```
1                       DEFINE( *NEXTITEM(NULL)*,*NEXTITEM*)
2                       DEFINE( *NEWSTATE(OLD,TKN)*,*NEWSTATE*)
3                       DEFINE( *NEXTTKN(NULL)*,*NEXTTKN*)
4                       DEFINE( *NEXTFNC(NULL)*,*NEXTFNC*)
5                       DEFINE( *EXTEND(KEY)*,*EXTEND*)
6                       DEFINE( *ASSEM(LOC,OP,RAND,COMM;A,B* )
7                       DEFINE( *SAVE(NULL)* )
8                       DEFINE( *SPUT(ITEM)TEMP* )
9                       DEFINE( *PUT(ITEM)* )
                *
                *       DEFINE( *DUMPFNC(NULL)*,*DUMPFNC*)
                *       DEFINE( *DUMPTKN(NULL)*,*DUMPTKN*)
                *       DEFINE( *DUMPSTATET(NULL)*,*DUMPSTATET*)
                *       DEFINE( *DUMPRULES(NULL)*,*DUMPRULES*)
                *       DEFINE( *DUMPNONTERM(NULL)*,*DUMPNONTERM*)
10                      DEFINE( *PRINTRULES(NULL)LINE* )
11                      DEFINE( *PRINTS(S)NM,LN,T* )
12                      DEFINE( *PRINTTKNS(NULL)* )
                *
                *
13                      DATA( *LISTEL(VALUE,NEXT)* )
                *
14                      F1 = *F1,*
15                      F2 = *F2,*
16                      F3 = *F3,*
17                      T1 = *T1,*
18                      T2 = *T2,*
19                      T3 = *T3,*
20                      T4 = *T4,*
21                      T5 = *T5,*
22                      T6 = *T6,*
23                      T7 = *T7,*
24                      T8 = *T8,*
25                      T9 = *T9,*
26                      T10 = *T10,*
27                      T11 = *T11,*
28                      S2 = *S2,*
29                      S4 = *S4,*
30                      S5 = *S5,*
31                      R4 = *R4,*
32                      R5 = *R5,*
33                      R6 = *R6,*
34                      R7 = *R7,*
35                      R8 = *R8,*
                *
36                      DETACH( *INPUT* )
37                      INPUT( *INPUT*, *GRAMMAR*, 80 )
38                      OUTPUT( *ASSEMLINE*, *SCMMND*, ** )
39                      OUTPUT( *SAVELINE*, *SAVFF*, ** )
                *
                *
40                      SYNERR = *****  SYNTAX ERROR, *
                *
                *
```

```
        *
41                  :(GO)
        *
42      NEXTTKN     ITEM = NEXTITEM()
43                  IDENT( ITEM , ≠.≠ ) :S(FRETURN)
44                  NEXTTKN = $( T1 ITEM )
45                  IDENT( NEXTTKN , ≠≠ ) :F(RETURN)S(FAIL1)
        *
46      NEXTFNC     ITEM = NEXTITEM()
47                  IDENT( ITEM , ≠.≠ ) :S(FRETURN)
48                  NEXTFNC = $( F1 ITEM )
49                  IDENT( NEXTFNC , ≠≠ ) :F(RETURN)S(FAIL2)
        *
50      NEXTITEM    LINE  SPAN(≠ ≠) BREAK(≠ ≠) . NEXTITEM = ≠≠ :S(RETURN)
51      NEXTITEM1   LINE = INPUT
52                  OUTPUT = LINE
53                  LINE  POS(0) ≠*≠    :F(NEXTITEM)S(NEXTITEM1)
        *
54      NEWSTATE    NEWSTATE = $( S4 OLD ≠.≠ TKN )
55                  IDENT( $(S4 OLD ≠.≠ TKN) , ≠≠ ) :F(RETURN)
56                  STATECNT = STATECNT + 1
57                  $(S2 STATECNT ) = $( S2 OLD) ≠.≠ TKN
58                  $( S4 OLD ≠.≠ TKN ) = STATECNT
59                  NEWSTATE = STATECNT  :(RETURN)
        *
        *
        *
        *                   EXTEND A LIST BY TRANSITIVITY THROUGH NON TERMINALS
        *
60      EXTEND      TKN = TERMTKNCNT + 1
61      EXTEND1     GT( TKN, TKNCNT )   :S(EXTEND2)
62                  $(T6 TKN) = ≠X≠
63                  TKN = TKN + 1       :(EXTEND1)
        *
64      EXTEND2     FLAG = ≠OFF≠
65                  TKNHD = TERMTKNCNT + 1
        *
66      EXTEND3     GT(TKNHD, TKNCNT )  :S(EXTEND9)
67                  CHKTKN = $(T6 TKNHD )
68                  $(T6 TKNHD) = $(KEY TKNHD)
69                  TKN = $(KEY TKNHD)
        *
70      EXTEND4     IDENT( TKN, CHKTKN)  :S(EXTEND8)
71                  LE( TKN, TERMTKNCNT)          :S(EXTEND7)
72                  SUBTKN = $(KEY TKN)
        *
73      EXTEND5     IDENT(SUBTKN, ≠X≠ ) :S(EXTEND7)
74                  IDENT( $(KEY TKNHD ≠.≠ SUBTKN), ≠≠ )    :F(EXTEND6)
75                  $(KEY TKNHD ≠.≠ SUBTKN) = $(KEY TKNHD)
76                  $(KEY TKNHD) = SUBTKN
77                  FLAG = ≠ON≠
        *
78      EXTEND6     SUBTKN = $(KEY TKN ≠.≠ SUBTKN)          :(EXTEND5)
        *
```

```
79      EXTEND7   TKN = $(KEY TKNHD '.' TKN )    :(EXTEND4)
          *
80      EXTEND8   TKNHD = TKNHD + 1    :(EXTEND3)
          *
81      EXTEND9   IDENT( FLAG, 'OFF' )          :S(RETURN)F(EXTEND2)
          *
          *
          *
          *                  OUTPUT AN ASSEMBLY LINE
          *
          *
82      ASSEM     A = LOC '                '
83                A TAB(10) . B = '  '
84                A = B OP '       '
85                A  TAB(20) . B = '  '
86                IDENT( COMM, '' )    :S(ASSEM1)
87                A = B RAND '                    '
88                A  TAB(50) . B = '  '
89                ASSEMLINE = B COMM    :(RETURN)
          *
90      ASSEM1    ASSEMLINE = B RAND    :(RETURN)
          *
          *
          *                  SUBROUTINE TO SAVE INFO FOR SECOND PASS
          *
91      SAVE      REWIND( 'SAVEF' )
92                PUT( TERMTKNCNT )
93                PUT( TKNCNT )
94                PUT(RULECNT )
95                PUT( FNCCNT )
96                PUT( STATECNT )
          *
97                I = 1
          *
98      SAVE1     GT( I, FNCCNT )     :S(SAVE2)
99                SPUT( F2 I )
100               SPUT( F3 I )
101                I = I + 1            :(SAVE1)
          *
102     SAVE2     I = 1
          *
103     SAVE3     GT( I, TERMTKNCNT)  :S(SAVE4)
104               SPUT( T2 I )
105               SPUT( T3 I )
106               SPUT( T4 I )
107               SPUT( T11 I )
108               PUT( T10 I )
109               PUT( 'X' )
110               I = I + 1      :(SAVE3)
          *
111     SAVE4     GT( I, TKNCNT )     :S(SAVE7)
112               SPUT( T2 I )
113               SPUT( T3 I )
114               SPUT( T4 I )
```

```
115                 K = TIO I
116                 SPUT( K )
117                 J = $(K)
                *
118         SAVE5   IDENT( J, "X" )         :S(SAVE6)
119                 L = K "." J
120                 SPUT(1)
121                 J = $(L)                :(SAVE5)
                *
122         SAVE6   I = I + 1               :(SAVE4)
                *
123         SAVE7   I = 1
                *
124         SAVE8   GT( I, RULECNT)         :S(SAVE9)
125                 SPUT( R4 I )
126                 SPUT( R5 I )
127                 SPUT( R6 I )
128                 I = I + 1               :(SAVE8)
                *
129         SAVE9   I = 0
                *
130         SAVE10  GT( I, STATECNT )       :S(SAVE13)
131                 SPUT( S2 I )
132                 SPUT( S4 I )
133                 SPUT( S5 I )
                *
134                 J = 1
                *
135         SAVE11  GT(J, TKNCNT)           :S(SAVE12)
136                 SPUT( S4 I "." J )
137                 J = J + 1               :(SAVE11)
                *
138         SAVE12  I = I + 1               :(SAVE10)
                *
139         SAVE13  PUT( "........" )
140                 OUT( "ITEMS OUT =" ITEMCNT )        :(RETURN)
                *
                *
                *                   ROUTINES USED BY SAVE
                *
                *
141         PUT     SAVELINE = ITEM
142                 ITEMCNT = ITEMCNT + 1           :(RETURN)
                *
                *
143         SPUT    TEMP = $(ITEM)
144                 IDENT( TEMP, "" )       :S(RETURN)
145                 PUT( ITEM)
146                 PUT( TEMP )             :(RETURN)
                *
                *
                *
                *
                *
```

```
*
*
*
*
*
*
*
*
*
*                    PRINT OUT A STATE
*
147     PRINTS     NM = $(S2 S) '.'
148                LN = '  '
149     PRINTS1    NM '.' BREAK('.') . T    = ''    :F(PRINTS2)
150                LN = LN '  ' $(T2 T)     :(PRINTS1)
151     PRINTS2    OUTPUT = LN   :(RETURN)
*
*
*                    PRINT OUT RULE TABLE
*
152     PRINTRULES         OUTPUT = ''
153                OUTPUT = ''
154                OUTPUT = '       RULE INFO'
155                OUTPUT = ''
156                I = 1
157     PRINTRULES1    GT(I, RULECNT)    :S(RETURN)
158                LINE = '  ' I '    ' $(T2 $(R4 I)) '    ' $(F2 $(R5 I))
159                LINE = LINE '    ' $(R6 I)
160                OUTPUT = LINE
161                I = I + 1     :(PRINTRULES1)
*
*
*
*                    PRINT OUT TOKEN INFO
*                    AND PLACE IN ASSEMBLY ALSO
*
162     PRINTTKNS  OUTPUT = ''
163                OUTPUT = ''
164                OUTPUT = '     TOKEN DEFINITIONS'
165                OUTPUT = ''
166                I = 1
167     PRINTTKNS1    GT( I, TKNCNT )    :S(RETURN)
168                TEMP = $(T2 I) '                              '
169                TEMP   TAB(24)  .  TEMP1  = ''
170                OUTPUT = TEMP1  $(T3 I)
171                ASSEM( 'XXX', 'SET', $(T3 I), $(T2 I) )
172                I = I + 1      :(PRINTTKNS1)
*
*
*
*
*
*
*
```

```
                *                       ACTUAL CODE STARTS HERE
                *
                *                       FIRST READ IN TOKEN DEFINITIONS, TERMINAL FIRST
                *
                *
                *
173             GO              ASSEM( ##, #IDENT#, NEXTITEM() )
174                             ASSEM( ##, #EXT#, #SUBGRAM# )
175                             ASSEM( ##, #EXT#, #GETTKN# )
176                             ASSEM( ##, #EXT#, #GETKYWD# )
177                             ITEM= NEXTITEM()
178                             IDENT( ITEM, #FL# ) :F(G01)
                *
179                             ASSEM( ##, #ENTRY#, #FL# )
                *
180             G01             KEYBASE = NEXTITEM()
                *
181                             TKNCNT = 0
182                             ASSEM( #KEYWDL#, #BSS#, 0 )
                *
183             TKNS1           ITEM1 = NEXTITEM()
184                             IDENT( ITEM1, #.# ) :S(TKNS2)
185                             ITEM2 = NEXTITEM()
186                             IDENT( ITEM2, #.# )                     :S(FAILE7)
187                             ITEM3 = NEXTITEM()
188                             IDENT( ITEM3, #.# )                     :S(FAIL7)
189                             TKNCNT = TKNCNT + 1
190                             IDENT( ITEM3, #GETKYWD# )               :F(TKNS1A)
                *
191                             KEYCNT = KEYCNT + 1
192                             ASSEM( ##, #DATA#, #0L# ITEM2 )
193                             ASSEM( ##, #VFD#, #6,#KEYBASE #.# KEYCNT )
194                             ITEM2 = KEYBASE #.# KEYCNT
195                             $(T1 TKNCNT ) = #KEYWDL#        :(TKNS1C)
                *
196             TKNS1A          IDENT( ITEM3, #SUBGRAM# )               :F(TKNS1B)
                *
197                             ITEM4 = NEXTITEM()
198                             IDENT( ITEM4, #.# )                     :S(FAIL7)
199                             ASSEM( ##, #EXT#, ITEM4 )
200                             $(T1 TKNCNT ) = ITEM4                   :(TKNS1C)
                *
201             TKNS1B          $(T1 TKNCNT) = 0        :(TKNS1C)
                *
202             TKNS1C          IDENT( NEXTITEM() , #.# )               :F(FAIL4)
203                             $(T1 ITEM1) = TKNCNT
204                             $(T2 TKNCNT) = ITEM1
205                             $(T3 TKNCNT) = ITEM2
206                             $(T4 TKNCNT) = ITEM3
207                             $(T7 TKNCNT) = TKNCNT
208                             $(T7 TKNCNT #.# TKNCNT ) = #X#          :(TKNS1)
                *
                *
                *                       NOW WE READ IN NON TERMINAL TOKEN DEFINITIONS
```

```
                *
209     TKNS2   TERMTKNCNT = TKNCNT
210             ASSEM( ##, #DATA#, 0 )
                *
211     TKNS3   ITEM1 = NEXTITEM()
212             IDENT( ITEM1, #.#)  :S(FNCS0)
213             ITEM2 = #2000B+# ( TKNCNT + 1 - TERMTKNCNT )
214             IDENT( NEXTITEM() , #.# )       :F(FAIL4)
215             TKNCNT = TKNCNT + 1
216             $(T1 ITEM1) = TKNCNT
217             $(T2 TKNCNT) = ITEM1
218             $(T3 TKNCNT) = ITEM2
219             $(T5 TKNCNT) = #X#
220             $(T7 TKNCNT) = #X#
221             $(T8 TKNCNT) = TKNCNT
222             $(T8 TKNCNT #.# TKNCNT) = #X#
223             $(T9 TKNCNT) = #X#
224             $(T10 TKNCNT) = #X#   :(TKNS3)
                *
                *
                *
                *                   NOW READ IN FUNCTION DEFINITIONS
                *
225     FNCS0   FNCCNT = 0
                *
226     FNCS1   ITEM1 = NEXTITEM()
227             IDENT(ITEM1, #.#)    :S(FNCS2)
228             ITEM2 = NEXTITEM()
229             IDENT( NEXTITEM(), #.# )        :F(FAIL4)
230             FNCCNT = FNCCNT + 1
231             ASSEM( ##, #EXT#, ITEM2 )
232             $(F1 ITEM1) = FNCCNT
233             $(F2 FNCCNT) = ITEM1
234             $(F3 FNCCNT) = ITEM2            :(FNCS1)
                *
235     FNCS2
                *
                *
                *
                *                   THIS CODE READS IN THE RULES
                *                   AND GENERATES THE SYMPLE STATE TABLE
                *
                *
236             STATECNT = 0
237             RULECNT = 0
                *
238             DMYEL = LISTEL( ##, ## )
                *
239     RULE1   RULEFNC = NEXTFNC()   :F(RULE4)
240             IDENT( NEXTITEM(), #:# )   :F(FAIL5)
241             RULETKN = NEXTTKN()     :F(FAIL)
242             IDENT( NEXTITEM(), #:=# )   :F(FAIL6)
243             RULECNT = RULECNT + 1
244             $(R4 RULECNT) = RULETKN
```

```
245                     $(R5 RULECNT) = RULEFNC
246                     STATE = 0
247                     STATE = NEWSTATE(STATE,RULETKN)
248                     RULESIZE = 0
249                     EL = DMYEL
250                     NEXT(EL) = ##
        *
251     RULE2           TKN = NEXTTKN()  :F(FAIL)
252                     IDENT( $(T5 RULETKN #,# TKN), ## ) :F(RULE3)
253                     $(T5 RULETKN #,# TKN) = $(T5 RULETKN)
254                     $(T5 RULETKN) = TKN
        *
255     RULE3           STATE = NEWSTATE(STATE,TKN)
256                     NEXT(EL) = LISTEL(TKN,##)
257                     EL = NEXT(EL)
258                     RULESIZE = RULESIZE + 1
259                     LASTTKN = TKN
260                     TKN = NEXTTKN()       :S(RULE3)
        *
261                     IDENT( $(T8 RULETKN #,# LASTTKN), ## )  :F(RULE3A)
262                     $(T8 RULETKN #,# LASTTKN) = $(T8 RULETKN)
263                     $(T8 RULETKN) = LASTTKN
        *
264     RULE3A          $(S5 STATE) = RULECNT
265                     $(R6 RULECNT) = RULESIZE
266                     $(R7 RULECNT) = STATE
267                     $(R8 RULECNT) = NEXT(DMYFL)
        *
268                     RAND = #12/0,18/# $(F3 RULEFNC ) #,12/# RULESIZE
269                     RAND = RAND #,18/# $(T3 RULETKN )
270                     ASSEM( #R,# RULECNT , #VFD#, RAND )       :(RULE1)
        *
271     RULE4
        *
272                     PRINTTKNS()
273                     PRINTRULES()
274                     OUTPUT = ##
275                     OUTPUT = ##
276                     OUTPUT = ##
        *
        *
        *
277                     EXTEND(T5)
        *
278                     EXTEND(T8)
        *
        *
        *
        *                       CONSTRUCTION OF LIST OF TERMINALS THAT
        *                       CAN BEGIN A NON TERM
        *
279                     HDTKN = TERMTKNCNT + 1
280     FSTTERMS1       GT(HDTKN, TKNCNT)     :S(FSTTERMS2)
281                     TKN = $(T5 HDTKN)
```

```
282     FSTTERMS2 IDENT( TKN, *X* )    :S(FSTTERMS4)
283               GT(TKN, TERMTKNCNT)  :S(FSTTERMS3)
284               $(T7 HDTKN *.* TKN ) = $(T7 HDTKN)
285               $(T7 HDTKN) = TKN
286     FSTTERMS3 TKN = $(T5 HDTKN *.* TKN)  :(FSTTERMS2)
287     FSTTERMS4 HDTKN = HDTKN + 1    :(FSTTERMS1)
288     FSTTERMS5
        *
        *
        *                    CONSTRUCTION OF LIST OF NONTERMS THAT CAN
        *                    END A NONTERMINAL
        *
289               HDTKN = TERMTKNCNT + 1
290     LASTNONT1 GT(HDTKN, TKNCNT )   :S(LASTNONT5)
291               TKN = $(T8 HDTKN)
292     LASTNONT2 IDENT( TKN, *X* )    :S(LASTNONT4)
293               LE( TKN, TERMTKNCNT)           :S(LASTNONT3)
294               $(T9 HDTKN *.* TKN ) = $(T9 HDTKN)
295               $(T9 HDTKN) = TKN
296     LASTNONT3 TKN = $(T8 HDTKN *.* TKN)  :(LASTNONT2)
297     LASTNONT4 HDTKN = HDTKN + 1    :(LASTNONT1)
298     LASTNONT5
        *
        *
        *
        *                    CONSTRUCT LIST OF TERMINALS THAT
        *                    CAN FOLLOW EACH NONTERM
        *
299               RULE = 1
300     NXTTERMS1 GT(RULE, RULECNT)    :S(NXTTERMS8)
301               L = $(R0 RULE)
        *
302     NXTTERMS2 NONTERM = VALUE(L)
303               L = NEXT(L)
304               LE( NONTERM, TERMTKNCNT)       :S(NXTTERMS8)
        *
305               IDENT( L, *X*)       :S(NXTTERMS7)
306               FOLLOWER = VALUE(L)
        *
307               TKN1 = $(T9 NONTERM)
308     NXTTERMS3 IDENT( TKN1, *X* )   :S(NXTTERMS5)
        *
309               TKN2 = $(T7 FOLLOWER)
310     NXTTERMS4 IDENT( TKN2, *X* )   :S(NXTTERMS6)
311               IDENT( $(T10 TKN1 *.* TKN2 ), *X* )  :F(NXTTERMS5)
312               $(T10 TKN1 *.* TKN2 ) = $(T10 TKN1)
313               $(T10 TKN1 ) = TKN2
        *
314     NXTTERMS5 TKN2 = $( T7 FOLLOWER *.* TKN2 )     :(NXTTERMS4)
        *
315     NXTTERMS6 TKN1 = $( T9 NONTERM *.* TKN1 )    :(NXTTERMS3)
        *
316     NXTTERMS7 RULE = RULE + 1      :(NXTTERMS1)
```

```
                *
17              NXTTERMS8  IDENT(L,'##')  :S(NXTTERMS7) F(NXTTERMS2)
                *
18              NXTTERMS0
                *
                *
19                         SAVE()
                *
                *
                *
                *                    READ IN LIST OF TOKENS TO BE LOOKED UP
                *                          IN THIS GRAMMAR
                *
                *
20              SPTKNS     TOKEN = NEXTTKN()    :F(SPTKNS1)
21                         EXTNAME = NEXTITEM()
22                         IDENT( EXTNAME, '.' )          :S(FAIL7)
23                         IDENT( NEXTITEM(), '.' )       :F(FAIL4)
24                         PUT( TOKEN )
25                         PUT( EXTNAME )       :(SPTKNS)
                *
26              SPTKNS1    PUT( '....' )
                *
27                         :(END)
                *
                *
28              FAIL1      OUT( SYNERR  ITEM '' NOT A TOKEN'' ) :(FAIL)
29              FAIL2      OUT( SYNERR  ITEM '' NOT A FUNCTION'' )  :(FAIL)
30              FAIL4      OUT( SYNERR  '' PERIOD EXPECTED'' ) :(FAIL)
31              FAIL5      OUT( SYNERR  ''' '' EXPECTED ''' )  :(FAIL)
32              FAIL6      OUT( SYNERR  ''' :'' EXPECTED ''' )  :(FAIL)
33              FAIL7      OUT( SYNERR  '' PERIOD NOT EXPECTED'' ) :(FAIL)
                *
34              FAIL       OUTPUT = SYNERR
35                         ASSEM( '', 'ERR', '', SYNERR )
                *
                *
                *
                *
                *
36              END
37              END

SUCCESSFUL COMPILATION
     CMMDD      NOFL       3000B
*
BLANK     00B       GETTKN    .
VMBSGN    03B       GETTKN    .
$         04B       GETTKN    .
(         10B       GETTKN    .
)         11B       GETTKN    .
*         12B       GETTKN    .
+         13B       GETTKN    .
,         14B       GETTKN    .
-         15B       GETTKN    .
/         17B       GETTKN    .
:         32B       GETTKN    .
;         33B       GETTKN    .
```

```
v          74B     GETTKN  .
CR         155B    GETTKN  .

INTEGER            1001B   GETTKN  .
IDENTIFIER         1002B   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   NUMBSGN  INTEGER.EXP           .
SUBP.INDX  :  LOC.EXP   ::=            NUMBSGN  INTEGER.EXP           .
REG.LOC    :  LOC.EXP   ::=  $ REG     NUMBSGN  INTEGER.EXP           .

IDENTITY        :  LOC.TERM  ::=  LOC.PRIM                             .
SCAN.LIST       :  LOC.TERM  ::=  LOC.PRIM  >  IDENTIFIER              .
DIRECTORY       :  LOC.TERM  ::=  LOC.TERM  :  IDENTIFIER  :  LOC.PRIM
                                                                       .
```

```
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  )

VAR.LOC  :  INTEGER.PRIM  ::=  ↑ IDENTIFIER
```

TOKEN DEFINITIONS

```
BLANK          60B
ABSGN          63B
               64B
               10B
               11B
               12B
               13B
               14B
               15B
               17B
               32B
               33B
               36B
               76B
               74B
R              155B
INTEGER        1001B
IDENTIFIER     1002B
NEG            3000B+1
WORD.EXP       2000B+1
WORD.PART      2000B+2
INTEGER.EXP    2000B+3
INTEGER.TERM   2000B+4
INTEGER.PRIM   2000B+5
LOC.EXP        2000B+6
LOC.TERM       2000B+7
LOC.PRIM       2000B+8
PARAM          2000B+9
EXP            2000B+10
LEXP           2000B+11
COMMD          2000B+12
```

RULE INFO

```
1     PARAM    IDENTITY     2
2     PARAM    IDENTITY     2
3     EXP    PARAM     1
4     LOC.EXP    IDENTITY     1
5     LOC.EXP    OBJ.INDX     3
6     LOC.EXP    SUBP.INDX     2
7     LOC.EXP    REG.LOC     4
8     LOC.TERM     IDENTITY     1
9     LOC.TERM     SCAN.LIST     3
10     LOC.TERM     DIRECTORY     5
11     LOC.TERM     DIRECTORY.NULLKEY     3
12     LOC.PRIM     IDENTITY     1
13     WORD.EXP     IDENTITY     1
14     WORD.EXP     WORD.PART     1
15     WORD.EXP     WORD.WORD.PART     3
16     WORD.PART     IDENTITY     3
17     INTEGER.EXP     IDENTITY     1
18     INTEGER.EXP     ADD     3
19     INTEGER.EXP     SUB     3
20     INTEGER.EXP     IDENTITY     2
21     INTEGER.EXP     NEG     2
22     INTEGER.TERM     IDENTITY     1
23     INTEGER.PRIM     IDENTITY     1
24     INTEGER.PRIM     IDENTITY     1
25     INTEGER.PRIM     IDENTITY     3
26     INTEGER.PRIM     VAR.LOC     2
```

PARAM S.PARAM