY FILE NAME.
*
*         ENTRY  (X7) = PROSPECTIVE FILE NAME.
*
*         EXIT   TO REREAD, IF INVALID.
*                (X7) = FILE NAME + COMPLETE BIT AND BINARY BIT.
*
*         CALLS  ERRMSG.
*



 RDRVFN   EQ     *+400000B
          ZR     X7,RDRVFN         IF NO NAME, RETURN
          MX6    42
          BX1    X6*X7
          MX5    -2
          RJ     =XVFN
          BX7    -X5+X1
          SA1    A1                RESTORE X1
          ZR     X6,RDRVFN         IF VALID
          SX5    =C* INVALID FILE NAME.*
          RJ     =XERRMSG
          RJ     =XREREAD
 RDRIOC   SPACE  4,21
**        RDRIOC - INITIALIZE I/O COMMAND.
*
*         ENTRY  (ILINE) = I/O COMMAND.
*
*         EXIT   ((A7)) = (FET1) = FILENAME.
*
*         CALLS  RDRGNT, RDRVFN.
*
```

```
 RDRIOC   EQ     *+400000B
          SA1    =XCHESS40+3
          BX6    X1
          SA6    A1-B1             SET CHESS40 BUFFER EMPTY
          SA1    ILINE
          SB2    54
          MX0    54
          RJ     =XRDRGNT          SKIP FIRST TOKEN
          RJ     =XRDRGNT          GET NEXT TOKEN
          RJ     =XRDRVFN          VERIFY FILE NAME
          SA7    FET1              INITIALIZE FET
          EQ     RDRIOC            RETURN
 BUFFERS  SPACE  4
 LBUF     EQU    2000B             BUFFER SIZE
 FET1     FILEB  BUF1,LBUF,FET=6
 FET2     FILEB  BUF2,LBUF,FET=6
          IF     -DEF,KRONOS,1
          IF     DEF,NUCC,2
 FET3     RFILEB BUF1,LBUF*2,FET=6
 FET4     RFILEB BUF1,LBUF*2,FET=6
 CHESS40  RFILEB BUF1,LBUF,FET=8

 BUF1     BSS    LBUF
 BUF2     BSS    LBUF

 RELCMD   SPACE  4,30
***       RELCMD - PROCESS RELIABILITY COMMAND.
*



 RELCMD   RJ     =XPRORELEL          PROCESS RELIABILITY
          ZR     X5,RELCMD1        IF NO ERROR
          RJ     =XERRMSG
 RELCMD1  RJ     =XREREAD
 PRORELEL SPACE  4,34
**        PRORELEL - PROCESS RELIABILITY COMMAND.
*
*         EXIT   (X5) = ADDRESS OF ERROR MESSAGE.
*                     = 0, IF NO ERROR.
*



 PRORELEL EQ     *+400000B
          IF     DEF,OVERLAY,2
          SA1    RELSW
          ZR     X1,PRORELEL1      IF NOT IN RELIABILITY MODE
          SX5    =C* RELIABILITY COMMAND IGNORED.*
          EQ     PRORELEL          RETURN ERROR

 OVERLAY  IF     DEF,OVERLAY
 PRORELEL1 SX6   B1
          SA6    A1                SET RELIABILITY
          CLOSE  RELFILE,UNLOAD,RECALL
 KRONOS   IF     -DEF,KRONOS,1
          SYSTEM REQ,RECALL,PRORELA
 KRONOS   IF     DEF,KRONOS
          SX2    RELFILE           PURGE DISK FILE
          RJ     PDF
          DEFINE RELFILE
 KRONOS   ENDIF
          OPEN   X2,ALTER,RECALL
          SX6    CHE$$30
          SX7    X6+B1
          SA6    RELFILE+2         SET IN
          SA7    RELFILE+4         SET LIMIT
          WRITER X2,RECALL
          CLOSE  =XLIBRARY,,RECALL
 KRONOS   IF     -DEF,KRONOS
          SA0    RELFDB
          RJ     =XCATLOG
          SA0    LIBFDB
          RJ     =XCATLOG
          SA0    GAMFDB
          RJ     =XCATLOG
 KRONOS   ELSE
          SX2    =XLIBRARY         DEFINE LIBRARY FILE
          RJ     DLF
          SX2    =XGAMFILE         PURGE DISK FILE
          RJ     PDF
          DEFINE =XGAMFILE
 KRONOS   ENDIF
          SA1    =XGAMFILE+3
          BX6    X1
          BX5    X5-X5             CLEAR ERROR
          SA6    GAMOUT
          EQ     PRORELEL          RETURN
 OVERLAY  ENDIF

 PRORELA  VFD    42/0LRELFILE,18/0
          CON    1S28+1S31
          CON    0
 KRONOS   IF     DEF,KRONOS
 DLF      SPACE  4
**        DLF - DEFINE LIBRARY FILE.


 DLF      PS                       ENTRY/EXIT
          SA1    X2+B1             SET USER ERROR PROCESSING
          SX6    B1
          LX6    44
          BX6    X6+X1
          SA6    A1
          DEFINE X2
          SA1    X2
          SX6    X1
          AX6    9
          SX6    X6-12B
          NZ     X6,DLF1           IF NOT DUPLICATE
          PURGE  X2
          DEFINE X2
 DLF1     SA1    X2+B1             CLEAR USER ERROR PROCESSING
          MX6    -1
          LX6    44
          BX6    X6*X1
          SA6    A1
          JP     DLF               RETURN
 PDF      SPACE  4
**        PDF - PURGE DISK FILE.
*         ENTRY  (X2) = FET ADDRESS.
*         EXIT   (X2) = FET ADDRESS.


 PDF      PS                       ENTRY/EXIT
          SA1    X2+B1             SET USER ERROR PROCESSING
          SX6    B1
```

```
         LX6    44
         BX6    X6+X1
         SA6    A1
         PURGE  X2
         SA1    X2+B1       CLEAR USER ERROR PROCESSING
         MX6    -1
         LX6    44
         BX6    X6*X1
         SA6    A1
         JP     PDF         RETURN
 KRONOS  ENDIF
 REICMD  SPACE  4,30
***      REICMD - PROCESS REINCARNATE COMMAND.
*


 REICMD  BSS    0
 OVERLAY IF     DEF,OVERLAY
         WRITER =XOUTPUT,RECALL
         CLOSE  =XLIBRARY,UNLOAD,RECALL
         CLOSE  =XGAMFILE,UNLOAD,RECALL
         IF     DEF,KRONOS,2
         RETURN REICMDB,R
         ATTACH X2
         REWIND REICMDB,RECALL
         READSK X2,RECALL
         SA1    X2
         BX6    X1
         SA6    =XRELFILE         SET RELFILE COMPLETE
         BX6    X6-X6
         SA6    =XLOADRA          CLOBBER LAST OVERLAY LOADED
 KRONOS  IF     -DEF,KRONOS
         SA0    LIBFDB
         RJ     =XATTACH
         SA0    GAMFDB
         RJ     =XATTACH
 KRONOS  ELSE
         ATTACH =XLIBRARY
         ATTACH =XGAMFILE
 KRONOS  ENDIF
         SA1    REICMDA           GAMFILE
         BX6    X1
         SA6    FET1
         SKIPEI A6,RECALL
         WRITE  =XOUTPUT
         SA1    =XINPUT+2         IN
         BX6    X1
         SA6    A1+B1             SET INPUT BUFFER EMPTY
         SA1    MEMORY            GET THE CM THAT WE THINK WE HAVE
         AX1    30
         RJ     =XMEM.
         SA1    MEMORY+1          GET THE ECS THAT WE THINK WE HAVE
         AX1    30
         LX1    30
         BX6    X1
         SA6    A1
         ZR     X1,REI1           IF NO ECS REQUIRED
         SYSTEM MEM,R,A6,1
 REI1    BSS    0
 OVERLAY ENDIF
         IF     -DEF,OVERLAY,2
         SX5    =C* REINCARNATE COMMAND IGNORED.*
         RJ     =XERRMSG
         RJ     =XREREAD
 REICMDA VFD    42/0LGAMFILE,18/1
 REICMDB FILEB  ORGR-1,CHE$$30+1-ORGR+1,FET=6
         ORG    REICMDB
         VFD    42/0LRELFILE,18/3
         ORG    REICMDB+6
 HELCMD  SPACE  4,8
***      HELCMD - PROCESS HELP COMMAND.
*


 HELCMD  RJ     =XINIEXP          INITIALIZE EXPLAIN
         RJ     =XPROHEL          PROCESS HELP
         EQ     EXPEXT            EXPLAIN EXIT
 EXPCMD  SPACE  4,18
***      EXPCMD - PROCESS EXPLAIN COMMAND.
*


 EXPCMD  RJ     =XINIEXP          INITIALIZE EXPLAIN
         RJ     =XRDRSFT          SKIP FIRST TOKEN
         RJ     =XRDRGNT          GET NEXT TOKEN
         ZR     X7,EXPCMD1        IF NO PARAMETER
         SA2    EXPCMDA
         RJ     =XRDRSKT          SEARCH KEYWORD TABLE
                                  (RETURNS ONLY IF NOT FOUND)
         SX3    2REX
         RJ     =XPROEXP          PROCESS EXPLAIN
         EQ     EXPEXT            EXPLAIN EXIT

 EXPCMD1 RJ     =XPROHEL          PROCESS HELP
         EQ     EXPEXT            EXPLAIN EXIT

 EXPCMDA CMND   ALL,EXPALL,PON
         CMND   SW,EXPSWI,PON
         CON    0
 EXPALL  SPACE  4,12
**       EXPALL - EXPLAIN ALL.
*


 EXPALL  RJ     =XPROHEL          PROCESS HELP
         SX3    2REX
         RJ     =XPREXAL          PROCESS EXPLAIN ALL
         SX3    2RSW
         RJ     =XPREXAL          PROCESS EXPLAIN ALL
         EQ     EXPEXT            EXPLAIN EXIT
 EXPSWI  SPACE  4,16
**       EXPSWI - EXPLAIN SWITCH.
*


 EXPSWI  RJ     =XRDRGNT          GET NEXT TOKEN
         NZ     X7,EXPSWI1        IF PRESENT
         SX7    2RSW
         SX3    2REX
         LX7    48
         RJ     =XPROEXP          PROCESS EXPLAIN
         EQ     EXPEXT            EXPLAIN EXIT

 EXPSWI1 SX3    2RSW
         RJ     =XPROEXP          PROCESS EXPLAIN
*        EQ     EXPEXT            EXPLAIN EXIT
 EXPEXT  SPACE  4,6
**       EXPEXT - EXPLAIN EXIT.
*


 EXPEXT  RECALL =XCHESS40
         RJ     =XREREAD
 PROEXP  SPACE  4,88
**       PROEXP - PROCESS EXPLAIN.
*
*        ENTRY  (X3) = TABLE NAME.
*               (X7) = ENTRY DESIRED.
*               (BUF2) = INDEX.
*


 PROEXP  EQ     *+400000B
         LX3    48
         MX6    12
         SA2    BUF2
 PROEXP1 BX4    X2-X3
         BX4    X6*X4
         ZR     X4,PROEXP2        IF FOUND
         SA2    A2+B1
         NZ     X2,PROEXP1        IF NOT END OF TABLE
         SX5    =C* SYSTEM ERROR - TUTOR FILE BAD.*
         RJ     =XERRMSG
         EQ     PROEXP

 PROEXP2 SA2    BUF2+X2-102B
         SA0    A2
         RJ     =XRDRSXT          SEARCH TRANSLATION TABLE
         ZR     X6,PROEXP3        IF NOT FOUND
         RJ     =XPRIEXP          PRINT BLOCK
         EQ     PROEXP            RETURN

 PROEXP3 SA2    A0
         SX6    B1
         IX3    X7-X6
         BX6    -X3+X7
         SA3    =4040404040404040404040B
         BX3    X6*X3
         BX6    X3
         LX6    -5
         IX6    X3-X6
         BX6    X6+X3             MASK OVER SPECIFIED CHARACTERS
         MX4    24
         BX5    X4*X6
         SB2    OLINE             FWA
         SB3    OLINE+7           LWA+1
         SB4    B0                UPPER/LOWER SWITCH
         SB5    B2
 PROEXP4 BX3    X2-X7
         BX3    X5*X3
         NZ     X3,PROEXP6        IF NO MATCH
         BX6    X4*X2
         NZ     B4,PROEXP5        IF LOWER HALF
         SA6    B2
         SB4    B1
         EQ     PROEXP6

 PROEXP5 LX6    30
         SA3    B2
         BX6    X3+X6
         SB4    B0
         SA6    A3
         SB2    B2+B1
         GE     B2,B3,PROEXP8     IF TOO MANY
 PROEXP6 SA2    A2+B1
         NZ     X2,PROEXP4        IF MORE IN TABLE
         NE     B2,B5,PROEXP9     IF MORE THAN 1 MATCH
         ZR     B4,PROEXP7        IF NO MATCHES
         SA3    B2
         BX7    X3
         SA2    A0
         RJ     =XRDRSXT          SEARCH TRANSLATION TABLE AGAIN
         RJ     =XPRIEXP          PRINT BLOCK
         EQ     PROEXP            RETURN

 PROEXP7 RJ     =XPROHEL
         EQ     PROEXP

 PROEXP8 SA2    =5R....
         BX6    X4*X6
         BX6    X6+X2             CLOBBER LAST PROMPT WITH ....
         SA6    A6
 PROEXP9 BX6    X6-X6
         SA6    B2+B4             STORE TERMINATOR
         SA1    OLINE
 PROEXP10 RJ    =XSFA             SPACE FILL ANYTHING
         SA6    A1
         SA1    A1+B1
         NZ     X1,PROEXP10       LOOP TO END OF LINE
         SX5    =C* AMBIGUOUS EXPLAIN REQUEST.  CHOOSE ONE OF*
         RJ     =XERRMSG
         SX5    OLINE
         RJ     =XWRLINE
         EQ     PROEXP
 PROHEL  SPACE  4,11
**       PROHEL - PROCESS HELP.
*


 PROHEL  EQ     *+400000B
         SX7    2RHE
         SA2    BUF2
         LX7    48
         RJ     =XRDRSXT
         RJ     =XPRIEXP
         EQ     PROHEL
 PREXAL  SPACE  4,32
**       PREXAL - PROCESS EXPLAIN ALL.
*
*        ENTRY  (X3) = SUB-TABLE NAME.
*


 PREXAL  EQ     *+400000B
         SA2    BUF2
         LX3    48
         MX4    12
 PREXAL1 BX6    X2-X3
         BX6    X4*X6
         ZR     X6,PREXAL2        IF FOUND
         SA2    A2+B1
         NZ     X2,PREXAL1
         BX6    X6-X6
         RJ     =XPRIEXP          ISSUE ERROR MESSAGE
```

```
          EQ     PREXAL

 PREXAL2  SA0    X2+BUF2-102B

 PREXAL3  SX5    =1L
          RJ     =XWRLINE
          SA2    A0+
          SX6    X2
          SA0    A0+B1             ADVANCE INDEX
          NG     X2,PREXAL         IF DONE
          RJ     =XPRIEXP          PRINT BLOCK
          EQ     PREXAL3           LOOP
 PRIEXP   SPACE  4,30
**        PRIEXP - PRINT EXPLAIN BLOCK.
*
*         ENTRY  (X6) = DISK WORD ADDRESS OF FIRST WORD.
*                (X6) = 0, IF ERROR.
*                ((A2)+1) = DISK LWA+1.
*


 PRIEXP2  SX5    =C* SYSTEM ERROR - BAD TUTOR FILE.*
          RJ     =XERRMSG

 PRIEXP   EQ     *+400000B
          ZR     X6,PRIEXP2
          SA2    A2+B1
          SX0    X2
          SX3    X6
 PRIEXP1  SX2    =XCHESS40
          SX1    =XRDC=
          SB6    OLINE
          SB7    7
          SX5    B6
          RJ     =XRWA=
          NZ     X1,PRIEXP2        IF EOX
          RJ     =XWRLINE
          SA3    =XCHESS40+7
          IX6    X3-X0
          NG     X6,PRIEXP1        IF NOT DONE
          EQ     PRIEXP            RETURN
 INIEXP   SPACE  4,32
**        INIEXP - INITIALIZE EXPLAIN COMMAND.
*


 INIEXP   EQ     *+400000B
          SA3    INIEXPA
          NZ     X3,INIEXP         IF ALREADY INITIALIZED
          SX6    100B
          SA6    =XCHESS40+7
          REWIND =XCHESS40,RECALL
          SX1    =XRDW=
          SX3    101B
          SB6    A3
          SB7    B1
          RJ     =XRWA=            READ LENGTH WORD
          NZ     X1,INIEXP1        IF ERROR
          SX1    =XRDW=
          SX3    102B
          SA4    B6-B1
          SB7    X4                LENGTH
          SB6    BUF2
          SB5    LBUF
          GT     B7,B5,INIEXP1     IF BUFFER NOT BIG ENOUGH
          RJ     =XRWA=            READ WA FILE
          ZR     X1,INIEXP         IF NO ERROR
 INIEXP1  SX5    =C* SYSTEM ERROR - TUTOR INDEX BAD.*
          RJ     =XERRMSG
          RJ     =XREREAD

 INIEXPA  CON    0                 INDEX LOADED FLAG
 RDRSFT   SPACE  4,24
***       RDRSFT - SKIP FIRST TOKEN.
*
*         ENTRY  (ILINE) = INPUT LINE.
*
*         EXIT   (A1) = ADDRESS OF INPUT WORD.
*                (X1) = ((A1))
*                (B2) = NEXT CHARACTER SHIFT COUNT.
*                (X0) = 7777 7777 7777 7777 7700
*
*         USES   X6, X7, B4.
*         CALLS  RDRGNT.
*


          ENTRY  RDRSFT
 RDRSFT   EQ     *+400000B
          SA1    ILINE
          SB2    54
          MX0    54
          RJ     =XRDRGNT          GET NEXT TOKEN
          EQ     RDRSFT            RETURN
 RDRSXT   SPACE  4,88
***       RDRSXT - SEARCH TRANSLATION TABLE.
*
*         ENTRY  (A2) = FWA OF TABLE.
*                (X2) = ((A2))
*                (X7) = KEY.
*                TABLE FORMAT:
*                36/KEY
*                6/LENGTH OF KEY-1
*                18/VALUE
*         TERMINATED BY ZERO WORD.
*
*         EXIT   (X6) = TRANSLATION.
*                (X6) = 0, IF NOT FOUND.
*
*         USES   A2, X2, X3, X4, X5, B4.
*


          ENTRY  RDRSXT
 RDRSXT   EQ     *+400000B
          MX4    1
          MX5    36
 RDRSXT1  SX6    X2
          ZR     X2,RDRSXT         IF END OF TABLE
          BX3    -X5*X2
          AX3    18
          SB4    X3                MASK SIZE
          BX3    X2-X7             COMPARE
          AX2    X4,B4             BUILD MASK
          BX2    X2*X3
          ZR     X2,RDRSXT         IF SAME
          SA2    A2+B1
          EQ     RDRSXT1           LOOP
 CATLOG   SPACE  4,16
 KRONOS   IF     -DEF,KRONOS
***       CATLOG - CATALOG PERMANENT FILE.
*
*         ENTRY  ((A0)) = FDB FOR FILE.
*
*         USES   ALL REGISTERS, OLINE.
*         CALLS  PFM=, PFMERR.
*


 CATLOG   EQ     *+400000B
          SX6    3RPFC
          SX7    20B
          SX2    A0
          RJ     =XPFM=
          RJ     =XPFMERR
          EQ     CATLOG
 ATTACH   SPACE  4,16
***       ATTACH - ATTACH PERMANENT FILE.
*
*         ENTRY  ((A0)) = FDB FOR FILE.
*
*         USES   ALL REGISTERS, OLINE.
*         CALLS  PFM=, PFMERR.
*


 ATTACH   EQ     *+400000B
          SX6    3RPFA
          SX7    10B
          SX2    A0
          RJ     =XPFM=
          RJ     =XPFMERR
          EQ     ATTACH
 PFMERR   SPACE  4,75
***       PFMERR - OUTPUT LFN, PFN, AND DIAGNOSTIC.
*
*         ENTRY  ((A0)) = COMPLETED FDB FOR FILE.
*
*         CALLS  WRLINE, SFN, ERRMSG.
*         USES   ALL REGISTERS, OLINE.
*


 PFMERR   EQ     *+400000B
          SA1    A0
          MX0    42
          BX1    X0*X1
          RJ     =XRDRSFN          SPACE FILL NAME
          SX7    2R- &2R
          BX6    X6-X7
          SB3    A0                LWA+1 FROM
          SA6    OLINE
          SA1    A0-4              FWA FROM
          SX5    A6
          BX7    X7-X7
 PFMERR1  BX6    X1
          SA6    A6+B1
          SA1    A1+B1
          SB2    A1
          LT     B2,B3,PFMERR1
          SA7    A6+B1
          RJ     =XWRLINE          OUTPUT LFN - PFN
          SA1    A0
          LX1    -9
          MX0    -9
          BX1    -X0*X1
          ZR     X1,PFMERR         RETURN IF NO ERROR
          SX2    X1-PFMERRAL/2-2
          SX5    =C* PFMERR ERROR.*
          PL     X2,PFMERR2        IF ERROR OUT OF RANGE
          LX1    1
          SX5    PFMERRA-2+X1
 PFMERR2  RJ     =XERRMSG
          EQ     PFMERR            RETURN

 PFMERRA  DATA   18L ID ERROR
          DATA   18L LFN IN USE
          DATA   18L LFN NON-EXISTANT
          DATA   18L BLANK PFN
          DATA   18L PFD FULL
          DATA   18L RBTC FULL
          DATA   18L PF DEVICE DOWN
          DATA   18L NOT CLOSED
          DATA   18L NOT ON PF DEVICE
          DATA   18L UNKNOWN PFN
          DATA   18L UNKNOWN CYCLE
          DATA   18L INVALID CYCLE
          DATA   18L DUPLICATE NAME
          DATA   18L ALREADY PF
          DATA   18L LFN NOT LOCAL
          DATA   18L NOT PERMANENT
          DATA   18L LFN PURGED
          DATA   18L EMPTY FILE
          DATA   18L INCOMPLETE CYCLE
          DATA   18L DUPLICATE ATTACH
          DATA   18L FILE IN USE

 PFMERRAL EQU    *-PFMERRA
 KRONOS   ENDIF
 PIDCMD   SPACE  4,20
***       PIDCMD - PROCESS PERMANENT FILE ID COMMAND.
*


 PIDCMD   SX2    14B               ID CODE
          RJ     =XRDRIPC          INITIALIZE PF COMMAND
          SA6    RELFDB+1          INTO ALL FDBS
          SA6    LIBFDB+1
          SA6    GAMFDB+1
          RJ     =XREREAD
 PPWCMD   SPACE  4,20
***       PPWCMD - PROCESS PERMANENT FILE PW COMMAND.
*


 PPWCMD   SX2    20B               PW CODE
          RJ     =XRDRIPC          INITIALIZE PF COMMAND
          SA6    RELFDB+2          INTO ALL FDBS
          SA6    LIBFDB+2
          SA6    GAMFDB+2
          RJ     =XREREAD
 RDRIPC   SPACE  4,25
**        RDRIPC - INITIALIZE PERMANENT FILE COMMAND.
*
```

```
*         ENTRY  (ILINE) = PF COMMAND.
*                (X2) = PERMANENT FILE CODE.
*
*         EXIT   (X6) = ASSEMBLED PF PARAMETER.
*
*         CALLS  RDRGNT.
*



 RDRIPC   EQ     *+400000B
          SA1    ILINE
          SB2    54
          MX0    54
          RJ     =XRDRGNT          SKIP FIRST TOKEN
          RJ     =XRDRGNT          GET NEXT TOKEN
          SB3    60
          SB3    B3-B4
          LX7    X7,B3             RIGHT ADJUST
          BX6    X7+X2             INSERT PF CODE
          EQ     RDRIPC            RETURN
*CALL     COMCCDD
 RDRCDD   SPACE  4,23
***       RDRCDD - CONVERT DECIMAL DIGITS.
*
*         ENTRY  (X1) = NUMBER TO BE CONVERTED.
*
*         EXIT   (X6) = CONVERTED RESULT.
*
*         USES   X1, X2, X3, X4, X6, X7, A2, A3, A4, B2, B3, B4, B5.
*         CALLS  CDD.
*



          ENTRY  RDRCDD
 RDRCDD1  RJ     CDD               CONVERT POSITIVE NUMBER

 RDRCDD   EQ     *+400000B
          PL     X1,RDRCDD1        IF POSITIVE
          BX1    -X1
          RJ     CDD               CONVERT ABSOLUTE VALUE
          SX7    1R-&1R
          LX7    X7,B2
          BX6    X6-X7             APPEND SIGN
          EQ     RDRCDD
 XEQCMD   SPACE  4,60
 NUCC     IF     DEF,NUCC
***       XEQCMD - PROCESS XEQUTE COMMAND.
*



 XEQCMD   SA1    ILINE
          SB2    54
          MX5    54
          MX0    54
          RJ     =XRDRSDS          SKIP DOLLAR SIGN
          ZR     X6,XEQCMD1        IF DOLLAR SIGN
          RJ     =XRDRGNT          SKIP FIRST TOKEN
 XEQCMD1  RJ     =XRDRAOD          ASSEMBLE FL
          SA6    XEQCMDB           SAVE FL
          SB2    B2+6              BACK UP ONE CHARACTER
          SB3    60D
          NE     B2,B3,XEQCMD2     IF NOT EARLIER WORD
          SA1    A1-B1
          SB2    B0
 XEQCMD2  AX2    X1,B2
          BX7    -X0*X2            EXTRACT CHARACTER
          SX3    X7-1R0
          SB2    B2-6
          PL     B2,XEQCMD3        IF NOT NEW WORD
          SB2    54
          SA1    A1+B1
 XEQCMD3  PL     X3,XEQCMD2        IF NOT ALPHABETIC
          ZR     X7,XEQCMD7        IF END OF CARD, FORGET IT
          SB7    OLINE
 XEQCMD4  AX2    X1,B2             NEXT INPUT CHARACTER
          BX2    -X0*X2
          LX7    6
          LX5    6
          BX7    X7+X2             INSERT CHARACTER
          SB2    B2-6
          ZR     X2,XEQCMD6        IF END OF INPUT
          PL     B2,XEQCMD5        IF NOT NEW INPUT WORD
          SB2    54
          SA1    A1+B1
 XEQCMD5  NG     X5,XEQCMD4        IF NOT NEW OUTPUT WORD
          SA7    B7
          BX7    X7-X7
          SB7    B7+B1
          EQ     XEQCMD4

 XEQCMD6  LX7    6
          LX5    6
          NG     X5,XEQCMD6        IF LAST WORD NOT LEFT ADJUSTED
          SA7    B7
          BX7    X7-X7
          SA7    A7+B1             INSURE ZERO WORD
          WRITER =XHARDCPY,RECALL
          WRITER =XOUTPUT,RECALL   FLUSH THE BUFFER
          RJ     =XDUDDROP         NO DUD PROCESSING FOR A WHILE
          SA1    XEQCMDA
          RJ     =XRDRESP          EXECUTE SYSTEM PROGRAM
          SX5    170477B
          SB2    X6                ERROR CODE
          AX5    X5,B2
          CX5    X5
          LX5    1
          SX5    XEQCMDC+X5        ERROR ADDRESS
          RJ     =XWRLINE
          WRITE  =XOUTPUT,RECALL   ACTIVATE CONSOL FET
          WRITE  X2,RECALL
          SX2    7700B
          RJ     =XERRSET

 XEQCMD7  RJ     =XREREAD          RETURN

 XEQCMDA  CON    OLINE             CARD IMAGE ADDRESS
          CON    XEQCMDB           FL ADDRESS

 XEQCMDB  CON    0                 FL

 XEQCMDC  DATA   18L PROCESS TIME LIM
          DATA   18L HUNG IN AUTO-RCL
          DATA   18L HUNG IN AUTO-RCL
          DATA   18L AUTO-RCL ERROR
          DATA   18L CONTROL CARD ERR
          DATA   18L OPERATOR DROP
          DATA   18L PP CALL ERROR
          DATA   18L CPU ABORT
          DATA   18L PPU ABORT
          DATA   18L ARITHMETIC ERROR
          DATA   18L TIME LIMIT
          DATA   28L SUCCESSFUL EXECUTION

*CALL     COMCESP
 RDRESP   EQU    ESP=
*CALL     COMCCPM
*CALL     COMCJCM
 NUCC     ENDIF
          IF     -DEF,KRONOS,1
 NUCC     IF     DEF,NUCC
*CALL     COMCPPC
 NUCC     ENDIF
*CALL     COMCAIM
*CALL     COMCSET
*CALL     COMCWFW
*CALL     COMCSFN
 RDRSFN   EQU    SFN
          ENTRY  RDRSFN
*CALL     COMCCIO
*CALL     COMCSYS
*CALL     COMCRWA
*CALL     COMCSFA
*CALL     COMCVFN
 RLW      SPACE  4,88,
***               RLW TRANSFERS WORDS FROM A CIRCULAR BUFFER TO AN ECS
*         WORKING BUFFER.
*
*         ENTRY  (X2) = ADDRESS OF FET FOR FILE.
*                (X0) = FWA WORKING BUFFER.
*                (B6) = WORD COUNT OF WORKING BUFFER.
*                (B1) = 1.
*
*         EXIT   (X1) = 0, IF TRANSFER COMPLETE.
*                     = -1, IF EOF DETECTED ON FILE.
*                     = NUMBER OF WORDS NOT TRANSFERRED INTO WORKING
*                       BUFFER IF EOR DETECTED ON FILE BEFORE TRANSFER
*                       WAS COMPLETED.
*                (X2) = ADDRESS OF FET FOR FILE.
*
*         USES   ALL REGISTERS EXCEPT A5, X5.
*         CALLS  EWE=, LCB=, RDX=.
*



  +       EQ     RLW1

 RLW=     EQ     *+400000B   ENTRY/EXIT
          SA4    RLW=        SET RETURN ADDRESS
          SA1    X2+4        (B5) = LIMIT
          SA3    X2+1        (X3) = FIRST
          SB5    X1

*         INITIALIZE REGISTERS FOR TRANSFER.

 RLW1     SA1    A3+B1       (B2) = IN
          SA2    A1+B1       (B4) = OUT
          SB2    X1+
          SB4    X2

*         TRANSFER.

 RLW2     SB3    B2-B4       AMOUNT OF DATA IN BUFFER
          PL     B3,RLW3     IF NO WRAP
          SB3    B5-B4       USE ONLY DATA AT END OF BUFFER
 RLW3     SA0    B4          CM FWA = OUT
          GE     B6,B3,RLW4  IF ALL DATA TO BE TRANSFERRED
          SB3    B6          TRANSFER ONLY TO WORD COUNT
 RLW4     ZR     B3,=XLCB=   IF BUFFER EMPTY
          SB6    B6-B3       COUNT TRANSFERRED WORDS
          SB4    B4+B3       ADVANCE OUT
          WE     B3          TRANSFER BLOCK
          RJ     =XEWE=      IF ERROR
          SX6    B3
          NE     B4,B5,RLW5  IF OUT ' LIMIT
          SB4    X3          RESET TO FIRST
 RLW5     IX0    X0+X6       ADVANCE ECS ADDRESS
          NZ     B6,RLW2     IF MORE TO TRANSFER
          SB3    B7
          EQ     =XRDX=      EXIT
 STE      SPACE  4,88,
***               STE PRESETS AN AREA OF EXTENDED CORE STORAGE TO A
*         SPECIFIED VALUE.
*
*         ENTRY  (X0) = FWA ECS.
*                (X1) = VALUE.
*                (A0) = FWA OF CM BUFFER.
*                (B3) = LENGTH OF CM BUFFER.
*                (X4) = LWA+1 OF ECS.
*                (B1) = 1.
*
*         USES   A6, A7, B3, X0, X2, X3, X6, X7.
*         CALLS  EWE=, SET.
*



 STE3     WE     B3          WRITE LAST BLOCK
          RJ     =XEWE=      IF ERROR

 STE      EQ     *+400000B   ENTRY/EXIT
          SX2    A0          FWA CM BUFFER
          SX3    X2+B3       LWA+1 CM BUFFER
          RJ     =XSET       PRESET CM
          SX3    B3+
          IX3    X4-X3       LWA+1 OF COMPLETE BLOCKS
          IX6    X0-X3
          PL     X6,STE2     IF NO COMPLETE BLOCKS
 STE1     WE     B3          WRITE ONE BLOCK
          RJ     =XEWE=      IF ERROR
          SX0    X0+B3       ADVANCE ADDRESS
          IX6    X0-X3
          NG     X6,STE1     IF MORE COMPLETE BLOCKS
 STE2     BX6    -X6
          SB3    B3+X6       SIZE OF LAST BLOCK
          EQ     STE3        WRITE LAST BLOCK
          END    'END'
CHESS40
          IDENT  CHE$$40
          SST    BASE
          LIST   F
 CHESS40  TITLE  CHESS40 - TUTOR TEXT.
 OVERLAY  IF     DEF,OVERLAY
          ENTRY  CHE$$40
 40       THIS
*CALL     VERSION
```

```
BASE     CON    START-BASE-1      INDEX LENGTH
CHE$$40  BSS    0
         SPACE  4
*        MACRO  P
A        MICRO  1,,)P)
A        MICCNT A
         ERRPL  A-65
B        DECMIC A+1
         DATA   'B'C P
         ENDM
         NOREF  A
         SPACE  4
/        MACRO  A,B
         QUAL   B
L!A      BSS    0
         USE    B
         XLAT   A,L!A-BASE+101B
         USE    *
         QUAL   *
         ENDM
         USE    SP
         USE    EX
         USE    SW
         USE    TEXT
START    BSS    0
         SPACE  4
         CHAR   1R[,1R(
         CHAR   1R],1R)
         CHAR   1R",1R'
         CODE   OTHER
         SPACE  4

 / HE,SP

 * (     THIS IS NORTHWESTERN UNIVERSITY"S CHESS-PLAYING COMPUTER)
 * (PROGRAM.  IT ACCEPTS MOVES IN A SLIGHTLY RESTRICTED ENGLISH)
 * (DESCRIPTIVE NOTATION.  IT ALSO ACCEPTS COMMANDS THAT)
 * (CHANGE ITS PLAYING STYLE, VARY THE AMOUNT OF TIME)
 * (SPENT PONDERING MOVES, SAVE AND SETUP POSITIONS, PRINT)
 * (OUT STATISTICS ABOUT THE THINKING PROCESSES, AND PERFORM)
 * (A NUMBER OF OTHER USEFUL AND INTERESTING FUNCTIONS.)
 * (BEFORE YOUR FIRST GAME WITH CHESS 'V', TYPE:)
 * (          EXPLAIN,OPERATION)
 * (FOR ASSISTANCE IN TYPING MOVES, TYPE:)
 * (          EXPLAIN,MOVES)
 * (FOR A DESCRIPTION OF THE VARIOUS COMMANDS, TYPE:)
 * (          EXPLAIN,COMMANDS)
 * (FOR A SUMMARY OF PLAYING STRATEGY AND PRINCIPLES, TYPE:)
 * (          EXPLAIN,STRATEGY)
 * (FOR A COPY OF EVERYTHING, TYPE:)
 * (          EXPLAIN,ALL)
 * (NOTE:  EXPLAIN,ALL  PRINTS OUT AN AWFUL LOT AND IS RECOMMENDED)
 * (ONLY FOR FAST TERMINALS AND THE BATCH PRINTER.)

 / $,SP

         USE    EX
 / EX,SP
         USE    *

 / COPY,EX

 * ()
 * (CHESS 4.5.)
 * ()
 * (COPYRIGHT NORTHWESTERN UNIVERSITY 1976, ALL RIGHTS RESERVED.)
 * ()

 / OPE,EX

 * (     WHEN A QUESTION MARK APPEARS AT THE TERMINAL, THE PROGRAM)
 * (IS USUALLY PREPARED TO ACCEPT EITHER A MOVE OR A COMMAND AS)
 * (INPUT.  BEFORE YOU START PLAYING, YOU SHOULD SET THE PACE OF)
 * (THE GAME WITH THE BLITZ COMMAND [TYPE: EXPLAIN,TOURNAMENT)
 * (FOR AN ALTERNATE METHOD].  EXAMPLE:  BLITZ,3  MEANS THAT)
 * (CHESS 'V' WILL AVERAGE 3 CPU SECONDS PER MOVE.  TO START)
 * (A GAME, TYPE A MOVE IF YOU WISH TO MOVE FIRST.  ELSE)
 * (TYPE  GO  TO MAKE THE COMPUTER MOVE FIRST.  TO PRINT OUT)
 * (THE BOARD POSITION IN ABBREVIATED FORM, TYPE: PRINT.)
 * (TO END A GAME, OBTAIN A SCORE, AND RESET THE PIECES FOR A NEW)
 * (GAME, TYPE: RESIGN.  OR TYPE: INIT.  TO END THIS CHESS)
 * (PLAYING SESSION ENTIRELY, TYPE:  END.  OR TYPE: DROP.)
 * ()
 * (     TO MAKE THE PROGRAM USABLE ON HIGH SPEED CRT TERMINALS, IT)
 * (BREAKS LARGE OUTPUT SEQUENCES INTO SHORT PAGES.  THE LENGTH OF)
 * (EACH PAGE IS DETERMINED BY THE LET VARIABLE   LINEPP .  [SEE)
 * (EXPLAIN,LET.]  WHEN THE PROGRAM HAS OUTPUT A FULL PAGE, IT)
 * (STOPS WITH THE MESSAGE "[PAUSING]".  AT THAT POINT YOU MAY)
 * ([1.] PRESS CARRIAGE RETURN, AND THE PROGRAM WILL CONTINUE, OR)
 * ([2.] ENTER A NEW LINE OF COMMANDS;  ANY PREVIOUS COMMANDS WILL)
 * (NOT BE PROCESSED.)
 * ()
 * (     TO RUN THE CHESS PROGRAM WITH INPUT FILE  A  AND)
 * (OUTPUT FILE B [INSTEAD OF THE DEFAULT FILES],)
 * (USE THE CONTROL CARD  CHESS,A,B. INSTEAD OF JUST CHESS.)
 * (OPENINGS LIBRARY INITIALIZATION IS CONTROLLED BY THE THIRD)
 * (PARAMETER ON THE CONTROL CARD.  ITS OPERATION DEPENDS ON THE)
 * (OPERATING SYSTEM THE PROGRAM IS RUNNING ON.  AT NORTHWESTERN)
 * (AND ON KRONOS TYPE SYSTEMS, THE THIRD PARAMETER IS THE NAME)
 * (OF A LOCAL OR PERMANENT FILE THAT CONTAINS A QUICK-LOAD)
 * (VERSION OF THE OPENINGS LIBRARY.  A  *  AS THE THIRD)
 * (PARAMETER CAUSES THE SLOW-LOAD VERSION OF THE LIBRARY TO)
 * (BE LOADED FROM THE SYSTEM.  ON OTHER SYSTEMS, THE SLOW-LOAD)
 * (VERSION IS USED AUTOMATICALLY.  TO SUPPRESS THAT LIBRARY LOAD)
 * ([WHICH MAY TAKE 10 MINUTES TO LOAD] SPECIFY A NULL THIRD)
 * (PARAMETER.)
 * (THUS, TO GET A FULL USERS GUIDE, YOU MAY RUN A BATCH JOB:)
 * ([JOB CARD])
 * (CHESS,INPUT,OUTPUT,.)
 * (7/8/9 EOR)
 * (EXPLAIN,ALL)
 * (END)
 * (6/7/8/9 EOI)

 / MOV,EX

 * (     MOVES MAY BE ENTERED IN EITHER A HIGHLY RESTRICTED)
 * (ALGEBRAIC NOTATION, OR IN ENGLISH DESCRIPTIVE NOTATION.  WHEN)
 * (USING ALGEBRAIC NOTATION,     MOVES ARE SPECIFIED BY TWO SQUARE)
 * (NAMES, SEPARATED BY A DASH [-] FOR SIMPLE MOVES, OR A STAR [*])
 * (FOR CAPTURES.  SQUARE NAMES ARE ALWAYS A LETTER [A-H])
 * (INDICATING THE FILE, AND A NUMBER [1-8] INDICATING A RANK.)
 * (RANKS AND FILES ARE COUNTED FROM WHITE'S QUEEN ROOK SQUARE,)
 * (WHICH IS  A1 .)
 * ()
 * (     WHEN USING ENGLISH DESCRIPTIVE NOTATION, NON-)
 * (CAPTURES GIVE THE PIECE TYPE MOVED, A DASH [-], AND THE SQUARE)
 * (MOVED TO, WHICH IS ONE OR TWO LETTERS SPECIFYING THE FILE)
 * (AND A DIGIT FOR THE RANK [ALWAYS COUNTING FROM THE SIDE MAKING)
 * (THE MOVE, CONTRARY TO ALGEBRAIC NOTATION].  LETTERS FOR PIECES)
 * (ARE P = PAWN, R = ROOK, N = KNIGHT, B = BISHOP, Q = QUEEN,)
 * (K = KING.  THUS, P-K4 MEANS PAWN TO 4TH RANK OF THE KINGS FILE)
 * (OR JUST "PAWN TO KING FOUR".  N-QB3 MEANS KNIGHT TO THIRD RANK)
 * (OF QUEENS BISHOPS FILE, OR "KNIGHT TO QUEEN BISHOP THREE".)
 * (IF A KNIGHT CANNOT GO TO KB3, THEN N-B3 SUFFICES. IF 2 KNIGHTS)
 * (CAN GO TO THE SAME SQUARE, THEN THE PIECE MAY BE CLARIFIED AS)
 * (FOLLOWS:  N/QN1-Q2 MEANS KNIGHT ON QUEEN KNIGHT ONE GOES TO)
 * (QUEEN TWO.  USUALLY, LESS INFORMATION SUFFICES TO CLARIFY A)
 * (MOVE.  THUS:  N/1-Q2 SPECIFIES A KNIGHT ON THE FIRST RANK, AND)
 * (R/B-N1 SPECIFIES A ROOK ON A BISHOP FILE.)
 * ()
 * (     CAPTURES ARE SPECIFIED BY GIVING THE PIECE TYPE MOVED, AN X)
 * (OR *, AND THE PIECE TYPE CAPTURED.  THUS, PXQ MEANS PAWN TAKES)
 * (QUEEN.  IF NECESSARY, THE RANK AND/OR FILE OF EITHER OR)
 * (BOTH PIECES MAY BE CLARIFIED.  AN EXTREME CASE MIGHT BE:)
 * (Q/QB3*P/KB6.  NOTE THAT THE KB6 SPECIFIES THE 6TH RANK)
 * (COUNTING FROM THE SIDE DOING THE CAPTURING.)
 * ()
 * (     KING SIDE CASTLING IS O-O, AND QUEEN SIDE IS O-O-O.)
 * ()
 * (     PAWN PROMOTION IS SPECIFIED BY APPENDING AN = [EQUALS SIGN])
 * (AND A PIECE LETTER TO THE MOVE.  THUS P-Q8=Q PROMOTES A PAWN)
 * (TO A QUEEN, AND P*R/R8=N PROMOTES TO A KNIGHT.)
 * ()
 * (     NOTE ON CLARIFICATION OF MOVES:  IT IS OK TO OVER-CLARIFY,)
 * (SO N/N1-KB3 IS ACCEPTED EVEN IF N-B3 IS SUFFICIENT.  BUT THE)
 * (PROGRAM COMPLAINS "AMBIGUOUS MOVE" IF MORE CLARIFICATION)
 * (IS NEEDED.  NOTE THAT THE NOTATION: KR-Q1, MEANING "KING)
 * (ROOK TO QUEEN ONE" IS UNACCEPTABLE.  NECESSARY MIGHT BE)
 * (SOMETHING LIKE  R/B-Q1.)
 * ()
 * (     THE PROGRAM COMPLAINS "ILLEGAL MOVE" WHEN APPROPRIATE.)
 * (IT SIGNALS ACCEPTANCE OF YOUR MOVE BY ECHOING IT BACK, E.G.)
 * ("YOUR MOVE-P-Q4."  YOU MAY PLACE A COMMENT AFTER ANY MOVE)
 * (IF IT IS PRECEDED BY A PERIOD, E.G. R-R7. CHECK.)
 * ()
 * (     EXAMPLE:  IN CHESS 'V' DESCRIPTIVE NOTATION, THE FIRST 6)
 * (MOVES OF THE FRIED LIVER ATTACK IN THE TWO KNIGHTS DEFENSE)
 * (WOULD BE:    P-K4    P-K4)
 * (             N-KB3   N-QB3)
 * (             B-B4    N-B3)
 * (             N-N5    P-Q4)
 * (             PXP     NXP)
 * (             NXP/B7  KXN)
 * ()
 * (     IN ALGEBRAIC NOTATION, THE ABOVE EXAMPLE LOOKS LIKE:)
 * (             E2-E4   E7-E5)
 * (             G1-F3   B8-C6)
 * (             F1-C4   G8-F6)
 * (             F3-G5   D7-D5)
 * (             E4*D5   F6*D5)
 * (             G5*F7   E8*F7)

 / STRA,EX

 * (         BRIEF DESCRIPTION OF CHESS 'V' PRINCIPLES)
 * ()
 * (     CHESS 'V' IS A CHESS-PLAYING COMPUTER PROGRAM WRITTEN)
 * (IN ASSEMBLY LANGUAGE [COMPASS] FOR CONTROL DATA 6000 OR)
 * (CYBER SERIES MACHINES.  IN ITS NORMAL MODE OF OPERATION, THE)
 * (PROGRAM ACCEPTS ITS INPUT FROM [AND SENDS ITS OUTPUT TO] A)
 * (TIME-SHARING INTERACTIVE TERMINAL.  INPUT IS IN THE FORM OF)
 * (CHESS MOVES IN DESCRIPTIVE NOTATION [E.G. P-K4] OR SPECIAL)
 * (COMMANDS, AND OUTPUT CONSISTS OF THE MACHINES MOVES IN)
 * (DESCRIPTIVE NOTATION PLUS DIAGNOSTIC INFORMATION.)
 * ()
 * (     CHESS 'V' WAS WRITTEN AT NORTHWESTERN UNIVERSITY IN)
 * (EVANSTON, ILL. BY LAWRENCE R. ATKIN AND DAVID J. SLATE.  IT)
 * (IS A SLIGHTLY IMPROVED VERSION OF CHESS 4.0, WHICH WAS)
 * (CREATED IN 1973.  CHESS 4.0 WAS A NEW PROGRAM RATHER THAN JUST)
 * (A MODIFICATION OF EARLIER N.U. PROGRAMS SUCH AS CHESS 2.0,)
 * (3.0, 3.5, OR 3.6.  STILL, 4.0 BORROWED MANY IDEAS FROM THESE)
 * (EARLIER PROGRAMS AND DOES NOT REPRESENT A MAJOR CONCEPTUAL)
 * (ADVANCE OVER THEM.  ITS PLAYING STRENGTH IS SOMEWHAT GREATER)
 * (THAN THAT OF CHESS 3.5.  4.0 IMPROVES OVER EARLIER VERSIONS)
 * (MAINLY IN TERMS OF MODULARITY, EASE OF MODIFICATION, AND)
 * (EFFICIENCY.  BASICALLY, THE PROGRAM DOES DEPTH-FIRST ALPHA-)
 * (BETA PRUNED SEARCHES OF THE MOVE TREE.  THE SEARCH IS)
 * (DEPTH-FIRST IN THE SENSE THAT THE WHOLE GAME TREE IS NOT)
 * (CONTINUALLY RETAINED.  PREVIOUS BRANCHES CAN BE [AND IN)
 * (CERTAIN CONDITIONS ARE] REVISITED, BUT ONLY AT THE COST OF)
 * (COMPLETELY REGENERATING THEM.  ALL THE HEURISTIC MOVE)
 * (SEARCHING AND POSITION EVALUATION DECISIONS ARE)
 * (CONCENTRATED IN ONE ROUTINE, CALLED EVALU8.  EVALU8 DIRECTS)
 * (THE PROCESSING OF A NODE IN THE TREE.  WHEN IT CALLS FOR A)
 * (MOVE TO BE SEARCHED, IT WILL BE CALLED AGAIN AT THE NEXT)
 * (HIGHER PLY LEVEL TO PROCESS THE RESULTING NODE.  THUS EVALU8)
 * (MUST BE RECURSIVE AND RE-ENTRANT, BECAUSE IT IS IN)
 * (SIMULTANEOUS USE AT DIFFERENT PLY LEVELS.  EVALU8 INVOKES)
 * (MODULAR, "NON-CONTROVERSIAL" ROUTINES TO DO THE BOOKKEEPING OF)
 * (UPDATING THE DATA BASE DURING TREE-SEARCHING.  THE DATA BASE)
 * (CONSISTS LARGELY OF BIT VECTORS DESCRIBING THE LOCATION OF THE)
 * (PIECES AND THE SQUARES THEY ATTACK.  ROUTINES MAKE EFFICIENT)
 * (USE OF HARDWARE FEATURES TO DO SUCH THINGS AS DELETE, ADD, AND)
 * (PROPOGATE ATTACKS OF MOVING PIECES.  EVALU8 ITSELF IS WRITTEN)
 * (MOSTLY IN HIGH LEVEL MACRO INSTRUCTIONS THAT RESEMBLE LISP)
 * (FUNCTIONS IN SYNTACTICAL FORM.  THESE MACROS ALLOW FOR ACCESS)
 * (AND MANIPULATION OF LOCATIONS OF PIECES, SQUARES THEY ATTACK,)
 * (LISTS OF SQUARES WITH CERTAIN PROPERTIES, AND OTHER TYPES OF)
 * (DATA.  THEY EXPAND INTO RELATIVELY EFFICIENT CODE WHILE SAVING)
 * (THE PROGRAMMER FROM PROBLEMS OF MACHINE REGISTER ALLOCATION,)
 * (ETC.)
 * (NOTE... FOR CHESS 4.6, EVALU8 WAS BROKEN INTO SEVERAL ROUTINES,)
 * (THE FIRST OF WHICH IS STILL CALLED EVALU8.  THE REMAINDER)
 * (HAVE NAMES BEGINNING WITH EV.)
 * ()
 * (     THE ACTUAL HEURISTICS IN EVALU8 ARE PRIMITIVE AND DO NOT)
 * (EXPLOIT VERY WELL THE POTENTIAL OF THE STRUCTURE [MACROS,)
 * (SUBROUTINES, DATA BASE] IN WHICH THEY ARE IMBEDDED.  THE TREE)
 * (SEARCH IS CARRIED OUT AS A SERIES OF ITERATIONS, EACH OF WHICH)
 * (ITSELF IS A COMPLETE FULL-WIDTH SEARCH TO A FIXED DEPTH, WITH)
 * (EXTENSIONS BEYOND THAT DEPTH ONLY FOR PIECE CAPTURE SEQUENCES)
 * (AND, IN SOME CASES, SEQUENCES OF CHECKS.  THE FIRST ITERATION)
 * (GOES TO DEPTH 2, AND EACH SUCCESSIVE ONE GOES A PLY DEEPER,)
 * (USING MOVE ORDERING AND EVALUATION INFORMATION FROM THE)
 * (PRECEDING ITERATION AS A GUIDE TO GAIN EFFICIENCY.  THE DEPTH)
 * (OF THE LAST ITERATION IS DETERMINED EITHER BY A PARAMETER)
 * (SETTING OR [AS IN A TOURNAMENT] BY AN AUTOMATIC TIME CONTROL)
 * (MECHANISM.  WITH THIS MECHANISM IN USE IN A TOURNAMENT,)
 * (DEPTHS RANGE FROM 3 TO 5 PLY, IN OPENING AND MIDDLE GAME, UP)
 * (TO 10 PLY IN THE ENDGAME.  THE USE OF A FULL-WIDTH SEARCH IS)
 * (A CONCESSION TO THE NOTION THAT FORWARD PRUNING HEURISTICS)
 * (SUCH AS WERE USED IN CHESS 3.5 WERE TOO NAIVE TO JUSTIFY THE)
 * (TIME SPENT ON THEM.)
```

```
* ()
* (     THE BRANCH ENDPOINT POSITION SCORES ARE BASED PRIMARILY)
* (ON MATERIAL, EXCEPT FOR THE OBVIOUS CASES OF CHECKMATE AND)
* (STALEMATE.  EACH PIECE IS GIVEN A STANDARD VALUE. THE MATERIAL)
* (SCORE IS THE SUM OF THESE VALUES PLUS A SECOND ORDER TERM THAT)
* (EXPRESSES THE PRINCIPLE THAT IT IS ADVANTAGEOUS TO TRADE)
* (PIECES [EXCEPT PAWNS] WHEN ONE IS AHEAD IN MATERIAL.  THERE IS)
* (ALSO A SPECIAL TABLE OF VALUES FOR VERY SIMPLE ENDGAMES, SO)
* (THAT, FOR EXAMPLE, KING + BISHOP VS. KING IS KNOWN TO BE)
* (DRAWN.)
* ()
* (     IN ADDITION TO THE MATERIAL SCORE, TERMS ARE ADDED WHICH)
* (EXPRESS, IN A PRIMITIVE WAY, NOTIONS OF MOBILITY [NO. OF)
* (SQUARES ATTACKED], PAWN STRUCTURE [PASSED, ISOLATED, DOUBLED,)
* (BACKWARD, ETC.], PIECE PLACEMENT [E.G. ROOKS ON 7-TH RANK],)
* (AND KING SAFETY [KING IN CASTLED POSITION, ADEQUATE PAWN)
* (COVER].  IN ENDGAMES IN WHICH ONE SIDE HAS AN OVERWHELMING)
* (ADVANTAGE, SEPARATE HEURISTICS ARE USED WHICH DRIVE THE)
* (OPPONENTS KING TO A CORNER OF THE BOARD.  EVALU8 DETECTS)
* (POSITION REPETITIONS BOTH WITHIN THE TREE AND IN A SHORT)
* (HISTORY OF ACTUAL MOVES, AND ASSIGNS THE DRAW VALUE TO THEM.)
* (IN ADDITION, A HEURISTIC IS USED THAT EXPRESSES THE 50-MOVE)
* (RULE.  THE SCORE IS PULLED TOWARD THE DRAW VALUE AS THE)
* (NUMBER OF CONSECUTIVE MOVES [BOTH WITHIN AND PRIOR TO THE)
* (TREE] WITHOUT A PIECE CAPTURED OR PAWN MOVED APPROACHES 50.)
* (ON THE AVERAGE, THE PROGRAM SCORES ABOUT 300 POSITIONS PER CPU)
* (SECOND ON A CDC 6400.  CHESS 'V' HAS A LIBRARY OF OPENING)
* (POSITIONS AND ASSOCIATED MOVES.  THIS LIBRARY IS CONSULTED)
* (BEFORE THE TREE SEARCH IS BEGUN.  IF A MATCH IS FOUND,)
* (THE ASSOCIATED MOVE IS PLAYED AND NO TREE SEARCH IS)
* (CONDUCTED.  PRESENTLY 'V'S LIBRARY CONTAINS ABOUT 5000)
* (ASSORTED POSITIONS.)

/ COMM,EX

* (     BESIDES TYPING MOVES, THE USER MAY TYPE COMMANDS THAT)
* (PERFORM VARIOUS FUNCTIONS.  A COMMAND HAS A KEYWORD AND,)
* (POSSIBLY, ONE OR MORE PARAMETERS, SEPARATED BY DELIMITERS.  A)
* (KEYWORD MAY BE ABBREVIATED, OFTEN DOWN TO THE MINIMUM NUMBER)
* (OF CHARACTERS NECESSARY TO DISTINGUISH IT FROM OTHER KEYWORDS.)
* (COMMANDS ARE TERMINATED BY END OF LINE OR CARD, OR BY A)
* (SEMICOLON, SO THAT SEVERAL COMMANDS [AND/OR MOVES] MAY)
* (BE TYPED ON THE SAME LINE, SEPARATED BY SEMICOLONS.  A)
* (DELIMITER IS, IN GENERAL, ANY COMBINATION OF BLANKS OR)
* (COMMAS OR MOST OTHER SPECIAL CHARACTERS [INCLUDING RIGHT)
* (PARENTHESIS AND PERIOD].  THUS TWO CONSECUTIVE COMMAS)
* (CONSTITUTE ONLY ONE DELIMITER AND DO NOT INDICATE A NULL)
* (PARAMETER.  SO  LET,FDEPTH=4  IS EQUIVALENT TO)
* (LET FDEPTH 4   AND TO   LET,,FDEPTH,,,4.   FOLLOWING)
* (IS A SUMMARY OF COMMAND KEYWORDS AND THEIR FUNCTIONS.)
* (FOR A FULLER EXPLANATION OF ANY ONE COMMAND, TYPE  EXPLAIN)
* (FOLLOWED BY THE KEYWORD OF THE COMMAND, AS IN  EXPLAIN,SAVE.)
* ()
* (KEYWORD        FUNCTION)
* ()
* (AFILE          SET FILE NAME FOR ALTER COMMAND.)
* (ALTER          READ COMMANDS AND/OR MOVES FROM ALTERNATE FILE.)
* (ANY            RESUME COMPUTATION IF PAUSING AT BREAKPOINT.)
* (BKP            SET BREAKPOINT LOCATION [FOR DEBUGGING PURPOSE])
* (BLITZ          SET TIMING PARAMETERS FOR FAST GAME.)
* (BOARD          SETUP POSITION USING MODIFIED FORSYTHE NOTATION)
* (CHANGE         CHANGE CONTENTS OF MEMORY LOCATION [DEBUGGING].)
* (CLOCK          SET ELAPSED REAL TIME IN MINUTES ON CHESS CLOCK)
* (COMPLEMENT     CHANGE BOARD TO REVERSE COLORS AND SIDES.)
* (CREATE         CHANGE BOARD BY SETTING UP AND ERASING)
* (               INDIVIDUAL PIECES.)
* (DISPLAY        PRINT CONTENTS OF MEMORY LOCATION [DEBUGGING].)
* (DROP           TERMINATE SESSION WITH PROGRAM [SAME AS END].)
* (DUMP           SAVE A QUICK-LOAD COPY OF THE OPENINGS LIBRARY.)
* (ECS            TURN ON OR OFF USE OF EXTENDED CORE STORAGE.)
* (END            TERMINATE SESSION WITH PROGRAM [SAME AS DROP].)
* (EXPLAIN        DESCRIPTION OF PROGRAM OPERATION.)
* (GO             MAKE A MOVE [BY COMPUTER].)
* (HARDCOPY       COPY SUBSEQUENT DIALOGUE TO PRINT-DISPOSED FILE)
* (HELP           INTRODUCTION TO PROGRAM DESCRIPTION.)
* (HOLD           TELL DUD TO TEMPORARILY RELEASE CENTRAL CONSOLE)
* (               TO OPERATING SYSTEM.)
* (ID             SET ID FOR PERMANENT FILE COMMANDS.)
* (INITIALIZE     TERMINATE GAME AND SET UP PIECES FOR A NEW ONE.)
* (KILL           TELL DUD TO SIMULATE OPERATOR KILL OF JOB.)
* (LET            SET VALUE OF INTEGER PARAMETER.)
* (LOAD           LOAD A QUICK-LOAD COPY OF THE OPENINGS LIBRARY.)
* (MSG            INSERT COMMENT INTO USER-PROGRAM DIALOGUE.)
* (MULTI          ALLOW SLAVE MONITOR TERMINALS.)
* (NAME           ENTER YOUR NAME [OF HUMAN OPPONENT].)
* (NULL           CHANGE WHOSE SIDE IT IS TO MOVE.)
* (PARAMETERS     PRINT VALUES OF LET AND/OR SWITCH PARAMETERS.)
* (PINFO          PRINT OUT CERTAIN DEBUGGING INFORMATION.)
* (PRINT          PRINT OUT CURRENT BOARD IN ABBREVIATED FORM.)
* (PSLIST         PRINT OUT SQUARE LIST [FOR DEBUGGING].)
* (PURGE          ERASE ALL PIECES FROM BOARD.)
* (PVARIABLE      PRINT VALUE OF A VARIABLE [FOR DEBUGGING].)
* (PWRD           SET PASSWORD FOR PERMANENT FILE COMMANDS.)
* (REINCARNATE    RECOVER PREVIOUSLY CHECKPOINTED STATE OF)
* (               PROGRAM.)
* (RELIABILITY    SAVE INFO NECESARY TO REINCARNATE IF SYSTEM)
* (               CRASH.)
* (RESIGN         SAME AS INITIALIZE.)
* (RETURN         RETURN A SCOPE FILE.)
* (REWIND         REWIND A SCOPE FILE.)
* (SAVE           SAVE CURRENT POSITION ON LIBRARY.)
* (SETUP          SETUP POSITION PREVIOUSLY SAVED.)
* (SPEED          SET TIMING PARAMETERS FOR SPEED PLAY.)
* (STATUS         CHANGE CASTLING AND ENPASSANT SETTINGS.)
* (STRIP          WRITE POSITIONS LIBRARY TO LEARN AND SETUP)
* (               SEQUENTIAL FILES.)
* (SWITCH         SET TWO-VALUED SWITCH PARAMETER.)
* (TOURNAMENT     SET PARAMETERS FOR TOURNAMENT PLAY.)
* (USE            MERGE STRIPPED FILE INTO CURRENT LIBRARY.)
* (WHAT           REPEAT PREVIOUS OUTPUT LINE.)
* (XEQ            EXECUTE SCOPE CONTROL CARD.)

/ RER,EX

* (     AN RE ERROR IS A PARITY ERROR [FAILURE] WHEN READING DATA)
* (FROM EXTENDED CORE STORAGE.  EXECUTION OF THE LAST MOVE OR)
* (COMMAND WAS INTERRUPTED, AND WILL HAVE TO BE REPEATED.  IF YOU)
* (HAD TYPED A MOVE, THAT MOVE HAS PROBABLY BEEN ENTERED, BUT THE)
* (MACHINES REPLY MUST BE RE-REQUESTED VIA THE  GO  COMMAND.)
* (PRINT THE BOARD WITH THE  PRINT  COMMAND TO BE SURE.  IF THE)
* (ERROR OCCURS REPEATEDLY, YOU SHOULD TYPE: ECS,OFF   TO SWITCH)
* (TO CENTRAL MEMORY USE ONLY.  IF YOU ARE IN TOURNAMENT MODE,)
* (TRANSPOSITIONS HASH TABLE LENGTH MAY BE A PROBLEM.  SEE:)
* (EXPLAIN,ECS;EXPLAIN,PRINT;EXPLAIN,GO;EXPLAIN,WER;)
* (EXPLAIN,TOURN.)

/ WER,EX

* (     A WE ERROR IS A PARITY ERROR [FAILURE] WHEN WRITING)
* (DATA TO EXTENDED CORE STORAGE.  IT IS A RARER AND MORE PECULIAR)
* (CONDITION THAN AN RE [READ] ERROR, BUT IS SIMILAR IN ITS EFFECT)
* (ON THE PROGRAM.  SEE:  EXPLAIN,RER.)

/ TDIS,EX

* (     UNDER KRONOS TYPE OPERATING SYSTEMS, THE TI-DISPLAY UNDER)
* (DIS MAY BE USED IN CONJUCTION WITH THE DIS  M.  COMMAND)
* (FOR COMMUNICATING WITH THE PROGRAM.  THE ADDRSS OF THE)
* (T DISPLAY BUFFER IS IN LOCATION 0, BYTE 4.  THE PROGRAM)
* (MUST BE IN DSD MODE FOR THE  M.  COMMAND TO BE EFFECTIVE.)
* (SEE THE KRONOS OPERATORS MANUAL FOR DETAILS.)

/ AF,EX

* (AFILE,LFN)
* (AFILE [ABBREVATION AF] SETS FROM ITS SINGLE PARAMETER THE)
* (DEFAULT FILE NAME TO BE USED FOR AN ALTER COMMAND. A FILE)
* (NAME GIVEN ON THE ALTER COMMAND OVERRIDES THE DEFAULT.)
* (BEFORE ANY AFILE COMMAND IS ISSUED THE DEFAULT ALTER FILE)
* (NAME IS ALTER.  SEE  EXPLAIN,ALTER.   SEE ALSO:)
* (EXPLAIN,SWITCH,MALTER  AND  EXPLAIN,SWITCH,DALTER.)

/ AL,EX

* (ALTER,LFN)
* (ALTER  [ABBREVIATION AL] CAUSES THE ALTERNATE INPUT FILE TO BE)
* (REWOUND AND SUBSEQUENT COMMANDS AND MOVES TO BE READ FROM THAT)
* (FILE UNTIL END-OF-RECORD IS REACHED, AFTER WHICH READING IS)
* (RESUMED FROM THE PRIMARY INPUT FILE.  THE ALTERNATE FILE MAY)
* (ITSELF CONTAIN AN ALTER COMMAND, BUT NO STACK OF SUCH FILES)
* (IS KEPT, AND AFTER SUCH A COMMAND IS COMPLETED CONTROL IS GIVE)
* (TO THE PRIMARY INPUT FILE, NOT THE ONE CONTAINING THE ALTER)
* (COMMAND.  THE FILE READ BY AN ALTER COMMAND MAY BE SPECIFIED)
* (AS A PARAMETER ON THE COMMAND. IF NONE IS GIVEN, THEN THE FILE)
* (NAMED ON THE LAST AFILE COMMAND IS USED.  IF NO SUCH COMMAND)
* (HAD BEEN GIVEN, THEN THE FILE NAMED ALTER IS USED.  SEE ALSO)
* (EXPLAIN,AFILE ; EXPLAIN,SW,MALTER ; EXPLAIN,SW,DALTER.)

/ AN,EX

* (ANY)
* (ANY [ABBREVIATION A] RESUMES PROCESSING IF TYPED WHILE PAUSING)
* (AT A BREAKPOINT ADDRESS.  IT IS ILLEGAL AT ANY OTHER TIME.)
* (SEE ALSO: EXPLAIN,BKP ; EXPLAIN,SW,DEBUG ; EXPLAIN,SW,DALTER.)

/ BK,EX

* (BKP,ADDRESS)
* (BKP [ABBREVIATION BK] IS A DEBUGGING COMMAND THAT SETS [OR)
* (CLEARS] A BREAKPOINT ADDRESS IN THE THINKING [1,0] OVERLAY.)
* (THE ADDRESS IS SPECIFIED IN OCTAL [E.G. BKP,13604].  IF NO)
* (ADDRESS IS SPECIFIED, THEN ANY EXISTING BREAKPOINT IS CLEARED.)
* (WHEN THE P-REGISTER REACHES THE BREAKPOINT ADDRESS, EXECUTION)
* (IS INTERRUPTED, REGISTER CONTENTS ARE SAVED, AND THE INPUT)
* (FILE IS READ FOR COMMANDS.  COMMANDS MAY THEN BE GIVEN TO)
* (PRINT OUT CONTENTS OF MEMORY LOCATIONS, VALUES OF VARIABLES,)
* (ETC.  EXECUTION IS RESUMED BY TYPING  ANY  .  SOME COMMANDS,)
* (SUCH AS INI, PREVENT THE RESUMPTION OF THE INTERRUPTED MOVE.)
* (ENTERING A NEW BREAKPOINT ADDRESS CLEARS ANY OTHER STILL IN)
* (EFFECT.  WHEN THE PROGRAM PAUSES AT A BREAKPOINT, A MESSAGE IS)
* (PRINTED GIVING THE STRING  BKPT  , THE CURRENT PLY NUMBER, THE)
* (LABEL ON THE LAST CALL TO THE TREE SEARCH ROUTINE, AND THE)
* (OCTAL REPRESENTATION OF THE MOVE SEARCHED TO GET TO THE)
* (CURRENT PLY LEVEL. SAMPLE MESSAGE:  BKPT  1  BAS4B  1024000120)
* (NOTE: WHILE BKP ALLOWS BREAKPOINTING AT ARBITRARY ADDRESSES,)
* (THERE ARE 2 PRESET BREAKPOINTS THAT MAY BE INVOKED BY OTHER)
* (COMMANDS AND CAN OPERATE INDEPENDENTLY AND SIMULTANEOUSLY WITH)
* (EACH OTHER AND WITH BKP.  THE DEBUG SWITCH TURNS ON A BREAK-)
* (POINT IN EVALU8 AT THE BASE [ZERO] PLY LEVEL JUST BEFORE EACH)
* (FULL WIDTH SEARCH ITERATION IS BEGUN.  ITS PAUSING MESSAGE)
* (LOOKS LIKE:  ITER  0  BAS4B  300000.  THE 300000 HERE IS NOT A)
* (MOVE, BUT IS MEANINGLESS, SINCE WE ARE AT THE BASE LEVEL.)
* (THE LETS  STEPLO  AND  STEPHI  CONTROL ANOTHER BREAKPOINT.)
* (THIS ONE IS IN THE TREE-SEARCH ROUTINE IN CREATE JUST AFTER)
* (THE MOVE [TO BE SEARCHED] HAS BEEN MADE, THE DATA BASE)
* (UPDATED, AND THE PLY NUMBER INCREMENTED, AND JUST BEFORE)
* (EVALU8 IS CALLED TO PROCESS THE UPDATED POSITION.  THE)
* (BREAKPOINT IS ACTIVE ONLY IF THE PLY NUMBER LIES WITHIN THE)
* (RANGE STEPLO THRU STEPHI.  SAMPLE PAUSING MESSAGE:)
* (TSRCH  2  FUL11  101404334 .  SEE ALSO:  EXPLAIN,SW,DEBUG;)
* (EXPLAIN,SW,DALTER.)

/ BL,EX

* (BLITZ,N)
* (BLITZ  [ABBREVIATION BL] SWITCHES ON TIMING AND SETS LET)
* (VARIABLES TO PLAY A FAST GAME AT AN AVERAGE RATE OF N CPU)
* (SECONDS OF MACHINE THINK TIME PER MOVE, WHERE N IS SPECIFIED)
* (ON THE BLITZ COMMAND [EXAMPLE:  BLITZ,7. ].  IF N IS OMITTED,)
* (5 IS ASSUMED.  BLITZ MODE IS USUALLY MORE DESIRABLE THAN THE)
* (DEFAULT [2-PLY MINIMUM FIXED DEPTH] MODE THAT THE PROGRAM)
* (COMES UP IN.  NOTE THAT SINCE AT MINIMUM SEARCH DEPTH SEVERAL)
* (SECONDS MAY BE CONSUMED BY SOME MOVES IN COMPLICATED)
* (POSITIONS, SETTING N VERY LOW [LIKE:  BLITZ,1. ] MAY NOT WORK)
* (WELL FOR A FULL GAME EXCEPT WITH A VERY FAST CPU.  ALSO NOTE)
* (THAT SINCE THE BLITZ COMMAND CHANGES VALUES OF LET VARIABLES,)
* (THOSE LETS: RULTM1,RULTM2,MDEPTH,OVRHED,PERCNT,EXTRAS,EXTRAT,)
* (JIGMVS, JIGSIZ,)
* (RATEMY,RATEYR,RULMV1,RULMV2,TQFRST, AND TQNEXT  MUST BE)
* (PROPERLY RESET BEFORE A SUBSEQUENT TOURNAMENT COMMAND IS)
* (ISSUED.  SEE:  EXPLAIN,SWITCH,TIMING ; EXPLAIN,TOURNAMENT.)

/ BO,EX

* (BOARD,LIST)
* (BOARD [ABBREVIATION BO] SETS PIECES UP ON THE BOARD.  SINCE)
* (THE BOARD IS NOT CLEARED AUTOMATICALLY, A PURGE COMMAND)
* (USUALLY PRECEDES THE BOARD COMMAND.  BOARD SPECIFIES SQUARES)
* (BY ROWS STARTING WITH WHITE"S FIRST RANK, AND WITHIN ROWS BY)
* (SQUARES STARTING AT THE QUEEN ROOK [A] FILE.  BOARD IS BEST)
* (EXPLAINED WITH AN EXAMPLE: BOARD,3LK486DN183K2X1LR788.  MEANS:)
* (SKIP QR1,QN1, AND QB1, PUT A WHITE [L] KING ON Q1, SKIP)
* (REMAINING 4 SQUARES ON FIRST RANK, SKIP ALL 8 SQUARES ON 2ND)
* (RANK, SKIP 6 SQUARES ON 3RD RANK, PUT A BLACK [D] KNIGHT ON)
* (KN3, SKIP KR3, SKIP 4TH RANK, SKIP QR5,QN5,QB5, PUT A BLACK)
* (KING [SINCE NEITHER L FOR LIGHT NOR D FOR DARK WAS GIVEN,)
* (THE COLOR OF THE PREVIOUS PIECE IS USED] ON Q5, SKIP K5)
* (AND KB5, CLEAR [X] THE KN5 SQUARE [IF IT HAD A PIECE ON IT],)
* (SKIP KR5, PUT A WHITE ROOK ON QR6, SKIP REST OF 6TH RANK,)
* (AND SKIP 7TH AND 8TH RANKS.  SEE ALSO  EXPLAIN,PURGE.  AND)
* (EXPLAIN,CREATE.  AND  EXPLAIN,STATUS.)

/ CH,EX
```

```
* (CHANGE,ADDRESS,VALUE)
* (CHANGE [ABBREVIATION CH] TAKES TWO PARAMETERS: AN OCTAL)
* (ADDRESS AND AN OCTAL VALUE [UP TO 20 DIGITS] SEPARATED BY)
* (A MANDATORY COMMA OR OTHER NON-BLANK DELIMITER.  EXAMPLE:)
* (CH,2156,42703671567124567320.  CHANGE IS A DEBUGGING COMMAND)
* (THAT CHANGES THE CONTENTS OF THE SPECIFIED MEMORY LOCATION)
* (TO THE SPECIFIED VALUE.  IF THE ADDRESS IS NOT IN THE MAIN)
* (OVERLAY, WHATEVER OVERLAY IS CURRENTLY LOADED IS THE ONE THAT)
* (IS CHANGED.  TO MAKE A CHANGE TO THE [1,0] THINKING OVERLAY)
* (THAT IS RETAINED FOR THE WHOLE PLAYING SESSION, BRACKET THE)
* (CHANGE COMMAND WITH BKP COMMANDS.  E.G. BKP;CH,12437,4025;BKP.)
* (THE FIRST BKP LOADS THE [1,0] OVERLAY.  THE SECOND BKP CAUSES)
* (THE OVERLAY TO BE RE-WRITTEN WITH THE CHANGE.  NOTE: CHANGE)
* (SHOULD BE USED CAUTIOUSLY.)

/ CL,EX

* (CLOCK,TIME)
* (CLOCK [ABBREVIATION CL] TAKES ONE PARAMETER, THE ELAPSED)
* (TIME IN MINUTES ON THE COMPUTERS SIDE OF THE CHESS CLOCK)
* (SINCE THE START OF THE GAME. CLOCK ADJUSTS THE REAL TIME CLOCK)
* (ESTIMATE USED IN TOURNAMENT [OR TIMING] MODE.  SINCE THE)
* (PROGRAM AUTOMATICALLY ASKS FOR THE TIME AT APPROPRIATE)
* (INTERVALS DURING A TOURNAMENT GAME, THE CLOCK COMMAND IS)
* (NEEDED ONLY UNDER EXTRAORDINARY CIRCUMSTANCES, SUCH AS)
* (MACHINE CRASHES, RETRACTED MOVES, ETC.  SEE ALSO)
* (EXPLAIN,SWITCH,TIMING. AND EXPLAIN,TOURNAMENT.)

/ COMP,EX

* (COMPLEMENT)
* (COMPLEMENT [ABBREVIATION CO] TAKES EACH PIECE, REVERSES ITS)
* (COLOR, AND MOVES IT TO THE CORRESPONDING SQUARE ON THE OTHER)
* (SIDE OF THE BOARD.  THUS A WHITE KNIGHT ON WHITES KB3 BECOMES)
* (A BLACK KNIGHT ON BLACKS KB3, ETC.  THE SIDE TO MOVE, THE)
* (CASTLING STATUSES, AND THE ENPASSANT SQUARE, IF ANY, ARE ALSO)
* (REVERSED.  THE NET EFFECT IS THAT WHITE GETS THE SAME POSITION)
* (THAT BLACK HAD, AND VISA VERSA.)

/ CR,EX

* (CREATE,LIST)
* (CREATE  [ABBREVIATION CR] SETS UP OR REMOVES PIECES FROM)
* (THE BOARD.  EACH PIECE IS SPECIFIED BY ITS COLOR, L)
* ([FOR LIGHT OR WHITE] OR D [FOR DARK OR BLACK], ITS TYPE, P,)
* (R,N,B,Q,K, OR X [FOR CLEARING THE SQUARE], AND THE SQUARE)
* (IT IS TO GO ON.  RANKS ARE COUNTED FROM THE SIDE TO MOVE, NOT)
* (THE SIDE WHOSE PIECE IS BEING SET UP.  EXAMPLE:)
* (CR,LPQR4,DNKB6,XKR2.  SETS A WHITE PAWN ON [WHITES] QR4, A)
* (BLACK KNIGHT ON [WHITES] KB6, AND CLEARS [WHITES] KR2,)
* (ASSUMING WHITE WERE ON THE MOVE.  THE COLOR AND/OR PART OR ALL)
* (OF THE FILE DESIGNATOR MAY BE OMITTED IF IT IS THE SAME AS)
* (THAT OF THE PRECEDING PIECE.  SO: CR,LPKR3,PN4,P5.)
* (SETS WHITE PAWNS ON KR3,KN4, AND KN5.  SEE ALSO)
* (EXPLAIN,STATUS. AND EXPLAIN,BOARD. AND EXPLAIN,PURGE.)

/ DI,EX

* (DISPLAY,ADDRESS)
* (DISPLAY  [ABBREVIATION DI] IS A DEBUGGING COMMAND THAT)
* (DISPLAYS, IN OCTAL, THE CONTENTS OF ONE OR MORE CONTIGUOUS)
* (WORDS OF MEMORY.  THE TWO PARAMETERS ARE: THE OCTAL ADDRESS)
* (OF THE FIRST WORD TO DISPLAY, AND AN OCTAL COUNT OF THE)
* (NUMBER OF WORDS TO DISPLAY.  IF THE COUNT IS OMITTED, IT IS)
* (ASSUMED TO BE ONE.  THE ADDRESS AND COUNT SHOULD BE SEPARATED)
* (BY COMMAS.  A REQUEST TO DISPLAY A WORD OUT OF THE FIELD)
* (LENGTH CAUSES *OUT OF RANGE* TO BE PRINTED.  EXAMPLES:)
* (DI 1276,3   DISPLAYS CONTENTS OF 1276, 1277, AND 1300.)
* (DISPLAY,4356   DISPLAYS CONTENTS OF 4356.)
* (SEE ALSO  EXPLAIN,PVARIABLE  AND  EXPLAIN,PSLIST.)

/ DR,EX

* (DROP)
* (DROP  [ABBREVIATION DR] TERMINATES THE PROGRAM NORMALLY.)
* (THE MESSAGE *THANK YOU FOR AN ENJOYABLE TIME* IS WRITTEN)
* (TO THE OUTPUT FILE, AND *GOODBYE* APPEARS IN THE DAYFILE.)
* (THE POSITIONS LIBRARY IS PROPERLY CLOSED.  A DROP DURING)
* (A GAME INHIBITS PRINTING OF THE GAME SCORE.  TO GET A)
* (SCORE, TYPE  INI  OR  RESIGN  BEFORE DROPPING.  THE FILE)
* (HARDCPY  IS NOT AUTOMATICALLY DETACHED TO THE PRINT QUEUE)
* (BY THE DROP COMMAND, BUT IT DOES HAVE END-OF-JOB PRINT)
* (DISPOSITION.   SEE ALSO  EXPLAIN,END.)

/       DUMP,EX

* (DUMP LFN)
* (DUMP  [NO ABBREVIATION] COPIES OUT A QUICK DUMP VERSION OF THE)
* (OPENINGS LIBRARY TO  LFN .  IF THE PARAMETER IS OMITTED, THE)
* (FILE  OPENING  IS USED.  DUMP WORKS ONLY)
* (AT NORTHWESTERN AND ON KRONOS.  SEE  EXPLAIN,LOAD.)

/ EC,EX

* (ECS,ONOFF)
* (ECS  [ABBREVIATION EC] DIRECTS THE PROGRAM TO USE [OR)
* (NOT USE] EXTENDED CORE STORAGE.  AT ANY TIME DURING A)
* (GAME, ECS,ON  TURNS ON USE OF EXTENDED CORE STORAGE, WHILE)
* (ECS,OFF  TURNS IT OFF.  IF THE PARAMETER IS OMITTED, ECS)
* (STATUS IS TOGGLED [I.E., TURNED ON IF IT IS OFF AND OFF IF IT)
* (IS ON].  EXTENDED CORE STORAGE IS AN AUXILIARY FAST MEMORY)
* (DEVICE AVAILABLE ON SOME COMPUTERS.  THE PROGRAM USES ECS IN)
* (THE DIRECT ACCESS MODE, WHICH MAY NOT BE ALLOWED BY THE)
* (OPERATING SYSTEM EVEN IF THE MACHINE HAS ECS.  USE)
* (OF ECS IMPROVES THINKING SPEED BY AS MUCH AS 2 TO 1, AND)
* (CUTS MAIN MEMORY USE BY AS MUCH AS 3 TO 1.  THE PROGRAM)
* (ASSUMES  ECS,ON  IF THE JOB HAS NON-ZERO ECS FIELD LENGTH)
* (AT THE TIME CHESS IS CALLED UP.  OTHERWISE,  ECS,OFF  IS)
* (ASSUMED, AND THE MESSAGE:  ENTER *ECS ON* FOR BETTER)
* (RESPONSES.   IS PRINTED.  WHEN ECS IS ON, THE PROGRAM)
* (WILL GET AS MUCH ECS AS IT NEEDS DYNAMICALLY.  MAXIMUM)
* (AMOUNTS USED VARY FROM ABOUT 20000B IN BLITZ MODE TO ABOUT)
* (152000B IN TOURNAMENT MODE, ASSUMING DEFAULT VALUES FOR)
* (TRANSPOSITIONS/REFUTATIONS TABLE SIZE.  SEE EXPLAIN,TOURNAMENT.)

/ EN,EX

* (END)
* (END  [ABBREVIATION EN] ENDS THE SESSION WITH THE CHESS)
* (PROGRAM.  END IS EXACTLY EQUIVALENT TO  DROP.)
* (SEE    EXPLAIN,DROP.)

/ GO,EX

* (GO,NUMBER)
* (GO  [NO ABBREVIATION] CAUSES THE PROGRAM TO MAKE MOVES.  THE)
* (NUMBER OF MOVES TO MAKE IS SPECIFIED AS THE SINGLE DECIMAL)
* (PARAMETER, WHICH IS ASSUMED ONE IF OMITTED.  MOVES ARE)
* (MADE FOR ALTERNATE SIDES.  THUS IF BLACK IS ON THE MOVE,)
* (GO,3 SELECTS AND MAKES A MOVE FOR BLACK, ONE)
* (FOR WHITE, AND ANOTHER FOR BLACK.  SO, FROM THE INITIAL)
* (BOARD POSITION,  GO  CAN BE USED TO GIVE THE PROGRAM THE)
* (WHITE PIECES.)

/ HA,EX

* (HARDCOPY,LFN)
* (HARDCOPY  [ABBREVIATION HA] CAUSES SUBSEQUENT DIALOGUE)
* ([MOVES AND COMMANDS TYPED BY THE USER AND RESPONSES TYPED BY)
* (THE PROGRAM] TO BE LOGGED ON A DISK FILE WHOSE NAME IS)
* (GIVEN AS THE PARAMETER TO THE HARDCOPY COMMAND.  IF NO)
* (PARAMETER IS SPECIFIED, THE NAME  HARDCPY  IS USED.  ONCE)
* (HARDCOPY IS IN EFFECT, IT CANNOT BE TURNED OFF, NOR CAN)
* (THE FILE NAME BE CHANGED.  ADDITIONAL HARDCOPY COMMANDS)
* (ARE GREETED WITH THE ERROR MESSAGE:  HARD COPY COMMAND)
* (IGNORED.)
* ()
* (THE HARD COPY FILE IS GIVEN PRINT DISPOSITION BY THE)
* (PROGRAM, BUT IS NOT AUTOMATICALLY DISPOSED TO THE PRINT)
* (QUEUE BY AN END OR DROP COMMAND.)

/ HO,EX

* (HOLD)
* (HOLD  [NO ABBREVIATION] IS VALID ONLY WHEN PLAYING AT THE)
* (CENTRAL CONSOLE THROUGH THE DUD [DYNAMIC USER DISPLAY])
* (PP PROGRAM.  HOLD RELEASES THE CONSOLE EQUIPMENT BACK)
* (TO DSD.  TO RESUME PLAY, THE CONSOLE MUST BE MANUALLY)
* (RE-ASSIGNED TO THE PROGRAM.)

/ ID,EX

* (ID,IDENT)
* (ID  SETS [FROM ITS SINGLE PARAMETER] THE ID TO BE USED FOR)
* (PERMANENT FILE REQUESTS WHICH THE PROGRAM MAY MAKE IN RESPONSE)
* (TO COMMANDS SUCH AS RELIABILITY AND REINCARNATE.)
* (SEE ALSO  EXPLAIN,PWRD.)

/ IN,EX

* (INITIALIZE)
* (INIIIALIZE  [ABBREVIATION IN] TERMINATES THE GAME IN PROGRESS)
* (AND SETS THE PIECES UP IN THEIR INITIAL POSITIONS FOR A NEW)
* (GAME.  A GAME SCORE OF THE INTERRUPTED OR COMPLETED GAME IS)
* (PRINTED AT THE TERMINAL [OR WHATEVER THE STANDARD OUTPUT)
* (FILE IS].  ADDITIONAL COPIES OF THE SCORE, DEPENDING ON THE)
* (VALUE OF THE LET COPIES PARAMETER, ARE DISPOSED TO THE BATCH)
* (PRINT QUEUE.  A NEW RANDOM KEY IS GENERATED FOR SELECTING)
* (OPENING MOVES FOR THE NEXT GAME.  TIMING CONTROLS ARE)
* (REINITIALIZED BASED ON THE LATEST VALUES OF THE ASSOCIATED)
* (LET PARAMETERS.  SEE ALSO:  EXPLAIN,RESIGN.)

/ KI,EX

* (KILL)
* (KILL  [NO ABBREVIATION] IS VALID WHEN PLAYING VIA DUD)
* ([DYNAMIC USER DISPLAY PP PROGRAM].  KILL SIMULATES AN OPERATOR)
* (KILL OF THE JOB.  THE GAME CURRENTLY IN PROGRESS IS)
* (TERMINATED AND NO OUTPUT IS PRINTED.  IF TYPED WHEN NOT)
* (UNDER DUD, KILL ACTS LIKE DROP OR END.)

/ LE,EX

* (LET,PARAMETER=VALUE)
* (LET  [ABBREVIATION LE] CHANGES THE VALUE OF ONE OF SOME)
* (93 ADJUSTABLE VARIABLES THAT CONTROL THE PLAY OF THE)
* (PROGRAM.  THE FIRST PARAMETER IS THE NAME OF THE VARIABLE,)
* (AND THE SECOND IS ITS NEW VALUE.  THE VALUE MAY OR MAY NOT)
* (HAVE A POST-RADIX D OR B SPECIFYING ITS BASE.  IF NO BASE)
* (IS GIVEN, THE PROGRAM USES A DEFAULT THAT WAS PRESET WITH THE)
* (DEFINITION OF THE GIVEN VARIABLE.)
* ()
* (EXAMPLES:)
* ()
* (LET,RULMV1,30  CHANGES THE VALUE OF RULMV1 TO 30 [DECIMAL].)
* (LET FQMOBL 100 CHANGES THE VALUE OF FQMOBL TO 100 [OCTAL].)
* (LET FQMOBL 64D CHANGES THE VALUE OF FQMOBL TO 64 [DECIMAL].)
* (LET RULMV1=36B CHANGES THE VALUE OF RULMV1 TO 36 [OCTAL].)
* ()
* (FOLLOWING IS A LIST OF LET VARIABLES, THEIR DEFAULT VALUES,)
* (AND BRIEF DESCRIPTIONS OF THEIR FUNCTIONS.  NOTE -- SOME LETS)
* (ARE ALTERED BY THE PROGRAM EITHER AUTOMATICALLY IN RESPONSE)
* (TO CERTAIN PLAYING CONDITIONS OR AS A SIDE EFFECT OF CERTAIN)
* (OTHER COMMANDS.  SOME OF THESE ARE NOT USEFULLY CHANGED BY)
* (LET COMMANDS.)
* ()
* ( NAME   BASE           FUNCTION                        VALUE)
* ()
* (ASKFCT   D   PERCENT TO TIME CONTROL TO ASK TQ        40)
* (COPIES   D   NO. OF EXTRA GAME SCORES AT CENTRAL      0)
* (EXTRAS   D   SACRED MINIMUM EXTRAT                    5D)
* (EXTRAT   D   MINUTES TO STAY AHEAD OF TIME CONTROL    20)
* (FBBKRK   B   PENALTY FOR BISHOP ON BACK RANK          700B)
* (FBMOBL   B   BISHOP MOBILITY FACTOR                   200B)
* (FCHKOP   B   MATING KING 2-FROM-EDGE BONUS            120B)
* (FDEPTH   D   DEPTH OF MAX ITERATION IF TIMING OFF     2)
* (FDRAWA   D   DRAW ADJUST MAX MOVE FACTOR              77D)
* (FDRAWG   B   DRAW ADJUST MAX INCREASE                 20B)
* (FDRAWI   D   DRAW ADJUST IRREV. MOVE FACTOR           2)
* (FDRAWM   D   DRAW ADJUST MAX MOVES COUNT              55D)
* (FDRAWT   D   DRAW ADJUST FRACTIONAL TOLERANCE         3)
* (FEZTOL   B   THAT-WAS-EASY SCORE TOLERANCE            20B)
* (FKCORN   B   MATED KING CORNER HATRED                 300B)
* (FKCTRP   B   ENDGAME KING CENTER TROPISM              140B)
* (FKKNMT   B   MATING KING AND KNIGHT TROPISM           100B)
* (FKPTRP   B   ENDGAME KING TO PAWN ATTRACTION          500B)
* (FKSANQ   B   KING SAFETY SANCTUARY FACTOR             200B)
* (FKSATK   B   KING SECTOR ATTACK FACTOR                24B)
* (FKSCNQ   B   KING SANCTUARY PENALTY IF CAN CASTLE     120B)
* (FKSCOF   B   KING OPEN FILE PENALTY IF CAN CASTLE     100B)
* (FKSCRM   B   KING SAFETY CRAMP FACTOR                 40B)
* (FKSFRD   B   KING SAFETY CLOSE FRIENDS FACTOR         100B)
* (FKSISO   B   KING WITH ISOLATED PAWN FACTOR           300B)
* (FKSMAT   B   KING SAFETY ATTACK COUNT THRESHOLD       2)
* (FKSMNF   B   KING SAFETY MIN FRIEND COUNT             2B)
* (FKSOPF   B   KING ON OPEN FILE FACTOR                 340B)
* (FKSQBN   B   KING SAFETY QUEEN PRESENCE FACTOR        2B)
* (FKS2BS   B   KING FILE WITH 2 BARE SQS PENALTY        100B)
* (FLXTOL   B   ALPHA-BETA WINDOW MARGIN                 100B)
* (FMAXMT   B   MAX MATERIAL EDGE SANS TRADE BONUS       17700B)
* (FMTEGE   B   MIN EDGE FOR USING MATING ALGORITHM      400B)
* (FNATKD   B   ATTACKED KNIGHT PENALTY                  200B)
* (FNATNP   B   ATT. KNIGHT UNDEF. BY PAWN PENALTY       400B)
* (FNBKRK   B   PENALTY FOR KNIGHT ON BACK RANK          600B)
```

```
* (FNCTRP   B   KNIGHT CENTER TROPISM                    100B)
* (FNKTRP   B   KNIGHT KING TROPISM                      60B)
* (FORKBN   B   PENALTY FOR MULTIPLE HUNG PIECES         1700B)
* (FPDCQR   B   PAWN ADVANCE CREDIT FOR QR COL           -20B)
* (FPDCQN   B   PAWN ADVANCE CREDIT FOR QN COL           -20B)
* (FPDCQB   B   PAWN ADVANCE CREDIT FOR QB COL           200B)
* (FPDCQC   B   PAWN ADVANCE CREDIT FOR QC COL           300B)
* (FPDCKC   B   PAWN ADVANCE CREDIT FOR KC COL           360B)
* (FPDCKB   B   PAWN ADVANCE CREDIT FOR KB COL           120B)
* (FPDCKN   B   PAWN ADVANCE CREDIT FOR KN COL           -40B)
* (FPDCKR   B   PAWN ADVANCE CREDIT FOR KR COL           -40B)
* (FPNBSC   B   BLOCKED 2ND RNK CTR PAWN PENALTY         600B)
* (FPNDBL   B   DOUBLED PAWN PENALTY                     500B)
* (FPNISC   B   LESS EXPOSED ISOLATED PAWN PENALTY       700B)
* (FPNISO   B   ISOLATED PAWN PENALTY                    1500B)
* (FPNLBK   B   SEMI-FLUID PAWN PENALTY                  240B)
* (FPNMAJ   B   PAWN MAJORITY ADVANCE CREDIT             101B)
* (FPNPAS   B   PASSED PAWN ROW**2 BONUS                 140B)
* (FPNPSA   B   PASSED PAWN ATTACK NEXT SQ CREDIT        20B)
* (FPNPSB   B   PASSED PAWN BLOCK CREDIT                 36B)
* (FPNPSD   B   PASSED PAWN DEFEND NEXT SQ CREDIT        14B)
* (FPNVBK   B   SUPER-OPPOSED PAWN PENALTY               500B)
* (FQKTRP   B   QUEEN KING TROPISM                       40B)
* (FQMOBL   B   QUEEN MOBILITY FACTOR                    40B)
* (FRHUNG   B   PENALTY FOR HAVING HUNG PIECE            1500B)
* (FRKDBL   B   DOUBLED ROOKS BONUS                      500B)
* (FRKOPF   B   ROOK ON OPEN FILE BONUS                  500B)
* (FRKTRP   B   ROOK-KING TROPISM                        100B)
* (FRKVNP   B   ROOK ATTACK VULNERABLE PAWN BONUS        500B)
* (FRK7TH   B   ROOK ON 7TH RANK BONUS                   1600B)
* (FRMOBL   B   ROOK MOBILITY FACTOR                     100B)
* (FTMBON   B   TEMPO BONUS FOR SIDE ON MOVE             500B)
* (FTRADE   B   TRADE-DOWN BONUS FACTOR                  44B)
* (FTRDSL   B   TRADE-DOWN TUNING FACTOR                 12044B)
* (FTRPOK   B   PAWN TRADE-DOWN RELAXATION               2)
* (FTRPWN   B   PAWN TRADE-DOWN FACTOR                   10B)
* (GCLOCK   D   GAME CLOCK [MINUTES]                     0)
* (JIGMVS   D   NO. OF NON-TRIVIAL MOVES TO JIGGLE       0)
* (JIGSIZ   B   100B * MAX FRACTION TO JIGGLE            0)
* (LINEPP   D   NUMBER OF LINES PER OUTPUT PAGE          23D)
* (LOGTRA   D   TRANS/REF TABLE SIZE. SEE EXPLAIN,TOUR.  9D)
* (MDEPTH   D   MINIMUM TIME CONTROL SEARCH DEPTH        3)
* (MOVNUM   D   CURRENT MOVE NUMBER                      0)
* (MSCODE   B   LIBRARY MOVE SELECTION CODE              0B)
* (MSMASK   B   LIBRARY MOVE SELECTION MASK              0B)
* (MVRL50   D   50 MOVE RULE                             50)
* (OVRHED   D   SECONDS TO TYPE IN MOVES, ETC            20)
* (PERCNT   D   ESTIMATED PERCENTAGE OF CPU              80)
* (QADPTH   D   EXTENDED DEPTH FOR SAFE CHECKS           4)
* (QDEPTH   D   MAX QUIESCENCE EXTENSION DEPTH           2)
* (RANDOM   B   15 BIT PSEUDO-RANDOM NUMBER              0)
* (RATEMY   D   CHESS 'V' ESTIMATED USCF RATING          1550)
* (RATEPN   D   RATING DIFF. = 100B SCORE                200)
* (RATEYR   D   OPPONENTS ESTIMATED USCF RATING.         1350)
* (RULMV1   D   NUMBER OF MOVES IN FIRST TIME CONTROL    40)
* (RULMV2   D   NUMBER OF MOVES IN EXTRA TIME CONTROL    10)
* (RULTM1   D   MINUTES IN FIRST TIME CONTROL            120)
* (RULTM2   D   MINUTES IN SUBSEQUENT TIME CONTROLS      30)
* (STEPHI   D   UPPER PLY DIAGNOSTIC STEPPING LIMIT      0)
* (STEPLO   D   LOWER PLY DIAGNOSTIC STEPPING LIMIT      NPLY)
* (TGRACE   D   TIME CONTROL GRACE PERIOD IN MS          60000D)
* (TMAXOM   B   100B*MAX MULTIPLE OF TIMPM TO USE        400B)
* (TPMOKR   B   100B * RATIO USED TO COMPUTE OTMPM       63B)
* (TPMRAT   B   MINIMUM OK TIME-PER-MOVE REDUCTION       50B)
* (TPMRMX   B   100B * MAX MULT OF MTMPM TO USE          170B)
* (TPONBN   B   100B * PONDER TIME SHARING BONUS         40B)
* (TQFRST   D   FIRST TIME QUESTION MOVE NUMBER          15)
* (TQNEXT   D   SUBSEQUENT TIME QUESTION MOVE NUMBER     15)
* (TRATIC   B   NTRLO ADJUSTMENT INCREMENT               1B)
* (TRATLO   B   100B * MIN-MS-THIS-MOVE / TIMPM          32B)
* (TRATMN   B   NTRLO MINIMUM                            14B)
* (TRATMX   B   NTRLO MAXIMUM                            37B)
* (TRATOL   B   TIMPM FLUCTUATION TOLERANCE              0)
* (TRCLVL   D   LAST TRACING DEPTH + 1                   99D)
* (VALUEX   B   VALUES OF PIECES [EMPTY SQ = 0]          0B)
* (VALUEP   B   VALUE OF PAWN                            100B)
* (VALUER   B   VALUE OF ROOK                            500B)
* (VALUEN   B   VALUE OF KNIGHT                          321B)
* (VALUEB   B   VALUE OF BISHOP                          334B)
* (VALUEQ   B   VALUE OF QUEEN                           1130B)
* (VALUEK   B   VALUE OF KING                            10000B)
* (WODREL   B   MINIMUM MATERIAL SCORE RELIABILITY       60B)

/       LOAD,EX

* (LOAD LFN)
* (LOAD  [NO ABBREVIATION] COPIES IN A QUICK LOAD VERSION OF THE)
* (OPENINGS LIBRARY FROM  LFN .  IF THE PARAMETER IS OMITTED, THE)
* (FILE  OPENING  IS USED.  IF NECESSARY, THE PROGRAM WILL ATTACH)
* (A PERMANENT FILE WITH THE SPECIFIED NAME.  LOAD WORKS ONLY)
* (AT NORTHWESTERN AND ON KRONOS.  SEE  EXPLAIN,DUMP.)

/ MS,EX

* (MSG COMMENT)
* (MSG  [ABBREVIATION M] HAS NO EFFECT OTHER THAN TO INSERT)
* (INTO THE GAME SCORE, AS A COMMENT, THE ENTIRE TEXT OF)
* (THE MSG COMMAND UP TO A TERMINATING SEMICOLON OR END-OF-LINE.)
* (THE COMMAND IS ALSO ECHOED TO THE OUTPUT AND HARDCPY FILES)
* (IF APPROPRIATE.  EXAMPLE:  MSG, THIS GAME IS WEIRD.)

/ MU,EX

* (MULTI,NUMBER)
* (MULTI  [ABBREVIATION MU] PERMITS CONNECTING ADDITIONAL)
* (TERMINALS TO THE PROGRAM WHEN RUNNING UNDER THE NORTHWESTERN)
* (UNIVERSITY ONLINE TIME-SHARING SYSTEM.  THESE TERMINALS ARE)
* ("SLAVE" TERMINALS AT WHICH THE GAME MAY BE MONITORED.)
* (ALL TEXT THAT IS SENT TO THE HARDCPY FILE IS ALSO PRINTED AT)
* (THE SLAVE TERMINALS, BUT NO COMMANDS MAY BE TYPED AT THE)
* (SLAVE TERMINALS.  THE MULTI COMMAND HAS ONE PARAMETER:)
* (THE NUMBER [NOT TO EXCEED 4] OF SLAVE TERMINALS THAT MAY BE)
* (HOOKED UP TO THE MASTER JOB.  THE OPERATOR AT THE MASTER)
* (TERMINAL SHOULD TYPE %MULTI WHEN THE PROGRAM NEXT READS THE)
* (TERMINAL.  AFTER THE %MULTI, OPERATORS OF PROSPECTIVE MONITOR)
* (TERMINALS MAY TYPE %HOOKUP,JOBNAME , WHERE JOBNAME IS THE NAME)
* (OF THE MASTER JOB.  WHEN THE NUMBER OF TERMINALS SPECIFIED)
* (ON THE MULTI COMMAND ARE HOOKED UP, ADDITIONAL HOOKUPS WILL)
* (BE IGNORED.)

/ NA,EX

* (NAME,USER)
* (NAME  [ABBREVIATION NA] SETS THE NAME OF CHESS 'V' S OPPONENT.)
* (THIS NAME, WHICH IS THE PARAMETER TO THE NAME COMMAND AND MAY)
* (BE UP TO 10 CHARACTERS IN LENGTH, APPEARS ON THE GAME SCORE)
* (IN PLACE OF THE DEFAULT NAME, WHICH IS  "CHALLENGER".)
* (EXAMPLE:  NAME,H.BERLINER)

/ NU,EX

* (NULL)
* (NULL  [ABBREVIATION NU] CHANGES THE SIDE TO MOVE IN THE)
* (CURRENT BOARD POSITION, AS IF A "NULL" MOVE HAD BEEN MADE. THE)
* (COMMAND IS NOT USUALLY USED IN A REGULAR GAME, BUT IS USEFUL)
* (AS AN AID TO SETTING UP DESIRED BOARD POSITIONS FOR COMPUTER)
* (ANALYSIS.)

/ PA,EX

* (PARAMETERS [ABBREVIATION PA] PRINTS OUT THE VALUE OF THE)
* (SINGLE LET OR SWITCH VARIABLE LISTED ON THE TYPIN.)
* (IF NONE IS SPECIFIED, VALUES OF ALL THE VARIABLES ARE PRINTED.)
* (THE CHARACTER  *  WILL MATCH ANY CHARACTER.)
* (EXAMPLE:   PA,FDEPTH  PRINTS THE VALUE OF FDEPTH.)
* (           PA,F**TRP  PRINTS ALL OF THE TROPISM PARAMETERS.)
* (           PA,RULE    PRINTS THE GAME TIME-CONTROL PARAMETERS.)
* (           PA         PRINTS VALUES OF ALL LETS AND SWITCHES.)

/ PI,EX

* (PINFO,KEY)
* (PINFO  [ABBREVIATION PI] PRINTS OUT CERTAIN KINDS OF DEBUGGING)
* (INFORMATION, DEPENDING ON THE PARAMETER TO THE COMMAND:)
* ()
* (PI SU   PRINTS OUT SUMMARY DATA PERTAINING TO THE LAST MOVE)
* (MADE BY THE PROGRAM.  THE SAME DATA IS PRINTED AFTER EACH)
* (MOVE AUTOMATICALLY IF THE SU SWITCH IS ON.  INCLUDED ARE:)
* (MOVE NUMBER, PROGRAM"S ELAPSED TIME IN MINUTES, NUMBER OF)
* (NODES SEARCHED, COMPUTATION TIME IN CPU SECONDS, DEPTH OF)
* (FINAL ITERATION, RATE OF COMPUTATION IN MICROSECS/NODE,)
* (AND ESTIMATED CPU TIME REMAINING PER MOVE.)
* (ALSO THE VALUE OF THE RANDOM SEED RANDOM.)
* (SEE:  EXPLAIN,SWITCH,SU.)
* ()
* (PI NC   PRINTS OUT NODE COUNT AND OTHER DATA, PERTAINING)
* (TO LAST MOVE, WHICH IS ALSO GIVEN AUTOMATICALLY VIA THE)
* (NC SWITCH:  NUMBER OF NODES PER PLY AND PER DEPTH ITERATION,)
* (CPUSEC/MOVE EQUILIBRIUM GOAL, NEXT ITERATION RELUCTANCE)
* (RATIO, NUMBER OF NODES SUBMITTED FOR ENDPOINT SCORING, THOSE)
* (SCORED BY REGULAR SCORER, THOSE SCORED BY MOP-UP SCORER,)
* (AND ALSO THE PRELIMINARY, MINIMUM, AND MAXIMUM POSITIONAL)
* (SCORES.  NOTE: NODE COUNTS PRINTED BY  PI NC  WILL EXCEED)
* (THOSE PRINTED BY  SW NC  BY THE NUMBER OF NODES SEARCHED)
* (TO GENERATE THE PLAYERS LEGAL MOVES.  THE  SW NC  COUNTS)
* (INCLUDE ONLY THOSE NODES SEARCHED TO COMPUTE THE MACHINES)
* (MOVES.  SEE:  EXPLAIN,SWITCH,NC.)
* ()
* (PI VA   PRINTS THE MOVES OF THE PREDICTED BRANCH, WHICH IS)
* (ALSO AVAILABLE AUTOMATICALLY VIA THE VA SWITCH.)
* (IF THE MOVE WAS PONDERED, THE LINE BEGINS WITH THE WORD)
* (PONDR FOLLOWED BY THE PONDER FREE TIME IN SECONDS.)
* (SEE:  EXPLAIN,SWITCH,VA.)
* ()
* (PI WE   IS INTENDED FOR USE WHEN PAUSING AT A BREAKPOINT)
* (DURING MOVE COMPUTATION.  IT PRINTS THE VALUE OF THE)
* (WEIRD WORD, WHICH CONTROLS THE ISSUING OF SUCH MESSAGES AS)
* (OH, YOU HAD THAT,  TIME SURE FLIES, ETC.)
* ()
* (PI MO   IS INTENDED FOR USE WHEN PAUSING AT A BREAKPOINT.)
* (IT PRINTS THE CONTENTS OF THE GENERATED MOVES LIST.)
* ()
* (SEE:  EXPLAIN,BKP.)

/ PR,EX

* (PRINT,WHICH)
* (PRINT  [ABBREVIATION PR] DISPLAYS THE CURRENT BOARD IN A SHORT)
* (FORM.  LETTERS A,B,C,D,E,F REPRESENT, RESPECTIVELY, THE WHITE)
* (PAWN, ROOK, KNIGHT, BISHOP, QUEEN, KING.  DIGITS 1,2,3,4,5,6)
* (ARE THE CORRESPONDING BLACK PIECES.  WHOSE SIDE IS TO MOVE IS)
* (ALSO PRINTED.  WHEN PAUSING AT A BREAKPOINT,  PRINT,N)
* (PRINTS OUT THE CURRENT LOOK-AHEAD BOARD.  SEE:  EXPLAIN,BKP.)

/ PS,EX

* (PSLIST,SQUARELIST)
* (PSLIST  [ABBREVIATION PS] IS A DEBUGGING COMMAND THAT PRINTS)
* (THE VALUE OF THE SQUARE LIST [WITHIN THE SQRLST COMMON BLOCK])
* (SPECIFIED BY NAME ON THE PSLIST COMMAND.  IT IS PRINTED AS AN)
* (8 X 8 ARRAY WITH THE WHITE SIDE ON THE BOTTOM.  SET BITS ARE)
* (DISPLAYED AS ASTERISKS, AND CLEAR BITS ARE MINUS SIGNS.)
* (IF THE SQUARE LIST IS EMPTY, THE MESSAGE:  NO BITS SET.)
* (IS PRINTED.  PSLIST IS USEFUL AT A BREAKPOINT.)
* (EXAMPLE:  PS,AWLOC  SHOWS WHERE ALL THE WHITE PIECES ARE.)

/ PU,EX

* (PURGE)
* (PURGE  [ABBREVIATION PU] ERASES ALL PIECES FROM THE BOARD, AND)
* (SETS WHITE AS THE SIDE TO MOVE.  PURGE IS USUALLY DONE PRIOR)
* (TO SETTING UP SOME DESIRED POSITION WITH A CREATE OR BOARD)
* (COMMAND.)

/ PV,EX

* (PVARIABLE,NAME)
* (PVARIABLE  [ABBREVIATION PV] IS A DEBUGGING COMMAND THAT)
* (PRINTS OUT, IN OCTAL, THE ADDRESS AND VALUE OF THE VARIABLE)
* (NAMED ON THE COMMAND.  THE VARIABLE MUST BE DEFINED IN ONE OF)
* (THE COMMON BLOCKS: SQRLST, TLET, IOCOM, OR MEMORY.)
* (EXAMPLE:  PV,ITERS)

/ PW,EX

* (PWRD,PASSWORD)
* (PWRD  [ABBREVIATION PW] SETS [FROM ITS SINGLE PARAMETER])
* (THE PASSWORD TO BE USED FOR PERMANENT FILE REQUESTS WHICH THE)
* (PROGRAM MAY MAKE IN RESPONSE TO COMMANDS SUCH AS RELIABILITY)
* (AND REINCARNATE.  SEE ALSO  EXPLAIN,ID.)

/ REI,EX

* (REINCARNATE)
* (REINCARNATE  [ABBREVIATION REI] RECOVERS A PREVIOUSLY)
* (CHECKPOINTED STATE OF THE PROGRAM.  THE STATE OF THE PROGRAM)
* ([INCLUDING THE BOARD, ALL PARAMETER SETTINGS, TIME CONTROL)
* (INFORMATION, ETC.] IS SAVED ONCE PER MOVE ON PERMANENT FILES)
* (IF RELIABILITY IS ON.  THE KEY TO THE REINCARNATION PROCESS)
* (IS THE PF CHESSRELFILE, WHICH SHOULD BE ATTACHED AS LFN)
* (RELFILE [WITH FULL PERMISSIONS] BEFORE THE PROGRAM IS BROUGHT)
* (BACK UP.  WHEN REINCARNATE IS TYPED, RELFILE IS READ TO)
* (RESTORE THE CONTENTS OF THE FIELD LENGTH TO THE STATE OF THE)
* (CHECKPOINT.  THEN THE CHECKPOINTED STATE OF GAMFILE [GAME)
```

```
* (SCORE FILE] AND LIBRARY ARE RESTORED.  PLAY MAY THEN RESUME)
* (FROM THE POINT OF THE CHECKPOINT AS IF NO INTERRUPTION HAD)
* (OCCURED.)
* ()
* (ON KRONOS TYPE SYSTEMS, EACH USER NUMBER HAS ITS OWN)
* (CHESS RELIABLITY FILES, SO THE PROGRAM AUTOMATICALLY)
* (ATTACHES RELFILE.  SEE EXPLAIN,RELIABLITY.)

/ REL,EX

* (RELIABILITY)
* (RELIABILITY  [ABBREVIATION REL] SETS A MODE IN WHICH THE STATE)
* (OF THE PROGRAM IS SAVED ON PERMANENT FILES ONCE PER MOVE,)
* (SO THAT IN THE EVENT OF A SYSTEM CRASH OR OTHER INTERRUPTION)
* (PLAY MAY BE QUICKLY RESUMED AFTER THE SYSTEM IS BROUGHT BACK)
* (UP.  THIS MODE IS VERY USEFUL WHEN PLAYING IN TOURNAMENTS)
* ([AND IN FACT IS AUTOMATICALLY SET BY THE TOURNAMENT COMMAND])
* (SINCE IT SAVES TIME-CONSUMING AND ERROR-PRONE RESETTING)
* (OF THE BOARD, PARAMETER SETTINGS, TIME-CONTROL INFORMATION,)
* (ETC.  AFTER EACH TIME THE PROGRAM MAKES A MOVE OR EXECUTES)
* (A COMMAND, THREE PERMANENT FILES ARE UPDATED:)
* (1. CHESSRELFILE, WHICH SAVES THE CURRENT CONTENTS OF MEMORY,)
* (2. CHESSGAMFILE, WHICH SAVES THE CURRENT GAMESCORE, AND)
* (3. CHESSLIBFILE, WHICH SAVES THE CURRENT STATE OF THE)
* (POSITIONS LIBRARY.)
* (IT IS POSSIBLE THAT SOME OTHER COPY OF THE PROGRAM IS ALSO)
* (EXECUTING IN RELIABILITY MODE, SO THAT BOTH PROGRAMS WILL)
* (ATTEMPT TO WRITE PF"S OF THE SAME NAME.  AN ATTEMPT TO)
* (CATALOG CHESSRELFILE WHEN ONE ALREADY EXISTS WILL)
* (RESULT IN A FILE CATALOGUED WITH AN ALTERED NAME)
* (SUCH AS 7CHESSRELFILE.  THAT IS WHY THE REINCARNATE)
* (COMMAND REQUIRES THAT RELFILE BE PRE-ATTACHED BY THE USER,)
* (SO THAT THE CORRECT PERMANENT FILE NAME MAY BE SPECIFIED.)
* (THE ACTUAL CATALOGUED NAMES OF CHESSGAMFILE AND)
* (CHESSLIBFILE ARE SAVED ON CHESSRELFILE, SO THAT THEY)
* (WILL BE PROPERLY RESTORED WITHOUT SPECIAL INSTRUCTIONS)
* (FROM THE USER.)
* ()
* (ON KRONOS TYPE OPERATING SYSTEMS, THE PERMANENT FILE)
* (NAMES ARE THE SAME AS THE LOCAL FILE NAMES.  RATHER THAN)
* (RENAMING THE PERMANENT FILE NAMES, PREVIOUSLY EXISTING)
* (FILES OF THE SAME NAME ARE PURGED.  SEE EXPLAIN,REINCARNATE.)

/ RES,EX

* (RESIGN)
* (RESIGN  [ABBREVIATION RES] TERMINATES THE GAME IN PROGRESS AND)
* (SETS UP THE PIECES FOR A NEW ONE.  IT HAS THE SAME EFFECT AS)
* (THE INITIALIZE COMMAND.  SEE  EXPLAIN,INITIALIZE.)

/ RET,EX

* (RETURN,LFN)
* (RETURN  [ABBREVIATION RET] RETURNS THE SINGLE LOCAL FILE)
* (SPECIFIED ON THE COMMAND.  EXAMPLE:  RET,LGO  HAS)
* (THE SAME EFFECT AS THE SCOPE CONTROL CARD:  RETURN,LGO.)

/ REW,EX

* (REWIND,LFN)
* (REWIND  [ABBREVIATION REW] REWINDS THE SINGLE LOCAL FILE)
* (SPECIFIED ON THE COMMAND.  EXAMPLE:  REW,LGO  HAS)
* (THE SAME EFFECT AS THE SCOPE CONTROL CARD:  REWIND,LGO.)

/ SA,EX

* (SAVE,NAME)
* (SAVE  [ABBREVIATION SA] SAVES THE CURRENT BOARD POSITION [WITH)
* (CASTLING AND ENPASSANT STATUSES AND WHOSE TURN IT IS TO MOVE)
* (AND THE CURRENT MOVE NUMBER FROM THE MOVNUM LET])
* (IN THE POSITIONS LIBRARY UNDER THE NAME SPECIFIED ON THE)
* (COMMAND.  THE NAME MUST BE 1 TO 7 ALPHA-NUMERIC CHARACTERS.)
* (IF MORE THAN 7 ARE GIVEN, THE EXCESS CHARACTERS ARE IGNORED)
* (WITHOUT COMMENT.  IF THE NAME IS THE SAME AS THAT OF ONE USED)
* (ON A PREVIOUS SAVE COMMAND, THE NEW POSITION REPLACES THE)
* (OLD ONE.  SAVED POSITIONS MAY BE SETUP LATER IN THE SAME)
* (SESSION OR WRITTEN TO A FILE TO BE USED IN LATER SESSIONS.)
* (EXAMPLE:  SAVE,POS  SAVES CURRENT BOARD UNDER NAME: POS)
* (NOTE: THE NUMBER OF MOVES SINCE THE LAST IRREVERSIBLE MOVE)
* (IS NOT SAVED IN THE LIBRARY.)
* (SEE:  EXPLAIN,SETUP  AND  EXPLAIN,STRIP)
* (AND  EXPLAIN,USE  AND  EXPLAIN,SWITCH,RECORD.)

/ SE,EX

* (SETUP,NAME)
* (SETUP  [ABBREVIATION SE] SETS UP A POSITION PREVIOUSLY SAVED)
* (IN THE LIBRARY FILE.  THE BOARD PLUS CASTLE AND ENPASSANT)
* (STATUSES, WHOSE TURN IT IS TO MOVE, AND THE MOVE NUMBER,)
* (WHICH GOES INTO THE MOVNUM LET, ARE RESTORED.  THE NUMBER OF)
* (MOVES SINCE THE LAST IRREVERSIBLE MOVE IS NOT RESTORED.  THE)
* (NAME OF THE POSITION MUST BE SPECIFIED ON THE SETUP)
* (COMMAND.  IF MORE THAN 7 ALPHANUMERIC CHARACTERS ARE GIVEN,)
* (THE EXCESS ARE IGNORED.  THE MESSAGE:  POSITION NOT FOUND.)
* (IS PRINTED IF THE LIBRARY CONTAINS NO POSITION OF THE GIVEN)
* (NAME.  EXAMPLE:  SETUP,POS.  SEE ALSO:)
* (EXPLAIN,SAVE  AND  EXPLAIN,STRIP  AND  EXPLAIN,USE)
* (AND  EXPLAIN,SWITCH,RECORD.)

/ SP,EX

* (SPEED,N)
* (SPEED,N  [ABBREVIATION SP] SWITCHES ON TIMING AND SETS)
* (LET VARIABLES TO PLAY A FAST GAME AT AN AVERAGE RATE OF N CPU)
* (SECONDS OF MACHINE THINK TIME PER MOVE.  THE SPEED)
* (COMMAND IS ESSENTIALLY EQUIVALENT TO THE FOLLOWING:)
* ()
* (RELIABILITY;HARDCOPY;SW SU ON;SW RE ON;SW AD ON;SW EC OFF;)
* (LET RULTM1 N;LET RULTM2 N;LET EXTRAT 0;LET OVRHED 0;)
* (LET PERCNT 100;LET RATEMY 1200;LET RATEYR 1200;LET RULMV1 60;)
* (LET RULMV2 60;LET TQFIRST 9999;LET TQNEXT 99999;LET MDEPTH 1;)
* (LET COPIES 2;LET LOGTRA 15;LET JIGMUS 9999;LET JIGSIZ 20B)
* ([THE LETS:  RULTM1,RULTM2,MDEPTH,OVRHED,PERCNT,EXTRAT,)
* (RATEMY,RATEYR,RULMV1,RULMV2,TQFRST, AND TQNEXT  MUST BE)
* (PROPERLY RESET BEFORE A SUBSEQUENT TOURNAMENT COMMAND IS)
* (ISSUED.]  SEE:  EXPLAIN,SWITCH,TIMING ; EXPLAIN,TOURNAMENT.)

/ STA,EX

* (STATUS,LIST)
* (STATUS  [ABBREVIATION STA] CHANGES SUCH INFORMATION AS)
* (WHOSE TURN IT IS TO MOVE, CASTLING AND ENPASSANT FLAGS, AND)
* (HOW MANY MOVES HAVE GONE BY SINCE THE LAST IRREVERSIBLE [PAWN)
* (MOVE OR PIECE CAPTURE] MOVE.  THIS COMMAND IS USUALLY USED)
* (IN CONJUNCTION WITH THE BOARD AND CREATE COMMANDS TO SET)
* (UP A DESIRED POSITION.  PARAMETERS TO THE STATUS COMMAND)
* (SPECIFY THE FLAGS TO BE CHANGED AND THEIR NEW VALUES.)
* ()
* (PARAMETER   MEANING)
* ()
* (CL          CLEAR ALL STATUSES  [INVALIDATE CASTLING AND)
* (            ENPASSANT CAPTURES, SET NO MOVES SINCE LAST)
* (            IRREVERSIBLE, AND SET WHITE TO MOVE.  THIS)
* (            IS THE STATUS SET BY THE PURGE COMMAND.])
* (L           SUBSEQUENT OOO, OO, EP, AND M PARAMETERS APPLY)
* (            TO THE WHITE PIECES.)
* (D           SUBSEQUENT OOO, OO, EP, AND M PARAMETERS APPLY)
* (            TO THE BLACK PIECES.)
* (OOO         QUEEN SIDE CASTLING IS LEGAL.)
* (OO          KING SIDE CASTLING IS LEGAL.)
* (M           THE SIDE LAST SPECIFIED IS TO MOVE.)
* (EP          THE SIDE LAST SPECIFIED HAS A PAWN WHICH HAS)
* (            JUST MOVED 2 SQUARES AND SO IS ELIGIBLE TO BE)
* (            CAPTURED ENPASSANT.  THE FILE OF THE PAWN IS)
* (            GIVEN MNEMONICALLY BY THE NEXT PARAMETER AFTER)
* (            THE EP, AS ONE OF KR,QR,KN,QN,KB,QB,K, OR Q.)
* (C           SET THE NUMBER OF MOVES SINCE LAST IRREVERSIBLE)
* (            MOVE.  THIS PARAMETER IS OF IMPORTANCE IN RELATION)
* (            TO THE SO-CALLED 50 MOVE RULE.  THE NUMBER, WHICH)
* (            MUST BE LESS THAN 50, IS GIVEN AS THE NEXT)
* (            PARAMETER.  NOTE THAT THIS NUMBER REPRESENTS WHOLE)
* (            MOVES -- THERE IS NO WAY TO SPECIFY A HALF MOVE)
* (            WITH THE STATUS COMMAND.  HOWEVER, THE NULL)
* (            MOVE [SEE:  EXPLAIN,NULL.] IS A REVERSIBLE MOVE.)
* (N           SAME AS C.)
* ()
* (EXAMPLE:  STA L OO EP QR D OOO M C 25     SETS WHITE KING SIDE)
* (AND BLACK QUEEN SIDE CASTLING LEGAL, SHOWS WHITE TO HAVE A)
* (PAWN ON THE QUEEN ROOK FILE ELIGIBLE TO BE CAPTURED EN-)
* (PASSANT, SETS BLACK TO MOVE, AND SHOWS THAT 25 MOVES HAVE)
* (BEEN MADE SINCE THE LAST CAPTURE OR PAWN MOVE.)
* (SEE:  EXPLAIN,BOARD  AND  EXPLAIN,CREATE  AND  EXPLAIN,PURGE.)

/ STRI,EX

* (STRIP,LEARN,SETUP)
* (STRIP  [ABBREVIATION STR] COPIES THE CONTENTS OF THE POSITIONS)
* (LIBRARY OUT TO UP TO TWO SEQUENTIAL LOCAL FILES, WHOSE NAMES)
* (ARE SPECIFIED AS PARAMETERS TO THE STRIP COMMAND.  FILE NAMES)
* (MUST, OF COURSE, BE NO MORE THAN 7 CHARACTERS -- EXCESS)
* (CHARACTERS ARE IGNORED WITHOUT COMMENT.  THE FIRST FILE)
* (SPECIFIED RECEIVES ALL POSITIONS CONTAINING ROTE LEARNING)
* ([OPENINGS LIBRARY] INFORMATION.  THE SECOND FILE RECEIVES)
* (ALL POSITIONS ENTERED BY NAME BY EITHER THE SAVE COMMAND)
* (OR AUTOMATICALLY WHILE THE RECORD SWITCH WAS ON.  IF NO)
* (FILES ARE SPECIFIED, THE STRIP COMMAND DOES NOTHING.  IF)
* (ONE FILE IS SPECIFIED, THAT FILE RECEIVES THE ROTE LEARNING)
* (POSITIONS.  UPON COMPLETION OF THE STRIP COMMAND, THE PROGRAM)
* (REPORTS THE NUMBER OF POSITIONS WRITTEN TO EACH FILE.  A)
* (SEQUENTIAL STRIP FILE HAS THE FORMAT OF AN ABSOLUTE OVERLAY)
* (CALLED CHE$$99 WITH LEVEL [9,9], SO IT CAN BE EDITLIBED)
* (ONTO THE SYSTEM LIBRARY.)
* (EXAMPLE:  STRIP,A,B  WRITES ALL ROTE LEARNING POSITIONS TO)
* (FILE A AND ALL NAMED [SETUP] POSITIONS TO FILE B.)
* (SEE ALSO:  EXPLAIN,SAVE ; EXPLAIN,SETUP ; EXPLAIN,USE ;)
* (EXPLAIN,SW,RECORD ; EXPLAIN,SW,LW ; EXPLAIN,SW,LB.)

       USE    SW
/ SW,SP
       USE    *

/ SW,EX

* (SWITCH,VARIABLE,ONOFF)
* (SWITCH  [ABBREVIATION SW] SETS THE VALUE OF ANY ONE OF SOME 24)
* (TWO-VALUED VARIABLES CALLED SWITCHES.  THESE VARIABLES CONTROL)
* (VARIOUS ASPECTS OF THE PROGRAM-USER AND PROGRAM-SYSTEM)
* (INTERFACES.  SOME OF THEM ARE ALTERED IMPLICITLY VIA THE)
* (EXECUTION OF OTHER COMMANDS, SUCH AS TOURNAMENT, WHICH AFFECTS)
* (SWITCHES EC,RE,SU,NC,VA,TI, AND AD.  THE DESIRED SWITCH AND)
* (ITS NEW VALUE ARE PARAMETERS TO THE SWITCH COMMAND.  ONLY THE)
* (2-LETTER MNEMONIC SWITCH NAME NEED BE GIVEN; EXCESS CHARACTERS)
* (ARE IGNORED.  THE VALUE IS EITHER  ON  OR  OFF .  IT MAY BE)
* (OMITTED, IN WHICH CASE THE CURRENT VALUE IS TOGGLED [CHANGED)
* (TO ON IF OFF AND OFF IF ON].  SOME OF THE SWITCHES ARE)
* (HANDLED A LITTLE DIFFERENTLY FROM THE ABOVE DESCRIPTION.  THE)
* (I/O MODE SWITCHES:  DI,CA,DS [AND OL,IM ON SYSTEMS ON WHICH)
* (THEY EXIST] ARE INTER-RELATED SUCH THAT TURNING ANY OF THEM ON)
* (TURNS THE OTHERS OFF.  FOR THESE SWITCHES THE VALUE [ON OR)
* (OFF] PARAMETER IS IGNORED -- ON IS ALWAYS ASSUMED.)
* (EXAMPLES OF SWITCH COMMANDS:  SW CARDS ; SWITCH,VA,OFF ;)
* (SWITCH TIMING ON ; SW RE .)
* ()
* (FOLLOWING IS A LIST OF SWITCHES, THEIR DEFAULT VALUES, THEIR)
* (NAMES AS THEY APPEAR IN A PARAMETERS PRINTOUT, AND BRIEF)
* (EXPLANATIONS OF THEIR FUNCTIONS.  TO GET A FULLER DESCRIPTION)
* (OF ANY ONE SWITCH, TYPE  EXPLAIN,SWITCH,NAME  , WHERE NAME IS)
* (THE 2-LETTER MNEMONIC OF THE SWITCH.)
* (EXAMPLE:  EXPLAIN,SWITCH,VA  [EXPLAIN SW VA  IS ALSO OK].)
* ()
* (SWITCH VALUE  PA NAME      FUNCTION)
* ()
* (DI     OFF    DISPLAY      COMMANDS COME FROM DUD)
* (EC     ON     ECHO         ECHO BACK TYPE-INS)
* (NO     OFF    NO REPLY     DONT REPLY TO YOUR MOVE WITH A MOVE)
* (36     OFF    3.6 SYNTAX   GENERATE PRE-CHESS4.0 MOVE NOTATION)
* (CA     ON     CARDS        COMMANDS COME FROM A FILE)
* (DS     OFF    DSD          COMMANDS COME FROM CFO TYPE-INS)
* (DE     OFF    DEBUG        PAUSE BETWEEN SEARCH ITERATIONS)
* (TR     OFF    TRACE        PRINT OUT THE WHOLE TREE)
* (LW     OFF    LEARN WHITE  WHITE MOVES GO INTO LIBRARY)
* (LB     OFF    LEARN BLACK  BLACK MOVES GO INTO LIBRARY)
* (EX     ON     EXPERIENCE   CONSULT POSITIONS LIBRARY)
* (RE     OFF    RECORD       SAVE POSITION AFTER EACH MOVE)
* (WE     ON     WEIRD WORD   MAKE SNIDE REMARKS)
* (SU     OFF    SUMMARY      DO  PINFO SU  AFTER MOVING)
* (NC     OFF    NODE COUNTS  DO  PINFO NC  AFTER MOVING)
* (VA     OFF    VARIATION    DO  PINFO VA  AFTER MOVING)
* (LI     OFF    LIST         PRINT BOARD AFTER MOVING)
* (TI     OFF    TIMING       DO AUTOMATIC TIME CONTROL)
* (AD     OFF    A-DISPLAY    WRITE MOVES TO DAYFILE)
* (BE     ON     BELL         RING BELL AFTER MOVING)
* (DA     OFF    DBG ALTER    AUTO-ALTER AT BREAKPOINT PAUSES)
* (MA     OFF    MVE ALTER    AUTO-ALTER AT EACH MOVE)
* (OL     OFF    ONE-LINE     COMMANDS COME FROM ONE-LINE SYSTEM)
* (IM     OFF    IMLAC        MOVES COME FROM LIGHT PEN)
* (RA     OFF    RANDOM       GET RANDOM SEED BEFORE EACH MOVE)
* (PO     OFF    PONDER       THINK ON OPPONENTS TIME)
* (AL     OFF    ALGEBRAIC    GENERATE ALGEBRAIC MOVE NOTATION)
* (CB     OFF    CBD         ELECTRONIC CHESS BOARD)

/ TO,EX

* (TOURNAMENT)
```

```
* (TOURNAMENT  [ABBREVIATION TO] SWITCHES TIMING ON AND SETS)
* (OTHER OPTIONS FOR PLAYING IN A REGULARLY TIMED CHESS)
* (TOURNAMENT, SUCH AS THE ACM AND IFIPS COMPUTER EVENTS OR USCF)
* (RATED HUMAN TOURNAMENTS.  THE TOURNAMENT COMMAND IS)
* (ESSENTIALLY EQUIVALENT TO THE FOLLOWING:)
* ()
* (RELIABILITY;HARDCOPY;SW NC ON;SW SU ON;SW VA ON;SW RE ON;)
* (SW AD ON;SW TI ON;SW EC OFF;LET,COPIES=10;LET LOGTRA=15.)
* (SW PO ON.)
* ()
* (NOTE: THE LARGER LOGTRA IS, THE LARGER THE ECS)
* (TRANSPOSITIONS/REFUTATIONS HASH TABLE IS, AND THE MORE)
* (EFFICIENTLY THE PROGRAM RUNS.  LENGTH OF THE TABLE =)
* (5*[2**LOGTRA].  THE TOURNAMENT SETTING REQUIRES 120000B)
* (ECS [OR CM, IF ECS IS OFF], WHICH MAY BE TOO BIG FOR INSTAL-)
* (LATIONS WITH LITTLE OR NO ECS.  LOGTRA MAY BE SET TO ANY VALUE)
* (FROM 8 TO 16 WITH THE LET COMMAND.  SEE EXPLAIN,LET.  THE)
* (GREATER THE NUMBER OF SEARCH ITERATIONS, THE GREATER IS)
* (THE UTILITY OF THE TABLE.  ALSO, THE MORE CPU TIME THAT IS)
* (SPENT PER MOVE, THE GREATER IS THE ADVANTAGE OF A LARGE TABLE.)
* (SEE:  EXPLAIN,RELIABILITY ; EXPLAIN,HARDCOPY ; EXPLAIN,SWITCH)
* (EXPLAIN,SW,NC ; EXPLAIN,SW,SU ; EXPLAIN,SW,VA ;)
* (EXPLAIN,SW,RE ; EXPLAIN,SW,AD ; EXPLAIN,SW,TI;)
* (EXPLAIN,SW,EC.  FOR A FASTER, INFORMAL GAME USING TIMING)
* (CONTROL, SEE:  EXPLAIN,BLITZ.)

/ US,EX

* (USE,LFN)
* (USE  [ABBREVIATION US] MERGES ONE RECORD FROM A FILE SPECIFIED)
* (ON THE USE COMMAND [E.G. USE,POSFILE ] INTO THE POSITIONS)
* (LIBRARY.  THE RECORD MUST BE IN THE SAME FORMAT THAT IS)
* (WRITTEN BY THE STRIP COMMAND.  USE DOES NOT REWIND THE)
* (FILE; POSITIONING IS UP TO THE PLAYER.  WHEN THE USE)
* (COMMAND IS DONE, 3 MESSAGES ARE PRINTED SUMMARIZING THE)
* (RESULTS:)
* (           IIIIII POSITIONS IGNORED.)
* (           JJJJJJ POSITIONS REPLACED.)
* (           KKKKKK POSITIONS ADDED.)
* ()
* (FOR LEARN POSITIONS:)
* (     IIIIII IS THE  NUMBER OF POSITIONS ON THE USED FILE)
* (THAT WERE DISCARDED BECAUSE THEY EXACTLY MATCHED EXISTING)
* (ENTRIES IN THE CURRENT LIBRARY.)
* (     JJJJJJ IS THE NUMBER THAT REPLACED EXISTING ENTRIES)
* (THAT WERE IDENTICAL IN BOARD POSITION AND MSCODE, BUT)
* (DID NOT HAVE THE SAME LEARNED MOVE.)
* (     KKKKKK IS THE NUMBER THAT WERE APPENDED TO THE END)
* (OF THE CURRENT LIBRARY BECAUSE EITHER THE BOARD POSITION)
* (OR THE MSCODE DIFFERED FROM ANY EXISTING ENTRIES.)
* ()
* (FOR NAMED POSITIONS:)
* (     IIIIII MEANS THE SAME AS FOR LEARNED POSITIONS.)
* (     JJJJJJ IS THE NUMBER OF POSITIONS ON THE USED FILE)
* (THAT REPLACED EXISTING ENTRIES IN THE LIBRARY WHICH HAD)
* (THE SAME NAMES BUT DIFFERENT BOARD POSITIONS.)
* (     KKKKKK IS THE NUMBER THAT WERE APPENDED TO THE LIBRARY)
* (BECAUSE THEIR NAMES DID NOT MATCH THOSE OF EXISTING ENTRIES.)
* ()
* (NOTE:  WHEN THE CHESS PROGRAM IS BROUGHT UP, IT ATTEMPTS TO)
* (LOAD AND USE A RECORD FROM THE SYSTEM LIBRARY CALLED CHE$$99,)
* (UNLESS THERE IS A 3RD PARAMETER ON THE CHESS CONTROL CARD.  A)
* (RECORD CREATED BY STRIP MAY BE EDITLIBED ONTO THE SYSTEM TO BE)
* (SO USED, FOR EXAMPLE, AS A STANDARD OPENINGS LIBRARY.)
* (SEE:  EXPLAIN,STRIP.)

/ WH,EX

* (WHAT)
* (WHAT  [ABBREVIATION WH] CAUSES THE PROGRAM TO REPEAT THE LAST)
* (PROMPTING MESSAGE THAT IT SENT TO THE OUTPUT FILE, SUCH AS:)
* ("MY MOVE P-K4."  OR  "ENTER MOVE OR TYPE GO." , IN CASE THE)
* (USER MISSED IT.  LINES SUCH AS MIGHT BE PRINTED BECAUSE THE)
* (SU SWITCH IS ON OR AS A RESULT OF, SAY, THE PV COMMAND ARE)
* (NOT COUNTED AS PROMPTING MESSAGES.)

/ X,EX

* (XEQ,FL,CONTROLCARD.)
* (NOTE: XEQ WORKS ONLY AT NORTHWESTERN UNIVERSITY.)
* (XEQ  [ABBREVIATIONS X OR $] EXECUTES A SCOPE CONTROL CARD USING)
* (THE ESP FACILITY, AND THEN RETURNS CONTROL TO CHESS.  AN OCTAL)
* (FIELD LENGTH MAY BE SPECIFIED AS THE FIRST PARAMETER, IN WHICH)
* (CASE THE REMAINDER OF THE COMMAND [AFTER A DELIMITER] IS TAKEN)
* (AS THE CONTROL CARD IMAGE.  THE FIELD LENGTH MAY BE OMITTED;)
* (THEN THE CURRENT FIELD LENGTH IS USED.  EXAMPLES:)
* (      X.COPYBR,A,B.)
* (      $[50000]COMPASS,I,G=CHESTXT,L8.)
* (AFTER CONTROL RETURNS TO CHESS, A MESSAGE IS PRINTED:)
* ("SUCCESSFUL EXECUTION." MEANS NO ABORT.  OTHER MESSAGES)
* (DESCRIBE THE VALUE OF THE ERROR FLAG THAT WAS SET, SUCH AS:)
* ("CPU ABORT."  OR  "ARITHMETIC ERROR.")
/ $,EX

/ DI,SW

* (SWITCH,DISPLAY)
* (DI IS AN I/O MODE SWITCH; IT IS TURNED ON BY  SW DI  AND IS)
* (TURNED OFF BY TURNING ON ANY OF THE OTHER I/O MODES.  WHEN)
* (DI IS ON, THE PP PROGRAM DUD [DYNAMIC USER DISPLAY] IS)
* (CALLED UP AND REQUESTS USE OF THE CENTRAL CONSOLE DISPLAY AND)
* (KEYBOARD.  WHEN THE CONSOLE EQUIPMENT IS ASSIGNED, DUD)
* (DISPLAYS THE CHESS BOARD ON THE LEFT-HAND SCREEN.  MOVES)
* (MAY THEN BE TYPED AT THE KEYBOARD; RESPONSES WILL BE)
* (DISPLAYED ON THE SCREEN, AS WELL AS WRITTEN TO THE CHESS)
* (OUTPUT FILE.  WHILE DUD)
* (IS IN CONTROL, ASTERISK [ * ] MAY BE TYPED TO)
* (TEMPORARILY RELENQUISH THE CONSOLE TO THE OPERATING SYSTEM;)
* (WHEN TYPED AGAIN UNDER DSD, * RETURNS CONTROL TO DUD.  TWO)
* (OTHER COMMANDS ARE USEFUL ONLY UNDER DUD:  HOLD AND KILL.)
* (SOME COMMANDS MUST BE TYPED A LITTLE DIFFERENTLY IN DI MODE)
* (THAN IN OTHER I/O MODES.  E.G., PRINT BECOMES: PRINT BOARD.)
* (DI NORMALLY STARTS OUT OFF;  HOWEVER, IF THE NORMAL CHESS)
* (INPUT FILE IS ON AN ALLOCATABLE DEVICE AND RETURNS AN)
* (IMMEDIATE END-OF-RECORD TO THE FIRST ATTEMPT OF THE PROGRAM)
* (TO READ IT, THEN DI WILL BE AUTOMATICALLY TURNED ON.  THIS)
* (ENABLES CHESS TO BE RUN FROM THE CONSOLE AS AN OPERATOR-)
* (INITIATED JOB WITH ONLY ONE TYPE-IN AND NO PREPARED INPUT)
* (FILE.  SEE ALSO:  EXPLAIN,HOLD ; EXPLAIN,KILL ;)
* (EXPLAIN,SW,CA ; EXPLAIN,SW,DS  [ALSO EXPLAIN,SW,OL  AND/OR)
* (EXPLAIN,SW,IM  ON SYSTEMS ON WHICH OL AND/OR IM ARE DEFINED].)

/ EC,SW

* (SWITCH,ECHO,ONOFF)
* (EC CAUSES COMMANDS AND MOVES TO BE ECHOED TO THE OUTPUT)
* (FILE, SO THAT THEIR CORRECT TRANSMISSION MAY BE VERIFIED.)
* (THE TOURNAMENT TYPE-IN TURNS OFF EC TO REDUCE THE VOLUME OF)
* (DATA PRINTED AT THE TERMINAL.)

/ NO,SW

* (SWITCH,NOREPLY,ONOFF)
* (NO INHIBITS THE MACHINE"S AUTOMATIC RESPONSE TO A MOVE OF)
* (YOURS WITH A MOVE OF ITS OWN.  IF NO IS ON AND YOU TYPE A)
* (MOVE, SUCH AS P-K4, THE MACHINE RESPONDS:  YOUR MOVE P-K4.)
* (AND ENTERS THE MOVE ON THE BOARD BUT MAKES NO RESPONDING)
* (MOVE.  THE PROGRAM IS THEN READY TO ACCEPT A MOVE FOR THE)
* (OTHER SIDE.  THIS SWITCH IS USEFUL FOR ENTERING FIXED)
* (SEQUENCES OF MOVES AS MIGHT BE DONE WHEN ADDING TO)
* (THE OPENINGS LIBRARY.  THEN NO IS USED IN CONJUNCTION)
* (WITH THE SWITCHES LW AND LB.  SEE  EXPLAIN,SW,LW  AND)
* (EXPLAIN,SW,LB.)

/ 36,SW

* (SWITCH,36,ONOFF)
* (36 CAUSES GENERATION OF THE FORM OF ENGLISH DESCRIPTIVE MOVE)
* (NOTATION USED BY VERSIONS 3.6 AND EARLIER.  VERSIONS 4.0 AND)
* (BEYOND CLARIFY AMBIGUITIES DIFFERENTLY, USING, FOR EXAMPLE,)
* (N/K3*P INSTEAD OF N[K3]*P.  PAWN PROMOTION IS ALSO INDICATED)
* (DIFFERENTLY.  36 MODE FACILITATES AUTOMATIC COMMUNICATION)
* (BETWEEN TWO VERSIONS OF THE PROGRAM.  THE NEWER VERSIONS)
* (CAN UNDERSTAND THE NOTATION OF THE OLDER VERSIONS, BUT THE)
* (CONVERSE IS NOT TRUE WITHOUT 36 MODE.)

/ CA,SW

* (SWITCH,CARDS)
* (CA  IS AN I/O MODE SWITCH; IT IS TURNED ON BY  SW,CA  AND)
* (TURNED OFF BY TURNING ON ONE OF THE OTHER I/O MODES.  CA)
* (IS INITIALLY ON WHEN THE PROGRAM IS BROUGHT UP.  WHEN CA)
* (IS ON, COMMANDS AND MOVES ARE READ FROM THE CHESS INPUT)
* (FILE, WHOSE DEFAULT NAME IS KEYBRD [WHICH AT NORTHWESTERN)
* (UNIVERSITY IS AUTOMATICALLY CONNECTED TO THE INTERACTIVE)
* (TERMINAL KEYBOARD].  IF THIS FILE IS ON AN ALLOCATABLE DEVICE)
* ([LIKE A DISK], AND IS INITIALLY EMPTY, THE PROGRAM SWITCHES)
* (MODES TO DI  [SEE  EXPLAIN,SW,DI.].  NON-INITIAL END-OF-RECORD)
* (AND END-OF-FILE CAUSE THE PROGRAM TO END IF ON AN ALLOCATABLE)
* (DEVICE; ELSE THEY ARE IGNORED.  THE FILE BUFFER IS ONLY 101B)
* (LONG, SO INPUT CANNOT BE READ FROM MAGNETIC TAPE.  RESPONSES)
* (FROM THE PROGRAM GO TO THE CHESS OUTPUT FILE, WHOSE DEFAULT)
* (NAME IS CONSOL [CONNECTED AT NUCC TO THE TERMINAL PRINTER OR)
* (DISPLAY].  SEE ALSO:  EXPLAIN,ALTER ; EXPLAIN,SW,DI ;)
* (AND  EXPLAIN,SW,DS.)

/ DS,SW

* (SWITCH,DSD)
* (DS  IS AN I/O MODE SWITCH.  IT IS TURNED ON BY SW DS AND)
* (TURNED OFF BY TURNING ON ONE OF THE OTHER I/O MODES.  WHEN DS)
* (IS ON, MOVES AND COMMANDS ARE READ FROM THE N.CFO [COMMENT)
* (FROM OPERATOR] TYPE-INS AT THE CENTRAL CONSOLE.  WHILE)
* (WAITING FOR INPUT, THE PROGRAM IS IN PERIODIC RECALL FLASHING)
* (ITS LAST RESPONSE ON THE B-DISPLAY.  RESPONSES ARE ALSO)
* (WRITTEN TO THE CHESS OUTPUT FILE, WHOSE DEFAULT NAME IS)
* (CONSOL.  SEE  EXPLAIN,SW,DI ; EXPLAIN,SW,CA.)

/ DE,SW

* (SWITCH,DEBUG,ONOFF)
* (DE  ACTIVATES A DEBUGGING BREAKPOINT BEFORE EACH TREE SEARCH)
* (ITERATION.  SEE:  EXPLAIN,BKP.)

/ TR,SW


* (SWITCH,TRACE,ONOFF)
* (TR  CAUSES A TRACE OF THE TREE SEARCH TO BE PRINTED.  FOR)
* (COMPACTNESS, THIS TRACE IS FORMATTED FOR FULL [130 CHARACTER])
* (WIDTH PRINTERS.  20 CHARACTERS OF INFORMATION ARE OUTPUT EACH)
* (TIME THE TREE SEARCH ROUTINE IS CALLED:  THE PLY LEVEL FROM)
* (WHICH THE MOVE IS BEING MADE [COUNTING THE BASE LEVEL AS)
* (ZERO], THE LABEL ON THE CALL TO THE TREE-SEARCHER, AND THE)
* (MOVE IN DESCRIPTIVE NOTATION.  6 MOVES ARE PACKED PER LINE,)
* (EXCEPT THAT MOVES AT THE FIRST 2 PLY LEVELS EACH BEGIN A)
* (NEW LINE AND ARE NUMBERED IN OCTAL STARTING AT ZERO.)
* (ALSO, FOR THE FIRST 2 PLIES, THE VALUES OF ALPHA AND)
* (BETAR ARE PRINTED.)

/ LW,SW

* (SWITCH,LWHITE,ONOFF)
* (LW  CAUSES EACH SUBSEQUENT WHITE MOVE TOGETHER WITH THE)
* (POSITION FROM WHICH IT WAS MADE TO BE RECORDED ON THE)
* (POSITIONS LIBRARY.  THEN LATER, IF THE EXPERIENCE SWITCH IS)
* (ON, THE PROGRAM WILL SUBSTITUTE THE RECORDED MOVES FOR ITS OWN)
* (THINKING PROCESS IN POSITIONS THAT MATCH ENTRIES IN THE)
* (LIBRARY.  THUS OPENINGS LIBRARIES MAY BE CREATED.  NOTE THAT)
* (INFORMATION ABOUT CASTLING, ENPASSANT CAPTURES, AND WHOSE TURN)
* (IT IS TO MOVE IS RECORDED, BUT NOT THE NUMBER OF MOVES SINCE)
* (THE LAST IRREVERSIBLE MOVE.  TWO LETS, MSCODE AND MSMASK,)
* (PROVIDE, IN A SOMEWHAT OBSCURE WAY, THE FACILITY TO ENTER TWO)
* (OR MORE DIFFERENT MOVES WITH THE SAME POSITION, SUCH THAT)
* (DIFFERENT OPENINGS MAY BE SELECTED EITHER RANDOMLY [BY THE)
* (PROGRAM] OR DELIBERATELY [BY THE HUMAN USER] IN DIFFERENT)
* (GAMES.  THESE LETS MUST BE SET WITH CARE DURING LIBRARY)
* (CREATION TO ACHIEVE THE DESIRED EFFECTS:  WHEN A MOVE AND)
* (POSITION ARE RECORDED, THE CURRENT VALUE OF MSCODE IS)
* (RECORDED WITH THEM.  BEFORE RECORDING THEM, THE PROGRAM)
* (SEARCHES THE LIBRARY FOR AN EXISTING ENTRY HOLDING THE SAME)
* (POSITION AND A VALUE OF MSCODE WHICH MATCHES THE CURRENT)
* (VALUE IN THOSE BIT POSITIONS SPECIFIED BY THE ONES BITS IN)
* (MSMASK [MSCODE AND MSMASK HAVE 10 SIGNIFICANT BITS].  IF SUCH)
* (AN ENTRY IS FOUND, IT IS REPLACED BY THE ENTRY TO BE RECORDED,)
* (AND THE MESSAGE:  DUPLICATE POSITION REPLACED.  IS PRINTED.)
* (IF IT IS NOT FOUND, THE NEW ENTRY IS RECORDED IN A NEW SLOT.)
* (NOTE: THE EX SWITCH SHOULD BE TURNED OFF DURING LIBRARY)
* (CREATION; OTHERWISE, EACH INI COMMAND CHANGES THE VALUE OF)
* (MSMASK TO A NEW [6-BIT] RANDOM VALUE.  FOR THE FUNCTION OF)
* (MSCODE AND MSMASK WHEN THE LIBRARY IS SEARCHED DURING PLAY,)
* (SEE: EXPLAIN,SW,EX.  FOR INFORMATION ABOUT SAVING, RESTORING,)
* (AND MERGING LIBRARIES, SEE:  EXPLAIN,STRIP ; EXPLAIN,USE.  SEE)
* (ALSO EXPLAIN,SW,LB.  HERE IS SAMPLE LIBRARY CREATION:)
* ()
* (MSG. THE FOLLOWING COMMANDS CREATE A LIBRARY SUCH THAT DURING)
* (MSG. PLAY, WITH THE EX SWITCH ON AND MSCODE SET TO ITS DEFAULT)
* (MSG. VALUE OF 00B, THE PROGRAM WILL OPEN EITHER P-K4 OR P-Q4)
* (MSG. RANDOMLY WITH EQUAL PROBABILITY.)
* (INI.)
* (SW EXP OFF.  SO MSMASK IS NOT CHANGED BY AN INI.)
* (SW LW ON. LEARN WHITE"S MOVES.)
* (SW NO ON. NO REPLY MODE.)
* (LET MSMASK 77B. SO DIFFERENT MSCODES ARE DISTINGUISHED.)
```

```
* (MSG. NOTE THAT TRADITIONALLY, ONLY 6-BIT VALUES ARE USED)
* (MSG. FOR MSCODE AND MSMASK.)
* (LET MSCODE 01B.)
* (P-K4. WILL GO UNDER CODE OF 01B.)
* (INI. BACK TO INITIAL POSITION.)
* (LET MSCODE 00B.)
* (P-Q4. WILL GO UNDER CODE OF 00B.)
* (SW EX ON.)
* (SW LW OFF)
* (SW NO OFF)
* (LET MSCODE 00B)
* (INI.)
* (MSG. NOW, DURING NORMAL PLAY, MSMASK WILL ACQUIRE A RANDOM)
* (MSG. 6-BIT VALUE [WITH 3 BITS SET].  WHEN THE LIBRARY IS)
* (MSG. SEARCHED FOR THE INITIAL POSITION, THE MSCODE OF THE)
* (MSG. FIRST ENTRY WILL MATCH 00B IF THE LOWEST BIT OF MSMASK IS)
* (MSG. CLEAR; THEN P-K4 WILL BE PLAYED. OTHERWISE, THE 2ND ENTRY)
* (MSG. WILL BE FOUND [ITS MSCODE OF 00B WILL ALWAYS MATCH], AND)
* (MSG. P-Q4 WILL BE PLAYED.  NOTE THAT USE HAS BEEN MADE OF)
* (MSG. THE FACT THAT THE ORDER OF THE POSITIONS IN THE LIBRARY)
* (MSG. IS THE SAME DURING RECORDING AND SEARCHING, AND IS)
* (MSG. PRESERVED BY STRIP AND USE.  VARIOUS MOVES CAN BE SET)
* (MSG. UP TO BE PLAYED WITH VARIOUS PROBABILITIES IF THE USER)
* (MSG. BECOMES SKILLED IN THE ART OF MANIPULATING MSCODE AND)
* (MSG. MSMASK BOTH DURING LIBRARY CREATION AND BEFORE PLAY.)
* (MSG. OF COURSE, THE USER MAY WONDER WHY THE PROGRAM)
* (MSG. AUTHORS DEVISED SUCH A WEIRD SCHEME IN THE FIRST PLACE.)

/ LB,SW

* (SWITCH,LBLACK,ONOFF)
* (LB  WORKS THE SAME AS LW EXCEPT FOR BLACK MOVES.  BOTH LW AND)
* (LB MAY BE [BUT NOT USUALLY ARE] ON TOGETHER.  SEE:)
* (EXPLAIN,SW,LW.)

/ EX,SW

* (SWITCH,EXPERIENCE,ONOFF)
* (EX  CAUSES THE POSITIONS LIBRARY TO BE SEARCHED FOR A MATCH)
* (WITH THE CURRENT POSITION EACH TIME THE PROGRAM IS TO MAKE)
* (A MOVE.  IF A MATCH IS FOUND [AND IF THE MSCODE OF THE)
* (LIBRARY ENTRY MATCHES THE CURRENT MSCODE IN THOSE BITS)
* (SPECIFIED BY MSMASK], THE RECORDED MOVE IS MADE IMMEDIATELY)
* (AND NO THINKING IS DONE BY THE PROGRAM.  WHEN EX IS ON,)
* (MSMASK IS CHANGED TO A RANDOM 6-BIT VALUE [WITH 3 BITS SET] BY)
* (EACH INI COMMAND, TO FACILITATE RANDOM SELECTION OF OPENING)
* (MOVES DURING PLAY.  WHEN EX IS OFF, GAME SCORES ARE SAVED UP)
* (BUT NOT PRINTED.  THIS FEATURE IS USEFUL DURING OPENINGS)
* (LIBRARY CREATION, WHEN IT RESULTS IN CLEANER LISTINGS.  IF)
* (EX IS TURNED BACK ON, THEN AN INI COMMAND WILL PRINT ALL THE)
* (ACCUMULATED GAME SCORES.  FOR THE DETAILS OF LIBRARY CREATION)
* (AND THE FUNCTION OF MSCODE AND MSMASK, SEE:  EXPLAIN,SW,LW.)
* (FOR HOW TO SAVE, RESTORE, AND MERGE POSITIONS LIBRARIES, SEE:)
* (EXPLAIN,STRIP ; EXPLAIN,USE.  ALSO SEE:  EXPLAIN,INI  AND)
* (EXPLAIN,SW,LB.)

/ RE,SW

* (SWITCH,RECORD,ONOFF)
* (RE  [RECORD] CAUSES THE EQUIVALENT OF A SAVE COMMAND TO BE)
* (EXECUTED AFTER EACH MACHINE MOVE [ACTUALLY, BEFORE EACH TIME)
* (THE PROGRAM READS A NEW LINE OF INPUT].  THE NAME USED IS THE)
* (CURRENT VALUE OF THE MOVNUM LET IN DECIMAL WITH LEADING ZEROES)
* (SUPPRESSED.  MOVNUM REPRESENTS THE NUMBER OF FULL MOVES MADE)
* (SO FAR IN THE GAME;  IT IS INCREMENTED AFTER EACH WHITE MOVE.)
* (RE IS TURNED ON AUTOMATICALLY BY THE TOURNAMENT TYPE-IN.  IT)
* (MAKES IT EASY TO BACK UP TO AN EARLIER POSITION IN THE GAME.)
* (SEE:  EXPLAIN,SAVE ; EXPLAIN,SETUP ; EXPLAIN,TOURNAMENT.)

/ WE,SW

* (SWITCH,WEIRD,ONOFF)
* (WE  [WEIRD WORD] ENABLES THE PROGRAM TO MAKE CERTAIN SNIDE)
* (REMARKS IN WHAT ARE HOPEFULLY [BUT AREN"T ALWAYS] APPROPRIATE)
* (TIMES DURING PLAY.  MESSAGES INCLUDE THE FOLLOWING:)
* (     BE CAREFUL.)
* (     OH, YOU HAD THAT.)
* (     TIME SURE FLIES.)
* (     THAT WAS EASY.)

/ SU,SW

* (SWITCH,SUMMARY,ONOFF)
* (SU  CAUSES SUMMARY INFORMATION TO BE PRINTED AUTOMATICALLY)
* (AFTER EACH MACHINE MOVE.  FOR ITS FORMAT, SEE:  EXPLAIN,PINFO.)

/ NC,SW

* (SWITCH,NCOUNTS,ONOFF)
* (NC  CAUSES NODE COUNT [AND OTHER] INFORMATION TO BE PRINTED)
* (AUTOMATICALLY AFTER EACH MACHINE MOVE.  FOR ITS FORMAT, SEE:)
* (EXPLAIN,PINFO.)

/ VA,SW

* (SWITCH,VARIATION,ONOFF)
* (VA  CAUSES THE MAIN EXPECTED VARIATION TO BE PRINTED)
* (AUTOMATICALLY AFTER EACH MACHINE MOVE.  SEE:  EXPLAIN,PINFO.)

/ LI,SW

* (SWITCH,LIST,ONOFF)
* (LI  [LIST] CAUSES THE BOARD TO BE PRINTED AUTOMATICALLY AFTER)
* (EACH MACHINE MOVE.  SEE:  EXPLAIN,PRINT.  FOR THE BOARD)
* (FORMAT.)

/ TI,SW

* (SWITCH,TIMING,ONOFF)
* (TI  [TIMING] TURNS ON AUTOMATIC TIME CONTROL, WHICH IS A)
* (DEVICE THAT REGULATES TIME SPENT PER MOVE SO AS TO AVERAGE A)
* (DESIRED AMOUNT.  INITIALLY, TI IS OFF, AND EACH MOVE IS)
* (SEARCHED TO THE FIXED PLY DEPTH SPECIFIED BY THE LET FDEPTH,)
* (WHOSE DEFAULT VALUE IS 2, THE MINIMUM.  AT ANY GIVEN VALUE OF)
* (FDEPTH, THE TIME SPENT PER MOVE MAY VARY OVER A WIDE RANGE,)
* (DIFFERING BY ORDERS OF MAGNITUDE BETWEEN A COMPLICATED MIDDLE)
* (GAME AND A SIMPLE END GAME.  EVEN MOVE-TO-MOVE VARIATIONS MAY)
* (BE LARGE.  SO THE TIME CONTROL MECHANISM SERVES TO DISTRIBUTE)
* (TIME MORE EVENLY, AND ALSO TO MEET THE PRECISE TIME)
* (CONSTRAINTS IMPOSED BY TOURNAMENT PLAY.  [THE TOURNAMENT)
* (TYPE-IN AUTOMATICALLY TURNS ON TIMING].  WITH TIMING ON,)
* (MOVE-TO-MOVE VARIATIONS MAY STILL BE FAIRLY LARGE [UP TO A)
* (FACTOR OF 10 OR SO], BUT THE AVERAGE TIME OVER TIME-CONTROL)
* (PERIODS IS WELL-CONTROLLED.  A SERIES OF LET VARIABLES SPECIFY)
* (THE TIMING CONDITIONS.  THEIR DEFAULT VALUES ARE SET FOR THE)
* (ACM AND IFIPS EVENTS, WHICH ARE TIMED VARY MUCH LIKE USCF)
* (RATED HUMAN TOURNAMENTS.  THE MAIN LETS TO BE ADJUSTED FOR A)
* (TOURNAMENT ARE:  RULMV1,RULTM1,RULMV2,AND RULTM2.  THESE GIVE)
* (THE SIZES IN MOVES AND MINUTES OF THE FIRST AND SUBSEQUENT)
* (TIME CONTROL PERIODS.  SO, FOR A TOURNAMENT RUN AT THE PACE)
* (OF 40 MOVES IN THE FIRST HOUR-AND-A-HALF, AND 30 PER HOUR)
* (THERE-AFTER, SET:  LET,RULMV1=40;LET,RULTM1=90;)
* (LET,RULMV2=30;LET,RULTM2=60.  ANOTHER LET, EXTRAT, SPECIFIES)
* (A SAFETY MARGIN -- HOW EARLY, IN MINUTES, THE PROGRAM SHOULD)
* (AIM TO FINISH EACH TIME CONTROL PERIOD.  IN CASE THE)
* (PROGRAM GETS INTO TIME PRESSURE DUE TO SYSTEM CRASH)
* (OR TREE EXPLOSIONS OR OTHER REASONS, IT MAY DIP INTO)
* (EXTRAT DOWN TO A MINIMUM SAFETY MARGIN OF EXTRAS.)
* ()
* (IF CPU TIME AND REAL TIME WERE EQUIVALENT, THE ABOVE SETTINGS)
* (MIGHT SUFFICE.  HOWEVER, THE PROGRAM KNOWS ONLY HOW MUCH CPU)
* (TIME IT CONSUMES, AND DOESNT KNOW HOW MUCH REAL TIME ELAPSES)
* (ON ITS CHESS CLOCK AT THE TOURNAMENT SITE.  SO THERE ARE LETS)
* (THAT GUIDE THE PROGRAM IN ESTIMATING THE REAL TIME FROM THE)
* (CPU TIME, AND OTHER LETS THAT SPECIFY HOW OFTEN THE PROGRAM)
* (ASKS TO HAVE THE REAL TIME TYPED IN, SO THAT IT MAY CORRECT)
* (ITS ESTIMATE.  THE LET OVRHED SPECIFIES, IN SECONDS, A FIXED)
* (OVERHEAD TIME PER MOVE.  IT REPRESENTS SUCH THINGS AS THE)
* (TIME BETWEEN THE MOMENT THE PROGRAMS CLOCK IS RESTARTED)
* (AND THE MOMENT ITS REPRESENTATIVE FINISHES TYPING IN THE)
* (OPPONENTS MOVE, SO THE PROGRAM MAY BEGIN THINKING AGAIN.)
* (THE LET PERCNT IS AN ESTIMATE OF THE PERCENTAGE OF TOTAL)
* (CPU TIME THAT IS AVAILABLE TO THE PROGRAM.  THUS IF THE)
* (PROGRAM IS TO RESIDE IN MEMORY WHENEVER IT NEEDS TO, BUT)
* (MUST SHARE THE CPU EQUALLY WITH ANOTHER DEDICATED JOB, THEN)
* (SET: LET PERCNT 50.  THE LET TQFRST SPECIFIES THE FIRST MOVE)
* (NUMBER AT WHICH THE PROGRAM WILL ASK [AFTER MAKING A MOVE]:)
* (HOW MUCH TIME HAVE I USED.  THE TOTAL NUMBER OF MINUTES ON)
* (THE PROGRAMS CLOCK SINCE THE START OF THE GAME SHOULD THEN)
* (BE TYPED IN.  THE LET TQNEXT IS INITIALLY SET TO TQFRST, AND)
* (IS RESET TO TQFRST AUTOMATICALLY BY AN INI COMMAND.  TQNEXT)
* (NEED NOT BE SET MANUALLY, BUT IS ADVANCED AUTOMATICALLY EACH)
* (TIME THE PROGRAM ASKS ABOUT THE TIME.  TQNEXT SPECIFIES THE)
* (NEXT MOVE NUMBER AT WHICH THE TIME QUESTION WILL BE ASKED.)
* (THE PROGRAM INQUIRES MORE FREQUENTLY AS THE TIME CONTROL MOVE)
* (APPROACHES, DEPENDING ON THE SETTING OF THE LET ASKFCT.)
* (ASKFCT [ASK FACTOR] IS A PERCENTAGE OF THE NUMBER OF MOVES)
* (BETWEEN THE LAST TIME INQUIRY AND TIME CONTROL.  WHEN THAT)
* (FRACTION OF MOVES HAS ELAPSED, THE TIME QUESTION WILL BE ASKED)
* (AGAIN.)
* ()
* (THE LET MDEPTH SHOULD BE LOWERED IF A GAME IS TO BE PLAYED)
* (MUCH FASTER THAN THE DEFAULT RULES.  WHEN TIMING IS ON, THE)
* (PROGRAM WILL SEARCH TO A MINIMUM DEPTH OF MDEPTH AND CONFIRM)
* (THE SELECTED MOVE TO A DEPTH OF MDEPTH + 1. THIS IS TO PROTECT)
* (AGAINST OCCASIONAL SHORT SEARCHES THAT MIGHT LEAD TO AVOIDABLE)
* (BLUNDERS.  THE DEFAULT SETTING OF MDEPTH [3] WOULD PROBABLY)
* (STILL BE OK AT THE DEFAULT TIMING RULES EVEN IF ONLY HALF THE)
* (CPU POWER OF A CDC 6400 WERE AVAILABLE.  TO PLAY A FAST)
* (INFORMAL TIMED GAME, SEE:  EXPLAIN,BLITZ.  THE GCLOCK LET)
* (CONTAINS THE ESTIMATED REAL TIME [LIKE WHAT IS TYPED IN ANSWER)
* (TO: HOW MUCH TIME HAVE I USED.].  IF THE USER MUST ADJUST IT)
* (MANUALLY, HE SHOULD USE THE CLOCK TYPE-IN; SEE: EXPLAIN,CLOCK.)
* (THE OTHER LETS RELATED TO TIMING FINE-TUNE THE MECHANISM.)
* (THEY ARE:  TGRACE,TMAXOM,TPMRAT,TPMRMX,TRATIC,TRATLO,TRATMN,)
* (TRATMX, AND TRATOL.  NOTE:  IF FOR SOME REASON LETS RELATED TO)
* (TIME CONTROL [OR THE LET MOVNUM] MUST BE CHANGED DURING A)
* (GAME, THEN THE CLOCK COMMAND SHOULD BE ISSUED AFTER ALL THE)
* (LET TYPE-INS ARE COMPLETED SO THAT THE PROGRAM PROPERLY TAKES)
* (THE CHANGES INTO ACCOUNT.)
* ()
* (SEE ALSO:  EXPLAIN,TOURNAMENT.)

/ AD,SW

* (SWITCH,ADISPLAY,ONOFF)
* (AD  [A-DISPLAY] CAUSES SUBSEQUENT  YOUR MOVE  AND  MY MOVE)
* (MESSAGES TO APPEAR IN THE SYSTEM DAYFILE SO THEY MAY BE)
* (READ FROM THE DSD A-DISPLAY.  THIS FEATURE MAY BE USEFUL IF)
* (A REMOTE GAME IS TO BE MONITORED AT THE CENTRAL COMPUTER)
* (CONSOLE.  CAUTION:  THE USER MUST BEWARE OF POSSIBLE SYSTEM)
* (DEFINED DAYFILE MESSAGE LIMITS WHEN USING AD.)

/ BE,SW

* (SWITCH,BELL,ONOFF)
* (NOTE: BELL WORKS ONLY AT NORTHWESTERN UNIVERSITY.)
* (BE  [BELL] CAUSES THE BELL TO BE RUNG AFTER EACH MACHINE)
* (MOVE IF THE GAME IS PLAYED FROM AN INTERACTIVE TERMINAL.)
* (THIS IS USEFUL TO ALERT THE USER AT A SILENT [CRT] TERMINAL)
* (DURING A SLOW GAME WHEN MACHINE MOVES TAKE A LONG TIME.)

/ DA,SW

* (SWITCH,DALTER,ONOFF)
* (DA  [DEBUG-ALTER] IS A USEFUL DEBUGGING MODE.  WHEN DA IS ON,)
* (EACH TIME THE PROGRAM PAUSES AT A BREAKPOINT THE CURRENT)
* (ALTER FILE IS REWOUND AND READ FOR COMMANDS, JUST AS IF AN)
* (ALTER COMMAND HAD BEEN ISSUED.  THIS ALLOWS A REPETITIVE)
* (DEBUGGING PROCEDURE [SUCH AS THE PRINTING OF CERTAIN)
* (VARIABLES] TO BE EXECUTED AUTOMATICALLY AT EACH BREAKPOINT.)
* (SEE ALSO:  EXPLAIN,ALTER ; EXPLAIN,AFILE ; EXPLAIN,ANY ;)
* (EXPLAIN,BKP ; EXPLAIN,SWITCH,MA.)

/ MA,SW

* (SWITCH,MALTER,ONOFF)
* (MA  [MOVE-ALTER] MAY BE USEFUL FOR DEBUGGING OR OTHER)
* (PURPOSES.  WHEN MA IS ON, THE CURRENT ALTER FILE IS REWOUND)
* (AND READ FOR COMMANDS AFTER EACH MACHINE MOVE, JUST AS IF AN)
* (ALTER COMMAND WERE ISSUED.  THUS A REPETITIVE PROCEDURE, SUCH)
* (AS THE PRINTING OF CERTAIN VARIABLES, MAY BE PERFORMED)
* (AUTOMATICALLY AFTER EACH MACHINE MOVE.  SEE ALSO:)
* (EXPLAIN,ALTER ; EXPLAIN,AFILE ; EXPLAIN,BKP ; EXPLAIN,SW,DA.)

/ OL,SW

* (SWITCH,OLINE)
* (NOTE: OLINE REQUIRES A SPECIAL PP PROGRAM CALLED 1OL.)
* (OL  [ONE-LINE] IS AN I/O MODE SWITCH.  WHEN ON, INPUT/OUTPUT)
* (TAKES PLACE BETWEEN THE PROGRAM [RUNNING AS A BATCH JOB AT)
* (A FIXED CONTROL POINT] AND A REMOTE TERMINAL VIA THE ONELINE)
* (SUBSYSTEM OF EI200.  THE ONELINE SYSTEM WAS JURY-RIGGED FOR)
* (THE IFIPS 74 TOURNAMENT BECAUSE THE STOCKHOLM CDC DATA CENTER)
* (6600 COMPUTER SYSTEM HAD NO INTERACTIVE TIME-SHARING SYSTEM OF)
* (ITS OWN.  CONNECTING THE CHESS BATCH JOB TO THE DESIRED)
* (TERMINAL REQUIRES MANUAL SETTING OF SOME WORDS WITHIN THE)
* (EI200 FIELD LENGTH.  WHILE OL IS ON, COMMANDS AND MOVES MAY)
* (ALSO BE ENTERED VIA THE N.CFO TYPE-IN AT THE CENTRAL CONSOLE)
* (AS IN DS MODE.  ALSO AS IN DS MODE, PROGRAM RESPONSES ARE)
* (ALSO WRITTEN TO THE CHESS OUTPUT FILE.  SEE:  EXPLAIN,SW,DS ;)
* (EXPLAIN,SW,CA ; EXPLAIN,SW,DI.)

/ IM,SW
```

```
 * (SWITCH,IMLAC)
 * (NOTE: IMLAC WORKS ONLY AT NORTHWESTERN UNIVERSITY.)
 * (IM  [IMLAC] IS AN I/O MODE.  THE IMLAC GRAPHICS TERMINAL AT)
 * (NUCC CAN SERVE AS AN ORDINARY INTERACTIVE TERMINAL UNDER THE)
 * (ONLINE TIME-SHARING SYSTEM.  SO, CHESS MAY BE PLAYED FROM)
 * (THE IMLAC AS FROM A TTY OR OTHER CRT.  SWITCHING ON IM)
 * (ACTIVATES AN ENHANCED PROGRAM/PLAYER COMMUNICATION MODE THAT)
 * (WAS IMPLEMENTED WITH THE GYPSY GRAPHICS SUBROUTINE PACKAGE.)
 * (IN IM MODE, THE CHESS BOARD AND PIECES ARE DISPLAYED [AS IN)
 * (A CHESS DIAGRAM], AND THE PLAYER MAKES MOVES WITH THE LIGHT)
 * (PEN INSTEAD OF TYPING THEM IN.  TO MAKE A MOVE USING THE)
 * (LIGHT PEN:)
 * ()
 * (1.  HOLD LIGHT-PEN OVER PIECE TO BE MOVED.)
 * (2.  PRESS LIGHT PEN BUTTON.  PIECES WILL NOW DIM, INDICATING)
 * (    PIECE SELECTION IS COMPLETE.)
 * (3.  WAIT FOR TRACKING CROSS TO APPEAR UNDER LIGHT PEN.)
 * (4.  MOVE PEN CAREFULLY, SO THAT TRACKING CROSS FOLLOWS, TO)
 * (    DESIRED DESTINATION SQUARE.)
 * (5.  RELEASE LIGHT PEN BUTTON.)
 * (6.  MOVE LIGHT PEN TO X-BAR, IN BOTTOM CENTER OF THE SCREEN.)
 * (7.  PRESS THE LIGHT PEN BUTTON BRIEFLY TO SELECT THE X-BAR)
 * (    AND COMPLETE THE MOVE.)
 * (8.  IF THE MOVE IS LEGAL, THE MOVING PIECE WILL BECOME)
 * (    ANIMATED AND SLIDE FROM ITS ORIGIN TO ITS DESTINATION)
 * (    IN A SERIES OF SMALL STEPS [EXCEPT FOR CASTLING AND)
 * (    ENPASSANT CAPTURES].)
 * (9.  THE PROGRAM WILL THEN THINK ABOUT ITS MOVE.)
 * (10. THE PROGRAMS MOVE WILL ALSO BE SHOWN IN ANIMATION.  IN)
 * (    ADDITION, THE  MY MOVE  MESSAGE WILL BE DISPLAYED.)
 * (11. THE PIECES WILL BRIGHTEN, INDICATING THAT IT IS AGAIN)
 * (    THE PLAYERS TURN TO MOVE.)
 * ()
 * (NOTE:)
 * (A. CASTLING IS DONE BY MOVING THE KING THE 2 SQUARES)
 * (   TO ITS DESTINATION.)
 * (B. IF THE PLAYER WISHES TO RETRACT A PIECE HE HAS SELECTED AND)
 * (   MOVE ANOTHER ONE, HE MAY DO SO [IF HE HASNT YET SELECTED)
 * (   THE X-BAR] BY MOVING THE PIECE BACK TO ITS ORIGINAL SQUARE)
 * (   [OR ANY ILLEGAL DESTINATION] AND SELECTING THE X-BAR.  THE)
 * (   MESSAGE:  ILLEGAL MOVE   WILL BE DISPLAYED, AND THE PLAYER)
 * (   MAY START OVER.)
 * (C. WHEN A PLAYER MOVES A PAWN TO THE EIGHTH RANK, MODELS OF)
 * (   A QUEEN, ROOK, BISHOP, AND KNIGHT WILL APPEAR.  THE PLAYER)
 * (   SELECTS WITH THE LIGHT PEN THE PIECE HE WISHES TO PROMOTE)
 * (   HIS PAWN TO.)
 * (D. TO TYPE IN A COMMAND, THE PLAYER SELECTS [WITH LIGHT PEN])
 * (   THE BOX LABELLED:  TYPE-IN.  AFTER A QUESTION MARK APPEARS,)
 * (   THE COMMAND MAY BE TYPED IN, FOLLOWED BY A CARRIAGE RETURN.)
 * (   SOMETIMES WHEN IT IS INAPPROPRIATE FOR THE PLAYER TO MAKE A)
 * (   MOVE [SUCH AS AT END OF GAME], THE PROGRAM AUTOMATICALLY)
 * (   WILL PREPARE FOR A TYPE-IN AND DISPLAY A QUESTION MARK.)
 * (E. CONTRARY TO WHAT HAPPENS IN DS OR DI MODE, PROGRAM)
 * (   RESPONSES ARE NOT WRITTEN TO THE CHESS OUTPUT FILE.  SEE:)
 * (   EXPLAIN,HARDCOPY  FOR HOW TO GET A PRINTED RECORD OF THE)
 * (   GAME.  THE LET COPIES MAY ALSO BE USED TO PRINT COPIES OF)
 * (   THE GAME SCORE AT THE CENTRAL SITE.)
 * ()
 * (SEE ALSO:  EXPLAIN,SW,CA ; EXPLAIN,SW,DS ; EXPLAIN,SW,DI.)

 / RA,SW

 * (SWITCH,RANDOM)
 * (RA  [RANDOM] CAUSES A NEW RANDOM SEED FOR LET)
 * (JIGGLING TO BE OBTAINED BEFORE EACH MOVE.)

 / PO,SW

 * (SWITCH,PONDER)
 * (PO  [PONDER] ENABLES THINKING ON THE OPPONENTS TIME.)
 * (THE PROGRAM ASSUMES THAT THE OPPONENT WILL MAKE THE MOVE)
 * (PREDICTED IN THE PRINCIPAL VARIATION, AND SO BEGINS ITS)
 * (CALCULATION WITHOUT WAITING FOR A MOVE TO BE TYPED.  WHEN)
 * (A MOVE IS ACTUALLY TYPED, THE SEARCH IS INTERRUPTED.  IF)
 * (THE MOVE IS AS PREDICTED, THE SEARCH IS RESUMED.  IF NOT,)
 * (IT IS CANCELLED AND RESTARTED WITH THE CORRECT OPPONENT)
 * (MOVE.  PONDER MODE IS AUTOMATICALLY SELECTED BY THE)
 * (TOURNAMENT TYPE-IN.)

 / AL,SW

 * (SWITCH,ALGEBRAIC)
 * (AL [ALGEBRAIC] CAUSES ALL MOVES PRINTED BY THE PROGRAM)
 * (TO BE EXPRESSED IN ALGEBRAIC NOTATION.  SEE: EXPLAIN,MOVES)

 / CP,SW

 * (SWITCH,CBD)
 * (CBD  CALLS THE PP PROGRAM TO THE CONTROL POINT AND SWITCHES)
 * (THE PROGRAM TO DSD INPUT MODE.  IT IS USED FOR THE)
 * (ELECTRONIC CHESS BOARD UNDER DEVELOPMENT.)

 / $,SW

         USE    SP
         CON    0
         USE    SW
         CON    0
         USE    EX
         CON    0

OVERLAY  ELSE   1
END      MICRO  1,,

         END    'END'
W/BIRDS
MSG BIRDS OPENING.
LET MSMASK=77;LET MSCODE=0
INI
YES.  BIRDS OPENING.
SW NO ON;SW LB OFF;SW LW ON;SW EXP OFF
LET MSCODE=30    SELECT 15 5/8 PERCENT OF THE TIME.
P-KB4
LET MSCODE 0
SAVE MOVE1 KINGS GAMBIT
P-K4
P-K4
SAVE MOVE2 COTAP 7-1
PXP
N-KB3
SAVE MOVE3 MCO10 99 11
P-Q3
B-B4
P-KR3
P-Q4
P-KN4
O-O
B-N2
P-B3
SAVE MOVE7
N-QB3
P-KN3
B-R6
PXP
BXR
QXB
PXP
BXP/B4
Q-B3
B-KN3
O-O-O
N/QN1-Q2
N/KN1-K2
Q-R3.CHECK
K-N1
R-KB1
Q-N3
N-R4
SAVE MOVE16
Q-R2.?
P-N4
SET MOVE16
Q-N4.!
N/R4-B3
SET MOVE7
N-K2
P-KN3
SAVE MOVE8
P-Q4
PXP/Q
PXP
PXP
O-O
Q-N3
Q-Q3
K-N2
SET MOVE8 MCO10 99 F
N-N3
Q-N3
O-O
PXP
PXP
K-R1
N-B3
Q-B2
N/QB3-K2
N/QN1-Q2
SET MOVE8
P-N5
N-R4
P-B6
NXP
PXN
BXP/KB7.CHECK
KXB
QXP.CHECK
SAVE MOVE12
K-K1
Q-R5.CHECK
K-Q2
Q-N4.CHECK
K-B3
P-Q5.CHECK
K-N3
B-K3.CHECK
P-B4
SET MOVE12
K-N1
Q-B7
K-R2
R-B6.!
N-B4.!
Q-N6.CHECK. DRAW.
SET MOVE3 MCO10 99 13
P-KN4
P-KR4
P-N5
N-K5
SAVE MOVE5 COTAP 7-1/PV1
N-KB3
P-Q4
P-Q3
N-Q3
NXP
BXP
SAVE MOVE9 MCO10 99 13
B-N2
P-B3
Q-K2
Q-K2
B-B4
SET MOVE9
Q-K2
Q-K2
B-N2
P-B3
SAVE MOVE10 COTAP 7-1/IV3
P-KR4
N-Q2
NXN
KXN
QXQ
BXQ
N-B3
R/QR1-K1
SET MOVE10
B-B4
N-Q2
NXN
QXQ.CHECK
KXQ
KXN
SET MOVE5 COTAP 7-1/PV2
B-N2
P-Q4
SAVE MOVE6
P-Q3
NXP/N
BXN
QXB
BXP
P-B3
B-K4
```

```
BXP
BXB
QXB
Q-B3
P-KN3
N-Q2
N-Q2
QXQ
PXQ
N/KN-B3
K-B2
SET MOVE6
N-KB3
N-QB3
P-Q3
N-Q3
O-O
NXP
NXP
NXN
R-K1
K-B2
RXN
P-B3
Q-B3
P-KN3
B-R3
B-Q3
BXN
BXB
SET MOVE5 COTAP 7-1/PV3
P-Q4
P-Q4
N-KB3
BXP
NXP
N-Q2
NXN
QXN
B-Q3
O-O-O
B-K3
B-Q3
SAVE MOVE11 COTAP NOTE 6
P-KB3.?
R/Q1-K1
SET MOVE11
N-Q2
R/Q1-K1
NXN
BXN
BXB
RXB
Q-Q2
Q-N5.!
Q-K2
B-B5
QXQ
PXQ
SET MOVE5 COTAP 7-1/PV4
P-Q3
NXP/N
SAVE MOVE6 COTAP NOTE 8
P-KR4
N-B2
N-KB3
P-Q4
B-R3
N-B3
N-N5
NXN.!
SAVE MOVE10
PXN
N-Q5.!
SET MOVE10
BXN
Q-Q3
SET MOVE6
B-K2
P-Q4
BXP.CHECK
N-B2
SAVE MOVE8 COTAP NOTE 9
Q-N4
Q-B3
N-QB3
QXP
QXQ
BXQ
BXN.CHECK
KXB
NXP
B-Q3
SET MOVE8
Q-B3
Q-B3
B-N6
N-B3
QXP
BXP
BXN.CHECK
QXB
QXQ
KXQ
N-KB3
B-K2
N/QN1-Q2
N-N5
K-Q1
B-B3. SLIGHTLY FAVORS WHITE. KOSTIC-TOMOVOC, ZAGREB 1949.
SET MOVE5 COTAP 7-1/PV5
P-KR4
B-B4
SAVE MOVE6 COTAP NOTE 12
N-KR3
P-Q4
P-Q3
N-Q3
P-B6
PXP
PXP
QXP
B-N5
Q-B2
Q-K2
N-B3.! WITH ADVANTAGE TO WHITE.
SET MOVE6
R-R2
P-Q4
SAVE MOVE7 MCO10 103 HA
B-R3
N-QB3
P-Q3
N-Q3
P-B6
PXP
SET MOVE7
B-Q3
N-QB3
N-QB3
BXP/KB7.CHECK
RXB
NXR
KXN
BXP
BXB
O-O
QXP
RXB.CHECK
SET MOVE5 COTAP NOTE 11-1
B-K2
B-B4.!
BXP/KR5.CHECK
K-B1
SAVE MOVE7
N-KR3
NXP/N
SET MOVE7
P-Q4
BXP
N-KR3
P-Q4
B-N4
N-QB3
SET MOVE5 COTAP NOTE 11-2
Q-K2
P-Q4.!
P-Q3
NXP/N
SAVE MOVE7
QXP/K.CHECK
Q-K2
SET MOVE7
P-KB4
N-B2
SAVE MOVE8
PXP
Q-R5.CHECK
SET MOVE8
N-KB3
BXP
SAVE MOVE9
PXP
P-Q5.!
SET MOVE9
NXP
Q-R5.CHECK
SET MOVE5 COTAP NOTE 11-3
N-QB3
P-Q4.!
NXN
PXN
P-Q3
BXP
SAVE MOVE8
Q-K2
B-QN5.CHECK
SET MOVE8
B-N2
N-B3
PXP
QXQ.CHECK
KXQ
O-O-O.CHECK
B-Q2
B-K3
SET MOVE3 COTAP 7-4/SV3
P-KR3
P-Q4
P-KN4
P-KR4
B-N2
P-KN3
SAVE MOVE6 MCO10 99 AB
P-Q3
PXP/B
P-N5
N-N1
SET MOVE6
P-N5
N-R2
PXP
NXP
SAVE MOVE8 MCO10 99 AB
P-Q4
P-K5
B-B4
B-KB4
SET MOVE8
P-Q3
P-B3
N-KB3
NXN.CHECK
QXN
B-K3. = PACHMAN.
SET MOVE3 COTAP 7-4/SV5
N-KB3
P-K5
N-R4
P-Q4
SAVE MOVE5 MCO10 98 8
P-Q4
N-B3
P-KN4
B-K2
P-N5
O-O
SAVE MOVE8 MCO10 102 K
PXN
BXP/3
N-N2
NXP
SET MOVE8
R-N1
```

```
N-K1
B-R3
N-Q3
SET MOVE5
P-Q3
N-B3
SAVE MOVE6 COTAP NOTE 11
PXP
Q-K2.!
N-QB3
P-Q5
SET MOVE6
B-N5
B-B4
SET MOVE5 MCO10 98 H
P-KN4
P-KN4
PXP/6.EP
NXP
SET MOVE3 COTAP 7-3/PV12
B-K2
B-B4
SAVE MOVE4 COTAP 7-3 A
B-R5.CHECK
K-B1
SAVE MOVE5 MCO10 102 P A
P-Q4
BXP
N-KB3
N-B3
O-O
P-Q4
P-B3
B-N3
B-N5
BXP/B4
N-R4
SET MOVE5
B-B3
P-Q4
P-KN4
P-KR4
SAVE MOVE7
P-N5
N-K5
SET MOVE7
P-KR3
N-K5
SET MOVE4 COTAP 7-3/PV12
N-KB3
P-K5
N-N5
N-B3.!
SAVE MOVE6
P-Q3
PXP
SAVE MOVE7 COTAP NOTE 5
QXP
P-Q4
O-O
O-O
B-K3
SET MOVE7
BXP
Q-K2.CHECK
Q-K2
QXQ.CHECK
SAVE MOVE9 COTAP NOTE 6
BXQ
P-Q4
B-Q3
N-K4. BRONSTEIN-LEMOINE, MUNICH 1958.
N-Q2
NXB.CHECK
PXN
BXP/4
SET MOVE9
KXQ
N-Q5.CHECK
K-Q1
P-Q4. =
SET MOVE6 COTAP 7-3/PV13
P-Q4.?
BXP
B-R5.CHECK
K-B1
SAVE MOVE8 MCO10 102 S B
N-B7.?
Q-K1
SET MOVE8
N-QB3
BXN.CHECK
PXB
P-Q3
O-O
BXP
P-B3
P-K6
P-KB4
NXB
QXN
Q-K1.! PANOV
SET MOVE6 MCO10 102 S A
B-R5.CHECK
K-B1.!
O-O
P-Q4
P-Q3
P-KN3
SET MOVE4 MCO10 102 P B
P-Q3
O-O
B-K3.!?
BXB
PXB
N-Q4.!
Q-Q2
Q-N4
SET MOVE3 COTAP 7-4
P-Q4
PXP
SAVE MOVE4
QXP
N-B3
Q-K3.CHECK
K-B2.!
Q-QN3.CHECK
P-Q4
N-KB3
B-N5.CHECK
P-B3
R-K1.CHECK
B-K2
B-B4
SET MOVE4
B-Q3
P-Q4
P-KN4
P-B4
P-QN3
B-Q3
SET MOVE4 COTAP 7-4/PV12
N-KB3
B-N5.CHECK
SAVE MOVE5 COTAP 7-4/IV12
N/QN1-Q2
O-O
NXP
P-B4
SAVE MOVE7
N-N3
R-K1.CHECK
B-K2
P-B5
SET MOVE7
N-B3
P-Q4
SAVE MOVE8
P-B3
B-R4
B-Q3
R-K1.CHECK
K-B1
N-B3
SET MOVE8
B-K2
BXP
SET MOVE5
B-Q2
B-B4
SAVE MOVE6 MCO10 98 E B
Q-K2.CHECK
Q-K2
QXQ.CHECK
KXQ
B-Q3
P-Q4
O-O
SET MOVE6
B-Q3
O-O
SET MOVE5 COTAP 7-4/PV16
P-B3
PXP
PXP
B-B4
SAVE MOVE7
B-Q3
Q-K2.CHECK
Q-K2
QXQ.CHECK
KXQ
O-O
B-K3
R-K1
N/QN1-Q2
P-Q4. BHEND-BARCZA, ZURICH 1959.
SET MOVE7 COTAP 7-4/PV18
N-Q4.!
O-O.!
B-Q3
N-B3
SAVE MOVE9 COTAP NOTE 7
NXN
R-K1.CHECK!
SET MOVE9
B-K3
N-K4
B-K2
B-N3
O-O
P-Q4
N-Q2
Q-K2.!
P-N4
P-B4
N/4-N3
P-KR4.!
P-KR3
PXP
PXP
N/3XP.!
BXN
BXP. WITH A STRONG ATTACK FOR THE PIECE. SPASSKI-LUTIKOV, USSR 1960.
SET MOVE2 MCO10 100 A A
P-Q3
P-Q3
N-KB3
PXP
SET MOVE2 MCO10 100 A B
N-KB3
PXP
NXP
N-KB3
P-Q4
P-Q3
N-B4
P-Q4
SET MOVE2 COTAP 7-5
B-B4
N-KB3
P-Q3
P-B3
SAVE MOVE4 MCO10 104 O A
N-KB3
PXP
PXP
P-Q4
PXP
PXP
SAVE MOVE7
B-QN5.CHECK
B-Q2
BXB.CHECK
```

```
N/QN1XB
O-O
B-Q3
P-B4
P-Q5
B-N5
O-O
N/QN1-Q2
Q-B2
SET MOVE7
B-N3
P-K5
N-Q4
B-QB4
B-K3
Q-N3
O-O
N-B3
P-QB3
O-O
N-B2
B-K3
N-Q2
SET MOVE4 COTAP 7-5/PV22
B-KN5
PXP
PXP
Q-R4.CHECK
SAVE MOVE6 COTAP NOTE 4
Q-Q2
B-N5
P-QB3
NXP.!
SET MOVE6
N-B3
NXP
Q-R5.CHECK
P-N3
B-B7.CHECK
KXB
Q-B3.CHECK
K-N1
QXN
B-N2
SET MOVE6
B-Q2
Q-B2
SAVE MOVE7 MCO10 104 M
Q-K2
P-Q4
PXP
PXP
B-QN5.CHECK
N-B3
B-B3
B-Q3
BXN.CHECK
PXB
BXP
BXB
P-KB4
O-O
PXB
Q-N3.! EUWE-MAROCZY, MATCH 1921.
SET MOVE7
N-QB3
P-QN4
B-Q3
B-QB4
N-B3
P-Q3
SAVE MOVE10 MCO10 104 N
N-K2
O-O
N-N3
P-QR4
SET MOVE10
Q-K2
O-O
O-O-O
P-QR4.! BRONSTEIN-PANOV, MOSCOW 1947.
P-QR4
P-N5
N-QN1
N/QN1-Q2
B-KN5
N-N3
P-QN3
B-K3
SET MOVE4 COTAP 7-5/PV23
P-KB4.
PXP/K.!
SAVE MOVE5 MCO10 104 Q
P/BXP.?
Q-R4.CHECK
SET MOVE5
P/QXP
P-Q4
PXP/Q
B-QB4.!
SAVE MOVE7 COTAP NOTE 6
N-KB3
P-K5
N-K5
PXP
SAVE MOVE9
B-N5.CHECK
B-Q2
SET MOVE9
B-N3
N-B3
N-QB3
B-K3
N-R4
B-Q3
NXN
PXN
O-O
P-B4
SET MOVE7
PXP/K
N-K5
N-KB3
N-B7
Q-K2
NXR
P-Q6
B-KN5
B-B7.CHECK
KXB
Q-B4.CHECK
B-K3
QXB/B5
P-KR3
B-K3
N-Q2
Q-Q4
P-KN4.!
SET MOVE7
N-QB3
P-QN4.!
B-N3
Q-N3
SAVE MOVE9 MCO10 104 RB
N-B3
P-N5
N-QR4
B-B7.CHECK
SET MOVE9 MCO10 104 RB
N/KN1-K2
B-B7.CHECK
K-B1
O-O.!
SET MOVE9
N-R3
B-KN5
Q-Q3
N/1-Q2
SET MOVE7 MCO10 104 RB
P-Q6
B-KN5.!
N-KB3
P-K5
P-KR3
B-R4
P-KN4
PXN
PXB
N-K5
SET MOVE4 COTAP 7-5/IV14
N-QB3
B-N5.!
SET MOVE2 COTAP 7-6
P-Q4
PXP/Q
SAVE MOVE3 COTAP 7-6 B
P-QB3
N-QB3.!
PXP/B
N-B3
N-B3
P-Q4
NXP
NXN
QXN
BXP
Q-K5
Q-K2
QXQ
BXQ
B-K2
O-O
B-K3
P-B4
O-O
N-N5
SET MOVE3 TRANSPOSE INTO MODERN VARIATION
PXP
N-KB3
SET MOVE3
QXP
N-QB3
SET MOVE3 COTAP 7-6/PV26
P-K5
P-Q3
SAVE MOVE4 COTAP 7-6/IV21
QXP
Q-K2.!
N-KB3
N-QB3
B-QN5
B-Q2
BXN
BXB
B-N5
PXP
SET MOVE4 MCO 101 D B
PXP
QXP
N-KB3
N-QB3
B-QB4
B-Q2
O-O
O-O-O
N/QN1-Q2
P-KN3. +
SET MOVE4
N-KB3
PXP
NXP/K
N-KB3
SAVE MOVE6 MCO10 101 F
B-QN5.CHECK
P-B3
B-QB4
Q-K2
B-B4
N/QN1-Q2
Q-K2
NXN
BXN
P-B4
SET MOVE6
B-QB4
Q-K2
SAVE MOVE7
B-B7.CHECK
K-Q1
QXP.CHECK
N/KB3-Q2.!
SET MOVE7
B-B4
N-B3
```

```
Q-K2
B-K3
SAVE MOVE9 MCO10 101 G
NXN
BXB
NXQ
BXQ
NXP
P-Q6.!
SET MOVE9
BXB
QXB
NXN
QXQ.CHECK
KXQ
PXN
BXP
K-Q2.! WHEATCROFT-KERES, MARGATE 1939.
SET MOVE1 SOLTIS 4
N-QB3
N-KB3. TRANSPOSES
SET MOVE1
P-QN4.!?
N-KB3
B-N2
P-K3
P-N5
P-QR3
SET MOVE1 SOLTIS 4A
P-KB4
P-K4.!
SAVE MOVE2
P-Q3
PXP
BXP
P-Q4
SET MOVE2
PXP
P-Q3.!
SAVE MOVE3
PXP
BXP
N-KB3
N-KB3
SAVE MOVE5
P-K3
N-N5.!
P-KN3
P-KR4
B-R3
P-R5.! BIRD-GILBFUSS VIENNA 1873
SET MOVE5
P-Q4
O-O
N-B3
N-K5
SET MOVE3
P-KN3
PXP
P-K4
N-KB3
N-KB3
B-B4
Q-K2
N-B3
PXP
Q-K2
P-Q3
BXP
B-K3
O-O-O
N-B3
B-QN5
B-Q2
B-N5
B-N2
N-Q5.!
Q-Q1
P-K5. AND WINS RANNIKU-KARAKASH, BRYANSK 1965.
SET MOVE3
N-KB3
PXP
NXP
B-Q3. TRANSPOSES
SET MOVE3
P-K6
BXP
N-KB3
P-Q4
P-K3
B-Q3
N-B3
P-QR3.!
N-K2.!
N-KR3
P-QN3
O-O
B-N2
N-Q2
P-N3
B-B2
B-N2
P-B3
O-O
Q-K2
P-QR4
P-KN4.! PELIKAN-ALEKHINE, PODEBRADY 1936.
SET MOVE1 SOLTIS 3A
P-QB4
N-KB3
SAVE MOVE2
N-QB3
P-K3
P-KN3
B-N5
B-N2
O-O
N-B3
P-Q3
O-O
BXN
P/NXB
P-B4
P-Q3
N-B3
R-N1
Q-B2
N-Q2
P-QN3
P-K4
B-N2
R-K1
N-K4. SMEDEREVAC-MINICH, 1960 YUGOSLAV CHAMPIONSHIP.
SET MOVE2
P-KN3. ALSO GOOD FOR BLACK
P-K3
B-N2
P-Q4
PXP
PXP
P-Q3.!
B-Q3
Q-N3
P-B3
N-KR3
O-O
O-O
K-R1
SET MOVE2
N-KB3
P-K3
P-KN3
P-QN3
B-N2
B-N2
SAVE MOVE5
N-B3
B-N5
P-Q3
O-O
O-O
BXN/QB6
PXB
Q-K1
P-QR4
P-QR4
Q-N3
P-Q3
N-Q4
BXB
KXB
N/QN1-Q2
Q-N5
P-B5
Q-B6
P-K4
N-N5
R-B1
N-R7
N-B4. G. PETROSIAN-SEDOV, MOSCOW 1961.
SET MOVE5
P-Q3.!
B-K2
O-O
O-O
N-B3
Q-K1
P-K4
PXP
N-KN5
N-B3. WORKS A BIT BETTER:
N/KN5XP/4
R-N1
B-K3
NXN
PXN
N-K4
P-N3
Q-N3
P-B3
R-B2
R-B1
B-B4.= PRAKHOV-DIMITROV, BULGARIAN CHAMPION 1961.
SET MOVE1 SOLTIS 3B
N-KB3
P-K3
P-KN3
P-QN3
B-N2
B-N2
P-Q3
N-KB3
O-O
B-K2
SAVE MOVE6
P-K4
PXP
N-KN5
O-O
SET MOVE6
N/QN1-Q2
O-O
P-K4
PXP
N-N5
P-KR3
N/5XP/4
P-Q4
NXN.CHECK
BXN
P-Q4.?
BXP
Q-N4
Q-B3
N-B3
P-K4
NXB
PXN
B-B4
N-R3
P-QB4
PXP/QB6.EP
PXP
P-B3
SET MOVE6
N-B3
O-O
P-K4.!
PXP
N-KN5
P-K4
N/KN5XP/4
NXN
NXN
```

```
P-Q4
N-B3
P-B3
Q-N4
B-B3
B-Q2
N-R3.= CHOLODKEVITSCH-KOTZ, 1951.
SET MOVE1 THOMPSON MN OPEN
P-KN3
N-KB3
B-N2
P-K3
N-KB3
B-K2
SET MOVE1 SOLTIS MAIN LINE
P-Q4
N-KB3
SAVE MOVE2 SOLTIS 2E2A
P-Q5
P-K3
SAVE MOVE3
P-QB4
B-N5.CHECK
B-Q2
Q-K2
SET MOVE3
PXP
P-Q4
SET MOVE2 SOLTIS 2E2B
N-QB3
P-K3
B-N5
B-K2
SET MOVE2 SOLTIS 2E2C
B-N5
P-K3
SAVE MOVE3
P-K4.!?
PXP
N-Q2
P-Q4
P-KB3
B-K2.!
SET MOVE3
N-Q2
B-K2
P-K3
O-O
B-Q3
P-B4
PXP
N-R3
N-N3
NXP
NXN
BXN
SET MOVE2 SOLTIS 2C3
P-QB4
P-K3
N-QB3
B-N5
SAVE MOVE4
P-KN3
N-K5
Q-B2
P-QN3
B-N2
B-N2
N-B3
O-O
O-O
BXN
PXB
N-QB3
SET MOVE4
P-K3
BXN.CHECK
SET MOVE4
B-N5
BXN.CHECK
PXB
O-O
P-K3
Q-K1
BXN
RXB
N-B3
P-Q3
Q-N3
P-QR4
P-N3
P-R5
Q-N1
P-QN3. SPIRO-SPITZER, PITTSBURGH 1949
SET MOVE4
Q-N3
P-B4.!
SAVE MOVE5
P-Q5
N-K5
SET MOVE5
P-K3
N-K5
SET MOVE5
PXP
BXP
P-K3
N-B3
N-B3
O-O
P-QR3
P-QN3
Q-B2
B-N2
P-QN4
B-K2
B-N2
Q-K1
R-Q1
R-B1
B-Q3
Q-N3
R-KN1.?
NXP
PXN
BXN
SET MOVE2 SOLTIS 1D4
P-KN3
P-K3
B-N2
B-K2
SAVE MOVE4
N-KR3
O-O
O-O
P-Q4
SET MOVE4
P-QB4
O-O
N-QB3
P-Q3
SAVE MOVE6
P-N3
Q-K1
B-N2
P-K4
N-N5
N-R3.!?
PXP
PXP
BXP/K
B-Q2.!
BXP/QN
BXN
PXB
R-Q1
Q-B1
B-N5.CHECK
SAVE MOVE14
B-B3
N-B4
SET MOVE14
K-B1
QXB
PXN
B-B6
N-B3
Q-B4
R-QN1
N-N5
P-K3
P-B5.! AND WINS. HORNSTEIN-GOODBOLD, CORRESPONDENCE 1963.
SET MOVE6
P-K4.? IS PREMATURE
PXP
NXP
NXN
BXN
P-Q4
B-Q3
B-N5.CHECK
B-Q2
Q-B3
SET MOVE6
N-R3
P-K4.!
PXP
PXP
QXQ
BXQ
N-Q5
P-B3
NXN.CHECK
BXN
B-N5
B-K3
P-N3
R-Q1
O-O
N-R3.TOT-BAJEC 1951.
SET MOVE2 SOLTIS MAIN LINE
N-KB3
P-K3
SAVE MOVE3 SOLTIS 2B
B-B4
P-QN3
SAVE MOVE4
P-B4
B-N5.CHECK
N-B3.?
N-K5
R-B1
BXN.CHECK
SET MOVE4
P-K3
B-N2
SAVE MOVE5
B-K2
N-B3.!?
O-O
B-Q3
N-K5.?
BXN
PXB
N-K5
N-Q2
N-N4.!
B-N3
N-B2. BIRD-BLACKBURNE, VIENNA 1882.
SET MOVE5
N/QN1-Q2
P-N3
B-Q3
B-N2
Q-K2
O-O
P-K4
PXP
NXP
NXN
BXN
P-Q4
B-N5
Q-Q3
B-Q3
P-B4
SET MOVE3 SOLTIS MAIN LINE
P-B4
P-QN3
SAVE MOVE4 SOLTIS 2C1
N-B3
B-N5
SAVE MOVE5
Q-N3. INADEQUATE
```

```
BXN.CHECK
QXB
N-K5.!
Q-R3
B-N2
N-Q2
P-Q3
NXN
BXN
SET MOVE5
P-QR3
BXN
SET MOVE5
B-Q2
B-N2
SAVE MOVE6
P-KN3
P-B4.!
P-Q5.?!
PXP
NXP
BXN
PXB
NXP
BXB
NXB
P-QR3
N/5-B3
Q-Q5
SET MOVE6
P-QR3.?
B/N5XN
BXB
N-K5
SET MOVE6
P-K3
O-O
B-K2
P-Q3
O-O
B/5XN
BXB
N-K5
SAVE MOVE10
B-K1
N-Q2
N-Q2
NXN
QXN
P-K4
P-B3
Q-B3. NIMZOVICH-RUBINSTEIN, SEMMERING 1929.
SET MOVE10
Q-B2
N-Q2
N-Q2
Q-N4
P-B4
Q-R3
B-B3
N/Q2-B3
SET MOVE10
R-B1
N-Q2
N-Q2
Q-N4
NXN
BXN
B-B3
R-B3
SAVE MOVE14
BXB
PXB
Q-B2
Q-R5. NIMZOVICH-SPIELMANN, NEW YORK 1927.
SET MOVE14
Q-K2
R/QR1-KB1
P-QR4
R-N3. FISCHER-MECKING, PALMA 1970.
SET MOVE4 SOLTIS 2C2
P-QR3
B-N2
P-K3
P-KN3
SAVE MOVE6
P-KN3
B-N2
B-N2
O-O
O-O
P-B4
SET MOVE6
B-K2
B-N2
O-O
O-O
N-B3
Q-K2
N-Q2
SET MOVE3 SOLTIS 2A3
B-N5
P-QN3
SAVE MOVE4
P-B4.?!
B-N5.CHECK!
SAVE MOVE5
N-B3
BXN.CHECK!
PXB
P-KR3
BXN
QXB
SET MOVE5
N/QN1-Q2
P-KR3
BXN
QXB
SET MOVE4
N/QN1-Q2
B-N2
SET MOVE4
P-B3
B-N2
N/QN1-Q2
B-K2
BXN
BXB
P-K3
O-O
SET MOVE4
P-KN3
B-K2
B-N2
B-N2
P-B3
O-O
O-O
N-K5
BXB
QXB
SAVE MOVE9
N/KB3-Q2
P-Q4
NXN
P/BXN
P-KB4.LARSEN-FUSTER, PORTOROZ 1958.
SET MOVE9
N/QN1-Q2
P-Q4
SET MOVE3 SOLTIS 1B2
P-KN3
B-K2
B-N2
O-O
SAVE MOVE5
P-B4
P-Q3
N-B3
Q-K1
Q-B2
SET MOVE5
O-O
P-Q3
SAVE MOVE6
P-N3
Q-K1
B-N2
N/QN1-Q2
N/QN1-Q2
Q-N3
P-K3
N-K5
N-K1
N/Q2-B3
N-Q3
Q-R3
Q-K2
N-N5
P-KR3
N/N-B3
R/KB1-Q1
B-Q2
N-KB1
B-B3
SET MOVE6
P-B4
Q-K1
N-B3
P-B3
SAVE MOVE8
B-N5
N/QN1-Q2
Q-B2
N-N3
P-N3
P-Q4.!
PXP
P/KXP
N-K5
B-K3
BXN
RXB
P-B4. SMYSLOV-ILIVITSKY 1955 USSR CHAMPIONSHIP.
R/QR1-B1
SET MOVE8
R-K1.!
N/QN1-Q2
SET MOVE8
P-N3
N-R3
B-N2
Q-R4
P-K3
P-KN4
N-K1
Q-R3
N-Q3
P-K4
PXP
N-KN5
P-KR3
NXP/4
SET MOVE3 SOLTIS 2D
P-K3
P-QN3
SAVE MOVE4 SOLTIS 2D1
P-B4
B-N2
SAVE MOVE5
N-B3.?!
B-N5
SET MOVE5
B-Q3
P-N3.!?
SET MOVE5
B-K2
B-N5.CHECK
B-Q2
P-QR4.!?
O-O
O-O
N-B3
Q-K2.?!
P-QR3
B/QN5XN
BXB
N-K5
R-B1
P-R5
N-Q2
NXB
RXN
P-B4.=
```

```
SET MOVE4 SOLTIS 2D2
B-K2
B-N2
N/QN1-Q2
B-Q3
SAVE MOVE6
N-K5
P-B4
SET MOVE6
O-O
O-O
P-QN3
N-B3
B-N2
N-K2
SAVE MOVE9
P-B4.?!
N-N3
Q-B2
N-K5
NXN
BXN
Q-B3
Q-K2
P-QR3.?!
N-R5
NXN
BXP/R7.CHECK!
KXB
QXN.CHECK
K-N1
BXP.! AND WHITE WON A SMASHING VICTORY. LASKER-BAUER 1889.
SET MOVE9
N-B4
N-N3
NXB
PXN
P-B4. = (EUWE)
SW EXP ON
W/RUYW
*CWEOR
MSG       RUY LOPEZ WITH 3....P-QR3
INI
LET MSMASK=77B.
LET MSCODE=0.
SWITCH EXPERIENCE,OFF
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH LW ON;SWITCH NOR ON
YES
P-K4
P-K4
N-KB3
N-QB3
LET MSCODE=40B.   DISABLE HALF THE TIME.
B-N5
LET MSCODE=0
P-QR3
SAVE,BASE
B-R4
P-QN4
B-N3
N-R4.  WING VARIATION
P-Q4.  VARIATION 39,COTAP-111
P*P
Q*P
N*B
P(R2)*N
N-K2
B-N5.  IDEA VARIATION 21, COTAP-110
P-KB3
B-K3
N-B3
Q-Q5
B-K2
O-O
B-N2
P-B4
INI
SETUP,BASE
B-R4
P-QN4
B-N3
B-B4.  GRAZ VARIATION
P-B3.  VARIATION 49, COTAP-113
P-Q3
P-Q4
P*P
P*P
B-N3
P-QR4
R-N1
N-B3
P-N5
N-Q5
N-R4
N*B
N*B
Q*N
R*N
B-K3
N-K2
P-Q5
R-QN1
O-O
O-O
N-Q4
K-R1
N-B6
INI
SETUP,BASE
B-R4
P-B4.  SCHLIEMANN DEFENSE DEFERRED
N-B3.  VARIATION 59, COTAP-117
P-QN4
B-N3
P-N5
N-Q5
P*P
SAVE,21
P-Q4
P*N
SAVE,23
Q*P
B-K2
N*P(B7).CHECK
Q*N
Q-B7.CHECK
K-Q1
Q*P
P-Q4
Q*R
B-K3
P*P
Q*P.CHECK
Q*Q
N*Q
B-KB4
N-N3
B-N3
INI
SETUP,23
Q*P
N-B3.  NOTE 23, COTAP-118
P*P
N*N
B*N
N*P
Q-R5.CHECK
N-N3
SAVE,SUB1
O-O
P-B3
R-K1.CHECK
B-K2
B-N5
P*B
B*B
Q*B
R*Q.CHECK
K*R
Q*P(Q5).
R-QN1
R-K1.CHECK
INI
SETUP,SUB1
O-O
B-K2.  ALSO NOTE 23, COTAP-118
R-K1
K-B1
R*B
Q*R
B-N5
Q-K4
Q-B3.CHECK
INI
SETUP,21
P-Q4
P-Q3.  NOTE 21, COTAP-118
B-N5
B-K2
B*B
N(N1)*B
N-N5
N*P
Q-R5.CHECK
K-Q2
N-B7
Q-K1
B-R4.CHECK
INI
SETUP,BASE
B-R4
P-Q3.  NEO-STEINITZ DEFENSE
SAVE,VARY
P-B3.  VARIATION 61,COTAP-121
P-B4
P*P
B*P
P-Q4
P-K5
N-N5
P-Q4
SAVE,5
P-B3
P-K6
SAVE,7
P-KB4
B-Q3
Q-R5.CHECK
P-N3
Q-B3
Q-B3
Q*P(K3).CHECK
N-K2
O-O
O-O
INI
SETUP,7
P-KB4
N-B3.  NOTE 7, COTAP-122
N-B3
B-KN5
B*N.CHECK
P*B
O-O
INI
SETUP,5
P-B3
P*P.  (Q) NOTE 5, COTAP-122
O-O.  KERES
INI
SETUP,VARY
P-B3
B-Q2.  VARIATION 62, COTAP-121
SAVE,V63
P-Q4
P-KN3
O-O
B-N2
SAVE,11
P*P
P*P
B-KN5
N(N1)-K2
Q-Q3
P-R3
SAVE,12
B-K3
N-R4
B*B.CHECK
Q*B
Q*Q
K*Q
N-R3
```

```
K-K3.  WADE-BRONSTEIN, AMSTERDAM 1954
INI
SETUP,12
B-K3
B-N5.  NOTE 12, COTAP-122
Q-K2
O-O
B-B5.  ALEKHINE
INI
SETUP,11
P*P
N*P. (Q)  NOTE 11, COTAP-122
N*N
P*N
P-KB4
N(N1)-K2
P-B5
P-KB3
B-K3
B*B
Q*B.CHECK
Q-Q2
Q*Q.CHECK
K*Q
P-B4.  KERES-GOLDENOV, 1952
INI
SETUP,V63
P-Q4
N(N1)-K2.  VARIATION 63, COTAP-121
B-N3
P-R3
N(N1)-Q2
N-N3
N-B4
B-K2
O-O
O-O
SAVE,17
N-K3
R-K1
R-K1
B-KB1
B-B2
N-R5
N*N
Q*N
N-Q5
R(R1)-B1
R-B1
N-K2
N-K3
N-N3
P-KN3
Q-R6
P-KB4
P*P(B5)
P*P
P-KB4
INI
SETUP,17
N-K3
B-B3.  NOTE 17, COTAP-122
N-Q5.  IDEA VARIATION 27, COTAP-120
R-K1
SAVE,SUB1
P*P
B*P
N*B
P*N
Q-B3.  SMYSLOV-RESHEVSKY, MOSCOW 1948
INI
SETUP,SUB1
P*P
N(N3)*P.  VARIATION TO IDEA VARIATION 27, COTAP-120
N*N
N*N
P-KB4
INI
SETUP,V63
P-Q4
N-B3.  NOTE 14, COTAP-122
O-O
INI
SETUP,VARY
P-B3
N-B3.  LOPEZ GAME 26, COTAP-123
P-Q4
N-Q2
P-QR3
B-K2
P-QN4
O-O
O-O
B-B3
B-K3
P-QN4
P-Q5
N-K2
B-N3
N-QN3
N(N1)-Q2
B-N2
P-B4
P*P
N*P(B4)
N*N
B*N
P-B3
P*P
B*P
N-Q2
P-Q4
P*P
N*P
B-B5
B-K2
N-K4
B*B
N*B
N-B5
Q*Q
R(B1)*Q
B*P(R6)
B*P
R(B1)-Q1
B-B6
R*R.CHECK
R*R
B-N7
N-R6.CHECK
K-B1
B*B
N*B
R-Q7
P-R4
R*P.CHECK
K-K1
R*P
R-R3
R-R8.CHECK
K-K2
N-B5.CHECK
K-B2
P-R4
P-R5
N-Q4
P-R6
N-N3
R-R5
P-B3
P-R7
R-R7.CHECK
K-N3
R-QN7
R-N5
N-R1
N-Q6
K-R2
R-N8
N-B2
R-QB8
P-R5.CHECK
K*P
R*P.CHECK
K-N3
R-N6.CHECK
K-B2
R-N7.CHECK
K-K1
R-N8.CHECK
K-Q2
R-N3
N-K8.  TAL-AVERBACH, THILISI 1959
INI
SETUP,BASE
B-R4
P-KN3.  SUPP VARIATION 6, COTAP-125
SAVE,V7
P-Q4
N*P
N*N
P*N
Q*P
Q-B3
P-K5
INI
SETUP,V7
P-Q4
P*P.  SUPP VARIATION 7, COTAP-125
N*P
B-N2
B-K3
N(N1)-K2
N-QB3
INI
SETUP,BASE
B-R4
N(N1)-K2
P-Q4
P*P
N*P
N*N
Q*N
N-B3
Q-Q3
INI
SETUP,BASE
B-R4
LET MSCODE=22B  DISABLE 87.5 PERCENT OF THE TIME
N-B3.  SUPP VARIATION 10, COTAP-126
SAVE,2
P-Q4
LET MSCODE=0.
P*P
O-O
B-K2
P-K5
N-K5
SAVE,8
N*P
N*N
Q*N
N-B4
N-B3
O-O
B-K3
N*B
Q*N
P-Q4
P*P(Q6).EP
B*P
R(R1)-Q1.  LOMBARDY-HOROWITZ, NEW YORK 1955-56
INI
SETUP,8
N*P
O-O.  LOPEZ GAME 31, COTAP-128
N-B5
P-Q4
P*P
B*N
P*B
Q*P
B*N
P*B
Q-B3
Q-B3
N-R3
Q-N3
B-B4
N-Q3
P-B3
R(B1)-K1
R(B1)-K1.
R-K5
```

```
R*R
B*R
Q-N3.  GUFELD-KERES, THILISI 1959
INI
SETUP,8
N*P
N-B4.  NOTE 8, COTAP-127
N-B5
O-O
Q-N4
P-KN3
B*N
P(Q2)*B
N*B.CHECK
Q*N
Q-N5
R-K1
Q*Q
R*Q
P-KB4
B-B4
N-R3
P-B3
P*P
R-K3
R-B2
R-K8.CHECK
R-B1
R-K7.  OKELLY-SMYSLOV, 1955
INI
SETUP,2
P-Q4
N*P(K5).  NOTE 2, COTAP-126
O-O.  TRANSPOSINT INTO OPEN DEFENSE
INI
SETUP,2
P-Q4
N*P(Q5).  NOTE 2, COTAP-126
N*N
P*N
P-K5
N-K5
Q*P
N-B4
N-B3
B-K2
Q-KN4
K-B1.  SZABO-PACHMAN,1948
INI
SETUP,BASE
B-R4
N-B3
O-O
P-Q3.  RUSSIAN DEFENSE
SAVE,VARY
P-B3.  VARIATION 68, COTAP-132
B-Q2
SAVE,2
P-Q4
B-K2
R-K1
O-O
SAVE,3
N(N1)-Q2
B-K1
B-N3
N-Q2
N-B1
B-B3
N-K3
N-K2
N-N4
N-KN3
P-N3
B-K2
P-KR4.  SMYSLOV-LJUBINSKI, MOSCOW 1949
INI
SETUP,3
N(N1)-Q2
R-K1.  NOTE 3, COTAP-133
B-N3
P-R3
N-B1
B-KB1
N-N3
N-QR4
B-B2
P-KN3
N-Q2
P-B4.  NESHMETDINOV-SZABO, BUCHAREST 1954
INI
SETUP,3
N(N1)-Q2
P*P.  VARIATION 69, COTAP-132
P*P
N-QN5
B*B
Q*B
SAVE,7
N-B1
P-Q4
N-K5
Q-K1
P-QR3
N-B3
N*N
Q*N
P-K5
N-K5
P-B3
N-N4
P-B4
N-K5
P-B5
INI
SETUP,7
N-B1
P-B4.  IDEA VARIATION 30, COTAP-131
P-QR3
N-B3
P-Q5
N-K4
N*N
P*N
N-K3.  BRONSTEINS RECOMMENDATION
N*P(K5)
N-B4
Q-B4
Q-B2
N-Q3
Q*Q
N*Q
R*P
N-Q3
INI
SETUP,2
P-Q4
N*P(K5).  NOTE 2, COTAP-132
R-K1
N-B3
B*N
B*B
P*P
P*P
Q*Q.CHECK
R*Q
N*P
B-K5
B-N5. (E)
B-K2
N-Q2
B-Q4
N(Q2)-B3
O-O
N-Q4
R(B1)-K1
N-B5
B-K3
N*B.CHECK
R*N
N-N4.  NESHMETDINOV-SLIWA, BUCHAREST, 1954
INI
SETUP,VARY
P-B3
N*P.  VARIATION 71, COTAP-132
P-Q4
B-Q2
R-K1
N-B3
SAVE,10
P*P
P*P
B*N
B*B
SAVE,15
Q*Q.CHECK
R*Q
N*P
B-K5
N(N1)-Q2
B-K2
N*B
N*N
SAVE,18
B-R6
N*P(QB6)
B*P
R-KN1
N-B6
P*N
B*N
R-N3
R-K4.  BOLESLAVSKI-SLIWA,1955
INI
SETUP,18
B-R6
P*B.  NOTE 18, COTAP-133
R*N
INI
SETUP,15
Q*Q
K*Q.  BETTER.  NOTE 15, COTAP-133
N*P
B-Q4
INI
SETUP,10
P*P
N*P.  NOTE 10, COTAP-133
B-B4
B*B
Q*B.CHECK
Q-Q2
Q-Q4
B-K2
B*N
P*B
Q*P
K-KB1
N(N1)-Q2.  PILNIK-AVERBACH, 1952
INI
SETUP,BASE
B-R4
N-B3
O-O
N*P
P-Q4
P-QN4
B-N3
P-Q4
N*P
N*N
SAVE,VARY
P*N.  OPEN DEFENSE I, COTAP-135
B-K3.  VARIATION 75, COTAP 136
B-K3
B-QB4
B*B
N*B
P-KB4
P-KN3
N-Q2
O-O
N-B3
N*B
P(R2)*N
P-QB4
Q-Q2
P-N5
Q-B2
Q-N3
Q-R4.  KIENINGER-BOGOLJUBUV, 1951
INI
SETUP,VARY
P*N
```

```
P-QB3.  VARIATION 77, COTAP-136
SAVE,V78
P-QB3
B-KB4. (Q)
SAVE,7
B-B2
B-B4
Q-B3
INI
SETUP,7
B-B2
Q-Q2.  IDEA VARIATION 33, COTAP-135
P-QR4
R-B1
P*P
P(R3)*P
B-K3
N-B4
N-Q2
B*B
Q*B
B-K2
P-KB4
O-O
P-B5
R(KB1)-K1
B-Q4
B-B1
R-B3
N-K5
N*N
P*N
Q*P
P-B4
B-B2
B-Q3
R-Q1
Q-K2
P-K6
P*P
B-R4
Q-QB2
P-B6.  BRONSTEIN-PACHMAN, PORTOROZ 1958
INI
SETUP,V78
P-QB3
B-K2.  VARIATION 78, COTAP-136
B-K3
O-O
N-Q2
N*N
Q*N
B-KB4
R(B1)-K1
Q-Q2
R(R1)-Q1
Q-K3
B-N5
B-B4.  TRIFUNOVIC-DONNER, WAGENINGEN 1957
INI
SETUP,VARY
P*N
B-N2.  IDEA VARIATION 34, COTAP-135
P-QB3
B-B4
N-Q2
Q-K2
N*N
P*N
P-K6
P-KB3
B-KB4
R-Q1
Q-N4
B-Q3
R(R1)-Q1
O-O
P-KR4
B*B
Q*B
P-QB4.  DARGA-MILEV, MUNICH 1957
INI
SETUP,BASE
B-R4.  CLOSED DEFENSE, CLASSICAL VARIATION
N-B3
O-O
B-K2
R-K1
P-QN4
B-N3
P-Q3
SAVE,A
P-B3
O-O
P-KR3
N-QR4
B-B2
P-B4
P-Q4
Q-B2
SAVE,VARY
N(N1)-Q2
N-B3.  VARIATION 106, COTAP-160
P*P(B5)
P*P
SAVE,3
N-B1
B-Q3
SAVE,4
N-R4
N-K2
SAVE,5
Q-B3
B-K3
N-N3
Q-B3
N(R4)-B5
N*N
N*N
B*N
Q*B
N-K1.  TAL-FILIP, REYKJAVIK 1957
INI
SETUP,5
Q-B3
R-Q1.  LOPEZ GAME 60, COTAP-164
N-K3
Q-N2
P-KN4
P-B5
N(R4)-B5
N*N
N*N
B*N
P(N4)*B
K-R1.  TAL-FILIP, PORTOROZ 1958
INI
SETUP,4
N-R4
P-N3.  NOTE 4, COTAP-160
B-R6
R-Q1
Q-B3
N-K1
N-K3
P-B3
N-Q5
Q-B2
N-N6.  SMYSLOV-BOTVINNIK, 1957
INI
SETUP,3
N-B1
B-K3.  IDEA VARIATION 47, COTAP-159
N-K3
R(R1)-Q1
Q-K2
P-N3
N-N5
B-B1
B-Q2
K-N2
R(R1)-Q1
P-R3
N-B3
B-K3
P-QR4
Q-N1
B-B1
R*R
R*R
R-Q1
R*R
B*R
P*P
P*P
N-Q5.  EUWE-SMYSLOV, THE HAGUE 1948
INI
SETUP,VARY
N(N1)-Q2
P(B4)*P.  VARIATION 108, COTAP-160
SAVE,10
P*P
B-N2
SAVE,12
P-Q5
B-B1
R-N1
P-N5
N-B1
N-N2
B-K3
B-Q2
R-B1
R(B1)-B1
Q-Q2
Q-R4
B-N1
B-Q1
N-N3
R*R
R*R
R-B1
R*R
B*R
Q-B1.  GLIGORIC-RESHEVSKY, 1952
INI
SETUP,12
P-Q5
R(B1)-B1.  NOTE 12, COTAP-161
B-Q3
N-Q2
N-B1
N-B4
N-K3
N*B
Q*N
N-B5
N-B5
B-B1
P-QN3
N-N3
B-Q2.  ARONIN-TOLUSH,1950
INI
SETUP,10
P*P
N-B3.  IDEA VARIATION 48, COTAP-159
SAVE,SUB1
N-N3
P-QR4
B-K3
P-R5
N(N3)-Q2
N-QN5
B-N1
B-Q2
P-R3
N-B3
B-Q3
Q-N1
P-QN4.  BOLESLAVSKI-GOLDENOV, LENINGRAD 1947
INI
SETUP,SUB1
N-N3
B-N2.  LOPEZ GAME 61, COTAP-164
B-N5
P-R3
B-R4
N-QN5
B-N1
R(R1)-B1
R-K2
N-R4
P-R3. (E)
N-QB3
```

```
P-Q5
N-N1
R-B2
Q-Q1
N-R5
R*R
N*B
Q-B2
Q*R. (EE)
Q*N
B*B
R-B1
B*P
R*Q
B*R
P-B3
B-N3
N-B5
R-Q1
N-Q2
R-Q2
N-QN3. (E)
B-B7
N-B5
P-Q6
N-K3
B-R5. (E)
N-B4. (Q)
B-N4
N-Q2
R-B2
P-QR4
B*P
Q*P
N-Q2. (E)
Q-Q6
R*N
K-R2
B-B2.  UNZICKER-KERES, MOSCOW 1956
INI
SETUP,VARY
N(N1)-Q2
B-Q2.  VARIATION 109, COTAP-160
N-B1
R(B1)-K1
N-K3
P-N3
P*P(B5)
P*P
N-R2
B-K3
Q-B3
R(R1)-Q1
N(R2)-N4
N*N
P*N
Q-B3
P-KN5.  LOPEZ GAME 64, COTAP-166
N-B5
N-N4
B*N
Q*B
P-B3
P*P
B*P
P-R4
N-N3
P*P
P*P
B-K3
R-R1
R(K1)-Q1
K-R1
P-QN3
B-N2
Q-R4
B-B3
B-N5
B*B
Q*B
R*R
R*R
N-Q2
B-Q1
N-B3
R-R7
Q-Q3
B-K2
R-K2
R*R
Q*R
B*P
K-N2
B-K2
Q-QB2
Q-K3
Q-R4
P-N3
Q-R6
K-N2
Q-R4
Q-Q3
Q-N3
Q-B4
Q-B3
B-Q3
Q-N3
P-QN4
P*P
P*P
N-N5
Q-B5
Q*Q
P*Q
K-B2
P-B4
K-K2
K-B3
N-B3
B-N5
K-K3
B-B4.CHECK
K-K2
P-B6
N-K1
P*P
P-R3
K-K3
N-B2
K-Q4
P-R4
K-K3
P-N4
B-K2
P-R5
P*P
P*P
B-B4
N-K1
K-B4
K-Q1
K-N4
K-B2
B-B7
N-N2
K*P
K*P
K-N5.  FISCHER-UNZICKER, ZURICH 1959
INI
SETUP,VARY
N(N1)-Q2.  NOTE 15, COTAP-162
R-K1
P-QN4
P*P(N5).
P*P(N4).
N-B3
B-N2
N*P(N5).
B-N3
N-Q6
B*P.CHECK TAHL-GURGENDZE, BAKU 1961
INI
SETUP,VARY
N(N1)-Q2
B-N2.  VARIATION 113, COTAP-161
N-B1
P(B4)*P
P*P
R(R1)-B1
SAVE,30
B-Q3
P-Q4
P(Q4)*P
N*P
SAVE,32
N-N3
P-B4
P*P.EP
B*P
N*N
P*N
B*P(K4)
R(KB1)-Q1
Q-K2
R-K1
N-Q2
Q-Q2
Q-B1
N-B3
N-B3
N-Q5
B*B
Q*B
N*N
B*N.  GELLER-KERES, AMSTERDAM 1956
INI
SETUP,32
N-N3.  NOTE 32, COTAP-162
N-B5
R-K2
P-B4
P*P.EP
N*N
P*N
Q*P
B-B5
P-Q5
B-K6.CHECK
K-R1
B*R
B*N
Q-Q3
R*B
Q*B
Q*Q
P*P.CHECK  PETROSIAN-TRIFUNOVIC 1958
INI
SETUP,30
B-Q3
N-B3.  LOPEZ GAME 68
N-K3
R(KB1)-K1
N-B5
B-B1
B-N5
N-Q2
R-QB1
Q-N1
B-N1
N*P
N(B3)*N
R*R
B*R
P*N
N-R6.CHECK
P*N
Q-N4.CHECK
K-R1
Q*N
B-Q4
Q-B5
R-K4
Q-B3
P-B4
B-B4
R-K1
Q-R5
B*P(K5).
P-B3
B-B3
R-QB1
B-Q2
B*P(R6).
R-K3
```

```
B*B
Q*B
Q-R4
Q-B3
Q*Q.CHECK
R*Q
K-B2
K-N2
R-B7
R-B2
K-K2
P-B5
R-R7
K-B3
R*P
R-K2.CHECK
K-B2
B-K3
R*P
K-K4
R-B6
B-Q4
R-KR6
R-QB2
R-R5.CHECK
K-Q3
R-R6.CHECK
K-K4
R-R5.CHECK
K-Q3
R-B5
R-B8
B-Q3
R-Q8
K-K2
R-KN8
K-B2
R-Q8
K-K2
R-KN8
R-N5
B*P(R7)
B*P(N5).
R-N8
K-Q3
P-R3
R-R5
R*P
K*P
R*P
R*P.CHECK
K-K2
K-K4
R-N4
B-R6
B-B2
B-B8
R-N3
R-R7
K-B1
B-N4
R-N2
R-R6
R-N3
R*R
B*R.CHECK
K*P
K-N2
K-N5
B-Q6
P-B4
B-K5
P-R4
B-Q6
P-R5
B-K5
P-R6.CHECK
K-R1
B-B5
B-Q4
B-N6
B-K3
K-B6
B-B5
K-N5
B-K3
B-R5
K-R2
B-N4
B-B5
P-B5
B-B2
B-R5
B-B5
B-N6.CHECK
K-N1
P-B6.  FISCHER-KERES, ZURICH 1959
INI
SETUP,VARY
N(N1)-Q2
R-Q1.  VARIATION 114, COTAP-161
N-B1
P-Q4
N-N3
P*P(K5)
N(B3)*P
P*P
SAVE,36
P*P
B-Q3
Q-K2
B*N
P*B
Q*P
N*P
B-B4
B-N5
B*N
B*N
Q*B.  NILSSON-KERES, AMSTERDAM 1954
INI
SETUP,36
P*P
B-K3.  NOTE 36, COTAP-163
N*P(K4).
R(R1)-B1
R-K2
N-B5
N-KB3
B-Q4
N*N.CHECK
B*N
B-K4.  GLIGORIC-MILIC. 1953
INI
SETUP,A
P-B3
B-N5.  NOTE A, COTAP-158
P-Q3
O-O
N(N1)-Q2
INI
SETUP,BASE
B-R4
N-B3
O-O
B-K2
R-K1
P-QN4
B-N3
O-O
P-B3
P-Q4
SAVE,VARY
P*P
P-K5.  VARIATION 125, COTAP-180
P*N
P*N
SAVE,V126
P-Q4
P*P
B-N5
Q-Q3
Q-B3
N-N5
B-KB4
Q-B3
N-Q2
B-Q3
R-K4
B*B
R*B.  GILMAN-ESTRIN, 1949
INI
SETUP,V126
P-Q4
B-Q3.  VARIATION 126, COTAP-180
B-N5
B*P.CHECK
K*B
N-N5.CHECK
K-N1
Q*B
Q*P
B-B4
N-Q2.  KOFMAN-KOPAJEV, 1948
INI
SETUP,VARY
P*P
N*P.  VARIATION 129, COTAP-180
N*P
N*N
SAVE,V133
R*N
N-B3
P-Q4
B-Q3
R-K1
N-N5
P-KR3
Q-R5
SAVE,V131
Q-B3
N*P
B-Q2
B*P
P*B
N*P.CHECK
K-B1
INI
SETUP,V131
Q-B3
P-KR4
N-Q2
B-N2
N-K4
R(QR1)-K1
B-N5
B*N
R*B
B-R7.CHECK
K-B1.  BOLESLAVSKI-SCHAMKOWITSCH, 1956
INI
SETUP,V133
R*N
N-N3.  VARIATION 133, COTAP-181
P-Q4
B-Q3
B-N5
Q-Q2
R-K1
B-N2
N-Q2
R(QR1)-K1
P-B3
Q-B4
B-KR4
P-B4
N-B1
P*P
P*P
N-Q4
B-N3
N-B5
Q-Q2
R-Q1.  NEDELJKOVIC-GELLER, 1957
INI
SETUP,V133
R*N
P-QB3.  VARIATION 137, COTAP-181
B*N
P*B
P-Q4
B-Q3
SAVE,V138
R-K3
```

```
P-B4
SAVE,32
N-Q2
P-B5
SAVE,33
R-K1
P-B6
N*P
B-KN5
SAVE,34
R-K3
R-R2
P-KR3
INI
SETUP,34
R-K3
B-B5.  NOTE 34, COTAP-182
R-Q3
B-B4
B*B
B*R
B-N5
Q*B
INI
SETUP,33
R-K1
P-N4.  NOTE 33, COTAP-182
Q-R5
Q-B3
P-KR4
P-R3
N-B3
R-R2
B-Q2
R-KN2
R-K8.  NILSSON-NYMAN
INI
SETUP,32
N-Q2
R-R2.  NOTE 32, COTAP-182
Q-N3.  (E)  L. SCHMID-EIDENFELDT
INI
SETUP,V138
R-K3
Q-R5.  VARIATION 138, COTAP-181
SAVE,35
P-KR3
B-K3
N-Q2
R(QR1)-K1
N-B3
Q-R4
B-Q2
P-B3
P-QR4.  (E)  NILSSON-AHMAN, 1954
INI
SETUP,35
P-KR3
B-B5.  (Q)  NOTE 35, COTAP-182
P-KN3
Q-N4
R-K5.  (E)
W/RUYWO
MSG  RUY LOPEZ WITHOUT 3....P-QR3
INI
LET MSMASK=77B.
LET MSCODE=0.
SWITCH EXPERIENCE,OFF
SWITCH LW ON;SWITCH NOR ON
YES.
P-K4
P-K4.
N-KB3
N-QB3
B-N5
N-B3.  BERLIN DEFENSE I, COTAP-85
SAVE,A
O-O
N*P
SAVE,C
P-Q4
B-K2
SAVE,E
Q-K2
N-Q3
SAVE,F
B*N
P(N2)*B
SAVE,G
P*P
N-N2
SAVE,VARY
N-B3.  PRACTICAL VARIATION 2, COTAP-86
O-O
N-Q4.  SUGGESTED BY FINE
B-B4
R-Q1
B*N
R*B
P-Q4
P*P(Q6). EP
P*P
SAVE,V3.
P-QN4
Q-B3
B-K3
B-B4
R(R1)-Q1
P-QR3
P-N4.  SCHLECTER-RETI, 1914
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,V3
P-QN4
SWITCH LW ON;SWITCH NOR ON
R-K1.  VARIATION 3, COTAP-87
B-K3
B-K3
SAVE,11
Q-B3
Q-Q2
SAVE,12
N-K4
B-Q4
P-B4
B*N
R*B
P-QR4
INI
SETUP,12
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
N-K4
SWITCH LW ON;SWITCH NOR ON
B-B4.  (Q)  LOPEZ GAME 2, COTAP-88
N-N3
B*P
R-QB1
B-R5
N-R5
P-KB4
R-KB4
R-K2
R*P(KB5)
R-B2
N*P
R*N
B-R6
Q-K2
B*R
Q*B
P-R4
P-KR3
R-QB4.  KERES-UNZICKER, 1956
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,11
Q-B3
SWITCH LW ON;SWITCH NOR ON
P-Q4.  NOTE 11, COTAP-87
SAVE,11SUB
P-N5
P*P
N*P(Q5)
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,11SUB
P-N5
SWITCH LW ON;SWITCH NOR ON
P-QB4
R-Q3
Q-N3
N*P
B*N
Q*B
Q*P. (Q)
R-N3.
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,VARY
N-B3
SWITCH LW ON;SWITCH NOR ON
N-B4.  INFERIOR.  NOTE 1, COTAP-87
N-Q4
B-R3
Q-N4
B*R
Q*P(N7)
R-KB1
K*B
N-K3
N*N
P(B2)*N
Q*P
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,G
P*P
SWITCH LW ON;SWITCH NOR ON
N-B4.  NOTE H, COTAP-86
Q-K4
P-N3
INI
SETUP,F
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
B*N
SWITCH LW ON;SWITCH NOR ON
P(Q2)*B.  NOTE F, COTAP-85
P*P
N-B4
R-Q1
B-Q2
P-K6
P*P
N-K5
B-Q3
Q-R5.CHECK
P-N3
N*P(N6)
N-N2
Q-R6
N-B4
Q-R3
R-KN1
Q*P
R-N2
Q-R5
Q-B3
N-K5.CHECK
K-K2
N-N4
Q-N3
Q*Q
R*Q
P-KR3
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,E
Q-K2
SWITCH LW ON;SWITCH NOR ON
P-Q4.  NOTE E, COTAP-85
N*P
B-Q2
B*N
P*B
R-K1
O-O
```

```
P-KB3
INI
SETUP,E
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
Q-K2
SWITCH LW ON;SWITCH NOR ON
P-B4.  DIFFICULT FOR BLACK, NOTE E, COTAP-85
P*P
O-O
N-B3
N*N
Q-B4.CHECK
K-R1
Q*N(B6)
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,C.
P-Q4
SWITCH LW ON;SWITCH NOR ON
N-Q3.  BERLIN DEFENSE II. COTAP-90
SAVE,D
B*N
P(Q2)*B
P*P
N-B4
Q*Q.CHECK
K*Q
SAVE,2
N-B3
K-K1
N-K2
B-K3
N-B4
B-Q4
N*B
P*N
P-KN4
N-K2
SAVE,V8
B-B4
N-N3
B-N3
B-B4
K-N2
N-B1
N-K1
P-KN4
N-Q3
N-K3
P-KB4
P*P
N*P
N*N
R*N
R-KN1
R(R1)-KB1
R-N2
K-B3
B-K2
R-Q1
P-QB3
P-B4
P*P
R*P(QB4)
R-Q1
R*R
B*R
K-B4
P-KR4
P*P
R-N4
R-N4
P-N3
R-R4
P-R4
R-Q4
R*P(R4)
R-Q6
P-QB4
K-K4
B-B2
R-QB6
K-Q2
R-B6
K-K2
K-Q5
B-Q1
P-QR4
K-K1
R-B4
P-B3
R-R4
R-N4
K-K6
R-N1
P*P.  PANNO-BENKO  1958.  COTAP-91
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,V8
B-B4
SWITCH LW ON;SWITCH NOR ON
P-QB4.  VARIATION 8, COTAP-90
SAVE,5
R(R1)-Q1
P-KR4
P-KR3
P*P
P*P
P-Q5
P-B3
N-B3
R(B1)-K1
B-K2
P*P
P*P
P-QR3
P-QR4
N*P
R-R5
B-N3
R*P
N-N5
R-Q1
N-Q6.CHECK
B*N
P*B.CHECK
K-Q2
R-Q5
P-B3
K-N2
P-KN4
R-QN5
R-QN1.
R-KR1
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,5
R(R1)-Q1
SWITCH LW ON;SWITCH NOR ON
R-Q1.  IDEA VARIATION 5, COTAP-90
P-B3
N-B3
R-Q2
P-Q5.  GLIGORIC-SANGUINETTI, PORTOROZ 1958
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,2
N-B3
SWITCH LW ON;SWITCH NOR ON
B-K2.  IDEA VARIATION 4, COTAP-90
B-N5
P-KR4
R(R1)-Q1.CHECK
K-K1
N-K2
B-K3
N-B4
P-B4
B*B
K*B
N-N5
B-B5
R(B1)-K1
N-Q5
P-K6
N*P(K3)
N(N5)*N
B*N
N-Q5.CHECK.  SCHMID-TORAN, MUNICH 1958
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,2
N-B3
SWITCH LW ON;SWITCH NOR ON
P-KR3.  NOTE 2, COTAP-91
B-Q2
B-K3
N-K2
P-B4
B-B3.  TARRASCH-LASKER, 1895
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
YES
P-K4
P-K4
N-KB3
N-QB3
B-N5
B-B4.  CORDEL DEFENSE.  COTAP-92
SWITCH LW ON;SWITCH NOR ON
SAVE,A
P-B3
N-B3
SAVE,B
O-O
O-O
P-Q4
B-N3
P*P
N*P(K5)
Q-Q5
N-B4
B-N5
N-K2
Q-Q1
N-K5
B-KR4
P-Q4
N(N1)-Q2
P-QB3
B-Q3
P-KB4
P*P.EP
N*P(B3)
Q-B2
P-N3
R(R1)-K1
B-KB4
N-Q4.  BRONSTEIN-OKELLY,  HASTINGS 1953-54
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SETUP,A
SWITCH NOR,ON
P-B3
SWITCH LW ON;SWITCH NOR ON
P-B4.  NOTE B, COTAP-92
B*N
P(Q2)*B
N*P
B-Q3
P-Q4
P*P
O-O
N-B3
B-N5
Q-K2
N-Q2
B-KB4
R-K1
O-O-O
N*P(K4)
B*N(K4)
N-N3.  UNZICKER-EISINGER, GERMAN CHAMPIONSHIP, 1953
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
YES
```

```
P-K4
P-K4
N-KB3
N-QB3
B-N5
N-Q5.  BIRDS DEFENSE.  COTAP-94
SWITCH LW ON;SWITCH NOR ON
N*N
P*N
SAVE,A
O-O
P-QB3
SAVE,B
B-B4
N-B3
R-K1
P-Q3
P-Q3
B-K2
N-Q2
O-O
P-B4
P-Q4
P*P
B-QN5
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,B
B-B4
SWITCH LW ON;SWITCH NOR ON
P-Q4.  NOTE B, COTAP-94
P*P
P*P
R-K1.CHECK
B-K2
B-N5.CHECK
B-Q2
Q-N4
INI
SETUP,B
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
B-B4
SWITCH LW ON;SWITCH NOR ON
N-K2.  LOPEZ GAME 7, COTAP-95
P-Q3
P-Q4
B-N3
P*P
P*P
N-N3
P-QB3
B-QB4
Q-R5
Q-K2
B-N5
Q*P
N-Q2
Q-N5
R(R1)-K1.CHECK
K-B1
Q*Q
B*Q
P*P
B-K2
P-B4
B-KB4
B*B
N*B
R-B3
R-Q1
R(B3)-K3
R-Q2
N-B4
B-K3
N-K5
R*P
N*P(B7)
K*N
R*B
N-Q4
R-K7.CHECK
K-B3
R(K1)-K6.CHECK
K-B4
B-B2
K*P
R*P(KN7)
R-KB1
R*P(R7)
N-K6
R-R3
N-B4
R-KB3.CHECK
K-N4
R-K5
R-KB5
K-B2
R-B3
P-KR4.CHECK
K-N5
R*R.CHECK
K*R
R-K4.CHECKMATE.  ALEKHINE-GRIGORIEV, MOSCOW 1919
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,A
O-O
SWITCH LW ON;SWITCH NOR ON
N-K2.  NOTE A, COTAP-94
P-QB4
P-KN3
P-Q3
B-N2
B-B4
P-QB3
B-R4
P-Q3
Q-B1
O-O
B-R6
Q-R4.  ORBAAN-DR. SCHOLTENS, GRONINGEN 1953
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
YES
P-K4
P-K4
N-KB3
N-QB3
B-N5
P-B4.  SCHLIEMANN DEFENSE
SAVE,A
SWITCH LW ON;SWITCH NOR ON.
N-B3
P*P
SAVE,VARY
N(QB3)*P
N-B3.  RUY LOPEZ VARIATION 17.  COTAP-98
SAVE,1
N*N
P*N
SAVE,2
P-Q4
P-Q3
P-Q5
P-QR3
B-K2
N-K2
SAVE,4
N-R4
P-B3
B-R5.CHECK
K-Q2
P*P.CHECK
P*P
P-QB4
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,4
N-R4.
SWITCH LW ON;SWITCH NOR ON
N-N3.  NOTE 4, COTAP-98
B-R5
R-KN1
Q-Q3
K-B2
O-O
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,2
P-Q4
SWITCH LW ON;SWITCH NOR ON
P-K5.  IDEA VARIATION 13, COTAP-97
SAVE,SUB1
N-N5
B-N5.CHECK
SAVE,SUB2
P-B3
P*N
Q-R5
K-B1
SAVE,SUB3
B*P
B-K2
B-KR6.CHECK
K-N1
B-QB4.CHECK
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,SUB3
B*P
SWITCH LW ON;SWITCH NOR ON
Q-K1
Q-R6.CHECK
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,SUB3
B*P
SWITCH LW ON;SWITCH NOR ON
N-K2
B-QB4
P-Q4
B*P
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,SUB2
P-B3
SWITCH LW ON;SWITCH NOR ON
O-O
P-Q5
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,SUB1
N-N5
SWITCH LW ON;SWITCH NOR ON
P*N
Q-R5.CHECK
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,1
N*N
SWITCH LW ON;SWITCH NOR ON
Q*N.  NOTE 1, COTAP-98
Q-K2
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,VARY
N(QB3)*P
SWITCH LW ON;SWITCH NOR ON
B-K2.  VARIATION 18, COTAP-98
P-Q4
P*P
SAVE,5
O-O
P-Q4
N(K4)-N5
Q-Q3
Q*P
B-B3
Q-QR4
N-K2
B-KB4
Q-Q1
```

```
P-B4
O-O
P*P
N*P
R(R1)-Q1.  MILIC-DUCKSTEIN,  ZAGREB 1955
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,5
O-O
SWITCH LW ON;SWITCH NOR ON
N-B3.  LOPEZ GAME 12, COTAP-100
N*N.CHECK
B*N
R-K1.CHECK
N-K2
N-N5
O-O
N*P
K*N
Q-R5.CHECK
K-N1
B-Q3
R-K1
P-KN4
P-Q3
P-N5
B-K4
B-R7.CHECK
K-B1
Q-B3.CHECK.  TRIFUNIVIC-KOSTIC.  ROGASKA SLATINA 1930
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,VARY
N(QB3)*P
SWITCH LW ON;SWITCH NOR ON
P-Q4.  VARIATION 19, COTAP-98
N*P
P*N
SAVE,V20
N*N
P*N
B*P.CHECK
B-Q2
Q-R5.CHECK
K-K2
Q-K5.CHECK
B-K3
B*R
Q*B
Q*P(B7).CHECK
B-Q2
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,V20
N*N
SWITCH LW ON;SWITCH NOR ON
Q-N4.  VARIATION 20.  COTAP-98
Q-K2
N-B3
P-KB4
Q*P(B5)
N-K5.CHECK
P-B3
P-Q4
P*P.EP
B*P(Q3)
Q-QN5.CHECK
B-Q2
Q-K2
O-O-O
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SETUP,V20
SWITCH NOR,ON
N*N
SWITCH LW ON;SWITCH NOR ON
Q-Q4.  LOPEZ VARIATION 21.  COTAP-98
SAVE,9
P-QB4
Q-Q3
N*P
K-Q1
N*B
K*N
P-Q4
P*P(Q6).EP
O-O
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,9
P-QB4
SWITCH LW ON;SWITCH NOR ON
Q-KN4. (Q)  IDEA VARIATION 12, COTAP-97
P-Q4
Q*P
Q-R5.CHECK
P-N3
Q-K5.CHECK
K-B2
N-Q8.CHECKMATE
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
YES
P-K4
P-K4
N-KB3
N-QB3
B-N5
P-Q3.  STEINITZ DEFENSE I.  COTAP-101
SAVE,A
SWITCH LW ON;SWITCH NOR ON.
P-Q4
B-Q2
SAVE,B
N-B3
P*P.
SAVE,C
N*P
N-B3
SAVE,D
O-O
B-K2
N(Q4)-K2.  VARIATION 23, COTAP-102
O-O
SAVE,5
N-N3
R-K1
SAVE,6
P-KR3
P-KR3
B-K3
B-KB1
Q-Q2
K-R2
R(R1)-K1
P-KN3
P-B4
B-N2
B-B4.  YATES-GIBSON,1931
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,6
P-KR3
SWITCH LW ON;SWITCH NOR ON
B-KB1.  BETTER.  NOTE 6, COTAP-102
B-K3
P-KN3
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,5
N-N3
SWITCH LW ON;SWITCH NOR ON
N-K4. (Q)
B*B
Q*B
P-N3
R(R1)-Q1
B-N2
N-B3
N-B5.  LASKER-WALBRODT, 1895
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,D
O-O
SWITCH LW ON;SWITCH NOR ON
N*N.  NOT SO GOOD.  NOTE G, COTAP-101
B*B.CHECK
Q*B
Q*N
B-K2
R-Q1
O-O
P-K5
N-K1
B-B4
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,C
N*P
SWITCH LW ON;SWITCH NOR ON
N*N.  ANOTHER POSSIBILITY.  NOTE E, COTAP-101
Q*N
B*B
N*B
N-K2
B-K3
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,C
N*P
SWITCH LW ON;SWITCH NOR ON
P-KN3.  LOPEZ GAME 14.  COTAP-105
B-K3
B-N2
Q-Q2
N-B3
P-B3
O-O
O-O-O
P-QR3
B-K2
P-QN4
P-KR4
N-K4
B-R6
N-B5
B*N
P*B
P-R5
P-B3
B*B
K*B
P*P
P(B2)*P
N(Q4)-K2
R-B2
Q*P
Q-N3
Q-Q4
P-B4
N-Q5
Q-N2
Q-B3
R(R1)-KB1
N*N
R*N
R-Q6.  EM LASKER-VIDMAR, ST. PETERSBURG 1909
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,A
P-Q4
SWITCH LW ON;SWITCH NOR ON
P*P.  LOPEZ GAME 13.  COTAP-105
Q*P
B-Q2
B*N
B*B
N-B3
N-B3
B-N5
B-K2
O-O-O
O-O
```

```
P-KR4
P-KR3
N-Q5
P*B
N*B.CHECK
Q*N
P*P
N*P
R-R5
Q-K3
R(Q1)-R1
P-B4
N-K5
P*N
P-N6.  ALEKHINE-VAN MINDENO, SIMULTANEOUS DISPLAY, AMSTERDAM 1933
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
YES
P-K4
P-K4
N-KB3
N-QB3
B-N5
P-Q3
P-Q4
B-Q2
SWITCH LW ON;SWITCH NOR ON
N-B3
N-B3
B*N
B*B
SAVE,VARY
Q-Q3
P*P.  VARIATION 25.  COTAP-103
SAVE,V26
N*P
P-KN3
B-N5
B-N2
SAVE,2
O-O-O
Q-Q2
P-KR3
O-O
R(R1)-K1
K-R1
Q-B3
N-N1
P-KN4
R(R1)-K1
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,2
O-O-O
SWITCH LW ON;SWITCH NOR ON
O-O.  NOTE 2, COTAP-103
N*B
P*N
P-K5
P*P
Q*Q
R(R1)*Q
R*R
R*R
N-K4
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,V26
N*P
SWITCH LW ON;SWITCH NOR ON
B-Q2.  VARIATION 26, COTAP-103
B-N5
B-K2
SAVE,4
O-O-O
O-O
SAVE,5
P-B4
N-K1
B*B
Q*B
N-Q5
Q-Q1
P-KN4.  SPIELMANN-MAROCZY, 1920
B*P
R(Q1)-N1
B-Q2
R-N2
P-QB3
N-K3
P-KN3
P-KR4
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,5
P-B4
SWITCH LW ON;SWITCH NOR ON
N-N5.  NOTE 5, COTAP-103
B*B
Q*B
R-Q2
R(B1)-K1
R-K2.
P-QB3
P-KR3
N-B3
P-KN4
R(R1)-Q1
Q-B3
B-B1
N-B5
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,4
O-O-O
SWITCH LW ON;SWITCH NOR ON
N-N5.  NOTE 4, COTAP-103
B*B
Q*B
Q-N3
N-B3
R(R1)-K1
O-O-O
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,VARY
Q-Q3
SWITCH LW ON;SWITCH NOR ON
N-Q2.  VARIATION 27, COTAP-103
SAVE,7
B-K3
P*P
B*P
N-B4
B*N
P*B
Q-K3
Q-B3
O-O-O
B-K2
N-Q5
B*N
R*B
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,7
B-K3
SWITCH LW ON;SWITCH NOR ON
P-QN3.  NOTE 7, COTAP-103
P-Q5
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
YES
P-K4
P-K4
N-KB3
N-QB3
SAVE,VARY
B-N5
Q-B3.  VARIATION 1, COTAP-106
SWITCH LW ON;SWITCH NOR ON.
N-B3
N(N1)-K2
P-Q3
N-Q5
N*N
P*N
N-K2
P-B3
B-R4
P-Q4
O-O
P-KN3
P-QN4
Q-Q3
P-QR3
B-N2
B-N2.  BOGOLJUBOV-ED. LASKER, 1924
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,VARY
B-N5
SWITCH LW ON;SWITCH NOR ON
P-B3.  VARIATION 2, COTAP-106
O-O
N(N1)-K2
P-Q4
N-N3
P-QR3
B-K2
B-QB4
P-Q3
P-R3
B-Q2
N-B3
Q-B1
K-R2
N-Q1
N-Q5.  TARRASCH-STEINITZ, 1896
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,VARY
B-N5
SWITCH LW ON;SWITCH NOR ON
N(N1)-K2.  VARIATION 3, COTAP-106
P-Q4
P*P
N*P
P-KN3
N-QB3
B-N2
B-K3
O-O
O-O
P-B4
P*P
N*P
N*N(B5)
R*N
B-QB4.CHECK
K-R1
Q-Q2
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,VARY
B-N5
SWITCH LW ON;SWITCH NOR ON
P-KN4.  VARIATION 4, COTAP-106
P-Q4
N*P
N*N
P*N
Q*P
Q-B3
P-K5
Q-KN3
N-B3
B-K2
N-Q5
B-Q1
O-O
P-QB3
B-Q3
```

```
Q-N2
N-K3
P-Q4
N-B5
B*N
B*B
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,VARY
B-N5
SWITCH LW ON;SWITCH NOR ON
P-KN3.  VARIATION 5, COTAP-106
O-O
B-N2
P-B3
P-Q3
P-Q4
B-Q2
SAVE,11SUB
P*P
P*P
B-N5
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,11SUB
P*P
SWITCH LW ON;SWITCH NOR ON
N*P.  NOTE 11, COTAP-107
N*N
P*N
Q-N3.  GELLER-TRIFUNIC, BELGRADE 1956.
W/PETRF
MSG       PETROFFS DEFENSE
INI
LET MSMASK=77B.
LET MSCODE=0.
SWITCH EXPERIENCE,OFF
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
YES
P-K4
P-K4
N-KB3
N-KB3
SWITCH LW ON;SWITCH NOR ON
P-Q4
P*P
P-K5
N-K5
Q*P
P-Q4
P*P
N*P(Q3)
SAVE,VARY
B-Q3.  VARIATION 14, COTAP-76
N-B3
SAVE,V15
Q-KB4
P-KN3
O-O
B-N2
SAVE,17
B-Q2.  DUCKSTEINS VARIATION
Q-B3
Q*Q
B*Q
N-B3
B-K3
N-KN5
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,17
B-Q2
SWITCH LW ON;SWITCH NOR ON
B*P.  NOTE 17, COTAP-77
SAVE,SUB1
B-B3
B*R
B*B
R-KN1
B-B6
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,SUB1
B-B3
SWITCH LW ON;SWITCH NOR ON
B*B.  PETROFF DEFENSE GAME 5, COTAP-80
N*B
B-K3
R(B1)-K1
O-O
N-KN5
N-K1
Q-KR4
N-B3
Q-R6
B-Q4
B-B4
N-QN5
B*B
N(N5)*B
R(R1)-Q1
P-B3
N(B3)-K4
R-K1
P-QB4.  NEUMANN-SPALA, EUROPEAN POSTAL CHAMPIONSHIP 1958
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,V15
Q-KB4
SWITCH LW ON;SWITCH NOR ON
Q-K2.CHECK (Q)  VARIATION 15,COTAP-76
B-K3
P-KN3
N-B3
B-K3
O-O
B-N2
R(B1)-K1
O-O
B-QB5
P-N3
B-R3
B*N
P*B
Q-Q2
R(R1)-Q1
N-R4.  PETROFF DEFENSE GAME 6, COTAP-80
Q-R6
P-KB3
B*P
P*B
Q*P.CHECK
K-R1
B*N
P*B
R*B
Q*R
R-Q4
Q-B5
R*Q
N*R
Q-R5.CHECK
K-N1
Q-N4.CHECK  MATAMPVOC-KIENINGER,HAMBERG 1955
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,VARY
B-Q3
SWITCH LW ON;SWITCH NOR ON
Q-K2.CHECK  VARIATION 16, COTAP-76
SAVE,21
B-K3
B-B4
N-B3
N-B3
Q-QR4
B*B
P*B
Q-Q2
P-Q4
B-K2
P-Q5
N-N1
Q-N3
O-O
O-O
B-B3
R(R1)-B1
P-QR4
R(KB1)-Q1.  BRONSTEIN-CHOLMOV,USSR 1958
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,21
B-K3
SWITCH LW ON;SWITCH NOR ON
N-B4.  NOTE 21, COTAP-77
B*N
B*B
N-B3
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,21
B-K3
SWITCH LW ON;SWITCH NOR ON
N-B3.  AND BACK TO VARIATION 15
Q-KB4.
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
YES
P-K4
P-K4
N-KB3
N-KB3
P-Q4
SWITCH LW ON;SWITCH NOR ON
N*P.  VARIATION 5, COTAP-76
B-Q3
P-Q4
SAVE,V6
N*P
B-Q3
SAVE,14
P-QB4
B-QN5.CHECK  NOTE 14A,COTAP-79
K-B1
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,14
P-QB4
SWITCH LW ON;SWITCH NOR ON
N-QB3. (Q) NOTE 14B, COTAP-79
P*P
N*P(Q5)
Q-R4.CHECK
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,14
P-QB4
SWITCH LW ON;SWITCH NOR ON
P-QB3
O-O
O-O
N-QB3.  BACK TO THE COLUMN
N*N
P*N
B*N
P*B
P*P
B*P(B4)
Q*Q
R*Q
B-B4
B-R3
R-K1
P-KB4.  ANALYSIS BY KERES (EXCEPT NOTE 14)
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,V6
N*P
SWITCH LW ON;SWITCH NOR ON
B-K2.  VARIATION 6, COTAP-76
```

```
O-O
O-O
P-QB4
N-KB3
SAVE,18
N-QB3
N(N1)-Q2
B-N5
P*P
B*P(B4)
N-N3
B-N3
N(B3)-Q4
B*B
N*B
Q-B3.  GLIGORIC-GUDMUNDSSON, AMSTERDAM 1950
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,18
N-QB3
SWITCH LW ON;SWITCH NOR ON
B-K3.  NOTE 18, COTAP-79
P-B5.  ALEKHINE-LEVITSKY, ST. PETERSBURG 1913
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,18
N-QB3
SWITCH LW ON;SWITCH NOR ON
P*P.  NOTE 18,COTAP-79
B*P(B4)
N-B3
B-K3
B-Q3
P-B4
N-K2
Q-N3
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
SETUP,18
N-QB3
SWITCH LW ON;SWITCH NOR ON
P-B3.  NOTE 18, COTAP-79
Q-N3
Q-N3
P-B5
Q*Q
P*Q
W/FRNCH
MSG      FRENCH DEFENSE
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
LET MSMASK=77B.
LET MSCODE=0.
SWITCH EXPERIENCE,OFF
SWITCH LW ON;SWITCH NOR ON
INI
YES.  ALBIN-CHATARD-ALEKINE ATTACK
P-K4
P-K3
P-Q4
P-Q4
SAVE,BASE
N-QB3
N-KB3
B-KN5
B-K2
P-K5
N(B3)-Q2
SAVE,VARY
P-KR4
P-KB3.  VARIATION 6, COTAP-292
SAVE,2
Q-R5,CHECK
K-B1
P*P
N*P
Q-B3
P-B4
SAVE,4
P*P
N(QN1)-Q2
O-O-O
N*P
N-R3
B-Q2
N-B4
INI
SETUP,4
P*P
P-QN3. (QE)  NOTE 4, COTAP-293
P-R5
P*P
P-R6. (E)
P-N3
O-O-O
N(QN1)-Q2
R-K1
Q-N3
B-N5. (E)
K-B2
N-R3.  UNZICKER-STAHLBERG, 1960
INI
SETUP,2
Q-R5.CHECK
P-N3.  NOTE 2, COTAP-293
SAVE,SUB1
P*P
N*P
Q-K2
O-O
O-O-O
P-B4
N-B3
N-B3
P*P
INI
SETUP,SUB1
P*P
P*Q
P*B
INI
SETUP,VARY
P-KR4
P-KR3.  VARIATION 7, COTAP-292
Q-R5.  NOTE 7, COTAP-293
P-QR3
O-O-O
P-QB4
P*P
INI
SETUP,VARY
P-KR4
P-QR3
SAVE,10
Q-N4
K-B1
P-B4
P-QB4
N-B3
N-QB3
O-O-O
P-N4
B*B.CHECK
N*B
P-R5
N-B4
Q-R3
INI
SETUP,10
Q-N4
P-KB4.  NOTE 10, COTAP-294
Q-R5.CHECK
P-N3
Q-R6
B*B
P*B
K-B2
N(KN1)-K2
P-B4
N-B4
N-B1.  FRENCH DEFENSE GAME 2, COTAP-295
P*P. (E)
N-B3
P-KN4. (E)
N*P
B-K2
Q-B2
P*P
N-B6.CHECK
B*N
Q*N
P-KB6. (E)
R-KN1
R-R3
B-Q2
N*P. (E)
Q-K4.CHECK
N-K3
Q*P(QN7)
R(QR1)-Q1
R-Q1
P-B6
P*P
K-B1
P-B4
N-B4
B-N4
Q*P(R7).CHECK
N*Q
R*N.CHECK
K-B1
R*R.CHECK.  UNZICKER-CZERNIAK, AMSTERDAM 1954
INI
SETUP,VARY
P-KR4
P-QB4.  VARIATION 10, COTAP-292
SAVE,15
B*B
K*B. (E)
SAVE,21
P-B4
P*P
Q*P
N-QB3
SAVE,22
Q-Q2
P-QR3
O-O-O
P-QN4
N-B3
N-N3
INI
SETUP,22
Q-Q2
Q-R4.  NOTE 22, COTAP-295
N-B3
R-Q1
R-R3
K-B1
B-Q3
N-N3
N-QN5
INI
SETUP,21
P-B4
Q-N3
SAVE,SUB1
N-B3
N-QB3. (Q)
N-QR4
Q-R4.CHECK
P-B3
P*P
P-QN4
Q-B2
N*P
P-QR3
R-R3.  GLIGORIC-YANOFSKY, SALTSJOBADEN 1946
INI
SETUP,SUB1
N-B3
Q*P
SAVE,SUB2
N-QN5
Q-N5.CHECK
K-B2. (E)
N-QB3
P-B4. (E)
INI
SETUP,SUB2
N-QN5
P-QR3
```

```
N-B7
Q-N5.CHECK
K-B2
R-R2
P-B4. (E)
INI
SETUP,15
B*B
Q*B. (Q)
N-N5
O-O
N-B7
P*P
SAVE,SUB1
N*R
P-B3
Q*P
N-B3
Q-Q2
P*P
O-O-O
N-B3
P-KB3
Q-Q3
N-K2
B-Q2
N-B3
R*N
N-K4. (E)
INI
SETUP,SUB1
N*R
N-QB3
N-B3
P-B3
N*P
P*P
N*P. (E)
Q*N
N-B7
INI
SETUP,BASE
N-QB3
N-KB3
B-KN5
B-N5.  MAC CUTCHEON VARIATION, COTAP-298
P-K5
P-KR3
B-Q2
B*N
P*B
N-K5
SAVE,B
Q-N4
P-KN3
SAVE,VARY
B-Q3.  VARIATION 17, COTAP-299
N*B
K*N
P-QB4
P-KR4
N-B3
N-B3
Q-B2
P*P
N*P(K4)
N*N
Q*N
R(QR1)-QN1
Q-B3.  NOTE 15, COTAP-300
Q-Q4
Q*Q
P*Q
K-Q1.  FRENCH DEFENSE GAME 5
P-N4
K-B2
P-R5
P-KN4
P-KB4
P*P
R(QN1)-KB1
P-K4
P*P
B*P
R*P
B-K3
K-B3
R(QR1)-KN1
K-Q4
R-N5
R*R
B*R
R-KB1
B*P
R-B6
K-Q2
B-B5.CHECK
K-K2
R-Q6
R-QN1
R*P(R6)
B-B6
R-Q6
B-K5
B*B
P*B
K*P
R-KR1
P-B6
P*P
R*P
R-R5.CHECK
K-Q5
R-R5
R-B7.CHECK
K-K1
P-B4
R*P(R7)
P-B5
R-Q7.CHECK
K-B6
R-K7
R*P(R7)
R*P
R-QN7
R-K3.CHECK
K-B7
P-B4
P-B6
K-K2
R-N5
K-B3
K-N7
K-N4
P-B7
R-K1
K-B6
R-QB1
R-N8
R*P.CHECK
K*R
K-B5
K-Q6.  NESHMETDINOV-STAHLBERG, BUCHAREST 1954
INI
SETUP,BASE
N-QB3
N-KB3
B-KN5
P*P
N*P
B-K2
SAVE,C
B*N
B*B
N-KB3
N-Q2
SAVE,1
Q-Q2.  VARIATION 1, COTAP-304
P-QN3
B-N5
B-N2
SAVE,2
N*B.CHECK
P*N
Q-B3
Q-K2
Q*P(B7)
Q-N5.CHECK
P-QB3
Q*B
Q*B
Q*P(N7)
O-O
R-Q1
P-B4
Q-R6
R(KB1)-K1
O-O
P-Q5. (E)  L STEIMER-STAHLBERG, SALTSTOBADEN 1948
INI
SETUP,2
N*B.CHECK
Q*N.  NOTE 2, COTAP-304
N-K5
INI
SETUP,1
Q-Q2
O-O.  NOTE 1, COTAP-304
O-O-O
Q-K2
P-KN4. (E)
INI
SETUP,BASE
N-QB3
B-N5.  NIMZOWITSCH VARIATION, COTAP-309
P-K5
P-QB4
SAVE,B
P-QR3
B*N.CHECK
SAVE,C
P*B
N-K2
Q-N4
P*P
Q*P(N7)
R-KN1
SAVE,F
Q*P(R7)
Q-B2
N-K2
N(QN1)-B3
P-KB4
B-Q2
Q-Q3
P*P
SAVE,VARY
R-QN1
N-B4.  VARIATION 31, COTAP-310
P-KR4
P-QR3
P-R5
O-O-O
R-R3
P-Q5
N-N3
N(QB3)-K2
N-K4. (E)  CHISTJAKOV-BENAVENETZ, MOSCOW 1937
INI
SETUP,VARY
R-QN1
P-Q5.  VARIATION 32, COTAP-310
N*P(Q4).  NOTE 4, COTAP-311
N*N
Q*N
N-B4
Q-B2.  (KERES)
INI
SETUP,VARY
R-QN1
O-O-O.  IDEA VARIATION 18, COTAP-310
N*P
N-R4
N-N5
B*N
R*B.  TSCHERBAKOV-KRESCHNOV, 1959
INI
SETUP,F
Q*P(R7)
Q-R4. (Q)  WEAK.  NOTE F, COTAP-310
R-N1
Q*P(B6).CHECK
B-Q2
Q-B2
```

```
P-KB4
N(QN1)-B3
N-B3
B-Q2
N-N5. (E)  ALEXANDER-BOTVINNIK, 1946
INI
SETUP,C
P*B
Q-B2.  NOTE C, COTAP-309
Q-N4
P-B4
SAVE,SUB1
Q-N3
P*P
P*P
N-K2
B-Q2
O-O
B-Q3
P-QN3
N-K2
B-R3
N-B4. (E)  RESHEVSKY-BOTVINNIK, 1948
INI
SETUP,SUB1
Q-N3
N-K2
Q*P
R-N1
Q*P
P*P
K-Q1. (E)  TAL-BOTVINNIK, 1960
INI
SETUP,B
P-QR3
B-R4
SAVE,SUB1
P-QN4
P*P(Q5)
Q-N4
N-K2.  FRENCH DEFENSE GAME 7, COTAP-313
P*B
P*N
Q*P(N7)
R-N1
Q*P(R7)
N-Q2
N-B3
N-B1
Q-Q3
Q*P
P-KR4. (E)
B-Q2
B-N5
R-B1
N-Q4
N-B4
R-QN1. (E)
R-B5
N*N
P*N
R*P
R-K5.CHECK
Q*R. (E)
P(Q4)*Q
R-N8.CHECK
B-B1
B-QN5.CHECK
Q*B
R*Q
N-K3
B-B6
R*P
P-R5
B-R3
P-R6. (E)  SMYSLOV-BOTVINNIK, 1954
INI
SETUP,SUB1
P-QN4
P*P(N5)
N-N5
P*P.CHECK
P-B3
INI
SETUP,B
P-QR3
P*P
P*B
P*N
N-B3. (E)
INI
SETUP,BASE
N-QB3
P*P
SAVE,A
N*P
N-Q2
SAVE,B
N-KB3
N(KN1)-B3
N*N.CHECK
N*N
SAVE,VARY
B-Q3
P-QN3.  VARIATION 42, COTAP-323
Q-K2
B-N2
B-KN5
B-K2
O-O
O-O
R(QR1)-Q1
B*N
Q*B
Q-Q4
Q*Q
N*Q
B-Q2.  PACHMAN-DOERNER, HILVERSUM 1947
INI
SETUP,VARY
B-Q3
B-K2.  VARIATION 44, COTAP-323
Q-K2
O-O
B-KN5
P-QB4
P*P
Q-R4.CHECK
P-B3
Q*P(B4)
O-O
R-Q1
SAVE,9
N-K5
R*B.  NOTE 9-1, COTAP-324
P-QN4
Q-Q4
P-QB4
Q-K5
Q*Q
N*Q
B*B
R-Q7
R(QR1)-Q1
P-B3.  FRENCH DEFENSE GAME 14, COTAP-325
R*R
N*R
R-Q1
P*N
R*N
K-B2
B-Q6
B-Q2
B*P
B-K1
R-Q4
P-KN3
P-KR4
R-B1
P-N4
P-KR4
P-B3
P-R3
K-B2
B-B3
K-K3
R-KN1
K-B4
K-K2
P-KN5
R-QB1
B-Q6.CHECK
K-B2
K-K5
K-N2
R-B4
R-K1
R-B6
P-N4
P-B5
B-Q4
P-R3
R-Q1
R*P(K6)
B*R
K*B
R-QR1
P-B6
R-K1.CHECK
K-Q7
K-B2
P-B7.  EUWE-LANDAU, 1939
INI
SETUP,9
N-K5
B-Q2.  NOTE 9-2, COTAP-324
B*N
B*B
B*P.CHECK
INI
SETUP,9
N-K5
P-KR3
B-R4
B-Q2
R(QR1)-Q1
INI
SETUP,B
N-KB3
B-K2.  NOTE B, COTAP-322
B-Q3
N(KN1)-B3
SAVE,SUB1
Q-K2
N*N
SAVE,SUB2
B*N
P-QB4
B-K3
O-O
O-O-O
N-B3
B-Q3
Q-B2
P*P.  KERES-PAVEY, 1954
INI
SETUP,SUB2
B*N
N-B3
B*P(N7)
INI
SETUP,SUB2
B*N
O-O
O-O
N-B3
B-Q3
INI
SETUP,SUB1
Q-K2
P-QB4
N*P. (E)
INI
SETUP,A
N*P
N-KB3
SAVE,SUB1
N*N.CHECK
Q*N
N-B3
B-Q2
B-KN5
Q-N3
B-Q3.  TARRASCH-LASKER, 1908
```

```
INI
SETUP,SUB1
N*N.CHECK
P*N
N-B3
P-N3
B-KB4
B-QN2
P-B3
B-Q3
B-N3
N-Q2
Q-R4
P-QR3
B-Q3.  ARONIN-UFIMZEW, 1947
W/SICIL
MSG      SICILIAN DEFENSE
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
LET MSMASK=77B.
LET MSCODE=0.
SWITCH EXPERIENCE,OFF
SWITCH LW ON;SWITCH NOR ON
INI
YES.  CLASSICAL DRAGON, COTAP-363
P-K4
P-QB4
N-KB3
N-QB3
LET MSCODE 40.  SELECT 50 PERCENT OF THE TIME
P-Q4
LET MSCODE 0
P*P
N*P
N-B3
N-QB3
P-KN3.  NOTE A, COTAP-363
SAVE,SUB1
N*N
P(Q2)*N
Q*Q.CHECK
K*Q
B-KB4
INI
SETUP,SUB1
N*N
P(N2)*N
P-K5
INI
YES.  RICHTER-ROUZER ATTACK, COTAP-370
P-K4
P-QB4
N-KB3
N-QB3
P-Q4
P*P
N*P
N-B3
N-QB3
P-Q3
SAVE,A
B-KN5
P-K3
SAVE,VARY
Q-Q2
B-K2.  VARIATION 7, COTAP-373
SAVE,1
O-O-O
O-O
SAVE,3
P-B4
N*N
SAVE,4
Q*N
P-KR3
SAVE,6
B-R4
Q-R4
SAVE,9
B-B4
P-K4
P*P
P*P
SAVE,10
Q-Q3
Q-B4
B*N
B*B
K-N1
B-K3
B*B
P*B
Q-Q6
Q-B5
Q-Q3
Q*Q
R*Q.  GLIGORIC-TAIMANOV, STOCKHOLM 1952
INI
SETUP,10
Q-Q3
R-Q1. (Q) NOTE 10. COTAP-374
N-Q5. (E)
INI
SETUP,10
Q-Q3
B-KN5.  ALSO NOTE 10. COTAP-374
B*N
B*B
R(Q1)-B1
R(QR1)-B1
R*B
P*R
N-Q5
INI
SETUP,9
B-B4
R-Q1.  NOTE 9, COTAP-374
P-K5
P*P
Q*P(K5)
Q-N5
B-QN3
B-Q2
R-Q4.  PACHMAN-ILLIVIZRY, 1956
Q-N3
INI
SETUP,6
B-R4
B-Q2. (EQ) NOTE 6, COTAP-374
B*N
B*B
P-K5
B-K2
P*P
B-KB3
INI
SETUP,4
Q*N
Q-R4.  NOTE 4, COTAP-374
B-B4
P-KR3
SAVE,SUB1
P-KR4
P-K4
Q-N1
P*P
B*P(B4)
B-N5
R-Q3
N-R4
B*P(Q6)
B*B
R*B
N-N6
Q-Q4
INI
SETUP,SUB1
P-KR4
P*B
P(R4)*P
N-N5
R-R4
N-R3
R(Q1)-R1
P-K4
Q-B2
P*P
P*N
B*R
Q*B
P-KN4
P-R7.CHECK
K-R1
Q-R6
Q-Q1
R-R5
P-B3
Q-N6
INI
SETUP,SUB1
P-KR4
R-Q1.  BEST
B*N
INI
SETUP,3
P-B4
P-KR3.  NOTE 3, COTAP-374
B-R4
N*P. (QE)
B*B
N*Q
B*Q
N*B(B8)
N*N
P*N. (E)  UNZICKER-STAHLBERG, 1956
B-K7
R-K1
R(R1)*N
R*B
R*P
B-N2
P-KN3. (E) UNZICKER-STAHLBERG, 1956
INI
SETUP,3
P-B4
P-K4
N-B5. (E)
B*N
P*B
R-B1
K-N1
INI
SETUP,3
P-B4
P-QR3.  SICILIAN DEFENSE GAME 9, COTAP-377
P-K5
P*P
N*N
P*N
P*P
N-Q2
P-KR4. (E)
R-N1
Q-K3
R-K1
R-R3
Q-R4
B*B
R*B
R-N3
R-K1
R*N. (E)
B*R
B-Q3
P-R3
Q-B4
K-B1
R*P. (E)
K*R
Q-B6.CHECK
K-B1
B-N6. (E)
INI
SETUP,1
O-O-O
N*N.  NOTE 1, COTAP-373
Q*N
O-O
P-K5. (E)
P*P
Q*P(K5)
Q-N3
B-K3
N-N5
B*Q
```

```
N*Q
B-B7
N-N5
B-N3.  VASJUKOV-BOLESLAVSKI, USSR 1957
INI
SETUP,VARY
Q-Q2
P-QR3.  VARIATION 8, COTAP-373
SAVE,V9
O-O-O
P-R3
B-KB4
B-Q2
N*N
B*N
SAVE,15
P-B3
P-Q4
Q-K1
B-K2
P*P
P*P
N-K2. (E)
O-O
N-Q4
B-Q3
Q-Q2.  KERES-CUELLAR, 1959
INI
SETUP,15
P-B3
Q-N3
B-B4
O-O-O
B-K3
Q-B2
Q-B2
N-Q2
P-B4
P-QN4
B-K2
Q-N2
P-QR3
N-B3
P-K5
N-Q4.  TAL-DJURASEVIC, WARNA, 1958
INI
SETUP,V9
O-O-O
B-Q2.  VARIATION 9, COTAP-373
SAVE,17
P-B4
B-K2
N-B3
P-N4
P-K5
P-N5. (E)
P*N
P*N
Q*P(B3)
P*P
B-R4
P-Q4
K-N1
N-N5
N-Q4
R-QB1
Q-K3
Q-N3
P-QR3
N-B3.  MATANOVIC-PEREZ, MUNICH 1958
INI
SETUP,17
P-B4
P-R3.  NOTE 17-1, COTAP-375
B-R4
N*P
Q-K1
N-B3
N-B5
Q-R4
N*P(Q6).CHECK
B*N
R*B
Q-B2
R-Q2. (E)
Q*P
SAVE,SUB1
B-K2. (E)
N-K5
N*N
Q*N
SAVE,SUB2
Q-B2
N-R4
R(KR1)-Q1
R-QB1
B-R5.  GLIGORIC-BARDAN, 1957
INI
SETUP,SUB2
Q-B2
P-K4
B-B3
Q-R5
R(KR1)-Q1
N-Q5
R*N. (E)
INI
SETUP,SUB1
B-K2
O-O-O
R-B1
Q-K4
Q-B2
INI
SETUP,17
P-B4
R-B1.  NOTE 17-2, COTAP-375
N-B3
Q-R4
K-N1
P-N4
P-K5. (E)
P-N5
P*N
P*N
SAVE,SUB1
P*P(N7)
B*P
Q*P(Q6)
R-B2
N-K5.  YANOFSKY-OFAFSSON, DALLAS 1957
INI
SETUP,SUB1
P*P(N7)
R-QN1
P-QN3. (E)
INI
SETUP,17
P-B4
P-KR3.  NOTE 17-2, COTAP-375
B-R4
R-B1
N-B3
Q-R4
K-N1
P-QN4
P-K5
P-N5
P*N
P*N
P*P(N7)
P*Q
P*R=Q
N-N5
P-QR3
N*P
N*P. (E)
N*P.CHECK
P*N
P-K4
N-B4. (E)
B-B4.CHECK
K-R2
B-K3
SAVE,SUB1
R*P. (E)
Q-B6
R*B.CHECK (E)
P*R
N-Q6.CHECK
K-Q2
Q-R7.CHECK
INI
SETUP,SUB1
R*P
R*N
R-Q8.CHECK
INI
SETUP,17
P-B4
P-QN4.  NOTE 17-4, COTAP-376
SAVE,SUB1
B*N
Q*B
B*P. (E)
P*B
N(Q4)*P(N5)
Q-Q1
N*P.CHECK
B*N
Q*B
N-K2
P-B5. (E)
INI
SETUP,VARY
Q-Q2
P-KR3.  VARIATION 10, COTAP-373
SAVE,20
B*N
P*B
O-O-O
P-R3
P-B4
B-Q2
SAVE,21
B-K2
P-KR4
K-N1
Q-N3
N-N3
O-O-O
R(KR1)-B1
N-R4
R-B3
N*N
P(R2)*N
K-N1
N-R4
Q-R2
P-B5
B-K2
P*P
P*P
R*P. (E)  KERES-BOTVINNIK, 1956
INI
SETUP,21
B-K2
Q-N3.  NOTE 21, COTAP-376
B-R5
Q*N
Q*Q
N*Q
R*N
R-KN1
P-KN3
B-K2
R(KR1)-B1.  BONDAREVSKY-BOTVINNIK, 1951
INI
SETUP,20
B*N
Q*B. (Q) NOTE20, COTAP-376
N(Q4)-N5
Q-Q1
O-O-O
INI
SETUP,A
B-KN5
Q-R4.  NOTE A-1, COTAP-370
B*N
P(N2)*B
B-N5
B-Q2
O-O
O-O-O
```

```
N-N3
Q-N3
P-QR4.  ALEKHINE-FRENTZ, PARIS 1933
INI
SETUP,A
B-KN5
B-Q2.  NOTE A-2, COTAP-370
B*N
P(N2)*B
N-B5
Q-B1
N-Q5. (E)
B*N
P*B
R-QN1
B-Q3.  BALOGH-VAN KOL, 1934
INI
YES.  ACCELERATED FIANCHETTO, COTAP-383
P-K4
P-QB4
N-KB3
N-QB3
P-Q4
P*P
N*P
P-KN3
N-QB3.  NOTE A, COTAP-383
B-N2
B-K3
N-B3
B-QB4
O-O
SAVE,SUB1
B-N3
N-QR4. (Q)
P-K5. (E)
N-K1
B*P.CHECK (E)
K*B
N-K6. (E)  FISCHER-RESHEVSKY, 1958
INI
SETUP,SUB1
B-N3
N-KN5
Q*N
N*N
Q-Q1
N*B
P(R2)*N
P-N3
Q-Q5
B*N.CHECK (E)
P*B
Q-B2. (E)
O-O-O
Q*P(B6). (E)
B-Q4
Q-B3
Q-K5
P-B3
Q*P(K7)
B-N2
P-KB3
P-QR4.  FISCHER-RESHEVSKY, 1961
INI
YES.  MODERN DRAGON VARIATION, COTAP-390
P-K4
P-QB4
N-KB3
P-Q3.  AVOIDING THE RICHTER-ROUZER
SAVE,B
LET MSCODE 40.  SELECT 50 PERCENT OF THE TIME
P-Q4
LET MSCODE 0
P*P
N*P
N-KB3
N-QB3
P-KN3
P-B3
B-N2
B-K3
O-O
SAVE,D
Q-Q2
N-B3
SAVE,VARY
O-O-O
N*N.  VARIATION 21, COTAP-391
SAVE,1
B*N
Q-R4
B-B4
B-K3
SAVE,2
B-N3
P-QN4
K-N1
P-N5
N-Q5
B*N
P*B
Q-N4
R(KR1)-K1
P-QR4
Q-K2
Q*Q
R*Q
P-R5
B-B4
R(KB1)-B1
P-QR3. (E)
INI
SETUP,2
B-N3
B*B.  NOTE 2, COTAP-392
P(B2)*B
R(KB1)-Q1
K-N1
R-Q2
P-KR4
R(QR1)-Q1
Q-KB2. (E)
P-QN4
P-R5
P-K4
B-K3
P-Q4
P-R6
P-N5. (Q)
P*B
P*N
B-N5
P*P(K5)
R*R
R*R
B*N
R-Q7
R*P. (Q)
INI
SETUP,1
B*N
B-K3.  NOTE 1, COTAP-391
SAVE,SUB1
K-N1. (E)
Q-R4
N-Q5. (E)
INI
SETUP,SUB1
K-N1
P-QR3
P-KR4
P-KR4
N-Q5. (E)
B*N
P*B
N-Q2
B*B
K*B
Q-Q4.CHECK
P-B3
P-KN4. (E)
Q-N3
Q*Q
N*Q
P*P
P*P
B-R3
INI
SETUP,VARY
O-O-O
B-K3.  VARIATION 23, COTAP-391
K-N1. (E)
R-B1
N*B
P*N
B-QB4
Q-Q2
B-N3
INI
SETUP,VARY
O-O-O
P-Q4. (E)  VARIATION 25, COTAP-391
P*P
N*P
N(Q4)*N
P*N
B-Q4
P-K4
B-B5
B-K3. (E)
N-K4
R-K1
B-R6
Q-B2
P-KN4
R(K1)-Q1
Q-K1
N-B3
R*R.CHECK
R*R
P-N5
N*N
P*N
B-KB1
B*B
K*B.  KERES-AVERBACH, 1959
INI
SETUP,D
Q-Q2
P-Q4.  NOTE D, COTAP-390
P-K5
N-K1
P-B4
P-B3
O-O-O. (E)
P*P
SAVE,SUB1
P*P
N-QB3
N-B3
P-K3
B-KR6. (E)
INI
SETUP,SUB1
P*P
B*P
N-B3
INI
SETUP,B
P-Q4
N-KB3.  NOTE B-1 COTAP-390
P*P
N*P
SAVE,SUB1
P*P
N*P(Q3)
B-KB4
P-K3
N-B3
N-B3
Q-Q2
P-B3
O-O-O
P-K4
N-QN5. (E)
INI
SETUP,SUB1
P*P
Q-N3.  NOTE B-2, COTAP-390
Q-Q4
Q*Q
N*Q
N*P(Q3)
```

```
N-QB3
B-Q2
N-Q5. (E)
K-Q1
B-KB4
P-K3
N-QB3
P-B3
N*P.CHECK (E)
B*N
O-O-O
INI
SETUP,SUB1
P*P
P-K3.  NOTE B-3, COTAP-390
B-Q3
N*P(Q3)
B-KB4
INI
YES.  NAJDORF SYSTEM WITH 6 B-K2
P-K4
P-QB4
N-KB3
P-Q3
P-Q4
P*P
N*P
N-KB3
N-QB3
P-QR3
B-K2
P-K4
SAVE,B
N-N3
B-K2
O-O
O-O
SAVE,VARY
B-K3
N(N1)-Q2.  VARIATION 31, COTAP-403
SAVE,1
P-B3
Q-B2
P-QR4
P-QN3
Q-Q2
B-N2
R(B1)-Q1
R(B1)-Q1
Q-K1
P-Q4. (EQ)
P*P
B-N5
SAVE,3
R-Q3
N-B4
N*N
P*N
Q-B2
P-B5
B-N6
Q-B1
R-K3
R-Q3
P-R5
N*P
N*N
R*N
R-R4. (E)  BISUIIER-GLIGORIC, ZAGREB 1955
INI
SETUP,3
R-Q3
B*N.  NOTE 3, COTAP-403
R*B
Q-N1
B-KN5
P-R3
SAVE,SUB1
B-R4
P-KN4. (Q)
B-N3
N*P
R-Q3
INI
SETUP,SUB1
B-R4
B*P
R-Q1
INI
SETUP,1
P-B3
P-QN4.  NOTE 1, COTAP-403
P-QR4
P-N5
N-Q5
N*N
SAVE,SUB1
Q*N
R-N1
R(QR1)-B1
B-N2
Q-R5.  BISGUIER-UDOVIC, ZAGREB 1955
INI
SETUP,SUB1
Q*N
N-N3
Q-R5. (E)
INI
SETUP,VARY
B-K3
Q-B2.  VARIATION 32, COTAP-403
SAVE,4
P-QR4
B-K3.  IDEA VARIATION 18, COTAP-402
P-R5
Q-B3
B-B3
N(N1)-Q2
N-Q5
B*N
P*B
Q-N4
Q-Q3
R(KB1)-B1
R(KB1)-B1
Q*Q
P*Q
P-KR3
R-B3.  SMYSLOV-TAL, 1959
INI
SETUP,4
P-QR4
P-QN3
Q-Q2
B-K3
N-Q5
INI
SETUP,VARY
B-K3
B-K3.  VARIATION 33, COTAP-403
P-B4
P*P
B*P(B4)
N-B3
K-R1. (E)
Q-N3
Q-Q2
R(QR1)-B1
B-K3
Q-B2
N-Q4
N*N
B*N
INI
SETUP,B
N-N3
B-K3.  NOTE B, COTAP-402
O-O
N(N1)-Q2
P-B4. (E)
Q-B2
P-B5
B-B5
P-QR4. (E)
R-B1
B-K3
B-K2
SAVE,SUB1
P-R5
P-R4
B*B
Q*B
R-R4
Q-B2
P-R3
INI
SETUP,SUB1
P-R5
O-O
P-N4
MSG SICILIAN B-N5 GLIGORIC
INI
MSG. B-N5 SICILIAN, GLIGORIC TEXT
P-K4. COL 48.
P-QB4
N-KB3
N-QB3
B-N5.   DUPLICATE POSITION OVERRIDES 50 PERCENT OF TIME.
B-N5
SAVE,C48M3.
P-QR3
BXN
P/QXB
P-KR3
P-K4
P-Q3
B-Q3
P-QR4
P-QR4
N/1-Q2
N-K2
N-B4
P-B3
N-R4
O-O
P-KN4
B-B2
B-K3
P-QN3
Q-K2
B-K3
P-QN3
Q-Q2
R-KN1
K-R1
O-O-O. ECO B30-2-156
SETUP,C48M3. COL 49.
Q-N3
N-B3
P-K3
O-O
SAVE,C49M5.
N/1-K2
R-K1
N-Q5
B-B4
N/2-B3
P-Q3
P-Q3
NXN
PXN
N-K2
N-R4
P-QB3
NXB
PXN
PXP
NXP
B-K2
P-QN3
O-O
B-K3
Q-B3
Q-Q3
P-QN3
R/QR1-Q1
R-Q1
B-Q4. ECO B30-3-156 USSR  1965
SETUP,C49M5. COL 50.
N-Q5
B-B4
N-K2
R-K1
N-N3
```

```
P-Q3
B-K2
NXN
PXN
N-K2
0-0
P-QB3
PXP
PXP
Q-B2
B-N3
P-N3
B-N2
B-N2
N-N3. ECO B30-3-NOTE 13 USSR 1965
SETUP,C48M3. COL 51.
Q-B2
O-O
N-B3
N-B3
P-K3
R-K1
P-Q3
P-Q4
PXP
N-Q5
SAVE,C51M8.
Q-Q1
NXP
B-Q2
B-N5
B-K2
NXB
NXN/Q5
BXB
QXB
BXN
PXB
N-Q5. GLIGORIC P146 RICHER-GORMAN 1950
SETUP,C51M8. COL 52.
PXN
PXP
B-K2
NXP
NXP
NXN
PXN
QXN. ECO B30-1-156 MOSCOW-BUDAPEST 1949
SETUP,C48M3. COL 53.
P-K3
O-O
N-B3
R-K1
P-Q4
PXP
NXP
N-K5
Q-B2
Q-B3
B-Q3
NXN
SAVE,C53M9.
P/NXN. DUBIOUS MOVE
QXN. GLICGORIC P147 NOTE 3A 1948
SETUP,C53M9. COL 54.
O-O. WITH AN EXCLEMATION POINT.
NXP
RXN. AND EQUALITY
INI
SW EX OFF
SW LB OFF
SW LW ON
SW NO ON
P-K4
P-QB4
N-KB3
P-Q3
SAVE,1S3
B-N5
B-Q2
B*B
SAVE,1S4
N*B.      SCHWARZ ECO B52/2 P. 235
O-O
N/1-B3
Q-K2
P-KN3
P-QB3
B-N2
P-Q4
P-K3
R-Q1
Q-B2
N-R3
SETUP,1S4
Q*B.      ECO B52/8/236
P-QB4
SAVE,2S5
N-QB3
O-O
P-KN3
P-Q4
P*P
N*P
B-N2
B-K3
N-R3
P-B3
P-B4
N-B3
P*P
Q-Q2
SAVE,2S12
N-B4
N*N/5
P*N
P*P
SETUP,2S12
N-B2.     NEI-SPASSKY TALLINN 1973 ECO B52/8/236
N*P
O-O
R/QR1-Q1
SETUP,2S5
P-K4.     SPASSKY-BYRNE 1974 ECO B52/9/236
N-QB3
N-QB3
P-Q3
P-KN3
P-QR3
B-KN2
R-QN1
N/1-K2
P-QN4
P-QN3
O-O
O-O
N-Q5
N*N
P/B*N
N-Q5
N*N
P/B*N
B-Q2
RQR1-B1
Q-N3
SETUP,1S3
B-N5
N-QB3
O-O
P-QR3
B*N
P*B
P-Q4
P*P
Q*P
P-K4
SAVE,5S8
Q-Q3
N-B3.     SZABO-KOTOV BUDAPEST 1950.  GLIGORIC P. 154
B-N5
B-K2
R-Q1
O-O
B*N
PXB
N-R4
K-R1
N-Q2
B-N5
R-K1
P-Q4
N-B1
R-KN1
N-K3
B-K3
P-QB3
Q-R4
SETUP,5S8
Q-Q3
B-K2
SAVE,6S9
R-Q1
B-K3.     MILEV-TROIANESCU BUCHAREST 1953
P-B4.     GLIGORIC P.154.
Q-B2
N-B3
P-R3
P-QN3
N-B3
B-R3
P-B4
N-Q2
O-O
N-B1
R/KB1-N1
N-K3
B-KB1
P-B3
Q-Q1
SETUP,6S9
R-Q1
Q-B2.     ARONIN-KOTOV 17TH U.S.S.R. 1949
N-B3.     GLIGORIC P.155.
P-QR4
B-N5
P-B3
B-K3
P-N3
Q-B4
B-R3
Q-K6
Q-B1
Q-N3
Q-N2
N-Q2
Q-N5
Q-K6
K-B1
SETUP,6S9
R-Q1
P-R3.     TRIFUNOVIC-SANGUINETTI MAR DE PLATA 1950
N-R3
B-K3
N-B4
B*N
Q*B
MSG SICILIAN B-N5 GLIGORIC
SWITCH LW OFF;SWITCH LB OFF; SWITCH NOR OFF
LET MSMASK=77B.
LET MSCODE=0.
SWITCH EXPERIENCE,OFF
SWITCH LW ON;SWITCH NOR ON
INI
MSG. B-N5 SICILIAN, GLIGORIC TEXT
P-K4
P-QB4
N-KB3
N-QB3
B-N5
SAVE,C20M3
P-KN3
O-O
B-N2
P-B3
SAVE,C20M5.
N-B3
R-K1
0-0
P-Q4
PXP
PXP
SAVE,C20M8
P-Q4
P-K5
```

```
SAVE,C20M9
N-K1
P-KR3
SAVE,C20M10.
Q-N3
N-QB3
N-B2
B-B1
P-QR3
P-QN3
R-Q1
N-QR4
Q-R2
B-R3
N-K3
R-QB1. HALLE 1967 GLIGORIC WITH ADVANT TO W
SETUP,C20M8
P-Q3
P-KR3
P-QR3
B-B1
P-K4
N-B3
N-KR4
B-K3
N-B5
Q-Q2
Q-B3
P-Q5
N-K2
N-R2. EQUAL USSR 65 GL
SETUP,C20M8. COL 22.
P-QR3
B-Q3
P-Q4
P-K5
N-K5
N-QB3
B-N5
NXN
BXN
QXB
N*P/Q
Q-N4
PXN
QXP/4. WITH ADVANT TO WHITE NICE 67 GL P 162
SETUP,C20M9. COL 23.
N-K5
N-B3
NXN
PXN
SAVE,C23M11
B-N5
P-KR3
BXN
QXB
R-B1. WITH EQUAL GL P167 1963
SETUP,C23M11. COL 24.
N-QR4
Q-R4
P-QR3
B-B1
B-N5
N-Q2
R-B1
N-N3
NXN
QXN
Q-B2
B-Q2
R/KB1-Q1
R/QR1-N1. WITH ADV TO W 74 ECO B31 15 159
SETUP,C20M10
N-B2
BXN
PXB
P-QN3. WITH EQU GL P168
SETUP,C20M5. COL 36.
Q-N3
N-R3
SAVE,C36M6
N-B3
P-K5
N-Q4
B-B4
N-B2
R-K1
O-O
B-N3
N-R4
P-Q4
PXP
NXP
NXB
PXN
P-Q4
B-K3. WITH AHHA GL 34 P163
SETUP,C36M6. COL 48.
P-QR3
B-R4
Q-B2
P-Q4
P-QN4
B-N3
P-Q3
R-K1
P-B5
B-B2
P-K4
P-KR3. WITH ADV TO W ECO B31 10 42 P159
SETUP,C20M5. COL 37.
P-Q4
Q-R4
SAVE,C37M6
PXP
BXN. CHECK
PXB
QXP/B.CHECK
B-Q2
QXP/K4.
N-B3
Q-KR4
O-O
P-Q4
PXP
Q*P/4. WITH AD TO W GL P164 ECO B31 11 P159
SETUP,C37M6
P-K3. WEAK MOVE
P-Q4. S T R O N G MOVE
PXP/Q5. MOVE IS PXQP
NXP
N-K2
PXP
QXP
R-Q1. W ADV TO W ECO B31 11 44 P161 1971
SETUP,C20M5. COL 28.
P-K4
P-Q4
SAVE,C28M6
P/B*P
PXP
SAVE,C28M7
PXP
B-KB4
SAVE,C28M8
P-QR3
B-R4
SAVE,C28M9
P-QN4
B-QN3
P-Q3
P-QR4. A HURRAH MOVE
SAVE,C28M11
P-N5
N/1-Q2
B-K3
R-B1
SAVE,C28M13
N/1-K2
BXB
PXB
N-N5
Q-Q2
Q-N3. JANSA ECO B31 13 159
SETUP,C28M6. COL 29.
P/K*P
PXP
NXP
NXN
BXN
N-B3. WITH ADV TO W SHOULD BE ANALYIZED
SETUP,C28M7. COL 30.
NXP
NXN
PXN
P-K5
P-QR3
B-R4
P-QN4
B-N3
B-N2
QXP
N-K2
B-N5. WITH WHOOPEE ECO B31 13 55 P161
SETUP,C28M8. COL 31.
N/1-K2
B-Q3
0-0
R-K1. ADV TO W ECO B31 13 56 P161
SETUP,C28M9. COL 32.
N/1-K2
B-Q6
0-0
N/1-Q2. ADV TO W ECO 31 13 58 P161
SETUP,C28M11. COL 33.
B-N2
PXP
PXP
RXR
BXR
N-R3. W ADV TO W ECO B31 13 60 P162
SETUP,C28M11. COL 34.
R-N1
PXP
PXP
N-R3
N/1-K2
N-B2. ECO B31 13 60 ADV TO WHITE
SETUP,C28M13. COL 35.
B*B.   THIS WAS A NONO
Q*B.
N/1-K2
N-B4
0-0
B*P.  W ADV ECO B31 13 159 1974
SETUP,C20M5. COL 26.
P-QR3
BXN
P/Q*B
P-Q4
N-B3
R-K1
SAVE,C26M8
PXP
PXP
B-N5
N-B3
O-O
B-K3
P-K4. WITH SMILES
PXP
N-Q2
P-KR3
BXN
QXB
NXP
Q-K2
Q-Q3. EQUALITY ECO B31 16 75 P162
SETUP,C26M8. COL 27.
P-QR4
P-K5
N-Q4
P-QR4
PXP
NXP
O-O
N-R3
Q-B2
Q-K2
P-N3
P-KR4. W ADV TO W  USSR CHAMP ECO B31 16 75 P162
SETUP,C20M3. COL 39.
N-B3
P-K5
```

```
SAVE,C39M4
N-Q4
O-O
SAVE,C39M5
P-KN3
Q-K2
N-B2
BXN
P/Q*B
P-KR3
B-N2
P-Q3
O-O
R-K1
P-KR3
N/1-Q2
B-K3
N-K4
P-N3
P-QN3
N-N4
P-B4
N-Q5
NXN
PXN
B-B4. W ADV TO W PETROSIAN OBERHAUSEN 1961
SETUP,C39M5.  COL 40.
P-K3
BXN
P/N*B
P-B4
N-N5
P-Q4
PXP
P-QR3
N-R3
QXP
P-QB4
Q-B4
B-K2
Q-N4
K-B1
N-B3
B-N2
P-KR4. SPARTAKIAD USSR 1959
SETUP,C39M5.  COL 41.
N-B2
P-QR4
P-KN3
R-K1
B-N2
BXN
P/Q*B
P-Q3
B-N5
N/N-Q2
Q-Q4
R-K4
P-KR4
P-R3
BXN
NXB
O-O-O
Q-K1
N-K3
P-QN4. BANIK CHEREPKOV 1964
SETUP,C39M4. COL 42.
N/KB3-N5
BXN
P/Q*B
0-0
P-KN3
R-K1
B-N2
P-KR3
N-R3
N-B3
P-N3. DUBIOUS MOVE
P-Q4
PXP
NXP
P-QB4
N-B6
Q-Q2
NXP/K
KXN
BXN
BXB
Q-B3
B-KN2
N-Q5
K-Q1
R/QR1-Q1. WITH A WIN ALMOST CERTAIN XXVI USSR 1959
SETUP,C20M3. COL 43.
P-Q3
O-O
SAVE,C43M4
N-B3
R-K1
P-K4
P-B3
B-Q2
P-Q4
Q-B2
B-N5
B-K2
PXP/B
PXP
N/QN1-Q2
O-O
N-B4
N-N1
BXB
N/NXB.
BXN
BXB
N-K3
R/R1-Q1. MOVE IS QR-Q1
Q-K2
N-N3
P-QR4
P-N3
P-R5
N-B1
N-Q5. W ADV TO W GLIGORIC P150
SETUP,C43M4. COL 44.
B-Q2
P-B3
N-B3
R-K1
P-QR3
B-R4
SAVE,C44M7
P-K4
P-Q4
P-QN4
PXP/K
PXP
B-B2
B-K2
N/QN1-Q2
O-O
N-B1
B-K3
B-Q2
R-K1
N-K3
P-N3
P-QN4
R-N1
Q-K2
Q-B2
N-N5
B-Q2
N-Q5. SPANISH VAR GLIGORIC P152
SETUP,C44M7. COL 45.
P-QN4
B-B2
P-K4
P-Q4
P/B*P
PXP
B-KN5
B-K3
B-K2
P-Q5
N-N5
B-N3
P-QR4
N-B3
P-R5
P-QR3
PXB
PXN
O-O
RXR
QXR
P-KR3
B-Q2
N-Q2. W ADV TO W VII YUG 1951 GLIG P152
SETUP,C43M4. COL 46.
B-N5
P-B3
N-B3
R-K1
SAVE,C46M6
P-QR3
BXN
PXB
P-Q3
P-K3
N/QN1-Q2. MOVE IS QN-Q2
N-Q2
Q-R4
Q-N3
P-KR3
B-R4
N-R2
B-K2
N-B4
Q-N4
Q-B2
P-Q4
N-K3
P-B5
P/Q*P
PXP/B
P-QN3. EQUAL USSR 1950 GLIG P153
SETUP,C46M6. COL 47.
Q-N3
N-R3
P-K3
B-R4
B-K2
N-B4
Q-B2
P-Q3
O-O
B-B4
R/QR1-Q1
P-KR3. WITH ADV TO W ECO B51 5 18 P235
INI
YES.  SCHEVENINGEN VARIATION
P-K4
P-QB4
N-KB3
P-K3
P-Q4
P*P
N*P
N-KB3
N-QB3
P-Q3
SAVE,D
B-K2
B-K2
O-O
N-B3
P-QN3.  VARIATION 45, COTAP-417
O-O
SAVE,12
B-N2
Q-R4
Q-Q2
R-Q1
R(QR1)-Q1
N*N
Q*N
N-K1
P-QN4
Q-B2
N-N5
INI
SETUP,12
```

```
B-N2
P-QR3.  NOTE 12, COTAP-418
K-R1
B-Q2
P-B4
Q-R4
B-B3
R(KB1)-Q1
P-KN4.  TAIMANOV-PANOV, BAKU 1944
INI
SETUP,D
B-K2
P-QR3.  NOTE D, COTAP-416
O-O
N-B3
B-K3
Q-B2
P-B4
N-QR4. (Q)
SAVE,SUB1
P-B5
N-B5. (Q)
B*N
Q*B
P*P
P*P. (Q)
R*N
P*R
Q-R5.CHECK
K-Q1
Q-B7.  LASKER-PIRC, MOSCOW 1935
INI
SETUP,SUB1
P-B5
B-K2
Q-Q3
N-B3
INI
YES.  MODERN VARIATION WITH  5 P-QB4
P-K4
P-QB4
N-KB3
P-K3
SAVE,A416
P-Q4
P*P
N*P
P-QR3
P-QB4
N-KB3
B-Q3.  NOTE A, COTAP-424
N-B3
N*N
P(Q2)*N
O-O
P-K4
Q-B2
B-QB4
N-Q2
B-K3.  BENKO-SMYSLOV, 1959
INI
SETUP,A416
P-Q4
P-Q4.  NOTE A, COTAP-416
P*P(Q5)
P(K3)*P
B-QN5.CHECK
N-B3
O-O
N-B3
N-K5
B-Q2
B*N
P*B
R-K1
B-K2
P*P
O-O
B-N5.  KERES-KONSTANTINOPOLSKY, 1950
INI
YES.  NIMZOWITCH VARIATION, COTAP-436
P-K4
P-QB4
N-KB3
N-KB3
SAVE,VARY
N-B3
P-Q4.  VARIATION 64, COTAP-437
P*P
N*P
SAVE,16
B-N5.CHECK
N-B3
N-K5
N*N
P(Q2)*N
Q*Q.CHECK
K*Q
B-Q2
B*N
B*B
N*B
P*N
P-QB4
INI
SETUP,16
B-N5.CHECK
B-Q2.  NOTE 16, COTAP-438
SAVE,SUB1
N-K5. (E)
B*B
SAVE,SUB2
Q-B3
P-B3
SAVE,SUB3
N*B
N-R3
Q-R5.CHECK
P-N3
N*P(N6)
P*N
Q*R
Q-Q2
N-B3
Q-K3.CHECK
K-B1
N(R3)-N5
Q-R3
Q*Q
P*Q
N-B2. (E)  SOZIN-KYRILLOF, MOSCOW 1931
INI
SETUP,SUB3
N*B
P*N
Q*N. (E)
INI
SETUP,SUB2
Q-B3
N-KB3
Q*P
B-R3
Q*R
Q-B2
N-B6. (E)
INI
SETUP,SUB1
N-K5
N*N. (Q)
Q-B3. (E)
INI
SETUP,SUB1
N-K5
P-K3
Q-B3
Q-K2
INI
SETUP,VARY
N-B3
N-B3.  VARIATION 65, COTAP-437
SAVE,19
P-Q4
P-Q4
P*P(Q5)
N(KB3)*P
N*N
Q*N
B-K3
P*P
SAVE,21
N*P
Q-QR4.CHECK
P-B3
B-Q2
B-K2
N*N
B*N
P-K4
B-K3
B-R5
Q-B1
B-B3
O-O
B-K2
INI
SETUP,21
N*P
P-K4.  NOTE 21, COTAP-439
N-N5
INI
SETUP,21
N*P
N*N
Q*N
Q*Q
B*Q
INI
SETUP,19
P-Q4
P*P
N*P
W/PIRC
MSG PIRC DEFENSE
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
LET MSMASK=77B.
LET MSCODE=0.
SWITCH EXPERIENCE,OFF
SWITCH LW ON;SWITCH NOR ON
INI
YES.  PIRC DEFENSE, COTAP-356
P-K4
P-Q3
P-Q4
N-KB3
N-QB3
P-KN3
P-B4
B-N2
SAVE,A
N-B3
O-O
B-K2.  VARIATION 3, COTAP-358
P-B4
SAVE,V4
P*P
Q-R4
O-O
Q*P(B4).CHECK
K-R1
N-B3
SAVE,7
N-Q2. (E)
B-K3
N-N3
Q-N3
P-KN4
R(QR1)-B1.  NOTE 8, COTAP-359
P-B5
B*N
P(R2)*B
N-QN5
B-QB4
Q-B3
Q-B3
N*P(B7)
P-N5
N*R
P*N
P(K2)*P
P*P
P(R2)*P
R-KN1
P-KN4
B*P(N5). (E)  ESTRIN-ZUGOVITSZKY, 1958
```

```
INI
SETUP,7
N-Q2. (E)
N-Q5.  NOTE 7-1, COTAP-359
N-N3
N*N
P(R2)*N
P-QN4. (Q)
P-K5. (E)
P*P
P*P
Q*P
B-KB4.  AND WHITE WINS THE EXCHANGE
INI
SETUP,7
N-Q2. (E)
P-QR4.  NOTE 7-2, COTAP-359
N-N3
Q-N3
P-QR4
N-QN5
P-B5
INI
SETUP,V4
P*P
P*P.  VARIATION 4, COTAP-358
Q*Q
R*Q
SAVE,9
P-K5
N-K1
B-K3
P-N3
R-Q1. (E)
R*R.CHECK
K*R
N-QB3
K-B1.  BOLESLAVSKI-PIRC,HELSINKI 1952
INI
SETUP,9
P-K5
N-Q4.  NOTE 9, COTAP-359
N*N
R*N
B-B4
R-Q1
B-K3
P-N3
N-N5.  DE MOURA-MATANOVIC, BAD PYRMONT 1951
INI
SETUP,A
N-B3
P-B4.  NOTE A,COTAP-356
B-N5.CHECK
B-Q2
P-K5
N-N5
P-K6. (E)  KROGIOUS-POLUGAYEVSKY, USSR CHAMPIONSHIP 1958
W/NIMZO
MSG NIMZOWITSCH DEFENSE
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
LET MSMASK=77B.
LET MSCODE=0.
SWITCH EXPERIENCE,OFF
SWITCH LW ON;SWITCH NOR ON
INI
YES.  NIMZOWITSCH DEFENSE, COTAP-352
P-K4
N-QB3
SAVE,VARY
P-Q4
P-Q4.  VARIATION 2, COTAP-353
SAVE,4
N-QB3
P*P
SAVE,5
P-Q5
N-N1
B-QB4
N-KB3
B-B4
P-B3
SAVE,7
N(KN1)-K2
P*P
N*P(Q5)
N*N
B*N(Q5)
P-K3
B*P(K4)
Q*Q.CHECK
R*Q.  ANALYSIS BY PANOV
INI
SETUP,7
N(KN1)-K2
P-QN4. (Q)  NOTE 7, COTAP-354
P*P. (E)
INI
SETUP,5
P-Q5
N-K4.  LESS COMMENDABLE.  NOTE 5, COTAP-354
B-KB4
N-N3
B-N3
P-QR3
B-QB4
N-B3
Q-Q4
INI
SETUP,4
N-QB3
P-K3.  NOTE 4, COTAP-354
N-B3
INI
SETUP,VARY
P-Q4
P-K3.  VARIATION 3, COTAP-353
N-KB3.  IDEA VARIATION 3, COTAP-352
P-Q4
P-K5
P-QN3
P-B3
N(QB3)-K2
B-Q3
P-QR4
Q-K2. (E)  SPIELMANN-NIMZOWITSCH, NEW YORK 1927
INI
SETUP,VARY
P-Q4
P-K4
P*P
N*P
N-QB3
B-B4
P-B4
N-N3
N-B3
P-Q3
B-B4
B-K3
Q-K2
B*B
Q*B
Q-Q2
P-B5
N(N3)-K2.  NIMZOWITSCH DEFENSE GAME 2, COTAP-355
B-N5
P-KB3
B-B4
N-B3
O-O-O
O-O-O
P-KN4
P-KN4
B-N3
P-KR4
P-KR3
Q-R2
R-R2
B-K6.CHECK
K-N1
P-R5
B-B2
B*B
R*B
R-Q2
N-Q4
N*N
Q*N(Q4)
P-N3
N-Q5
Q-B2
Q-R4
K-N2
R-B3
N-R3
N-N4
P-R4
Q-B6.CHECK
K-N1
N-R6.CHECK
K-R2
N-B5.  KERES-MIKENAS, TIFLIS 1946
INI
SETUP,VARY
P-Q4
P-Q3.  VARIATION 5, COTAP-353
N-KB3
B-N5
P-Q5.  NOTE 16, COTAP-354.  SETS AN ASTONISHING TRAP
N-K4. (Q)
N*N.(E)
B*Q
B-QN5.CHECK
P-B3
P*P
Q-R4.CHECK
N-B3
O-O-O
N-B4.  WITH COMPLICATIONS ALL IN WHITES FAVOR
W/CAROK
MSG CARO-KANN DEFENSE--MODERN VARIATION
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
LET MSMASK=77B.
LET MSCODE=0.
SWITCH EXPERIENCE,OFF
SWITCH LW ON;SWITCH NOR ON
INI
YES.  MODERN VARIATION, COTAP-272
P-K4
P-QB3
SAVE,B
P-QB4
P-Q4
P(K4)*P
P*P
SAVE,VARY
P-Q4.  VARIATION 17, COTAP-274
N-KB3
SAVE,V18
N-QB3
P-K3
SAVE,2
N-B3
B-K2
SAVE,8
B-N5
N-B3
R-B1
O-O
B-Q3
P-QN3
O-O
N-QN5.  KERES-TAIMANOV, ZURICH 1953
INI
SETUP,8
B-N5
O-O.  EQUALLY PLAYABLE.  NOTE 8, COTAP-275
R-B1.  CARO-KAHN DEFENSE GAME 9, COTAP-276
N-B3
P-B5
N-K5
B*B
Q*B
B-K2
B-Q2
P-QR3
P-B4. (Q)
B-N5
N-N4
B*N
N*N.CHECK
Q*N
P*B
```

```
Q-B4. (E)
R(QR1)-K1
O-O
P-K4
Q*P(K5)
Q*Q
P*Q
R*P
P-B4
R-K2
R(KB1)-K1
R(KB1)-K1
R*R
R*R
K-B2
K-B2
R-Q1. (E)
R-K1
R-Q2
P-KR3
R-K2
R-QN1
K-K3
R-N6
K-Q4
K-B3
N-R2
R-N1
P-QN4
P-N4
P-N3
P*P
P*P
P-R3
N-B3
R-N1
P-QR4
R-N5
R-KB2
B-K3
P-N5. (E)
P(R3)*P
P*P
P*P
N*P(N5)
R-N8
N-B3. (E)
K-B2
R-QN2
R-KB8
N-K2. (E)
R-K8
K-K5. (E)
P-Q5
K*P
K-N3
N-B3
K-R4
R-K2
R*R
N*R
K-N5
K-K5
B-B1
N-Q4
P-R4
N*P. (E)
B-Q2
N-N7
B-R5
P-B5
K-N4
N-K6.CHECK
INI
SETUP,2
N-B3
N-B3.  NOTE 2, COTAP-274
P-B5. (E)
N-K5
B-QN5
N*N
P*N
INI
SETUP,V18
N-QB3
N-B3.  VARIATION 18, COTAP-274
SAVE,V19
B-N5
P*P
SAVE,11
P-Q5. (E)
N-QR4
P-QN4. (E)
P*P.EP
SAVE,12
P*P
P-QN3
P-QN4
N-N2
B-N5.CHECK
B-Q2
N-B3
INI
SETUP,12
P*P
P-K3.  NOTE 12, COTAP-275
B-N5.CHECK IS STRONG
INI
SETUP,12
P*P
P-K4.  SAVE NOTE 12, COTAP-275
B-N5.CHECK STILL IS STRONG
INI
SETUP,11
P-Q5. (E)
N-K4.  NOTE 11, COTAP-275
Q-Q4
N-Q6.CHECK
B*N
P*B
N-B3
INI
SETUP,V19
B-N5
Q-N3.  VARIATION 19, COTAP-274
SAVE,15
P*P
N*P(Q5)
N(KN1)-K2
N-B4
Q-Q2
N-Q3
N-N3
INI
SETUP,15
P*P
Q*P(N7). (Q) A WELL-KNOWN MISTAKE.  NOTE 15, COTAP-275
R-B1
N-QN5
N-R4
Q*P(R7)
B-QB4
B-N5
N-KB3.  AND WINS.  BOTVINNIK-SPIELMANN, 1935
INI
SETUP,V19
B-N5
Q-R4.  NOTE 14, COTAP-275
B*N.  IDEA VARIATION 10, COTAP-273
P(K2)*B
P*P
B-QN5
Q-Q2
B*N
P*B
Q*P(Q4)
N-B3
B-R6
P-B4
INI
SETUP,V19
B-N5
P-K3.  NOTE 14, COTAP-275
SAVE,SUB1
P-B5.  IDEA VARIATION 11, COTAP-273
B-K2
B-N5
O-O
N-B3
N-K5. (E)
SAVE,SUB2
B*B
Q*B
Q-B2
N-N4. (E)
N*N
Q*N
SAVE,SUB3
B*N
P*B
O-O
P-K4. (E)  KERES-ALEKHINE, 1938
INI
SETUP,SUB3
B*N
Q*P
B*P(Q5)
P*B
O-O-O
INI
SETUP,SUB2
B*B
N*B
R-QB1
N-N3
O-O
B-Q2
B-Q3
P-B4
P-QN4. (E)  BOTVINNIK-KMOCH, LENINGRAD 1934
INI
SETUP,SUB1
P-B5
P-QN3
P-QN4. (E)
P*P
P(N4)*P
INI
SETUP,V18
N-QB3
P-KN3.  VARIATION 21, COTAP-275
Q-N3
B-N2. (E)
P*P
O-O
B-K2
N(QN1)-Q2
B-B3
N-N3
SAVE,9
B-N5
B-N5
B*N
B(KN5)*B
SAVE,10
N*B
P*B
O-O
Q-Q2
INI
SETUP,10
N*B
B*B.  LESS GOOD.  NOTE 10, COTAP-276
O-O
R-B1
R(KB1)-K1
Q-B2
N-K5.  EUWE-VAN DER HOEK, 1942
INI
SETUP,9
B-N5
B-B4.  NOTE 9, COTAP-276
R-Q1
INI
SETUP,VARY
P-Q4
P*P.  NOTE 1, COTAP-274
B*P
P-K3
N-KB3
INI
SETUP,VARY
P-Q4
N-QB3. (Q) NOTE 1, COTAP-274
```

```
P*P
Q*P
N-KB3
P-K4
N-B3. (E)
INI
SETUP,B
P-QB4
P-K4.  NOTE B, COTAP-272
N-KB3
P-Q3
P-Q4
N-Q2.  TRANSPOSING INTO OLD INDIAN DEFENSE
INI
SETUP,B
P-QB4
P-K3
P-Q4.  ALSO NOTE B, COTAP-272
P-Q4
P(B4)*P
P(B3)*P
P-K5.  TRANSPOSING INTO THE FRENCH DEFENSE
W/CENTR
MSG CENTER COUNTER GAME
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
LET MSMASK=77B.
LET MSCODE=0.
SWITCH EXPERIENCE,OFF
SWITCH LW ON;SWITCH NOR ON
INI
YES.  ANDERSSEN VARIATION
P-K4
P-Q4
P*P
Q*P
SAVE,A
N-QB3
Q-QR4
SAVE,VARY
P-Q4
P-K4.  VARIATION 2, COTAP-254
SAVE,V3
N-B3
B-KN5
B-QN5.CHECK
P-QB3
B-K2
B-N5
SAVE,7
O-O
B*N(KB6)
SAVE,8
B*B
N-K2
P*P
O-O
B-Q2
Q*P(K4)
R-K1
Q-B2
N-N5. (E)
P*N
B*B
R-K1
R*N
R*R
B*R
Q*B
B*P. (E)  LYSKOV-PERSITZ, 1955
INI
SETUP,8
B*B
B*N.  NOTE 8, COTAP-254
P*B
Q*P(B6)
R-N1
INI
SETUP,7
O-O
B*N(QB6)
P*B
Q*P(B6)
N*P
INI
SETUP,V3
N-B3
B-QN5.  VARIATION 3, COTAP-254
B-Q2
B-KN5
SAVE,10
B-K2
P*P
N*P
Q-K4
N(B3)-N5. (E)
B*B(K7)
Q*B
B*B.CHECK
K*B
Q*Q.CHECK
K*Q
N-QR3.  CENTER COUNTER GAME 1, COTAP-255
R(KR1)-K1
O-O-O
N*P(R7).CHECK
K-N1
N(R7)-B6.CHECK
P*N
N*P.CHECK
K-B1
N*R
K*N
R(QR1)-Q1.CHECK
K-K1
K-Q3.CHECK
N-K2
K-B4
P-R4
R-Q3
N-N1
R(Q3)-K3
N-B3
P-QN4
P-B3
P-B4
K-B2
P-QR4
R-QN1
P-B3
R-Q1
R-Q3
R*R
K*R
K-K1
P-R5
K-Q2
P-R6
N-Q4
R-QR1
N-R2
P-N3
P-B3
R-R4
N-N3
R-R5
P-N3
P-B4
N(N3)-B1
R-R1
N-Q3
K-Q4
N(Q3)-B1
K-B5
K-B2
R-K1
N-N3
R-K7.CHECK
N-Q2.CHECK
R*N.CHECK  TARRASCH-MIESES, GOTEBORG 1920
INI
SETUP,10
B-K2
N-QB3.  NOTE 10, COTAP-254
P-QR3. (E)
INI
SETUP,VARY
P-Q4
N-KB3.  VARIATION 5, COTAP-254
SAVE,12
N-B3
B-N5
P-KR3. (E)
B*N
Q*B
P-B3
B-QB4
P-K3
O-O
N(QN1)-Q2
B-B4
B-K2
R(KB1)-K1
O-O.  CENTER COUNTER GAME 2, COTAP-255
P-QR3
R(KB1)-K1
B-KN3
Q-N3
Q-Q3
R(QR1)-Q1
P-N4
N-B1
R(QR1)-Q1
B-Q3
N-K4
N*N
R*N
B*B
Q*B
R-Q2
P-QB3
Q-B2
Q-N5
Q-Q1
Q-KR5
Q-B3
P-QR4
P-QN3
R(Q1)-K1
R(K1)-Q1
P-B4
R-N2
Q-B3
R-B2
P-N4
P-N3
R-K5
K-N2
Q-N3
P-B4
P(N4)*P
P*P.  BOTVINNIK-KONSTANTINOPOLSKI
INI
SETUP,A
N-QB3
Q-Q1.  OBSOLETE.  NOTE A, COTAP-252
P-Q4
N-KB3
SAVE,SUB1
B-QB4
P-K3
N-B3
B-K2
O-O
O-O
Q-K2
N(QN1)-Q2
R-K1
N-N3
B-N3.  AND WHITE HAS THE EDGE.  ALEKHINE-SCHLECTER, CARLSBAD 1911
INI
SETUP,SUB1
B-QB4
B-N5. (Q)  ALSO NOTE A, COTAP-252
SAVE,SUB2
P-B3
B-B1
B-KN5
P-K3
P-B4. (E)
N(QN1)-Q2
N-B3
N-N3
B-N3
P-QR4
```

```
P-QR4
B-K2
O-O
O-O
Q-K2
P-B3
R(QR1)-Q1
N(N3)-Q4
N-K5.  FUDERER-BRONSTEIN, YUGOSLAVIA VS USSR, KIEV 1959
INI
SETUP,SUB2
P-B3
B-B4.  MORE OF NOTE A, COTAP-252
P-KN4
B-N3
N(KN1)-K2
INI
YES.  MARSHALL GAMBIT, COTAP-256
P-K4
P-Q4
P*P
N-KB3
P-Q4
N*P
N-KB3.  VARIATION 10, COTAP-257
P-KN3
B-K2
B-N2
O-O
O-O
SAVE,9
R-K1
N-N3
P-B3
N-B3
B-KB4
N-Q4
B-N3
P-QR3
N(QN1)-Q2
B-B4
N-B1
B-R3. (Q) CENTER COUNTER GAME 5
B-QB4
B-K3
Q-K2. (E)
P-QN4. (Q)
B-N3
R-K1
Q-K4
N-R4
Q-R4. (E)
K-N2
R*B. (E) AND WINS.  RABAR-MILIC, YUGOSLAVIAN CHAMPIONSHIP 1954
INI
SETUP,9
R-K1
N-QB3.  NOTE 9, COTAP-257
SAVE,SUB1
P-B3
P-K4
P*P
N*P(K4). (Q)
N*N
B*N
B-B3.  WINNING A PIECE
INI
SETUP,SUB1
P-B3
R-K1.  ALSO NOTE 9, COTAP-257
B-QN5
W/ALEKN
MSG  ALEKHINES DEFENSE--MODERN VARIATION.
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
LET MSMASK=77B.
LET MSCODE=0.
SWITCH EXPERIENCE,OFF
SWITCH LW ON;SWITCH NOR ON
INI
YES.  ALEKINES DEFENSE, COTAP-240, VARIATION 11
P-K4
N-KB3
P-K5
N-Q4
P-Q4
P-Q3
SAVE,A
N-KB3
B-N5
SAVE,VARY
B-K2
P-QB3
N-N5. (E)
B-B4
P-K6. (E)
B*P(K3)
N*B
P*N
B-N4
N-B2
O-O
N-Q2
R-K1
P-K4. (E)
B*N. (E)
Q*B
P*P
O-O-O
SAVE,4
Q-K2. (E)
P-K3
P*P
B*P
N-Q2.  PACHMAN-SEIMEANES, BUCHAREST, 1949
INI
SETUP,4
Q-K2.  NOTE 4, COTAP-241
Q-K3
N-Q2
P-Q4
N-B3
INI
SETUP,VARY
B-K2
P-K3
O-O
B-K2.  VARIATION 14, COTAP-240
P-B4
N-N3
P*P
P*P
N-B3
N-B3
P-QN3
B-B3
B-K3. (E)
P-Q4
P-B5
N-Q2
P-QN4. (E)
N*P(QN5)
R-N1
N-B3
SAVE,8
R*P
N*P(B4)
P*N
B*N(QB6)
Q-R4
Q-B1
B-QN5.  AND WINS.
INI
SETUP,8
R*P
B*N.  NOTE 8, COTAP-241
SAVE,8A
B*B
N*P(Q5)
N*P. (E)  NEDELKOVIC-JANOSEVIC, YUGOSLAVIAN CHAMPIONSHIP 1948
INI
SETUP,8A
B*B
O-O.  NOTE 8A, COTAP-241
Q-R4
Q-B1
R-N3
INI
SETUP,VARY
B-K2
P-K3
O-O
N-QB3.  VARIATION 15, COTAP-240
SAVE,9
P-B4
N(Q4)-K2
P*P
Q*P
SAVE,10
N-B3
B*N
SAVE,11
B*B
O-O-O
P-Q5. (E)
N-K4
B-B4.  UNZICKER-POMAR, BAD PYRMONT 1951
INI
SETUP,11
B*B
Q*P(Q5).  NOTE 11, COTAP-241
Q*Q
N*Q
B*P
R-QN1
B-K4
INI
SETUP,10
N-B3
N-N3. (Q)  GAME 10, COTAP-244
P-Q5. (E)
P*P
P*P
B*N
P*B
N(B3)-K4
N-N5
Q-Q2
P-B4
N-R5
P*N. (E)
Q-R6
N*P(B7).CHECK
K-Q1
N-K6.CHECK  ARONIN-MIKENAS, SEMIFINALS, USSR CHAMPIONSHIP 1951
INI
SETUP,VARY
B-K2
P*P.  IDEA VARIATION 10.
N*P
B*B
SAVE,10A
Q*B
N-N3
SAVE,10B
O-O
N(QN1)-Q2
R-Q1
INI
SETUP,10B
O-O
Q*P. (Q)  IDEA VARIATION 10B, COTAP-239
R-Q1
INI
SETUP,10A
Q*B
P-QB3.  IDEA VARIATION 10A, COTAP-239
Q-B3
N-B3
Q-QN3. (E)
INI
SETUP,VARY
B-K2
N-QB3.  IDEA VARIATION 11, COTAP-240
P-K6. (E)
P*P
N-N5
B*B
Q*B
P-K4
Q-B4.  KEPPER-NIEVERGELT, ZURICH 1954
INI
SETUP,VARY
B-K2
N-Q2.  IDEA VARIATION 12, COTAP-240
```

```
P-KR3. (E)
B-R4
SAVE,12A
N-N5. (E)
B-N3
P-K6. (E)
N(Q2)-B3
B-Q3.  MAROCZY-VOKOVIC, LONDON 1927
INI
SETUP,12A
N-N5. (E)
B*B.  IDEA VARIATION 12A, COTAP-240
P-K6. (E)
INI
SETUP,VARY
B-K2
B*N.  IDEA VARIATION 13, COTAP-240
SAVE,13A
B*B
P-QB3
P*P. (E)
Q*P
O-O
INI
SETUP,13A
B*B
P*P.  IDEA VARIATION 13A, COTAP-240
P-B4. (E)
INI
SETUP,VARY
B-K2
P-QB3.  IDEA VARIATION 15, COTAP-240
N-N5. (E)
B*B
Q*B
P*P
P*P
P-K3
O-O
N-Q2
P-QB4
N-K2
N-QB3
Q-B2
R-K1
N-KB4
N-B3
B-N5
B-Q2.  UNZICKER-SCHMID, NUREMBERG 1959
INI
SETUP,A
N-KB3
N-QB3.  NOTE A-1, COTAP-239
P-B4
N-N3
P-K6. (E)
P*P
N-B3
INI
SETUP,A
N-KB3
P*P.  NOTE A-2, COTAP-239
N*P. (E)
INI
SETUP,A
N-KB3
B-B4.  NOTE A-3, COTAP-239
B-Q3. (E)
INI
SETUP,A
N-KB3
P-KN3.  NOTE A-4, COTAP-239
N-N5. (E)
W/GORIN
MSG GORING GAMBIT
SW NO ON
SW EXP OFF
LET MSCODE 0
LET MSMASK 77
SW LW ON
SW LB OFF
INI
P-K4
P-K4
N-KB3
N-QB3
P-Q4
P*P
P-B3
SAVE GG4
P*P
B-QB4.  A2 GG-9
SAVE GG5
P*P
B*P/N2
SAVE GG6
P-Q4
P*P
SAVE GG7
N/B3-K2
O-O
N-KB3
R-K1
B-N5
Q-R4.  CHECK
B-Q2
Q-N3
P-QN4
B*P
B*B
Q*B.  CHECK
Q-Q2
Q-N7
Q-B1
Q-B6.  CHECK
N-Q2
B-R3
R-QN1
P-Q6
P*P
Q*P
Q-Q1
N-Q4
N-N3
Q-N3
R-N2
N-B5
N-B1
N-B3
P-N3
N-K4
SETUP GG7
Q-K2.  CHECK
B-K2
N-Q1
O-O
N-KB3
R-K1
N-K5
B-N5.  CHECK
P-B3
B-R3
Q-B3
R*N. CHECK
N-K3
P*P
SETUP GG6
P-Q3
O-O
SAVE GG7
B-N5
Q-N3
Q-Q2
N-N5
B-R4
N*P/B7
B*N
B*B.  CHECK
Q*B
Q*P
SETUP GG7
B-K3
B*B
P*B
Q-N3
Q-B1
N-N5
N-Q1
P-B4
SETUP GG6
B-N5.  CHECK
N-B3
SAVE GG7
P-Q3.  INACCURATE GG - 10
Q-N3
SAVE GG8
B-K3.  I
B*B
P*B
O-O
Q-K2
N-K2.  (E)
SETUP GG8
N-R3.  II
O-O-O
O-O
P-N4
B*P
R/R1-N1.  (E)
B/KN5*N
N-Q5.  (E)
SETUP GG8
Q-K2.  III
O-O
B*N
B*B
N-B3
R/KB1-K1
O-O
P-K5
P*P
N*P
N*N
R*N
B-K3
B-N4
SETUP GG7
N/KN1-K2
N-N5
N-K4
Q-R5
P-KN3
Q-R6
N*B
Q-N7
R-B1
N*P/R7
N-B3
N-B6.  CHECK
K-K2
O-O-O
B-R6
N/B3-Q5.  CHECK
K-K3
N-B4.  CHECK
K-K2
N*P/N6
K-K3
N-B4.  CHECK
K-K2
N/B4-Q5.  CHECK
K-K3
Q-N4.  CHECK
K-Q3
B*B.  CHECK
N*B
P-K5.  CHECK
K-B4
N-K4.  CHECK
K-N4
N/K4-B3.  CHECK
K-R3
Q-QR4.  CHECK
N-R4
Q-N5.  CHECK
N*Q
N-N4.  CHECK
K-N3
N-R4.  MATE
SETUP GG7
Q-B3
Q-N3
N/KN1-K2
O-O
```

O-O
N-Q5
SETUP GG7
N-B3
Q-B2
P-Q3
O-O-O
SAVE GG9
B*N
Q*B
B-K3
R/KR1-K1
B*B
Q*B
O-O
P-K5
N-K1
R-K3
SETUP GG9
O-O
P-K5
N-N5
N-Q5
B-QB4
P*P
P*P
P-KR4
P-KR3
N-N5. (E)
P*N
P*P
Q*P. CHECK
P-B4
B-B4
P*Q
B*Q
K*B
SAVE GG18
N-B7. (Q)
N-B6. CHECK (EE)
P*N
B*P/B6
N*R/R8
R*N
SETUP GG18
N-N5. CHECK
N*N
SAVE GG19
N-K6. CHECK
K-N3
SETUP GG19
B*N
K-N3. (E)
SETUP GG18
N-K6. CHECK
N*N
B*N
R-R5
SETUP GG5
N-B3. A22 GG - 12
N*P
SAVE GG6
B-N5. A221
P-K5. (E)
SAVE GG7
B*N. CHECK (Q)
P*B
P-Q4
B-N3
N-K5
B*P
N*P/QB6
B*N. CHECK
P*B
Q-B2
SAVE GG12
N-N4
B-N2
P-QR4
R-Q1
B-Q2
P-QR4
N-R2
B-R3
N-B1
O-O
N-K2
R/KB1-K1
SAVE GG18
Q-B1
Q-QB5
N-N3
P-K6. (E)
P*P
N-K5
SETUP GG18
O-O
Q-Q2
SETUP GG12
N-Q4
Q*P/B6
SETUP GG7
P-Q4
P*N
P*B
Q*Q. CHECK
N*Q
P*P
R-KN1
B-R6
SAVE GG11
B-N5
N-N5
SETUP GG11
N-K3
O-O-O
SAVE GG12
N*P. (Q)
N-Q5. (E)
SETUP GG12
B*N
P*B
SAVE GG13
N*P
R/KR1-K1. CHECK
SAVE GG14
N-K3
B-B4
SAVE GG15
B-Q2
B*P
R-QB1
B-N3
B-B3
N-Q4. (E)
SAVE GG18
B*P
P-B4. (E)
K-B1
N*N. CHECK
P*N
R*P
SETUP GG18
R-N3
P-B4
SETUP GG15
R*P. (Q)
B-N3
SETUP GG14
B-K3
P-N4. (E)
R-Q1
N-N5
R*R. CHECK
K*R
SETUP GG13
P-KB3
R/KR1-K1
K-B2
P-N4
N*P
B*N
SAVE GG16
R*B
R-Q8
R-KN1
R*R
K*R
R-K8. CHECK
K-B2
R-R8
K-N2
R-Q8
P-N4
N-Q4
SETUP GG16
K*B
R-K7. CHECK
SETUP GG13
B-Q2
R/KR1-K1
SAVE GG14
O-O-O
N-N5. (E)
SETUP GG14
P-KB3
P-N4. (E)
SETUP GG6
P-Q3. A222 GG - 13 AN INSIPID MOVE
Q-N3
Q-Q2
N-KN5
N-K4
B-N5
P-B3
P-B4
SAVE GG10
N-N3. (QE)
P-K5. (E) (A) (III) VELIMIROVIG GG - 14
SAVE GG11
P*P
B-B4
SETUP GG11
P*B
P*N
P*P
N/N5-K4
SETUP GG10
P-KR3. (Q) (B)
P*N
P*N
P*N
P*B
N-Q5
P-KN3
B-K3
SETUP GG10
P*B
P*N
SAVE GG11
P*P. (I)
B-K3
SAVE GG12
B-Q3
N*P/N5
O-O
R-Q1
N-K1
O-O
Q-K2
N*B
N*N
Q-R3
R-Q1
N*P/B7. (E)
B-N5
R*N
R-K1
B-N5
Q-B2
Q-QN3
B-K7
N*P. CHECK
K-R1
R-B1
R-KB1
Q-Q1. (E)
Q-R4
Q*B
Q*N
R-Q5. ALEKHINE-VARLINSKY, OIDESSA 1918
SETUP GG12
P-QR3
R-Q1

SAVE GG13
Q-B2
B-N6
Q-B5
R-Q8. CHECK
K-K2
Q-Q1
Q-B3
R-K8. CHECK ZHURAVLEV-BORINSKY USSR TEAM CH. 1969
SETUP GG13
Q-K2
B-B5. (E)
B-K3
B*Q
B*Q
B*B
B*R
B*P. WINNING
SETUP GG11
N-N5. (E) (II)
P-K6. (E)
P*P
N*P/N5
SETUP GG10
N/K4-N5. (E)
B-K2. (E)
P-KR3
N-B3
SAVE GG12
P-KR4
P-KR3
N-R3
SETUP GG12
N-R4
O-O
SETUP GG12
P-Q4. (EQ.
P-K5
N-R4
N-Q4. (E)
Q-Q1
P-N3
Q-N3
Q-Q1
SETUP GG5
B-K2. (QQ) A23 GG-15
Q-Q5
SAVE GG6
P-Q3
Q*P/B7. CHECK
K-Q2
B-K6. MATE
SETUP GG6
N-R3
B*N
O-O
N*P
P*B
Q-KR5. WITH A CRUSHING POSITION
SETUP GG5
P-B7
Q*P/B2. STEIN - SHISHOV, MOSCOW 1959
B-N5. CHECK
N-B3
P-Q3
O-O
B*N
Q*B
N-B3
B-KN5
P-KR3
B-R4
O-O
R/KB1-K1
R-K1
R/QR1-Q1
Q-K2
B-QN5
B-N5
P-K5
P*P
B*N/QB6
P*B
R*P
Q-B1
B*N
P*B
R*R
Q*R. ALTHOUGH SOMEHOW WHITE FAILED TO WIN
SETUP GG5
P-KR3. (Q) A25 GG-16
N*P
B-N5
O-O
B*N
P*B
P-Q3
Q-N3
Q-B3
N-Q4
B-Q2
P-B4. LEVY-BOUAZIZ,OREBRO 1966
SETUP GG5
B-QN5
0-0
SAVE GG6
P-Q3
P-QR3
B-R4
P-QN4
B-N3
Q-N3
Q-B3
N*P/B3
N/KN1-K2
B-N2
0-0
N-Q5
Q-N3
P-QR4. ECO
SETUP GG6
P*P
B*P/N2
SAVE GG7
N-B3
N-N5
O-O
P-K5
SAVE GG9
P-Q4
PXN
PXB
Q-R5
P-KR3
N-K4
P-B6
Q/N*P
N-Q5
R/QR1-Q1
P-B4
B-B1
SETUP GG9
N-N5
Q*N
P-Q4
P-K6
P*P
N*P/K
R-B2
N*Q
B*Q
N*R
P*B
N-N5
SETUP GG9
NXP
BXN
P-Q4
B-Q3
N-N5
N-KB3
NXB
N*N
Q-B3
P-B4
B-B4
K-R1
P-KN3
B-B2
SETUP GG7
P-B3
Q-N3
K-B1
P-K5
B-K2
B*N
R*B
P*P
B*P
N-B3
P-KN3
N-K4
BXB
QXB
SAVE GG14
P-Q4
N-B6
SETUP GG14
R-N2
N-B6
P-Q4
N-N5
P-KR3
R/KB1-K1
N-K2
N-K6
BXN
RXB
R-QN1
R/1-K1
SETUP GG6
Q-B3
P-K5
P*P
B*P/N2
Q-N3
B-Q3
Q-K3
P-QR3
B-R4
Q-B2
SETUP GG6
P-B7
Q*P/B2
N/1-K2
P-QR3
B-R4
P-QN4
B-N3
B-N2
O-O
Q-B3
SETUP GG5
B-QB4
B*P. CHECK
K*B
Q-Q5. CHECK
SAVE GG7
K-K1
Q-R5. CHECK
SAVE GG8
K-B1
SETUP GG8
P-N3
Q*B
Q-K2
Q*P/B3
SETUP GG7
K-B1
Q*B. CHECK
Q-K2
Q*P/B3
SAVE GG9
Q*P/K5. CHECK
B-K3
SAVE GG10
P-Q4
O-O. BARDEN
SETUP GG9
Q-N5
O-O
Q*Q
N*Q
SETUP GG5

P-Q3. A28 GG-16
N*P
SAVE GG6
B-K2. A281
Q-N3
N-R4
B*P. CHECK
K-B1
Q-R4
P-B3
B*N
K*B
O-O
SAVE GG11
P-QN4. (A)
Q-B2
B-N5
N-Q4
B-B3
B-K3
Q-K1
P-B3
B-Q2
R/QR1-Q1
N-B5
B-B1
Q-K4
P-QN3
N-N3
N/B3-K2
P-B4
P-B4
Q-K2
N-KB3
R-K1
N-N3
B-B3
P-K5. (E)
B*N
N-B5. (E)
Q-QB2
R*B
P*P
N-Q6
R-K3. (Q)
P*P
B*P
Q-B5. (E)
Q-K2
N-B8. (E)
B-Q5. CHECK
R*B
R-K8. CHECK
K-B2
R-K7. CHECK
K-N3
Q-QB2. CHECK
R/Q4-KB4
P-KR3
N*N
P*N
Q-Q5. CHECK
SETUP GG11
P-KR3. (QE) (B)
B-B4
P-QN4
Q-B2
B-N5
N-Q4
B-B3
N-B5
B*N/KB4
P*B
P-Q4
R/KB1-K1
K-B2
R-K6
R-K1
R/QR1-K1
R*R
P*R. CHECK
K-N1
N-K2
P-Q5
Q-K4
N-B5
P-QN3
N-N3
N-N3
N-Q4
N-B5
Q-R4
K-B1
K-R1. PENROSE-SZABO, HASTINGS 1956/57
B-K5
SETUP GG11
K-B2
P-K5. (E)
R-K1. (E)
P*P
B*P
R-Q1. (E)
SETUP GG6
B-K3. A282 GG-19
B*B
P*B
Q-N3
SAVE GG8
Q-Q2
Q*P/N7
R-N1
Q-R6
B-K2
O-O
SAVE GG11
N-B3
R-Q1
O-O
Q-K2
P-K4
N-Q5
N*N
P*N
N-Q1
B-K3
P-B4
P*P. EP
N*P
R/QR1-B1
K-R1
P-QN3
Q-N2
N-N5
N-Q5
Q-Q3
N-B4
N-K4
P-N3
N-B3
N-R5
N-Q5
B-Q1
P-B4. (E)
P*P
B*P/B4
B-N3. CHECK
K-R1
N*P
B-R6. (E)
R-B7. (E)
SETUP GG11
B-B3. (E) (II)
R-Q1
N/N1-K2
SETUP GG8
Q-B1
N-KN5
SAVE GG9
N-Q5. (Q)
Q-R4. CHECK
N-B3
N-N5. TREMENDOUS FOR WHITE
SETUP GG9
N-Q1
P-B4
SAVE GG10
P-B3
O-O
Q-Q2
P-B5
P-K4
N-K6. ONE WAY FOR BLACK TO GIVE UP THE FIGHT
SETUP GG10
N-R3. REFUTED IN STEIN-MURATOV,TALLIN 1959
P-B5
N/R3-B2
N*N
N*N
P*P
N-Q1
P-K7. (E)
B*P
N-Q5
Q-Q2
O-O
P-B3
N-B4
B-B3
N-R5
B-Q5. CHECK
B-K3
B*B. CHECK
Q*B
Q-N5
R-B5
N-K3
Q-QN3
Q*P/K5
Q*P
Q*R
N*P. CHECK
N*N
Q*R. CHECK
SETUP GG10
P-KR3. FAILED IN BONNER-SACARELLO, SIEGEN 1970
N-B3
N-K2
O-O
N/K2-B3
N-KR4
B-K2
N-N6
R-KN1
P-B5
B-B3
B-K3
Q-Q2
N-B4
N-Q5
P*P
Q*P
N*Q
N*Q
N*P/B7. CHECK
K-Q2
P*N
K*N
N-Q5. CHECK
K-Q2
R*B
SETUP GG10
B-K2
P-B5. (E)
SAVE GG11
P-K4. B/NNER-RUBINETTI,SEIGEN 1970
O-O
SETUP GG11
P*P
O-O. (E)
SAVE GG12
N-R3
N-Q5
B*N
B*B
N/R3-B2
B*N
Q*B
R*P. KLOVAN-DARENICK, RIGA 1962
Q-Q2
Q*P
O-O
R/QR1-KB1
R/QR1-B1
Q*P/R7

P-KR3
N-N6
SETUP GG12
P-KR3
N-R3
SAVE GG13
P-KN4. (I)
P*P
SETUP GG13
N-KB3. (II)
N-B4
P-B3
N-N6
R-KN1
P*P
N-B2
B-K3
Q-Q2
R/QR1-K1
SAVE GG18
O-O-O. (Q)
Q*N
SETUP GG18
N-K4
N*N
P*N
R-Q1. (E)
SETUP GG13
P*P. (III)
Q-R4. CHECK
SETUP GG11
B*N
B*B
SAVE GG12
N-KB3. WISE-HOOGENDOOREN, HASTINGS 1965/66
B*N
P*B
O-O
P-K4
N-Q5
R-B1
Q-KR3. (E)
P-B3
Q-R5. CHECK
K-Q2
Q*P. CHECK
N-B2
N*P. CHECK
K-B2
Q-N7. (E)
K-N3
N-R7
SETUP GG12
N-B2
P*P. (E)
N*B
Q-N5. CHECK
P-B3
Q*N
P-KN3
O-O
Q*P
N-Q5. (E)
P*N
P*P
Q-Q2
R/QR1-K1. CHECK
N-K2
Q-B6
SAVE GG20
R-KN1
Q-B7. CHECK
K-Q1
R*N. WINS
SETUP GG20
O-O-O
R*N
Q-N4
Q-QB3. CHECK
K-N1
Q-B7. CHECK
K-R1
R/KB1-B7
R-QN1
Q*P/Q6
R/R1-QB1
P-KR3
R-B8. CHECK
K-R2
R-Q8. (QQ)
Q*R. CHECK
SETUP GG4
P-Q4
P/K4*P
Q*P
P*P
SAVE GG6
N-B3
N-B3
B-QN5
B-K2
SETUP GG6
B-KN5
B-K2
SAVE GG7
O-O-O
B-K3. (A)
SAVE GG8
N-B3
N-B3
SAVE GG9
Q-KR4
Q-R4. (E)
SETUP GG9
Q-QR4
O-O
SETUP GG8
B-N5. CHECK (B)
N-B3
SAVE GG9
N/N1-K2
O-O
Q-Q2
R-QB1. IS GOOD FOR WHITE - KERES
SETUP GG9
Q-QR4. (II)
O-O
B*N/QB6
P*B
Q*P/B6
R-B1
Q-R6
R*N. (E)
P*R
N-K5. GIVES AN ATTACK - KERES
SETUP GG9
N-B3
O-O
SAVE GG10
Q-Q2
Q-R4
N-Q4
N*N
Q*N
P-QR3
B-Q3
P-R3
B-R4
P-QN4
P-QR3
R/KB1-QB1
N-QN1
R/QR1-QN1. LEVY - KRAIDMAN, LUGANO 1968
SETUP GG10
B*N/QB6
P*B
N-K5
Q-B2
R/KR1-K1
R/QR1-N1. (E)
SAVE GG13
N-N4
B*N
B*N
B*B. (E)
SETUP GG13
N-Q3
P-B4
Q-K5
B-Q3
Q-K2
N-Q2. (E)
SAVE GG16
K-N1
Q-R4. (E)
SETUP GG16
P-QN3
P-B5. (E)
N-N2
P*P
N*P
B*N
R*B
P*P/R7
Q-R6
Q-B6
R*B
R*N
SETUP GG8
N-R3
N-B3
Q-KR4
Q-R4
SAVE GG10
Q-R4
Q*Q
N*Q
N-K5
SAVE GG12
B*B
B*N
SETUP GG12
B-K3
B*N
P*B
P-Q5
SETUP GG10
N-B4
P-Q5. (E)
N*B
P*N/K3
B*N. (Q)
B*B
Q-K4
O-O-O. (E)
Q*P/K6. CHECK
K-N1
SAVE GG15
N-K4
R/KR1-K1
Q-N4
Q*P
SETUP GG15
N-K2
P-Q6. (E)
SETUP GG15
N-N5
P-Q6. (E)
B*P
R*B
R*R
Q*N
R-QN3
Q-N4. CHECK
K-N1
Q*P
R-QB1
R-Q1. PENROSE, VITUMEN LUGANO 1968
SETUP GG8
N/KN1-K2
N-B3
Q-QR4
P-KR3
B-R4. PADEVSKY, FILIP, LYONS, 1955
P-KN4
B-N3
P-R3
N-Q4
B-Q2
N*N
P*N
Q-Q4. CHANCES BOTH SIDES
SETUP GG7
B*N. (B112) GG-21

B*B
Q-B5
B*N. CHECK
P*B
Q-K2. CHECK
Q*Q. CHECK
K*Q
O-O-O
B-K3
B-Q3
N-B3
SETUP GG7
N-B3. (G113) GG-21
N-B3
SAVE GG8
B-N5
O-O
B*N/QB6
P*B
SETUP GG8
Q-KR4
P-KR3
SAVE GG9
O-O-O
O-O
B-Q3
P*B
N*P/N5
R-K1. (E)
N*P/Q5
Q*N
B-R7. CHECK
K-B1
R*Q
N*R
SETUP GG9
B-Q3
B-K3
O-O-O
Q-R4
R/KR1-K1
O-O-O
SETUP GG8
Q-QR4
O-O
O-O-O
B-K3
SAVE GG10
B-Q3
P-KR3. (E)
SAVE GG11
B-R4
P-R3
SETUP GG11
P-KR4
N-QN5
B*N
N*B. CHECK
R*N
B*B
R/KR1-Q1
Q-N3
SETUP GG10
B-K3. (Q)
P-QR3
N-Q4
B-Q2
N-N3
P-Q5
N*P
N-QR4
SETUP GG10
B-QB4
P-QR3. (E)
SAVE GG11
B-N3
N-QN5. (E) PENROSE - PRAMESHUBER, MUNICH, 1958
B*N
P*B
N*P
N*N
R-Q3
P-QN4
Q-R4
R-B1
SETUP GG11
B*N
B*B
SAVE GG12
N*P
P-QN4
SETUP GG12
B*P/Q5
B*N
SAVE GG13
P*B
B*B
P-B4
P-QN4. (E)
SETUP GG13
B*N
B*P/N7. CHECK
K*B
Q-B3. CHECK
SETUP GG13
B*B
B*P. CHECK
K*B
Q-B3. CHECK
SETUP GG6
B-QN5. CHECK
N-B3
SAVE GG7
N/KN1-K2. PENROSE - BOLBOCHAN MOSCOW 1956
B-K2
O-O
P-QR3
B*N. CHECK
P*B
N-R4
O-O
B-B4
Q-R4
B-Q2
Q-N4
N/K2-B3
Q-N2
R/KB1-K1
B-KB4
R/QR1-B1
R/KB1-K1
P-KR3
N-Q2
B-K3
B-K3
B-B4
P-QB4
SETUP GG7
B-N5
B-K2
SAVE GG8
N-B3
O-O
Q-QR4
N-KN5. (E)
B*B
Q*B. CHECK
B-K2
R-Q1
O-O
P-Q5
SETUP GG8
B*N/KB6
B*B
Q-B5. (E)
B*N. CHECK
SAVE GG10
Q*B. (Q)
O-O
N-K2
Q-N3
B*N
P*B
O-O
P-QB4
SAVE GG14
N-B4
P-Q5
Q-KN3
B-B4
N-Q5
Q-Q1
Q-K5
B*P
N-K7. CHECK
K-R1
Q*P/B. (E)
P-Q6. (E) PENROSE - BARDEN LONDON 1958
SETUP GG14
P-QN3
P-Q5
Q-N3
B-B4
R/QR1-B1
Q-R4
P-QR4
Q-Q7. WITH ADVANTAGES TO WHITE - BARDEN
SETUP GG10
P*B
Q-K2. CHECK
Q*Q. CHECK
K*Q
SETUP GG7
N-B3
B-K2
SAVE GG8
N-K5
B-Q2
B*N
P*B
SAVE GG10
O-O
O-O
N*B
Q*N
B-N5
Q-B4
Q-Q2
R/QR1-N1. GHITESCU - PUREVZHAV VARNA 1962
SETUP GG10
O-O
O-O
N-R4
R-K1
P-QN4
B-Q3
P-KB4
P-QR4. (E) NYHOLM - ALEKHINE STOCKHOLM 1912
SETUP GG10
N*B
Q*N
O-O
R-QN1
P-QN3
O-O
Q-Q3
B-Q3
SAVE GG14
P-KR3
R/KB1-K1
B-Q2
R-K3
R/QR1-K1
R/QN1-K1
R*R
Q*R. KLOVAN-AVERBACH USSR 1969
SETUP GG14
B-N5
N-N5
P-KR3
N-K4
Q-Q1
P-KB4
B-B4
P-N4. (E)
SAVE GG18
B*P
P-B5
SETUP GG18
B-Q2
P-B5
SETUP GG8
O-O
O-O

```
SAVE GG9
Q-Q1
B-KN5
P-KR3
B-R4
B-K2.  VELIMIROVIL-MOLMOV YUGO.  VS.  USSR 1966
R-B1
B-KN5
N-K5
B*B
N*B
N-QN5
Q-N3
P-QR4
R/KB1-Q1
SAVE GG16
N/KB3-Q4
B-N3
SETUP GG16
N/QN5-Q4
R-Q3
SETUP GG9
Q-QR4
B-Q2
SAVE GG10
N-Q4
N*N
Q*N
B*B
N*B.  BLACK BAD RETI - BREYER BADEN 1914
N-K5
B-K3
P-QR3
N-B3
B-B3.  (E)
SETUP GG10
R-Q1
P-QR3
B-K2.  PENROSE - FAIRHURST GLASGOW 1958
P-Q5.  (E)
SAVE GG12
N*P
N-QR4
SETUP GG12
N-N1
B-QB4
P-B3
N-K5
Q-B2
N*P/KB7
P*P
B-R2
K*N
N*P
N*N
Q-R5.  CHECK
SAVE GG18
P-KN3
B*N.  CHECK
K-K1
Q*P/R7
R*B
Q-N8.  CHECK
SETUP GG18
K-B1
B*N
B-B3
B-N4.  CHECK
SETUP GG18
K-N1
R/KB1-K1
N-QB3
B*N.  CHECK
SAVE GG20
K-R1
B-QB3.  (E)
SETUP GG20
R*B
Q*R.  CHECK
K-R1
Q-B7
B-KN5
B-B3
R-KN1
P-R3.  (E)
B-QB1
R/QR1-Q1
P-KR3
R-Q5
B-Q3
Q-N6
B-K4
B*B
N*B
R/Q5*N
B-Q2
R-K7
Q-Q1
Q-B7
B-B3
R/K1-K6
Q-Q8.  CHECK
K-R2
Q-Q7
R-N6
P-R3
R*P/KN7
Q-Q3.  CHECK
SETUP GG4
N-KB3
P-K5
SAVE GG5
N-KN5.  PENROSE - BLAV - HASTINGS 57158
P*P
P-Q4
B-QN5
B-Q2
N-B3
N-K2
O-O
P-QB3
B-Q3
Q-B1
R-K1
B-K3
N-KN5
P-KN3
N*B
Q*N
B-KN5
P-KR4
Q-N3.  (E)
SAVE GG15
O-O-O
R/QR1-B1
SETUP GG15
P-N3
P-KR3
N-R3
B-B6
R-KN1
R/QR1-B1
SETUP GG15
Q-Q2
R-K2
B-N2
Q-R3
P-N3
P-K6.  (E)
P*P
N*P
SAVE GG19
P/B3*N
B-N5
SETUP GG19
N-B4
B*N
P*B
R*P.  CHECK
Q*R
N-B7
SETUP GG5
N-Q4
B-QN5
P-QR3
B-R4
B-K2
N*P
N*N
Q*N
N-N3
B-B2
P-QB4
Q-K4.  TCHIGORIN - GUNSBERG 1890
SETUP GG5
N-K5
Q-K2
SAVE GG6
P-Q4
P/K5*P.  (EP)
P-B4
N*P
SAVE GG8
B*P.  (Q)
N*N
P*N
P-B3.  LEVY-LITTLEWOOD HASTINGS 1969/70
SETUP GG8
N*N
P*N
B*P
P-B3
SAVE GG10
B-N5.  CHECK
K-Q1
Q*P.  CHECK
K-B2
SETUP GG10
Q-R5.  CHECK
P-N3
B*P.  CHECK
P*B
Q*R
N-B3
O-O
P*N
P*P
B-KB4. VELIMIROVIC-TRIFUNOVIC YUGO.  1963
SETUP GG6
N-B4
P*P
N-K3
P-Q5
N/K3-Q5
N*N
N*N
Q-K4.  (E)
SETUP GG6
P-B4.  (E)
N*P
N*N
P*N
P-Q4
P-B3
SAVE GG9
B-N5.  CHECK
K-Q1
SETUP GG9
N-N4
Q-N5.  CHECK

P-B3
Q-N3
B-K2.  POSITION CHANCES TO BOTH SIDES
SETUP GG4
P-Q6
B*P
SAVE GG5
P-Q4
N/QN1-Q2.  (E)
SETUP GG5
B-B4
O-O
P-Q3
P-QN4.  (E)
B-N3
P-QR4
P-QR3
N-R3
B-N5
N-B4
SAVE GG10
B-R2
P-N5.  (E)
```

```
P*P
P*P
N-K4
P-N6.  (E)
SETUP GG10
N/KN1-K2
N*B
P*N
B-K3
O-O
B-K2
P-Q4
N-Q2.  (E)
B*B
Q*B
P*P
N*P
Q-B2
Q-B4.  (E)
SETUP GG5
P-Q3
P-KR3
N-B3
O-O
B-K2
N-Q4
O-O
N-Q2
R-K1
P-KB4
B-B1
Q-B2
P-KN3
N/Q2-B3
SAVE GG12
B-N2
B-Q2
B-Q2
R/QR1-K1
P-QR3
N-KN5
R-K2
P-K5
P*P
N*N
B*N
P*P
Q-Q4
N-B3
N-Q2
P-B4
Q-B4.  CHECK
B-K3
Q-R4
B-Q2
Q-B4.  CHECK
B-K3
SETUP GG12
N-Q2
B-B4
N-N3
B-Q3
N-N1
P-B5.  (E)
N/N1-Q2
P*P
P/B2*P
B-KN5
N-B3
Q-N3.  CHECK
B/QGA
MSG QUEEN-S GAMBIT ACCEPTED
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
LET MSMASK=77B.
LET MSCODE=0.
SWITCH EXPERIENCE,OFF
SWITCH LB ON;SWITCH NOR ON
INI
YES.  QUEENS GAMBIT ACCEPTED, COTAP-451
P-Q4
LET MSCODE=40    SELECT 50 PERCENT OF THE TIME.
P-Q4
LET MSCODE 0
P-QB4
P*P
N-KB3
N-KB3
P-K3
P-K3
B*P
P-B4
O-O
P-QR3
SAVE,7PQR4
Q-K2
P-QN4
SAVE,VARY
B-Q3
P*P.  COLUMN 8, COTAP-452
SAVE,NOTE8
P*P
B-K2
P-QR4
P*P
R*P
B-N2
SAVE,NOTE11
N(QN1)-Q2
O-O
N-N3
B-B3
R-R1
Q-N3.  BARCZA-KERES, BUDAPEST 1952
INI
SETUP,NOTE11
N-B3.  IDEA VARIATION 4, COTAP-452
O-O
INI
SETUP,NOTE8
N*P.  NOTE8, COTAP-454
B-K2
P-QR4
P*P
INI
SETUP,VARY
B-N3.  COLUMN 9, COTAP-452
B-N2
SAVE,IDEA5
P-QR4
N(QN1)-Q2
SAVE,10PK4
P*P(QN5)
P*P(QN4)
R*R
Q*R
N-B3
P-N5
N-QN5
Q-R4
P-K4
B-K2
P-K5
N-Q4
B-N5
B-R3
B*N
B*N
B-QB4
B*B(QB5)
Q*B
N-N3
Q-B1
P-R3
SETUP,10PK4
P-K4.  A GAMBIT WHICH CAN BE DANGEROUS.
P*P(Q5)
SAVE,MOVE11
P-K5
N-Q4
P*P
P-Q6
Q*P
N-B4
Q-B4
N-N3
Q-B2
P*P
SETUP,MOVE11
N*P
B-B4
R-Q1
Q-N3
B-K3
O-O
SETUP,MOVE11
P*P
N-B4
B-QB4
P-Q6
Q-K3
P-QR4.  NEISHTADT QGA
INI
SETUP,IDEA5
R-Q1.  IDEA VARIATION 5, COTAP-452
N(QN1)-Q2
N-B3
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
Q-B2.NOP
SWITCH LB ON;SWITCH NOR ON
P-Q5
P-B5
P*P
P*P
B-B2
B-Q3
P-KR3
O-O
P-K4
N-B4
B-N5
N-R4
INI
SETUP,VARY
B-N3
B-N2
P-QR4.  COLUMN 12, COTAP-453
N(QN1)-Q2
SAVE,COL13
P*P(QN5)
P*P(QN4)
R*R
Q*R
N-B3
P-N5
N-QN5
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH NOR,ON
Q-N1.NOP
SWITCH LB ON;SWITCH NOR ON
P-K4
P*P
N(KB3)*P
N-B4. (E)
P-K5
N(KB3)-Q2
B-KB4
N*B
N*N
B-K2
R-Q1
B-R3
N-B7.CHECK
Q*N
Q*B
O-O.  KOPILOV-FLOHR,  MOSCOW 1953
INI
SETUP,COL13
P-K4.  COLUMN 13, COTAP-453
P*P(Q5)
SAVE,NOTE29
P-K5
SW LB OFF
N-N5
SW LB ON
P*P(QN5)
N-B4.  NOTE 31 TAKEN IN LIEU OF REST OF COLUMN.
B-QB4
P-Q6
INI
SETUP,VARY
B-N3
B-N2
N-B3
```

```
N(QN1)-Q2
R-Q1
B-Q3.  COLUMN 16, COTAP-453
SAVE,NOTE41
P-K4
P*P
N*P(Q4)
Q-N1
SAVE,CHAOS
P-N3
P-N5
N-R4
O-O
SETUP,CHAOS
N-B3
P-N5
N-Q5
P*N
P-K5
N*P
N*N
O-O
INI
SETUP,NOTE41
P-Q5.  NOTE 41, COTAP-455
P-K4
SETUP,7PQR4
P-QR4.  RUBINSTEIN-BOTVINNIK VARIATION
N-B3
Q-K2
Q-B2
N-B3
B-Q3
SAVE,MOVE10
P-Q5
P*P
B*P/Q
O-O
P-R3
B-Q2
P-K4
N-Q5
Q-Q3
N*N.  EQUALITY  ECO-10-113
SETUP,MOVE10
B-Q2
O-O
R(QR1)-B1
B-Q2
P*P
B*P/QB4
N-KN5
Q-N3
R-N1
B-N5
R(KB1)-Q1.
N-K4.  EQUALITY ECO D27-10-113
SETUP,MOVE10
P-QN3
O-O
B-N2
P*P
P*P
P-K4
SETUP,MOVE10
P*P
B*P/QB4
B-Q3
P-K4
SETUP,MOVE10
R-Q1
O-O
P-R3
B-Q2
INI
YES,  SUPPLIMENTAL VARIATIONS, QUEEN GAMBIT ACCEPTED, COTAP-456.
P-Q4
P-Q4
P-QB4
P*P
N-KB3
N-KB3
P-K3
P-K3
B*P
P-B4
Q-K2.  COLUMN 7, COTAP-456
P-QR3
P*P
B*P
SAVE,NOTE23
P-K4
P-QN4
B-Q3
N(QN1)-Q2
O-O
B-N2
N(QN1)-Q2
O-O
P-K5
N-N5
N-K4.  TAIMANOV-BAZIN, BUENOS AIRES 1960
INI
SETUP,NOTE23
O-O.  NOTE 23, COTAP-458
N-B3
P-K4
P-QN4
P-K5
P*B
P*N
P*P
INI
YES, QUEEN GAMBIT ACCEPTED, SUP VARIATIONS 14-24, COTAP-465
P-Q4
P-Q4
P-QB4
P*P
SAVE,VARY
N-KB3
N-KB3
SAVE,NOTE19
Q-R4.CHECK  (MANNHEIM VARIATION)
B-Q2.  COLUMN 20, COTAP-465
Q*P(QB4)
P-K3
N-B3
N-R3
P-K4
P-B4
SAVE,NOTE32
B-K2
P*P
N*P
R-B1
Q-Q3
N-QN5
Q-N1
P-K4
N-B3
B-QB4
INI
SETUP,NOTE32
P-Q5.  NOTE 32, COTAP-466
P*P
P*P
N-QN5
INI
SETUP,NOTE19
N-B3.  NOTE 19, COTAP-466
P-QR3
P-QR4
N-B3
P-K4
B-N5
INI
SETUP,VARY
P-K3.  COLUMN 23, COTAP-465
P-K4
SAVE,NOTE45
B*P
P*P
P*P
B-QN5.CHECK
N-B3
N-KB3
N-B3
O-O
O-O
B-N5
B-KN5
N-B3.  MARSHALL-JANOWSKI, NEW YORK 1924
INI
SETUP,NOTE45
P*P.  NOTE 45, COTAP-467
Q*Q.CHECK
K*Q
B-K3
INI
SETUP,VARY
P-K4.  COLUMN 24, COTAP-465
P-QB4
SAVE,NOTE49
P-Q5
N-KB3
N-QB3
P-K3
B*P
P*P
P*P
B-Q3
N-B3
O-O
O-O
P-QR3
P-QR4
B-N5
INI
SETUP,NOTE49
N-KB3.  NOTE 49, COTAP-467
P*P
Q*P
Q*Q
N*Q
B-Q2.  CAPABLANKA-RUBINSTEIN, KISSENGEN 1928
INI
YES.  QUEEN PAWN OPENINGS WITHOUT EARLY P-QB4
P-Q4
P-Q4
SAVE,VARY
N-KB3
N-KB3
P-K3
B-B4.  COLUMN 6, COTAP-531
P-B4
P-K3. (E)
Q-N3
Q-B1
N-B3
P-B3
B-Q2
N(QN1)-Q2
R-B1
B-Q3
B-K2
O-O
O-O
P-KR3
INI
SETUP,VARY
N-KB3
N-KB3
B-B4.  COLUMN 6, COTAP-531
P-B4
P-K3
P*P.  NOTE 24, COTAP-532 TRANSPOSES INTO CARO-KANN
INI
SETUP,VARY
P-K4.  COLUMN 8, COTAP-531
P*P
SAVE,NOTE33
N-QB3
N-KB3
P-B3
P*P
SAVE,NOTE36
N*P
P-KN3
B-QB4
B-N2
N-K5
O-O
B-QN5
N(QN1)-Q2
```

```
O-O
P-B4
N*N
B*N. (E)
P*P
Q-B2
P-QN4
B-B3
INI
SETUP,NOTE36
Q*P.  NOTE36, COTAP-533
Q*P
B-K3
Q-KN5
INI
SETUP,NOTE33
P-KB3.  NOTE33, COTAP-533
P-K4. (E)
P-Q5
B-KB4
INI
SETUP,VARY
N-QB3.  COLUMN 9, COTAP-531
N-KB3
SAVE,NOTE41
P-B3
B-B4
B-N5
P-B3
Q-Q2
N(QN1)-Q2
P-K4
P*P
Q-B4
Q-R4
O-O-O
P-K3
B*N
N*B
P*P
B-N3
B-Q3
B-N5
N(KN1)-K2
P-K4. (E)
P*P
N-Q2
INI
SETUP,NOTE41
P-K4.  NOTE 41, COTAP-533
P*P
B/BIRDS
MSG BIRDS OPENING
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
LET MSMASK=77B.
LET MSCODE=0.
SWITCH EXPERIENCE,OFF
SWITCH LB ON;SWITCH NOR ON
INI
YES,BIRDS OPENING, COTAP-734
P-KB4
P-Q4
N-KB3
N-KB3
P-K3
P-K3.  COLUMN 2, COTAP-734
P-QN3
B-K2
B-N2
N(QN1)-Q2
B-Q3
O-O
O-O
P-QN3
N-B3
B-N2
N-K2
N-B4. (E)
N-N3
N*B
P*N
P-B4
B/SICIL
MSG SICILIAN DEFENSE, CLOSED AND UNUSUAL VARIATIONS.
SWITCH EXPERIENCE,OFF
LET MSCODE=0.
LET MSMASK=77B.
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH LB ON;SWITCH NOR ON
YES,CLOSED SICILIAN, COTAP-441
P-K4
LET MSCODE=60    SELECT 25 PERCENT OF THE TIME.
P-QB4
LET MSCODE 0
N-QB3
N-QB3
P-KN3
P-KN3
B-N2
B-N2
P-Q3
P-N3.  COLUMN 70, COTAP-441.
SAVE,NOTE9
P-B4
B-N2
N-B3
P-K3
O-O
N(KN1)-K2
B-Q2
P-QR3
R-N1
P-Q4
P-K5
P-QN4
N-K2
Q-N3
P-B3
P-N5
Q-K1
P-QR4
P-B4
R-Q1
Q-B2
P*P
P*P
N-B4.  SMYSLOV-KOTOV, TCHIGORIN TOURNEYMENT, MOSCOW 1947
INI
SETUP,NOTE9
N(KN1)-K2.  NOTE 9, COTAP-442.
P-Q3
O-O
B-N2
P-B4
P-B4. (E)
P-KN4. (QE)
P*P(KN5)
P-B5
Q-Q2.  SMYSLOV-BOTVINNIK, 13TH MATCH GAME 1954.
INI
YES,IRREGULAR SICILIAN DEFENSE.
P-K4
P-QB4
N-K2
P-Q3.  COLUMN 197, MCO-226
P-KN3
P-KN3
B-N2
B-N2
O-O
N-QB3
P-QB3
P-K4
P-Q3
N(KN1)-K2
P-QR3
O-O
P-QN4
P-N3
P-KB4
P*P(KB5)
P*P(KB4)
P-Q4
P-K5
B-N5
P-KR3
B*N
Q*B
P-B3.  KERES-FISCHER, STOCKHOLM 1962.
INI
YES
P-K4
P-QB4
P-QB3
P-Q4.  COLUMN 2, COTAP-444.
P*P
Q*P
P-Q4
P*P
P*P
N-QB3
N-KB3
SAVE,RIBBIT
P-K3
SET,RIBBIT
SW LB OFF
B-N5
SW LB ON
B-K2
P-K3
N-B3
Q-Q2
B-K3
B-Q3
O-O
N-B3
Q-Q2
O-O.  PERLIS-RUBINSTEIN, VIENNA 1908.
INI
YES
P-K4
P-QB4
P-KN3.  COLUMN 196, MCO-226
P-Q4. (E)
P*P
Q*P
N-KB3
B-N5
B-N2
Q-K3.CHECK
K-B1
B-R6
SAVE,NOTEB
P-Q4
P*P
N*P
Q-Q2
N-QB3
N-QB3.  PACHMAN-TAIMANOV,  BUENOS AIRES 1960
INI
SETUP,NOTEB
N-B3.  NOTE B, MCO-226.
N-QB3
P-Q3
N-B3
B-K3
P-QN3
N-KN5
B*B.CHECK
K*B
Q-Q2
Q-B3
R-B1.  TRIFUNOVICH-BOLBOCHAN, RIO DE JANEIRO 1950.
INI
YES
P-K4
P-QB4
P-Q4.  COLUMN 4, COTAP-444 (MORRA GAMBIT).
P*P
P-QB3
P*P
N*P
N-QB3
N-B3
P-Q3
B-QB4
P-K3
O-O
N-B3
Q-K2
B-K2
R-Q1
P-K4. (E)
```

```
P-KR3
O-O
B-KN5
B-K3
B*N
B*B(KB3)
N-QN5
B-K2
B*B
P*B
Q-Q3
Q-Q2
R-Q2
R(QR1)-Q1
R(QR1)-Q1
Q-B1
Q-N3
P-QR3
N*P(Q6)
B*N
R*B
N-Q5.  LOKBENC-BOLBOCHAN 1950
INI
YES
P-K4
P-QB4
P-QN4.  NOTE 13, COTAP-445.
P*P
SAVE,NOT13A
P-QR3
P-Q4
P*P(Q5)
Q*P
N-KB3
P-K4
P*P
B*P
N-R3
N-KB3
N-QN5
O-O. (E)
N-B7
Q-B4
N*R
P-K5
INI
SETUP,NOT13A
P-Q4.  NOTE 13 VARIATION, COTAP-445.
P-Q4. (E)
MSG SICILIAN DEFENSE--MAIN VARIATIONS.
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH LB ON;SWITCH NOR ON
YES,SICILIAN DEFENSE-CLASSICAL DRAGON, COTAP-363 ETC.
P-K4
P-QB4
N-KB3
N-QB3
P-Q4
P*P
N*P
N-B3
N-QB3
P-Q3
B-K2
P-KN3
B-K3
B-N2
O-O
O-O
N-N3
B-K3
P-B4
Q-B1.  COLUMN 3, COTAP-364.
SAVE,VARY
P-KR3
R-Q1
SAVE,NOTE9
B-B3
B-B5
SAVE,NOTE11
R-K1
P-QR4
N-Q5
N*N
P*N
N-N5
P-R3
B*N
P*N
Q*P(QB7)
INI
SETUP,NOTE11
R-B2.  NOTE 11, COTAP-365
P-K4. (E)
INI
SETUP,NOTE9
N-Q4.  NOTE 9, COTAP-365
N*N
B*N
B-B5
P-B5. (E)
P-Q4. (E)
P-K5
N-K5
P-B6
P*P
P*P.  GELLER-LIPNITZKY, USSR 1951
INI
SETUP,VARY
K-R1.  NOTE 8, COTAP-365
P-QR4
P-QR4
N-QN5. (E)
INI
SETUP,VARY
Q-K1.  COLUMN 5, COTAP-364
P-QR4
P-QR4
N-QN5
N-Q4
B-B5
P-B5
N-Q2
INI
YES,SICILIAN DEFENSE-DRAGON SUPPLIMENTAL VARIATIONS, COTAP-368
P-K4
P-QB4
N-KB3
N-QB3
P-Q4
P*P
N*P
N-B3
N-QB3
P-Q3
B-K2
P-KN3
SAVE,VARY
B-K3.  COLUMN 1, COTAP-368
B-N2
O-O
O-O
SAVE,NOTE6
P-B4
Q-N3. (E)
Q-Q3
N-KN5. (E)
N-Q5
B*N. (E)
SAVE,NOTE4
B*N. (E)
B*B(K6).CHECK
Q*B
Q*Q.CHECK
N*Q
B*B
N*B
INI
SETUP,NOTE4
N*Q.  NOTE 4, COTAP-369
B*B.CHECK
K-R1
B*N
B*N
B*B
P-B5
B-KR4
R(QR1)-K1
N-K4
Q-KR3
P-B3
Q-R4
B-N5. (E)
INI
SETUP,NOTE6
Q-Q2
N-KN5.  COLUMN 2, COTAP-368
B*N
B*B
SAVE,NOTE8
P-B4
B-Q2
R(QR1)-Q1
R-B1
N-Q5
N*N
B*N
B*B
Q*B
R*P
P-B5
B-B3
N-K3
R-K7. (E)
N-N4
P-KR4
N-R6.CHECK
K-R2
N*P. (E)
Q-N3. (E)  RICHTER-PETROV, HAMBURG 1938
INI
SETUP,NOTE8
N-Q5.  NOTE 8, COTAP-369
B-Q2
R(QR1)-Q1
R-B1
P-KB4.  TRANSPOSES BACK TO COLUMN 2, COTAP-368
INI
SETUP,NOTE6
K-R1.  NOTE 6, COTAP-369, VARIATION 1.
P-Q4. (E)
P*P
N*P
N*N(Q5)
Q*N
B-B3
Q-QR4. (E)
N*N
P*N
B*P(QB6)
R-N1
INI
SETUP,NOTE6
P-KR3.  NOTE 6, COTAP-369, VARIATION 2.
P-Q4. (E)
P*P
N*P
N*N(Q5)
Q*N
B-B3
Q-QR4. (E)
N*N
P*N
B*P(QB6)
R-N1
INI
SETUP,VARY
B-K3.  COLUMN 3, COTAP-368
B-N2
N-N3
O-O
P-B4
P-QR4.  NOTE 12 TAKEN IN LIEU OF COLUMN.
P-QR4
B-K3
SAVE,NOT12A
B-B3
N-QN5
O-O
N-Q2
N-Q4
B-B5
R-B2
```

```
P-K4. (E)  BRONSTEIN-KORCHNOI, SPARTAKIADE 1959
INI
SETUP,NOT12A
N-Q4.  NOTE 12 VARIATION, COTAP-369
Q-N3. (E)
N*B
Q*B
N*R
N-KN5
INI
SETUP,VARY
O-O.  COLUMN 5, COTAP-368
B-N2
N-N3
O-O
SAVE,COL4
P-B3
B-K3
N-Q5
P-QN4
P-QR4
P*P.  EUWE-LANDAU, HOLLAND 1939
INI
SETUP,COL4
K-R1.  COLUMN 4, COTAP-368
P-QR3.  NOTE 19 TAKEN.
P-B4
Q-B2
P-N4
P-K3
P-B5
R-K1
B-KB4
N-K4.  ALEKHINE-FOLTYS,  MUNICH 1942
MSG SICILIAN DEFENSE--RICHTER ROUZER ATTACK.
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH LB ON;SWITCH NOR ON
YES,RICHTER ROUZER VARIATION, SICILIAN DEFENSE, COTAP-373
P-K4
P-QB4
N-KB3
N-QB3
P-Q4
P*P
N*P
N-B3
N-QB3
P-Q3
B-KN5
P-K3
SAVE,VARY
Q-Q2.  COLUMN 6, COTAP-373
B-K2
O-O-O
O-O
SAVE,NOTE2
P-B4
N*N
Q*N
P-KR3
SAVE,NOTE5
B-R4
Q-R4
SAVE,COL7
P-K5
P*P
Q*P(K5)
Q*Q
P*Q
N-Q4
B*B
N*B
SAVE,NOTE7
B-Q3
P-QN3
B-K4
R-N1
R(KR1)-K1
B-N2
R-Q7
B*B
R*B
N-B3
N-N5
R(KB1)-Q1
R-B7
R(QN1)-B1.  GLIGORIC-BENKO, CANDIDATES TOURNEYMENT 1959.
INI
SETUP,NOTE7
B-N5.  NOTE 7, COTAP-373
P-QR3
B-Q3
P-QN4
B-K4
R-N1
P-QR3
P-QR4
P-QN4
P*P
P*P
B-N2
R(KR1)-K1
R(KB1)-B1
K-N2
R-B5.  SCHMID-ELISKASES, MUNICH 1958
INI
SETUP,COL7
B-B4.  COLUMN 7, COTAP-373.
R-Q1.  NOTE 9, COTAP-374.
P-K5
P*P
Q*P(K5)
Q-N5
B-QN3
B-Q2
R-Q4.  PACHMAN-ILLIVITZKY MATCH 1956.
Q-N3. (E)
INI
SETUP,NOTE5
B*N.  NOTE 5, COTAP-374, VARIATION 1.
B*B
Q*P(Q6)
Q-R4
P-K5
R-Q1
INI
SETUP,NOTE5
P-KR4. (EQ)  NOTE 5, COTAP-374, VARIATION 2.
Q-R4
B-B4
R-Q1
INI
SETUP,NOTE5
P-KR4
Q-R4
B-K2.  NOTE 5, COTAP-374, VARIATION 3.
P-K4. (E)
INI
SETUP,NOTE2
N-N3.  NOTE 2, LINE 1, 1ST VARIATION, COTAP-373.
Q-N3. (E)
B*N
B*B
SAVE,NOT2A
Q*P
Q*P
INI
SETUP,NOT2A
N-R4.  NOTE 2, LINE 1, 2ND VARIATION, COTAP-373.
Q-B2
Q*P
B-N4.CHECK
K-N1
R-Q1
INI
SETUP,NOTE2
N(Q4)-N5.  NOTE 2, LINE 2, COTAP-373.
Q-R4
B*N
B*B
N*P(Q6)
R-Q1)
P-B4
P-K4
Q-Q5
Q-B2
P-B5
N-Q5
N(Q6)-N5
Q-R4
Q-B4
B*P. (E)
P*B
R(QR1)-B1
Q-R4
Q*Q
N*Q
N-N6.CHECK
P*N
B-N4.CHECK
R-Q2
B*R.CHECK
K-N1
P-QR3
N(QN5)-B3
B*N
N*B
R*N
P*R
R-Q8.CHECK
K-N2
P-K5
INI
SETUP,VARY
N*N.  SUP VARIATIONS, RICHTER-ROUZER, COTAP-378, COLUMN 6.
P*N
P-K5
Q-R4
B-N5
P*B
P*N
P-N5
SAVE,NOTE2
N-K4
Q-K4
P-KB3
P-Q4
Q-Q2
P-KR3. (E)
B-R4
P-N4. (E)
SAVE,NOTE3A
B-N3
Q*P(N7)
R-Q1
B-R3
INI
SETUP,NOTE3A
B-B2.  NOTE 3 VARIATION, COTAP-379
P*N
O-O-O
Q-B2
INI
SETUP,NOTE2
Q-B3.  NOTE 2, COTAP-378
Q-K4.CHECK
INI
SETUP,VARY
B-N5.  COLUMN 7, COTAP-378
B-Q2
SAVE,NOTE4
O-O
P-KR3
SAVE,NOTE5
B-KR4
P-R3
B-K2
B-K2
N-N3
Q-B2
P-B4
P-KN4. (E)
SAVE,NOTE6
B-N3
P*P
R*P
N-K4
INI
SETUP,NOTE6
P*P. (Q)  NOTE 6 VARIATION, COTAP-379
P*P
B*P(KN5)
P-Q4
```

```
P-KR3
R*P
INI
SETUP,NOTE5
B-K3.  NOTE 5, COTAP-379
P-R3
B-K2
Q-B2
N-N3
B-K2
INI
SETUP,NOTE4
B(QN5)*N.  NOTE 4, COTAP-379
P*B
Q-B3
P-K4
INI
SETUP,VARY
N-N3.  COLUMN 8, COTAP-378
B-K2
Q-Q2
P-KR3
B-K3
O-O
INI
SETUP,VARY
B-K2.  COLUMN 9, COTAP-378
B-K2
O-O
O-O
N(Q4)-N5
P-QR3
B*N
P*B
N-Q4
K-R1
K-R1
R-KN1
P-B4
B-Q2
INI
SETUP,VARY
Q-Q3.  COLUMN 10, COTAP-378.
B-K2
SAVE,NOTE12
R-Q1
O-O
B-K2
P-Q4
P*P
N-QN5
Q-N3
N(QN5)*P(Q4)
N*N
N*N
B*B
Q*B.  KERES-ARONIN, USSR CHAMPIONSHIP 1951
INI
SETUP,NOTE12
B-K2.  NOTE 12, COTAP-379
O-O
O-O
P-KR3. (E)
SAVE,NOT12A
B-B1.  NOTE 12 MAIN VARIATION.
N*N
Q*N
B-Q2
P-K5
P*P
Q*P(K5)
B-B3
R-Q1
Q-N1.  KERES-BRONSTEIN, BUDAPEST 1950
INI
SETUP,NOT12A
B-R4.  NOTE 12 VARIATION, COTAP-379
N*N
Q*N
N*P. (E)  OH WOH.
MSG SICILIAN DEFENSE, SOZIN VARIATION, COTAP-381
INI
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH LB ON;SWITCH NOR ON
YES,SOZIN VARIATION, SICILIAN DEFENSE, COTAP-381
P-K4
P-QB4
N-KB3
N-QB3
P-Q4
P*P
N*P
N-B3
N-QB3
P-Q3
B-QB4
P-K3
O-O
P-QR3
B-N3
Q-B2
B-K3
N-QR4
P-B4
B-K2
Q-B3
P-QN4
P-K5
B-N2
Q-N3
P*P
P*P
N-R4
Q-R3
N*B
N*N
Q*P
N-R5
P-N5. (E)
N-B4
Q-B2
Q*N
P-KN3
Q-K5
Q*Q
N*Q.  SMAILLOGOVIC-DANTAR, YUGOSLAVIA 1957
B/ENGLI
MSG  ENGLISH DEFENCE -- SICILIAN VARIATION
SWITCH,NOREPLY,ON
SW LW OFF
SW LB ON
SW EXP OFF
LET MSCODE=0
LET MSMASK=77B
INI
P-QB4
N-KB3
SAVE EB2P738
N-QB3
P-Q4
P*P
N*P
SAVE EB33740
P-KN3
N*N
P/N2*N
P-KN3
B-KN2
B-N2
SAVE EB63741
R-QN1
N-Q2
P-QB4
O-O
N-B3
R-N1
SETUP EB63741
P-KR4
P-KR3
N-R3
O-O
Q-N3
N-Q2
N-B4
P-K3
SETUP EB63741
Q-N3
N-B3
SAVE EB74741
N-B3
O-O
O-O
N-R4  .NEXT MOVES FROM P486 MCO
Q-B2
P-QB4
P-Q3
B-B4
P-K4
B-Q2
SETUP EB74741
B*N
P*B
SETUP EB33740
P-K4
N-N5
B-B4
B-K3
B*B
P*B
N/N1-K2
N-Q6 .NEXT MOVES FROM P486 MCO
K-B1
N-B3
Q-N3
Q-Q2
SETUP EB33740
N-B3
P/QB2-B4
P-K4
N-N5
B-B4
N-Q6
K-K2
N-B5
K-B1
N-K3
B/PETRF
MSG PETROV DEFENCE AS BLACK
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
LET MSMASK=77B.
LET MSCODE=0.
SWITCH EXPERIENCE,OFF
SWITCH LB ON;SWITCH NOR ON
INI
YES.
P-K4.
LET MSCODE=20    SELECT 25 PERCENT OF THE TIME.
P-K4.
LET MSCODE 0
N-KB3.
N-KB3. PETROV DEFENCE AS BLACK
SAVE A
NXP.
P-Q3.
SAVE Q
N-B4
NXP
SAVE Q1
N-B3
NXN
P/NXN
P-KN3
SAVE Q3
P-Q4
B-N2
SAVE Q2
P-KR4
O-O
B-N5
Q-K1. CHECK
N-K3
P-QB4
P-R5
N-B3
P/5XP
P/KBXP
B-QB4. CHECK
B-K3
P-Q5
B-B2. FUDERER MATAVOVIC SARAJEVO 1951
SETUP Q2
B-Q3
O-O
O-O
```

```
N-B3
R-QN1
R-QN1
P-B4
P-KB4
Q-B3
N-K2. MILIC GERMEK 1951
SETUP Q3
B-K2
B-N2
O-O
O-O
P-Q4
N-Q2
N-K3
N-N3
P-QB4
B-K3
P-QB3
P-KB4. MATANOVIC ALEXANDER 1951
SETUP Q1
P-Q3
N-KB3
P-Q4
B-K2
B-Q3
O-O
O-O
N-B3
P-QB3
R-K1
B-N5
P-Q4
N-K3
N-K5
BXB
NXB
SETUP Q1
P-Q4
P-Q4
N-K3
B-K3
B-Q3
P-KB4
O-O
B-Q3. PAULSEN SCHALLOP FRANKFURT 1887
SETUP Q1
Q-K2
Q-K2
N-K3
P-QB3
P-Q3
N-B3
N-Q2
P-Q4
N-B3
P-KN3
B-Q2
B-N2
O-O-O
O-O
R-K1
Q-Q3. PILNIK BOGOLJUBOV ZURICH 1951
SETUP Q
N-KB3.
NXP.
SAVE B
P-Q4.
P-Q4.
B-Q3.
N-QB3.
SAVE C
P-B4
B-QN5. CHECK
SETUP C
O-O.
B-K2.
SAVE C
P-B4.
B-KN5.
SAVE D
PXP.
QXP.
N-B3.
NXN.
PXN.
O-O.
B-KB4.
B-Q3. EQUALITY.
SETUP D
N-B3
N-B3.
PXP.
NXP(Q4).
B-K4.
B-K3.
Q-Q3.
N(B3)-N5.
Q-K2.
P-QB3.
P-QR3.
N-R3. =
SETUP C
R-K1.
B-KN5.
SAVE D
BXN.
PXB.
RXP.
BXN.
SAVE E
QXB.
NXP.
Q-Q3.
N-K3.
SETUP E
PXB.
P-B4.
R-B4.
O-O.
P-Q5.
B-N4.
R-QR4
BXB
QXB
QXP
SETUP D
P-B4.
N-B3.
PXP.
NXP(Q4).
N-B3.
O-O.
B-K4.
B-K3.
Q-Q3.
P-KR3.
P-QR3.
B-B3.
B-Q2.
P-R3. MEYER-LIEGL CORR 1952
SETUP B
P-B4
B-K2
SAVE R1
N-B3
B-B4
SAVE R2
NXN
BXN
P-Q4
P-Q4
Q-N3
N-B3
SETUP R2
N-Q5
O-O
P-Q3
N-KB3
NXB. CHECK
QXN. CHECK
B-K2
R-K1
SETUP R1
P-Q4
P-Q4
B-Q3
B-QN5. CHECK
SETUP B
N-B3
NXN
P/QXN
B-K2
B-Q3
B-N5
Q-K2
N-B3
B-KB4
Q-Q2
P-KR3
B-R4
P-KN4
B-N3
O-O-O
BXB
RXB
O-O-O. ROMANOVSKY TRAVIN CCCP 1923
SETUP B
P-Q3.
N-KB3.
P-Q4.
B-N5.
B-K2.
B-K2.
O-O.
O-O.
P-B4.
P-Q4.
P-B5.
N-B3.
P-KR3.
B-R4.
B-K3.
N-K5. PIETZSCH-MIKENAS
SETUP B
Q-K2.
Q-K2.
P-Q3.
N-KB3.
B-N5.
N(N1)-Q2.
SAVE C
QXQ.CHECK.
BXQ.
N-B3.
P-B3.
O-O-O.
O-O. =
SETUP C
N-B3.
QXQ.CHECK.
BXQ.
P-KR3.
SAVE D
BXN.
NXB.
N-QN5.
K-Q1.
SETUP D
B-B4.
P-KN3.
SAVE E
O-O-O.
B-N2.
P-KR3.
N-N3.
N-Q2.
N(B3)-Q4.
NXN.
NXN.
B-R2.
O-O. TRIFUNOVIC-BRONSTEIN LENINGRAD 1957
SETUP E
N-Q4.
N-N3.
SETUP D
B-R4.
P-KN3.
O-O-O.
B-N2.
SAVE E
N-QN5.
K-Q1.
R(R1)-K1.
```

```
R-K1.
N(B3)-Q4.
N-B1. KHOLMOV-BRONSTEIN BAKU 1961
SETUP E
R(R1)-K1.
N-N3.
B-B1.CHECK.
B-K3.
N-Q4.
K-Q2.
SETUP D
B-Q2.
P-KN3.
O-O-O.
B-N2.
SETUP A
P-Q4.
NXP.
SAVE S
NXP
P-Q3
SETUP S
PXP
B-B4
SAVE S1
Q-Q5
BXP. CHECK
SAVE S2
K-K2
P-KB4
PXP. EP
NXP
SETUP S2
K-Q1
P-KB4
B-QB4
R-B1
N/1-Q2
P-B3
Q-Q3
P-Q4
SETUP S1
B-QB4
NXP
BXP. CHECK
K-B1
Q-Q5
NXR
SAVE S3
QXB. CHECK
KXB
Q-Q5. CHECK
K-B1
SETUP S3
B-R5
Q-K2
B-N5
B-B7. CHECK
K-K2
Q-K3
N-B3
SETUP S1
B-K3
BXB
PXB
P-Q4
SETUP S
B-Q3.
P-Q4.
SAVE S10
PXP
N-QB3
O-O
B-KN5
B-KB4
N-Q5
P-B3
N-K3
B-K3
N/3-B4
SETUP S10
NXP.
B-Q3.
SAVE S11
P-QB4
BXN
PXB
N-QB3
SETUP S11
N-QB3
NXN
PXN
N-Q2
SETUP S11
Q-K2
BXN
PXB
N-B4
N-B3
O-O. MIESES GROB 1934
SETUP S11
O-O.
O-O.
SAVE B
P-QB4.
BXN.
PXB.
N-QB3.
SAVE C
P-B4.
N-N5.
PXP.
QXP.
BXN.
QXB.
N-B3.
Q-N3.
B-K3.
B-K3.
Q-B3.
N-B7.
R(R1)-Q1.
NXB. DAMJANOVIC-TRIFUNOVIC VRNACKA BANJA 1963
SETUP C
B-B4.
N-N5.
SETUP C
R-K1.
NXP(K4).
PXP.
NXB.
QXN.
N-Q3.
N-B3.
B-B4.
SETUP C
PXP.
QXP.
SAVE D
Q-B3.
P-B4.
PXP.EN PASSANT.
NXP(B3).
QXQ.CHECK.
NXQ.
B-QB4.
B-K3.
R-Q1.
N(B3)-K2.
N-B3.
P-B3.
B-KN5.
B-B2.
N-K4. GUFELD-MIKENAS USSR XXXIII
SETUP D
Q-B2.
N-N5.
BXN.
NXQ.
BXQ.
B-B4.
P-KN4.
B-N3.
P-B4.
B-Q6.
R-Q1.
B-R3.
N-B3.
NXR.
P-N4.
P-QB3.
B-K4.
R(R1)-Q1. BROWNE-ACERS
SETUP B
P-KB3.
N-B4.
SETUP B
N-Q2.
NXN.
BXN.
N-Q2.
SETUP B
R-K1.
P-KB4.
P-QB4.
B-K3.
SETUP B
N-QB3.
NXN.
PXN.
N-Q2.
SETUP A
P-Q3
P-Q3
P-KN3
P-KN3
B-N2
B-N2
O-O
O-O
R-K1
P-KR3
P-KR3
N-B3
P-B3
N-R2
N/1-Q2
P-B4
P-QN4
P-QR3
B-N2
P-B5. CSOM CHERNIKOV LIPECK 1968
SETUP A
B-B4
NXP
SAVE P1
P-Q3
N-KB3
NXP
P-Q4. KRRES
SETUP P1
N-B3
NXN
P/QXN
P-KB3
SAVE P2
O-O
P-Q3
N-R4
P-KN3
P-B4
Q-K2
P-B5
Q-N2
Q-B3
B-K2
SETUP P2
N-R4
P-KN3
P-KB4
P-QB3
P-B5
P-Q4
PXP
PXB
Q-R5
K-Q2
SETUP P1
Q-K2
P-Q4
B-N3
N-QB3. SOZIN
SETUP P1
NXP
```

```
Q-K2
P-Q4
P-Q3
BXP. CH
K-Q1
O-O
PXN
PXP. DIS CH
B-Q2
B-Q5
N-B4
P-QN4
P-B3
PXN
PXB. SOZIN
SETUP A
N-B3
B-N5
SAVE P3
B-B4
O-O
O-O
N-B3
P-Q3
BXN
PXB
P-Q4
PXP
NXP
R-K1
B-N5
B-Q2
Q-Q3
R-N1
N-N3. BERNSTEIN ALEKHINE 1933
SETUP P3
NXP
O-O
SAVE P4
B-K2
R-K1
N-Q3
BXN
P/QXB
NXP
O-O
P-Q3. TARRASCH GRUNFELD VIENNA 1922
SETUP P4
P-Q3
P-Q4
P-QR3
BXN. CHECK
PXB
R-K1
P-KB4
PXP
P-Q4
N-Q4
P-B4
N-K2
B-K2
N-B4. LUPI ALEKHINE LISBON 1946
B/KGA
MSG KINGS GAMBIT ACCEPTED ETC AS BLACK
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
LET MSMASK=77B.
LET MSCODE=0.
SWITCH EXPERIENCE,OFF
SWITCH LB ON;SWITCH NOR ON
INI
YES.
P-K4.
P-K4.
SAVE A1
P-KB4.
PXP.
SAVE A
B-B4.
N-KB3.
N-QB3.
P-B3.
P-Q4.
B-N5.
P-K5.
N-K5.
Q-B3.
P-Q4.
PXP.EN PASSANT
O-O.
N/1-K2.
Q-R5.CHECK
P-KN3.
PXP.
PXP.
Q-N5. KERES
SETUP A
N-QB3.
N-QB3.
SETUP A
N-KB3.
P-KN4.
SAVE B
P-KR4.
P-N5.
SAVE C
N-K5.KIESERITSKY
N-KB3.
SAVE D
P-Q4.
P-Q3.
N-Q3.
NXP.
BXP.
Q-K2.
Q-K2.
B-N2.
P-B3.
P-KR4.
N-Q2.
NXN.
KXN.
QXQ.CHECK   EQUALITY
SETUP D
B-B4.
P-Q4.
PXP.
B-Q3.
SAVE E
P-Q4.
N-R4.
O-O.
QXP.
Q-K1.
QXQ.   R BYRNE-KERES
SETUP E
O-O.
BXN.
R-K1.
Q-K2.
P-B3.
N-R4.
P-Q4.
N-Q2.
PXB.
NXP.
P-QN3.
O-O.
B-R3.
N-B6.CHECK
PXN.
QXP.
R-K5.
Q-N6.CHECK ANAL CAPABLANCA,BURN,ED LASKER
SETUP C
N-N5. ALLGAIER
P-KR3.
NXP.
KXN.
B-B4.CHECK.
P-Q4.
BXP.CHECK
K-K1.
P-Q4.
N-KB3.
N-B3.
P-B6.
PXP.
B-N5.
B-N3.
N-B3.
B-K3.
PXP.
SETUP B
B-B4.
P-N5.
O-O. MUZIO-POLERIO
PXN.
QXP.
Q-B3.
P-K5.
QXP.
P-Q3.
B-R3.
N-B3.
N-K2.
B-Q2.
N/1-B3.
R/R-K1
Q-KB4.
N-Q5.
K-Q1.
B-B3.
R-K1.
B-B6.
B-N4.
P-KN4.
Q-N3.
SETUP A1
N-QB3.
N-QB3.
SAVE A1A
B-B4.
B-B4.
Q-N4.
P-KN3.
Q-B3
N-B3.
N/1-K2.
P-Q3.
P-Q3.
B-KN5
Q-N3.
P-KR3.
P-B4.
Q-K2.
N-Q5.
NXN.
QXB.
N-K6. LARSEN-PORTISCH SANTA MONICA 1966
SETUP A1A
P-B4.
PXP.
SAVE A1B
N-B3.
P-KN4.
SETUP A1B
P-Q4.
Q-R5.CHECK
K-K2.
P-Q4.
PXP.
B-KN5.CHECK
N-B3.
O-O-O.
PXN.
B-QB4
SAVE X2
PXP.CHECK
K-N1.
N-QN5.
N-B3.
P-B3.
R/R-K1.CHECK
K-Q3.
B-B4.CHECK
K-B4.
B-K3.CHECK
KXB.
P-QR4.
NXP.
Q-R4.CHECK
N-K5.
N-Q2.CHECK
K-N5.
```

```
QXQ.
SAVE X
N-B6.CHECK
KXP.
SETUP X
NXN.CHECK
KXP.
N-B5.CHECK
K-R2.
K-B6.
Q-N5.
SETUP X
BXP.
QXR.
K-R6.
NXN.
SAVE X1
BXN.
B-B5.CHECK
BXB.
RXB.
SETUP X1
NXR.
P-B3.
PXN.
P-B4.
B-K3.
RXN.
B-QN5.
QXR.
B-R7.CHECK
K-B2.
B-B5.
B-B1.
SETUP X2
Q-K1.
Q-R4.
PXP.CHECK
K-N1.
SAVE X3
K-Q1.
BXP.
B-Q2.
B/QXN.
PXB.
BXN.CHECK
PXB.
QXP/6.CHECK
B-K2.
QXP/6.  STEINITZ-CITY OF LIVERPOOL 1899
SETUP X3
BXP.
R-K1.CHECK
K-Q2.
RXQ.
RXR
BXN.
P/4XB.
B-B3.
P-QN4.
Q-N5.
SETUP X3
K-Q2.
BXN.
P/2XB
BXP.
B/BISHP
MSG BISHOPS OPENING AS BLACK
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
LET MSMASK=77B
LET MSCODE=0.
SWITCH EXPERIENCE OFF
SWITCH LB ON
SWITCH NOR ON
INI
YES.
P-K4
P-K4
B-B4
N-KB3
SAVE N
P-Q4
PXP
SAVE M1
N-KB3
NXP
QXP
N-KB3
SAVE M
B-KN5
B-K2
N-B3
P-B3
O-O-O
P-Q4
R/R-K1
B-K3
B-Q3
N/1-Q2
Q-KR4
N-B4
SAVE N1
N-Q4
N-N1
SAVE N2
BXB
QXB
Q-N3
P-KN3. ESTRIN BIHOVSKY CCCP 1964
SETUP N2
P-B4
BXB
PXB
N-K2. TORRE THOLFSEN USA 1924
SETUP N1
B-B5
BXB
BXN
N-K3
BXB
QXB
NXP
PXN
Q-R4. CHECK
K-B1.
RXP
B-N3. ESTRIN KHATCHATUROV CCCP 1943
SETUP M
N-B3
P-B3
SETUP M1
QXP
N-B3
Q-K3
B-N5.CHECK
SAVE M11
B-Q2
O-O
N-K2
P-Q4
SETUP M11
P-QB3
B-R4
N-B3
B-N3
Q-B4
O-O
SETUP M1
P-K5
P-Q4
B-N3
N-K5
N-K2
N-B4
NXP
NXB
NXN
P-QB4
O-O
N-B3
SETUP N
P-Q3
P-B3
SAVE N3
Q-K2
B-K2
SAVE N4
N-KB3
O-O
B-N3
P-Q4
SAVE N5
O-O
B-KN5
P-KR3
B-R4
N/1-Q2
N/1-Q2
P-B3
R-K1. UBILAVA GULKO CCCP 1969
SETUP N5
N/1-Q2
N/1-Q2
P-B3
Q-B2. WADE TESCHNER HASTINGS 1954
SETUP N4
P-B4
P-Q4
P/KXP
PXP/5
BXP
O-O
SAVE N41
N-Q2
PXP
B-QN3
P-QR4. GONSIOROVSKI ALEKHINE CCCP 1918
SETUP N41
PXP
NXP
N-QB3
N-Q5
Q-Q2
B-QN5
SETUP N3
P-B4
PXP
BXP/4
P-Q4
PXP
NXP
Q-B3
B-K3
N-K2
NXB
QXN
BXB
QXB
B-Q3
B/CENTR
MSG DANISH GAMBIT
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
LET MSMASK=77B.
LET MSCODE=0.
SWITCH EXPERIENCE,OFF
SWITCH LB ON;SWITCH NOR ON
INI
YES.
P-K4
P-K4
P-Q4
PXP
SAVE D6
P-QB3
PXP
SAVE D5
B-QB4
PXP
BXP/2
P-Q4
SAVE D3
BXP/5
B-QN5. CHECK
SAVE D2
N-QB3
BXN. CHECK
BXB
N-KB3
SAVE D1
Q-B3
NXB
SETUP D1
N-B3
NXB
```

```
PXN
Q-K2. CHECK
K-B1
O-O. RADEVIC ASATURYAN CCCP 1968
SETUP D2
K-B1
N-KB3
Q-R4. CHECK
N-B3
B/5XN. CHECK
PXB
QXB
R-QN1
SETUP D3
PXP
N-KB3
N-QB3
B-Q3
SAVE D4
Q-B2
Q-K2
N/1-K2
O-O
O-O-O
B-QR6. TARTAKOVER
SETUP D4
N-KB3
O-O
O-O
B-KN5
Q-Q4
N/1-Q2
R/R-K1
R-K1. OPOCENSKY RETI BADEN 1914
SETUP D5
NXP
N-QB3
N-B3
B-QN5
SAVE D51
B-KN5
N/1-K2
Q-B2
P-Q3
O-O-O
BXN. CHECK
QXB
O-O. GUFELD STEIN CCCP 1959
SETUP D51
B-QB4
P-Q3
Q-N3
BXN. CHECK
PXB
Q-Q2
SAVE D52
Q-B2
N-KB3. CIOCALTEA KOVACS BAJA 1971
SETUP D52
O-O
N-QR4
Q-N4
NXB
SETUP D6
N-KB3
B-B4
NXP
N-KB3
N-QB3
P-Q4
PXP
O-O
SAVE D61
B-K2
NXP. KERES
SETUP D61
B-KN5
Q-Q3
BXN
QXB
N-KB3
B-KN5
B-K2
BXN
BXB
R-K1. CHECK
K-B1
N-Q2. BLUMENFELD ALEKHINE CCCP 1920
SETUP D61
B-QB4
R-K1
B-K3
P-QB3
SETUP D6
P-KB4
B-B4
N-KB3
N-QB3
B-Q3
N-B3
SETUP D6
QXP
N-QB3
SAVE E
Q-R4
N-B3
N-QB3
B-QN5
SETUP E
Q-Q1
N-B3
B-Q3
P-Q4
SETUP E
Q-B4
N-B3
N-QB3
P-Q4
NXP
NXN
PXN
N-QN5. MIESES LEONHARDT BERLIN  920
SETUP E
Q-K3
N-B3
SAVE E1
P-K5
N-KN5
SAVE E12
Q-K2
P-Q3
SAVE E10
P-KB3
N-R3
SAVE E11
BXN
Q-R5. CHECK
SETUP E11
PXP. DIS CH
B-K3
PXP
QXP
SETUP E10
P-KR3
N/5XP/4
P-KB4
Q-R5. CHECK
K-Q1
N-Q5
Q-K4
Q-B7
SETUP E10
PXP. DIS CH
B-K3
SETUP E12
Q-K4
P-Q4
PXP. DIS CH
B-K3
SAVE E121
B-QR6
QXP
BXP
Q-N5
QXQ
NXQ
N-QR3
R-QN1. MIESES BURN BRESLAU 1921
SETUP E121
PXP
Q-Q8. CHECK
SETUP E1
B-Q2
N-KN5
SETUP E1
B-K2
Q-K2
N-QB3
P-Q4
PXP
N-QN5
B-Q3
N/3XP. MASON SCHLECHTER PARIS 1900
SETUP E1
N-QB3
B-N5
SAVE F2
B-Q2
O-O
O-O-O
R-K1
SAVE F3
B-QB4
N-QR4
SAVE F1
B-Q3
P-Q4
SAVE F
Q-N3
PXP
NXP
NXN
BXN
BXB
RXB
Q-K2
SETUP F
P-K5
P-Q5
Q-N3
N-KR4
Q-B3
PXN
SETUP F1
B-K2
P-Q4
NXP
NXN
Q-Q3
Q-B3
PXN
B-KB4
Q-KB3
BXB
RXB
N-B5
BXN
R-K8. CHECK
R-Q1
Q-N4. CHECK
K-N1
RXR. CHECK JAKOBSEN URZICA GRONINGEN 1970
SETUP F2
B-QB4
O-O
B-Q2
BXN
BXB
NXP. KUPREICHIK LEIN CCCP 1969
SETUP F3
Q-N3
NXP
NXN
RXN
B-KB4
Q-B3
SAVE F31
N-KR3
P-Q3
B-Q3
N-Q5
SETUP F31
BXP
P-Q3
```

```
BXP
Q-R3. CHECK
SETUP F3
P-KB3
P-Q4
Q-B2
PXP
NXP
NXN
PXN
Q-K2
B/BENON
LET MSCODE=0;LET MSMASK=77
SW LB ON;SW LW OFF
SW EXP OFF;SW NOR ON
MSG MODERN BENONI
INI
YES
P-Q4
N-KB3
SAVE Q
P-QB4
P-B4
SAVE Q1
P-Q5
P-K3
SAVE Q2
N-QB3
PXP
SAVE Q3
PXP
P-Q3
SAVE A1
MSG VARIATIONS WITH BOTH P-K4 AND P-KB4
P-K4
P-KN3
SAVE A2
P-B4
B-N2
SAVE A3
B-N5. CHECK
N/3-Q2
SAVE A4
B-Q3
O-O
SAVE A5
N-B3
N-R3
O-O
N-B2
SAVE A51
N-Q2
R-N1
P-QR4
P-QR3
N-B4
N-K1
Q-B3
Q-B2
P-R5
P-QN4. SAIDY EVANS USA 1964
SETUP A51
K-R1
R-N1
P-QR4
P-QR3
P-B5
P-QN4
P/RXP
P/RXP. SPASSKY SAVON MOSCOW 1971
SETUP A5
N/1-K2
N-R3
O-O
N-B2
P-QR4
P-QR3
R-R3
R-N1
R-N3
P-QN4
PXP
P-B5
BXP
N-B4
R-R3
PXP
P-QN4
NXP/5
B-Q3
NXN
NXN
B-N2
P-B5
NXP
NXN
BXN
PXP
Q-N3.CHECK
K-R1
P/BXP. BERLINER EVANS USA 1963
SETUP A4
N-B3
O-O
O-O
P-QR3
B-Q3
P-QN4. CHEREPKOV SUETIN CCCP 1961
SETUP A4
P-QR4
Q-R5. CHECK
P-KN3
Q-K2
N-B3
O-O
O-O
N-R3
R-K1
N-N5
B-B1
P-N3. LUTIKOV VASYUKOV CCCP 1959
SETUP A3
P-K5
N/3-Q2
SAVE A31
PXP
O-O
N-B3
N-KB3
N-K5
N/1-Q2
B-K2
NXN
PXN
N-Q2
P-K6
PXP
PXP
Q-R5. CHECK
P-KN3
BXN.CHECK
PXB
Q-K5
R-KN1
QXP. MIKENAS SHCHERBAKOV CCCP 1961
SETUP A31
N-N5
PXP
N-Q6. CHECK
K-K2
NXB. CHECK
QXN
SAVE A31A
B-B4
R-K1
N-B3
N-N3
SETUP A31A
P-Q6. CHECK
K-B1
N-B3
P-K5
N-N5
P-KR3
NXP/4
Q-K1
Q-K2
N-QB3. PARTOS HOLM SKOPJE 1972
SETUP A31A
N-B3
R-K1
B-B4
N-N3
P-Q6. CHECK
K-B1
B-N5
N-B3
O-O
K-N1
PXP
BXP. ANALYSIS BY SUETIN
SETUP A3
B-K2
O-O
SETUP A3
N-B3
O-O
SAVE A32
B-K2
R-K1
SAVE A33
P-K5
N/3-Q2
PXP
P-QR3
P-QR4
N-KB3
O-O
B-N5
N-K5
BXB
QXB
QXP. LEHMAN TORAN MUNICH 1954
SETUP A33
Q-B2
NXP/5
NXN
B-B4
N/3-Q2
Q-K2
SETUP A33
N-Q2
P-B5
SAVE A331
P-QR4
N/1-Q2
O-O
N-B4
B-B3
B-R3
Q-B2
N-Q6
NXP
NXB
QXN
B-Q2
P-QN3
Q-B2
Q-K3
P-QR3. PADEVSKI CIOCALTEA HAVANA 1966
SETUP A331
BXP
NXP/5
SETUP A331
B-B3
N/1-Q2
SAVE A332
O-O
P-QN4
K-R1
P-QR3
P-QR4
R-N1
PXP
PXP
P-K5
PXP
N/2-K4
NXN
NXN
N-B3
P-Q6
B-K3
N-B5
```

P-K5. POMAR FISCHER HAVANA 1966
SETUP A3
B-Q3
O-O
SETUP A32
B-Q3
Q-N3
SAVE A34
Q-N3
QXQ
PXQ
N-R3. EUWE
SETUP A34
K-B1
N-R3
P-QR3
B-Q2
Q-B2
R/R-B1
B-K3
N-KN5
B-N1
P-B4. HORSEMAN LARSEN HASTINGS 1957
MSG VARIATIONS WITH P-K4 BUT NOT P-KB4
SETUP A2
B-Q3
B-N2
N/1-K2
O-O
O-O
P-QR3
P-QR4
Q-B2
P-R3
N/1-Q2
P-B4
R-K1
N-N3
P-B5
B-B2
N-B4
Q-B3
N/3-Q2
B-K3
P-QN4
PXP
R-N1. OJANEN KERES HELSINKI 1960 / PENROSE TAL LEIPZIG 1960
SETUP A2
N-B3
B-N2
SAVE A21
B-K2
O-O
SAVE A21A
O-O
R-K1
SAVE B
N-Q2
N-R3
PR
MSG CURRENTLY FASHIONABLE CRITICAL POSITION IN BENONI
SAVE B1
P-B3
N-B2
SAVE B2
P-QR4
P-N3
SAVE B4
N-B4
B-QR3
SAVE B3
B-N5
P-R3
SAVE B31
B-K3
BXN
BXB
P-R3
Q-Q2
K-R2
R/R-N1
R-QN1
P-QN4
P-QN4
P/RXP
P/RXP
B-K2
P-B5
R-R1
R-QR1. RESHEVSKY MATULOVIC PALMA 1971
SETUP B31
B-Q2
BXN
BXB
P-R3
K-R1
N-Q2
R-QN1
R-N1
Q-K2
Q-B1
P-QN4
P-QN4. GLIGORIC MATULOVIC BELGRADE 1970
SETUP B3
N-R3
BXB
NXB
N-Q2
K-R1
N-K4
N-B3
P-B4
N/R-N5
P-QR3. DONNER JANOSEVIC AMSTERDAM 1971
SETUP B3
R-N1
BXN
BXB
N-Q2
N-N5
N-K4
NXN
QXN
B-QN5
N-Q2. GLIGORIC LOBIGAS MANILA 1968
SETUP B3
N-K3
BXB
QXB
P-QR3
SETUP B3
B-K3
BXN
BXB
P-QR3
R-K1
N-Q2
B-KB1
R-N1
Q-B2
P-QN4. PARTOS MATULOVIC BUCHAREST 1966
SETUP B3
B-B4
N-R4
SAVE B32
B-K3
P-B4
N-Q2
P-KB5
B-B2
B-QB1. KCHOUK FORINTOS HAVANA 1967
SETUP B32
BXP
BXN/5
BXN
BXB
SETUP B4
P-R5
P-QN4
NXP
N/3XP/4
PXN
B-QR3
N-B3
BXN
BXB
B-Q5. CHECK
K-R1
NXB
N-B4
N-N5
B-B4
NXP
BXP
N-K6. DESPOTOVIC JANOSEVIC YUGOSLAVIA 1970
SETUP B4
K-R1
N-Q2
N-B4
N-K4
N-K3
P-B4
P-B4
N-B2
PXP
PXP
B-Q3
Q-B3. NAJDORF FISCHER HAVANA 1967
SETUP B2
K-R1
N-Q2
N-B4
N-K4
N-K3
P-B4
P-B4
N-B2
PXP
PXP. O,KELLY DAMJANOVIC BAD PYRMONT 1970
SETUP B2
N-B4
N-Q2
P-QR4
P-B4
PXP
PXP
K-R1
N-K4
B-B4
NXN
BXN
Q-B3. BUKIC NICEVSKI YUGOSLAVIA 1970
SETUP B1
P-B4
N-B2
SAVE B11
B-B3
R-N1
SAVE B113
N-B4
P-QN4
SAVE B112
N-R5
B-Q2
P-K5
PXP
PXP
RXP
B-B4
R-B4
B-N3
P-N5
SAVE B111
N-B6
BXN
PXB
PXN. SOOS MATULOVIC SKOPJE 1968
SETUP B111
N-R4
N/3XP
NXP
B-N4
SETUP B112
NXP/6
QXN
P-K5
Q-Q1
PXN
QXP/3
SETUP B113
P-QR4
N-Q2. STEINMEYER EVANS USA 1964
SETUP B11
P-QR4

P-N3
R-K1
B-QR3
B-B3
R-N1
P-K5
PXP
PXP
N/3XP
NXN
NXN
N-K4
RXP
B-N5
RXB
NXR
B-Q5. CHECK KARAKLAIC BOSCOVIC YUGOSLAVIA 1970
SETUP B1
K-R1
N-B2
P-QR4
P-N3
SETUP B1
R-K1
N-B2
P-QR4
P-N3
Q-B2
N-N5
SAVE B51
BXN
BXB
N-B4
Q-B3
B-K3
B-Q2
Q-Q2
B-KB1. HORT PRYBIL LUHACOVICE 1971
SETUP B51
P-R3
NXP/7
KXN
Q-R5. CHECK
SAVE B511
K-B1
B-Q5
N-Q1
QXP/6
B-B3
Q-R7
SAVE B512
N-K3
P-B4
N/2-B4
PXP
BXP
B-R3
B-B3
R-K4
R-R3
R/1-K1
SAVE B513
B-Q2
NXP
BXN.CHECK
RXB
SAVE B514
K-K2
BXN/6
RXB
BXN. CHECK
K-Q1
RXR
RXR
QXP. GURGENIDZE TAL CCCP 1957
SETUP B514
R-K2
Q-R8. CHECK
K-B2
BXN/5
QXB
R-KB1. CHECK
K-N3
B-K4. CHECK
SETUP B514
R-Q3
R-B4. CHECK
K-K2
Q-R4. CHECK
SETUP B513
R-Q3
NXP
RXB
PXR
BXN. CHECK
RXB
SAVE B5131
NXR
Q-R8. CHECK
K-B2
QXR. CHECK
K-B3
R-KB1. CHECK
B-B4
Q-KB8. CHECK
K-N3
BXN
SETUP B5131
Q-Q3
Q-R8. CHECK
K-K2
QXP. CHECK
SETUP B512
N-B2
P-B4
SETUP B511
P-KN3
B-Q5. CHECK
SAVE B5111
K-N2
QXP/R. CHECK
K-B3
P-KN4
P-K5
RXP
SETUP B5111
K-B3
QXP/R
B-B1
B-N5. CHECK
K-B4
P-KN4. CHECK
KXP
Q-R4. CHECK
K-B4
P-B3
SETUP B1
R-N1
B-Q2
R-K1
R-N1
P-QN3
P-QN4
B-N2
N-B2
Q-B2
Q-K2. GLIGORIC TAL CANDIDATES 1959
SETUP B
Q-B2
N-R3
SAVE B6
R-K1
B-N5
B-KN5
P-R3
B-R4
BXN
BXB
P-B5
B-K2
R-QB1
P-B4
Q-R4
K-R1
P-QN4
P-QR3
N-B4. KLUGER MATANOVIC SOMBOR 1969
SETUP B6
P-QR3
N-B2
R-K1
Q-K2
B-KN5
P-KR3
B-R4
P-KN4
B-N3
N-R4
R/R-Q1
NXB
P/RXN
P-N5
N-KR4
Q-N4. KORCHNOI BILEK SOUSSE 1968
SETUP B6
B-KN5
P-R3
B-R4
N-QN5
Q-N1
P-KN4
B-N3
N-R4
P-QR3
NXB
P/BXN
N-R3
N-Q2
N-B2
N-B4
R-B1. MOISEEV SHAMKOVICH CCCP 1968
SETUP B6
B-KB4
N-QN5
Q-N1
N-R4
B-KN5
P-B3
B-K3
P-B4
SAVE B61
P-QR3
PXP
N-KN5
N-Q6
SAVE B62
BXN/5
PXB
SAVE B63
N/5XP/4
P-B5
SAVE B64
Q-Q1
B-B4
Q-B3
Q-Q2
B-N5. PORTISCH ADAMSKI RAACH 1970
R-KB1
Q-N3
B-N3
SETUP B64
B-N5
Q-N3
N-B6. CHECK
BXN
BXB
R-B1
SETUP B63
N/3XP
P-B5
Q-B2
P-KR3
QXP
N-K4. REE TRINGOV TITOVO UZICE 1967
SETUP B63
P-B3
N-K4
SETUP B62
N/5XP/4
NXP/N
R-R2
B-B4
SETUP A2
B-KB4

P-QR3
P-QR4
B-N2
N-B3
O-O
B-K2
B-N5
O-O
R-K1
P-R3
NXP/5
SAVE AA2
NXN
RXN
B-KN5
Q-K1
B-Q3
BXN
QXB
R-QN5. UHLMANN FISCHER PALMA 1971
SETUP AA2
PXB
BXN
PXB
NXP/6
SETUP A21
B-Q3
B-N5
O-O
O-O
P-KR3
BXN
QXB
P-QR3
P-QR4
N/1-Q2. SHASHIN SAVON CCCP 1970
SETUP A21
B-KN5
P-QR3
P-QR4
P-R3
SAVE A211
B-R4
P-KN4
B-N3
N-R4
B-K2
O-O
N-Q2
NXB
P/RXN
P-B4
PXP
BXP
N-B4
Q-K2
N-N6
R-R2
O-O
N-Q2. BILEK EVANS AMSTERDAM 1964
SETUP A211
B-KB4
B-N5
B-K2
O-O
O-O
R-K1
Q-B2
Q-B2
R/B-K1
N/1-Q2
P-R3
BXN
BXB
P-B5
B-K2
R/R-B1
P-R5
N-B4. GELLER TAL CCCP 1957
SETUP A21
Q-R4. CHECK
B-Q2
Q-N3
Q-B2
B-K2
O-O. VAGANJAN KORELOV CCCP 1973
SETUP A21A
B-KN5
P-KR3
SAVE A21B
B-KB4
P-KN4
SAVE A21A1
B-K3
N-N5
B-Q2
P-B4. PALMASSON MATULOVIC HAVANA 1966
SETUP A21A1
B-N3
N-R4
N-Q2
NXB
P/RXN
N-Q2
N-B4
Q-K2. UHLMANN GLIGORIC HAVANA 1966
SETUP A21B
B-R4
P-KN4
SETUP A21B
B-K3
P-QN4
SAVE A21C
BXP/N
NXP/5
NXN
Q-R4. CHECK
SAVE A21D
N-B3
BXN. CHECK
PXB
QXB
Q-N3
B-R3. DONNER PORTISCH LUGANO 1968
SETUP A21C
P-K5
N-N5
SETUP A21D
Q-Q2
QXB
NXP/6
QXP
R-QB1
B-R3
MSG VARIATIONS WITH NEITHER P-K4 NOR P-KB4
SETUP A1
N-B3
P-KN3
SAVE A6
N-Q2
B-N2
SAVE A7
N-B4
O-O
B-B4
N-K1
SAVE A61
Q-Q2
P-N3
SAVE A62
P-K3
B-QR3
P-QR4
B/3XN
BXB
P-QR3
O-O
N-Q2
R/R-N1
P-B4. BORISENKO SUETIN CCCP 1958
SETUP A62
N-N5
B-QR3
SAVE A621
P-QR4
BXN
PXB
N-Q2
NXP/Q
N/2-B3
NXN
RXN
R-Q1
N-K5
Q-B2
Q-B3. GOLDIN SHAMKOVICH CCCP 1958
SETUP A621
N/5XP/6
NXN
NXN
P-KN4
B-N3
P-B4
SETUP A61
P-K4
P-B4
SETUP A61
P-K3
P-KN4
B-N3
P-B4
P-B4
Q-K2
Q-Q2
PXP
BXP
N-Q2
B-K2
N-K4. SUETIN KAGAN CCCP 1956
SETUP A61
N-N5
B-Q2
N/5XP/6
P-QN4
NXN
BXN
N-K5
Q-Q3
N-Q3
QXP
SETUP A6
B-N5
B-N2
SAVE A63
N-Q2
O-O
P-K3
R-K1
B-K2
P-QR3
P-QR4
N/1-Q2
O-O
Q-B2
Q-B2
N-N3. BOTVINNIK TAL 1960
SETUP A63
P-K4
P-KR3
B-R4
P-R3
P-QR4
P-KN4
B-N3
N-R4
N-Q2
NXB
P/RXN
N-Q2
B-K2
N-K4. TOLUSH SUETIN CCCP 1959
SETUP A6
P-KN3
B-N2
B-N2
O-O
O-O
B-N5
SAVE A66
P-KR3
BXN
BXB
Q-Q2
B-N2

P-QN4. ASAFOV NEJELOV 1967
SETUP A66
B-B4
N-R4
B-N5
Q-Q2
Q-N3
N-R3. FURMAN GUFELD CCCP 1971
SETUP A6
B-B4
B-N2
Q-R4. CHECK
B-Q2
Q-N3
Q-B2
P-K4
O-O
SAVE A64
N-Q2
N-R4
B-K3
P-B4
PXP
PXP
P-N3
N-R3
B-K2
P-KB5
PXP
NXP
R-KN1
R/R-K1
N/2-K4
K-R1. CHUKAYEV SUETIN CCCP 1961
SETUP A64
B-Q3
N-R3
O-O
N-R4
B-K3
R/R-N1
P-QR4
N-N5
B-K2
B-N5. TORAN NIEVERGELT LUGANO 1966
SETUP A64
B-K2
P-QR3
P-QR4
B-N5
O-O
N/1-Q2
R/B-K1
BXN
BXB
R/B-K1. LUTIKOV SUETIN CCCP 1961
MSG VARIOUS TRANSPOSITIONAL POSSIBILITIES IN THE MAIN LINE
SETUP A6
P-K4
B-N2
B-K2
O-O
SETUP A7
P-K4
O-O. SPASSKY FISCHER REYKJAVIK 1970
SETUP A21
N-Q2
O-O
B-K2
R-K1
SETUP A21A
N-Q2
R-K1
MSG QUEEN PAWN VARIATIONS SIDESTEPPING THE BENONI
SETUP Q
N-QB3
P-Q4
SAVE QQ1
B-N5
P-KR3
SAVE QQ2
BXN
P/KXB
P-K3
P-B3
B-Q3
B-Q3
Q-B3
O-O
N/1-K2
R-K1
O-O-O
P-QN4
P-KN4
P-N5
N-R4
N/1-Q2
P-KR4
N-N3. TAL GELLER CURACAO 1962
SETUP QQ2
B-R4
P-K3
P-K3
P-B4
N-B3
PXP
QXP
P-R3
B-K2
N-B3
Q-Q3
B-K2. ROSETTO EVANS BUENOS AIRES
SETUP QQ1
B-B4
P-B4
P-K3
P-QR3
N-B3
B-N5
B-K2
BXN
BXB
PXP
PXP
P-K3
O-O
B-Q3
N-K2
O-O
Q-Q2
BXB
QXB
Q-N3
P-QN3
N-B3
P-B4
R/B-K1. TOTH MATULOVIC YUGOSLAVIA 1957
SETUP QQ1
P-K4
NXP
NXN
PXN
P-KB3
P-K4
P/QXP/K
QXQ. CHECK
KXQ
PXP
NXP
B-QB4
B-Q3
O-O
B-KB4
N-B3
R-K1
B-K3
SETUP QQ1
P-B3
B-B4
B-N5
P-B4
P-K4
PXP/Q
BXN
PXN
BXP/B
PXP
QXQ. CHECK
KXQ
O-O-O. CHECK
K-K1. KROGIUS ARONIN CCCP 1952
SETUP Q
B-N5
P-Q4
BXN
P/KXB
P-K3
B-Q3
P-QB4
PXP
BXP
O-O. TROMPOVSKI VAITONIS BUENOS AIRES 1939
SETUP Q
P-KB3
P-Q4
P-K4
PXP
N-B3
B-B4
SAVE Q31
P-KN4
B-N3
P-N5
N-Q4
NXP
P-K3
P-QB4
N-K2
N-N3
N-B4
NXN
BXN. TARTAKOVER SIMONOVIC PARIS 1954
SETUP Q31
PXP
NXP
Q-B3
N-Q3
B-KB4
Q-B1
SETUP Q
N-KB3
P-Q4
SAVE Q4
P-B4
PXP
SETUP Q4
P-K3
P-KN3
B-Q3
B-N2
O-O
O-O
N/1-Q2
P-B4
P-B3
N/3-Q2
Q-K2
N-QB3
P-KR3
R-K1
B-N5
P-QR3
B-R4
P-QN4
B-B2
B-N2. ROSETTO KORCHNOI BUENOS AIRES 1960
SETUP Q4
N-B3
B-B4
SETUP Q4
B-N5
N-K5
SETUP Q
P-K3
P-Q4
B-Q3
P-B4
P-QB3
N-B3
P-KB4
B-N5
N-B3
P-K3
O-O

```
B-K2
Q-K1
O-O
N-K5
B-B4
SETUP Q1
PXP
P-K3
N-KB3
BXP
N-B3
P-Q4
P-K3
O-O
SETUP Q1
N-KB3
PXP
NXP
P-K4
SAVE Q11
N-B2
P-Q4
PXP
QXP
QXQ
NXQ
P-K4
N-B2
N-B3
B-K3. KRAIDMAN CZERNIAK ISRAEL 1961
SETUP Q11
N-N5
B-B4
N-Q6. CHECK
K-K2
SETUP Q2
PXP
P/BXP
B-N5
P-Q4
P-K4
P-KR3
BXN
QXB
P/BXP
PXP
PXP
B-Q3
SAVE Q21
B-N5. CHECK
N-Q2
N-QB3
O-O
N-B3
N-K4. FOGUELMAN MECKING BUENOS AIRES 1967
SETUP Q21
N-QB3
O-O
SETUP Q2
P-Q6
Q-N3
B-B4
N-K5
SETUP Q2
N-KB3
PXP
PXP
P-Q3
SETUP Q3
NXP
NXN
QXN
N-B3
N-B3
P-Q3
P-K4
B-K3
SW LW OFF;SW LB OFF;SW NOR OFF
SW EXP ON
B/NIMZD
INI
YES.  NIMZOVICH DEFENSE.
SW NO ON;SW LW OFF;SW LB ON;SW EXP OFF
LET MSCODE=0;LET MSMASK=77
P-K4
N-QB3. (PAGE 1)
SAVE MOVE2 MYERS V-C
N-KB3
P-K3
P-Q4
P-Q4
SAVE MOVE4 MYERS V-C1B
P-K5
P-B3
SAVE MOVE5 MYERS V-C1B1
PXP
NXP/B3
SAVE MOVE6
P-B3
B-Q3
SET MOVE6
B-QN5
B-Q2
SET MOVE6
N-B3
B-Q3
B-KN5
O-O
SAVE MOVE8
Q-Q2
Q-K1
O-O-O
P-KR3
B-R4.?
N-K5
Q-K1
B-N5
SET MOVE8
B-Q3
N-QN5
O-O
NXB
QXN
P-B4
SET MOVE5 MYERS V-C1B2
B-QN5
B-Q2
Q-K2
Q-K2
O-O
O-O-O
R-K1
R-K1
P-B3
Q-B2
SET MOVE5 MYERS V-C1B3
N-B3
B-Q2
B-QN5
B-N5
BXN
BXB
O-O
BXN
PXB
B-N4
SET MOVE5 MYERS V-C1B4
N-R4
PXP
Q-R5.CHECK
P-N3
NXP
N-B3
Q-R4
PXN
QXR
NXP
SAVE MOVE10
B-Q3
P-K5
SET MOVE10
B-KN5
K-B2
SET MOVE10
K-Q1
N-K5
SAVE MOVE11
B-K3
N-KB4
K-K1
NXB
PXN
Q-N4
K-K2
P-N3
SET MOVE11
QXP
N-QB3
Q-B4
B-Q3
Q-B3
N-Q5
Q-K3
B-B4
SET MOVE11
B-KR6
Q-K2
QXB.CHECK
QXQ
BXQ
KXB
R-N1
NXP/KB7.CHECK
K-B1
P-K5
SET MOVE4 MYERS V-C2
PXP
PXP
SAVE MOVE5
B-KN5
B-K2
BXB
QXB.CHECK
Q-K2
B-B4
P-B3
B-K5
N/QN1-Q2
O-O-O. VON HOLZHAUSEN-NIMZOVICH, HANOVER 1926
SET MOVE5
B-Q3
B-Q3
SAVE MOVE6
B-K3
B-KN5
P-B3
N/KN1-K2
N/QN1-Q2
Q-Q2
N-N3
O-O-O
P-KR3
B-R4
Q-B2
P-B4
O-O-O
P-B5
B-Q2
BXN
PXB
Q-K3
SET MOVE6
O-O
B-KN5
P-B3
N/KN1-K2
R-K1
Q-Q2
N/QN1-Q2
O-O-O
SAVE MOVE10
Q-B2
P-B3
P-KR3
B-K3
N-B1
P-KN4
P-QN4
R/Q1-N1
P-QR4
N-Q1
P-N5
P-KR4
SET MOVE10
```

```
P-N4
R/Q1-K1
N-N3
P-B3
N-B5
BXN/B4
P/NXB
N-Q1
SET MOVE4 MYERS V-C3
N/QN1-Q2
Q-K2
SAVE MOVE5
PXP
PXP.CHECK
B-K2
B-N5
SET MOVE5
P-B3
P-B3
B-Q3
B-Q2
O-O
O-O-O
R-K1
Q-B2
SET MOVE2
N-QB3. (LINE 5)
P-K3
SAVE MOVE3 MYERS IV-A
P-Q4
P-Q4
SAVE MOVE4 MYERS IV-A2
P-K5
P-B3
SAVE MOVE5
PXP
N/KN1XP
SET MOVE5
B-QN5
B-Q2
SET MOVE5
N-B3
B-Q2
B-KB4
PXP
N/KB3XP
NXN
BXN
N-B3
B-Q3
P-B4
PXP
BXP
Q-K2
O-O
O-O-O
Q-R4
K-N1
Q-N3
P-KN4
B-K1
P-N5.?!
B-R4
SET MOVE5
P-B4
Q-Q2
N-B3
Q-B2
P-QR3
B-Q2
P-QN4
N/KN1-K2
B-K3
O-O-O
B-Q3
K-N1
O-O
N-B1
N-Q2
B-K1
Q-K2
P-KR4.GENIN-VINOGRADOV, LENINGRAD 1958.
SET MOVE4 MYERS IV-A3
N-B3
B-N5
SAVE MOVE5
B-Q3
N/KN1-K2
SET MOVE5
P-K5
N/KN1-K2
SAVE MOVE6
B-Q3
N-B4
SET MOVE6
B-Q2
N-B4
N-K2
B-K2
P-B3
O-O
SET MOVE6
P-QR3
BXN.CHECK
PXB
N-R4
SET MOVE3 MYERS IV-B2
N-B3
P-Q4
SAVE MOVE4
B-N5
B-N5
SET MOVE4
P-Q3
B-N5
B-Q2
P-Q5
N-K2
BXB
QXB
P-K4
P-KN3
P-B3
SET MOVE2 ARNOLD LINE 6, MYERS VI-C
P-KB4. (LINE 6)
P-K3
SAVE MOVE3
P-Q4
P-Q4
SAVE MOVE4
P-K5
N-R3
N-KB3
B-K2. = BLACK PLANS N-KB4, P-KR4-R5, THEN P-QB4 EVENTUALLY.
SET MOVE4 ARNOLD LINE 7
N-QB3.? (LINE 7)
PXP. WITH A CLEAR ADVANTAGE.
SET MOVE4 ARNOLD LINE 8
PXP. (LINE 8)
PXP.! WITH A SLIGHT ADVANTAGE. SIMILAR TO FRENCH DEFENSE.
SET MOVE3 ARNOLD LINE 9
N-KB3. (LINE 9)
P-Q4
SAVE MOVE4
PXP
PXP
B-N5
B-Q3
P-Q4
N-B3
O-O
O-O. =
SET MOVE4 ARNOLD LINE 11
N-B3. (LINE 11)
N-B3
SAVE MOVE5
P-K5
N-K5.!
NXN.?!
PXN
N-KN5
Q-Q4. =+
SET MOVE5 ARNOLD LINE 12
P-Q3. (LINE 12)
PXP
NXP
NXN
PXN
QXQ.CHECK
KXQ. (NOTE A)
B-Q3. (NOT P-K4)
B-N5
B-Q2. =+
SET MOVE5 ARNOLD LINE 13
Q-K2. (LINE 13)
P-Q5
SAVE MOVE6
N-QN1
N-QN5
P-Q3
P-QB4.! =+
SET MOVE6 ARNOLD NOTE B
N-QN5. (NOTE B)
P-QR3
N-R3
B-B4. =+
SET MOVE2 ARNOLD LINE 15, MYERS VI-E
P-KN3. (LINE 15)
P-Q4.!
SAVE MOVE3
PXP.?!
QXP
SAVE MOVE4
N-KB3
B-N5
B-N2
N-B3
N-B3
Q-Q2
O-O
P-K4.! =
SET MOVE4 ARNOLD LINE 16
Q-B3. (LINE 16)
Q-K3. CHECK
Q-K2
N-Q5
QXQ
BXQ
N-QR3
B-Q4. WINNING (MYERS P. 43)
SET MOVE3 ARNOLD LINE 17
P-Q4.!? (LINE 17)
PXP
SAVE MOVE4
P-Q5.?!
N-K4. =+
SET MOVE4
B-QN5.? (LINE 18)
P-QR3. CLEAR ADVANTAGE
SET MOVE3
N-QB3
P-Q5
N/QB3-K2
P-K4
SET MOVE2 ARNOLD LINE 19, MYERS VI-A
B-B4. (LINE 19)
N-B3
SAVE MOVE3
P-Q3
P-K3
N-KB3
P-Q4
PXP
PXP
B-N3
B-K2
SET MOVE3
N-QB3
P-K3
SAVE MOVE4
P-Q3
P-Q4
B-QN5
PXP
SAVE MOVE6
BXN.CHECK
PXB
NXP
NXN
PXN
QXQ.CHECK
KXQ
B-R3
SET MOVE6
```

```
NXP
B-K2
N-K2. MJAGMARSUREN-KERES, TALLIN 1971.
B-Q2.! KERES SUGGESTION
SET MOVE4 ARNOLD LINE 20
P-Q4. (LINE 20)
P-Q4
PXP
PXP
B-QN5
B-Q3
B-N5
O-O.! =
BXN/KB6
QXB
NXP
QXP/Q. =+
SET MOVE2 MYERS VI-B
B-N5
P-QR3
SAVE MOVE3 MYERS VI-B1
BXN
P/NXB
P-Q4
P-Q4
N-QB3
P-K3
N/KN1-K2
N-B3
B-N5
B-K2
P-K5
N-Q2
BXB
QXB
SET MOVE3 MYERS VI-B2
B-R4
P-QN4
B-N3
N-B3
P-Q3
P-K3
N-KB3
P-Q4
SET MOVE2 MYERS VI-D
P-Q3
P-K4
SAVE MOVE3
P-KB4
B-B4
SAVE MOVE4
PXP
BXN
RXB
Q-R5.CHECK
P-N3
QXP/R
R-N2
Q-R8
P-Q4
P-Q3
SET MOVE4
N-KB3
P-Q3
SET MOVE3
P-KN3
B-B4
B-N2
P-Q3
SET MOVE2 MYERS VI-F
P-QN3
P-Q4
PXP
QXP
N-QB3
Q-QR4
B-N2
N-B3
B-N5
B-Q2
SET MOVE2 MYERS VI-G
P-QR3
N-B3
N-QB3
P-Q4
PXP
NXP
SAVE MOVE5
B-N5
NXN
P/NXN
Q-Q4
SET MOVE5
B-B4
B-K3
B-N3
NXN
P/NXN
BXB
PXB
P-K4
SET MOVE2 MYERS VI-H2
P-QB4
P-K3
N-QB3
B-N5
P-Q4
P-Q4
P/BXP
PXP
P-K5
B-KB4
SET MOVE2 MYERS VI-I
P-QB3
P-Q4
PXP
QXP
P-Q4
P-K4
SET MOVE2 ARNOLD PAGE 5
P-Q4. (PAGE 5)
P-Q4
SAVE MOVE3 MYERS I-B
P-K5
P-B3
SAVE MOVE4 MYERS I-B1
P-KB4
B-B4
SAVE MOVE5 ARNOLD LINE 21, MYERS I-B1D
P-B3. (LINE 21)
P-K3
SAVE MOVE6 ARNOLD LINE 21
N-K2
N/KN1-K2
N-N3
B-N3
B-Q3
Q-Q2
O-O
O-O-O. =
SET MOVE6 ARNOLD LINE 22, MYERS I-B1D
N-Q2. (LINE 22)
N-R3
P-KN3
B-K2
B-R3
PXP. = (SCHWARZ P. 208)
P-KN4
B-R5.! CLEAR ADVANTAGE
SET MOVE5 ARNOLD LINE 23, MYERS I-B1A
N-KB3. (LINE 23)
P-K3
SAVE MOVE6 MYERS I-B1A1
B-Q3
BXB
QXB
N-N5
SAVE MOVE8
Q-N5.CHECK
K-B2.!
N-R3
P-QR3
SAVE MOVE10
Q-K2
P-QB4. UNCLEAR
SET MOVE10
QXP/N7.?? (NOTE E)
R-N1
Q-R7
Q-B1.! CLEAR ADVANTAGE
SET MOVE8
Q-K2
P-KB4. ETC =
SET MOVE6 ARNOLD LINE 24, MYERS I-B1A2
B-N5. (LINE 24)
Q-Q2
O-O.?
P-QR3
B-Q3
B-K5
P-B3
P-B4
Q-K2
N-R3
B-K3
B-K2
N/QN1-Q2
O-O
P-KR3
N-Q1. =+ (SCHWARZ P. 208)
SET MOVE6 ARNOLD LINE 25, MYERS I-B1A3
N/QN1-Q2. (LINE 25)
N-N5.! CLEAR ADVANTAGE. (SCHWARZ P. 208)
SET MOVE5 ARNOLD LINE 26, MYERS I-B1C
N-K2. (LINE 26)
P-K3
N-N3
PXP
SAVE MOVE7
NXB
PXN
P/QXP
P-QR3. UNCLEAR.  INTENDING TO PLACE N AT K3: Q-Q2, N-Q1-K3
SET MOVE7 ARNOLD LINE 27
P/BXP. (LINE 27)
Q-R5.!?
SAVE MOVE8
P-B3
B-KN5.!?
SAVE MOVE9
B-K2
P-KR4
O-O
N-R3. UNCLEAR
SET MOVE9 ARNOLD NOTE B1
Q-R4.!? (NOTE B1)
N-R3
B-QN5
K-Q2
O-O
B-K2. UNCLEAR
SET MOVE8 ARNOLD LINE 28
B-QN5. (LINE 28)
N/KN1-K2
O-O.?!
P-KN4.!!
SAVE MOVE10
Q-R5.CHECK
B-N3
SAVE MOVE11
QXP/N
QXP/Q.CHECK  WITH A CLEAR ADVANTAGE.
SET MOVE11 ARNOLD NOTE E
QXQ. (NOTE E)
PXQ
N-K2
BXP. =+
SET MOVE10 ARNOLD NOTE D
N-R5.? (NOTE D)
O-O-O.!
SET MOVE8 ARNOLD NOTE C
B-K3. (NOTE C)
N-R3. ETC
SET MOVE5 MYERS I-B1B
B-N5
P-K3
N-KB3
Q-Q2
B-K3
P-QR3
B-K2
N/KN1-K2
SET MOVE4 ARNOLD LINE 29
N-KB3. (LINE 29)
```

```
B-B4
SAVE MOVE5
B-Q3
Q-Q2
SAVE MOVE6
N-QB3.!
BXB
QXB
P-K3
Q-N5
O-O-O
PXP
PXP. UNCLEAR
SET MOVE6 ARNOLD NOTE G1
O-O.? (NOTE G1)
B-K5.! =
SET MOVE6 ARNOLD NOTE G2
B-KB4.?! (NOTE G2)
P-KN4.! =+
SET MOVE6 ARNOLD NOTE G3
P-KR3.? (NOTE G3)
P-KN4
P-B3
P-KR3
B-K3
O-O-O
Q-B2
BXB
QXB
B-N2.! =+ (SCHWARZ PP. 207-208)
SET MOVE5 ARNOLD NOTE F
B-QN5.!? (NOTE F)
Q-Q2. ETC =
SET MOVE4 MYERS I-B3
B-Q3
P-KN3
SAVE MOVE5 MYERS I-B3A
N-KB3
B-N5
PXP
N/KN1XP
P-B3
P-K4
PXP
NXP
Q-K2
B-Q3
N/QN1-Q2
Q-K2. WITH A CLEAR ADVANTAGE
SET MOVE5 MYERS I-B3B
PXP
N/KN1XP
P-QB3
P-K4
PXP
NXP
Q-K2
Q-K2. WITH A CLEAR ADVANTAGE
SET MOVE5 MYERS I-B3C
P-KB4
NXP/Q
BXP.CHECK
PXB
QXN
B-N2
SET MOVE3 MYERS IV-A
N-QB3
P-K3
SET MOVE3 ARNOLD PAGE 7
PXP. (PAGE 7)
QXP
SAVE MOVE4 MYERS II-A
N-KB3
B-N5
SAVE MOVE5 ARNOLD LINE 30, MYERS II-A3
N-B3. (LINE 30)
BXN
NXQ
BXQ
NXP/B7.CHECK
K-Q1.!
NXR
BXP
SAVE MOVE9
B-KB4
NXP
SET MOVE9
P-Q5
N-Q5
SAVE MOVE10
B-KB4
P-K4
PXP.EP
B-N5.CHECK
B-Q2
B-Q3
SET MOVE10
B-K3
P-K4
SAVE MOVE11
BXN
PXB
SAVE MOVE12
N-B7
B-N5.CHECK
K-K2
P-Q6.CHECK
SET MOVE12
K-Q2
B-K5
R-B1
B-N5.CHECK
K-Q1
B-Q3
SET MOVE12
R-B1
B-N5.CHECK
K-K2
P-Q6.CHECK
K-B3
P-Q7. WINS
SET MOVE11
PXP.EP
B-N5.CHECK
B-Q2
B-Q3.!
PXP
N-K2.CLEAR ADVANTAGE
SET MOVE5 ARNOLD LINE 31
P-B4. (LINE 31)
BXN
PXQ
BXQ
PXN
B-R5
SAVE MOVE8
PXP
R-N1. =
SET MOVE8 ARNOLD NOTE B
P-Q5.!? (NOTE B)
P-K3.!
SET MOVE5 MYERS II-A2
B-K2
O-O-O
SAVE MOVE6 ARNOLD LINE 32, MYERS II-A2A
N-B3. (LINE 32)
Q-QR4
B-K3
N-B3
SAVE MOVE8
O-O
P-K4
SET MOVE8
P-KR3
BXN
BXB
NXP
SAVE MOVE10
BXP.CHECK
KXB
BXN
P-K4
Q-B3.CHECK
P-B3
B-K3
B-N5
SET MOVE10
BXN
Q-N5
BXP/N.CHECK
QXB
SET MOVE8
N-N5
BXB
QXB
NXP
BXN
RXB
NXP/B.?
R-K5.WINS
SET MOVE8
N-Q2.!
BXB
QXB
Q-KB4
N-N3
P-K4.! = MCO-11 P. 193 NOTE (C)
SET MOVE6 ARNOLD LINE 33
P-QB4. (LINE 33)
Q-KB4
B-K3
BXN
BXB
NXP
SAVE MOVE9
B-N4
N-B7.CHECK
QXN
QXB
SET MOVE9
BXN
Q-K3.CHECK
B-K2
P-QB4
SET MOVE6
B-K3
P-K3
SET MOVE4 MYERS II-C
B-K3
P-K4
SAVE MOVE5 MYERS II-C1
N-QB3
B-QN5
SET MOVE5
P-QB4
Q-R4.CHECK
SAVE MOVE6
N-QB3
PXP
BXP
NXB
SET MOVE6
B-Q2
N-QN5
P-Q5
B-KB4
P-QR3
B-B4
Q-N3
Q-N3
PXN
BXP/B.CHECK
K-Q1
BXN/KN8
P-B5
Q-KN3
RXB
BXN
SET MOVE5 MYERS II-C2
PXP
QXQ.CHECK
KXQ
NXP
P-KB3
B-KB4
N-Q2
O-O-O
SET MOVE5 MYERS II-C3
N-K2
B-KN5
P-KB3.?
BXP.!
N/QN1-B3
B-QN5
```

```
B-Q2
BXN/QB6
SET MOVE5 MYERS II-C4
N-KB3
B-KN5
B-K2
BXN
BXB
P-K5
B-K2
O-O-O
P-QB3
P-B4
O-O
B-Q3
P-QB4
Q-B2
P-B4.?
B-B4.!
SW EXP ON
X/END
STR WSICIL,A
SWITCH LW OFF;SWITCH LB OFF;SWITCH NOR OFF
SWITCH EXPERIENCE,ON
INI
DROP
```