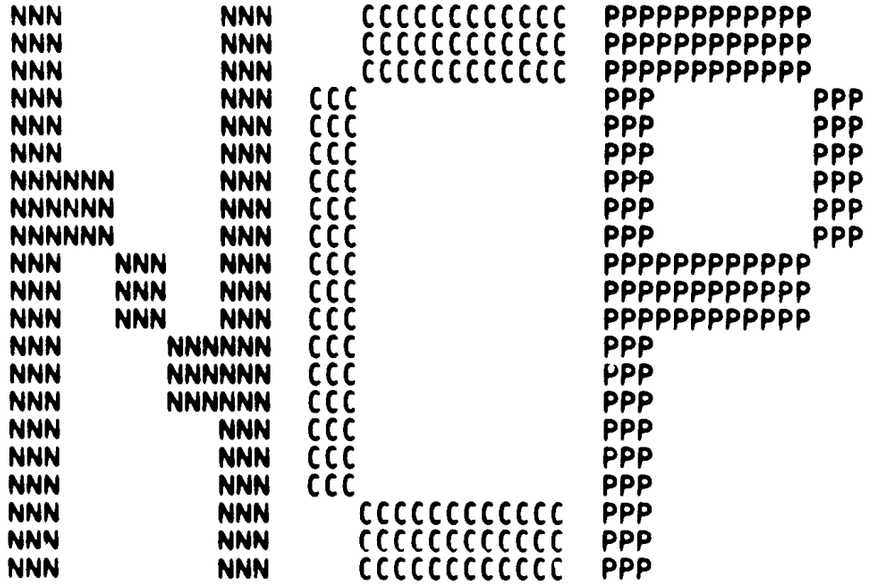
NNN NNN CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN CCC PPP PPP
NNN NNN CCC PPP PPP
NNN NNN CCC PPP PPP
NNNNNN NNN CCC PPP PPP
NNNNNN NNN CCC PPP PPP
NNNNNN NNN CCC PPP PPP
NNN NNN NNN CCC PPPPPPPPPPPP
NNN NNN NNN CCC PPPPPPPPPPPP
NNN NNN NNN CCC PPPPPPPPPPPP
NNN NNNNNN CCC PPP
NNN NNNNNN CCC PPP
NNN NNNNNN CCC PPP
NNN NNN CCC PPP
NNN NNN CCC PPP
NNN NNN CCC PPP
NNN NNN CCCCCCCCCCC PPP
NNN NNN CCCCCCCCCCC PPP
NNN NNN CCCCCCCCCCC PPP

FILEID**NCPSTAVRB

```
NN       NN    CCCCCCCC  PPPPPPPP     SSSSSSSS  TTTTTTTTTT    AAAAAA    VV      VV  RRRRRRRR    BBBBBBBB
NN       NN    CCCCCCCC  PPPPPPPP     SSSSSSSS  TTTTTTTTTT    AAAAAA    VV      VV  RRRRRRRR    BBBBBBBB
NN       NN  CC          PP      PP  SS            TT       AA      AA  VV      VV  RR      RR  BB      BB
NN       NN  CC          PP      PP  SS            TT       AA      AA  VV      VV  RR      RR  BB      BB
NNNN     NN  CC          PP      PP  SS            TT       AA      AA  VV      VV  RR      RR  BB      BB
NNNN     NN  CC          PP      PP  SS            TT       AA      AA  VV      VV  RR      RR  BB      BB
NN  NN   NN  CC          PPPPPPPP      SSSSSS       TT       AA      AA  VV      VV  RRRRRRRR    BBBBBBBB
NN  NN   NN  CC          PPPPPPPP      SSSSSS       TT       AA      AA  VV      VV  RRRRRRRR    BBBBBBBB
NN    NNNN  CC          PP                  SS      TT       AAAAAAAAAA  VV      VV  RR  RR      BB      BB
NN    NNNN  CC          PP                  SS      TT       AAAAAAAAAA  VV      VV  RR  RR      BB      BB
NN      NN  CC          PP                  SS      TT       AA      AA    VV  VV    RR    RR    BB      BB      ....
NN      NN  CC          PP                  SS      TT       AA      AA    VV  VV    RR    RR    BB      BB      ....
NN      NN    CCCCCCCC  PP          SSSSSSSS        TT       AA      AA      VV      RR      RR  BBBBBBBB        ....
NN      NN    CCCCCCCC  PP          SSSSSSSS        TT       AA      AA      VV      RR      RR  BBBBBBBB        ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

```
;  1          0001  0 %TITLE  'Verb Parse States and Data'
;  2          0002  0 MODULE NCPSTAVRB (IDENT = 'V04-000', LIST(NOOBJECT)) =
;  3          0003  1 BEGIN
;  4          0004  1
;  5          0005  1 !
;  6          0006  1 !*****************************************************************************
;  7          0007  1 !*                                                                           *
;  8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;  9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
; 10          0010  1 !*  ALL RIGHTS RESERVED.                                                     *
; 11          0011  1 !*                                                                           *
; 12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
; 13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
; 14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
; 15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
; 16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
; 17          0017  1 !*  TRANSFERRED.                                                             *
; 18          0018  1 !*                                                                           *
; 19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
; 20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
; 21          0021  1 !*  CORPORATION.                                                             *
; 22          0022  1 !*                                                                           *
; 23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
; 24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
; 25          0025  1 !*                                                                           *
; 26          0026  1 !*                                                                           *
; 27          0027  1 !*****************************************************************************
; 28          0028  1 !
; 29          0029  1
; 30          0030  1 !++
; 31          0031  1 ! FACILITY:     Network Control Program (NCP)
; 32          0032  1 !
; 33          0033  1 ! ABSTRACT:
; 34          0034  1 !
; 35          0035  1 !       States and data for the parsing of NCP command verbs
; 36          0036  1 !
; 37          0037  1 ! ENVIRONMENT:  VAX/VMS Operating System
; 38          0038  1 !
; 39          0039  1 ! AUTHOR:       Darrell Duffy   , CREATION DATE: 10-September-79
; 40          0040  1 !
; 41          0041  1 ! MODIFIED BY:
; 42          0042  1 !
; 43          0043  1 !       V03-008 PRD0099         Paul R. DeStefano       01-May-1984
; 44          0044  1 !               Set the ACT$GL_NO_XAREA_Q flag if command is TELL
; 45          0045  1 !               or SET EXEC and no area address is specified.
; 46          0046  1 !               ACT$SAVPRM (in module NCPVRBACT) will key off this
; 47          0047  1 !               flag and the fact that the parameter block address
; 48          0048  1 !               will be PBK$G_VRB_XID and set the area to 1.
; 49          0049  1 !
; 50          0050  1 !       V03-007 PRD0064         Paul R. DeStefano       05-Feb-1984
; 51          0051  1 !               Change ACT$GL_NOADR_Q references to ACT$GL_ADR_Q.
; 52          0052  1 !
; 53          0053  1 !       V03-006 PRD0062         Paul R. DeStefano       05-Feb-1984
; 54          0054  1 !               Allow OBJECT parameter to accept both name and number.
; 55          0055  1 !
; 56          0056  1 !       V03-005 PRD0055         Paul R. DeStefano       05-Feb-1984
; 57          0057  1 !               Enable X25-Access parsing.
```

```
;    58          0058  1 ;
;    59          0059  1 ;       V03-004 RPG0004         Bob Grosso              24-Mar-1983
;    60          0060  1 ;               Add CONNECT CONSOLE.
;    61          0061  1 ;
;    62          0062  1 ;       V03-003 RPG0003         Bob Grosso              24-Sep-1982
;    63          0063  1 ;               Parse for node area.
;    64          0064  1 ;               Set/Def Module configurator, console, loader, looper.
;    65          0065  1 ;
;    66          0066  1 ;       V03-002 RPG0002         Bob Grosso              14-Sep-1982
;    67          0067  1 ;               Fix prompting so X-S is ambiguous.
;    68          0068  1 ;               Clear ncp$gl_qualprs so that ALL checking works.
;    69          0069  1 ;               Make X25-P a noise word.
;    70          0070  1 ;               Clear ncp$gl_noparms so that parameter can be turned off.
;    71          0071  1 ;
;    72          0072  1 ;       V03-001 RPG0001         Bob Grosso              27-Jul-1982
;    73          0073  1 ;               Add SET X25-TRACE and SET X29-SERVER
;    74          0074  1 ;
;    75          0075  1 ;       V003    TMH0003         Tim Halvorsen           20-Jan-1982
;    76          0076  1 ;               Fix prompt for object name to reflect an increase
;    77          0077  1 ;               in its maximum size (now 12 characters).
;    78          0078  1 ;
;    79          0079  1 ;       V002    TMH0002         Tim Halvorsen           10-Jul-1981
;    80          0080  1 ;               Add MODULE entity for SET/DEFINE.
;    81          0081  1 ;
;    82          0082  1 ;       V001    TMH0001         Tim Halvorsen           18-Jun-1981
;    83          0083  1 ;               Add CIRCUITS and misc. parameters.
;    84          0084  1 ;--
```

```
;    86          0085  1 %SBTTL  'Definitions'
;    87          0086  1
;    88          0087  1 !
;    89          0088  1 ! INCLUDE FILES:
;    90          0089  1 !
;    91          0090  1
;    92          0091  1         LIBRARY 'LIB$:NMALIBRY';
;    93          0092  1         LIBRARY 'LIB$:NCPLIBRY';
;    94          0093  1         LIBRARY 'SYS$LIBRARY:TPAMAC';
;    95          0094  1
;    96          0095  1 !
;    97          0096  1 ! OWN STORAGE:
;    98          0097  1 !
;    99          0098  1
;   100          0099  1 GLOBAL
;   101          0100  1
;   102          0101  1         NCP$GL_OPTION,                          ! Place to build option
;   103          0102  1         NCP$GL_FNC_CODE.                        ! Place to build function code
;   104          0103  1
;   105          0104  1         ACT$GL_ACC_MASK,                        ! Mask for access parsing
;   106          0105  1         ACT$GL_XIDACC_Q,                        ! Flag for access control with Node specification
;   107          0106  1         ACT$GL_NO_XAREA_Q,                      ! Flag for no exec area specified.
;   108          0107  1
;   109          0108  1 !
;   110          0109  1 !       String descriptors for access parameters
;   111          0110  1 !
;   112          0111  1         ACT$GQ_ACCUSR_DSC : VECTOR [2], ! User id
;   113          0112  1         ACT$GQ_ACCACC_DSC : VECTOR [2], ! Account
;   114          0113  1         ACT$GQ_ACCPSW_DSC : VECTOR [2], ! Password
;   115          0114  1
;   116          0115  1         ACT$GQ_NODEID_DSC : VECTOR [2]  ! Node id descriptor
;   117          0116  1         ;
;   118          0117  1
```

```
;  120     0118  1
;  121     0119  1 !
;  122     0120  1 ! EXTERNAL REFERENCES:
;  123     0121  1 !
;  124     0122  1
;  125     0123  1 EXTERNAL ROUTINE
;  126     0124  1         ACT$INV_COMMAND,        ! Signal invalid command
;  127     0125  1         ACT$SAVPRM,             ! Save a parameter
;  128     0126  1         ACT$TMPSTR,             ! Save a temporary string
;  129     0127  1         ACT$BLNK_SIG,           ! Blanks are now significant
;  130     0128  1         ACT$BLNK_NSIG,          ! Blanks are not significant
;  131     0129  1         ACT$ZAPTMPDSC,          ! Clear temporary descriptors
;  132     0130  1         ACT$PRMPT,              ! Prompt for a parameter
;  133     0131  1         ACT$NUM_RNG,            ! Validate a number
;  134     0132  1         ACT$STR_LEN,            ! Validate a string length
;  135     0133  1         ACT$NXT_STATE,          ! Set vector to next state table
;  136     0134  1         ACT$PMT_ON,             ! Enable prompting
;  137     0135  1         ACT$PMT_OFF,            ! Disable prompting
;  138     0136  1         ACT$PMT_Q,              ! Control prompting
;  139     0137  1         ACT$CLRLONG,            ! Clear a longword
;  140     0138  1         ACT$VRB_EXIT,           ! Return control to VMS
;  141     0139  1         ACT$VRB_TELL,           ! Perform tell function
;  142     0140  1         ACT$VRB_SETEXEC,        ! Set executor node
;  143     0141  1         ACT$VRB_CLEXEC,         ! Clear executor node
;  144     0142  1         ACT$HELP;               ! Print help text
;  145     0143  1
;  146     0144  1 !
;  147     0145  1 !       External Data
;  148     0146  1 !
;  149     0147  1
;  150     0148  1 EXTERNAL
;  151     0149  1         ACT$GL_ADR_Q,           ! True for node address, object number
;  152     0150  1         ACT$GL_NODAREA,         ! Node area
;  153     0151  1         NCP$GL_QUALPRS,         ! Set when a qualifier is parsed
;  154     0152  1         NCP$GL_NOPARMS;         ! Set when an entity does not take parameters
;  155     0153  1
;  156     0154  1 !
;  157     0155  1 !       Error status values
;  158     0156  1 !
;  159     0157  1
;  160     0158  1 EXTERNAL LITERAL
;  161     0159  1         NCP$_INVVAL,            ! Invalid value
;  162     0160  1         NCP$_INVKEY             ! Invalid keyword
;  163     0161  1         ;
;  164     0162  1
```

```
;  166     0163  1
;  167     0164  1 !
;  168     0165  1 !       External state tables
;  169     0166  1 !
;  170     0167  1
;  171     0168  1 EXTERNAL
;  172     0169  1         NCP$G_STTBL_CLPU,                ! Clear and Purge commands
;  173     0170  1         NCP$G_KYTBL_CLPU,
;  174     0171  1         NCP$G_STTBL_CON,                 ! Connect command
;  175     0172  1         NCP$G_KYTBL_CON,
;  176     0173  1         NCP$G_STTBL_DIS,                 ! Disconnect command
;  177     0174  1         NCP$G_KYTBL_DIS,
;  178     0175  1         NCP$G_STTBL_DUM,                 ! Dump command
;  179     0176  1         NCP$G_KYTBL_DUM,
;  180     0177  1         NCP$G_STTBL_LIN,                 ! Line parameters
;  181     0178  1         NCP$G_KYTBL_LIN,
;  182     0179  1         NCP$G_STTBL_CIR,                 ! Circuit parameters
;  183     0180  1         NCP$G_KYTBL_CIR,
;  184     0181  1         NCP$G_STTBL_MODCNF,              ! Module Configurator parameters
;  185     0182  1         NCP$G_KYTBL_MODCNF,
;  186     0183  1         NCP$G_STTBL_MODCNS,              ! Module Console parameters
;  187     0184  1         NCP$G_KYTBL_MODCNS,
;  188     0185  1         NCP$G_STTBL_MODLOA,              ! Module Loader parameters
;  189     0186  1         NCP$G_KYTBL_MODLOA,
;  190     0187  1         NCP$G_STTBL_MODLOO,              ! Module Looper parameters
;  191     0188  1         NCP$G_KYTBL_MODLOO,
;  192     0189  1         NCP$G_STTBL_MAC,                 ! Module X25-ACCESS parameters
;  193     0190  1         NCP$G_KYTBL_MAC,
;  194     0191  1         NCP$G_STTBL_MPR,                 ! Module X25-PROTOCOL parameters
;  195     0192  1         NCP$G_KYTBL_MPR,
;  196     0193  1         NCP$G_STTBL_MPRDTE,              ! Module X25-PROTOCOL DTE parameters
;  197     0194  1         NCP$G_KYTBL_MPRDTE,
;  198     0195  1         NCP$G_STTBL_MPRGRP,              ! Module X25-PROTOCOL GROUP parameters
;  199     0196  1         NCP$G_KYTBL_MPRGRP,
;  200     0197  1         NCP$G_STTBL_MSE,                 ! Module X25-SERVER parameters
;  201     0198  1         NCP$G_KYTBL_MSE,
;  202     0199  1         NCP$G_STTBL_MTR,                 ! Module X25-TRACE parameters
;  203     0200  1         NCP$G_KYTBL_MTR,
;  204     0201  1         NCP$G_STTBL_MTRTPT,              ! Module X25-TRACE TRACEPOINT parameters
;  205     0202  1         NCP$G_KYTBL_MTRTPT,
;  206     0203  1         NCP$G_STTBL_M9S,                 ! Module X29-SERVER parameters
;  207     0204  1         NCP$G_KYTBL_M9S,
;  208     0205  1         NCP$G_STTBL_LOA,                 ! Load command
;  209     0206  1         NCP$G_KYTBL_LOA,
;  210     0207  1         NCP$G_STTBL_LOG,                 ! Logging parameters
;  211     0208  1         NCP$G_KYTBL_LOG,
;  212     0209  1         NCP$G_STTBL_LOO,                 ! Loop command
;  213     0210  1         NCP$G_KYTBL_LOO,
;  214     0211  1         NCP$G_STTBL_NOD,                 ! Remote node parameters
;  215     0212  1         NCP$G_KYTBL_NOD,
;  216     0213  1         NCP$G_STTBL_OBJ,                 ! Object parameters
;  217     0214  1         NCP$G_KYTBL_OBJ,
;  218     0215  1         NCP$G_STTBL_SHL,                 ! Show and List commands
;  219     0216  1         NCP$G_KYTBL_SHL,
;  220     0217  1         NCP$G_STTBL_TRI,                 ! Trigger command
;  221     0218  1         NCP$G_KYTBL_TRI,
;  222     0219  1         NCP$G_STTBL_ZER,                 ! Zero command
```

```
;  223     0220  1         NCPSG_KYTBL_ZER
;  224     0221  1         ;
```

```
;   226          0222  1 %SBTTL  'Parameter blocks'
;   227          0223  1
;   228          0224  1 !
;   229          0225  1 ! BIND DATA:
;   230          0226  1 !
;   231          0227  1
;   232        P 0228  1         BUILD_PBK
;   233        P 0229  1
;   234        P 0230  1         (VRB,
;   235        P 0231  1
;   236        P 0232  1         ALL, LITB, 0, ,
;   237        P 0233  1         XID, TKN, , ,
;   238        P 0234  1         KWN, LITB, NMA$C_ENT_KNO, VRB_ENT,
;   239        P 0235  1
;   240          0236  1         )
;   241          0237  1
;   242        P 0238  1         BUILD_PBK
;   243        P 0239  1
;   244        P 0240  1         (ENT,
;   245        P 0241  1
;   246        P 0242  1         ALI, TKN, , VRB_ENT,
;   247        P 0243  1         EXE, LITL, 0, VRB_ENT,
;   248        P 0244  1         CIR, TKN, , VRB_ENT,
;   249        P 0245  1         LIN, TKN, , VRB_ENT,
;   250        P 0246  1         NOD, NADR, , VRB_ENT,
;   251        P 0247  1         OBJ, TKN, , VRB_ENT,
;   252        P 0248  1
;   253          0249  1         )
;   254          0250  1
;   255        P 0251  1         BUILD_PBK
;   256        P 0252  1
;   257        P 0253  1         (LOG,
;   258        P 0254  1
;   259        P 0255  1         TYPCON, LITB, NMA$C_SNK_CON, VRB_ENT,
;   260        P 0256  1         TYPFIL, LITB, NMA$C_SNK_FIL, VRB_ENT,
;   261        P 0257  1         TYPMON, LITB, NMA$C_SNK_MON, VRB_ENT,
;   262        P 0258  1
;   263          0259  1         )
;   264          0260  1
;   265        P 0261  1         BUILD_PBK
;   266        P 0262  1
;   267        P 0263  1         (EVE,
;   268        P 0264  1
;   269        P 0265  1         ESET, ESET, , VRB_EVE,
;   270        P 0266  1         ECLS, ECLS, , VRB_EVE,
;   271        P 0267  1         EMSK, EMSK, , VRB_EVE,
;   272        P 0268  1         ERNG, ERNG, , VRB_EVE,
;   273        P 0269  1         EWLD, EWLD, , VRB_EVE,
;   274        P 0270  1         ESNO, ESNO, , VRB_EVE,
;   275        P 0271  1         ESLI, ESLI, , VRB_EVE,
;   276        P 0272  1         ESEX, ESEX, , VRB_EVE,
;   277        P 0273  1
;   278          0274  1         )
```

```
;  280     0275  1
;  281     0276  1 !
;  282     0277  1 !       Control blocks for ACT$ZAPTMPDSC
;  283     0278  1 !
;  284     0279  1
;  285     0280  1 GLOBAL BIND
;  286     0281  1
;  287     0282  1         PBK$G_ZAPACCDSC =               ! Zap descriptors for access control
;  288     0283  1                 PLIT    (
;  289     0284  1                         ACT$GQ_ACCUSR_DSC,
;  290     0285  1                         ACT$GQ_ACCACC_DSC,
;  291     0286  1                         ACT$GQ_ACCPSW_DSC
;  292     0287  1                         )
;  293     0288  1         ;
```

```
;  295          0289  1 %SBTTL  'Prompt Strings'
;  296          0290  1
;  297          0291  1 !
;  298          0292  1 !       Prompt Strings
;  299          0293  1 !
;  300          0294  1
;  301          0295  1 BIND
;  302        P 0296  1         PROMPT_STRINGS
;  303        P 0297  1         (ENT,
;  304        P 0298  1
;  305        P 0299  1         CIR, 'Circuit ID string       (16 characters): ',
;  306        P 0300  1         LIN, 'Line ID                      (dev-c-u.t): ',
;  307        P 0301  1         LOG, 'Type of logging  (CONSOLE, FILE, MONITOR): ',
;  308        P 0302  1         KWN, '(CIRCUITS, LINES, LOGGING, NODES, OBJECTS): ',
;  309        P 0303  1         NOD, 'Node ID               (node-name, address): ',
;  310        P 0304  1         OBJ, 'Object name                (12 characters): ',
;  311        P 0305  1 !!      MOD, %STRING('Module (X25-ACCESS, X25-PROTOCOL, X25-SERVER,',CRLF,
;  312        P 0306  1 !!                   '       X25-TRACE, X29-SERVER): '),
;  313        L 0307  1         MOD, %STRING('Module (CONFIGURATOR, CONSOLE, LOADER,',CRLF,
;  314        L 0308  1                      '       LOOPER, X25-ACCESS, X25-PROTOCOL,',CRLF,
;  315        P 0309  1                      '       X25-SERVER, X25-TRACE, X29-SERVER): '),
;  316        P 0310  1
;  317          0311  1         ),
;  318          0312  1
;  319        P 0313  1         PROMPT_STRINGS
;  320        P 0314  1         (VRB,
;  321        P 0315  1
;  322        P 0316  1         XID, 'Executor node ID     (node-name, address): ',
;  323        P 0317  1
;  324        P 0318  1         TELL, 'Executor node ID     (node-name, address): ',
;  325        P 0319  1
;  326        L 0320  1         SDF1, %STRING(
;  327        L 0321  1                 '(CIRCUIT, EXECUTOR, KNOWN, LINE,',CRLF,
;  328        P 0322  1                 '   LOGGING, MODULE, NODE, OBJECT): '),
;  329        P 0323  1
;  330        L 0324  1         VRB, %STRING (
;  331        L 0325  1                 '(SET, DEFINE, CONNECT, DISCONNECT, CLEAR, PURGE,',CRLF,
;  332        P 0326  1                 ' SHOW, LIST, DUMP, LOAD, TRIGGER, LOOP, ZERO): '),
;  333        P 0327  1
;  334          0328  2         )
;  335          0329  1         ;
```

```
;   337          0330  1 %SBTTL  'Root of the state table'
;   338          0331  1
;   339          0332  1 $INIT_STATE (NCP$G_STATE_TBL, NCP$G_KEY_TBL);
;   340          0333  1
;   341          0334  1 !
;   342          0335  1 !       Allow TELL or a VERB here or EOS
;   343          0336  1 !
;   344          0337  1
;   345        P 0338  1 $STATE  (ST_CMD,
;   346        P 0339  1          (TPA$_LAMBDA, , ACT$CLRLONG, , , NCP$GL_OPTION)
;   347          0340  1          );
;   348          0341  1
;   349        P 0342  1 $STATE  (,
;   350        P 0343  1          (TPA$_LAMBDA, , ACT$CLRLONG, , , NCP$GL_FNC_CODE)
;   351          0344  1          );
;   352          0345  1
;   353        P 0346  1 $STATE  (,
;   354        P 0347  1          (TPA$_LAMBDA, , ACT$CLRLONG, , , NCP$GL_QUALPRS)
;   355          0348  1          );
;   356          0349  1
;   357        P 0350  1 $STATE  (,
;   358        P 0351  1          (TPA$_LAMBDA, , ACT$CLRLONG, , , NCP$GL_NOPARMS)
;   359          0352  1          );
;   360          0353  1
;   361        P 0354  1 $STATE  (,
;   362        P 0355  1          ('CLEAR',       ST_VRB_CLE),
;   363        P 0356  1          ('EXIT',        TPA$_EXIT, ACT$VRB_EXIT),
;   364        P 0357  1          ('HELP',        ST_HELP),
;   365        P 0358  1          ('SET',         ST_VRB_EXN),
;   366        P 0359  1          ('TELL',        ST_VRB_TELL),
;   367        P 0360  1          (TPA$_EOS,      TPA$_EXIT),
;   368        P 0361  1          (TPA$_LAMBDA,   ST_VRB_VRB)
;   369          0362  1          );
;   370          0363  1
;   371          0364  1 !
;   372          0365  1 !       For TELL require a node-id next
;   373          0366  1 !
;   374          0367  1
;   375        P 0368  1          COMMAND_PROMPT
;   376        P 0369  1          (VRB, TELL, NCP$_INVVAL,
;   377        P 0370  1
;   378        P 0371  1          ( (SE_NODE_SPEC), , ACT$SAVPRM, , , PBK$G_VRB_XID)
;   379        P 0372  1
;   380          0373  1          )
;   381          0374  1 !
;   382          0375  1 !       And optional access information next
;   383          0376  1 !
;   384          0377  1
;   385        P 0378  1 $STATE  (,
;   386        P 0379  1          ( (SE_ACCESS) ),
;   387        P 0380  1          (TPA$_LAMBDA)
;   388          0381  1          );
;   389          0382  1
;   390        P 0383  1 $STATE  (,                              ! Dummy state to provide action
;   391        P 0384  1          (TPA$_LAMBDA, ST_VRB_VRB, ACT$VRB_TELL)
;   392          0385  1          );
;   393          0386  1
```

```
;  395          0387  1
;  396          0388  1 !
;  397          0389  1 !       Decode and perform Help command
;  398          0390  1 !
;  399          0391  1
;  400          0392  1 !
;  401          0393  1 !       Call the help action routine.
;  402          0394  1 !
;  403          0395  1
;  404        P 0396  1 $STATE  (ST_HELP,
;  405        P 0397  1          (TPA$_LAMBDA, TPA$_EXIT, ACT$HELP)      ! Call the action routine
;  406          0398  1          );
```

```
;  408          0399  1 %SBTTL  'Decode verbs'
;  409          0400  1
;  410          0401  1 !
;  411          0402  1 !       Verb decoding states
;  412          0403  1 !
;  413          0404  1
;  414        P 0405  1         COMMAND_PROMPT
;  415        P 0406  1         (VRB, VRB, NCP$_INVKEY,
;  416        P 0407  1
;  417        P 0408  1         ('CLEAR',       ST_VRB_CLPU),
;  418        P 0409  1         ('CONNECT',     ST_VRB_CON),
;  419        P 0410  1         ('DEFINE',      ST_VRB_SDF  , NMA$M_OPT_PER, NCP$GL_OPTION, ),
;  420        P 0411  1         ('DISCONNECT',  ST_VRB_DIS),
;  421        P 0412  1         ('DUMP',        ST_VRB_DUM),
;  422        P 0413  1         ('LIST',        ST_VRB_SHL  , NMA$M_OPT_PER, NCP$GL_OPTION, ),
;  423        P 0414  1         ('LOAD',        ST_VRB_LOA),
;  424        P 0415  1         ('LOOP',        ST_VRB_LOO),
;  425        P 0416  1         ('PURGE',       ST_VRB_CLPU , NMA$M_OPT_PER, NCP$GL_OPTION, ),
;  426        P 0417  1         ('SET',         ST_VRB_SDF),
;  427        P 0418  1         ('SHOW',        ST_VRB_SHL),
;  428        P 0419  1         ('TRIGGER',     ST_VRB_TRI),
;  429        P 0420  1         ('ZERO',        ST_VRB_ZER),
;  430        P 0421  1         (TPA$_SYMBOL, TPA$_EXIT, ACT$INV_COMMAND)
;  431          0422  1         )
```

```
;  433          0423  1 %SBTTL  'Set and Define Verbs'
;  434          0424  1
;  435          0425  1 !
;  436          0426  1 !       Set and Define Verbs
;  437          0427  1 !
;  438          0428  1
;  439        P 0429  1 $STATE  (ST_VRB_SDF,
;  440        P 0430  1          (TPA$_LAMBDA, , , NMA$C_FNC_CHA, NCP$GL_FNC_CODE)
;  441          0431  1          );
;  442          0432  1
;  443        P 0433  1          COMMAND_PROMPT
;  444        P 0434  1          (VRB, SDF1, NCP$_INVKEY,
;  445        P 0435  1
;  446        P 0436  1          ('CIRCUIT',      ST_ENT_CIR),
;  447        P 0437  1          ('CONFIGURATOR',ST_ENT_CNF),
;  448        P 0438  1          ('CONSOLE',      ST_ENT_CNS),
;  449        P 0439  1          ('DTE',          ST_ENT_MPRDTE),
;  450        P 0440  1          ('EXECUTOR',     ST_ENT_EXE),
;  451        P 0441  1          ('GROUP',        ST_ENT_MPRGRP),
;  452        P 0442  1          ('KNOWN',        ST_ENT_KWN),
;  453        P 0443  1          ('LINE',         ST_ENT_LIN),
;  454        P 0444  1          ('LOADER',       ST_ENT_LOA),
;  455        P 0445  1          ('LOGGING',      ST_ENT_LOG),
;  456        P 0446  1          ('LOOPER',       ST_ENT_LOO),
;  457        P 0447  1          ('MODULE',       ST_ENT_MOD),
;  458        P 0448  1          ('NODE',         ST_ENT_NOD),
;  459        P 0449  1          ('OBJECT',       ST_ENT_OBJ),
;  460        P 0450  1          ('TRACEPOINT',   ST_ENT_MTRTPT),
;  461        P 0451  1          ('X25',          ST_ENT_X25),
;  462        P 0452  1          ('X29',          ST_ENT_X29),
;  463          0453  1          )
;  464          0454  1
;  465        P 0455  1 $STATE  (ST_VRB_EXN,                      ! SET as first VERB
;  466        P 0456  1          ('EXECUTOR'),
;  467        P 0457  1          (TPA$_LAMBDA, ST_VRB_SDF)
;  468          0458  1          );
;  469          0459  1
;  470        P 0460  1 $STATE  (,                                ! EXECUTOR NODE?
;  471        P 0461  1          ('NODE'),
;  472        P 0462  1          (TPA$_LAMBDA, ST_ENT_EXE, , NMA$C_FNC_CHA, NCP$GL_FNC_CODE,)
;  473          0463  1          );                               ! No, use normal SET EXECUTOR
;  474          0464  1
;  475        P 0465  1          COMMAND_PROMPT
;  476        P 0466  1          (VRB, XID, NCP$_INVVAL,
;  477        P 0467  1
;  478        P 0468  1          ( (SE_NODE_SPEC), , ACT$SAVPRM, , , PBK$G_VRB_XID)
;  479        P 0469  1
;  480          0470  1          )
;  481          0471  1
;  482        P 0472  1 $STATE  (,                                ! And optional access control
;  483        P 0473  1          ( (SE_ACCESS) ),
;  484        P 0474  1          (TPA$_LAMBDA)
;  485          0475  1          );
;  486          0476  1
;  487        P 0477  1 $STATE  (,                                ! Dummy state to perform action
;  488        P 0478  1          (TPA$_EOS, TPA$_EXIT, ACT$VRB_SETEXEC)
;  489          0479  1          );
```

```
;  491          0480  1 %SBTTL  'Set / Define Processing'
;  492          0481  1
;  493          0482  1 !
;  494          0483  1 !       Set / Define Processing
;  495          0484  1 !
;  496          0485  1
;  497          0486  1 !
;  498          0487  1 !       Executor node
;  499          0488  1 !
;  500          0489  1
;  501        P 0490  1 $STATE  (ST_ENT_EXE,
;  502        P 0491  1          (TPA$_LAMBDA, , ACT$SAVPRM, NMA$C_ENT_NOD,
;  503        P 0492  1                                      NCP$GL_OPTION, PBK$G_ENT_EXE)
;  504          0493  1          );
;  505          0494  1
;  506        P 0495  1 $STATE  (,
;  507        P 0496  1          (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE, , , NEXT_STATE (NOD) )
;  508          0497  1          );
;  509          0498  1
;  510          0499  1 !
;  511          0500  1 !       Circuits
;  512          0501  1 !
;  513          0502  1
;  514        P 0503  1          COMMAND_PROMPT
;  515        P 0504  1          (ENT, CIR, NCP$_INVVAL,
;  516        P 0505  1
;  517        P 0506  1          ( (SE_CIRC_ID), , ACT$SAVPRM, NMA$C_ENT_CIR,
;  518        P 0507  1                                      NCP$GL_OPTION, PBK$G_ENT_CIR)
;  519          0508  1          )
;  520          0509  1
;  521        P 0510  1 $STATE  (ST_KWN_CIR,
;  522        P 0511  1          (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE, , , NEXT_STATE (CIR) )
;  523          0512  1          );
;  524          0513  1
;  525          0514  1 !
;  526          0515  1 !       Lines
;  527          0516  1 !
;  528          0517  1
;  529        P 0518  1          COMMAND_PROMPT
;  530        P 0519  1          (ENT, LIN, NCP$_INVVAL,
;  531        P 0520  1
;  532        P 0521  1          ( (SE_LINE_ID), , ACT$SAVPRM, NMA$C_ENT_LIN,
;  533        P 0522  1                                      NCP$GL_OPTION, PBK$G_ENT_LIN)
;  534          0523  1          )
;  535          0524  1
;  536        P 0525  1 $STATE  (ST_KWN_LIN,
;  537        P 0526  1          (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE, , , NEXT_STATE (LIN) )
;  538          0527  1          );
;  539          0528  1
;  540          0529  1 !
;  541          0530  1 !       Modules
;  542          0531  1 !
;  543          0532  1
;  544        P 0533  1          COMMAND_PROMPT
;  545        P 0534  1          (ENT, MOD, NCP$_INVKEY,
;  546        P 0535  1
;  547        P 0536  1          ('CONFIGURATOR', ST_ENT_CNF),
```

```
;  548        P 0537  1          ('CONSOLE',      ST_ENT_CNS),
;  549        P 0538  1          ('LOADER',       ST_ENT_LOA),
;  550        P 0539  1          ('LOOPER',       ST_ENT_LOO),
;  551        P 0540  1          ('X25',          ST_ENT_X25),
;  552        P 0541  1          ('X29',          ST_ENT_X29),
;  553          0542  1          )
;  554          0543  1
;  555        P 0544  1 $STATE   (ST_ENT_X25,
;  556          0545  1          ('-'));
;  557        P 0546  1 $STATE   (
;  558        P 0547  1          ('ACCESS',       ST_ENT_MAC),
;  559        P 0548  1          ('PROTOCOL',     ST_ENT_MPR),
;  560        P 0549  1          ('SERVER',       ST_ENT_MSE),
;  561        P 0550  1          ('TRACE',        ST_ENT_MTR)
;  562          0551  1          );
;  563          0552  1
;  564        P 0553  1 $STATE   (ST_ENT_X29,
;  565          0554  1          ('-'));
;  566        P 0555  1 $STATE   (
;  567        P 0556  1          ('SERVER',       ST_ENT_M9S)
;  568          0557  1          );
```

```
;  570        P 0558  1 $STATE  (ST_ENT_CNF,                                        ! MODULE CONFIGURATOR
;  571        P 0559  1         (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE,
;  572        P 0560  1                 NMA$C_ENT_MOD, NCP$GL_OPTION, NEXT_STATE(MODCNF))
;  573           0561  1         );
;  574           0562  1
;  575        P 0563  1 $STATE  (ST_ENT_CNS,                                        ! MODULE CONSOLE
;  576        P 0564  1         (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE,
;  577        P 0565  1                 NMA$C_ENT_MOD, NCP$GL_OPTION, NEXT_STATE(MODCNS))
;  578           0566  1         );
;  579           0567  1
;  580        P 0568  1 $STATE  (ST_ENT_LOA,                                        ! MODULE LOADER
;  581        P 0569  1         (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE,
;  582        P 0570  1                 NMA$C_ENT_MOD, NCP$GL_OPTION, NEXT_STATE(MODLOA))
;  583           0571  1         );
;  584           0572  1
;  585        P 0573  1 $STATE  (ST_ENT_LOO,                                        ! MODULE LOOPER
;  586        P 0574  1         (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE,
;  587        P 0575  1                 NMA$C_ENT_MOD, NCP$GL_OPTION, NEXT_STATE(MODLOO))
;  588           0576  1         );
;  589           0577  1
;  590        P 0578  1 $STATE  (ST_ENT_MAC,                                        ! MODULE X25-ACCESS
;  591        P 0579  1         (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE,
;  592        P 0580  1                 NMA$C_ENT_MOD, NCP$GL_OPTION, NEXT_STATE(MAC))
;  593           0581  1         );
;  594           0582  1
;  595        P 0583  1 $STATE  (ST_ENT_MPR,                                          ! MODULE X25-PROTOCOL
;  596        P 0584  1         (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE,
;  597        P 0585  1                 NMA$C_ENT_MOD, NCP$GL_OPTION, NEXT_STATE(MPR))
;  598           0586  1         );
;  599        P 0587  1 $STATE  (ST_ENT_MPRDTE,                                       ! MODULE X25-PROTOCOL DTE
;  600        P 0588  1         (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE,
;  601        P 0589  1                 NMA$C_ENT_MOD, NCP$GL_OPTION, NEXT_STATE(MPRDTE))
;  602           0590  1         );
;  603        P 0591  1 $STATE  (ST_ENT_MPRGRP,                                       ! MODULE X25-PROTOCOL GROUP
;  604        P 0592  1         (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE,
;  605        P 0593  1                 NMA$C_ENT_MOD, NCP$GL_OPTION, NEXT_STATE(MPRGRP))
;  606           0594  1         );
;  607           0595  1
;  608        P 0596  1 $STATE  (ST_ENT_MSE,                                          ! MODULE X25-SERVER
;  609        P 0597  1         (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE,
;  610        P 0598  1                 NMA$C_ENT_MOD, NCP$GL_OPTION, NEXT_STATE(MSE))
;  611           0599  1         );
;  612           0600  1
;  613        P 0601  1 $STATE  (ST_ENT_MTR,                                          ! MODULE X25-TRACE
;  614        P 0602  1         (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE,
;  615        P 0603  1                 NMA$C_ENT_MOD, NCP$GL_OPTION, NEXT_STATE(MTR))
;  616           0604  1         );
;  617        P 0605  1 $STATE  (ST_ENT_MTRTPT,                                       ! MODULE X25-TRACE TRACEPOINT
;  618        P 0606  1         (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE,
;  619        P 0607  1                 NMA$C_ENT_MOD, NCP$GL_OPTION, NEXT_STATE(MTRTPT))
;  620           0608  1         );
;  621           0609  1
;  622        P 0610  1 $STATE  (ST_ENT_M9S,                                          ! MODULE X29-SERVER
;  623        P 0611  1         (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE,
;  624        P 0612  1                 NMA$C_ENT_MOD, NCP$GL_OPTION, NEXT_STATE(M9S))
;  625           0613  1         );
;  626           0614  1
```

```
;  628          0615  1
;  629          0616  1 !
;  630          0617  1 !       Known Entities
;  631          0618  1 !
;  632          0619  1
;  633        P 0620  1          COMMAND_PROMPT
;  634        P 0621  1          (ENT, KWN, NCP$_INVKEY,
;  635        P 0622  1
;  636        P 0623  1          ('CIRCUITS',    ST_KWN_CIR, ACT$SAVPRM, NMA$C_ENT_CIR,
;  637        P 0624  1                                      NCP$GL_OPTION, PBR$G_VRB_KWN),
;  638        P 0625  1          ('LINES',       ST_KWN_LIN, ACT$SAVPRM, NMA$C_ENT_LIN,
;  639        P 0626  1                                      NCP$GL_OPTION, PBR$G_VRB_KWN),
;  640        P 0627  1          ('LOGGING',     ST_KWN_LOG, ACT$SAVPRM, NMA$C_ENT_LOG,
;  641        P 0628  1                                      NCP$GL_OPTION, PBR$G_VRB_KWN),
;  642        P 0629  1          ('NODES',       ST_KWN_NOD, ACT$SAVPRM, NMA$C_ENT_NOD,
;  643        P 0630  1                                      NCP$GL_OPTION, PBR$G_VRB_KWN),
;  644        P 0631  1          ('OBJECTS',     ST_KWN_OBJ, ACT$SAVPRM, NMA$C_SENT_OBJ,
;  645        P 0632  1                                      NCP$GL_OPTION, PBK$G_VRB_KWN),
;  646        P 0633  1
;  647          0634  1          )
;  648          0635  1
;  649          0636  1 !
;  650          0637  1 !       Logging
;  651          0638  1 !
;  652          0639  1
;  653        P 0640  1          COMMAND_PROMPT
;  654        P 0641  1          (ENT, LOG, NCP$_INVKEY,
;  655        P 0642  1
;  656        P 0643  1          ( (SE_LOG_TYP)  , , NMA$C_ENT_LOG, NCP$GL_OPTION)
;  657        P 0644  1
;  658          0645  1          )
;  659          0646  1
;  660        P 0647  1 $STATE   (ST_KWN_LOG,
;  661        P 0648  1          (TPA$_LAMBDA,   TPA$_EXIT, ACT$NXT_STATE, , , NEXT_STATE (LOG) )
;  662          0649  1          );
```

```
;  664          0650  1
;  665          0651  1 !
;  666          0652  1 !       Nodes
;  667          0653  1 !
;  668          0654  1
;  669        P 0655  1          COMMAND_PROMPT
;  670        P 0656  1          (ENT, NOD, NCP$_INVVAL,
;  671        P 0657  1
;  672        P 0658  1          ( (SE_NODE_ID), , ACT$SAVPRM, NMA$C_ENT_NOD,
;  673        P 0659  1                                        NCP$GL_OPTION, PBK$G_ENT_NOD)
;  674        P 0660  1
;  675          0661  1          )
;  676          0662  1
;  677        P 0663  1 $STATE   (ST_KWN_NOD,
;  678        P 0664  1          (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE, , , NEXT_STATE (NOD) )
;  679          0665  1          );
;  680          0666  1
;  681          0667  1 !
;  682          0668  1 !       Objects
;  683          0669  1 !
;  684          0670  1
;  685        P 0671  1          COMMAND_PROMPT
;  686        P 0672  1          (ENT, OBJ, NCP$_INVVAL,
;  687        P 0673  1
;  688        P 0674  1          ( (SE_OBJECT_ID), , ACT$SAVPRM, NMA$C_SENT_OBJ,
;  689        P 0675  1                                        NCP$GL_OPTION, PBK$G_ENT_OBJ)
;  690        P 0676  1
;  691          0677  1          )
;  692          0678  1
;  693        P 0679  1 $STATE   (ST_KWN_OBJ,
;  694        P 0680  1          (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE, , , NEXT_STATE (OBJ) )
;  695          0681  1          );
;  696          0682  1
```

```
;  698          0683  1 %SBTTL  'Clear/Purge Dispatching'
;  699          0684  1
;  700          0685  1 !
;  701          0686  1 !       Dispatching for Clear and Purge
;  702          0687  1 !
;  703          0688  1
;  704          0689  1 !
;  705          0690  1 !       CLEAR EXECUTOR NODE
;  706          0691  1 !
;  707          0692  1
;  708        P 0693  1 $STATE  (ST_VRB_CLE,
;  709        P 0694  1          ( (SE_VRB_CLEX) ),                  ! Is this clear executor node?
;  710        P 0695  1          (TPA$_LAMBDA, ST_VRB_CLPU)
;  711          0696  1          );
   712          0697  1
;  713        P 0698  1 $STATE  (,                                   ! Perform the clear executor node
;  714        P 0699  1          (TPA$_EOS, TPA$_EXIT, ACT$VRB_CLEXEC)
;  715          0700  1          );
;  716          0701  1
;  717        P 0702  1 $STATE  (SE_VRB_CLEX,                        ! Succeed if executor node
;  718        P 0703  1          ('EXECUTOR')                        ! Otherwise fail
;  719          0704  1          );
;  720          0705  1
;  721        P 0706  1 $STATE  (,
;  722        P 0707  1          ('NODE', TPA$_EXIT)
;  723          0708  1          );
;  724          0709  1
;  725          0710  1 !
;  726          0711  1 !       Clear and Purge dispatch to another state table
;  727          0712  1 !
;  728          0713  1
;  729        P 0714  1 $STATE  (ST_VRB_CLPU,
;  730        P 0715  1          (TPA$_LAMBDA, , , NMA$M_OPT_CLE, NCP$GL_OPTION)
;  731          0716  1          );
;  732          0717  1
;  733        P 0718  1 $STATE  (,
;  734        P 0719  1          (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE, NMA$C_FNC_CHA,
;  735        P 0720  1                                          NCP$GL_FNC_CODE, NEXT_STATE (CLPU) )
;  736          0721  1          );
;  737          0722  1
```

```
;  739          0723  1 %SBTTL  'Connect Verb'
;  740          0724  1
;  741          0725  1 !
;  742          0726  1 !       Connect Verb
;  743          0727  1 !
;  744          0728  1
;  745        P 0729  1 $STATE  (ST_VRB_CON,
;  746        P 0730  1 !!      (TPA$_LAMBDA, , , NMA$M_OPT_CLE, NCP$GL_OPTION)
;  747        P 0731  1 !!      );
;  748        P 0732  1 !!
;  749        P 0733  1 !!$STATE      (,
;  750        P 0734  1         (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE, NMA$C_FNC_SYS,
;  751        P 0735  1                                 NCP$GL_FNC_CODE, NEXT_STATE (CON) )
;  752          0736  1         );
;  753          0737  1
;  754          0738  1
;  755          0739  1 %SBTTL  'Disconnect Verb'
;  756          0740  1
;  757          0741  1 !
;  758          0742  1 !       Disconnect Verb
;  759          0743  1 !
;  760          0744  1
;  7          P 0745  1 $STATE  (ST_VRB_DIS,
;  762        P 0746  1         (TPA$_LAMBDA, , , NMA$M_OPT_CLE, NCP$GL_OPTION)
;  763          0747  1         );
;  764          0748  1
;  765        P 0749  1 $STATE  (,
;  766        P 0750  1         (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE, NMA$C_FNC_CHA,
;  767        P 0751  1                                 NCP$GL_FNC_CODE, NEXT_STATE (DIS) )
;  768          0752  1         );
;  769          0753  1
;  770          0754  1 %SBTTL  'Dump Verb'
;  771          0755  1
;  772          0756  1 !
;  773          0757  1 !       Dump Verb
;  774          0758  1 !
;  775          0759  1
;  776        P 0760  1 $STATE  (ST_VRB_DUM,
;  777        P 0761  1         (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE, NMA$C_FNC_DUM,
;  778        P 0762  1                                 NCP$GL_FNC_CODE, NEXT_STATE (DUM) )
;  779          0763  1         );
;  780          0764  1
;  781          0765  1 %SBTTL  'Load Verb'
;  782          0766  1
;  783          0767  1 !
;  784          0768  1 !       Load Verb
;  785          0769  1 !
;  786          0770  1
;  787        P 0771  1 $STATE  (ST_VRB_LOA,
;  788        P 0772  1         (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE, NMA$C_FNC_LOA,
;  789        P 0773  1                                 NCP$GL_FNC_CODE, NEXT_STATE (LOA) )
;  790          0774  1         );
;  791          0775  1
;  792          0776  1
;  793          0777  1 %SBTTL  'Loop Verb'
;  794          0778  1
;  795          0779  1 !
```

```
;  796          0780  1 !       Loop Verb
;  797          0781  1 !
;  798          0782  1
;  799        P 0783  1 $STATE  (ST_VRB_LOO,
;  800        P 0784  1         (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE, NMA$C_FNC_TES,
;  801        P 0785  1                                 NCP$GL_FNC_CODE, NEXT_STATE (LOO) )
;  802          0786  1         );
```

```
;  804          0787  1 %SBTTL  'Show / List Verbs'
;  805          0788  1
;  806          0789  1 !
;  807          0790  1 !       Show / List Verbs
;  808          0791  1 !
;  809          0792  1
;  810        P 0793  1 $STATE  (ST_VRB_SHL,
;  811        P 0794  1          (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE, NMA$C_FNC_REA,
;  812        P 0795  1                                                  NCP$GL_FNC_CODE, NEXT_STATE (SHL) ),
;  813          0796  1          );
;  814          0797  1
;  815          0798  1
;  816          0799  1 %SBTTL  'Trigger Verb'
;  817          0800  1
;  818          0801  1 !
;  819          0802  1 !       Trigger Verb
;  820          0803  1 !
;  821          0804  1
;  822        P 0805  1 $STATE  (ST_VRB_TRI,
;  823        P 0806  1          (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE, NMA$C_FNC_TRI,
;  824        P 0807  1                                                  NCP$GL_FNC_CODE, NEXT_STATE (TRI) )
;  825          0808  1          );
;  826          0809  1
;  827          0810  1
;  828          0811  1 %SBTTL  'Zero Verb'
;  829          0812  1
;  830          0813  1 !
;  831          0814  1 !       Zero Verb
;  832          0815  1 !
;  833          0816  1
;  834        P 0817  1 $STATE  (ST_VRB_ZER,
;  835        P 0818  1          (TPA$_LAMBDA, TPA$_EXIT, ACT$NXT_STATE, NMA$C_FNC_ZER,
;  836        P 0819  1                                                  NCP$GL_FNC_CODE, NEXT_STATE (ZER) )
;  837          0820  1          );
```

```
;  839          0821  1 %SBTTL  'Define Subexpressions'
;  840          0822  1
;  841          0823  1 !
;  842          0824  1 !       Subexpression to decode a node specification
;  843          0825  1 !
;  844          0826  1
;  845        P 0827  1 $STATE  (SE_NODE_SPEC,
;  846        P 0828  1          ( (SE_NOD_SPC), , ACT$STR_LEN, , , LEN_FILE_SPEC),
;  847        P 0829  1          (TPA$_LAMBDA, TPA$_FAIL, ACT$BLNK_NSIG)
;  848          0830  1          );
;  849          0831  1
;  850        P 0832  1 $STATE  (,
;  851        P 0833  1          (':'),
;  852        P 0834  1          (TPA$_LAMBDA, TPA$_EXIT, ACT$BLNK_NSIG)
;  853          0835  1          );
;  854          0836  1
;  855        P 0837  1 $STATE  (,
;  856        P 0838  1          (':', TPA$_EXIT, ACT$BLNK_NSIG),
;  857        P 0839  1          (TPA$_LAMBDA, TPA$_FAIL, ACT$BLNK_NSIG)
;  858          0840  1          );
```

```
;  860          0841  1
;  861          0842  1 !
;  862          0843  1 !       Decode the specification string
;  863          0844  1 !
;  864          0845  1
;  865        P 0846  1 $STATE  (SE_NOD_SPC,                                          ! Symbol for the node name
;  866        P 0847  1          ((SE_NOD_AREA_Q), , ACT$CLRLONG, , , ACT$GL_XIDACC_Q),  ! If an area is present then the '.' was rej
;  867        P 0848  1                                                               !  so check for it here
;  868        P 0849  1
;  869        P 0850  1          ((SE_NOD_ADRS), , ACT$CLRLONG, , , ACT$GL_XIDACC_Q),    ! Flag node address without area.
;  870        P 0851  1          (TPA$_SYMBOL, , ACT$CLRLONG, , , ACT$GL_XIDACC_Q),      ! To allow logical names
;  871          0852  1          );
;  872          0853  1
;  873        P 0854  1 $STATE  (,
;  874        P 0855  1          ('...', , ACT$BLNK_SIG),                             ! Access control may follow
;  875        P 0856  1          (TPA$_LAMBDA, TPA$_EXIT)                             ! Or not
;  876          0857  1          );
;  877          0858  1
;  878        P 0859  1 $STATE  (,                                                    ! If access control,
;  879        P 0860  1          ((SE_SPC_STR), , , TRUE, ACT$GL_XIDACC_Q),           ! Get the string or
;  880        P 0861  1          ('', TPA$_EXIT, , TRUE, ACT$GL_XIDACC_Q),            ! Allow null accctl
;  881          0862  1          );
;  882          0863  1
;  883        P 0864  1 $STATE  (,                                                    ! Blank after string or
;  884        P 0865  1          (TPA$_BLANK),
;  885        P 0866  1          ('', TPA$_EXIT)                                      ! End it
;  886          0867  1          );
;  887          0868  1
;  888        P 0869  1 $STATE  (,
;  889        P 0870  1          ((SE_SPC_STR) )                                      ! Password string
;  890          0871  1          );
;  891          0872  1
;  892        P 0873  1 $STATE  (,
;  893        P 0874  1          (TPA$_BLANK),                                        ! And blank or end
;  894        P 0875  1          ('', TPA$_EXIT)
;  895          0876  1          );
;  896          0877  1
;  897        P 0878  1 $STATE  (,
;  898        P 0879  1          ((SE_SPC_STR) )                                      ! Account string
;  899          0880  1          );
;  900          0881  1
;  901        P 0882  1 $STATE  (,
;  902        P 0883  1          ('...', TPA$_EXIT)                                   ! And end it here or fail
;  903          0884  1          );
```

```
;  905          0885  1
;  906          0886  1 !
;  907          0887  1 !       Decode a string acceptable for access control in a node spec
;  908          0888  1 !
;  909          0889  1
;  910          0890  1
;  911        P 0891  1 $STATE  (SE_SPC_STR,                    ! Accept a string for access control
;  912        P 0892  1          ( (SE_SPC_CHR) )               ! Start with a character
;  913          0893  1          );
;  914          0894  1
;  915        P 0895  1 $STATE  (SE_SPC_STR1,                   ! And accept any after that
;  916        P 0896  1          ( (SE_SPC_CHR), SE_SPC_STR1),
;  917        P 0897  1          (TPA$_LAMBDA,   TPA$_EXIT)
;  918          0898  1          );
;  919          0899  1
;  920        P 0900  1 $STATE  (SE_SPC_CHR,                    ! A access control char is any except
;  921        P 0901  1          ('"',           TPA$_FAIL),    ! Double quote and
;  922        P 0902  1          (TPA$_BLANK,    TPA$_FAIL),    ! Space or tab
;  923        P 0903  1          (TPA$_ANY,      TPA$_EXIT)
;  924          0904  1          );
```

```
;  926          0905  1
;  927          0906  1 !
;  928          0907  1 !       Obtain access control in the more general case
;  929          0908  1 !
;  930          0909  1
;  931        P 0910  1 $STATE  (SE_ACCESS,                       ! Dummy state to clear descriptors
;  932        P 0911  1          (TPA$_LAMBDA,    , ACT$ZAPTMPDSC, , , PBK$G_ZAPACCDSC)
;  933          0912  1          );
;  934          0913  1
;  935        P 0914  1 $STATE  (,                                ! Take any one first but there must be
;  936        P 0915  1          ('ACCOUNT',      ST_ACCESS_ACC), ! at least one or fail
;  937        P 0916  1          ('PASSWORD',     ST_ACCESS_PSW),
;  938        P 0917  1          ('USER',         ST_ACCESS_USR)
;  939          0918  1          );
;  940          0919  1
;  941        P 0920  1 $STATE  (ST_ACCESS_1,                     ! Now there can be any remaining
;  942        P 0921  1          ('ACCOUNT',      ST_ACCESS_ACC), ! number but there need not be
;  943        P 0922  1          ('PASSWORD',     ST_ACCESS_PSW),
;  944        P 0923  1          ('USER',         ST_ACCESS_USR),
;  945        P 0924  1          (TPA$_LAMBDA,    TPA$_EXIT)
;  946          0925  1          );
;  947          0926  1
;  948        P 0927  1 $STATE  (ST_ACCESS_ACC,                   ! State for an account string
;  949        P 0928  1          ( (SE_ACCESS_ACC), ST_ACCESS_1),
;  950        P 0929  1          (TPA$_LAMBDA,    TPA$_FAIL, ACT$INV_COMMAND)
;  951          0930  1          );
;  952          0931  1
;  953        P 0932  1 $STATE  (SE_ACCESS_ACC,                   ! Subexpression for an account string
;  954        P 0933  1          ( (SE_ACC_ACC), TPA$_EXIT ,ACT$TMPSTR, , ,ACT$GQ_ACCACC_DSC)
;  955          0934  1          );
;  956        P 0935  1 $STATE  (ST_ACCESS_PSW,                   ! State for a password string
;  957        P 0936  1          ( (SE_ACCESS_PSW), ST_ACCESS_1),
;  958        P 0937  1          (TPA$_LAMBDA,    TPA$_FAIL, ACT$INV_COMMAND)
;  959          0938  1          );
;  960          0939  1
;  961        P 0940  1 $STATE  (SE_ACCESS_PSW,                   ! Subexpression for a password string
;  962        P 0941  1          ( (SE_ACC_PSW), TPA$_EXIT ,ACT$TMPSTR, , ,ACT$GQ_ACCPSW_DSC)
;  963          0942  1          );
;  964          0943  1
;  965        P 0944  1 $STATE  (ST_ACCESS_USR,                   ! State for a user id string
;  966        P 0945  1          ( (SE_ACCESS_USR), ST_ACCESS_1),
;  967        P 0946  1          (TPA$_LAMBDA,    TPA$_FAIL, ACT$INV_COMMAND)
;  968          0947  1          );
;  969          0948  1
;  970        P 0949  1 $STATE  (SE_ACCESS_USR,                   ! Subexpression for a user id string
;  971        P 0950  1          ( (SE_ACC_USR), TPA$_EXIT ,ACT$TMPSTR, , ,ACT$GQ_ACCUSR_DSC)
;  972          0951  1          );
```

```
;  974          0952  1 !
;  975          0953  1 !       See if the node address has an area in front.
;  976          0954  1 !       Format is area.adr, where area and adr are decimal.
;  977          0955  1 !
;  978        P 0956  1 $STATE  (SE_NOD_AREA_Q,
;  979        P 0957  1          (TPA$_DECIMAL)
;  980          0958  1          );
;  981          0959  1
;  982        P 0960  1 $STATE  (,
;  983        P 0961  1          ('.', , ACT$NUM_RNG, , ,                                  ! The last number parsed was indeed an area
;  984        P 0962  1                NUM_RANGE (LOW_AREA, HIGH_AREA)),                   !  so check the range
;  985        P 0963  1          (TPA$_LAMBDA, TPA$_FAIL)                                  ! There was no area so clear storage
;  986          0964  1          );
;  987          0965  1
;  988        P 0966  1 $STATE  (,
;  989        P 0967  1          (TPA$_DECIMAL, TPA$_EXIT, ACT$NUM_RNG, , ,                ! Check the range of the node address
;  990        P 0968  1                NUM_RANGE (LOW_NODE_ADR, HIGH_NODE_ADR))
;  991          0969  1          );
;  992          0970  1
;  993          0971  1 !
;  994          0972  1 !       If area test failed, but there is a valid node address,
;  995          0973  1 !       then accept the node address and set a flag so that
;  996          0974  1 !       ACT$SAVPRM will set the area to 1 if it's a TELL or
;  997          0975  1 !       SET EXEC command.
;  998          0976  1 !
;  999        P 0977  1 $STATE  (SE_NOD_ADRS,
; 1000        P 0978  1          (TPA$_LAMBDA, , ACT$CLRLONG, , , ACT$GL_NO_XAREA_Q)       ! Start with the flag clear.
; 1001          0979  1          );
; 1002          0980  1
; 1003        P 0981  1 $STATE  (,
; 1004        P 0982  1          (TPA$_DECIMAL, , ACT$NUM_RNG, , ,                         ! Check the range of the node address.
; 1005        P 0983  1                NUM_RANGE (LOW_NODE_ADR, HIGH_NODE_ADR))
; 1006          0984  1          );
; 1007          0985  1
; 1008        P 0986  1 $STATE  (,
; 1009        P 0987  1          (TPA$_LAMBDA, TPA$_EXIT, , TRUE, ACT$GL_NO_XAREA_Q)       ! Set the flag to indicate no exex area was
; 1010          0988  1          );
; 1011          0989  1
```

```
;  1013          0990  1 !
;  1014          0991  1 !       Call subexpressions we need from the library
;  1015          0992  1 !
;  1016          0993  1
;  1017          0994  1         SEM_NODE_ID             ! Node id parsing
;  1018          0995  1         SEM_ACCESS              ! General access string parsing
;  1019          0996  1         SEM_QUOT_STR            ! Quoted string
;  1020          0997  1         SEM_LINE_ID             ! Line ID
;  1021          0998  1         SEM_CIRC_ID             ! Circuit ID
;  1022          0999  1         SEM_LOG_TYP             ! Logging type
;  1023          1000  1         SEM_OBJECT_ID           ! Object name/number
```

```
;  1025          1001  1 %SBTTL  'Object Listing of Parse Table'
;  1026          1002  1
;  1027          1003  1 END                                   !End of module
;  1028          1004  0 ELUDOM
```

NCPSTANOD
LIS

NCPSTAVRB
LIS

NCPSTASHL
LIS

NCPSTAOBJ
LIS

NCPSTATRI
LIS

NCPSTAZER
LIS

NCPTERMIO
LIS

NCPTABLES
LIS

NCPVRBACT
LIS