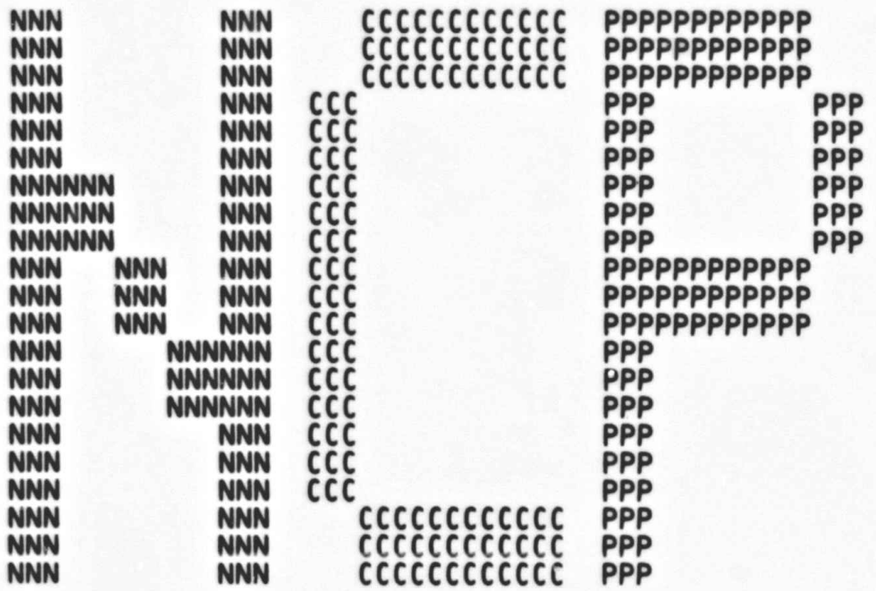
NNN NNN CCCCCCCCCCCC PPPPPPPPPPPP
NNN NNN CCCCCCCCCCCC PPPPPPPPPPPP
NNN NNN CCCCCCCCCCCC PPPPPPPPPPPP
NNN NNN CCC PPP PPP
NNN NNN CCC PPP PPP
NNN NNN CCC PPP PPP
NNNNNN NNN CCC PPP PPP
NNNNNN NNN CCC PPP PPP
NNNNNN NNN CCC PPP PPP
NNN NNN NNN CCC PPPPPPPPPPPP
NNN NNN NNN CCC PPPPPPPPPPPP
NNN NNN NNN 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 CCCCCCCCCCCC PPP
NNN NNN CCCCCCCCCCCC PPP
NNN NNN CCCCCCCCCCCC PPP

FILEID**NCPLIBRY

```
NN      NN   CCCCCCCC  PPPPPPPP    LL          IIIIII   BBBBBBBB   RRRRRRRR    YY      YY
NN      NN   CCCCCCCC  PPPPPPPP    LL          IIIIII   BBBBBBBB   RRRRRRRR    YY      YY
NN      NN  CC         PP      PP  LL            II     BB      BB  RR      RR  YY      YY
NN      NN  CC         PP      PP  LL            II     BB      BB  RR      RR  YY      YY
NNNN    NN  CC         PP      PP  LL            II     BB      BB  RR      RR    YY  YY
NNNN    NN  CC         PP      PP  LL            II     BB      BB  RR      RR    YY  YY
NN  NN  NN  CC         PPPPPPPP    LL            II     BBBBBBBB   RRRRRRRR        YY
NN  NN  NN  CC         PPPPPPPP    LL            II     BBBBBBBB   RRRRRRRR        YY
NN    NNNN  CC         PP          LL            II     BB      BB  RR  RR         YY
NN    NNNN  CC         PP          LL            II     BB      BB  RR  RR         YY
NN      NN  CC         PP          LL            II     BB      BB  RR    RR       YY        ....
NN      NN  CC         PP          LL            II     BB      BB  RR    RR       YY        ....
NN      NN   CCCCCCCC  PP          LLLLLLLLLL  IIIIII   BBBBBBBB   RR      RR     YY        ....
NN      NN   CCCCCCCC  PP          LLLLLLLLLL  IIIIII   BBBBBBBB   RR      RR     YY        ....

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
```

```
;  0001  0     %TITLE  'NCPLIBRY  Symbol Definition Library'
;  0002  0     !MODULE NCPLIBRY (IDENT = 'V04-000') =
;  0003  0     !BEGIN
;  0004  0
;  0005  0     !
;  0006  0     !*****************************************************************************
;  0007  0     !*                                                                           *
;  0008  0     !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;  0009  0     !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;  0010  0     !*  ALL RIGHTS RESERVED.                                                     *
;  0011  0     !*                                                                           *
;  0012  0     !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
;  0013  0     !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
;  0014  0     !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
;  0015  0     !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
;  0016  0     !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
;  0017  0     !*  TRANSFERRED.                                                             *
;  0018  0     !*                                                                           *
;  0019  0     !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
;  0020  0     !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
;  0021  0     !*  CORPORATION.                                                             *
;  0022  0     !*                                                                           *
;  0023  0     !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
;  0024  0     !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;  0025  0     !*                                                                           *
;  0026  0     !*                                                                           *
;  0027  0     !*****************************************************************************
;  0028  0     !
;  0029  0
;  0030  0     !++
;  0031  0     ! FACILITY:     NCP Network Control Program (NCP)
;  0032  0     !
;  0033  0     ! ABSTRACT:
;  0034  0     !
;  0035  0     !       NCP Library of common definitions
;  0036  0     !
;  0037  0     ! ENVIRONMENT:  VAX/VMS Operating System
;  0038  0     !
;  0039  0     ! AUTHOR:       Darrell Duffy   , CREATION DATE: 28-August-1979
;  0040  0     !
;  0041  0     ! MODIFIED BY:
;  0042  0     !
;  0043  0     !       V03-031 PRD0112         Paul R. DeStefano       31-Jul-1984
;  0044  0     !               Allow node address and executive node address of 0.
;  0045  0     !
;  0046  0     !       V03-030 PRD0104         Paul R. DeStefano       18-Jul-1984
;  0047  0     !               Allow underscores ("_") to be included in group,
;  0048  0     !               network, and destination names.
;  0049  0     !
;  0050  0     !       V03-029 PRD0050         Paul R. DeStefano       05-Feb-1984
;  0051  0     !               Added state expression to parse OBJECT parameter as
;  0052  0     !               a number.
;  0053  0     !               Changed ACT$GL_NODADR_Q to more general name ACT$GL_ADR_Q.
;  0054  0     !
;  0055  0     !       V03-028 RPG0028         Bob Grosso              10-Jun-1983
;  0056  0     !               Add service device UNA.
;  0057  0     !
```

```
;  0058  0     !       V03-027 RPG0027         Bob Grosso              22-Mar-1983
;  0059  0     !               Turn off BLANKS after termination of state expression
;  0060  0     !               macro to parse NI addresses.
;  0061  0     !
;  0062  0     !       V03-026 RPG0026         Bob Grosso              16-Mar-1983
;  0063  0     !               Update NCP version number to IV.
;  0064  0     !               Complete state expression macro to parse NI addresses.
;  0065  0     !
;  0066  0     !       V03-025 RPG0025         Bob Grosso              10-Mar-1983
;  0067  0     !               Add state expression macro to parse NI addresses.
;  0068  0     !
;  0069  0     !       V03-024 RPG0024         Bob Grosso              25-Feb-1983
;  0070  0     !               Remove syntax checking for NODE id and correct
;  0071  0     !               parsing of circuit names.
;  0072  0     !               Note, this packet cannot be backed off without taking
;  0073  0     !               RPG0023 with it.
;  0074  0     !
;  0075  0     !       V03-023 RPG0023         Bob Grosso              18-Feb-1983
;  0076  0     !               Remove syntax checking for line-id and circuit-id.
;  0077  0     !               Change High range for node adr from 255 to 1023.
;  0078  0     !               Add high and low for LINE BFS.
;  0079  0     !               Add high and low for NODE FBS and SBS.
;  0080  0     !
;  0081  0     !       V03-022 RPG0022         Bob Grosso              20-Oct-1982
;  0082  0     !               Allow '$' and '_' in object names.  Allow 12 character
;  0083  0     !               object names.
;  0084  0     !               Have SE_NODE_ADR flag Area present in node address.
;  0085  0     !
;  0086  0     !       V03-021 RPG0021         Bob Grosso              23-Sep-1982
;  0087  0     !               Parse for node area.
;  0088  0     !               Range for Module Console RTR
;  0089  0     !
;  0090  0     !       V03-020 RPG0020         Bob Grosso              15-Sep-1982
;  0091  0     !               Increase tracepoint name length to 30 from 16.
;  0092  0     !
;  0093  0     !       V03-019 RPG0019         Bob Grosso              03-Sep-1982
;  0094  0     !               Add range for LIN RTT.
;  0095  0     !               Change range for MTR BSZ.
;  0096  0     !               Add SEM_HEX_NUM and LEN_HEX_NUM to parse hex numbers.
;  0097  0     !
;  0098  0     !       V03-018 TMH0018         Tim Halvorsen           16-Aug-1982
;  0099  0     !               Change tracepoint name parsing to accept a string of
;  0100  0     !               any size, including periods as legal characters.
;  0101  0     !
;  0102  0     !       V03-017 RPG0017         Bob Grosso              03-Aug-82
;  0103  0     !               Add range for Module X25-Protocol MCI.
;  0104  0     !               Change QUERY_STATES_S to allow different ALL prompt
;  0105  0     !               strings to support sub-databases.
;  0106  0     !
;  0107  0     !       V03-016 RPG0016         Bob Grosso              23-Jul-82
;  0108  0     !               Support X25-Trace with subexpression for tracepoint names,
;  0109  0     !               SEM_TRCPNT_NAME.
;  0110  0     !
;  0111  0     !       V015    RPG0015         Bob Grosso              14-Jul-82
;  0112  0     !               Add NI support in Set Node by adding range values for
;  0113  0     !               AMC, AMH, BRT, MAR, MBE, MBR.
;  0114  0     !
```

```
;  0115  0     !       V014    RPG0014         Bob Grosso              15-Jun-82
;  0116  0     !               Add MODULE parameter table.
;  0117  0     !               Add macro QUERY_STATES_S patterned after QUERY_STATES
;  0118  0     !               to permit alternate prompting within entities without
;  0119  0     !               having multiply defined states.
;  0120  0     !               Add Subexpression and constant for channels lists.
;  0121  0     !
;  0122  0     !       V013    TMH0013         Tim Halvorsen   05-Apr-1982
;  0123  0     !               Add ACT$TESTLONG action routine to ACT_DFN macro.
;  0124  0     !               Allow numeric characters in line/circuit mnemonic.
;  0125  0     !               Add circuit MRT and RPR ranges.
;  0126  0     !               Allow any characters following initial dash after
;  0127  0     !               line/circuit mnemonic (such as X25-CHICAGO).
;  0128  0     !
;  0129  0     !       V012    TMH0012         Tim Halvorsen   08-Jan-1982
;  0130  0     !               Remove TMH0005, thus restoring RETRANSMIT TIMER
;  0131  0     !               to a line parameter, which is what NM V3.0 finally
;  0132  0     !               came up with.
;  0133  0     !
;  0134  0     !       V011    TMH0011         Tim Halvorsen   31-Dec-1981
;  0135  0     !               Add DMF as a MOP service device.
;  0136  0     !
;  0137  0     !       V010    TMH0010         Tim Halvorsen   25-Nov-1981
;  0138  0     !               Allow embedded spaces in filespecs as long as they
;  0139  0     !               appear in double quotas (access control string).
;  0140  0     !               This allows access control strings to be specified
;  0141  0     !               in the filespec after the TO clause in the SHOW command.
;  0142  0     !
;  0143  0     !       V009    TMH0009         Tim Halvorsen   22-Oct-1981
;  0144  0     !               Fix HEX_PSW sub-expression so that blank which terminates
;  0145  0     !               hex password string does not get included in string.
;  0146  0     !
;  0147  0     !       V008    LMK0001         Len Kawell      19-Sep-1981
;  0148  0     !               Change NICE version to 3.0.
;  0149  0     !
;  0150  0     !       V007    TMH0007         Tim Halvorsen   28-Aug-1981
;  0151  0     !               Add macro to parse link ID
;  0152  0     !
;  0153  0     !       V006    TMH0006         Tim Halvorsen   15-Aug-1981
;  0154  0     !               Add DMP, DMV and DPV service devices.
;  0155  0     !               Add EXECUTOR PIPELINE QUOTA range.
;  0156  0     !
;  0157  0     !       V005    TMH0005         Tim Halvorsen   05-Aug-1981
;  0158  0     !               Change RETRANSMIT TIMER to a circuit parameter
;  0159  0     !               from a line parameter.
;  0160  0     !
;  0161  0     !       V004    TMH0004         Tim Halvorsen   07-Jul-1981
;  0162  0     !               Rename maximum blocks to maximum transmits
;  0163  0     !               Allow dashs in circuit names.
;  0164  0     !
;  0165  0     !       V003    TMH0003         Tim Halvorsen   11-Jun-1981
;  0166  0     !               Add ranges for new V2.2 circuit parameters.
;  0167  0     !               Remove obsolete line polling parameters.
;  0168  0     !               Change NCP version number to 2.2.0
;  0169  0     !
;  0170  0     !       V02-002 LMK0001 Len Kawell      18-Dec-1980
;  0171  0     !               Fix file-id parsing.
```

```
;   0172  0     !--
```

```
;    0173  0      %SBTTL  'Definitions'
;    0174  0
;    0175  0      !
;    0176  0      ! TABLE OF CONTENTS:
;    0177  0      !
;    0178  0
;    0179  0      !
;    0180  0      ! MACROS:
;    0181  0      !
;    0182  0
;    0183  0      !
;    0184  0      !       Program Identification String
;    0185  0      !
;    0186  0
;    0187  0      MACRO
; M  0188  0          PRG_ID_STR =
; M  0189  0
; M  0190  0              %STRING ('V3.00 ')
;    0191  0              %
;    0192  0              ,
;    0193  0
;    0194  0      !
;    0195  0      !       Build a cr lf pair in a string
;    0196  0      !
;    0197  0
; M  0198  0          CRLF =
; M  0199  0              %CHAR (13, 10)
;    0200  0              %
;    0201  0              ;
;    0202  0
;    0203  0      !
;    0204  0      ! $FAB_DEV - a macro which defines a single FAB$L_DEV bit.
;    0205  0      !
;    0206  0      !       $FAB_DEV( bit_name )
;    0207  0      !
;    0208  0      !       where:
;    0209  0      !           "bit_name" is a 3-character device bit name
;    0210  0      !
;    0211  0      !
;    0212  0
;    0213  0      MACRO
; M  0214  0          $FAB_DEV( BIT_NAME ) =
;    0215  0              FAB$DEV( FAB$L_DEV, %NAME('DEV$V_',BIT_NAME) ) %,
;    0216  0
;    0217  0          FAB$DEV( FAB_BYTE, FAB_BIT, FAB_SIZE, FAB_SIGN, DEV_DISP,
; M  0218  0                   DEV_BIT, DEV_SIZE, DEV_SIGN ) =
;    0219  0              FAB_BYTE, DEV_BIT, DEV_SIZE, DEV_SIGN %
;    0220  0          ;
```

```
;    0221  0
;    0222  0     !
;    0223  0     !       Create a descriptor for a constant string
;    0224  0     !
;    0225  0
;    0226  0     MACRO
;M   0227  0         ASCID [] =
;M   0228  0             (UPLIT BYTE                         ! Use byte alignment to save space
;M   0229  0                 (
;M   0230  0                 LONG (                          ! Parts must be longwords
;M   0231  0                     %CHARCOUNT( %STRING( %REMAINING)),
;M   0232  0                     UPLIT( %STRING( %REMAINING))
;M   0233  0                     )
;M   0234  0                 )
;    0235  0             )%
;    0236  0         ;
;    0237  0
;    0238  0     !
;    0239  0     !       Create pointer to counted string
;    0240  0     !
;    0241  0
;    0242  0     MACRO
;M   0243  0         ASCIC [] =
;M   0244  0             ( UPLIT BYTE (%ASCIC %STRING (%REMAINING) ) )
;    0245  0             %;
;    0246  0
;    0247  0     !
;    0248  0     ! Structure declarations used for system defined structures to
;    0249  0     ! save typing.  These structures are byte sized.
;    0250  0     !       (Thanks to A. Goldstein)
;    0251  0     !
;    0252  0
;    0253  0     STRUCTURE
;    0254  0             BBLOCK [O, P, S, E; N] =
;    0255  0                 [N]
;    0256  0                 (BBLOCK+O)<P,S,E>,
;    0257  0
;    0258  0             BBLOCKVECTOR [I, O, P, S, E; N, BS] =
;    0259  0                 [N*BS]
;    0260  0                 ((BBLOCKVECTOR+I*BS)+O)<P,S,E>
;    0261  0             ;
;    0262  0
;    0263  0
;    0264  0
;    0265  0     !
;    0266  0     !       Concatenate text to the control string
;    0267  0     !
;    0268  0
;    0269  0     MACRO
;M   0270  0             ADDSTR (TXT) =
;M   0271  0                     NCP$ADDSTR (ASCIC (TXT), NCP$GQ_CTRDSC)
;    0272  0             %;
;    0273  0
;    0274  0     !
;    0275  0     !       Add an entry to the fao list
;    0276  0     !
;    0277  0
```

```
;    0278  0     MACRO
;  M 0279  0             ADDFAO (ITEM) =
;  M 0280  0
;  M 0281  0                 NCP$ADDFAO (ITEM)
;    0282  0             %;
;    0283  0
```

```
;    0284  0      %SBTTL  'Macros to Build State Tables'
;    0285  0
;    0286  0      !
;    0287  0      !       Macros to help build state tables
;    0288  0      !
;    0289  0
;    0290  0      !
;    0291  0      !   For the following macros:
;    0292  0      !
;    0293  0      !   CLS         Code for the sub-command
;    0294  0      !   NAM         Parameter name
;    0295  0      !
;    0296  0
;    0297  0      !
;    0298  0      !   All state names have the form ST_CLS_...
;    0299  0      !   There are two types of states, prompt and process. Prompt states
;    0300  0      !   sequence the prompts for parameters.  Process states allow any
;    0301  0      !   parameter in any order.
;    0302  0      !
;    0303  0
;    0304  0      !
;    0305  0      !       Build a sequence of prompt states
;    0306  0      !       A prompt is printed and then it is parsed.  No answer is required
;    0307  0      !       and if none is given the next prompt is issued.  If the response is
;    0308  0      !       "_DONE" then the remainder of the prompts are skipped and the
;    0309  0      !       function is performed.
;    0310  0      !
;    0311  0
;    0312  0      MACRO
;  M 0313  0          PROMPT_STATES (CLS) [NAM] =
;  M 0314  0
;  M 0315  0      $STATE  (%NAME ('ST_', CLS, '_PMT_', NAM),
;  M 0316  0              (TPA$_LAMBDA, ,
;  M 0317  0                  ACT$PRMPT, , , %NAME ('PMT$G_', CLS, '_', NAM) )
;  M 0318  0              );
;  M 0319  0
;  M 0320  0      $STATE  (,
;  M 0321  0              (TPA$_SYMBOL, %NAME ('ST_', CLS, '_DOIT'), ACT$PMTDONEQ ),
;  M 0322  0              ( (%NAME ('ST_', CLS, '_', NAM) ) ),
;  M 0323  0              (TPA$_EOS),
;  M 0324  0              (TPA$_LAMBDA, %NAME ('ST_', CLS, '_PMT_', NAM),
;  M 0325  0                      ACT$SIGNAL, , , NCP$_INVVAL)
;  M 0326  0              );
;    0327  0          %;
```

```
;    0328  0
;    0329  0     !
;    0330  0     !       Build a pair of states to accomplish command prompting
;    0331  0     !
;    0332  0     !       The idea is to cause prompting only if the state is entered
;    0333  0     !       with TPA$_EOS true.  If prompting is true, then the state should
;    0334  0     !       loop until either a transition is satisfied or the command is
;    0335  0     !       canceled.  This is done by using ACT$PMT_ON and _OFF to remember
;    0336  0     !       the state of prompting and ACT$PMT_Q to act on that state to
;    0337  0     !       either fail (not prompting) or succeed and issue and error message
;    0338  0     !       (prompting).
;    0339  0     !
;    0340  0
;    0341  0     MACRO
;  M 0342  0             COMMAND_PROMPT (CLS, NAM, STATUS) =
;  M 0343  0
;  M 0344  0     $STATE  (%NAME ('ST_', CLS, '_', NAM),
;  M 0345  0             (TPA$_EOS, , ACT$PMT_ON),
;  M 0346  0             (TPA$_LAMBDA, , ACT$PMT_OFF),
;  M 0347  0             );
;  M 0348  0
;  M 0349  0     $STATE  (%NAME ('ST_', CLS, '_', NAM, '_1'),
;  M 0350  0
;  M 0351  0             %REMAINING
;  M 0352  0
;  M 0353  0
;  M 0354  0             (TPA$_EOS, %NAME ('ST_', CLS, '_', NAM, '_1'),
;  M 0355  0                     ACT$PRMPT, , %NAME ('PMT$G_', CLS, '_', NAM) ),
;  M 0356  0             (TPA$_LAMBDA, %NAME ('ST_', CLS, '_', NAM, '_1'),
;  M 0357  0                     ACT$PMT_Q, , , STATUS)
;  M 0358  0             );
;  M 0359  0
;    0360  0             %;
```

```
;    0361  0
;    0362  0     !
;    0363  0     !       Build sequence of Query states
;    0364  0     !       Query states are states which save a parameter
;    0365  0     !       if the answer to a prompt is YES. No parameter is
;    0366  0     !       saved for NO or CR.  If the response is "_DONE" then
;    0367  0     !       the remainder of the queries are skipped and the function
;    0368  0     !       is performed.
;    0369  0     !
;    0370  0
;    0371  0     MACRO
;  M 0372  0             QUERY_STATES (CLS) [NAM] =
;  M 0373  0
;  M 0374  0     $STATE  (%NAME ('ST_', CLS, '_PMT_', NAM),
;  M 0375  0             (TPA$_LAMBDA, , ACT$PRMPT,
;  M 0376  0                     %NAME ('PMT$G_', CLS, '_', NAM) )
;  M 0377  0             );
;  M 0378  0
;  M 0379  0     $STATE  (,
;  M 0380  0             (TPA$_SYMBOL, %NAME ('ST_', CLS, '_DOIT'), ACT$PMTDONEQ ),
;  M 0381  0             ( (SE_QRY_YES),
;  M 0382  0                     %IF     %IDENTICAL (NAM, ALL)           ! ALL IS SPECIAL
;  M 0383  0                     %THEN   %NAME ('ST_', CLS, '_DOIT')     ! IT MUST BE LAST
;  M 0384  0                     %FI
;  M 0385  0                     , ACT$SAVPRM,
;  M 0386  0                     %NAME ('PBK$G_', CLS, '_', NAM) ),
;  M 0387  0             ( (SE_QRY_NO) ),
;  M 0388  0             (TPA$_EOS),
;  M 0389  0             (TPA$_LAMBDA, %NAME ('ST_', CLS, '_PMT_', NAM),
;  M 0390  0                     ACT$SIGNAL, , , NCP$_INVVAL)
;  M 0391  0             );
;  M 0392  0
;    0393  0             %;
;    0394  0
;    0395  0     !
;    0396  0     !       Slightly modified QUERY_STATES macro to permit using
;    0397  0     !       same prompt and PBK more than once with multiply defining
;    0398  0     !       parse table states.
;    0399  0     !
;    0400  0
;    0401  0     MACRO
;  M 0402  0             QUERY_STATES_S (CLS) [NAM, SNAM] =
;  M 0403  0
;  M 0404  0     $STATE  (%NAME ('ST_', CLS, '_PMT_', SNAM),
;  M 0405  0             (TPA$_LAMBDA, , ACT$PRMPT,
;  M 0406  0                     %NAME ('PMT$G_', CLS, '_', SNAM) )
;  M 0407  0             );
;  M 0408  0
;  M 0409  0     $STATE  (,
;  M 0410  0             (TPA$_SYMBOL, %NAME ('ST_', CLS, '_DOIT'), ACT$PMTDONEQ ),
;  M 0411  0             ( (SE_QRY_YES),
;  M 0412  0                     %IF     %IDENTICAL (NAM, ALL)           ! ALL IS SPECIAL
;  M 0413  0                     %THEN   %NAME ('ST_', CLS, '_DOIT')     ! IT MUST BE LAST
;  M 0414  0                     %FI
;  M 0415  0                     , ACT$SAVPRM,
;  M 0416  0                     %NAME ('PBK$G_', CLS, '_', NAM) ),
;  M 0417  0             ( (SE_QRY_NO) ),
```

```
;  M 0418  0               (TPA$_EOS),
;  M 0419  0               (TPA$_LAMBDA, %NAME ('ST_', CLS, '_PMT_', SNAM),
;  M 0420  0                           ACT$SIGNAL, , , NCP$_INVVAL)
;  M 0421  0               );
;  M 0422  0
;    0423  0               %;
;    0424  0
;    0425  0
;    0426  0           !   Build transitions in a dispatch state
;    0427  0           !
;    0428  0           !   KEY         Keyword for dispatch from state
;    0429  0           !
;    0430  0
;    0431  0           MACRO
;  M 0432  0               DISPATCH_STATES (CLS) [NAM, KEY] =
;  M 0433  0
;  M 0434  0                   (%STRING (KEY), %NAME ('ST_', CLS, '_PRC_', NAM) )
;    0435  0               %;
```

```
;    0436  0
;    0437  0     !
;    0438  0     !       Build a sequence of process states
;    0439  0     !
;    0440  0     !   NOISE       Noise keyword
;    0441  0     !
;    0442  0
;    0443  0     MACRO
;M   0444  0         PROCESS_STATES (CLS) [NAM, NOISE] =
;M   0445  0
;M   0446  0     $STATE  (%NAME ('ST_', CLS, '_PRC_', NAM),
;M   0447  0             %IF NOT %NULL (NOISE)
;M   0448  0             %THEN
;M   0449  0             (%STRING (NOISE)),
;M   0450  0             (TPA$_LAMBDA)
;M   0451  0             );
;M   0452  0     $STATE  (,
;M   0453  0             %FI
;M   0454  0             ( (%NAME ('ST_', CLS, '_', NAM) ),
;M   0455  0                 %NAME ('ST_', CLS, '_PRC') )
;M   0456  0             );
;    0457  0         %;
;    0458  0
;    0459  0     !
;    0460  0     !       Build a set of subexpressions to decode parameters
;    0461  0     !
;    0462  0     !   TYP         Type of transition desired
;    0463  0     !
;    0464  0
;    0465  0     MACRO
;M   0466  0         SUB_EXPRESSIONS (CLS) [NAM, TYP] =
;M   0467  0
;M   0468  0     $STATE  (%NAME ('ST_', CLS, '_', NAM),
;M   0469  0             (TYP,
;M   0470  0
;M   0471  0             %IF %IDENTICAL (TYP, TPA$_DECIMAL)
;M   0472  0             %THEN
;M   0473  0                 , ACT$NUM_RNG, ,
;M   0474  0                             NUM_RANGE
;M   0475  0                             (
;M   0476  0                             %NAME ('LOW_', CLS, '_', NAM),
;M   0477  0                             %NAME ('HIGH_', CLS, '_', NAM)
;M   0478  0                             )
;M   0479  0                     )
;M   0480  0             );
;M   0481  0     $STATE  (,
;M   0482  0             (TPA$_LAMBDA,
;M   0483  0
;M   0484  0             %FI
;M   0485  0
;M   0486  0             TPA$_EXIT, ACT$SAVPRM,
;M   0487  0                 %NAME ('PBK$G_', CLS, '_', NAM) )
;M   0488  0             );
;    0489  0         %;
;    0490  0
;    0491  0     !
;    0492  0     !       Build transitions in a keyword state
```

```
!
!       Each transition saves a parameter based on the keyword
!       and exits the subexpression.
!

MACRO
        KEYWORD_STATE (CLS) [NAM, KEY] =

        (%STRING (KEY), TPA$_EXIT, ACT$SAVPRM, , ,
                %NAME ('PBK$G_', CLS, '_', NAM) )
        %;
```

```
;    0504  0      %SBTTL  'Macro to Build Prompt Strings'
;    0505  0
;    0506  0      !
;    0507  0      !       Build prompt strings
;    0508  0      !
;    0509  0
;    0510  0      MACRO
;M   0511  0          PROMPT_STRINGS (CLS) [NAM, STR] =
;M   0512  0
;M   0513  0      %NAME ('PMT$G_', CLS, '_', NAM) =
;M   0514  0              ASCID (%STRING (STR) )
;    0515  0          %;
```

```
;    0516  0     %SBTTL  'Macros to Build Parameter Control Blocks'
;    0517  0
;    0518  0     !
;    0519  0     !       Build parameter blocks
;    0520  0     !
;    0521  0
;    0522  0     !       There are four structures associated with building messages:
;    0523  0     !
;    0524  0     !               SDB     Set/Define Block
;    0525  0     !
;    0526  0     !       This block is a parameter to the verb routines.  It serves
;    0527  0     !       to point to other structures and to declare the type of the
;    0528  0     !       entity so that message headers can be properly built.
;    0529  0     !
;    0530  0     !               PDB     Parameter Data Block
;    0531  0     !
;    0532  0     !       This is a data area which holds the actual parameter data.
;    0533  0     !       THe block is a status byte followed by the data as it
;    0534  0     !       appears in the message.  The action routine ACT$SAVPRM
;    0535  0     !       stores the data in this block in the correct format.
;    0536  0     !
;    0537  0     !               PBK     Parameter Block
;    0538  0     !
;    0539  0     !       This block is a parameter to ACT$SAVPRM and directs the
;    0540  0     !       storage of the parameter in the PDB.  It contains the type
;    0541  0     !       of the parameter, the PDB address and an optional parameter
;    0542  0     !       for the type code.
;    0543  0     !
;    0544  0     !               PCL     Parameter Control List
;    0545  0     !
;    0546  0     !       This block is a list of items which control the building of
;    0547  0     !       messages.  Each entry is a parameter type code, the
;    0548  0     !       parameter ID code and the PDB address.  Using this block the
;    0549  0     !       routines which build messages are able to add parameter
;    0550  0     !       values or codes to the end of messages in the proper format.
;    0551  0     !
```

```
;    0552  0
;    0553  0        !
;    0554  0        !       Build the SDB
;    0555  0        !
;    0556  0
;    0557  0        !
;    0558  0        !       CLS     Class of the command
;    0559  0        !       ENT     Entity type code.  If negative, then system-specific entity
;    0560  0        !       PDB     Parameter data block suffix
;    0561  0        !       PCL     PCL suffix
;    0562  0        !
;    0563  0
;    0564  0        MACRO
;  M 0565  0            BUILD_SDB (CLS, ENT, PDB, PCL) =
;  M 0566  0
;  M 0567  0        !
;  M 0568  0        !   Declare symbols which are not yet declared
;  M 0569  0        !
;  M 0570  0
;  M 0571  0            %IF NOT %DECLARED (%NAME ('PDB$G_', PDB) )
;  M 0572  0            %THEN
;  M 0573  0                EXTERNAL
;  M 0574  0                %NAME ('PDB$G_', PDB)
;  M 0575  0                ;
;  M 0576  0            %FI
;  M 0577  0
;  M 0578  0        !
;  M 0579  0        !   Build the PLIT for the SDB
;  M 0580  0        !
;  M 0581  0
;  M 0582  0            BIND
;  M 0583  0
;  M 0584  0            %NAME ('SDB$G_', CLS) =
;  M 0585  0
;  M 0586  0            UPLIT BYTE                                  ! Use byte alignment to
;  M 0587  0                (                                       ! Save space
;  M 0588  0                BYTE (ENT),
;  M 0589  0                LONG (%NAME ('PDB$G_', PDB) ),
;  M 0590  0                LONG (%NAME ('PCL$G_', PCL) )
;  M 0591  0                )
;  M 0592  0            ;
;    0593  0            %;
```

```
;    0594  0
;    0595  0     !
;    0596  0     !       Build a PCL
;    0597  0     !
;    0598  0
;    0599  0     !
;    0600  0     !       CLS     Class of command
;    0601  0     !
;    0602  0     !       remaining repeated
;    0603  0     !
;    0604  0     !       NAM     Name of parameter concerned
;    0605  0     !       TYP     Suffix for type code
;    0606  0     !       ID      Suffix for parameter ID code
;    0607  0     !       PDB     Suffix for PDB of data
;    0608  0     !
;    0609  0
;    0610  0     MACRO
;  M 0611  0         BUILD_PCL (CLS) =
;  M 0612  0
;  M 0613  0     !
;  M 0614  0     !   Declare the PDB's
;  M 0615  0     !
;  M 0616  0
;  M 0617  0         BUILD_PCL_PDB (CLS, %REMAINING)
;  M 0618  0
;  M 0619  0     !
;  M 0620  0     !   Build the PCL PLIT
;  M 0621  0     !
;  M 0622  0
;  M 0623  0         BIND
;  M 0624  0
;  M 0625  0         %NAME ('PCL$G_',CLS) =
;  M 0626  0
;  M 0627  0         UPLIT BYTE                          ! Use byte alignment to save space
;  M 0628  0             (
;  M 0629  0             BUILD_PCL_LST (CLS, %REMAINING)
;  M 0630  0             )
;  M 0631  0         ;
;    0632  0         %,
;    0633  0
;    0634  0     !
;    0635  0     !   Build the items in the PCL list
;    0636  0     !
;    0637  0
;  M 0638  0         BUILD_PCL_LST (CLS) [NAM, TYP, ID, PDB] =
;  M 0639  0
;  M 0640  0         BYTE (%NAME ('PBK$K_', TYP) ),      ! Data type code
;  M 0641  0         WORD (
;  M 0642  0             %IF %NULL (ID)                  ! Network management ID
;  M 0643  0             %THEN 0
;  M 0644  0             %ELSE %NAME ('NMA$C_',ID)
;  M 0645  0             %FI
;  M 0646  0             ),
;  M 0647  0         LONG (                              ! Address of PDB
;  M 0648  0             %IF %NULL (NAM)
;  M 0649  0             %THEN 0
;  M 0650  0             %ELSE %NAME ('PDB$G_',
```

```
;  M 0651  0                        %IF %NULL (PDB)
;  M 0652  0                        %THEN CLS, '_', NAM
;  M 0653  0                        %ELSE PDB
;  M 0654  0                        %FI
;  M 0655  0                        )
;  M 0656  0                    %FI
;  M 0657  0                    )
;  M 0658  0
;    0659  0                %,
;    0660  0
;    0661  0        !
;    0662  0        !   Declare the PDB as external
;    0663  0        !
;    0664  0
;  M 0665  0                BUILD_PCL_PDB (CLS) [NAM, I2, I3, PDB] =
;  M 0666  0
;  M 0667  0                %IF NOT %NULL (NAM)
;  M 0668  0                %THEN
;  M 0669  0                    %IF NOT %DECLARED
;  M 0670  0                          (%NAME ('PDB$G_',
;  M 0671  0                                %IF %NULL (PDB)
;  M 0672  0                                %THEN CLS, '_', NAM
;  M 0673  0                                %ELSE PDB
;  M 0674  0                                %FI
;  M 0675  0                                )
;  M 0676  0                          )
;  M 0677  0                    %THEN
;  M 0678  0
;  M 0679  0                        EXTERNAL
;  M 0680  0                              %NAME ('PDB$G_',
;  M 0681  0                                    %IF %NULL (PDB)
;  M 0682  0                                    %THEN CLS, '_', NAM
;  M 0683  0                                    %ELSE PDB
;  M 0684  0                                    %FI
;  M 0685  0                                    )
;  M 0686  0                                ;
;  M 0687  0                    %FI
;  M 0688  0                %FI
;    0689  0                %;
```

```
;    0690  0
;    0691  0     !
;    0692  0     !       Build a list of PBK's
;    0693  0     !
;    0694  0
;    0695  0     !
;    0696  0     !       CLS     Class of command
;    0697  0     !
;    0698  0     !       remaining are repeated
;    0699  0     !
;    0700  0     !       NAM     Suffix name of parameter
;    0701  0     !       TYP     Suffix of type code of parameter
;    0702  0     !       PRM     Value of type code parameter
;    0703  0     !       PDB     Suffix for PDB to save parameter
;    0704  0     !
;    0705  0
;    0706  0
;    0707  0     MACRO
;M   0708  0         BUILD_PBK (CLS) [NAM, TYP, PRM, PDB] =
;M   0709  0
;M   0710  0         %IF NOT %DECLARED                               ! Declare the pdb external
;M   0711  0             (%NAME ('PDB$G_',
;M   0712  0                     %IF %NULL (PDB)
;M   0713  0                     %THEN CLS, '_', NAM
;M   0714  0                     %ELSE PDB
;M   0715  0                     %FI
;M   0716  0                 )
;M   0717  0             )
;M   0718  0         %THEN
;M   0719  0
;M   0720  0             EXTERNAL
;M   0721  0                 %NAME ('PDB$G_',
;M   0722  0                         %IF %NULL (PDB)
;M   0723  0                         %THEN CLS, '_', NAM
;M   0724  0                         %ELSE PDB
;M   0725  0                         %FI
;M   0726  0                     )
;M   0727  0                 ;
;M   0728  0
;M   0729  0         %FI
;M   0730  0
;M   0731  0         GLOBAL BIND                                     ! Build PBK as a plit
;M   0732  0
;M   0733  0         %NAME ('PBK$G_', CLS, '_', NAM) =
;M   0734  0
;M   0735  0         UPLIT BYTE                                      ! Use byte alignment to save space
;M   0736  0             (
;M   0737  0             BYTE (%NAME ('PBK$K_', TYP) ),              ! Data type code
;M   0738  0             LONG (%NAME ('PDB$G_',                      ! PDB address
;M   0739  0                         %IF %NULL (PDB)
;M   0740  0                         %THEN CLS, '_', NAM
;M   0741  0                         %ELSE PDB
;M   0742  0                         %FI
;M   0743  0                     )
;M   0744  0                 ),
;M   0745  0             LONG (                                      ! Parameter for type code routine
;M   0746  0                 %IF %NULL (PRM)
```

```
;  M 0747  0                    %THEN 0
;  M 0748  0                    %ELSE PRM
;  M 0749  0                    %FI
;  M 0750  0                    )
;  M 0751  0                )
;  M 0752  0
;  M 0753  0            ;
;    0754  0            %;
```

```
;    0755  0
;    0756  0     !
;    0757  0     !       Build a PDB
;    0758  0     !
;    0759  0
;    0760  0     !
;    0761  0     !       CLS     Class of command
;    0762  0     !       NAM     Suffix for parameter
;    0763  0     !       SIZ     Size of parameter data in bytes
;    0764  0     !
;    0765  0
;    0766  0     MACRO
;M   0767  0         BUILD_PDB (CLS) [NAM, SIZ] =
;M   0768  0
;M   0769  0         %NAME ('PDB$G_', CLS, '_', NAM) :           ! Name in classic form
;M   0770  0                 BLOCK [(SIZ) + 1, 1]                ! Size + 1 for status byte
;M   0771  0                 ALIGN (0)                           ! Byte align to save space
;M   0772  0
;    0773  0         %;
```

```
;    0774  0
;    0775  0     !
;    0776  0     !       Build a numeric range parameter
;    0777  0     !
;    0778  0
;    0779  0     MACRO
;M   0780  0         NUM_RANGE (LOW, HIGH) =
;M   0781  0
;M   0782  0             ( UPLIT BYTE ( LONG ( (LOW), (HIGH) ) ) ) ! Byte align to save space
;    0783  0             %;
;    0784  0
;    0785  0     !
;    0786  0     !       Build a next state parameter
;    0787  0     !
;    0788  0
;    0789  0     MACRO
;M   0790  0             NEXT_STATE (COD) =
;M   0791  0             (UPLIT BYTE                     ! Byte align to save space
;M   0792  0                     (
;M   0793  0                     LONG                    ! Each is an address in a longword
;M   0794  0                         (
;M   0795  0                         %NAME ('NCP$G_STTBL_',COD),
;M   0796  0                         %NAME ('NCP$G_KYTBL_', COD)
;M   0797  0                         )
;M   0798  0                     )
;M   0799  0             )
;    0800  0             %;
;    0801  0
```

```
;    0802  0      %SBTTL  'Equated Symbols'
;    0803  0
;    0804  0      !
;    0805  0      ! EQUATED SYMBOLS:
;    0806  0      !
;    0807  0
;    0808  0      LITERAL
;    0809  0              TRUE            = 1,
;    0810  0              FALSE           = 0,
;    0811  0              SUCCESS         = 1,
;    0812  0              FAILURE         = 0,
;    0813  0
;    0814  0              NCP$C_VRS       = 4,    ! Version of NCP for messages
;    0815  0              NCP$C_ECO       = 0,    ! Eco for messages
;    0816  0              NCP$C_UECO      = 0,    ! User eco for messages
;    0817  0
;    0818  0              NCP$C_MBXSIZ = 40,      ! Size of the mailbox buffer for network io
;    0819  0              NCP$C_RSPSIZ = 1000,    ! Size of the response buffer for network io
;    0820  0
;    0821  0              LEN_OBJ_NAM     = 12,   ! Length of an object name
;    0822  0              LEN_ID_STR      = 32,   ! Length of an ID string
;    0823  0              LEN_NSP_PSW     = 8,    ! Length of a nsp password
;    0824  0              LEN_FILE_SPEC   = 64,   ! Length of a file spec
;    0825  0              LEN_FILE_NAM    = 9,    ! Length of a file name
;    0826  0              LEN_FILE_TYP    = 3,    ! Length of a file type
;    0827  0              LOW_NODE_ADR    = 0,    ! Low limit of node address
;    0828  0              HIGH_NODE_ADR   = 1023, ! High limit
;    0829  0              LEN_NODE_NAM    = 6,    ! Length of node name
;    0830  0              LEN_NI_ADR      = 6,    ! Length of NI Address
;    0831  0              LOW_AREA        = 1,    ! Low limit of a node area
;    0832  0              HIGH_AREA       = 65,   ! High limit
;    0833  0              LEN_CIRC_ID     = 16,   ! Length of a circuit id
;    0834  0              LEN_LINE_ID     = 16,   ! Length of a total line id
;    0835  0              LEN_HEX_NUM     = 32,   ! Length of Hex number (128 bits)
;    0836  0              LEN_HEX_PSW     = 16,   ! Length of Hex password (64 bits)
;    0837  0              LEN_ACC_ACC     = 39,   ! Length of the access account
;    0838  0              LEN_ACC_PSW     = 39,   ! Length of the access password
;    0839  0              LEN_ACC_USR     = 39,   ! Length of the access user id
;    0840  0              LOW_EVENT_CLS   = 0,    ! Low limit of event class
;    0841  0              HIGH_EVENT_CLS  = 511,  ! High limit
;    0842  0              LOW_EVENT_TYP   = 0,    ! Low limit of event type
;    0843  0              HIGH_EVENT_TYP  = 31,   ! High limit
;    0844  0              LEN_PRV_MSK     = 8,    ! Length in bytes of a priv mask
;    0845  0              LEN_SOFT_ID     = 16,   ! Length of a node software id
;    0846  0              LOW_UIC_PART    = 0,    ! Low limit of uic number
;    0847  0              HIGH_UIC_PART   = 255,  ! High limit
;    0848  0              LEN_DTE_NUM     = 16,   ! Length of X.25 circuit DTE address
;    0849  0              LEN_ENT_NAM     = 16,   ! Length of entity name
;    0850  0              LEN_GRP_NAME    = 16,   ! Length of X.25 closed user group name
;    0851  0              LEN_NET_NAME    = 16,   ! Length of X.25 network name
;    0852  0              LEN_DEST_NAME   = 16,   ! Length of X.25 destination name
;    0853  0              LEN_TRCPNT_NAME = 31,   ! Length of X.25 tracepoint name
;    0854  0              MAX_RNGLST_PAIRS= 16;   ! Maximum numbers of pairs in a range list
```

```
;    0855  0
;    0856  0     !
;    0857  0     !       Macro to help define ranges
;    0858  0     !
;    0859  0
;    0860  0     MACRO
;M   0861  0             DEFRNG (CLS) [NAM, LO, HI] =
;M   0862  0
;M   0863  0             LITERAL
;M   0864  0                     %NAME ('HIGH_', CLS, '_', NAM) = HI,
;M   0865  0                     %NAME ('LOW_', CLS, '_', NAM) = LO
;M   0866  0                     ;
;    0867  0             %;
;    0868  0
;    0869  0
;P   0870  0     DEFRNG (NOD,                    ! Executor node parameters
;P   0871  0
;P   0872  0             ADR, 0, 1023,           ! Node address
;P   0873  0             AMC, 1, 65535,          ! Area maximum cost
;P   0874  0             AMH, 1, 255,            ! Area maximum hops
;P   0875  0             BRT, 1, 65535,          ! Broadcast routing timer
;P   0876  0             BSZ, 1, 65535,          ! Buffer size
;P   0877  0             DFC, 1, 255,            ! Delay factor
;P   0878  0             DWT, 1, 255,            ! Delay weight
;P   0879  0             FBS, 1, 65535,          ! Forwarding buffer size
;P   0880  0             IAT, 1, 65535,          ! Inactivity timer
;P   0881  0             INT, 1, 65535,          ! Incoming timer
;P   0882  0             MAD, 1, 65535,          ! Max address
;P   0883  0             MAR, 1, 255,            ! Max area
;P   0884  0             MBE, 1, 65535,          ! Max broadcast nonrouters
;P   0885  0             MBR, 1, 65535,          ! Max broadcast routers
;P   0886  0             MBF, 0, 65535,          ! Max buffers
;P   0887  0             MCO, 1, 1023,           ! Max cost
;P   0888  0             MHP, 1, 31,             ! Max hops
;P   0889  0             MLN, 1, 65535,          ! Max lines
;P   0890  0             MLK, 1, 65535,          ! Max links
;P   0891  0             MVS, 1, 255,            ! Max visits
;P   0892  0             OTM, 1, 65535,          ! Outgoing timer
;P   0893  0             RFC, 1, 65535,          ! Retransmit factor
;P   0894  0             RTM, 1, 65535,          ! Routing timer
;P   0895  0             SBS, 1, 65535,          ! Segment buffer size
;    0896  0             PIQ, 0, 65535)          ! Pipeline quota
;    0897  0
;    0898  0
;P   0899  0     DEFRNG (CIR,                    ! Circuit parameters
;P   0900  0
;P   0901  0             CTM, 1, 65535,          ! Counter timer
;P   0902  0             COS, 1, 25,             ! Cost
;P   0903  0             MRT, 0, 255,            ! Maximum routers on NI
;P   0904  0             RPR, 0, 127,            ! Router priority on NI
;P   0905  0             HET, 1, 65535,          ! Hello timer
;P   0906  0             LIT, 1, 65535,          ! Listen timer
;P   0907  0             MRC, 0, 255,            ! Maximum recalls
;P   0908  0             RCT, 1, 65535,          ! Recall timer
;P   0909  0             CHN, 0, 4095,           ! Channel number
;P   0910  0             MBL, 1, 65535,          ! Maximum block
;P   0911  0             MWI, 1, 255,            ! Maximum window
```

```
;  P 0912  0              TRI, 0, 255,                ! Tributary address
;  P 0913  0              BBT, 1, 65535,              ! Babble timer
;  P 0914  0              TRT, 0, 65535,              ! Transmit timer
;  P 0915  0              MTR, 1, 255,                ! Maximum transmits
;  P 0916  0              ACB, 0, 255,                ! Active base
;  P 0917  0              ACI, 0, 255,                ! Active increment
;  P 0918  0              IAB, 0, 255,                ! Inactive base
;  P 0919  0              IAI, 0, 255,                ! Inactive increment
;  P 0920  0              IAT, 0, 255,                ! Inactive threshold
;  P 0921  0              DYB, 0, 255,                ! Dying base
;  P 0922  0              DYI, 0, 255,                ! Dying increment
;  P 0923  0              DYT, 0, 255,                ! Dying threshold
;    0924  0              DTH, 0, 255)                ! Dead threshold
;    0925  0
;    0926  0
;  P 0927  0      DEFRNG (LIN,                       ! Line parameters
;  P 0928  0
;  P 0929  0              CTM, 1, 65535,              ! Counter timer
;  P 0930  0              BLO, 0, 65535,              ! Block size
;  P 0931  0              COS, 1, 25,                 ! Cost of the line
;  P 0932  0              NTM, 1, 65535,              ! Normal timer
;  P 0933  0              STM, 1, 65535,              ! Service timer
;  P 0934  0              RTT, 1, 65535,              ! Retransmit timer
;  P 0935  0              HTI, 1, 65535,              ! Holdback timer
;  P 0936  0              MBL, 1, 65535,              ! Maximum block
;  P 0937  0              MRT, 1, 255,                ! Maximum retransmits
;  P 0938  0              MWI, 1, 255,                ! Maximum window
;  P 0939  0              TRB, 0, 255,                ! Tributary address
;  P 0940  0              SLT, 50, 65535,             ! Scheduling timer
;  P 0941  0              DDT, 1, 65535,              ! Dead timer
;  P 0942  0              DLT, 1, 65535,              ! Delay timer
;  P 0943  0              SRT, 0, 65535,              ! Stream timer
;  P 0944  0              BFN, 1, 1024,               ! Number of buffers
;    0945  0              BFS, 1, 65535)              ! Buffer size
;    0946  0
;    0947  0
;  P 0948  0      DEFRNG (LOO,                       ! Loop parameters
;  P 0949  0
;  P 0950  0              CNT, 1, 65535,              ! Count of messages
;    0951  0              LEN, 1, 65535)              ! Length of message in bytes
;    0952  0
;    0953  0
;  P 0954  0      DEFRNG (LNK,                       ! Link parameter
;  P 0955  0
;    0956  0              ADR, 1, 65535)              ! Link address
;    0957  0
;    0958  0
;  P 0959  0      DEFRNG (NOD,                       ! Node parameters
;  P 0960  0
;  P 0961  0              CTM, 1, 65535,              ! Counter timer
;    0962  0              DCT, 0, %X'FFFFFFFF')       ! Dump count
;    0963  0
;    0964  0
;  P 0965  0      DEFRNG (DUM,
;  P 0966  0
;    0967  0              COU, 0, %X'FFFFFFFF')       ! Dump count
;    0968  0
```

```
;    0969  0
;  P 0970  0      DEFRNG (OBJ,                    ! Object parameters
;  P 0971  0
;    0972  0              NUM, 0, 255)            ! Object number
;    0973  0
;  P 0974  0      DEFRNG (MCS,                    ! Module Console
;  P 0975  0
;    0976  0              RTR, 0, 65535)          ! Object number
;    0977  0
;  P 0978  0      DEFRNG (MPR,                    ! X25-PROTOCOL
;  P 0979  0
;  P 0980  0              CTM, 1, 65535,          ! Counter timer
;  P 0981  0              DBL, 1, 65535,          ! Default block
;  P 0982  0              DWI, 1, 127,            ! Default window
;  P 0983  0              MBL, 16, 4096,          ! Maximum block
;  P 0984  0              MWI, 1, 127,            ! Maximum window
;  P 0985  0              MCL, 1, 255,            ! Maximum clears
;  P 0986  0              MRS, 1, 255,            ! Maximum resets
;  P 0987  0              MST, 1, 255,            ! Maximum restarts
;  P 0988  0              CAT, 1, 255,            ! Call timer
;  P 0989  0              CLT, 1, 255,            ! Clear timer
;  P 0990  0              RST, 1, 255,            ! Reset timer
;  P 0991  0              STT, 1, 255,            ! Restart timer
;  P 0992  0              GNM, 0, 9999,           ! Closed user group number
;  P 0993  0              MCI, 1, 65535           ! Maximum circuits - VMS specific
;    0994  0              )
;    0995  0
;  P 0996  0      DEFRNG (MSE,                    ! X25-SERVER
;  P 0997  0
;  P 0998  0              CTM, 1, 65535,          ! Counter timer
;  P 0999  0              MCI, 1, 65535,          ! Maximum circuits
;    1000  0              PRI, 0, 255)            ! Priority
;    1001  0
;  P 1002  0      DEFRNG (MTR,                    ! X25-TRACE
;  P 1003  0
;  P 1004  0              BSZ, 1, 4096,           ! Buffer size
;  P 1005  0              CPL, 1, 65535,          ! Capture limit
;  P 1006  0              CPS, 1, 65535,          ! Capture size
;  P 1007  0              MBK, 1, 65535,          ! Maximum blocks
;  P 1008  0              MBF, 1, 65535,          ! Maximum buffers
;    1009  0              MVR, 1, 63)             ! Maximum versions
```

```
;    1010  0      %SBTTL  'Macro to Define External Symbols'
;    1011  0
;    1012  0      !
;    1013  0      ! EXTERNAL REFERENCES:
;    1014  0      !
;    1015  0
;    1016  0      !
;    1017  0      !       Define externals for action routines
;    1018  0      !
;    1019  0
;    1020  0      MACRO
;  M 1021  0              ACT_DFN =
;  M 1022  0
;  M 1023  0      EXTERNAL ROUTINE
;  M 1024  0              ACT$INV_COMMAND,                ! Signal invalid command
;  M 1025  0              ACT$SAVPRM,                     ! Save a parameter
;  M 1026  0              ACT$TMPSTR,                     ! Save a temporary string
;  M 1027  0              ACT$BLNK_SIG,                   ! Blanks are now significant
;  M 1028  0              ACT$BLNK_NSIG,                  ! Blanks are not significant
;  M 1029  0              ACT$ZAPTMPDSC,                  ! Clear temporary descriptors
;  M 1030  0              ACT$PRMPT,                      ! Prompt for a parameter
;  M 1031  0              ACT$NUM_RNG,                    ! Validate a number
;  M 1032  0              ACT$NUM_RNGSAV,                 ! Validate and store range list number
;  M 1033  0              ACT$NUM_SAV,                    ! Store a number from a range list
;  M 1034  0              ACT$STR_LEN,                    ! Validate a string length
;  M 1035  0              ACT$WRI_STR,                    ! Write a string to SYS$OUTPUT
;  M 1036  0              ACT$SIGNAL,                     ! Signal an error condition
;  M 1037  0              ACT$PMT_ON,                     ! Prompting on
;  M 1038  0              ACT$PMT_OFF,                    ! Prompting off
;  M 1039  0              ACT$PMT_Q,                      ! Check prompting
;  M 1040  0              ACT$VRB_LOOP,                   ! Loop Verb processing
;  M 1041  0              ACT$VRB_UTILITY,                ! Most other Verbs
;  M 1042  0              ACT$VRB_SHOLIS,                 ! Show and List Verbs
;  M 1043  0              ACT$CLRLONG,                    ! Clear a longword
;  M 1044  0              ACT$TESTLONG,                   ! Test a longword
;  M 1045  0              ACT$COPY_VALUE,                 ! Copy a longword
;  M 1046  0              ACT$PMTDONEQ                    ! See if prompting done
;  M 1047  0              ;
;  M 1048  0
;  M 1049  0
;  M 1050  0      EXTERNAL
;  M 1051  0              PBK$G_ZAPACCDSC,                ! Parameter block to zap descriptors
;  M 1052  0
;  M 1053  0              PBK$G_VRB_ALL,                  ! Block for All parameter
;  M 1054  0              PBK$G_LOG_TYPCON,               ! Block for logging types
;  M 1055  0              PBK$G_LOG_TYPFIL,
;  M 1056  0              PBK$G_LOG_TYPMON,
;  M 1057  0
;  M 1058  0              PBK$G_EVE_ESET,                 ! Parameter blocks for events
;  M 1059  0              PBK$G_EVE_ECLS,
;  M 1060  0              PBK$G_EVE_EMSK,
;  M 1061  0              PBK$G_EVE_ERNG,
;  M 1062  0              PBK$G_EVE_EWLD,
;  M 1063  0              PBK$G_EVE_ESNO,
;  M 1064  0              PBK$G_EVE_ESLI,
;  M 1065  0              PBK$G_EVE_ESEX,
;  M 1066  0
```

```
; M 1067  0               NCP$GL_OPTION,          ! Place to build option byte
; M 1068  0               NCP$GL_FNC_CODE         ! Place to build function code
; M 1069  0
; M 1070  0               ;
```

```
;  M 1071  0
;  M 1072  0       !
;  M 1073  0       !       String descriptors for access parameters
;  M 1074  0       !
;  M 1075  0
;  M 1076  0       EXTERNAL
;  M 1077  0
;  M 1078  0               ACT$GL_ADR_Q,                   ! Flag for address
;  M 1079  0
;  M 1080  0               ACT$GL_NODAREA,                 ! Node Area
;  M 1081  0
;  M 1082  0               ACT$GQ_ACCACC_DSC,              ! Account
;  M 1083  0               ACT$GQ_ACCPSW_DSC,              ! Password
;  M 1084  0               ACT$GQ_ACCUSR_DSC,              ! User id
;  M 1085  0
;  M 1086  0               ACT$GQ_NODEID_DSC,              ! Node id descriptor
;  M 1087  0
;  M 1088  0               ACT$GL_SAD_BEGIN,               ! Subaddress beginning value
;  M 1089  0               ACT$GL_SAD_END;                 ! Subaddress ending value
;  M 1090  0
;  M 1091  0       !
;  M 1092  0       !       Status return values
;  M 1093  0       !
;  M 1094  0
;  M 1095  0       EXTERNAL LITERAL
;  M 1096  0               NCP$_INVVAL,                    ! Unrecognised value
;  M 1097  0               NCP$_INVKEY                     ! Unrecognised keyword
;  M 1098  0               ;               !
;  M 1099  0
;  M 1100  0
;     1101  0               %;
;     1102  0
```

```
;    1103  0      %SBTTL  'Macros to Build Subexpressions'
;    1104  0
;    1105  0      !
;    1106  0      !       The state tables for the NCP language have been broken into
;    1107  0      !       smaller modules to reduce compile time of the separate
;    1108  0      !       modules to reduce development time. The developement time
;    1109  0      !       has been reduced at the expense of a slight increase in the
;    1110  0      !       size of the tables since keywords and subexpression states
;    1111  0      !       are duplicated in the separate tables.
;    1112  0      !
;    1113  0      !       These macros define whole state subexpressions to parse
;    1114  0      !       useful entities. Including these subexpressions as macros in
;    1115  0      !       the library avoids having multiple copies of the source of
;    1116  0      !       the subexpressions in each of the modules of the states
;    1117  0      !       tables where they are used.
;    1118  0      !
;    1119  0      !       States and subexpressions are named in a distinctive way.
;    1120  0      !       States are named ST_xxx. Subexpressions are named SE_xxx and
;    1121  0      !       subexpression defining macros are named SEM_xxx.
;    1122  0      !
```

```
;    1123  0
;    1124  0     !
;    1125  0     !       Subexpression for a File ID
;    1126  0     !
;    1127  0
;  M 1128  0     MACRO   SEM_FILE_ID =
;  M 1129  0
;  M 1130  0     $STATE  (SE_FILE_ID,                    ! Make blanks significant
;  M 1131  0             (TPA$_EOS, TPA$_FAIL),
;  M 1132  0             (TPA$_LAMBDA,    , ACT$BLNK_SIG));
;  M 1133  0
;  M 1134  0     !
;  M 1135  0     !       Accept any string of characters for a filespec.  Format is not
;  M 1136  0     !       enforced here.
;  M 1137  0     !
;  M 1138  0
;  M 1139  0     $STATE  (SE_FILE_ID1,
;  M 1140  0             (TPA$_EOS, SE_FILE_IDX),
;  M 1141  0             (TPA$_BLANK, SE_FILE_IDX),
;  M 1142  0             ('"', SE_FILE_ID2),             ! Handle quoted portion separately
;  M 1143  0             (TPA$_ANY, SE_FILE_ID1));
;  M 1144  0
;  M 1145  0     $STATE  (SE_FILE_ID2,
;  M 1146  0             ('"', SE_FILE_ID1),             ! If ending double quote, rejoin loop
;  M 1147  0             (TPA$_EOS, SE_FILE_IDE),
;  M 1148  0             (TPA$_ANY, SE_FILE_ID2));
;  M 1149  0
;  M 1150  0     $STATE  (SE_FILE_IDX,
;  M 1151  0             (TPA$_LAMBDA,   TPA$_EXIT, ACT$BLNK_NSIG));
;  M 1152  0
;  M 1153  0     $STATE  (SE_FILE_IDE,
;  M 1154  0             (TPA$_LAMBDA,   TPA$_FAIL, ACT$BLNK_NSIG));
;  M 1155  0
;    1156  0             %;                          ! End of File-id macro
```

```
;    1157  0
;    1158  0     !
;    1159  0     !       Subexpression for Node-ID
;    1160  0     !
;    1161  0
;  M 1162  0     MACRO   SEM_NODE_ID =
;  M 1163  0
;  M 1164  0     $STATE  (SE_NODE_ID,
;  M 1165  0             ( (SE_NODE_NAM), TPA$_EXIT),
;  M 1166  0             ( (SE_NODE_ADR), TPA$_EXIT)
;  M 1167  0             );
;  M 1168  0
;  M 1169  0     $STATE  (SE_NODE_ADR,
;  M 1170  0             ( (SE_NOD_ADR), TPA$_EXIT, , TRUE, ACT$GL_ADR_Q)
;  M 1171  0             );
;  M 1172  0
;  M 1173  0     $STATE  (SE_NOD_ADR,
;  M 1174  0             (TPA$_LAMBDA, , ACT$CLRLONG, , , ACT$GL_ADR_Q)
;  M 1175  0             );
;  M 1176  0     $STATE  (,
;  M 1177  0             ((SE_NODE_AREA_Q), TPA$_EXIT),                  ! If an area precedes the adr then check its range a
;  M 1178  0             (TPA$_DECIMAL, TPA$_EXIT, ACT$NUM_RNG,
;  M 1179  0                     NUM_RANGE (LOW_NODE_ADR, HIGH_NODE_ADR))
;  M 1180  0             );
;  M 1181  0
;  M 1182  0
;  M 1183  0     $STATE  (SE_NODE_NAM,
;  M 1184  0             (TPA$_LAMBDA,    , ACT$BLNK_SIG)
;  M 1185  0             );
;  M 1186  0
;  M 1187  0     $STATE  (,
;  M 1188  0             (TPA$_LAMBDA, , ACT$CLRLONG, , , ACT$GL_ADR_Q)
;  M 1189  0             );
;  M 1190  0
;  M 1191  0     $STATE  (,
;  M 1192  0             ( (SE_NODE_NAM1), , ACT$STR_LEN, , , LEN_NODE_NAM),
;  M 1193  0             (TPA$_LAMBDA,   TPA$_FAIL, ACT$BLNK_NSIG)
;  M 1194  0             );
;  M 1195  0
;  M 1196  0     $STATE  (,
;  M 1197  0             (TPA$_LAMBDA,   TPA$_EXIT, ACT$BLNK_NSIG)
;  M 1198  0             );
;  M 1199  0
;  M 1200  0     $STATE  (SE_NODE_NAM1,
;  M 1201  0             (TPA$_DIGIT,    SE_NODE_NAM1),                  ! Check for Node names with leading digits
;  M 1202  0             (TPA$_ALPHA),                                   ! If the node name has an alpha then drop to ST_NODE
;  M 1203  0             ('$')                                           !  Otherwise it was only digits and therefore an ADR
;  M 1204  0             );
;  M 1205  0
;  M 1206  0     $STATE  (ST_NODE_NAM2,
;  M 1207  0             (TPA$_DIGIT,    ST_NODE_NAM2),
;  M 1208  0             (TPA$_ALPHA,    ST_NODE_NAM2),
;  M 1209  0             ('$'            ST_NODE_NAM2),
;  M 1210  0             (TPA$_LAMBDA,   TPA$_EXIT)
;  M 1211  0             );
;  M 1212  0
;  M 1213  0     !
```

```
;  M 1214  0       !       See if the node address has an area in front.
;  M 1215  0       !       Format is area.adr, where area and adr are decimal.
;  M 1216  0       !
;  M 1217  0       $STATE  (SE_NODE_AREA_Q,
;  M 1218  0               (TPA$_DECIMAL, , , ,ACT$GL_NODAREA)                     ! Store it in case it was an area
;  M 1219  0               );
;  M 1220  0
;  M 1221  0       $STATE  (,
;  M 1222  0               ('.', , ACT$NUM_RNG, ,                                  ! The last number parsed was indeed an area
;  M 1223  0                       NUM_RANGE (LOW_AREA, HIGH_AREA)),               !  so check the range
;  M 1224  0               (TPA$_LAMBDA, TPA$_FAIL, ACT$CLRLONG,,, ACT$GL_NODAREA) ! There was no area so clear storage
;  M 1225  0               );
;  M 1226  0
;  M 1227  0       $STATE  (,
;  M 1228  0               (TPA$_DECIMAL, TPA$_EXIT, ACT$NUM_RNG, ,                ! Check the range of the node address
;  M 1229  0                       NUM_RANGE (LOW_NODE_ADR, HIGH_NODE_ADR))
;  M 1230  0               );
;  M 1231  0
;    1232  0               %;
```

```
;    1233  0      !
;    1234  0      !       Check range of node area number
;    1235  0      !
;    1236  0
;M   1237  0      MACRO   SEM_AREA_NUM =
;M   1238  0
;M   1239  0      $STATE  (SE_AREA_NUM,
;M   1240  0              (TPA$_DECIMAL, TPA$_EXIT, ACT$NUM_RNG,',
;M   1241  0                      NUM_RANGE (LOW_AREA, HIGH_AREA)),'
;M   1242  0              );
;    1243  0              %;
```

```
;    1244  0      !
;    1245  0      !       Subexpression to accept NI address of the form nn-nn-nn-nn-nn-nn
;    1246  0      !       and since we're really nice, as a bonus we'll take nnnnnnnnnnnn.
;    1247  0      !
;    1248  0
;  M 1249  0      MACRO   SEM_NI_ADR =
;  M 1250  0
;  M 1251  0      $STATE  (SE_NI_ADR,
;  M 1252  0              ((SE_NI_ADDR), TPA$_EXIT),
;  M 1253  0              ((SE_NI_NUM), TPA$_EXIT)
;  M 1254  0              );
;  M 1255  0
;  M 1256  0      !
;  M 1257  0      !       Only accepts nn-nn-nn-nn-nn-nn
;  M 1258  0      !
;  M 1259  0      $STATE  (SE_NI_ADDR,
;  M 1260  0              (TPA$_LAMBDA, , ACT$BLNK_SIG)
;  M 1261  0              );
;  M 1262  0
;  M 1263  0      $STATE  (,
;  M 1264  0              ((SE_NUM_PAIR)));
;  M 1265  0      $STATE  (,
;  M 1266  0              ('-')
;  M 1267  0              );
;  M 1268  0
;  M 1269  0      $STATE  (,
;  M 1270  0              ((SE_NUM_PAIR)));
;  M 1271  0      $STATE  (,
;  M 1272  0              ('-')
;  M 1273  0              );
;  M 1274  0
;  M 1275  0      $STATE  (,
;  M 1276  0              ((SE_NUM_PAIR)));
;  M 1277  0      $STATE  (,
;  M 1278  0              ('-')
;  M 1279  0              );
;  M 1280  0
;  M 1281  0      $STATE  (,
;  M 1282  0              ((SE_NUM_PAIR)));
;  M 1283  0      $STATE  (,
;  M 1284  0              ('-')
;  M 1285  0              );
;  M 1286  0
;  M 1287  0      $STATE  (,
;  M 1288  0              ((SE_NUM_PAIR)));
;  M 1289  0      $STATE  (,
;  M 1290  0              ('-')
;  M 1291  0              );
;  M 1292  0
;  M 1293  0      $STATE  (,
;  M 1294  0              ((SE_NUM_PAIR), TPA$_EXIT, ACT$BLNK_NSIG),
;  M 1295  0              (TPA$_LAMBDA, TPA$_FAIL, ACT$BLNK_NSIG)
;  M 1296  0              );
;  M 1297  0
;  M 1298  0      !
;  M 1299  0      !       Accept two Hex digits
;  M 1300  0      !
```

```
;  M 1301  0      $STATE  (SE_NUM_PAIR,
;  M 1302  0              ((SE_HEX_DIGIT)),
;  M 1303  0              (TPA$_LAMBDA, SE_PAIR_FAIL));
;  M 1304  0
;  M 1305  0      $STATE  (,                          ! 2nd Hex digit
;  M 1306  0              ((SE_HEX_DIGIT), TPA$_EXIT),
;  M 1307  0              (TPA$_LAMBDA, SE_PAIR_FAIL));
;  M 1308  0
;  M 1309  0      $STATE  (SE_PAIR_FAIL,
;  M 1310  0              (TPA$_LAMBDA, TPA$_FAIL, ACT$BLNK_NSIG));
```

```
;  M 1311  0      !
;  M 1312  0      !       Accept a twelve digit hex number
;  M 1313  0      !
;  M 1314  0      $STATE  (SE_NI_NUM,
;  M 1315  0              ((SE_HEX_DIGIT)) );
;  M 1316  0      $STATE  (,
;  M 1317  0              ((SE_HEX_DIGIT)) );
;  M 1318  0      $STATE  (,
;  M 1319  0              ((SE_HEX_DIGIT)) );
;  M 1320  0      $STATE  (,
;  M 1321  0              ((SE_HEX_DIGIT)) );
;  M 1322  0      $STATE  (,
;  M 1323  0              ((SE_HEX_DIGIT)) );
;  M 1324  0      $STATE  (,
;  M 1325  0              ((SE_HEX_DIGIT)) );
;  M 1326  0      $STATE  (,
;  M 1327  0              ((SE_HEX_DIGIT)) );
;  M 1328  0      $STATE  (,
;  M 1329  0              ((SE_HEX_DIGIT)) );
;  M 1330  0      $STATE  (,
;  M 1331  0              ((SE_HEX_DIGIT)) );
;  M 1332  0      $STATE  (,
;  M 1333  0              ((SE_HEX_DIGIT)) );
;  M 1334  0      $STATE  (,
;  M 1335  0              ((SE_HEX_DIGIT)) );
;  M 1336  0      $STATE  (,                      ! 12th hex digit
;  M 1337  0              ((SE_HEX_DIGIT), TPA$_EXIT, ACT$BLNK_NSIG)
;  M 1338  0              );
;  M 1339  0
;  M 1340  0      $STATE  (SE_HEX_DIGIT,          ! Accept valid hex digit
;  M 1341  0              (TPA$_DIGIT, TPA$_EXIT),
;  M 1342  0              ('A', TPA$_EXIT),
;  M 1343  0              ('B', TPA$_EXIT),
;  M 1344  0              ('C', TPA$_EXIT),
;  M 1345  0              ('D', TPA$_EXIT),
;  M 1346  0              ('E', TPA$_EXIT),
;  M 1347  0              ('F', TPA$_EXIT));
;  M 1348  0
;    1349  0              %;
```

```
;    1350  0
;    1351  0     !
;    1352  0     !       Subexpression for Link ID
;    1353  0     !
;    1354  0     !       (This subexpression restricts the link ID to be a number within
;    1355  0     !       the range of 0-65535.  However, the NADR entity is used to store
;    1356  0     !       the link ID because the format is similiar: format byte of zero,
;    1357  0     !       followed by the word link address.  The format byte is used to
;    1358  0     !       enable requests of known links; format byte of -1).
;    1359  0     !
;    1360  0
;  M 1361  0     MACRO   SEM_LINK_ID =
;  M 1362  0
;  M 1363  0     $STATE  (SE_LINK_ID,
;  M 1364  0             (TPA$_LAMBDA,, ACT$CLRLONG,,, ACT$GL_ADR_Q));
;  M 1365  0
;  M 1366  0     $STATE  (,
;  M 1367  0             (TPA$_DECIMAL, TPA$_EXIT, ACT$NUM_RNG, TRUE, ACT$GL_ADR_Q,
;  M 1368  0                             NUM_RANGE(0, 65535)));
;  M 1369  0
;    1370  0             %;
```

```
;    1371  0
;    1372  0     !
;    1373  0     !       Hex Password for Service Operations
;    1374  0     !
;    1375  0
;  M 1376  0     MACRO   SEM_HEX_PSW =
;  M 1377  0
;  M 1378  0     $STATE  (SE_HEX_PSW,
;  M 1379  0             ( (SE_HEX_STR), TPA$_EXIT, ACT$STR_LEN, , , LEN_HEX_PSW));
;  M 1380  0
;  M 1381  0     $STATE  (SE_HEX_STR,
;  M 1382  0             (TPA$_LAMBDA, , ACT$BLNK_SIG));
;  M 1383  0
;  M 1384  0     $STATE  (,
;  M 1385  0             ((SE_HEX_CHR)),          ! Ensure at least one character given
;  M 1386  0             (TPA$_LAMBDA, TPA$_FAIL, ACT$BLNK_NSIG));
;  M 1387  0
;  M 1388  0     $STATE  (SE_HEX_STR1,
;  M 1389  0             ((SE_HEX_CHR), SE_HEX_STR1),   ! Gobble remaining hex characters
;  M 1390  0             ((SE_HEX_NONTERM), TPA$_FAIL, ACT$BLNK_NSIG),
;  M 1391  0             (TPA$_LAMBDA,   TPA$_EXIT, ACT$BLNK_NSIG));
;  M 1392  0
;  M 1393  0     $STATE  (SE_HEX_NONTERM,         ! Return false if terminator, else true
;  M 1394  0             (TPA$_BLANK,TPA$_FAIL), ! (so that blank is not gobbled by TPARSE)
;  M 1395  0             (TPA$_EOS,TPA$_FAIL),
;  M 1396  0             (TPA$_LAMBDA,TPA$_EXIT));
;  M 1397  0
;  M 1398  0     $STATE  (SE_HEX_CHR,             ! True if valid hex char (gobbled), else false
;  M 1399  0             (TPA$_DIGIT, TPA$_EXIT),
;  M 1400  0             ('A', TPA$_EXIT),
;  M 1401  0             ('B', TPA$_EXIT),
;  M 1402  0             ('C', TPA$_EXIT),
;  M 1403  0             ('D', TPA$_EXIT),
;  M 1404  0             ('E', TPA$_EXIT),
;  M 1405  0             ('F', TPA$_EXIT));
;  M 1406  0
;    1407  0             %;
```

```
;    1408  0
;    1409  0
;    1410  0        !       Hex Number
;    1411  0        !
;    1412  0
;M   1413  0        MACRO   SEM_HEX_NUM =
;M   1414  0
;M   1415  0        $STATE  (SE_HEX_NUM,
;M   1416  0                ( (SE_HEX_STR), TPA$_EXIT, ACT$STR_LEN, , , LEN_HEX_NUM));
;M   1417  0
;M   1418  0        $STATE  (SE_HEX_STR,
;M   1419  0                (TPA$_LAMBDA, , ACT$BLNK_SIG));
;M   1420  0
;M   1421  0        $STATE  (,
;M   1422  0                ((SE_HEX_CHR)),         ! Ensure at least one character given
;M   1423  0                (TPA$_LAMBDA, TPA$_FAIL, ACT$BLNK_NSIG));
;M   1424  0
;M   1425  0        $STATE  (SE_HEX_STR1,
;M   1426  0                ((SE_HEX_CHR),  SE_HEX_STR1),   ! Gobble remaining hex characters
;M   1427  0                ((SE_HEX_NONTERM), TPA$_FAIL, ACT$BLNK_NSIG),
;M   1428  0                (TPA$_LAMBDA,   TPA$_EXIT, ACT$BLNK_NSIG));
;M   1429  0
;M   1430  0        $STATE  (SE_HEX_NONTERM,        ! Return false if terminator, else true
;M   1431  0                (TPA$_BLANK,TPA$_FAIL), ! (so that blank is not gobbled by TPARSE)
;M   1432  0                (TPA$_EOS,TPA$_FAIL),
;M   1433  0                (TPA$_LAMBDA,TPA$_EXIT));
;M   1434  0
;M   1435  0        $STATE  (SE_HEX_CHR,            ! True if valid hex char (gobbled), else false
;M   1436  0                (TPA$_DIGIT, TPA$_EXIT),
;M   1437  0                ('A', TPA$_EXIT),
;M   1438  0                ('B', TPA$_EXIT),
;M   1439  0                ('C', TPA$_EXIT),
;M   1440  0                ('D', TPA$_EXIT),
;M   1441  0                ('E', TPA$_EXIT),
;M   1442  0                ('F', TPA$_EXIT));
;M   1443  0
;    1444  0                %;
```

```
;    1445  0      !
;    1446  0      !       Subexpression for a circuit name
;    1447  0      !
;    1448  0
;    1449  0      MACRO
;  M 1450  0              SEM_CIRC_ID =
;  M 1451  0
;  M 1452  0      $STATE  (SE_CIRC_ID,
;  M 1453  0              ((SE_LINE), TPA$_EXIT, ACT$STR_LEN, , , LEN_CIRC_ID)
;  M 1454  0              );
;  M 1455  0
;    1456  0              %;
;    1457  0
;    1458  0      !
;    1459  0      !       Subexpression for a DTE call number
;    1460  0      !
;    1461  0
;  M 1462  0      MACRO   SEM_DTE_NUMBER =
;  M 1463  0
;  M 1464  0      $STATE  (SE_DTE_NUMBER,
;  M 1465  0              (TPA$_STRING, TPA$_EXIT, ACT$STR_LEN,,, LEN_DTE_NUM)
;  M 1466  0              );
;  M 1467  0
;    1468  0              %;
;    1469  0
;    1470  0      !
;    1471  0      !       Subexpression for a closed user group name
;    1472  0      !
;    1473  0
;  M 1474  0      MACRO   SEM_GRP_NAME =
;  M 1475  0
;  M 1476  0      $STATE  (SE_GRP_NAME,
;  M 1477  0              (TPA$_SYMBOL, TPA$_EXIT, ACT$STR_LEN,,, LEN_GRP_NAME)
;  M 1478  0              );
;  M 1479  0
;    1480  0              %;
;    1481  0
;    1482  0      !
;    1483  0      !       Subexpression for an X.25 network name
;    1484  0      !
;    1485  0
;  M 1486  0      MACRO   SEM_NET_NAME =
;  M 1487  0
;  M 1488  0      $STATE  (SE_NET_NAME,
;  M 1489  0              (TPA$_SYMBOL, TPA$_EXIT, ACT$STR_LEN,,, LEN_NET_NAME)
;  M 1490  0              );
;  M 1491  0
;    1492  0              %;
;    1493  0
;    1494  0      !
;    1495  0      !       Subexpression for an X.25 destination name
;    1496  0      !
;    1497  0
;  M 1498  0      MACRO   SEM_DEST_NAME =
;  M 1499  0
;  M 1500  0      $STATE  (SE_DEST_NAME,
;  M 1501  0              (TPA$_SYMBOL, TPA$_EXIT, ACT$STR_LEN,,, LEN_DEST_NAME)
```

```
;  M 1502  0              );
;  M 1503  0
;    1504  0              %;
```

```
;    1505  0
;    1506  0     !
;    1507  0     !       Subexpression for a subaddress range of the form:
;    1508  0     !
;    1509  0     !       number
;    1510  0     !       number-number
;    1511  0     !
;    1512  0
;  M 1513  0     MACRO   SEM_SUBADR_RANGE =
;  M 1514  0
;  M 1515  0     $STATE  (SE_SUBADR_RANGE,
;  M 1516  0             (TPA$_DECIMAL,, ACT$NUM_RNG,, ACT$GL_SAD_BEGIN,
;  M 1517  0                     NUM_RANGE (0, 9999)));
;  M 1518  0
;  M 1519  0     $STATE  (,
;  M 1520  0             (TPA$_LAMBDA,, ACT$COPY_VALUE,,, ACT$GL_SAD_END));
;  M 1521  0
;  M 1522  0     $STATE  (,
;  M 1523  0             ('-'),
;  M 1524  0             (TPA$_LAMBDA, TPA$_EXIT));
;  M 1525  0
;  M 1526  0     $STATE  (,
;  M 1527  0             (TPA$_DECIMAL,TPA$_EXIT, ACT$NUM_RNG,, ACT$GL_SAD_END,
;  M 1528  0                     NUM_RANGE (0, 9999)));
;  M 1529  0
;    1530  0             %;
;    1531  0     !
;    1532  0     !       Subexpression for a channels list range of the form:
;    1533  0     !
;    1534  0     !       number
;    1535  0     !       number, number
;    1536  0     !       number-number
;    1537  0     !       number[-number[,..., number[-number]]]
;    1538  0     !
;    1539  0     !       NOTE: values in channels lists have limit of 4095
;    1540  0     !
;    1541  0
;  M 1542  0     MACRO   SEM_RNG_LIST =
;  M 1543  0
;  M 1544  0     $STATE  (SE_RNG_LIST,
;  M 1545  0             (TPA$_DECIMAL,, ACT$NUM_RNGSAV,,, NUM_RANGE (0, 4095))
;  M 1546  0             );
;  M 1547  0
;  M 1548  0     $STATE  (,
;  M 1549  0             (',', SE_RNG_LIST, ACT$NUM_SAV),
;  M 1550  0             ('-', SE_RNG_HYPHEN),
;  M 1551  0             (TPA$_LAMBDA, TPA$_EXIT));
;  M 1552  0
;  M 1553  0     $STATE  (SE_RNG_HYPHEN,
;  M 1554  0             (TPA$_DECIMAL,, ACT$NUM_RNGSAV,,, NUM_RANGE (0, 4095)),
;  M 1555  0             (TPA$_LAMBDA, TPA$_EXIT));
;  M 1556  0
;  M 1557  0     $STATE  (,
;  M 1558  0             (',', SE_RNG_LIST),
;  M 1559  0             (TPA$_LAMBDA, TPA$_EXIT)
;  M 1560  0             );
;  M 1561  0
```

```
;   1562  0               %;
```

```
;    1563  0      !
;    1564  0      !       Subexpression for a tracepoint name
;    1565  0      !
;    1566  0
;    1567  0      MACRO
;M   1568  0              SEM_TRCPNT_NAME =
;M   1569  0
;M   1570  0      $STATE  (SE_TRCPNT_NAME,
;M   1571  0              ((SE_FILE_ID), TPA$_EXIT, ACT$STR_LEN, , , LEN_TRCPNT_NAME)
;M   1572  0              );
;M   1573  0
;    1574  0              %;
```

```
;    1575  0      !
;    1576  0      !       Subexpression for a line ID
;    1577  0      !
;    1578  0      !       Allow any string terminated with a blank
;    1579  0      !
;    1580  0
; M  1581  0      MACRO   SEM_LINE_ID =
; M  1582  0
; M  1583  0      $STATE  (SE_LINE_ID,
; M  1584  0              ( (SE_LINE),    TPA$_EXIT, ACT$STR_LEN, , , LEN_LINE_ID)
; M  1585  0              );
; M  1586  0
; M  1587  0      $STATE  (SE_LINE,
; M  1588  0              (TPA$_LAMBDA, , ACT$BLNK_SIG)
; M  1589  0              );
; M  1590  0
; M  1591  0      $STATE  (,
; M  1592  0              (TPA$_ALPHA),
; M  1593  0              (TPA$_DIGIT),
; M  1594  0              ('-'),
; M  1595  0              ('.'),
; M  1596  0              ('*'),
; M  1597  0              ('$')
; M  1598  0              );
; M  1599  0
; M  1600  0      $STATE  (SE_LINECHAR,
; M  1601  0              (TPA$_ALPHA, SE_LINECHAR),
; M  1602  0              (TPA$_DIGIT, SE_LINECHAR),
; M  1603  0              ('-', SE_LINECHAR),
; M  1604  0              ('.', SE_LINECHAR),
; M  1605  0              ('*', SE_LINECHAR),
; M  1606  0              ('$', SE_LINECHAR),
; M  1607  0              (TPA$_LAMBDA, TPA$_EXIT, ACT$BLNK_NSIG)
; M  1608  0              );
; M  1609  0
;    1610  0              %;
;    1611  0
```

```
;    1612  0      !
;    1613  0      !       Subexpression for the ALL parameter
;    1614  0      !
;    1615  0
;    1616  0      MACRO
;M   1617  0              SEM_ALL =
;M   1618  0
;M   1619  0      $STATE  (SE_ALL,
;M   1620  0              ('ALL')         ! If the word is here it must be last on the line
;M   1621  0              );
;M   1622  0
;M   1623  0      $STATE  (,
;M   1624  0              (TPA$_EOS, TPA$_EXIT, ACT$SAVPRM, , , PBK$G_VRB_ALL)
;M   1625  0              );
;M   1626  0
;    1627  0              %;
```

```
;    1628  0
;    1629  0     !
;    1630  0     !       Subexpression for Access Control Information
;    1631  0     !
;    1632  0
;  M 1633  0     MACRO   SEM_ACCESS =
;  M 1634  0
;  M 1635  0
;  M 1636  0     $STATE  (SE_ACC_ACC,
;  M 1637  0             ( (SE_QOOT_STR), TPA$_EXIT, ACT$STR_LEN, , , LEN_ACC_ACC)
;  M 1638  0             );
;  M 1639  0
;  M 1640  0     $STATE  (SE_ACC_PSW,
;  M 1641  0             ( (SE_QOOT_STR), TPA$_EXIT, ACT$STR_LEN, , , LEN_ACC_PSW)
;  M 1642  0             );
;  M 1643  0
;  M 1644  0     $STATE  (SE_ACC_USR,
;  M 1645  0             ( (SE_QOOT_STR), TPA$_EXIT, ACT$STR_LEN, , , LEN_ACC_USR)
;  M 1646  0             );
;  M 1647  0
;    1648  0             %;
```

```
;    1649  0
;    1650  0     !
;    1651  0     !       Subexpression for a quoted string
;    1652  0     !
;    1653  0
;M   1654  0     MACRO   SEM_QUOT_STR =
;M   1655  0
;M   1656  0     $STATE  (SE_QUOT_STR,
;M   1657  0             (TPA$_EOS,      TPA$_FAIL),                     ! Got to be something
;M   1658  0             (TPA$_BLANK,    ,       ACT$BLNK_SIG),          ! Make blanks significant
;M   1659  0             (TPA$_LAMBDA,   ,       ACT$BLNK_SIG)
;M   1660  0             );
;M   1661  0
;M   1662  0     $STATE  (,                                              ! Quoted string or just string
;M   1663  0             ('"',           ST_QUOT_STR3),
;M   1664  0             (TPA$_LAMBDA)
;M   1665  0             );
;M   1666  0
;M   1667  0     $STATE  (ST_QUOT_STR2,                                  ! Just a string
;M   1668  0             (TPA$_SYMBOL,   ST_QUOT_STR2),
;M   1669  0             (TPA$_BLANK,    ST_QUOT_STRX),
;M   1670  0             (TPA$_ANY,      ST_QUOT_STR2),
;M   1671  0             (TPA$_EOS,      ST_QUOT_STRX)
;M   1672  0             );
;M   1673  0
;M   1674  0     $STATE  (ST_QUOT_STR3,                                  ! A quoted string to be sure
;M   1675  0             ( (SE_QUOT_DBL), ST_QUOT_STR3),
;M   1676  0             ('"',           ST_QUOT_STRX),
;M   1677  0             (TPA$_ANY,      ST_QUOT_STR3),
;M   1678  0             (TPA$_EOS,      ST_QUOT_STRE)
;M   1679  0             );
;M   1680  0
;M   1681  0     $STATE  (ST_QUOT_STRX,
;M   1682  0             (TPA$_LAMBDA,   TPA$_EXIT, ACT$BLNK_NSIG)
;M   1683  0             );
;M   1684  0
;M   1685  0     $STATE  (ST_QUOT_STRE,
;M   1686  0             (TPA$_LAMBDA,   TPA$_FAIL, ACT$BLNK_NSIG)
;M   1687  0             );
;M   1688  0
;M   1689  0     $STATE  (SE_QUOT_DBL,                                   ! Do we have a double quote
;M   1690  0             ('"')
;M   1691  0             );
;M   1692  0
;M   1693  0     $STATE  (,
;M   1694  0             ('"',   TPA$_EXIT)
;M   1695  0             );
;M   1696  0
;    1697  0             %;
```

```
;    1698  0
;    1699  0     !
;    1700  0     !       Event list subexpression
;    1701  0     !
;    1702  0
;    1703  0     MACRO
;  M 1704  0             SEM_EVENT_LIST =
;  M 1705  0
;  M 1706  0     $STATE  (SE_EVENT_LIST,
;  M 1707  0             (TPA$_LAMBDA, , ACT$BLNK_SIG)
;  M 1708  0             );
;  M 1709  0
;  M 1710  0     $STATE  (,
;  M 1711  0             ( (SE_EVENT), TPA$_EXIT, ACT$BLNK_NSIG),
;  M 1712  0             (TPA$_LAMBDA, TPA$_FAIL, ACT$BLNK_NSIG)
;  M 1713  0             );
;  M 1714  0
;  M 1715  0
;  M 1716  0     !
;  M 1717  0     !       Parse a single event
;  M 1718  0     !
;  M 1719  0
;  M 1720  0     $STATE  (SE_EVENT,
;  M 1721  0             (TPA$_DECIMAL, ,ACT$NUM_RNG, , ,
;  M 1722  0                            NUM_RANGE (LOW_EVENT_CLS, HIGH_EVENT_CLS) ),
;  M 1723  0             );
;  M 1724  0
;  M 1725  0     $STATE  (,
;  M 1726  0             (TPA$_LAMBDA, , ACT$SAVPRM, , , PBK$G_EVE_ECLS)
;  M 1727  0             );
;  M 1728  0
;  M 1729  0     $STATE  (,
;  M 1730  0             ('.')
;  M 1731  0             );
;  M 1732  0
;  M 1733  0     $STATE  (ST_EVENT_1,
;  M 1734  0             ( (SE_EVENT_TYP), , ACT$SAVPRM , , PBK$G_EVE_EMSK),
;  M 1735  0             ('*', TPA$_EXIT, ACT$SAVPRM, 2^(14+8), PDB$G_VRB_EVE, PBK$G_EVE_EWLD)
;  M 1736  0             );
;  M 1737  0
;  M 1738  0     $STATE  (,
;  M 1739  0             ('.', ST_EVENT_1),
;  M 1740  0             ('-', ST_EVENT_2),
;  M 1741  0             (TPA$_BLANK, TPA$_EXIT),
;  M 1742  0             (TPA$_EOS, TPA$_EXIT)
;  M 1743  0             );
```

```
;  M 1744  0
;  M 1745  0       $STATE  (ST_EVENT_2,
;  M 1746  0               ( (SE_EVENT_TYP), , ACT$SAVPRM, , , PBK$G_EVE_ERNG)
;  M 1747  0               );
;  M 1748  0
;  M 1749  0       $STATE  (
;  M 1750  0               (',', ST_EVENT_1),
;  M 1751  0               (TPA$_BLANK, TPA$_EXIT),
;  M 1752  0               (TPA$_EOS, TPA$_EXIT)
;  M 1753  0               );
;  M 1754  0
;  M 1755  0       !
;  M 1756  0       !       Known events
;  M 1757  0       !
;  M 1758  0
;  M 1759  0       $STATE  (SE_EVENT_KNOWN,
;  M 1760  0               (TPA$_LAMBDA, TPA$_EXIT, ACT$SAVPRM, 3^(14+8), PDB$G_VRB_EVE,
;  M 1761  0                                       PBK$G_EVE_EWLD)
;  M 1762  0               );
;  M 1763  0
;  M 1764  0
;  M 1765  0       !
;  M 1766  0       !       Parse the type for an event
;  M 1767  0       !
;  M 1768  0
;  M 1769  0       $STATE  (SE_EVENT_TYP,
;  M 1770  0               (TPA$_DECIMAL, TPA$_EXIT, ACT$NUM_RNG, , ,
;  M 1771  0                               NUM_RANGE (LOW_EVENT_TYP, HIGH_EVENT_TYP) )
;  M 1772  0               );
;  M 1773  0
;    1774  0               %;
```

```
;    1775  0
;    1776  0
;    1777  0     !
;    1778  0     !       Logging type
;    1779  0     !
;    1780  0
;    1781  0     MACRO
;  M 1782  0             SEM_LOG_TYP =
;  M 1783  0
;  M 1784  0     $STATE  (SE_LOG_TYP,
;  M 1785  0
;  M 1786  0             KEYWORD_STATE
;  M 1787  0             (LOG,
;  M 1788  0
;  M 1789  0             TYPCON, 'CONSOLE',
;  M 1790  0             TYPFIL, 'FILE',
;  M 1791  0             TYPMON, 'MONITOR',
;  M 1792  0
;  M 1793  0             )
;  M 1794  0             );
;  M 1795  0
;    1796  0             %;
;    1797  0
;    1798  0     !
;    1799  0     !       Subexpression for Object ID
;    1800  0     !
;    1801  0
;    1802  0     MACRO
;  M 1803  0             SEM_OBJECT_ID =
;  M 1804  0
;  M 1805  0     $STATE  (SE_OBJECT_ID,
;  M 1806  0             ((SE_OBJECT_NAM), TPA$_EXIT),
;  M 1807  0             ((SE_OBJECT_NUM), TPA$_EXIT)
;  M 1808  0             );
;  M 1809  0
;  M 1810  0     $STATE  (SE_OBJECT_NUM,
;  M 1811  0             ((SE_OBJ_NUM), TPA$_EXIT, , TRUE, ACT$GL_ADR_Q)
;  M 1812  0             );
;  M 1813  0
;  M 1814  0     $STATE  (SE_OBJ_NUM,
;  M 1815  0             (TPA$_LAMBDA, , ACT$CLRLONG, , , ACT$GL_ADR_Q)
;  M 1816  0             );
;  M 1817  0     $STATE  (,
;  M 1818  0             (TPA$_DECIMAL, TPA$_EXIT, ACT$NUM_RNG, , ,
;  M 1819  0                     NUM_RANGE (LOW_OBJ_NUM, HIGH_OBJ_NUM))
;  M 1820  0             );
;  M 1821  0
;  M 1822  0     $STATE  (SE_OBJECT_NAM,
;  M 1823  0             (TPA$_LAMBDA, , ACT$CLRLONG, , , ACT$GL_ADR_Q)
;  M 1824  0             );
;  M 1825  0
;  M 1826  0     $STATE  (,
;  M 1827  0             (TPA$_SYMBOL, TPA$_EXIT, ACT$STR_LEN, , , LEN_OBJ_NAM)
;  M 1828  0             );
;  M 1829  0
;    1830  0             %;
;    1831  0
```

```
;    1832  0      !
;    1833  0      !       Subexpressions for a query state
;    1834  0      !
;    1835  0
;    1836  0      MACRO
;  M 1837  0              SEM_QUERY =
;  M 1838  0
;  M 1839  0      $STATE  (SE_QRY_YES,
;  M 1840  0              ('YES'),
;  M 1841  0              );
;  M 1842  0
;  M 1843  0      $STATE  (,
;  M 1844  0              (TPA$_EOS, TPA$_EXIT)
;  M 1845  0              );
;  M 1846  0
;  M 1847  0      $STATE  (SE_QRY_NO,
;  M 1848  0              ('NO'),
;  M 1849  0              );
;  M 1850  0
;  M 1851  0      $STATE  (,
;  M 1852  0              (TPA$_EOS, TPA$_EXIT)
;  M 1853  0              );
;  M 1854  0
;    1855  0              %;
```

```
;    1856  0
;    1857  0       !
;    1858  0       !       Subexpressions for load parameters
;    1859  0       !
;    1860  0
;    1861  0       MACRO
;M   1862  0               SEM_LOAD (CLS) =
;M   1863  0
;M   1864  0       !
;M   1865  0       !       Subexpression for service device
;M   1866  0       !
;M   1867  0
;M   1868  0       $STATE  (%NAME ('ST_',CLS,'_SDV'),
;M   1869  0
;M   1870  0               KEYWORD_STATE
;M   1871  0               (CLS,
;M   1872  0
;M   1873  0               SDVA,  'DA',
;M   1874  0               SDVL,  'DL',
;M   1875  0               SDVMC, 'DMC',
;M   1876  0               SDVP,  'DP',
;M   1877  0               SDVQ,  'DQ',
;M   1878  0               SDVTE, 'DTE',
;M   1879  0               SDVU,  'DU',
;M   1880  0               SDVUP, 'DUP',
;M   1881  0               SDVKL, 'KL8',
;M   1882  0               SDVMP, 'DMP',
;M   1883  0               SDVMV, 'DMV',
;M   1884  0               SDVPV, 'DPV',
;M   1885  0               SDVMF, 'DMF',
;M   1886  0               SDVUN, 'UNA',
;M   1887  0
;M   1888  0               )
;M   1889  0               );
;M   1890  0
;M   1891  0       !
;M   1892  0       !       Software identification
;M   1893  0       !
;M   1894  0
;M   1895  0       $STATE  (SE_SOFT_ID,
;M   1896  0               ( (SE_QUOT_STR), TPA$_EXIT, ACT$STR_LEN, , , LEN_SOFT_ID)
;M   1897  0               );
;M   1898  0
;M   1899  0       !
;M   1900  0       !       Software type
;M   1901  0       !
;M   1902  0
;M   1903  0       $STATE  (%NAME ('ST_',CLS,'_STY'),
;M   1904  0
;M   1905  0               DISPATCH_STATES
;M   1906  0               (CLS,
;M   1907  0
;M   1908  0               STSL, 'SECONDARY',
;M   1909  0               STTL, 'TERTIARY',
;M   1910  0               STOS, 'SYSTEM',
;M   1911  0
;M   1912  0               )
```

```
;  M 1913  0               );
;  M 1914  0
;  M 1915  0
;  M 1916  0       $STATE  (%NAME ('ST_', CLS, '_PRC_STSL'),               ! Secondary loader
;  M 1917  0               ( 'LOADER' ),
;  M 1918  0               (TPA$_LAMBDA)
;  M 1919  0               );
;  M 1920  0
;  M 1921  0       $STATE  (,
;  M 1922  0               ( (%NAME ('ST_',CLS,'_STSL') ), TPA$_EXIT)
;  M 1923  0               );
;  M 1924  0
;  M 1925  0       $STATE  (%NAME ('ST_', CLS, '_PRC_STTL'),               ! Tertiary loader
;  M 1926  0               ( 'LOADER' ),
;  M 1927  0               (TPA$_LAMBDA)
;  M 1928  0               );
;  M 1929  0
;  M 1930  0       $STATE  (,
;  M 1931  0               ( (%NAME ('ST_',CLS,'_STTL') ), TPA$_EXIT)
;  M 1932  0               );
;  M 1933  0
;  M 1934  0       $STATE  (%NAME ('ST_', CLS, '_PRC_STOS'),               ! System
;  M 1935  0               ( (%NAME ('ST_',CLS,'_STOS') ), TPA$_EXIT)
;  M 1936  0               );
;  M 1937  0
;  M 1938  0
;  M 1939  0               SUB_EXPRESSIONS
;  M 1940  0               (CLS,
;  M 1941  0
;  M 1942  0               STSL, TPA$_LAMBDA,
;  M 1943  0               STTL, TPA$_LAMBDA,
;  M 1944  0               STOS, TPA$_LAMBDA
;  M 1945  0
;  M 1946  0               )
;  M 1947  0
;  M 1948  0
;  M 1949  0       !
;  M 1950  0       !       Cpu type
;  M 1951  0       !
;  M 1952  0
;  M 1953  0       $STATE  (%NAME ('ST_',CLS,'_CPU'),
;  M 1954  0
;  M 1955  0               KEYWORD_STATE
;  M 1956  0               (CLS,
;  M 1957  0
;  M 1958  0               CPU10, 'DECSYSTEM1020',
;  M 1959  0               CPU11, 'PDP11',
;  M 1960  0               CPU8,  'PDP8',
;  M 1961  0               VAX,   'VAX',
;  M 1962  0
;  M 1963  0               )
;  M 1964  0               );
;  M 1965  0
;    1966  0               %;
```

```
;   1967  0      !END
;   1968  0      !ELUDOM
```

```
;  1969  0     !
;  1970  0     ! Version:      'V04-000'
;  1971  0     !
;  1972  0     !****************************************************************************
;  1973  0     !*                                                                          *
;  1974  0     !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;  1975  0     !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
;  1976  0     !*  ALL RIGHTS RESERVED.                                                    *
;  1977  0     !*                                                                          *
;  1978  0     !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;  1979  0     !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;  1980  0     !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;  1981  0     !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;  1982  0     !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;  1983  0     !*  TRANSFERRED.                                                            *
;  1984  0     !*                                                                          *
;  1985  0     !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;  1986  0     !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;  1987  0     !*  CORPORATION.                                                            *
;  1988  0     !*                                                                          *
;  1989  0     !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;  1990  0     !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
;  1991  0     !*                                                                          *
;  1992  0     !*                                                                          *
;  1993  0     !****************************************************************************
;  1994  0
;  1995  0     !++
;  1996  0     !
;  1997  0     ! NMAHEAD.B32
;  1998  0     !
;  1999  0     !       Define $EQULST macro to make library from the NMALIBRY.B32 file
;  2000  0     !
;  2001  0     !       This source is taken from the following source:
;  2002  0     !--
;  2003  0     !++
;  2004  0     !
;  2005  0     ! UTLDEF.B32 - UTILITY DEFINITION MACROS FOR BLISS PROCESSING
;  2006  0     ! OF STARLET DEFINITION MACROS.
;  2007  0     !
;  2008  0     !--
;  2009  0
;  2010  0
;  2011  0     !
;  2012  0     ! MACRO TO GENERATE EQULST CONSTRUCTS.
;  2013  0     !
;  2014  0     MACRO
;M 2015  0             $EQULST(P,G,I,S)[A]=
;M 2016  0                 %NAME(P,GET1ST_ A) =
;M 2017  0                     %IF NUL2ND_ A
;M 2018  0                     %THEN (I) + %COUNT*(S)  ! ASSUMES I, S ALWAYS GENERATED BY CONVERSION PROGRAM
;M 2019  0                     %ELSE GET2ND_ A
;  2020  0                     %FI %,
;  2021  0
;M 2022  0             GET1ST_(A,B)=
;  2023  0                 A %,
;M 2024  0             GET2ND_(A,B)=
;  2025  0                 B %,        ! KNOWN NON-NULL
```

```
;  M 2026  0               NUL2ND (A,B)=
;    2027  0                   %NULL(B) %;
;    2028  0
;    2029  0       !
;    2030  0       !           End of NMAHEAD
;    2031  0       !
```

```
;    2032  0     ! ****************************************************************************************************
;    2033  0     !  Created 15-SEP-1984 23:05:41  by  VAX-11 SDL V2.0      Source: 15-SEP-1984 22:47:31  _$255$DUA28:[NCP.SRC]NCPDEF.
;    2034  0     ! ****************************************************************************************************
;    2035  0
;    2036  0
;    2037  0     !*** MODULE $PBKDEF ***
;    2038  0     literal PBK$K_SIZE = 9;                 !  Size of the structure
;    2039  0     literal PBK$C_SIZE = 9;                 !  Size of the structure
;    2040  0     !  Parameter type values
;    2041  0     literal PBK$K_LOW = 1;                  !  Lowest value here
;    2042  0     literal PBK$K_LITB = 1;                 !  Literal byte
;    2043  0     literal PBK$K_NUMB = 2;                 !  Numeric byte
;    2044  0     literal PBK$K_NUMW = 3;                 !  Numeric word
;    2045  0     literal PBK$K_NUML = 4;                 !  Numeric longword
;    2046  0     literal PBK$K_TKN = 5;                  !  Token string
;    2047  0     literal PBK$K_TKNQ = 6;                 !  Quoted token
;    2048  0     literal PBK$K_NADR = 7;                 !  Node address
;    2049  0     literal PBK$K_HXPS = 8;                 !  Hex password
;    2050  0     literal PBK$K_STRQ = 9;                 !  Quoted string
;    2051  0     literal PBK$K_TRIPL = 10;               !  Version triple
;    2052  0     literal PBK$K_LITL = 11;                !  Long word literal
;    2053  0     literal PBK$K_PRVL = 12;                !  Privilege list
;    2054  0     literal PBK$K_PRVC = 13;                !  Privilege list clear
;    2055  0     literal PBK$K_ESET = 14;                !  Setup event parameter
;    2056  0     literal PBK$K_ECLS = 15;                !  Store event class
;    2057  0     literal PBK$K_EMSK = 16;                !  Store single event
;    2058  0     literal PBK$K_ERNG = 17;                !  Store event type range
;    2059  0     literal PBK$K_EWLD = 18;
;    2060  0     literal PBK$K_ESNO = 19;                !  Store source node
;    2061  0     literal PBK$K_ESLI = 20;                !  Store source line
;    2062  0     !           MODPRM,   added at bottom       /* Store module name
;    2063  0     literal PBK$K_ESEX = 21;                !  Source as executor node
;    2064  0     literal PBK$K_ENT = 22;                 !  Entity type and ID
;    2065  0     literal PBK$K_END = 23;                 !  End of PCL list
;    2066  0     literal PBK$K_SAD = 24;                 !  Subaddress range
;    2067  0     literal PBK$K_OBJ = 25;                 !  Object ID
;    2068  0     literal PBK$K_ESCI = 26;                !  Store source circuit
;    2069  0     literal PBK$K_RNGL = 27;                !  Range lists
;    2070  0     literal PBK$K_HEX = 28;                 !  Hexidecimal numbers
;    2071  0     literal PBK$K_AREA = 29;                !  byte of zero and byte of Node Area
;    2072  0     literal PBK$K_AADR = 30;                !  Node Area and Address
;    2073  0     !    NOTE: Used instead of NUMW to avoid hassle of handling area by action routine
;    2074  0     literal PBK$K_NIADR = 31;               !  NI address, HEX image printed backwardss
;    2075  0     literal PBK$K_DELTIM = 32;              !  Delta time, (Hours, Minutes, Seconds)
;    2076  0     literal PBK$K_DAYTIM = 33;              !  Day and time (Day, Month, Hour, Minutes, Seconds)
;    2077  0     literal PBK$K_LITLST = 34;              !  Variable length list of coded data
;    2078  0     literal PBK$K_MODPRM = 35;              !  Store module name
;    2079  0     literal PBK$K_HIGH = 35;                !  Highest value here
;    2080  0     literal PBK$S_PBKDEF = 9;
;    2081  0     macro PBK$B_TYPECODE = 0,0,8,0 %;       !  Type of parameter to store
;    2082  0     macro PBK$L_PDB_ADR = 1,0,32,0 %;       !  Address of parameter data block
;    2083  0     macro PBK$L_PARAM = 5,0,32,0 %;         !  Parameter for savparam routine
;    2084  0
;    2085  0     !*** MODULE $PDBDEF ***
;    2086  0     literal PDB$K_SIZE = 2;                 !  Size of the structure
;    2087  0     literal PDB$C_SIZE = 2;                 !  Size of the structure
;    2088  0     literal PDB$S_PDBDEF = 2;
```

```
;    2089  0     macro PDB$B_STS_FLG = 0,0,8,0 %;          !  Status flag
;    2090  0     macro PDB$T_DATA = 1,0,8,0 %;             !  Data is here
;    2091  0
;    2092  0     !*** MODULE $SDBDEF ***
;    2093  0     literal SDB$K_SIZE = 9;
;    2094  0     literal SDB$C_SIZE = 9;
;    2095  0     literal SDB$S_SDBDEF = 9;
;    2096  0     macro SDB$B_ENT_TYP = 0,0,8,1 %;          !  Entity type. If negative,
;    2097  0     !       then system-specific entity type.
;    2098  0     macro SDB$L_ENT_ADR = 1,0,32,0 %;         !  Entity parameter address
;    2099  0     macro SDB$L_PCL_ADR = 5,0,32,0 %;         !  Parameter control list address
;    2100  0
;    2101  0     !*** MODULE $PCLDEF ***
;    2102  0     literal PCL$K_SIZE = 7;                   !  Size of the structure
;    2103  0     literal PCL$C_SIZE = 7;                   !  Size of the structure
;    2104  0     literal PCL$S_PCLDEF = 7;
;    2105  0     macro PCL$B_PRM_TYP = 0,0,8,0 %;          !  Type of parameter
;    2106  0     macro PCL$W_PRM_ID = 1,0,16,0 %;          !  Code value for parameter
;    2107  0     macro PCL$L_PDB_ADR = 3,0,32,0 %;         !  Address of PDB for parameter
;    2108  0
;    2109  0     !*** MODULE $LCBDEF ***
;    2110  0     literal LCB$C_NCBSIZE = 100;              !  Size of NCB
;    2111  0     literal LCB$K_SIZE = 118;                 !  Size of structure
;    2112  0     literal LCB$C_SIZE = 118;                 !  Size of structure
;    2113  0     literal LCB$S_LCBDEF = 118;
;    2114  0     macro LCB$B_STS = 0,0,8,0 %;              !  Status, true for link open
;    2115  0     macro LCB$B_PH2 = 1,0,8,0 %;              !  Phase II, true for phase II NML
;    2116  0     macro LCB$W_CHAN = 2,0,16,0 %;            !  Link channel number
;    2117  0     macro LCB$W_MBXCHN = 4,0,16,0 %;          !  Mailbox channel number
;    2118  0     macro LCB$B_NMLVERS = 6,0,24,0 %;
;    2119  0     literal LCB$S_NMLVERS = 3;                !  NML version number (3 bytes)
;    2120  0     macro LCB$L_NCBCNT = 10,0,32,0 %;         !  Descriptor for NCB
;    2121  0     macro LCB$L_NCBPTR = 14,0,32,0 %;
;    2122  0     macro LCB$T_NCB = 18,0,0,0 %;
;    2123  0     literal LCB$S_NCB = 100;                  !  Network Control block
;    2124  0
;    2125  0     !*** MODULE $NCPDEF ***
;    2126  0     !
;    2127  0     !    Index the MODULE entities
;    2128  0     !
;    2129  0     literal NCP$C_ENT_MODCNF = 1;             !  Module Configurator
;    2130  0     literal NCP$C_ENT_MODCNS = 2;             !  Module Console
;    2131  0     literal NCP$C_ENT_MODLOA = 3;             !  Module Loader
;    2132  0     literal NCP$C_ENT_MODLOO = 4;             !  Module Looper
;    2133  0     literal NCP$C_ENT_MODACC = 5;             !  Module X25-Access
;    2134  0     literal NCP$C_ENT_MODPRO = 6;             !  Module X25-Protocol
;    2135  0     literal NCP$C_ENT_MODSER = 7;             !  Module X25-Server
;    2136  0     literal NCP$C_ENT_MODTRC = 8;             !  Module X25-Trace
;    2137  0     literal NCP$C_ENT_MOD29S = 9;             !  Module X29-Server
```

```
;    2138  0     !
;    2139  0     ! Version:      'V04-000'
;    2140  0     !
;    2141  0     !++
;    2142  0     !       NMATAIL.B32
;    2143  0     !
;    2144  0     !       Source to undeclare the macros required for the precompile of
;    2145  0     !       NMALIBRY.B32 so they do not appear in the library.
;    2146  0     !--
;    2147  0
;    2148  0
;    2149  0     UNDECLARE %QUOTE $EQULST,
;    2150  0               %QUOTE GET1ST_,
;    2151  0               %QUOTE GET2ND_,
;    2152  0               %QUOTE NUL2ND_
;    2153  0               ;
;    2154  0
;    2155  0     !
;    2156  0     !       End of NMATAIL.B32
;    2157  0     !



;                                   COMMAND QUALIFIERS
;
;       BLISS/LIB=LIB$:NCPLIBRY/LIS=LIS$:NCPLIBRY SRC$:NCPLIBRY+NMAHEAD+LIB$:NCPDEF+SRC$:NMATAIL

; Run Time:          00:16.4
; Elapsed Time:      00:26.7
; Lines/CPU Min:      7905
; Lexemes/CPU-Min: 39144
; Memory Used:  124 pages
; Library Precompilation Complete
```

NCPCONCAR
LIS

NCPERRMSG
LIS

NCPCONMAN
LIS

NCPLIBRY
B32

NCPMAIN
LIS

NCPNETIO
LIS

NMAHEAD
B32

NCPLIBRY
LIS

NMATAIL
B32