J 6

```
NNN         NNN  MMM           MMM  LLL
NNN         NNN  MMM           MMM  LLL
NNN         NNN  MMM           MMM  LLL
NNN         NNN  MMMMMM   MMMMMM  LLL
NNN         NNN  MMMMMM   MMMMMM  LLL
NNN         NNN  MMMMMM   MMMMMM  LLL
NNNNNN      NNN  MMM   MMM   MMM  LLL
NNNNNN      NNN  MMM   MMM   MMM  LLL
NNNNNN      NNN  MMM   MMM   MMM  LLL
NNN   NNN   NNN  MMM           MMM  LLL
NNN   NNN   NNN  MMM           MMM  LLL
NNN   NNN   NNN  MMM           MMM  LLL
NNN      NNNNNN  MMM           MMM  LLL
NNN      NNNNNN  MMM           MMM  LLL
NNN      NNNNNN  MMM           MMM  LLL
NNN         NNN  MMM           MMM  LLL
NNN         NNN  MMM           MMM  LLL
NNN         NNN  MMM           MMM  LLL
NNN         NNN  MMM           MMM  LLLLLLLLLLLLLLL
NNN         NNN  MMM           MMM  LLLLLLLLLLLLLLL
NNN         NNN  MMM           MMM  LLLLLLLLLLLLLLL
```

FILEID**NMLPARPRM


```
NN      NN  MM      MM  LL          PPPPPPPP      AAAAAA    RRRRRRRR    PPPPPPPP    RRRRRRRR    MM      MM
NN      NN  MM      MM  LL          PPPPPPPP      AAAAAA    RRRRRRRR    PPPPPPPP    RRRRRRRR    MM      MM
NN      NN  MMMM  MMMM  LL          PP      PP  AA      AA  RR      RR  PP      PP  RR      RR  MMMM  MMMM
NN      NN  MMMM  MMMM  LL          PP      PP  AA      AA  RR      RR  PP      PP  RR      RR  MMMM  MMMM
NNNN    NN  MM  MM  MM  LL          PP      PP  AA      AA  RR      RR  PP      PP  RR      RR  MM  MM  MM
NNNN    NN  MM  MM  MM  LL          PP      PP  AA      AA  RR      RR  PP      PP  RR      RR  MM  MM  MM
NN  NN  NN  MM      MM  LL          PPPPPPPP    AA      AA  RRRRRRRR    PPPPPPPP    RRRRRRRR    MM      MM
NN  NN  NN  MM      MM  LL          PPPPPPPP    AA      AA  RRRRRRRR    PPPPPPPP    RRRRRRRR    MM      MM
NN    NNNN  MM      MM  LL          PP          AAAAAAAAAA  RR  RR      PP          RR  RR      MM      MM
NN    NNNN  MM      MM  LL          PP          AAAAAAAAAA  RR  RR      PP          RR  RR      MM      MM
NN      NN  MM      MM  LL          PP          AA      AA  RR    RR    PP          RR    RR    MM      MM  ....
NN      NN  MM      MM  LL          PP          AA      AA  RR    RR    PP          RR    RR    MM      MM  ....
NN      NN  MM      MM  LLLLLLLLLL  PP          AA      AA  RR      RR  PP          RR      RR  MM      MM  ....
NN      NN  MM      MM  LLLLLLLLLL  PP          AA      AA  RR      RR  PP          RR      RR  MM      MM  ....

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
```

```
0 %TITLE 'NML NPARSE action routines for parsing parameters'
0 MODULE NML$PARPRM (
0                 LANGUAGE (BLISS32),
0                 ADDRESSING_MODE (NONEXTERNAL=GENERAL),
0                 ADDRESSING_MODE (EXTERNAL=GENERAL),
0                 IDENT = 'V04-000'
0                 ) =
1 BEGIN
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1
1 !++
1 ! FACILITY:  DECnet-VAX V2.0 Network Management Listener
1 !
1 !
1 ! ABSTRACT:
1 !
1 !     This module contains action routines called by NPARSE to parse and
1 !     store NICE entity parameters.
1 !
1 ! ENVIRONMENT:  VAX/VMS Operating System
1 !
1 ! AUTHOR:  Distributed Systems Software Engineering
1 !
1 ! CREATION DATE:  8-Nov-1979
1 !
1 ! MODIFIED BY:
1 !
1 !         V03-007 MKP0007         Kathy Perko             25-Mar-1984
1 !                 Add support to convert area 0 to 1 for Phase IV NCPs
1 !                 and to the executor's area for Phase III NCPs.
1 !
1 !         V03-006 MKP0006         Kathy Perko             9-Jan-1984
1 !                 Change name of NML$PRM_CHKHOST to the more general
1 !                 name, NML$PRM_CHKNODE.  Modify NML$PRM_CIRC_OWNER to
```

```
!               check the node to see if the executor.
!
!       V03-005 MKP0005         Kathy Perko             4-Aug-1983
!               Add support to make node permanent database faster.
!
!       V03-004 MKP0004         Kathy Perko             20-April-1983
!               Remove service functions from NML.
!
!       V03-003 MKP0003         Kathy Perko             31-Aug-1982
!               Fix X25-PROTOCOL GROUP checks to make sure right
!               combination of groups and dtes are in command.
!
!       V03-002 MKP0002         Kathy Perko             21-June-1982
!               Add routine to use NET string as entity ID if NICE
!               command is SET X-PROTOCOL NET <net-id>.  Otherwise,
!               NML uses a null string to indicate the active network.
!               Also, add a routine to make sure KNOWN qualifiers are
!               used only with ALL (for X25-PROTOCOL GROUPS).
!
!       V03-001 MKP0001         Kathy Perko             16-June-1982
!               Add routine to parse the circuit parameter, owner.
!
!       V02-004 MKP0003         Kathy Perko             23-Feb-1982
!               X25-PROTOCOL GROUP commands with DTE IDs included.  This
!               routine concatenates a root entity with a sub entity
!               to form the entity id.  Used when the ACPs data base keeps
!               the entities in a hierarchical form.  Also, delete the routine,
!               NML$PRM_CHK_PROTOCOL.
!
!       V02-003 MKP0002         Kathy Perko             15-Dec-1981
!               Change name NML$PRS_CHK_PROTOCOL to NML$PRM_CHK_PROTOCOL
!               to be consistent with routine names in this module.
!
!       V02-002 MKP0001         Kathy Perko             19-Nov-1981
!               Add parameter grouping check for X.25 Protocol module.
!
!       V02-001 LMK0001         Len Kawell              27-Jul-1981
!               Remove line name parsing, as NETACP handles it now.
!--
```

```
1 %SBTTL 'Declarations'
1
1 !
1 ! TABLE OF CONTENTS:
1 !
1
1 FORWARD ROUTINE
1     nml$prm_check,
1     nml$prm_objprv,
1     nml$prm_clear,
1     nml$prm_strchk,
1     nml$prm_chkexe,
1     nml$prm_chknod,
1     nml$prm_chkrem,
1     nml$prm_chkloo,
1     nml$prm_chkefi,
1     nml$prm_chkesi,
1     nml$prm_circ_owner,
1     nml$prm_err,
1     nml$prm_evtsrctyp,
1     nml$prm_evtsource,
1     nml$prm_evtclass,
1     nml$prm_evtmsktyp,
1     nml$prm_evtmask,
1     nml$prm_chkeve,
1     nml$prm_save_node,
1     nml$prm_set_net,
1     nml$prm_qual_format,
1     nml$prm_channels;
1
1 !
1 ! INCLUDE FILES:
1 !
1
1 LIBRARY 'LIB$:NMLLIB.L32';
1 LIBRARY 'SHRLIB$:NMALIBRY.L32';
1 LIBRARY 'SYS$LIBRARY:STARLET.L32';
1
1 !
1 ! EXTERNAL REFERENCES:
1 !
1
1 $nml_extdef;
1
1 EXTERNAL ROUTINE
1     nml$chkexe,
1     nml$chkprmval,
1     nml$error_2,
1     nml$fix_node_num,
1     nml$getnodadr,
1     nml$saveparam;
1
```

```
1 %SBTTL 'NML$PRM_CHECK  Check parameter value'
1 GLOBAL ROUTINE NML$PRM_CHECK =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This is a NPARSE action routine that checks a parameter value
1 !       against its description in the parameter semantic table.  If
1 !       the parameter is valid, a descriptor entry is created.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NPARSE_BLOCK (pointed to by AP) contains the parsed parameter data.
1 !           NPA$L_FLDCNT is the parameter length.
1 !           NPA$L_FLDPTR is a pointer to the parameter in the received
1 !               message buffer.
1 !       NML$GL_PRMCODE contains the index into the change parameter table
1 !           (NML$AB_CPTABLE).
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       If the parameter is valid then a descriptor entry will be created for
1 !       it in NML$AW_PRM_DES and NML$GW_PRMDESCNT will be incremented.
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       Always returns NML$_STS_SUC.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       If the parameter fails the semantic check, an error message
1 !       (NMA$C_STS_PVA) is signalled.  If the parameter descriptor buffer
1 !       is full then a software error (NMA$C_STS_MPR) is signalled with
1 !       optional text to identify the error.
1 !
1 !--
1
2     BEGIN
2
2     $NPA_ARGDEF;                          ! Define NPARSE block reference
2
2     LOCAL
2         MSGSIZE,                          ! Resultant message size
2         LEN,
2         PTR;                              ! Temporary parameter pointer
2 !
2 ! Add descriptor entry for this parameter.
2 !
2     LEN = .NPARSE_BLOCK [NPA$L_FLDCNT];
2     PTR = .NPARSE_BLOCK [NPA$L_FLDPTR];
2
2     NML$CHKPRMVAL (.NML$GL_PRMCODE, .LEN, .PTR);
```

```
      2        NML$SAVEPARAM (.NML$GL_PRMCODE, .LEN, .PTR);
      2
      2        NML$GL_PRMCODE = 0;                        ! Reset parsing code
      2
      2        RETURN NML$_STS_SUC
      2
      1        END;                                       ! End of NML$PRM_CHECK

                                                   .TITLE  NML$PARPRM NML NPARSE action routines for parsi
;                                                                     ng paramete
                                                   .IDENT  \V04-000\

                                                   .EXTRN  NML$GB_EVTSRCTYP
                                                   .EXTRN  NML$GQ_EVTSRCDSC
                                                   .EXTRN  NML$GW_EVTCLASS
                                                   .EXTRN  NML$GB_EVTMSKTYP
                                                   .EXTRN  NML$GQ_EVTMSKDSC
                                                   .EXTRN  NML$GW_EVTSNKADR
                                                   .EXTRN  NML$GW_ACP_CHAN
                                                   .EXTRN  NML$GL_LOGMASK, NML$GQ_ENTSTRDSC
                                                   .EXTRN  NML$AB_QIOBUFFER
                                                   .EXTRN  NML$GQ_QIOBFDSC
                                                   .EXTRN  NML$AB_EXEBUFFER
                                                   .EXTRN  NML$GL_EXEDATPTR
                                                   .EXTRN  NML$GQ_EXEDATDSC
                                                   .EXTRN  NML$GQ_EXEBFDSC
                                                   .EXTRN  NML$AB_RCVBUFFER
                                                   .EXTRN  NML$GQ_RCVBFDSC
                                                   .EXTRN  NML$AB_SNDBUFFER
                                                   .EXTRN  NML$GQ_SNDBFDSC
                                                   .EXTRN  NML$GL_RCVDATLEN
                                                   .EXTRN  NML$AB_CPTABLE, NML$AB_MSGBLOCK
                                                   .EXTRN  NML$AB_ENTITY_ID
                                                   .EXTRN  NML$AB_QUALIFIER_ID
                                                   .EXTRN  NML$AB_ENTITYDATA
                                                   .EXTRN  NML$AB_NML_NMV, NML$AB_PRMSEM
                                                   .EXTRN  NML$AB_RECBUF, NML$AL_ENTINFTAB
                                                   .EXTRN  NML$AL_PERMINFTAB
                                                   .EXTRN  NML$AW_PRM_DES, NML$GB_CMD_VER
                                                   .EXTRN  NML$GB_ENTITY_CODE
                                                   .EXTRN  NML$GB_ENTITY_FORMAT
                                                   .EXTRN  NML$GL_QUALIFIER_PST
                                                   .EXTRN  NML$GB_QUALIFIER_FORMAT
                                                   .EXTRN  NML$GB_FUNCTION
                                                   .EXTRN  NML$GB_INFO, NML$GB_OPTIONS
                                                   .EXTRN  NML$GL_PRMCODE, NML$GL_PRS_FLGS
                                                   .EXTRN  NML$GL_NML_ENTITY
                                                   .EXTRN  NML$GQ_NETNAMDSC
                                                   .EXTRN  NML$GQ_RECBFDSC
                                                   .EXTRN  NML$GW_PRMDESCNT
                                                   .EXTRN  NML$CHKEXE, NML$CHKPRMVAL
                                                   .EXTRN  NML$ERROR_2, NML$FIX_NODE_NUM
                                                   .EXTRN  NML$GETNODADR, NML$SAVEPARAM

                                                   .PSECT  $CODE$,NOWRT,2
```

```
                              001C 00000     .ENTRY  NML$PRM_CHECK, Save R2,R3,R4               ; 0151
            54 00000000G  00    9E 00002     MOVAB   NML$GL_PRMCODE, R4                         ;
            53          10  AC    D0 00009     MOVL    16(NPARSE_BLOCK), LEN                      ; 0203
            52          14  AC    D0 0000D     MOVL    20(NPARSE_BLOCK), PTR                      ; 0204
                            52    DD 00011     PUSHL   PTR                                        ; 0206
                            53    DD 00013     PUSHL   LEN                                        ;
                            64    DD 00015     PUSHL   NML$GL_PRMCODE                             ;
     00000000G  00          03    FB 00017     CALLS   #3, NML$CHKPRMVAL                          ;
                            52    DD 0001E     PUSHL   PTR                                        ; 0207
                            53    DD 00020     PUSHL   LEN                                        ;
                            64    DD 00022     PUSHL   NML$GL_PRMCODE                             ;
     00000000G  00          03    FB 00024     CALLS   #3, NML$SAVEPARAM                          ;
                            64    D4 0002B     CLRL    NML$GL_PRMCODE                             ; 0209
                    50      01    D0 0002D     MOVL    #1, R0                                     ; 0211
                                  04 00030     RET                                                ; 0213

; Routine Size:  49 bytes,    Routine Base:  $CODE$ + 0000
```

```
1 %SBTTL 'NML$PRM_OBJPRV  Check object privilege parameter value'
1 GLOBAL ROUTINE NML$PRM_OBJPRV =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This is a NPARSE action routine that handles the object privilege
1 !       mask.  The mask is generally a quadword but only the lower longword
1 !       is currently used by VMS.  If this situation changes than this routine
1 !       must also be changed.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NPARSE_BLOCK (pointed to by AP) contains the parsed parameter data.
1 !           NPA$L_FLDCNT is the parameter length.
1 !           NPA$L_FLDPTR is a pointer to the parameter in the received
1 !               message buffer.
1 !       NML$GL_PRMCODE contains the index into the change parameter table
1 !           (NML$AB_CPTABLE).
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       If the parameter is valid then a descriptor entry will be created for
1 !       it in NML$AW_PRM_DES and NML$GW_PRMDESCNT will be incremented.
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       Always returns NML$_STS_SUC.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       If the parameter fails the semantic check, an error message
1 !       (NMA$C_STS_PVA) is signalled.  If the parameter descriptor buffer
1 !       is full then a software error (NMA$C_STS_MPR) is signalled with
1 !       optional text to identify the error.
1 !
1 !--
1
2     BEGIN
2
2     $NPA_ARGDEF;                            ! Define NPARSE block reference
2
2     LOCAL
2         MSGSIZE,                            ! Resultant message size
2         LEN,
2         PTR;                                ! Temporary parameter pointer
2 !
2 ! Add descriptor entry for this parameter.
2 !
2     LEN = .NPARSE_BLOCK [NPA$L_FLDCNT] - 1;
2
2     IF .LEN GTRU 4
```

```
;  274   0271 2     THEN
;  275   0272 2         LEN = 4;                           ! Maximum of four bytes
;  276   0273 2
;  277   0274 2     PTR = .NPARSE_BLOCK [NPA$L_FLDPTR] + 1;
;  278   0275 2     NML$SAVEPARAM (.NML$GL_PRMCODE, .LEN, .PTR);
;  279   0276 2
;  280   0277 2     NML$GL_PRMCODE = 0;                    ! Reset parsing code
;  281   0278 2
;  282   0279 2     RETURN NML$_STS_SUC
;  283   0280 2
;  284   0281 1     END;                                   ! End of NML$PRM_OBJPRV



                                        0004 00000          .ENTRY  NML$PRM_OBJPRV, Save R2                       ; 0215
                      52 00000000G  00    9E 00002          MOVAB   NML$GL_PRMCODE, R2
          51       10     AC        01    C3 00009          SUBL3   #1, 16(NPARSE_BLOCK), LEN                     ; 0268
                   04               51    D1 0000E          CMPL    LEN, #4                                       ; 0270
                                    03    1B 00011          BLEQU   1$                                            ;
                   51               04    D0 00013          MOVL    #4, LEN                                       ; 0272
          50       14     AC        01    C1 00016  1$:     ADDL3   #1, 20(NPARSE_BLOCK), PTR                     ; 0274
                                    50    DD 0001B          PUSHL   PTR                                           ; 0275
                                    51    DD 0001D          PUSHL   LEN                                           ;
                                    62    DD 0001F          PUSHL   NML$GL_PRMCODE                                ;
                00000000G  00       03    FB 00021          CALLS   #3, NML$SAVEPARAM                             ;
                                    62    D4 00028          CLRL    NML$GL_PRMCODE                                ; 0277
                   50               01    D0 0002A          MOVL    #1, R0                                        ; 0279
                                          04 0002D          RET                                                   ; 0281

; Routine Size:  46 bytes,    Routine Base:  $CODE$ + 0031
```

```
%SBTTL 'NML$PRM_STRCHK  Check parameter value'
GLOBAL ROUTINE NML$PRM_STRCHK =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This is a NPARSE action routine that checks a parameter value
!       against its description in the parameter semantic table.  If
!       the parameter is valid, a descriptor entry is created.
!
! FORMAL PARAMETERS:
!
!       NONE
!
! IMPLICIT INPUTS:
!
!       NPARSE_BLOCK (pointed to by AP) contains the parsed parameter data.
!            NPA$L_FLDCNT is the parameter length.
!            NPA$L_FLDPTR is a pointer to the parameter in the received
!                 message buffer.
!       NML$GL_PRMCODE contains the index into the change parameter table
!            (NML$AB_CPTABLE).
!
! IMPLICIT OUTPUTS:
!
!       If the parameter is valid then a descriptor entry will be created for
!       it in NML$AW_PRM_DES and NML$GW_PRMDESCNT will be incremented.
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       Always returns NML$_STS_SUC.
!
! SIDE EFFECTS:
!
!       If the parameter fails the semantic check, an error message
!       (NMA$C_STS_PVA) is signalled.  If the parameter descriptor buffer
!       is full then a software error (NMA$C_STS_MPR) is signalled with
!       optional text to identify the error.
!
!--

    BEGIN

    $NPA_ARGDEF;                            ! Define NPARSE block reference

    LOCAL
        MSGSIZE,                            ! Resultant message size
        LEN,
        PTR;                                ! Temporary parameter pointer
!
! Add descriptor entry for this parameter.
!
    LEN = .NPARSE_BLOCK [NPA$L_FLDCNT] - 1;
    PTR = .NPARSE_BLOCK [NPA$L_FLDPTR] + 1;

    NML$CHKPRMVAL (.NML$GL_PRMCODE, .LEN, .PTR);
```

```
;  343      0339 2       NML$SAVEPARAM (.NML$GL_PRMCODE, .LEN, .PTR);
;  344      0340 2
;  345      0341 2       NML$GL_PRMCODE = 0;                    ! Reset parsing code
;  346      0342 2
;  347      0343 2       RETURN NML$_STS_SUC
;  348      0344 2
;  349      0345 1       END;                                   ! End of NML$PRM_STRCHK


                                 001C 00000        .ENTRY  NML$PRM_STRCHK, Save R2,R3,R4                   ; 0283
                 54 00000000G  00   9E 00002        MOVAB   NML$GL_PRMCODE, R4
      53      10 AC             01   C3 00009        SUBL3   #1, 16(NPARSE_BLOCK), LEN                       ; 0335
      52      14 AC             01   C1 0000E        ADDL3   #1, 20(NPARSE_BLOCK), PTR                       ; 0336
                                52   DD 00013        PUSHL   PTR                                             ; 0338
                                53   DD 00015        PUSHL   LEN                                             ;
                                64   DD 00017        PUSHL   NML$GL_PRMCODE                                  ;
              00000000G  00     03   FB 00019        CALLS   #3, NML$CHKPRMVAL                               ;
                                52   DD 00020        PUSHL   PTR                                             ; 0339
                                53   DD 00022        PUSHL   LEN                                             ;
                                64   DD 00024        PUSHL   NML$GL_PRMCODE                                  ;
              00000000G  00     03   FB 00026        CALLS   #3, NML$SAVEPARAM                               ;
                                64   D4 0002D        CLRL    NML$GL_PRMCODE                                  ; 0341
                 50             01   D0 0002F        MOVL    #1, R0                                          ; 0343
                                     04 00032        RET                                                     ; 0345

; Routine Size:  51 bytes,    Routine Base:  $CODE$ + 005F
```

```
 1 %SBTTL 'NML$PRM_CLEAR  Remove entity parameter from data base (action routine)'
 1 GLOBAL ROUTINE NML$PRM_CLEAR =
 1
 1 !++
 1 ! FUNCTIONAL DESCRIPTION:
 1 !
 1 !       This is a NPARSE action routine that removes a parameter from the
 1 !       specified entity by creating a parameter descriptor.
 1 !
 1 ! FORMAL PARAMETERS:
 1 !
 1 !       NONE
 1 !
 1 ! IMPLICIT INPUTS:
 1 !
 1 !       NPARSE argument block.
 1 !
 1 ! IMPLICIT OUTPUTS:
 1 !
 1 !       A parameter descriptor is added to NML$AW_PRM_DES and the count
 1 !       (NML$GW_PRMDESCNT) is incremented.
 1 !
 1 ! ROUTINE VALUE:
 1 ! COMPLETION CODES:
 1 !
 1 !       Always returns success (NML$_STS_SUC).
 1 !
 1 ! SIDE EFFECTS:
 1 !
 1 !       If no room remains in the descriptor area then a software error
 1 !       (NMA$C_STS_MPR) will be signalled indicating buffer overflow.
 1 !
 1 !--
 1
 2    BEGIN
 2
 2    $NPA_ARGDEF;                              ! Define NPARSE block reference
 2 !
 2 ! Add descriptor.
 2 !
 2    NML$GL_PRMCODE = .NPARSE_BLOCK [NPA$L_PARAM];
 2    NML$SAVEPARAM (.NML$GL_PRMCODE, 0, 0);
 2    NML$GL_PRMCODE = 0;
 2
 2    RETURN NML$_STS_SUC
 2
 1    END;                                      ! End of NML$PRM_CLEAR


                              0004 00000        .ENTRY  NML$PRM_CLEAR, Save R2                        ; 0347
             52 00000000G  00   9E 00002        MOVAB   NML$GL_PRMCODE, R2                            ;
             62         20 AC   D0 00009        MOVL    32(NPARSE_BLOCK), NML$GL_PRMCODE              ; 0386
                           7E   7C 0000D        CLRQ    -(SP)                                         ; 0387
                           62   DD 0000F        PUSHL   NML$GL_PRMCODE                                ;
```

```
                0000000G  00         03  FB 00011        CALLS   #3, NML$SAVEPARAM                ;
                                     62  D4 00018        CLRL    NML$GL_PRMCODE                   ; 0388
                          50         01  D0 0001A        MOVL    #1, R0                           ; 0390
                                         04 0001D        RET                                      ; 0392

; Routine Size:  30 bytes,    Routine Base:  $CODE$ + 0092


;  398          0393  1
```

```
1 %SBTTL 'NML$PRM_CHKEXE  Check executor parameter group'
1 GLOBAL ROUTINE NML$PRM_CHKEXE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This is a NPARSE action routine that checks the parameter
1 !       code to make sure it is in the same group (executor-only or
1 !       node) as any previously specified parameters.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NPARSE_BLOCK (pointed to by AP) contains the parsed parameter data.
1 !           NPA$L_FLDPTR is a pointer code to the parameter in the received
1 !               message buffer.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       If the parameter is in the correct group then the flag for that
1 !       group will be set in NML$GL_PRS_FLGS.
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       Always returns NML$_STS_SUC.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       If the parameter is in the wrong group then an invalid parameter
1 !       grouping applicable error (NMA$C_STS_PGP) will be signalled.
1 !
1 !--
1
2   BEGIN
2
2   $NPA_ARGDEF;                        ! Define NPARSE block reference
2
2   LOCAL
2       MSGSIZE;
2 !
2 ! If this is not the executor-only parameter group return error.
2 !
2   IF .NML$GL_FRS_FLGS [NML$V_PRS_NODPG]
2   THEN
2       NML$ERROR_2 (NMA$C_STS_PGP,
2                   .(.NPARSE_BLOCK [NPA$L_FLDPTR])<0,16>);
2
2   NML$GL_PRS_FLGS [NML$V_PRS_EXEPG] = 1; ! Set executor flag
2
2   RETURN NML$_STS_SUC
2
1   END;                                ! End of NML$PRM_CHKEXE
```

```
                                    0000 00000          .ENTRY  NML$PRM_CHKEXE, Save nothing                    ; 0395
      0E 00000000G  00        01    E1 00002          BBC     #1, NML$GL_PRS_FLGS+1, 1$                       ; 0440
                    7E   14   BC    3C 0000A          MOVZWL  @20(NPARSE_BLOCK), -(SP)                         ; 0443
                    7E        1B    CE 0000E          MNEGL   #27, -(SP)                                       ; 0442
         00000000G  00        02    FB 00011          CALLS   #2, NML$ERROR_2                                  ;
         00000000G  00        01    88 00018 1$:      BISB2   #1, NML$GL_PRS_FLGS+1                            ; 0445
                    50        01    D0 0001F          MOVL    #1, R0                                           ; 0447
                                    04 00022          RET                                                      ; 0449

; Routine Size:  35 bytes,    Routine Base:  $CODE$ + 00B0
```

```
%SBTTL 'NML$PRM_CHKNOD  Check node parameter group'
GLOBAL ROUTINE NML$PRM_CHKNOD =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This is a NPARSE action routine that checks the executor parameter
!       code to make sure it is in the same group (executor-only or
!       node) as any previously specified parameters.
!
! FORMAL PARAMETERS:
!
!       NONE
!
! IMPLICIT INPUTS:
!
!       NPARSE_BLOCK (pointed to by AP) contains the parsed parameter data.
!               NPA$L_FLDPTR is a pointer code to the parameter in the received
!                       message buffer.
!
! IMPLICIT OUTPUTS:
!
!       If the parameter is in the correct group then the flag for that
!       group will be set in NML$GL_PRS_FLGS.
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       Always returns NML$_STS_SUC.
!
! SIDE EFFECTS:
!
!       If the parameter is in the wrong group then an invalid parameter
!       grouping applicable error (NMA$C_STS_PGP) will be signalled.
!
!--

    BEGIN

    $NPA_ARGDEF;                            ! Define NPARSE block reference

    LOCAL
        MSGSIZE;
!
! If this is not the executor-only parameter group return error.
!
    IF .NML$GL_PRS_FLGS [NML$V_PRS_EXEPG]
    THEN
        NML$ERROR_2 (NMA$C_STS_PGP,
                     .(.NPARSE_BLOCK [NPA$L_FLDPTR])<0,16>);

    NML$GL_PRS_FLGS [NML$V_PRS_NODPG] = 1; ! Set node flag

    RETURN NML$_STS_SUC

    END;                                    ! End of NML$PRM_CHKNOD
```

```
                                   0000 00000          .ENTRY  NML$PRM_CHKNOD, Save nothing              ; 0451
              0E 00000000G  00       E9 00002          BLBC    NML$GL_PRS_FLGS+1, 1$                     ; 0496
              7E         14  BC      3C 00009          MOVZWL  @20(NPARSE_BLOCK), -(SP)                  ; 0499
              7E             1B      CE 0000D          MNEGL   #27, -(SP)                                ; 0498
    00000000G 00             02      FB 00010          CALLS   #2, NML$ERROR_2                           ;
    00000000G 00             02      88 00017 1$:      BISB2   #2, NML$GL_PRS_FLGS+1                     ; 0501
              50             01      D0 0001E          MOVL    #1, R0                                    ; 0503
                                     04 00021          RET                                               ; 0505

; Routine Size:  34 bytes,    Routine Base:  $CODE$ + 00D3
```

```
1 %SBTTL 'NML$PRM_CHKREM  Check remote node parameter group'
1 GLOBAL ROUTINE NML$PRM_CHKREM =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This is a NPARSE action routine that checks the remote node parameter
1 !       code to make sure it is in the same group (loop or remote
1 !       node) as any previously specified parameters.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NPARSE_BLOCK (pointed to by AP) contains the parsed parameter data.
1 !           NPA$L_FLDPTR is a pointer code to the parameter in the received
1 !               message buffer.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       If the parameter is in the correct group then the flag for that
1 !       group will be set in NML$GL_PRS_FLGS.
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       Always returns NML$_STS_SUC.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       If the parameter is in the wrong group then an invalid parameter
1 !       grouping applicable error (NMA$C_STS_PGP) will be signalled.
1 !
1 !--
1
2     BEGIN
2
2     $NPA_ARGDEF;                              ! Define NPARSE block reference
2
2     LOCAL
2         MSGSIZE;
2 !
2 ! If this is the loop node parameter group return error.
2 !
2     IF .NML$GL_PRS_FLGS [NML$V_PRS_LOOPG]
2     THEN
2         NML$ERROR_2 (NMA$C_STS_PNA,
2                 .(.NPARSE_BLOCK [NPA$L_FLDPTR])<0,16>);
2
2     NML$GL_PRS_FLGS [NML$V_PRS_REMPG] = 1; ! Set remote node flag
2
2     RETURN NML$_STS_SUC
2
1     END;                                      ! End of NML$PRM_CHKREM
```

```
                                      0000 00000        .ENTRY  NML$PRM_CHKREM, Save nothing          ; 0507
          0E 00000000G  00              03  E1 00002        BBC     #3, NML$GL_PRS_FLGS+1, 1$             ; 0552
                        7E        14    BC  3C 0000A        MOVZWL  @20(NPARSE_BLOCK), -(SP)              ; 0555
                        7E              16  CE 0000E        MNEGL   #22, -(SP)                            ; 0554
             00000000G  00              02  FB 00011        CALLS   #2, NML$ERROR_2                       ;
             00000000G  00              04  88 00018 1$:    BISB2   #4, NML$GL_PRS_FLGS+1                 ; 0557
                        50              01  D0 0001F        MOVL    #1, R0                                ; 0559
                                            04 00022        RET                                           ; 0561

; Routine Size:  35 bytes,    Routine Base:  $CODE$ + 00F5
```

```
1 %SBTTL 'NML$PRM_CHKLOO  Check loop node parameter group'
1 GLOBAL ROUTINE NML$PRM_CHKLOO =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This is a NPARSE action routine that checks the loop node parameter
1 !       code to make sure it is in the same group (remote or loop
1 !       node) as any previously specified parameters.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NPARSE_BLOCK (pointed to by AP) contains the parsed parameter data.
1 !           NPA$L_FLDPTR is a pointer code to the parameter in the received
1 !               message buffer.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       If the parameter is in the correct group then the flag for that
1 !       group will be set in NML$GL_PRS_FLGS.
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       Always returns NML$_STS_SUC.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       If the parameter is in the wrong group then a parameter not
1 !       applicable error (NMA$C_STS_PNA) will be signalled.
1 !
1 !--
1
2     BEGIN
2
2     $NPA_ARGDEF;                          ! Define NPARSE block reference
2
2     MAP
2         NML$GB_OPTIONS : BBLOCK [1];
2
2     EXTERNAL LITERAL
2         CPT$GK_PCNO_A$$;                  ! Loop node address
2
2     BIND
2         NODADR = UPLIT WORD (0) : WORD;
2 !
2 ! If this is the remote node parameter group return error.
2 !
2     IF .NML$GL_PRS_FLGS [NML$V_PRS_REMPG]
2     THEN
2         NML$ERROR_2 (NMA$C_STS_PNA,
2                      .(.NPARSE_BLOCK [NPA$L_FLDPTR])<0,16>);
2
```

```
 2      NML$GL_PRS_FLGS [NML$V_PRS_LOOPG] = 1; ! Set loop node flag
 2 !
 2 ! If this is a SET/DEFINE (not CLEAR/PURGE) operation then add a zero
 2 ! node address to get around an idiosyncracy in NETACP.
 2 !
 2      IF NOT .NML$GB_OPTIONS [NMA$V_OPT_CLE]
 2      THEN
 2          NML$SAVEPARAM (CPT$GK_PCNO_A$$, 2, NODADR);
 2
 2      RETURN NML$_STS_SUC
 2
 1      END;                                     ! End of NML$PRM_CHKLOO


                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

                          0000  00000 P.AAA:    .WORD   0                                        ;

                                      NODADR=           P.AAA
                                                .EXTRN  CPT$GK_PCNO_A$$

                                                .PSECT  $CODE$,NOWRT,2

                                0000  00000     .ENTRY  NML$PRM_CHKLOO, Save nothing          ; 0563
 0E 00000000G  00           02  E1  00002       BBC     #2, NML$GL_PRS_FLGS+1, 1$             ; 0614
            7E      14      BC  3C  0000A       MOVZWL  @20(NPARSE_BLOCK), -(SP)              ; 0617
            7E              16  CE  0000E       MNEGL   #22, -(SP)                            ; 0616
    00000000G  00           02  FB  00011       CALLS   #2, NML$ERROR_2                       ;
    00000000G  00           08  88  00018 1$:   BISB2   #8, NML$GL_PRS_FLGS+1                 ; 0619
 15 00000000G  00           06  E0  0001F       BBS     #6, NML$GB_OPTIONS, 2$                ; 0624
                00000000'   00  9F  00027       PUSHAB  NODADR                                ; 0626
                            02  DD  0002D       PUSHL   #2                                    ;
                00000000G   8F  DD  0002F       PUSHL   #CPT$GK_PCNO_A$$                      ;
    00000000G  00           03  FB  00035       CALLS   #3, NML$SAVEPARAM                     ;
            50              01  D0  0003C 2$:   MOVL    #1, R0                                ; 0628
                                04  0003F       RET                                           ; 0630

; Routine Size:  64 bytes,    Routine Base:  $CODE$ + 0118
```

```
1 %SBTTL 'NML$PRM_CHKKNO  Check for KNOWN entity option'
1 GLOBAL ROUTINE NML$PRM_CHKKNO =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This is a NPARSE action routine that checks the parameter
1 !       code to make sure it is not applied to KNOWN parameters.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NPARSE_BLOCK (pointed to by AP) contains the parsed parameter data.
1 !           NPA$L_FLDPTR is a pointer code to the parameter in the received
1 !                   message buffer.
1 !       NML$GB_ENTITY_FORMAT contains the entity format code.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       Always returns NML$_STS_SUC.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       If KNOWN entities is specified then a parameter not applicable
1 !       error (NMA$C_STS_PNA) is signalled.
1 !
1 !--
1
2    BEGIN
2
2    $NPA_ARGDEF;                              ! Define NPARSE block reference
2
2    MAP
2       NML$GB_ENTITY_FORMAT : BYTE SIGNED;
2
2    LOCAL
2       MSGSIZE;
2 !
2 ! If KNOWN entities is selected then return error.
2 !
2    IF .NML$GB_ENTITY_FORMAT EQL NMA$C_ENT_KNO
2    THEN
2       NML$ERROR_2 (NMA$C_STS_PNA,
2                   .(.NPARSE_BLOCK [NPA$L_FLDPTR])<0,16>);
2
2    RETURN NML$_STS_SUC
2
1    END;                                      ! End of NML$PRM_CHKKNO
```

```
                                      0000 00000           .ENTRY  NML$PRM_CHKKNO, Save nothing                     ; 0632
              FF    8F 00000000G  00    91 00002           CMPB    NML$GB_ENTITY_FORMAT, #-1                        ; 0679
                                  0E    12 0000A           BNEQ    1$                                               ;
                 7E         14    BC    3C 0000C           MOVZWL  @20(NPARSE_BLOCK), -(SP)                         ; 0682
                 7E               16    CE 00010           MNEGL   #22, -(SP)                                       ; 0681
       00000000G 00               02    FB 00013           CALLS   #2, NML$ERROR_2                                  ;
                 50               01    D0 0001A 1$:       MOVL    #1, R0                                           ; 0684
                                        04 0001D           RET                                                      ; 0686

; Routine Size:  30 bytes,    Routine Base:  $CODE$ + 0158
```

```
%SBTTL 'NML$PRM_CHKEFI  Check event logging parameter group'
GLOBAL ROUTINE NML$PRM_CHKEFI =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This is a NPARSE action routine that checks the parameter
!       code to make sure it is in the same group (event logging
!       information) as any previously specified parameters.
!
! FORMAL PARAMETERS:
!
!       NONE
!
! IMPLICIT INPUTS:
!
!       NPARSE_BLOCK (pointed to by AP) contains the parsed parameter data.
!           NPA$L_FLDPTR is a pointer code to the parameter in the received
!               message buffer.
!
! IMPLICIT OUTPUTS:
!
!       If the parameter is in the correct group then the flag for that
!       group will be set in NML$GL_PRS_FLGS.
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       Always returns NML$_STS_SUC.
!
! SIDE EFFECTS:
!
!       If the parameter is in the wrong group then an invalid parameter
!       grouping applicable error (NMA$C_STS_PGP) will be signalled.
!
!--

    BEGIN

    $NPA_ARGDEF;                        ! Define NPARSE block reference

    LOCAL
        MSGSIZE;
!
! If this is not the event logging information parameter group return error.
!
    IF .NML$GL_PRS_FLGS [NML$V_PRS_ESIPG]
    THEN
        NML$ERROR_2 (NMA$C_STS_PGP,
                     .(.NPARSE_BLOCK [NPA$L_FLDPTR])<0,16>);

    NML$GL_PRS_FLGS [NML$V_PRS_EFIPG] = 1; ! Set group flag

    RETURN NML$_STS_SUC

    END;                                ! End of NML$PRM_CHKEFI
```

```
                                              0000 00000          .ENTRY  NML$PRM_CHKEFI, Save nothing                        ; 0688
           0E 00000000G  00             04   E1 00002          BBC     #4, NML$GL_PRS_FLGS+1, 1$                            ; 0733
                         7E        14   BC   3C 0000A          MOVZWL  @20(NPARSE_BLOCK), -(SP)                            ; 0736
                         7E             1B   CE 0000E          MNEGL   #27, -(SP)                                           ; 0735
              00000000G  00             02   FB 00011          CALLS   #2, NML$ERROR_2                                      ;
              00000000G  00             08   88 00018 1$:      BISB2   #8, NML$GL_PRS_FLGS+1                                ; 0738
                         50             01   D0 0001F          MOVL    #1, R0                                               ; 0740
                                             04 00022          RET                                                          ; 0742

; Routine Size:  35 bytes,     Routine Base:  $CODE$ + 0176
```

```
1 %SBTTL 'NML$PRM_CHKESI  Check event sink parameter group'
1 GLOBAL ROUTINE NML$PRM_CHKESI =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This is a NPARSE action routine that checks the parameter
1 !       code to make sure it is in the same group (event sink
1 !       information) as any previously specified parameters.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NPARSE_BLOCK (pointed to by AP) contains the parsed parameter data.
1 !            NPA$L_FLDPTR is a pointer code to the parameter in the received
1 !                 message buffer.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       If the parameter is in the correct group then the flag for that
1 !       group will be set in NML$GL_PRS_FLGS.
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       Always returns NML$_STS_SUC.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       If the parameter is in the wrong group then an invalid parameter
1 !       grouping applicable error (NMA$C_STS_PGP) will be signalled.
1 !
1 !--
1
2    BEGIN
2
2    $NPA_ARGDEF;                          ! Define NPARSE block reference
2
2    LOCAL
2       MSGSIZE;
2 !
2 ! If this is not the event sink information parameter group return error.
2 !
2    IF .NML$GL_PRS_FLGS [NML$V_PRS_EFIPG]
2    THEN
2       NML$ERROR_2 (NMA$C_STS_PGP,
2                   .(.NPARSE_BLOCK [NPA$L_FLDPTR])<0,16>);
2
2    NML$GL_PRS_FLGS [NML$V_PRS_ESIPG] = 1; ! Set group flag
2
2    RETURN NML$_STS_SUC
2
1    END;                                  ! End of NML$PRM_CHKESI
```

```
                                 0000 00000          .ENTRY  NML$PRM_CHKESI, Save nothing                    ; 0744
        0E 00000000G  00      03  E1 00002          BBC     #3, NML$GL_PRS_FLGS+1, 1$                        ; 0789
                      7E  14  BC  3C 0000A          MOVZWL  @20(NPARSE_BLOCK), -(SP)                          ; 0792
                      7E      1B  CE 0000E          MNEGL   #27, -(SP)                                        ; 0791
           00000000G  00      02  FB 00011          CALLS   #2, NML$ERROR_2                                   ;
           00000000G  00      10  88 00018 1$:      BISB2   #16, NML$GL_PRS_FLGS+1                            ; 0794
                      50      01  D0 0001F          MOVL    #1, R0                                            ; 0796
                                  04 00022          RET                                                       ; 0798

; Routine Size:  35 bytes,    Routine Base:  $CODE$ + 0199
```

```
1 %SBTTL 'NML$PRM_CIRC_OWNER  Convert circuit owner parameter(action routine)'
1 GLOBAL ROUTINE NML$PRM_CIRC_OWNER =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !       This is a NPARSE action routine that is called when parsing a
1 !       SET CIRCUIT OWNER command.  It checks the node ID supplied
1 !       to make sure it specifies the executor node (the only value
1 !       currently allowed for OWNER).  This routine saves the OWNER
1 !       parameter as a set bit field since only one value is allowed.
1 !
1 ! IMPLICIT INPUTS:
1 !       NPARSE argument block.
1 !               NPA$L_PARAM contains the error code.
1 !               NPA$L_FLDPTR points to the parameter in the message.
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !       Signals NML$_STS_PVA if the node id is not the executor's
1 !       (for now, we only support the executor as a circuit owner).
1 !       Otherwise returns success.
1 !
1 !--
1
2 BEGIN
2
2 $NPA_ARGDEF;                          ! Define NPARSE block reference
2
2 LOCAL
2     len,
2     node_addr : WORD,
2     name_addr;
2
2 node_addr = 0;
2 len = .(.nparse_block [npa$l_fldptr])<0,8>;
2 IF .nparse_block [npa$l_param] EQL nma$c_pcno_add THEN
2     node_addr = .(.nparse_block [npa$l_fldptr]+1)<0,16>
2 ELSE
2     name_addr = .nparse_block [npa$l_fldptr] + 1;
2 !
2 ! If the node address in the NICE command matches the executor node
2 ! address then set the flag to indicate it.
2 !
2 IF nml$chkexe (.nparse_block [npa$l_param], .node_addr, .len, .name_addr) THEN
2     nml$saveparam (.nml$gl_prmcode, 1, UPLIT BYTE (1))
2 ELSE
2     nml$error_2 (nma$c_sts_pva,
2                 nma$c_pcci_own);
2 nml$gl_prmcode = 0;                           ! Reset parsing code
2 RETURN nml$_sts_suc;                          ! Return status
1 END;                                  ! End of NML$PRM_CIRC_OWNER
```

```
                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                          01  00002 P.AAB:  .BYTE   1                                    ;
```

```
                                                        .PSECT  $CODE$,NOWRT,2

                                     000C 00000         .ENTRY  NML$PRM_CIRC_OWNER, Save R2,R3                  ; 0800
              53 00000000G        00   9E 00002         MOVAB   NML$GL_PRMCODE, R3                              ;
                                  52   B4 00009         CLRW    NODE_ADDR                                       ; 0832
              50          14      AC   D0 0000B         MOVL    20(NPARSE_BLOCK), R0                            ; 0833
              51                  60   9A 0000F         MOVZBL  (R0), LEN                                       ;
   000001F6   8F          20      AC   D1 00012         CMPL    32(NPARSE_BLOCK), #502                          ; 0834
                                  06   12 0001A         BNEQ    1$                                              ;
              52          01      A0   B0 0001C         MOVW    1(R0), NODE_ADDR                                ; 0835
                                  02   11 00020         BRB     2$                                              ;
                                  50   D6 00022 1$:     INCL    NAME_ADDR                                       ; 0837
                                  50   DD 00024 2$:     PUSHL   NAME_ADDR                                       ; 0842
                                  51   DD 00026         PUSHL   LEN                                             ;
              7E                  52   3C 00028         MOVZWL  NODE_ADDR, -(SP)                                ;
                          20      AC   DD 0002B         PUSHL   32(NPARSE_BLOCK)                                ;
   00000000G  00                  04   FB 0002E         CALLS   #4, NML$CHKEXE                                  ;
              13                  50   E9 00035         BLBC    R0, 3$                                          ;
                    00000000'     00   9F 00038         PUSHAB  P.AAB                                           ; 0843
                                  01   DD 0003E         PUSHL   #1                                              ;
                                  63   DD 00040         PUSHL   NML$GL_PRMCODE                                  ;
   00000000G  00                  03   FB 00042         CALLS   #3, NML$SAVEPARAM                               ;
                                  0F   11 00049         BRB     4$                                              ;
              7E        044C      8F   3C 0004B 3$:     MOVZWL  #1100, -(SP)                                    ; 0845
              7E                  10   CE 00050         MNEGL   #16, -(SP)                                      ;
   00000000G  00                  02   FB 00053         CALLS   #2, NML$ERROR_2                                 ;
                                  63   D4 0005A 4$:     CLRL    NML$GL_PRMCODE                                  ; 0847
              50                  01   D0 0005C         MOVL    #1, R0                                          ; 0848
                                       04 0005F         RET                                                     ; 0849

; Routine Size:  96 bytes,    Routine Base:  $CODE$ + 01BC
```

```
1 %SBTTL 'NML$PRM_ERR  Build and signal error (action routine)'
1 GLOBAL ROUTINE NML$PRM_ERR =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This is a NPARSE action routine that signals parameter errors.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NPARSE argument block.
1 !
1 !               NPA$L_PARAM contains the error code.
1 !               NPA$L_FLDPTR points to the parameter in the message.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       Always returns success (NML$_STS_SUC).
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Error message is signalled.
1 !
1 !--
1
2   BEGIN
2
2   $NPA_ARGDEF;                            ! Define NPARSE block reference
2
2   LOCAL
2       ERR_CODE,                           ! Error code
2       ERR_DETAIL;                         ! Error detail
2
2   ERR_CODE = .NPARSE_BLOCK [NPA$L_PARAM]; ! Get error code
2 !
2 ! Check for parameters to move in addition to error status.
2 !
3   ERR_DETAIL = (
3       SELECTONEU .ERR_CODE OF
3           SET
3
3           [NMA$C_STS_PTY,
3            NMA$C_STS_PVA,
3            NMA$C_STS_PNA]:
3
3               .(.NPARSE_BLOCK [NPA$L_MSGPTR] - 2)<0,16,0>; ! Get detail code
3
3           [OTHERWISE]:
```

```
;  921          0907  3                  -1;
;  922          0908  3
;  923          0909  3              TES);
;  924          0910  2
;  925          0911  2          NML$ERROR_2 (.ERR_CODE, .ERR_DETAIL); ! Signal error message
;  926          0912  2
;  927          0913  2          RETURN NML$_STS_SUC                   ! Return success
;  928          0914  2
;  929          0915  2          END;                                  ! End of NML$PRM_ERR
;  930          0916  1



                                  0000 00000           .ENTRY  NML$PRM_ERR, Save nothing                    ; 0851
                  51       20  AC  D0 00002           MOVL    32(NPARSE_BLOCK), ERR_CODE                    ; 0892
        FFFFFFEA  8F           51  D1 00006           CMPL    ERR_CODE, #-22                                ; 0900
                               12  13 0000D           BEQL    1$                                            ;
        FFFFFFF0  8F           51  D1 0000F           CMPL    ERR_CODE, #-16                                ;
                               09  13 00016           BEQL    1$                                            ;
        FFFFFFFA  8F           51  D1 00018           CMPL    ERR_CODE, #-6                                 ;
                               0A  12 0001F           BNEQ    2$                                            ;
                  50       08  AC  D0 00021  1$:      MOVL    8(NPARSE_BLOCK), R0                           ; 0904
                  50       FE  A0  3C 00025           MOVZWL  -2(R0), ERR_DETAIL                            ;
                               03  11 00029           BRB     3$                                            ;
                  50           01  CE 0002B  2$:      MNEGL   #1, ERR_DETAIL                                ; 0908
                               50  DD 0002E  3$:      PUSHL   ERR_DETAIL                                    ; 0912
                               51  DD 00030           PUSHL   ERR_CODE                                      ;
        00000000G 00           02  FB 00032           CALLS   #2, NML$ERROR_2                               ;
                  50           01  D0 00039           MOVL    #1, R0                                        ; 0914
                                   04 0003C           RET                                                   ; 0916

; Routine Size:  61 bytes,    Routine Base:  $CODE$ + 021C


;  931          0917  1
```

```
1 %SBTTL 'NML$PRM_EVTSRCTYP  Get event source type code'
1 GLOBAL ROUTINE NML$PRM_EVTSRCTYP =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This is an NPARSE action routine  that saves the event source type
1 !       code.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NPARSE_BLOCK (pointed to by AP) contains the parsed parameter data.
1 !           NPA$L_FLDCNT is the parameter length.
1 !           NPA$L_FLDPTR is a pointer to the parameter in the received
1 !               message buffer.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       Always returns success (NML$_STS_SUC).
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
1
2     BEGIN
2
2     $NPA_ARGDEF;                            ! Define NPARSE block reference
2 !
2 ! Save the event source type code.
2 !
2     NML$GB_EVTSRCTYP = .(.NPARSE_BLOCK [NPA$L_FLDPTR])<0,8>;
2     NML$GL_PRS_FLGS [NML$V_PRS_EVE] = 1; ! Flag event parameter processed
2
2     RETURN NML$_STS_SUC
2
1     END;                                    ! End of NML$PRM_EVTSRCTYP


                                 0000 00000        .ENTRY  NML$PRM_EVTSRCTYP, Save nothing            ; 0919
00000000G  00         14    BC  90 00002        MOVB    @20(NPARSE_BLOCK), NML$GB_EVTSRCTYP        ; 0958
00000000G  00               20  88 0000A        BISB2   #32, NML$GL_PRS_FLGS+1                     ; 0959
       50                   01  D0 00011        MOVL    #1, R0                                     ; 0961
                                04 00014        RET                                                ; 0963
```

```
; Routine Size:  21 bytes,    Routine Base:  $CODE$ + 0259
```

```
%SBTTL 'NML$PRM_EVTSOURCE  Get event source descriptor'
GLOBAL ROUTINE NML$PRM_EVTSOURCE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This is an NPARSE action routine that saves the event source id.
!
! FORMAL PARAMETERS:
!
!       NONE
!
! IMPLICIT INPUTS:
!
!       NPARSE_BLOCK (pointed to by AP) contains the parsed parameter data.
!               NPA$L_FLDCNT is the parameter length.
!               NPA$L_FLDPTR is a pointer to the parameter in the received
!                       message buffer.
!
! IMPLICIT OUTPUTS:
!
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       Always returns success (NML$_STS_SUC).
!
! SIDE EFFECTS:
!
!       NONE
!
!--

BEGIN

$npa_argdef;                            ! Define NPARSE block reference

LOCAL
    addr    : WORD,
    namlen,
    namptr;
!
! Save the event source string descriptor information.
!
IF .nml$gb_evtsrctyp EQLU nma$c_ent_lin
   OR .nml$gb_evtsrctyp EQLU nma$c_ent_cir THEN
    BEGIN
    nml$gq_evtsrcdsc [dsc$w_length] =
            .(.nparse_block [npa$l_fldptr])<0,8>;
    nml$gq_evtsrcdsc [dsc$a_pointer] =
            .nparse_block [npa$l_fldptr] + 1;
    END
ELSE
    !
    ! The event source is node.  Save the node id.  If the user specified
    ! the node name (rather than the node address), get the node's address
    ! from the DECnet node database.
```

```
; 1037      1021  2      !
; 1038      1022  3      BEGIN
; 1039      1023  3      IF .(.nparse_block [npa$l_fldptr])<0,8> EQLU 0 THEN
; 1040      1024  4          BEGIN
; 1041      1025  4          nml$gq_evtsrcdsc [dsc$a_pointer] =
; 1042      1026  4                  .(.nparse_block [npa$l_fldptr] + 1)<0,16>;
; 1043      1027  4          nml$fix_node_num (nml$gq_evtsrcdsc [dsc$a_pointer]);
; 1044      1028  4          END
; 1045      1029  3      ELSE
; 1046      1030  4          BEGIN
; 1047      1031  4          namlen = .(.nparse_block [npa$l_fldptr])<0,8>;
; 1048      1032  4          namptr = .nparse_block [npa$l_fldptr] + 1;
; 1049      1033  4
; 1050      1034  4          IF nml$getnodadr (.namlen, .namptr, addr) THEN
; 1051      1035  4              nml$gq_evtsrcdsc [dsc$a_pointer] = .addr
; 1052      1036  4          ELSE
; 1053      1037  4              nml$error_2 (nma$c_sts_ide, nma$c_ent_nod);
; 1054      1038  3          END;
; 1055      1039  3
; 1056      1040  3      nml$gq_evtsrcdsc [dsc$w_length] = 0;
; 1057      1041  2      END;
; 1058      1042  2
; 1059      1043  2  RETURN nml$_sts_suc
; 1060      1044  1  END;                               ! End of NML$PRM_EVTSOURCE


                                    0004 00000          .ENTRY  NML$PRM_EVTSOURCE, Save R2                      ; 0965
           52 00000000G        00     9E 00002          MOVAB   NML$GQ_EVTSRCDSC+4, R2                          ;
           5E                  04     C2 00009          SUBL2   #4, SP                                          ;
           50 00000000G        00     9A 0000C          MOVZBL  NML$GB_EVTSRCTYP, R0                            ; 1008
           01                  50     91 00013          CMPB    R0, #1                                          ;
                               05     13 00016          BEQL    1$                                              ;
           03                  50     91 00018          CMPB    R0, #3                                          ; 1009
                               0C     12 0001B          BNEQ    2$                                              ;
              FC    A2    14   BC     9B 0001D 1$:      MOVZBW  @20(NPARSE_BLOCK), NML$GQ_EVTSRCDSC             ; 1012
     62       14    AC         01     C1 00022          ADDL3   #1, 20(NPARSE_BLOCK), NML$GQ_EVTSRCDSC+4        ; 1014
                               3E     11 00027          BRB     6$                                              ; 1008
           50         14       AC     D0 00029 2$:      MOVL    20(NPARSE_BLOCK), R0                            ; 1023
                               60     95 0002D          TSTB    (R0)                                            ;
                               0F     12 0002F          BNEQ    3$                                              ;
           62         01       A0     3C 00031          MOVZWL  1(R0), NML$GQ_EVTSRCDSC+4                       ; 1026
                               52     DD 00035          PUSHL   R2                                              ; 1027
        00000000G  00          01     FB 00037          CALLS   #1, NML$FIX_NODE_NUM                            ;
                               24     11 0003E          BRB     5$                                              ; 1023
           51                  80     9A 00040 3$:      MOVZBL  (R0)+, NAMLEN                                   ; 1031
                    4001       8F     BB 00043          PUSHR   #^M<R0,SP>                                      ; 1034
                               51     DD 00047          PUSHL   NAMLEN                                          ;
        00000000G  00          03     FB 00049          CALLS   #3, NML$GETNODADR                               ;
                   05          50     E9 00050          BLBC    R0, 4$                                          ;
                   62          6E     3C 00053          MOVZWL  ADDR, NML$GQ_EVTSRCDSC+4                        ; 1035
                               0C     11 00056          BRB     5$                                              ;
                               7E     D4 00058 4$:      CLRL    -(SP)                                           ; 1037
                   7E          09     CE 0005A          MNEGL   #9, -(SP)                                       ;
        00000000G  00          02     FB 0005D          CALLS   #2, NML$ERROR_2                                 ;
```

```
                     FC     A2  B4 00064 5$:     CLRW    NML$GQ_EVTSRCDSC                     ; 1040
             50             01  D0 00067 6$:     MOVL    #1, R0                               ; 1043
                                04 0006A         RET                                          ; 1044

; Routine Size:  107 bytes,    Routine Base:  $CODE$ + 026E
```


; 1

; R

```
1 %SBTTL 'NML$PRM_EVTCLASS  Get event class code'
1 GLOBAL ROUTINE NML$PRM_EVTCLASS =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This is a NPARSE action routine that saves the event class
1 !       information.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NPARSE_BLOCK (pointed to by AP) contains the parsed parameter data.
1 !           NPA$L_FLDCNT is the parameter length.
1 !           NPA$L_FLDPTR is a pointer to the parameter in the received
1 !               message buffer.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       Always returns success (NML$_STS_SUC).
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
1
2   BEGIN
2
2   $NPA_ARGDEF;                              ! Define NPARSE block reference
2 !
2 ! Save the event class code (only the low 9 bits).
2 !
2   NML$GW_EVTCLASS = .(.NPARSE_BLOCK [NPA$L_FLDPTR])<0,9>;
2
2   RETURN NML$_STS_SUC
2
1   END;                                      ! End of NML$PRM_EVTCLASS


                                 0000 00000       .ENTRY  NML$PRM_EVTCLASS, Save nothing         ; 1046
  50      14  BC         09    00  EF 00002       EXTZV   #0, #9, @20(NPARSE_BLOCK), R0          ; 1085
               00000000G 00    50  B0 00008       MOVW    R0, NML$GW_EVTCLASS                    ;
                         50    01  D0 0000F       MOVL    #1, R0                                 ; 1087
                                   04 00012       RET                                            ; 1089

; Routine Size:  19 bytes,    Routine Base:  $CODE$ + 02D9
```

```
; 1108       1090  1 %SBTTL 'NML$PRM_EVTMSKTYP  Get event mask type code'
; 1109       1091  1 GLOBAL ROUTINE NML$PRM_EVTMSKTYP =
; 1110       1092  1
; 1111       1093  1 !++
; 1112       1094  1 ! FUNCTIONAL DESCRIPTION:
; 1113       1095  1 !
; 1114       1096  1 !       This is a NPARSE action routine that saves the event mask type code.
; 1115       1097  1 !
; 1116       1098  1 ! FORMAL PARAMETERS:
; 1117       1099  1 !
; 1118       1100  1 !       NONE
; 1119       1101  1 !
; 1120       1102  1 ! IMPLICIT INPUTS:
; 1121       1103  1 !
; 1122       1104  1 !       NPARSE_BLOCK (pointed to by AP) contains the parsed parameter data.
; 1123       1105  1 !           NPA$L_FLDCNT is the parameter length.
; 1124       1106  1 !           NPA$L_FLDPTR is a pointer to the parameter in the received
; 1125       1107  1 !               message buffer.
; 1126       1108  1 !
; 1127       1109  1 ! IMPLICIT OUTPUTS:
; 1128       1110  1 !
; 1129       1111  1 !
; 1130       1112  1 ! ROUTINE VALUE:
; 1131       1113  1 ! COMPLETION CODES:
; 1132       1114  1 !
; 1133       1115  1 !       Always returns success (NML$_STS_SUC).
; 1134       1116  1 !
; 1135       1117  1 ! SIDE EFFECTS:
; 1136       1118  1 !
; 1137       1119  1 !       NONE
; 1138       1120  1 !
; 1139       1121  1 !--
; 1140       1122  1
; 1141       1123  2    BEGIN
; 1142       1124  2
; 1143       1125  2    $NPA_ARGDEF;                        ! Define NPARSE block reference
; 1144       1126  2 !
; 1145       1127  2 ! Save the event mask type code.
; 1146       1128  2 !
; 1147       1129  2    NML$GB_EVTMSKTYP = .(.NPARSE_BLOCK [NPA$L_FLDPTR])<6,2>;
; 1148       1130  2
; 1149       1131  2    RETURN NML$_STS_SUC
; 1150       1132  2
; 1151       1133  1    END;                                ! End of NML$PRM_EVTMSKTYP


                                          0000 00000      .ENTRY  NML$PRM_EVTMSKTYP, Save nothing          ; 1091
        50      14  BC          02          06  EF 00002      EXTZV   #6, #2, @20(NPARSE_BLOCK), R0            ; 1129
                       00000000G  00          50  90 00008      MOVB    R0, NML$GB_EVTMSKTYP                     ;
                                  50          01  D0 0000F      MOVL    #1, R0                                   ; 1131
                                                  04 00012      RET                                              ; 1133

; Routine Size:  19 bytes,    Routine Base:  $CODE$ + 02EC
```

```
%SBTTL 'NML$PRM_EVTMASK  Get event mask'
GLOBAL ROUTINE NML$PRM_EVTMASK =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This is a NPARSE action routine that saves the event mask.
!
! FORMAL PARAMETERS:
!
!       NONE
!
! IMPLICIT INPUTS:
!
!       NPARSE_BLOCK (pointed to by AP) contains the parsed parameter data.
!            NPA$L_FLDCNT is the parameter length.
!            NPA$L_FLDPTR is a pointer to the parameter in the received
!                message buffer.
!
! IMPLICIT OUTPUTS:
!
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       Always returns success (NML$_STS_SUC).
!
! SIDE EFFECTS:
!
!       NONE
!
!--

  BEGIN

  $NPA_ARGDEF;                          ! Define NPARSE block reference

  LOCAL
      STATUS;                           ! Return status
!
! Save the event mask descriptor information.
!
  NML$GQ_EVTMSKDSC [DSC$W_LENGTH] =
      .(.NPARSE_BLOCK [NPA$L_FLDPTR])<0,8>;

  NML$GQ_EVTMSKDSC [DSC$A_POINTER] =
      .NPARSE_BLOCK [NPA$L_FLDPTR] + 1;
!
! If the event mask is all zeroes then it is not valid.
!
  STATUS = NML$_STS_PVA;

  INCR I FROM 0 TO .NML$GQ_EVTMSKDSC [DSC$W_LENGTH] - 1 DO
      BEGIN

      IF .(.NML$GQ_EVTMSKDSC [DSC$A_POINTER] + .I)<0,8> NEQ 0
      THEN
```

```
; 1210      1191 4              BEGIN
; 1211      1192 4
; 1212      1193 4              STATUS = NML$_STS_SUC;
; 1213      1194 4              EXITLOOP;
; 1214      1195 4
; 1215      1196 3              END;
; 1216      1197 2          END;
; 1217      1198 2  !
; 1218      1199 2  ! If an error is detected then return an invalid parameter value error.
; 1219      1200 2  !
; 1220      1201 2      IF NOT .STATUS
; 1221      1202 2      THEN
; 1222      1203 2          NML$ERROR_2 (NMA$C_STS_PVA, NMA$C_PCLO_EVE);
; 1223      1204 2
; 1224      1205 2      RETURN .STATUS
; 1225      1206 2
; 1226      1207 1      END;                                    ! End of NML$PRM_EVTMASK


                                      001C 00000          .ENTRY  NML$PRM_EVTMASK, Save R2,R3,R4                  ; 1135
                  54 00000000G  00      9E 00002          MOVAB   NML$GQ_EVTMSKDSC, R4                            ;
                  64        14  BC      9B 00009          MOVZBW  @20(NPARSE_BLOCK), NML$GQ_EVTMSKDSC             ; 1177
   04    A4       14            AC  01  C1 0000D          ADDL3   #1, 20(NPARSE_BLOCK), NML$GQ_EVTMSKDSC+4        ; 1180
                  53                20  CE 00013          MNEGL   #32, STATUS                                     ; 1184
                  52                64  3C 00016          MOVZWL  NML$GQ_EVTMSKDSC, R2                            ; 1186
                  51                01  CE 00019          MNEGL   #1, I                                           ; 1189
                                    0E  11 0001C          BRB     2$                                              ;
                  50        04      A4  D0 0001E 1$:      MOVL    NML$GQ_EVTMSKDSC+4, R0                          ;
                                  6140  95 00022          TSTB    (I)[R0]                                         ;
                                    05  13 00025          BEQL    2$                                              ;
                  53                01  D0 00027          MOVL    #1, STATUS                                      ; 1193
                                    04  11 0002A          BRB     3$                                              ; 1191
         EE       51                52  F2 0002C 2$:      AOBLSS  R2, I, 1$                                       ; 1186
                  0E                53  E8 00030 3$:      BLBS    STATUS, 4$                                      ; 1201
                  7E        C9      8F  9A 00033          MOVZBL  #201, -(SP)                                     ; 1203
                  7E                10  CE 00037          MNEGL   #16, -(SP)                                      ;
         00000000G  00              02  FB 0003A          CALLS   #2, NML$ERROR_2                                 ;
                  50                53  D0 00041 4$:      MOVL    STATUS, R0                                      ; 1205
                                    04     00044          RET                                                     ; 1207

; Routine Size:  69 bytes,    Routine Base:  $CODE$ + 02FF
```

```
 1 %SBTTL 'NML$PRM_CHKEVE  Check for event parameter'
 1 GLOBAL ROUTINE NML$PRM_CHKEVE =
 1
 1 !++
 1 ! FUNCTIONAL DESCRIPTION:
 1 !
 1 !        This is a NPARSE action routine that checks for the presence
 1 !        of the event parameter.
 1 !
 1 ! FORMAL PARAMETERS:
 1 !
 1 !        NONE
 1 !
 1 ! IMPLICIT INPUTS:
 1 !
 1 !        NPARSE_BLOCK
 1 !
 1 ! IMPLICIT OUTPUTS:
 1 !
 1 !        NONE
 1 !
 1 ! ROUTINE VALUE:
 1 ! COMPLETION CODES:
 1 !
 1 !        Always returns NML$_STS_SUC.
 1 !
 1 ! SIDE EFFECTS:
 1 !
 1 !
 1 !--
 1
 2     BEGIN
 2
 2     $NPA_ARGDEF;                            ! Define NPARSE block reference
 2 !
 2 ! If this is the event filter (EFI) parameter group and no event logging
 2 ! parameter has been specified then return a parameter missing error.
 2 !
 2     IF .NML$GL_PRS_FLGS [NML$V_PRS_EFIPG]
 2     AND NOT .NML$GL_PRS_FLGS [NML$V_PRS_EVE]
 2     THEN
 2         NML$ERROR_2 (NMA$C_STS_PMS,
 2                      NMA$C_PCLO_EVE);
 2
 2     RETURN NML$_STS_SUC
 2
 1     END;                                    ! End of NML$PRM_CHKEVE


                                   0000 00000          .ENTRY  NML$PRM_CHKEVE, Save nothing        ; 1209
 16 00000000G   00           03   E1 00002          BBC     #3, NML$GL_PRS_FLGS+1, 1$           ; 1246
 0E 00000000G   00           05   E0 0000A          BBS     #5, NML$GL_PRS_FLGS+1, 1$           ; 1247
                7E      C9   8F   9A 00012          MOVZBL  #201, -(SP)                         ; 1249
                7E           1D   CE 00016          MNEGL   #29, -(SP)                          ;
```

```
            00000000G  00        02  FB 00019        CALLS   #2, NML$ERROR_2                          ;
                       50        01  D0 00020  1$:   MOVL    #1, R0                                   ; 1252
                                     04 00023        RET                                              ; 1254

; Routine Size:  36 bytes,    Routine Base:  $CODE$ + 0344
```

```
%SBTTL 'NML$PRM_SAVE_NODE        Check and save node parameter'
GLOBAL ROUTINE NML$PRM_SAVE_NODE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This is a NPARSE action that checks a node id for validity.
!       Some node parameters are saved in the DECnet databases as a node
!       address.  If the node is specified by name and no corresponding
!       address can be found in the data base then an error is returned.
!
! FORMAL PARAMETERS:
!
!       NONE
!
! IMPLICIT INPUTS:
!
!       NPARSE_BLOCK [NPA$L_FLDPTR] contains the pointer to the entity
!           format code and id string.
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       Always returns success (NML$_STS_SUC).
!
! SIDE EFFECTS:
!
!       If the host node id is invalid then an invalid identification format
!       error (NML$_STS_IDE) will be signalled.
!
!--

BEGIN

$npa_argdef;                            ! Define NPARSE block reference

OWN
    node_addr : WORD;

LOCAL
    length,
    addr;

!
! The NODE ADDRESS parameter is never specified as a name, and therefore,
! its NICE parameter type is a word.
!
IF .nparse_block [npa$l_param] EQL nml$c_node_num_param THEN
    BEGIN
    length = 0;
    addr = .nparse_block [npa$l_fldptr];
    END
ELSE
```

```
2 !
2 ! Other node parameters are saved in address form, but can be specified
2 ! by either the name or the address.  So the NICE format is node-id.
2 ! If leading byte of the parameter (the length) is a 0, the user typed a
2 ! node address rather than counted node name string.  Get the length and
2 ! address of the node parameter.
2 !
3     BEGIN
3     length = .(.nparse_block [npa$l_fldptr])<0,8>; ! Get length
3     addr = .nparse_block [npa$l_fldptr] + 1;
2     END;
2 !
2 ! If length is zero then id is a node address, otherwise it is a counted
2 ! node name string.
2 !
2 IF .length EQL 0 THEN
3     BEGIN
3     node_addr = .(.addr)<0,16>;
3     !
3     ! If the node area was specified as zero, change to to 1 (if NCP
3     ! is a Phase IV NCP) or the executor's area (if NCP is a Phase III
3     ! NCP).
3     !
3     nml$fix_node_num (node_addr);
3     END
2 ELSE
2     !
2     ! The parameter was specified as a name.  Look the node up in
2     ! the node database to get it's address.
2     !
3     BEGIN
3     IF NOT nml$getnodadr (.length, .addr, node_addr) THEN
3         nml$error_2 (nma$c_sts_ide, nma$c_ent_nod);
2     END;
2 !
2 ! Save the node address parameter.
2 !
2 nml$saveparam (.nml$gl_prmcode, 2, node_addr);
2
2 nml$gl_prmcode = 0;                         ! Reset parameter code
2 RETURN nml$_sts_suc
1 END;                                    ! End of NML$PRM_SAVE_NODE


                                            .PSECT  $OWN$,NOEXE,2

                                00000 NODE_ADDR:
                                            .BLKB   2


                                            .PSECT  $CODE$,NOWRT,2

                         000C 00000         .ENTRY  NML$PRM_SAVE_NODE, Save R2,R3               ; 1256
          53 00000000G   00  9E 00002       MOVAB   NML$GL_PRMCODE, R3                          ;
          52 00000000'   00  9E 00009       MOVAB   NODE_ADDR, R2                               ;
```

```
                              20   AC   D5 00010        TSTL    32(NPARSE_BLOCK)                   ; 1306
                                   08   12 00013        BNEQ    1$                                 ;
                                   51   D4 00015        CLRL    LENGTH                             ; 1308
                         50   14   AC   D0 00017        MOVL    20(NPARSE_BLOCK), ADDR             ; 1309
                                   09   11 0001B        BRB     2$                                 ; 1306
                         51   14   BC   9A 0001D 1$:    MOVZBL  @20(NPARSE_BLOCK), LENGTH          ; 1320
           50        14  AC        01   C1 00021        ADDL3   #1, 20(NPARSE_BLOCK), ADDR         ; 1321
                                   51   D5 00026 2$:    TSTL    LENGTH                             ; 1327
                                   0E   12 00028        BNEQ    3$                                 ;
                         62        60   B0 0002A        MOVW    (ADDR), NODE_ADDR                  ; 1329
                                   52   DD 0002D        PUSHL   R2                                 ; 1335
                00000000G  00      01   FB 0002F        CALLS   #1, NML$FIX_NODE_NUM               ;
                                   1A   11 00036        BRB     4$                                 ; 1327
                                   05   BB 00038 3$:    PUSHR   #^M<R0,R2>                         ; 1343
                                   51   DD 0003A        PUSHL   LENGTH                             ;
                00000000G  00      03   FB 0003C        CALLS   #3, NML$GETNODADR                  ;
                           0C      50   E8 00043        BLBS    R0, 4$                             ;
                                   7E   D4 00046        CLRL    -(SP)                              ; 1344
                           7E      09   CE 00048        MNEGL   #9, -(SP)                          ;
                00000000G  00      02   FB 0004B        CALLS   #2, NML$ERROR_2                    ;
                                   52   DD 00052 4$:    PUSHL   R2                                 ; 1349
                                   02   DD 00054        PUSHL   #2                                 ;
                                   63   DD 00056        PUSHL   NML$GL_PRMCODE                     ;
                00000000G  00      03   FB 00058        CALLS   #3, NML$SAVEPARAM                  ;
                                   63   D4 0005F        CLRL    NML$GL_PRMCODE                     ; 1351
                           50      01   D0 00061        MOVL    #1, R0                             ; 1352
                                        04 00064        RET                                        ; 1353

; Routine Size:  101 bytes,     Routine Base:  $CODE$ + 0368
```

```
1 %SBTTL 'NML$PRM_SET_NET  Set X25-Protocol network ID'
1 GLOBAL ROUTINE NML$PRM_SET_NET =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !       This is a NPARSE action routine that is called only when NML
1 !       gets a NICE command of SET X25-PROTOCOL NET <net-id>.  In this
1 !       case only, the entity ID passed to the ACP must be the network
1 !       name.  In all other cases, the network ID is a zero length string
1 !       to indicate the currently active network.  SET X25-PROTOCOL NET
1 !       is used to specify the active network in the first place.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NPARSE_BLOCK (pointed to by AP) contains the parsed parameter data.
1 !               NPA$L_FLDCNT is the parameter length.
1 !               NPA$L_FLDPTR is a pointer to the parameter in the received
1 !                       message buffer.
1 !       NML$GL_PRMCODE contains the index into the change parameter table
1 !               (NML$AB_CPTABLE).
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !       Always returns NML$_STS_SUC.
1 !
1 !--
1
2 BEGIN
2
2 MAP
2     NML$GB_OPTIONS : BBLOCK [1];
2
2 !
2 ! The entity ID has already been set up to be the active network
2 ! (NML$GB_ENTITY_FORMAT = 0) during initial parsing.  So, if the NICE
2 ! command is DEFINE, leave things as they are.  If it's a SET, pop
2 ! the network name off the parameter stack and save it as the entity ID
2 ! for the QIO P2 buffer.
2 !
2 IF NOT .NML$GB_OPTIONS [NMA$V_OPT_PER] THEN
3     BEGIN
3     !
3     ! Back up to the network parameter.  This pops the network parameter
3     ! off the parameter stack.
3     !
3     NML$GW_PRMDESCNT = .NML$GW_PRMDESCNT - 1;
3     !
3     ! Move the network ID to the entity ID globals.
3     !
3     NML$GB_ENTITY_FORMAT = .NML$AW_PRM_DES [.NML$GW_PRMDESCNT, PDB$W_COUNT];
3     CH$MOVE (.NML$GB_ENTITY_FORMAT,
3             .NML$AW_PRM_DES [.NML$GW_PRMDESCNT, PDB$A_POINTER],
3             NML$AB_ENTITY_ID);
2     END;
2
2 RETURN NML$_STS_SUC
2
```

```
; 1433          1411  1 END;                                  ! End of NML$PRM_SET_NET


                                      00FC 00000          .ENTRY  NML$PRM_SET_NET, Save R2,R3,R4,R5,R6,R7      ; 1355
                    57 00000000G   00   9E 00002          MOVAB   NML$GB_ENTITY_FORMAT, R7                     ;
                    56 00000000G   00   9E 00009          MOVAB   NML$GW_PRMDESCNT, R6                         ;
                       00000000G   00   95 00010          TSTB    NML$GB_OPTIONS                               ; 1393
                                   24   19 00016          BLSS    1$                                           ;
                                   66   B7 00018          DECW    NML$GW_PRMDESCNT                             ; 1399
                    50             66   3C 0001A          MOVZWL  NML$GW_PRMDESCNT, R0                         ; 1403
                       00000000G0040    7F 0001D          PUSHAQ  NML$AW_PRM_DES+2[R0]                         ;
                    67             9E   90 00024          MOVB    @(SP)+, NML$GB_ENTITY_FORMAT                 ;
                    51             67   9A 00027          MOVZBL  NML$GB_ENTITY_FORMAT, R1                     ; 1404
                       00000000G0040    7F 0002A          PUSHAQ  NML$AW_PRM_DES+4[R0]                         ; 1405
                    50             9E   D0 00031          MOVL    @(SP)+, R0                                   ;
       00000000G  00       60      51   28 00034          MOVC3   R1, (R0), NML$AB_ENTITY_ID                   ; 1404
                    50             01   D0 0003C 1$:      MOVL    #1, R0                                       ; 1409
                                        04 0003F          RET                                                  ; 1411

; Routine Size:  64 bytes,    Routine Base:  $CODE$ + 03CD
```

```
1 %SBTTL 'NML$PRM_QUAL_FORMAT  Check X25-Protocol Group DTE format'
1 GLOBAL ROUTINE NML$PRM_QUAL_FORMAT =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !       This is a NPARSE action routine that is called only when NML
1 !       gets a NICE SET/DEFINE X25 PROTOCOL GROUP command.
1 !       In this case some SET/DEFINE commands are illegal, because
1 !       with groups, you must always set DTE, number, and type
1 !       together.  Also, the same DTE will have different group numbers in
1 !       different groups but DTEs in the same group must have the same group
1 !       number.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NPARSE_BLOCK (pointed to by AP) contains the parsed parameter data.
1 !           NPA$L_FLDCNT is the parameter length.
1 !           NPA$L_FLDPTR is a pointer to the parameter in the received
1 !               message buffer.
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !--
1
2 BEGIN
2
2 MAP
2     NML$GB_OPTIONS : BBLOCK [1];
2 !
2 ! If KNOWN GROUPS allow only
2 !       SET KNOWN GROUPS KNOWN DTES ALL or
2 !       SET KNOWN GROUPS ALL
2 !
2 IF .NML$GB_ENTITY_FORMAT LEQ 0 THEN               ! If KNOWN groups and
3     BEGIN
4     IF (.NML$GB_OPTIONS [NMA$V_OPT_PER] OR        ! not SET ALL
3        NOT .NML$GL_PRS_FLGS [NML$V_PRS_ALL]) OR !        or
4        (.NML$GL_PRS_FLGS [NML$V_PRS_QUALIFIER] AND ! DTE nnnn
3         .NML$GB_QUALIFIER_FORMAT GTR 0) THEN
3         RETURN NML$_STS_IDE;                      !         Error.
3     END
2 ELSE
2     !
2     ! If a specific group, allow only
2     !   SET/DEFINE GROUP FRED DTE 333444555666 NUM 3  ... or
2     !   SET/DEFINE GROUP FRED ALL
2     !   SET/DEFINE GROUP FRED KNOWN DTES ALL
2     !
3     BEGIN
3     IF .NML$GB_ENTITY_FORMAT GTR 0 THEN
4         BEGIN
4         !
4         ! SET/DEFINE X25-PROTOCOL GROUP FRED KNOWN DTES ALL - is illegal.
4         !
4         IF .NML$GL_PRS_FLGS [NML$V_PRS_ALL] AND
5             (.NML$GL_PRS_FLGS [NML$V_PRS_QUALIFIER] AND
```

```
; 1492     1469 4              .NML$GB_QUALIFIER_FORMAT LEQ 0) THEN
; 1493     1470 4                   RETURN NML$_STS_IDE;
; 1494     1471 3           END;
; 1495     1472 2       END;
; 1496     1473 2
; 1497     1474 2 RETURN NML$_STS_SUC;
; 1498     1475 2
; 1499     1476 1 END;                                    ! End of NML$PRM_QUAL_FORMAT


                                    000C 00000          .ENTRY  NML$PRM_QUAL_FORMAT, Save R2,R3             ; 1413
                     53 00000000G  00  9E 00002          MOVAB   NML$GB_QUALIFIER_FORMAT, R3                 ;
                     52 00000000G  00  9E 00009          MOVAB   NML$GL_PRS_FLGS, R2                         ;
                     50 00000000G  00  9A 00010          MOVZBL  NML$GB_ENTITY_FORMAT, R0                    ; 1446
                                   16  14 00017          BGTR    1$                                          ;
                        00000000G  00  95 00019          TSTB    NML$GB_OPTIONS                              ; 1448
                                   1A  19 0001F          BLSS    2$                                          ;
              16            62     01  E1 00021          BBC     #1, NML$GL_PRS_FLGS, 2$                     ; 1449
              16            62     02  E1 00025          BBC     #2, NML$GL_PRS_FLGS, 3$                     ; 1450
                                   63  95 00029          TSTB    NML$GB_QUALIFIER_FORMAT                     ; 1451
                                   12  13 0002B          BEQL    3$                                          ;
                                   0C  11 0002D          BRB     2$                                          ; 1452
              0C            62     01  E1 0002F 1$:      BBC     #1, NML$GL_PRS_FLGS, 3$                     ; 1467
              08            62     02  E1 00033          BBC     #2, NML$GL_PRS_FLGS, 3$                     ; 1468
                                   63  95 00037          TSTB    NML$GB_QUALIFIER_FORMAT                     ; 1469
                                   04  12 00039          BNEQ    3$                                          ;
                            50     12  CE 0003B 2$:      MNEGL   #18, R0                                     ; 1470
                                       04 0003E          RET                                                 ;
                            50     01  D0 0003F 3$:      MOVL    #1, R0                                      ; 1474
                                       04 00042          RET                                                 ; 1476

; Routine Size:  67 bytes,    Routine Base:  $CODE$ + 040D
```

```
1 %SBTTL 'NML$PRM_CHANNELS  Check parameter value'
1 GLOBAL ROUTINE NML$PRM_CHANNELS =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 ! The X-25 Protocol DTE parameter, channels, is passed by NCP in the form of
1 ! parameter id, channel pair, parameter id, channel pair...
1 ! It must be converted to a string.  Accumulate the string here.
1 !
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !          NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !          NPARSE_BLOCK (pointed to by AP) contains the parsed parameter data.
1 !               NPA$L_FLDCNT is the parameter length.
1 !               NPA$L_FLDPTR is a pointer to the parameter in the received
1 !                   message buffer.
1 !          NML$GL_PRMCODE contains the index into the change parameter table
1 !               (NML$AB_CPTABLE).
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !          If the parameter is valid then a descriptor entry will be created for
1 !          it in NML$AW_PRM_DES and NML$GW_PRMDESCNT will be incremented.
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !          Always returns NML$_STS_SUC.
1 !
1 ! SIDE EFFECTS:
1 !
1 !--
1
2 BEGIN
2
2 $NPA_ARGDEF;                         ! Define NPARSE block reference
2
2 LITERAL CHAN_STR_SIZE = 80;
2
2 OWN
2     INDEX,               ! Parameter stack index for channels entry.
2     CHANNEL_STRING: BBLOCK [CHAN_STR_SIZE];
2
2 LOCAL
2     COUNT;
2
2 IF NOT .NML$GL_PRS_FLGS [NML$V_PRS_CHANNELS] THEN
2          !
2          ! This is the first channel range in the NICE SET command.  Set
2          ! up the parameter descriptor list to accumulate all the channel
2          ! ranges in a string which is how it must be passed to the PSI
2          ! ACP.
```

```
2           !
3           BEGIN
3           NML$GL_PRS_FLGS [NML$V_PRS_CHANNELS] = 1;
3           !
3           ! Save the index into the parameter descriptor stack in order to add
3           ! the length of later channel pairs.  Then add the descriptor entry
3           ! for channels to the stack.
3           !
3           INDEX = .NML$GW_PRMDESCNT;
3           NML$SAVEPARAM (.NML$GL_PRMCODE, 0, CHANNEL_STRING);
2           END;
2
2       !
2       ! Make sure the number of channel pairs in the NICE message isn't overflowing
2       ! the internal channel string buffer.
2       !
2       COUNT = .NML$AW_PRM_DES [.INDEX, PDB$W_COUNT];
2       IF .COUNT GEQ CHAN_STR_SIZE THEN
2           NML$ERROR_2 (NMA$C_STS_PLO,
2                        NMA$C_PCXP_CHN);
2
2       !
2       ! Move the channel pair to the end of the string of channel pairs.  Then
2       ! increment the string length in the parameter descriptor stack.
2       !
2       CHANNEL_STRING [.COUNT,0,32,0] = ..NPARSE_BLOCK [NPA$L_FLDPTR];
2       NML$AW_PRM_DES [.INDEX, PDB$W_COUNT] = .COUNT + .NPARSE_BLOCK [NPA$L_FLDCNT];
2
2       NML$GL_PRMCODE = 0;                     ! Reset parsing code
2
2       RETURN NML$_STS_SUC
2
1       END;                                    ! End of NML$PRM_CHANNELS


                                                        .PSECT  $OWN$,NOEXE,2

                                        00002           .BLKB   2
                                        00004 INDEX:    .BLKB   4
                                        00008 CHANNEL_STRING:
                                                        .BLKB   80


                                                        .PSECT  $CODE$,NOWRT,2

                                003C 00000              .ENTRY  NML$PRM_CHANNELS, Save R2,R3,R4,R5      ; 1478
                 55 00000000G    00  9E 00002           MOVAB   NML$AW_PRM_DES+2, R5                    ;
                 54 00000000G    00  9E 00009           MOVAB   NML$GL_PRMCODE, R4                      ;
                 53 00000000'    00  9E 00010           MOVAB   INDEX, R3                               ;
                 1C 00000000G    00  E8 00017           BLBS    NML$GL_PRS_FLGS+1, 1$                   ; 1528
        00000000G 00             01  88 0001E           BISB2   #1, NML$GL_PRS_FLGS+1                   ; 1536
                 63 00000000G    00  3C 00025           MOVZWL  NML$GW_PRMDESCNT, INDEX                 ; 1542
                          04     A3  9F 0002C           PUSHAB  CHANNEL_STRING                          ; 1543
                                 7E  D4 0002F           CLRL    -(SP)                                   ;
                                 64  DD 00031           PUSHL   NML$GL_PRMCODE                          ;
```

```
                   00000000G  00            03  FB 00033       CALLS   #3, NML$SAVEPARAM                              ;
                              50            63  D0 0003A 1$:   MOVL    INDEX, R0                                      ; 1550
                                          6540  7F 0003D       PUSHAQ  NML$AW_PRM_DES+2[R0]                           ;
                              52            9E  3C 00040       MOVZWL  @(SP)+, COUNT                                  ;
                   00000050   8F            52  D1 00043       CMPL    COUNT, #80                                     ; 1551
                                            0F  19 0004A       BLSS    2$                                             ;
                              7E     046A   8F  3C 0004C       MOVZWL  #1130, -(SP)                                   ; 1552
                              7E            17  CE 00051       MNEGL   #23, -(SP)                                     ;
                   00000000G  00            02  FB 00054       CALLS   #2, NML$ERROR_2                                ;
                                     04   A342  9F 0005B 2$:   PUSHAB  CHANNEL_STRING[COUNT]                          ; 1559
                              9E     14     BC  D0 0005F       MOVL    @20(NPARSE_BLOCK), @(SP)+                      ;
                              50            63  D0 00063       MOVL    INDEX, R0                                      ; 1560
                                          6540  7F 00066       PUSHAQ  NML$AW_PRM_DES+2[R0]                           ;
                  9E          52     10     AC  A1 00069       ADDW3   16(NPARSE_BLOCK), COUNT, @(SP)+                ;
                                            64  D4 0006E       CLRL    NML$GL_PRMCODE                                 ; 1562
                              50            01  D0 00070       MOVL    #1, R0                                         ; 1564
                                                04 00073       RET                                                    ; 1566

; Routine Size:  116 bytes,    Routine Base:  $CODE$ + 0450


; 1591          1567  1
; 1592          1568  1
; 1593          1569  1
; 1594          1570  1 END                                         ! End of module
; 1595          1571  1
; 1596          1572  0 ELUDOM




;                                   PSECT SUMMARY
;
;       Name                        Bytes                           Attributes
;
;  $CODE$                            1220  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $PLIT$                               3  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $OWN$                               88  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)




;                            Library Statistics
;
;                                        -------- Symbols --------      Pages      Processing
;       File                             Total    Loaded   Percent      Mapped     Time
;
;  _$255$DUA28:[NML.OBJ]NMLLIB.L32;1       341        39        11         27      00:00.1
;  _$255$DUA28:[SHRLIB]NMALIBRY.L32;1      887        17         1         47      00:00.2
;  _$255$DUA28:[SYSLIB]STARLET.L32;1      9776         2         0        581      00:02.2
```

```
;                              COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:NMLPARPRM/OBJ=OBJ$:NMLPARPRM MSRC$:NMLPARPRM/UPDATE=(ENH$:NMLPARPRM)

; Size:           1220 code + 91 data bytes
; Run Time:          00:27.1
; Elapsed Time:      01:05.5
; Lines/CPU Min:        3476
; Lexemes/CPU-Min: 11938
; Memory Used:  103 pages
; Compilation Complete
```

0285 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

NMLPURGE
LIS

NMLPARINI
LIS

NMLNODFIL
LIS

NMLREAD
LIS

NMLPARPRM
LIS

NMLPMANIP
LIS