```
                    CHNG = FALSE
                    HDBUF.PHACT = 0
                    XJ(OPER(DFWRIT),PROFILE,PHACT,HDBUF,PHACT,1) INP
          IF CHNG DO XJ(OPER(DFWRIT),PROFILE,POS,BUF,PERLKSZ)
          POS = POS + PERLKSZ INP
     10
// CLOSE THE FILES
XJ(OPER(DFCLO),PROFILE)
XJ(OPER(DFCLO),DAYFILE)
ISTRT
*
*
*
*                   CODE REMOVED FROM SYNTAXA
*
*                   COULD BE PUT BACK FOR DEBUGGING PURPOSES
*
*                   DUMP FUNCTION DATA
*
DUMPFNC   I = 1
          OUTPUT = ''
          OUTPUT = '      FUNCTION TABLE'
          OUTPUT = ''
DUMPFNC1  GT( I, FNCCNT)        :S(RETURN)
          OUTPUT = I ' ' $(F2 I) ' ' $(F3 I) ' ' $(F1 $(F2 I))
          I = I + 1 :(DUMPFNC1)
*
*                   DUMP TOKEN TABLE
*
DUMPTKN   OUTPUT = ''
          OUTPUT = ''
          OUTPUT = '      TOKENS TERMINAL '
          OUTPUT = ''
          I=1
DUMPTKN1  GT( I, TERMTKNCNT)   :S(DUMPTKN2)
          PART = I ' ' $(T2 I) ' ' $(T3 I) ' ' $(T4 I) '  ' $(T11 I )
          OUTPUT = PART ' ' $(T1 $(T2 I ))
          I= I + 1   :(DUMPTKN1)
*
DUMPTKN2  OUTPUT = ''
          OUTPUT = '      TOKENS   NON TERMINAL'
          OUTPUT = ''
*
DUMPTKN3  GT( I, TKNCNT )   :S(RETURN)
          OUTPUT = I ' ' $(T2 I) ' ' $(T3 I) ' ' $(T1 $(T2 I))
          I = I + 1 :(DUMPTKN3)
*
*
DUMPSTATE1          OUTPUT = ''
                    OUTPUT = ''
          OUTPUT = '      STATE TABLE'
          I=0
*
DUMPSTATE2          GT( I, STATECNT) :S(RETURN)
          OUTPUT = I ' ' $(S2 I) ' ' $(S5 I)
```

```
            J = 1
DUMPSTATET2             GT(J, TKNCNT) :S(DUMPSTATET4)
            IDENT($(S4 I ≠,≠ J), ≠≠ )  :S(DUMPSTATET3)
            OUTPUT = ≠                ≠ J ≠ ≠ $(S4 I ≠,≠ J )
DUMPSTATET3             J = J + 1   :(DUMPSTATET2)
DUMPSTATET4             I = I + 1    :(DUMPSTATET1)
*
*
DUMPRULES OUTPUT = ≠≠
            OUTPUT = ≠≠
            OUTPUT = ≠      RULE TABLE≠
            OUTPUT = ≠≠
            I = 1
DUMPRULES1              GT( I, RULECNT)    :S(RETURN)
            OUTPUT = I ≠ ≠ $(R4 I) ≠ ≠ $(R5 I) ≠ ≠ $(R6 I)
                OUTPUT = ≠            ≠ $(R7 I)
               E = $(R8 I)
DUMPRULES2 IDENT(E, ≠≠)    :S(DUMPRULES3)
            OUTPUT = ≠               ≠ VALUE(E)
            E = NEXT(E)   :(DUMPRULES2)
DUMPRULES3              I = I + 1    :(DUMPRULES1)
*
*
*
**                      DUMP NON TERMINAL EXTRA INFO
*
DMPNONTER               OUTPUT =≠≠
            OUTPUT = ≠       NON TERMINAL EXTRA INFO ≠
            OUTPUT = ≠≠
            I = TERMTKNCNT + 1
*
DMPNONTERM1             GT( I, TKNCNT )      :S(RETURN)
            OUTPUT = I ≠ ≠ $( T2 I )
            OUTPUT = ≠          INITIALS≠
            J = $( T5 I )
DMPNONTERM1A            IDENT( J , ≠X≠ )     :S(DMPNONTERM2)
                        OUTPUT  = ≠          ≠ J
                        J = $( T5 I ≠,≠ J )  :(DMPNONTERM1A)
*
DMPNONTERM2             OUTPUT = ≠≠
            OUTPUT = ≠          INITIAL TERMINALS≠
            J = $(T7 I )
DMPNONTERM3                      IDENT( J, ≠X≠ )      :S(DMPNONTERM4)
                        OUTPUT = ≠      ≠ J
                        J = $(T7 I ≠,≠ J )   :(DMPNONTERM3)
DMPNONTERM4             OUTPUT = ≠≠
            OUTPUT = ≠      FINALS≠
            J = $(T8  I)
DMPNONTERM5             IDENT( J, ≠X≠)  :S(DMPNONTERM6)
                        OUTPUT = ≠          ≠ J
                        J = $(T8 I ≠,≠ J )  :(DMPNONTERM5)
*
DMPNONTERM6             OUTPUT = ≠≠
            OUTPUT = ≠      FINAL NON TERMINALS≠
            J = $(T9 I)
```

```
DMPNONTERM7  INDENT(J , ≠X≠)   :S(DMPNONTERM8)
             OUTPUT = ≠    ≠ J
             J = $(T9 I ≠,≠ J) :(DMPNONTERM7)
*
DMPNONTERM8  OUTPUT = ≠≠
             OUTPUT = ≠    FOLLOWING TERMINALS≠
             OUTPUT = ≠≠
             J = $(T10 I)
DMPNONTERM9  INDENT(J, ≠X≠)   :S(DMPNONTERM10)
             OUTPUT = ≠    ≠ J
             J = $(T10 I ≠,≠ J )    :(DMPNONTERM9)
*
DMPNONTERM10 I = I + 1  :(DMPNONTERM1)
             DEFINE( ≠NEXTITEM(NULL)≠,≠NEXTITEM≠)
             DEFINE( ≠NEWSTATE(OLD,TKN)≠,≠NEWSTATE≠)
             DEFINE( ≠NEXTTKN(NULL)≠,≠NEXTTKN≠)
             DEFINE( ≠NEXTFNC(NULL)≠,≠NEXTFNC≠)
             DEFINE( ≠EXTEND/KEY)≠,≠EXTEND≠)
             DEFINE( ≠ASSEN(LOC,OP,RAND,COMM)A,B≠ )
             DEFINE( ≠SAVE(NULL)≠ )
             DEFINE( ≠SPUT(ITEM)TEMP≠ )
             DEFINE( ≠PUT(TTE4)≠ )
*
*            DEFINE( ≠DUMPFNC(NULL)≠,≠DUMPFNC≠)
*            DEFINE( ≠DUMPTKN(NULL)≠,≠DUMPTKN≠)
*            DEFINE( ≠DUMPSTATE(NULL)≠,≠DUMPSTATE≠)
*            DEFINE( ≠DUMPRULES(NULL)≠,≠DUMPRULES≠)
*            DEFINE( ≠DUMPNONTERM(NULL)≠,≠DUMPNONTERM≠)
             DEFINE( ≠PRINTRULES(NULL)LINE≠ )
             DEFINE( ≠PRINTS(S)NM,LN,T≠ )
             DEFINE( ≠PRINTTKNS(NULL)≠ )
*
*
*
             DATA( ≠LISTEL(VALUE,NEXT)≠ )
*
             F1 = ≠F1.≠
             F2 = ≠F2.≠
             F3 = ≠F3.≠
             T1 = ≠T1.≠
             T2 = ≠T2.≠
             T3 = ≠T3.≠
             T4 = ≠T4.≠
             T5 = ≠T5.≠
             T6 = ≠T6.≠
             T7 = ≠T7.≠
             T8 = ≠T8.≠
             T9 = ≠T9.≠
             T10 = ≠T10.≠
             T11 = ≠T11.≠
             S2 = ≠S2.≠
             S4 = ≠S4.≠
             S5 = ≠S5.≠
             B4 = ≠B4.≠
             B5 = ≠B5.≠
             B6 = ≠B6.≠
```

```
          R7 = ≠R7.≠
          R8 = ≠R8.≠
*
          DETACH( ≠INPUT≠ )
          INPUT( ≠INPUT≠, ≠GRAMMAR≠, 80 )
          OUTPUT( ≠ASSEMLINE≠, ≠SCMMDD≠, ≠≠ )
          OUTPUT( ≠SAVELINE≠, ≠SAVEF≠, ≠≠ )
*
*
          SYNERR = ≠***** SYNTAX ERROR. ≠
*
*
*
          :(GO)
*
NEXTTKN   ITEM = NEXTITEM()
          IDENT( ITEM , ≠.≠ ) :S(FRETURN)
          NEXTTKN = $( T1 ITEM )
          IDENT( NEXTTKN . ≠≠ ) :F(RETURN)S(FAIL1)
*
NEXTENC   ITEM = NEXTITEM()
          IDENT( ITEM . ≠.≠ ) :S(FRETURN)
          NEXTENC = $( E1 ITEM )
          IDENT( NEXTENC . ≠≠ ) :F(RETURN)S(FAIL2)
*
NEXTITEM  LINE  SPAN(≠ ≠) BREAK(≠ ≠) . NEXTITEM = ≠≠ :S(RETURN)
NEXTITEM1 LINE = INPUT
          OUTPUT = LINE
          LINE  POS(0) ≠*≠     :F(NEXTITEM)S(NEXTITEM1)
*
NEWSTATE  NEWSTATE = $( S4 OLD ≠.≠ TKN )
          IDENT( $(S4 OLD ≠.≠ TKN) . ≠≠ ) :F(RETURN)
          STATECNT = STATECNT + 1
          $(S2 STATECNT ) = $( S2 OLD) ≠.≠ TKN
          $( S4 OLD ≠.≠ TKN ) = STATECNT
          NEWSTATE = STATECNT  :(RETURN)
*
*
*
*                 EXTEND A LIST BY TRANSITIVITY THROUGH NON TERMINALS
*
EXTEND    TKN = TERMTKNCNT + 1
EXTEND1   GT( TKN, TKNCNT )    :S(EXTEND2)
          $(T6 TKN) = ≠X≠
          TKN = TKN + 1        :(EXTEND1)
*
EXTEND2   FLAG = ≠OFF≠
          TKNHD = TERMTKNCNT + 1
*
EXTEND3   GT(TKNHD, TKNCNT )   :S(EXTEND9)
          CHKTKN = $(T6 TKNHD )
          $(T6 TKNHD) = $(KEY TKNHD)
          TKN = $(KEY TKNHD)
*
EXTEND4   IDENT( TKN, CHKTKN)  :S(EXTEND8)
          LE( TKN, TERMTKNCNT)          :S(EXTEND7)
```

```
            SUBTKN = $(KEY TKN)
*
EXTEND5     IDENT(SUBTKN, ≠X≠ ) :S(EXTEND7)
            IDENT( $(KEY TKNHD ≠,≠ SUBTKN), ≠≠ )     :F(EXTEND6)
            $(KEY TKNHD ≠,≠ SUBTKN) = $(KEY TKNHD)
            $(KEY TKNHD) = SUBTKN
            FLAG = ≠ON≠
*
EXTEND6     SUBTKN = $(KEY TKN ≠,≠ SUBTKN)           :(EXTEND5)
*
EXTEND7     TKN = $(KEY TKNHD ≠,≠ TKN )    :(EXTEND4)
*
EXTEND8     TKNHD = TKNHD + 1    :(EXTEND3)
*
EXTEND9     IDENT( FLAG, ≠OFF≠ )          :S(RETURN)F(EXTEND2)
*
*
*
*                    OUTPUT AN ASSEMBLY LINE
*
*
ASSEM       A = LOC ≠                ≠
            A TAB(10) . B  = ≠≠
            A = B OP ≠                ≠
            A  TAB(30) . B  = ≠≠
            IDENT( COMM, ≠≠ )    :S(ASSEM1)
            A = B RAND ≠                                    ≠
            A  TAB(50) . B  = ≠≠
            ASSEMLINE = B COMM   :(RETURN)
*
ASSEM1      ASSEMLINE = B RAND   :(RETURN)
*
*
*                    SUBROUTINE TO SAVE INFO FOR SECOND PASS
*
SAVE        REWIND( ≠SAVEF≠ )
            PUT( TERMTKNCNT )
            PUT( TKNCNT )
            PUT(RULECNT )
            PUT( FNCCNT )
            PUT( STATECNT )
*
            I = 1
*
SAVE1       GT( I, FNCCNT )     :S(SAVE2)
            SPUT( F2 I )
            SPUT( F3 I )
             I = I + 1            :(SAVE1)
*
SAVE2       I = 1
*
SAVE3       GT( I, TERMTKNCNT) :S(SAVE4)
            SPUT( T2 I )
            SPUT( T3 I )
            SPUT( T4 I )
            SPUT( T11 I )
```

```
            PUT( T10 I )
            PUT( ≠X≠ )
            I = I + 1       :(SAVE3)
*
SAVE4       GT( I, TKNCNT )     :S(SAVE7)
            SPUT( T2 I )
            SPUT( T3 I )
            SPUT( T4 I )
            K = T10 I
            SPUT( K )
            J = $(K)
*
SAVE5       IDENT( J, ≠X≠ )     :S(SAVE6)
            L = K ≠,≠ J
            SPUT(L)
            J = $(L)   :(SAVE5)
*
SAVE6       I = I + 1 :(SAVE4)
*
SAVE7       I = 1
*
SAVE8       GT( I, RULECNT)     :S(SAVE9)
            SPUT( R4 I )
            SPUT( R5 I )
            SPUT( R6 I )
            I = I + 1           :(SAVE8)
*
SAVE9       I = 1
*
SAVE10      GT( I, STATECNT )   :S(SAVE13)
            SPUT( S2 I )
            SPUT( S4 I )
            SPUT( S5 I )
*
            J = 1
*
SAVE11      GT(J, TKNCNT)       :S(SAVE12)
            SPUT( S4 I ≠,≠ J )
            J = J + 1           :(SAVE11)
*
SAVE12      I = I + 1           :(SAVE10)
*
SAVE13      PUT( ≠,,,,,,≠ )
            OUT( ≠ITEMS OUT = ≠  ITEMCNT )          :(RETURN)
*
*
*                   ROUTINES USED BY SAVE
*
*
PUT         SAVELINE = ITEM
            ITEMCNT = ITEMCNT + 1          :(RETURN)
*
*
SPUT        TEMP = $(ITEM)
            IDENT( TEMP, ≠≠ )   :S(RETURN)
            PUT( ITEM)
```

```
          PUT( TEMP )          :(RETURN)
*
*
*
*
*
*
*
*
*
*
*
*
*
*                    PRINT OUT A STATE
*
PRINTS    NM = $(S2 S) ≠.≠
          LN = ≠                ≠
PRINTS1   NM ≠.≠ BREAK(≠.≠) . T    = ≠≠    :F(PRINTS2)
          LN = LN ≠ ≠ $(T2 T)      :(PRINTS1)
PRINTS2   OUTPUT = LN  :(RETURN)
*
*
*                    PRINT OUT RULE TABLE
*
PRINTRULES           OUTPUT = ≠≠
          OUTPUT = ≠       RULE INFO≠
          OUTPUT = ≠≠
          I = 1
PRINTRULES1   GT(I, RULECNT)    :S(RETURN)
          LINE = ≠  ≠ I ≠     ≠ $(T2 $(R4 I)) ≠    ≠ $(F2 $(R5 I))
          LINE = LINE ≠     ≠ $(R6 I)
          OUTPUT = LINE
          I = I + 1     :(PRINTRULES1)
*
*
*
*                    PRINT OUT TOKEN INFO
*                    AND PLACE IN ASSEMBLY ALSO
*
PRINTTKNS OUTPUT = ≠≠
          OUTPUT = ≠≠
          OUTPUT = ≠     TOKEN DEFINITIONS≠
          OUTPUT = ≠≠
          I = 1
PRINTTKNS1   GT( I, TKNCNT )    :S(RETURN)
          TEMP = $(T2 I) ≠                              ≠
          TEMP   TAB(24)  .  TEMP1  = ≠≠
          OUTPUT = TEMP1  $(T3 I)
          ASSEM( ≠XXX≠, ≠SET≠, $(T3 I), $(T2 I) )
          I = I + 1      :(PRINTTKNS1)
*
*
*
```

```
*
*
*
*
*                    ACTUAL CODE STARTS HERE
*
*                    FIRST READ IN TOKEN DEFINITIONS, TERMINAL FIRST
*
*
*
GO          ASSEM( ≠≠, ≠IDENT≠, NEXTITEM() )
            ASSEM( ≠≠, ≠EXT≠, ≠SUBGRAM≠ )
            ASSEM( ≠≠, ≠EXT≠, ≠GETTKN≠ )
            ASSEM( ≠≠, ≠EXT≠, ≠GETKYWD≠ )
            ITEM= NEXTITEM()
            IDENT( ITEM, ≠FL≠ ) :F(GO1)
*
            ASSEM( ≠≠, ≠ENTRY≠, ≠FL≠ )
*
GO1         KEYBASE = NEXTITEM()
*
            TKNCNT = 0
            ASSEM( ≠KEYWDL≠, ≠BSS≠, 0 )
*
TKNS1       ITEM1 = NEXTITEM()
            IDENT( ITEM1, ≠.≠ ) :S(TKNS2)
            ITEM2 = NEXTITEM()
            IDENT( ITEM2, ≠.≠ )             :S(FAIL7)
            ITEM3 = NEXTITEM()
            IDENT( ITEM3, ≠.≠ )             :S(FAIL7)
            TKNCNT = TKNCNT + 1
            IDENT( ITEM3, ≠GETKYWD≠ )       :F(TKNS1A)
*
            KEYCNT = KEYCNT + 1
            ASSEM( ≠≠, ≠DATA≠, ≠DL≠ ITEM2 )
            ASSEM( ≠≠, ≠VFD≠, ≠60/≠ KEYBASE ≠+≠ KEYCNT )
            ITEM2 = KEYBASE ≠+≠ KEYCNT
            $(T1 TKNCNT ) = ≠KEYWDL≠    :(TKNS1C)
*
TKNS1A      IDENT( ITEM3, ≠SUBGRAM≠ )       :F(TKNS1B)
*
            ITEM4 = NEXTITEM()
            IDENT( ITEM4, ≠.≠ )             :S(FAIL7)
            ASSEM( ≠≠, ≠EXT≠, ITEM4 )
            $(T1 TKNCNT ) = ITEM4           :(TKNS1C)
*
TKNS1B      $(T1 TKNCNT) = 0    :(TKNS1C)
*
TKNS1C      IDENT( NEXTITEM() , ≠.≠ )       :F(FAIL4)
            $(T1 ITEM1) = TKNCNT
            $(T2 TKNCNT) = ITEM1
            $(T3 TKNCNT) = ITEM2
            $(T4 TKNCNT) = ITEM3
            $(T7 TKNCNT) = TKNCNT
            $(T7 TKNCNT ≠.≠ TKNCNT ) = ≠X≠    :(TKNS1)
*
```

```
*
*                       NOW WE READ IN NON TERMINAL TOKEN DEFINITIONS
*
TKNS2       TERMTKNCNT = TKNCNT
            ASSEM( ≠≠, ≠DATA≠, 0 )
*
TKNS3       ITEM1 = NEXTITEM()
            IDENT( ITEM1, ≠.≠)   :S(FNCS0)
            ITEM2 = ≠200000≠ ( TKNCNT + 1 - TERMTKNCNT )
            IDENT( NEXTITEM() , ≠.≠ )       :F(FAIL4)
            TKNCNT = TKNCNT + 1
            $(T1 ITEM1) = TKNCNT
            $(T2 TKNCNT) = ITEM1
            $(T3 TKNCNT) = ITEM2
            $(T5 TKNCNT) = ≠X≠
            $(T7 TKNCNT) = ≠X≠
            $(T8 TKNCNT) = TKNCNT
            $(T8 TKNCNT ≠,≠ TKNCNT) = ≠X≠
            $(T9 TKNCNT) = ≠X≠
            $(T10 TKNCNT) = ≠X≠   :(TKNS3)
*
*
*
*                       NOW READ IN FUNCTION DEFINITIONS
*
FNCS0       FNCCNT = 0
*
FNCS1       ITEM1 = NEXTITEM()
            IDENT(ITEM1, ≠.≠)    :S(FNCS2)
            ITEM2 = NEXTITEM()
            IDENT( NEXTITEM(), ≠.≠ )        :F(FAIL4)
            FNCCNT = FNCCNT + 1
            ASSEM( ≠≠, ≠EXT≠, ITEM2 )
            $(F1 ITEM1) = FNCCNT
            $(F2 FNCCNT) = ITEM1
            $(F3 FNCCNT) = ITEM2            :(FNCS1)
*
FNCS2
*
*
*
*                   THIS CODE READS IN THE RULES
*                   AND GENERATES THE SYMPLE STATE TABLE
*
*
*
            STATECNT = 0
            RULECNT = 0
*
            DMYEL = LISTEL( ≠≠, ≠≠ )
*
RULE1       RULEFNC = NEXTFNC()   :F(RULE4)
            IDENT( NEXTITEM(), ≠:≠ )   :F(FAIL5)
            RULETKN = NEXTTKN()     :F(FAIL)
            IDENT( NEXTITEM(), ≠:=≠ )   :F(FAIL6)
            RULECNT = RULECNT + 1
            $(R4 RULECNT) = RULETKN
```

```
            $(R5 RULECNT) = RULEFNC
            STATE = 0
            STATE = NEWSTATE(STATE,RULETKN)
            RULESIZE = 2
            EL = DMYEL
            NEXT(EL) = ''
*
RULE2       TKN = NEXTTKN()   :F(FAIL7)
            IDENT( $(T5 RULETKN ',' TKN), '' ) :F(RULE3)
            $(T5 RULETKN ',' TKN) = $(T5 RULETKN)
            $(T5 RULETKN) = TKN
*
RULE3       STATE = NEWSTATE(STATE,TKN)
            NEXT(EL) = LISTEL(TKN,'')
            EL = NEXT(EL)
            RULESIZE = RULESIZE + 1
            LASTTKN = TKN
            TKN = NEXTTKN()       :S(RULE3)
*
            IDENT( $(T8 RULETKN ',' LASTTKN), '' )  :F(RULE3A)
            $(T8 RULETKN ',' LASTTKN) = $(T8 RULETKN)
            $(T8 RULETKN) = LASTTKN
*
RULE3A      $(S5 STATE) = RULECNT
            $(R6 RULECNT) = RULESIZE
            $(R7 RULECNT) = STATE
            $(R8 RULECNT) = NEXT(DMYEL)
*
            RAND = '12/0,T8/' $(F3 RULEFNC ) ',T2/' RULESIZE
            RAND = RAND ',T8/' $(T3 RULETKN )
            ASSEM( 'R,' RULECNT , 'VFD', RAND )       :(RULE1)
*
RULE4
*
            PRINTTKNS()
            PRINTRULES()
            OUTPUT = ''
            OUTPUT = ''
            OUTPUT = ''
*
*
*
            EXTEND(T5)
*
            EXTEND(T8)
*
*
*
*                   CONSTRUCTION OF LIST OF TERMINALS THAT
*                   CAN BEGIN A NON TERM
*
            HDTKN = TERMTKNCNT + 1
FSTTERMS1   GT(HDTKN, TKNCNT)    :S(FSTTERMS5)
            TKN = $(T5 HDTKN)
FSTTERMS2   IDENT( TKN, 'XX' )    :S(FSTTERMS4)
            GT(TKN, TERMTKNCNT) :S(FSTTERMS3)
```

```
            $(T7 HDTKN ≠.≠ TKN ) = $(T7 HDTKN)
            $(T7 HDTKN) = TKN
FSTTERMS3   TKN = $(T5 HDTKN ≠.≠ TKN) :(FSTTERMS2)
FSTTERMS4   HDTKN = HDTKN + 1    :(FSTTERMS1)
FSTTERMS5
*
*
*                   CONSTRUCTION OF LIST ON ONON TERMS THAT CAN
*                   END A NON TERMINAL
*
            HDTKN = TERMTKNCNT + 1
LASTNONT1   GT(HDTKN, TKNCNT )   :S(LASTNONT5)
            TKN = $(T8 HDTKN)
LASTNONT2   IDENT( TKN, ≠X≠ )    :S(LASTNONT4)
            LE( TKN, TERMTKNCNT)            :S(LASTNONT3)
            $(T9 HDTKN ≠.≠ TKN ) = $(T9 HDTKN)
            $(T9 HDTKN) = TKN
LASTNONT3   TKN = $(T8 HDTKN ≠.≠ TKN)   :(LASTNONT2)
LASTNONT4   HDTKN = HDTKN + 1    :(LASTNONT1)
LASTNONT5
*
*
*
*
*                   CONSTRUCT LIST OF TERMINALS THAT
*                   CAN FOLLOW EACH NON TERM
*
            RULE = 1
NXTTERMS1   GT(RULE, RULECNT)    :S(NXTTERMS9)
            L = $(R8 RULE)
*
NXTTERMS2   NONTERM = VALUE(L)
            L = NEXT(L)
            LE( NONTERM, TERMTKNCNT)        :S(NXTTERMS8)
*
            IDENT( L, ≠≠)          :S(NXTTERMS7)
            FOLLOWER = VALUE(L)
*
            TKN1 = $(T9 NONTERM)
NXTTERMS3   IDENT( TKN1, ≠X≠ )   :S(NXTTERMS2)
*
            TKN2 = $(T7 FOLLOWER)
NXTTERMS4   IDENT( TKN2, ≠X≠ )   :S(NXTTERMS6)
            IDENT( $(T10 TKN1 ≠.≠ TKN2 ), ≠≠ )   :F(NXTTERMS5)
            $(T10 TKN1 ≠.≠ TKN2 ) = $(T10 TKN1)
            $(T10 TKN1 ) = TKN2
*
NXTTERMS5   TKN2 = $( T7 FOLLOWER ≠.≠ TKN2 )     :(NXTTERMS4)
*
NXTTERMS6   TKN1 = $( T9 NONTERM ≠.≠ TKN1 )    :(NXTTERMS3)
*
NXTTERMS7   RULE = RULE + 1        :(NXTTERMS1)
*
NXTTERMS8   IDENT(L,≠≠)   :S(NXTTERMS7) F(NXTTERMS2)
*
NXTTERMS9
```

```
*
*
          SAVE()
*
*
*
*                    READ IN LIST OF TOKENS TO BE LOOKED UP
*                              IN THIS GRAMMAR
*
*
SPTKNS    TOKEN = NEXTTKN()    :F(SPTKNS1)
          EXTNAME = NEXTITEM()
          IDENT( EXTNAME, ≠.≠ )          :S(FAIL7)
          IDENT( NEXTITEM(), ≠.≠ )       :F(FAIL4)
          PUT( TOKEN )
          PUT( EXTNAME )       :(SPTKNS)
*
SPTKNS1   PUT( ≠...≠ )
*
          :(END)
*
*
FAIL1     OUT( SYNERR  ITEM ≠ NOT A TOKEN≠ )  :(FAIL)
FAIL2     OUT( SYNERR  ITEM ≠ NOT A FUNCTION≠ )  :(FAIL)
FAIL4     OUT( SYNERR  ≠ PERIOD EXPECTED≠ ) :(FAIL)
FAIL5     OUT( SYNERR  ≠≠ ≠:≠ EXPECTED ≠≠ )  :(FAIL)
FAIL6     OUT( SYNERR  ≠≠ ≠::=≠ EXPECTED ≠≠ )  :(FAIL)
FAIL7     OUT( SYNERR  ≠ PERIOD NOT EXPECTED≠ ) :(FAIL)
*
FAIL      OUTPUT = SYNERR
          ASSEM( ≠≠, ≠ERR≠, ≠≠, SYNERR )
*
*
*
*
*
END
END
*         DEFINE( ≠NEXTITEM(NULL)≠,≠NEXTITEM≠)
          DEFINE( ≠NEWSTATE(OLD,TKN)≠,≠NEWSTATE≠)
*         DEFINE( ≠NEXTTKN(NULL)≠,≠NEXTTKN≠)
*         DEFINE( ≠NEXTFNC(NULL)≠,≠NEXTFNC≠)
*         DEFINE( ≠EXTEND(KEY)≠,≠EXTEND≠)
          DEFINE( ≠EXTENDSS(NULL)FLAG,STATE,TKN,NSTATE,XSTATE≠ )
          DEFINE( ≠ASSEM(LOC,OP,RAND,COMM)A,B≠ )
          DEFINE( ≠UNSAVE(NULL)≠ )
          DEFINE( ≠GET(NULL)≠ )
*
*         DEFINE( ≠DUMPFNC(NULL)≠,≠DUMPFNC≠)
*         DEFINE( ≠DUMPTKN(NULL)≠,≠DUMPTKN≠)
*         DEFINE( ≠DUMPSTATET(NULL)≠,≠DUMPSTATET≠)
*         DEFINE( ≠DUMPRULES(NULL)≠,≠DUMPRULES≠)
*         DEFINE( ≠DUMPNONTERM(NULL)≠,≠DMPNONTERM≠)
          DEFINE( ≠PRINTSS(SS,FLAG)NM,S≠ )
          DEFINE( ≠PRINTRULES(NULL)LINE≠ )
          DEFINE( ≠PRINTS(S)NM,LN,T≠ )
```

```
          DEFINE( ≠PRINTTKNS(NULL)≠ )
*
*
          DATA( ≠LISTEL(VALUE,NEXT)≠ )
*
          F1 = ≠F1.≠
          F2 = ≠F2.≠
          F3 = ≠F3.≠
          T1 = ≠T1.≠
          T2 = ≠T2.≠
          T3 = ≠T3.≠
          T4 = ≠T4.≠
          T5 = ≠T5.≠
          T6 = ≠T6.≠
          T7 = ≠T7.≠
          T8 = ≠T8.≠
          T9 = ≠T9.≠
          T10 = ≠T10.≠
          T11 = ≠T11.≠
          S2 = ≠S2.≠
          S4 = ≠S4.≠
          S5 = ≠S5.≠
          R4 = ≠R4.≠
          R5 = ≠R5.≠
          R6 = ≠R6.≠
          R7 = ≠R7.≠
          R8 = ≠R8.≠
          SS1 = ≠SS1.≠
          SS2 = ≠SS2.≠
          SS4 = ≠SS4.≠
          SS5 = ≠SS5.≠
*
          DETACH( ≠INPUT≠ )
          INPUT( ≠INPUT≠, ≠GRAMMAR≠, 80 )
          OUTPUT( ≠ASSEMLINE≠, ≠SCMMDD≠, ≠≠ )
          INPUT(≠UNSAVELINE≠, ≠SAVEF≠, 80 )
          REWIND( ≠SAVEF≠ )
*
*
          SYNERR = ≠***** SYNTAX ERROR, ≠
*
*
*
          :(GO)
*
NEXTTKN   ITEM = NEXTITEM()
          IDENT( ITEM , ≠.≠ ) :S(FRETURN)
          NEXTTKN = $( T1 ITEM )
          IDENT( NEXTTKN , ≠≠ ) :F(RETURN)S(FAIL1)
*
NEXTENC   ITEM = NEXTITEM()
          IDENT( ITEM , ≠.≠ ) :S(FRETURN)
          NEXTENC = $( F1 ITEM )
          IDENT( NEXTENC , ≠≠ ) :F(RETURN)S(FAIL2)
*
NEXTITEM  LINE  SPAN(≠ ≠) BREAK(≠ ≠) . NEXTITEM = ≠≠ :S(RETURN)
```

```
NEXTITEM1 LINE = INPUT
          OUTPUT = LINE
          LINE  POS(3) *** :F(NEXTITEM)S(NEXTITEM1)
*
NEWSTATE  NEWSTATE = $( S4 OLD *.* TKN )
          IDENT( $(S4 OLD *.* TKN) , ** ) :F(RETURN)
          STATECNT = STATECNT + 1
          $(S2 STATECNT ) = $( S2 OLD) *.* TKN
          $( S4 OLD *.* TKN ) = STATECNT
          NEWSTATE = STATECNT  :(RETURN)
*
*
*                    EXTEND A LIST BY TRANSITIVITY THROUGH NON TERMINALS
*
EXTEND    TKN = TERMTKNCNT + 1
EXTEND1   GT( TKN, TKNCNT )    :S(EXTEND2)
          $(T6 TKN) = *Y*
          TKN = TKN + 1        :(EXTEND1)
*
EXTEND2   FLAG = *OFF*
          TKNHD = TERMTKNCNT + 1
*
EXTEND3   GT(TKNHD, TKNCNT )   :S(EXTEND9)
          CHKTKN = $(T6 TKNHD )
          $(T6 TKNHD) = $(KEY TKNHD)
          TKN = $(KEY TKNHD)
*
EXTEND4   IDENT( TKN, CHKTKN)  :S(EXTEND8)
          LE( TKN, TERMTKNCNT)          :S(EXTEND7)
          SUBTKN = $(KEY TKN)
*
EXTEND5   IDENT(SUBTKN, *X* ) :S(EXTEND7)
          IDENT( $(KEY TKNHD *.* SUBTKN), ** )     :F(EXTEND6)
          $(KEY TKNHD *.* SUBTKN) = $(KEY TKNHD)
          $(KEY TKNHD) = SUBTKN
          FLAG = *ON*
*
EXTEND6   SUBTKN = $(KEY TKN *.* SUBTKN)           :(EXTEND5)
*
EXTEND7   TKN = $(KEY TKNHD *.* TKN )   :(EXTEND4)
*
EXTEND8   TKNHD = TKNHD + 1   :(EXTEND3)
*
EXTEND9   IDENT( FLAG, *OFF* )          :S(RETURN)F(EXTEND2)
*
*
*                    EXTEND STATE SET REPRESENTED IN ARRAY SET
*                    BY LAMBDA MOVES
*
*                    SETS SET AND SETX EMPTY  AT END
*                              (ASSUMS S0 AT START FOR SETX )
*
*                    RETURNS NUMBER OF EXTENDED SET
*
EXTENDSS  FLAG = *OFF*
```

```
            STATE = 1
*
EXTENDSS1   GT( STATE, STATECNT) :S(EXTENDSS5)
            IDENT( SET[STATE], ** )    :S(EXTENDSS4)
            IDENT( SETX[STATE], *1* ) :S(EXTENDSS4)
            SETX[STATE] = 1
            TKN = TERMTKNCNT + 1
*
EXTENDSS2   GT(TKN, TKNCNT)       :S(EXTENDSS4)
            NSTATE = $(S4 STATE *.* TKN )
            IDENT( NSTATE, ** )     :S(EXTENDSS3)
            XSTATE = $(S4 STATE *.* TKN )
            IDENT( SET[XSTATE], 1 )  :S(EXTENDSS3)
            SET[XSTATE] = 1
            FLAG = *ON*
*
EXTENDSS3   TKN = TKN + 1          :(EXTENDSS2)
*
EXTENDSS4   STATE = STATE + 1      :(EXTENDSS1)
*
EXTENDSS5   IDENT( FLAG, *OFF* )           :F(EXTENDSS)
            STATE = 1
            SSTATE = **
*
EXTENDSS6   GT( STATE , STATECNT)  :S(EXTENDSS8)
            IDENT( SET[STATE], ** )        :S(EXTENDSS7)
            SSTATE = SSTATE *.* STATE
*
EXTENDSS7   SET[STATE] = **
            SETX[STATE] = **
            STATE = STATE + 1   :(EXTENDSS6)
*
EXTENDSS8   EXTENDSS = $( SS1  SSTATE)
            IDENT( EXTENDSS, ** )    :F(RETURN)
            SSCNT = SSCNT + 1
            EXTENDSS = SSCNT
            $( SS1 SSTATE) = SSCNT
            $(SS2 SSCNT ) = SSTATE     :(RETURN)
*
*
*                   OUTPUT AN ASSEMBLY LINE
*
*
ASSEM       A = LOC *                  *
            A TAB(1*) . B  = **
            A = B OP *                    *
            A  TAB(24) . B  = **
            IDENT( COMM, ** )    :S(ASSEM1)
            A = B RAND *                                        *
            A  TAB(56) . B  = **
            ASSEMLINE = B COMM  :(RETURN)
*
ASSEM1      ASSEMLINE = B RAND  :(RETURN)
*
*
*                   READ IN INFO FROM PASS 1
```

```
*
UNSAVE      TERMTKNCNT = GET()
            TKNCNT = GET()
            RULECNT = GET()
            FNCCNT = GET()
            STATECNT = GET()
UNSAVE1     ITEM1 = GET()
            IDENT( ITEM1 .*.*....* )  :S(UNSAVE2)
            *(ITEM1) = GET()    :(UNSAVE1)
UNSAVE2     OUT( *ITEMS IN = * ITEMCNT )   :(RETURN)
*
*
GET         GET = TRIM( UNSAVELINE )
            ITEMCNT = ITEMCNT + 1          :(RETURN)
*
*
*
*
*
*
*
*
*
*
*
*                   PRINT OUT A STATE SET
*
*
PRINTSS     NM = $(SS2 SS) *.*
PRINTSS1    NM *.* BREAK(*.*) . S = **   :F(RETURN)
            PRINTS(S)
            IDENT( FLAG, ** )    :S(PRINTSS1)
            OUT(OUTPUT )         :(PRINTSS1)
*
*
*
*                   PRINT OUT A STATE
*
PRINTS      NM = $(S2 S) *.*
            LN = *              *
PRINTS1     NM *.* BREAK(*.*) . T    = **    :F(PRINTS2)
            LN = LN * * $(T2 T)      :(PRINTS1)
PRINTS2     OUTPUT = LN   :(RETURN)
*
*
*                   PRINT OUT RULE TABLE
*
PRINTRULES              OUTPUT = **
            OUTPUT = **
            OUTPUT = *       RULE INFO*
            OUTPUT = **
            I = 1
PRINTRULES1   GT(I, RULECNT)    :S(RETURN)
            LINE = *   * I *     * $(T2 $(R4 I)) *    * $(F2 $(R5 I))
            LINE = LINE *     * $(R6 I)
            OUTPUT = LINE
```

```
            I = I + 1       :(PRINTRULES1)
*
*
*
*                       PRINT OUT TOKEN INFO
*                       AND PLACE IN ASSEMBLY ALSO
*
PRINTTKNS   OUTPUT = ≠≠
            OUTPUT = ≠≠
            OUTPUT = ≠      TOKEN DEFINITIONS≠
            OUTPUT = ≠≠
            I = 1
PRINTTKNS1  GT( I, TKNCNT )     :S(RETURN)
            TEMP = $(T2 I) ≠                                    ≠
            TEMP    TAB(24)  .  TEMP1  = ≠≠
            OUTPUT = TEMP1  $(T3 I)
            ASSEM( ≠XXX≠, ≠SET≠, $(T3 I), $(T2 I) )
            I = I + 1       :(PRINTTKNS1)
*
*
*
*
GO          UNSAVE()
*
*
*
            MVESS = ARRAY( ≠1:≠ TKNCNT )
            MVER  = ARRAY( ≠1:≠ TKNCNT )
            SET   = ARRAY( ≠1:≠ STATECNT )
            SETX  = ARRAY( ≠1:≠ STATECNT )
            SETZ  = ARRAY( ≠1:≠ STATECNT )
*
*
*
*                   GENERATE STATE SETS AND TRANSITION TABLE
*
*
*
SS0         TOKEN = GET()
            IDENT( TOKEN, ≠.≠ )             :S(SS0A)
            EXTNAME = GET()
            IDENT( EXTNAME, ≠≠ )            :S(FAILR)
            STATE = NEWSTATE( 0, TOKEN)
            SET[STATE] = 1
            SS = EXTENDSS()
            ASSEM( ≠≠, ≠ENTRY≠, EXTNAME )
            $(SS5 SS) = EXTNAME             :(SS0)
*
SS0A        OUTPUT = ≠≠
*
            OUTPUT = ≠≠
            OUTPUT = ≠≠
            OUTPUT = ≠      STATE SETS AND TRANSITION TABLE≠
*
            SS = 1
*
```

```
SS1     GT( SS, SSCNT)          :S(SSEND)
        OUTPUT = ''
        OUTPUT = ''
        SSNAME = $(SS$ CS) '.'
        SCNT = 0
*
        T = 1
SS1A    GT(T, TKNCNT)           :S(SS1B)
        MVESS[T] = ''
        MVER[T] = ''
        T = T + 1               :(SS1A)
SS1B    TYPESTKN = ''
        SKEY = ''
*
SS2     SSNAME '.' BREAK('.') . S  =  ''  :F(SS3)
        SCNT = SCNT + 1
        SETZ[SCNT] = S          :(SS2)
*
SS3     TKN = 1
*
SS4     GT( TKN, TKNCNT )       :S(SS9)
        I = 1
        TFLAG = 'OFF'
*
SS5     GT( I, SCNT)            :S(SS7)
        S = SETZ[I]
        K = $(S4 S '.' TKN )
        IDENT( K, '')           :S(SS6)
        SET[K] = 1
        TFLAG = 'ON'
*
SS6     I = I + 1 :(SS5)
*
SS7     IDENT( TFLAG, 'OFF' )           :S(SS8)
        MVESS[TKN] = EXTENDSS()
*
SS8     TKN = TKN + 1           :(SS4)
*
*
SS9     I = 1
*
SS10    GT(I, SCNT)             :S(SS13)
        S= SETZ[I]
        R = $(S5 S)
        IDENT( R, '')           :S(SS12)
        NTKN = $(R4 R)
        T = $(T1 NTKN)
*
SS11    IDENT( T, 'X')          :S(SS12)
        IDENT( MVESS[T], '' )           :S(SS11A)
*
        ASSERT( '', 'ERROR', '', 'RULE AND STATE MOVE CONFLICT' )
        OUTPUT = ''
        OUTPUT = ''
        OUTPUT = '******** ERROR, RULE AND STATE MOVE CONFLICT'
        OUT( OUTPUT )
```

```
          OUTPUT = ≠STATE SET = ≠ SS ≠     ≠ $(SS2 SS)
          OUT( OUTPUT )
          PRINTSS(SS, ≠YES≠ )
          OUTPUT = ≠TOKEN = ≠ $(T2 T)
          OUT( OUTPUT )
          OUTPUT = ≠≠
          OUTPUT = ≠≠
*
SS11A     IDENT( MVER[T1, ≠≠) :S(SS11B)
          ASSEM( ≠≠, ≠ERR≠, ≠≠, ≠2 RULES WITH SAME TOKEN≠ )
          OUTPUT = ≠≠
          OUTPUT = ≠≠
          OUTPUT = ≠******** ERROR, 2 RULES WITH SAME TOKEN≠
          OUT( OUTPUT )
          OUTPUT = ≠STATE SET = ≠ SS ≠     ≠ $(SS2 SS)
          OUT( OUTPUT )
          PRINTSS( SS, ≠YES≠ )
          OUTPUT = ≠TOKEN = ≠ $(T2 T)
          OUT( OUTPUT )
          OUTPUT = ≠≠
          OUTPUT = ≠≠
*
SS11B     MVER[T] = P
          T = $(T1A MTKN ≠,≠ T )          :(SS11)
*
*
SS12      T = I + 1 :(SS13)
*
SS13      T = 1
SS13A     GT( T, TERMTKNCNT)   :S(SS14)
          IDENT( MVESS[T], ≠≠)          :F(SS13B)
          IDENT( MVER[T1, ≠≠) :S(SS13D)
SS13B     IDENT( TYPESTKN, ≠≠)          :S(SS13C)
          IDENT( TYPESTKN, $(T4 T )) :S(SS13D)
          ASSEM( ≠≠, ≠ERR≠, ≠≠, ≠TOKEN TYPES CONFLICT≠ )
          OUTPUT = ≠≠
          OUTPUT = ≠≠
          OUTPUT = ≠******** ERROR, TOKEN TYPES CONFLICT≠
          OUT( OUTPUT )
          OUTPUT = ≠STATE SET = ≠ SS ≠     ≠ $(SS2 SS)
          OUT( OUTPUT )
          PRINTSS( SS, ≠YES≠ )
          OUTPUT = ≠KINDS OF GETTKN ARE ≠ TYPESTKN ≠ AND ≠ $(T4 T)
          OUT( OUTPUT )
          OUTPUT = ≠≠
          OUTPUT = ≠≠
                :(SS13C1)
SS13C     TYPESTKN = $(T4 T)
*
SS13C1    IDENT( SKEY, ≠≠ )    :S(SS13C2)
          IDENT( SKEY, $(T1 T) )          :S(SS13D)
          ASSEM( ≠≠, ≠ERR≠, ≠≠, ≠TOKEN SUB KEYS CONFLICT≠ )
          OUTPUT = ≠≠
          OUTPUT = ≠≠
          OUTPUT = ≠****** ERROR, TOKEN SUBKEYS CONFLICT≠
          OUT( OUTPUT )
```

```
          OUTPUT = ≠STATE SET = ≠ SS ≠    ≠ $(SS2 SS)
          OUT(OUTPUT)
          PRINTSS( SS, ≠YES≠ )
          OUTPUT = ≠KIND OF GETTKN IS ≠ TYPESTKN
          OUT(OUTPUT)
          OUTPUT = ≠SUBKEY TYPES ARE ≠ SKEY ≠ AND ≠ $(T11 T)
          OUT(OUTPUT)
          OUT(OUTPUT)
          OUTPUT = ≠≠
          OUTPUT = ≠≠
                    :(SS13D)
*
SS13C2    SKEY = $(T11 T)
SS13D     T = T + 1 :(SS13A)
*
SS14      OUTLINE = SS ≠  ≠ $(SS2 SS) ≠    ≠ TYPESTKN ≠   ≠ $(SS5 SS)
          OUTPUT = OUTLINE  ≠  ≠ SKEY
          OUTPUT = ≠≠
          PRINTSS(SS)
          ASSEM( ≠S,≠ SS, ≠VFD≠, ≠30/≠ SKEY ≠,30/≠ TYPESTKN )
          OUTPUT = ≠≠
           T = 1
SS14A     GT( T, TKNCNT )      :S(SS15)
          IDENT( MVESS[T], ≠≠) :S(SS14B)
          OUTPUT = ≠           STATE  ≠ $(T2 T) ≠   ≠ MVESS[T]
          RAND = ≠12/0,TB/S,≠ MVESS[T] ≠,18/≠ $(T3 T) ≠,12/0≠
          ASSEM( ≠,≠ , ≠VFD≠, RAND )      :(SS14C)
SS14B     IDENT( MVER[T], ≠≠)  :S(SS14C)
          OUTPUT = ≠           RULE   ≠ $(T2 T) ≠   ≠ MVER[T]
          RAND = ≠12/0,TB/R,≠ MVER[T] ≠,18/≠ $(T3 T) ≠,12/1≠
          ASSEM( ≠,≠ , ≠VFD≠, RAND )
*
SS14C     T = T + 1    :(SS14A)
*
SS15      OUTPUT = ≠           FAIL≠
          ASSEM( ≠≠, ≠DATA≠, 5 )
          IDENT( $(SS5 SS), ≠≠ )      :S(SS15B)
          ASSEM( $(SS5 SS), ≠EQU≠, ≠S,≠ SS )
*
SS15B     SS = SS + 1          :(SS1)
*
SSEND
*
          ASSEM( ≠EL≠, ≠RSS≠, ≠0≠ )
          ASSEM( ≠≠ , ≠END ≠ , ≠≠ )
*
          :(END)
*
*
FAIL1     OUT( SYNERR  ≠ITEM ≠ NOT A TOKEN≠ ) :(FAIL)
FAIL2     OUT( SYNERR  ≠ITEM ≠ NOT A FUNCTION≠ )  :(FAIL)
FAIL4     OUT( SYNERR  ≠ PERIOD EXPECTED≠ ) :(FAIL)
FAIL5     OUT( SYNERR  ≠≠ ≠:≠ EXPECTED ≠≠ )  :(FAIL)
FAIL6     OUT( SYNERR  ≠≠ ≠::=≠ EXPECTED ≠≠ )  :(FAIL)
FAIL7     OUT( SYNERR  ≠ PERIOD NOT EXPECTED≠ ) :(FAIL)
FAIL8     OUTPUT = ≠.... NOT EXPECTED ON SAVEF≠   :(FAIL)
```

```
*
FAIL        OUTPUT = SYNERR
            ASSEM( ##, #ERR#, ##, SYNERR )
*
*
*
*
*
END
      C...DU        NOEL        3000B
*
 BLANK      60B         GETTKN    .
 NMBSGL     62B         GETTKN    .
 ¶          64B         GETTKN    .
 (          108         GETTKN    .
 )          118         GETTKN    .
 *          128         GETTKN    .
 +          138         GETTKN    .
 ,          148         GETTKN    .
 -          158         GETTKN    .
 /          178         GETTKN    .
 :          328         GETTKN    .
 ;          338         GETTKN    .
 >          368         GETTKN    .
 ↑          768         GETTKN    .
 ∨          748         GETTKN    .
 CR         1558        GETTKN    .
*
 INTEGER    1608B       GETTKN    .
 IDENTIFIER 1628B       GETTKN    .
*
 REG        REG         GETKYWD   .
*
  .
*
*
 WORD.EXP               .
 WORD.PART              .
 INTEGER.EXP            .
 INTEGER.TERM           .
 INTEGER.PRIM           .
 LOC.EXP                .
 LOC.TERM               .
 LOC.PRIM               .
 PARAM                  .
 EXP                    .
 CEXP                   .
 CMMD                   .
*
  .
*
*
 IDENTITY   F.NOACT     .
 ADD        F.ADD       .
 SUB        F.SUB       .
 NEG        F.NEG       .
```

```
 MULT                    F.MULT      .
 DIV                     F.DIV       .
 WORD.PART               F.WP        .
 WORD.WORD.PART          F.WWP       .
 SCAN.LIST               F.SCNLS     .
 DIRECTORY               F.DRCTY     .
 DIRECTORY.NULLKEY       F.DRCTN     .
 OBJ.INDX                F.OBJX      .
 SUBP.INDX               F.SUBPX     .
 REG.LOC                 F.REGL      .
 VAR.LOC                 F.VARL      .
 PARAM                   F.PARAM     .
 CMMD                    F.CMMD      .
*
 .
*
*
*
 IDENTITY  :  PARAM  ::=  EXP  BLANK                          .
 IDENTITY  :  PARAM  ::=  EXP  CR                             .
*
 PARAM     :  EXP    ::=  LOC.EXP                             .
*
*
 IDENTITY  :  LOC.EXP  ::=  LOC.TERM                                    .
 OBJ.INDX  :  LOC.EXP  ::=  LOC.EXP  NMBSGN  INTEGER.EXP                .
 SUBP.INDX :  LOC.EXP  ::=          NMBSGN  INTEGER.EXP                 .
 REG.LOC   :  LOC.EXP  ::=  % REG   NMBSGN  INTEGER.EXP                 .
*
 IDENTITY        :  LOC.TERM  ::=  LOC.PRIM                             .
 SCAN.LIST       :  LOC.TERM  ::=  LOC.PRIM  >  IDENTIFIER              .
 DIRECTORY       :  LOC.TERM  ::=  LOC.TERM  :  IDENTIFIER  :  LOC.PRIM
                                                                        .
 DIRECTORY.NULLKEY : LOC.TERM ::=  LOC.TERM  :  IDENTIFIER              .
*
 IDENTITY  :  LOC.PRIM  ::=  WORD.EXP                                   .
*
 IDENTITY        :  WORD.EXP  ::=  INTEGER.EXP                          .
 WORD.PART       :  WORD.EXP  ::=  WORD.PART                            .
 WORD.WORD.PART  :  WORD.EXP  ::=  WORD.EXP  .  WORD.PART               .
*
 IDENTITY  :  WORD.PART  ::=  INTEGER.EXP  v  INTEGER.EXP               .
*
*
*
 IDENTITY : INTEGER.EXP   ::=  INTEGER.TERM                             .
 ADD      : INTEGER.EXP   ::=  INTEGER.EXP  +  INTEGER.TERM             .
 SUB      : INTEGER.EXP   ::=  INTEGER.EXP  -  INTEGER.TERM             .
 IDENTITY : INTEGER.EXP   ::=  +  INTEGER.TERM                          .
 NEG      : INTEGER.EXP   ::=  -  INTEGER.TERM                          .
*
 IDENTITY : INTEGER.TERM  ::=  INTEGER.PRIM                             .
*MULT     : INTEGER.TERM  ::=  INTEGER.TERM  *  INTEGER.PRIM            .
*DIV      : INTEGER.TERM  ::=  INTEGER.TERM  /  INTEGER.PRIM            .
*
 IDENTITY  : INTEGER.PRIM  ::=  IDENTIFIER                              .
```

```
 IDENTITY    : INTEGER.PRIM  ::=  INTEGER                                    .
 IDENTITY    : INTEGER.PRIM  ::=  (  LOC.EXP  )                              .
I
 VAR.LOC     : INTEGER.PRIM  ::=  ↑ IDENTIFIER                               .
*
*
*
 .
*
 PARAM               S.PARAM    .
*
 .
  CMMDE     FL      40000
*
 BLANK               303        GETTKN     .
 IDENT               1666R      GETTKN     .
 CR                  155R       SUBGRAM    S.PARAM    .
 PARAM               5667R      SUBGRAM    S.PARAM    .
*
 PCAP      PCAP      GETKYWD    .
 PDATA     PDATA     GETKYWD    .
 NEWV      NEWV      GETKYWD    .
 KILLV     KILLV     GETKYWD    .
 MDATA     MDATA     GETKYWD    .
 MCAP      MCAP      GETKYWD    .
 VIEW      VIEW      GETKYWD    .
 NEWDF     NEWDF     GETKYWD    .
 NEWDR     NEWDR     GETKYWD    .
 KILLDF    KILLDF    GETKYWD    .
 KILLDR    KILLDR    GETKYWD    .
 NEWBLK    NEWBLK    GETKYWD    .
 KILLBLK   KILLBLK   GETKYWD    .
 COPYBOFILE  COPYBOFILE  GETKYWD
 KILLOBJ   KILLOBJ   GETKYWD    .
 DELOWN    DELOWN    GETKYWD    .
 DELLINK   DELLINK   GETKYWD    .
 NEWKEY    NEWKEY    GETKYWD    .
 ADDKEY    ADDKEY    GETKYWD    .
 DELKEY    DELKEY    GETKYWD    .
*
 .
*
*
 SENTENCE            .
 LINE                .
*
 .
*
 IDENTITY            F.NOACT    .
 DSPCAP              F.DCAP     .
 PRNTWRD             F.PWRD     .
 PRNTWRDS            F.PWRDS    .
 NEWVAR              F.NEWV     .
 KILLVAR             F.KILLV    .
 MOVEDATA            F.MVD      .
 MOVECAP             F.MVC      .
```

```
 VIEW.STACK          F.VIEW    .
 NEWDSKFILE          F.NEWDF   .
 NEWDIR              F.NEWDR   .
 KILLDSKFILE         F.KLLDF   .
 KILLOBJ             F.KLLDR   .
 NEWBLK              F.NEWB    .
 KILLBLK             F.KILLB   .
 COPYBDFILE          F.CPYBF   .
 KILLOBJ             F.KLOBJ   .
 DELOWN              F.DLOWN   .
 DELLINK             F.DLNK    .
 NEWKEY              F.NEWKY   .
 ADDKEY              F.ADDKY   .
 DELKEY              F.DELKY   .
*
 .
*
*
 IDENTITY : LINE ::= SENTENCE CR .
*
*
 DSPCAP      : SENTENCE ::= PCAP  BLANK  PARAM                    .
 PRNTWRD     : SENTENCE ::= PDATA BLANK  PARAM                    .
 PRNTWRDS    : SENTENCE ::= PDATA BLANK  PARAM  PARAM             .
 NEWVAR      : SENTENCE ::= NEWV  BLANK  PARAM                    .
 KILLVAR     : SENTENCE ::= KILLV BLANK  PARAM                    .
 MOVECAP     : SENTENCE ::= MDATA BLANK  PARAM PARAM              .
 VIEW.STACK  : SENTENCE ::= VIEW  BLANK  PARAM                    .
 NEWDSKFILE  : SENTENCE ::= NEWDF BLANK  PARAM                    .
 NEWDIR      : SENTENCE ::= NEWDR BLANK  PARAM  PARAM  PARAM      .
 KILLDSKFILE : SENTENCE ::= KILLDF BLANK  PARAM                   .
 KILLOBJ     : SENTENCE ::= KILLDR BLANK  PARAM                   .
 NEWBLK      : SENTENCE ::= NEWBLK BLANK  PARAM                   .
 KILLBLK     : SENTENCE ::= KILLBLK  BLANK  PARAM                 .
 COPYBDFILE  : SENTENCE ::= COPYBDFILE  BLANK  IDENT  BLANK  IDENT
                                        BLANK  PARAM .
 KILLOBJ     : SENTENCE ::= KILLOBJ  BLANK  PARAM                 .
 DELOWN      : SENTENCE ::= DELOWN  BLANK  PARAM                  .
 DELLINK     : SENTENCE ::= DELLINK  BLANK  PARAM                 .
 NEWKEY      : SENTENCE ::= NEWKEY  BLANK PARAM                   .
 ADDKEY      : SENTENCE ::= ADDKEY  BLANK PARAM  PARAM  PARAM     .
 DELKEY      : SENTENCE ::= DELKEY  BLANK PARAM  PARAM            .
*
 .
*
 LINE      S.LINE    .
*
 .
*
```