```
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**NPARSE

```
NN      NN  PPPPPPPP     AAAAAA    RRRRRRRR      SSSSSSSS  EEEEEEEEEE
NN      NN  PPPPPPPP     AAAAAA    RRRRRRRR      SSSSSSSS  EEEEEEEEEE
NN      NN  PP      PP  AA    AA  RR      RR  SS            EE
NN      NN  PP      PP  AA    AA  RR      RR  SS            EE
NNNN    NN  PP      PP  AA    AA  RR      RR  SS            EE
NNNN    NN  PP      PP  AA    AA  RR      RR  SS            EE
NN  NN  NN  PPPPPPPP    AA    AA  RRRRRRRR      SSSSSS      EEEEEEEE
NN  NN  NN  PPPPPPPP    AA    AA  RRRRRRRR      SSSSSS      EEEEEEEE
NN    NNNN  PP          AAAAAAAAAA  RR  RR              SS  EE
NN    NNNN  PP          AAAAAAAAAA  RR  RR              SS  EE
NN      NN  PP          AA    AA  RR    RR            SS  EE          ....
NN      NN  PP          AA    AA  RR    RR            SS  EE          ....
NN      NN  PP          AA    AA  RR      RR  SSSSSSSS    EEEEEEEEEE  ....
NN      NN  PP          AA    AA  RR      RR  SSSSSSSS    EEEEEEEEEE  ....

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
```

Table of contents

```
0000     1          .TITLE  NPARSE  NETWORK (BINARY) MESSAGE PARSER
0000     2          .IDENT  'V04-000'
0000     3  ;
0000     4  ;*****************************************************************************
0000     5  ;*                                                                           *
0000     6  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8  ;*  ALL RIGHTS RESERVED.                                                     *
0000     9  ;*                                                                           *
0000    10  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    11  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    12  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    13  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    14  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    15  ;*  TRANSFERRED.                                                             *
0000    16  ;*                                                                           *
0000    17  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    18  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    19  ;*  CORPORATION.                                                             *
0000    20  ;*                                                                           *
0000    21  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    22  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23  ;*                                                                           *
0000    24  ;*                                                                           *
0000    25  ;*****************************************************************************
0000    26  ;
0000    27
0000    28  ;++
0000    29  ; FACILITY:  TABLE DRIVEN NETWORK (BINARY) MESSAGE PARSER
0000    30  ;
0000    31  ;
0000    32  ; ABSTRACT:
0000    33  ;
0000    34  ;       NPARSE is a binary parsing facility similar to the table-driven
0000    35  ;       ASCII parser, TPARSE.  State tables provide the information necessary
0000    36  ;       to validate and process binary data messages such as those of the
0000    37  ;       DECnet-VAX NICE protocol.
0000    38  ;
0000    39  ;
0000    40  ; ENVIRONMENT:  VAX/VMX Operating System
0000    41  ;
0000    42  ; AUTHOR: Scott Davidson        , CREATION DATE: 25-Sep-1979
0000    43  ;
0000    44  ;         Adapted from NPARS facility for DECnet-11M.
0000    45  ;
0000    46  ; MODIFIED BY:
0000    47  ;
0000    48  ;       , : VERSION
0000    49  ;       V03-001 MKP0001         Kathy Perko             19-Jan-1982
0000    50  ;               Add psects which are compatible with BLISS ones so that
0000    51  ;               NML doesn't use so many image sections.
0000    52  ;
0000    53  ;       V03-001 MKP0001         Kathy Perko             25-Oct-1981
0000    54  ;               Fix copyright and add documentation about NPARSE tables
0000    55  ;               which this module parses.
0000    56  ;--
```

```
0000     58 ;          NPARSE - Table Driven Parser for the NICE Protocol
0000     59 ;
0000     60 ;NPARSE is a VMS adaptation of the RSX NPARS facility.  NPARSE is a stripped
0000     61 ;down version of the standard table driven parser, TPARSE, (see Chapter 7
0000     62 ;of the VAX-11 Common Run-Time Procedure Library Reference Manual manual). The
0000     63 ;parse tables are set up in a similar manner using the following macros:
0000     64 ;
0000     65 ;1. Initialising the state table
0000     66 ;
0000     67 ;       The IMSG$ macro is used to initialise the state table which describes
0000     68 ;the format of the message to be parsed. The state table is placed in the psect
0000     69 ;'NPA$STATE'.  The IMSG$ macro takes a single argument which is the starting
0000     70 ;label of the state table. The macro format is:
0000     71 ;
0000     72 ;       IMSG$   statetable
0000     73 ;
0000     74 ;
0000     75 ;2. Defining a message field
0000     76 ;
0000     77 ;       The FIELD$ macro declares the beginning of field within the message.
0000     78 ;Syntactically, this macro delimits one message field from another. The
0000     79 ;FIELD$ macro is coded as follows:
0000     80 ;
0000     81 ;
0000     82 ;       FIELD$  [label]
0000     83 ;
0000     84 ;where label is an alphanumeric symbol that is equated to the address of the
0000     85 ;state.
0000     86 ;
0000     87 ;
0000     88 ;3. Defining a field specification
0000     89 ;
0000     90 ;       The general format for defining a field specification is as follows:
0000     91 ;
0000     92 ;       $Macro  [argument][,label    ][,action][,mask,mskadr][,param]
0000     93 ;                         [,NPA$_EXIT]
0000     94 ;                         [,NPA$_FAIL]
0000     95 ;
0000     96 ;where label, action, mask, mskadr and param have the same meanings as
0000     97 ;for TPARSE.  The following macros are used to define field specifications:
0000     98 ;
0000     99 ;Macro  Argument
0000    100 ;
0000    101 ;$IMAGE size           defines an image field with a maximum length of 'size'
0000    102 ;
0000    103 ;$WORD  number         matches a word (2 bytes) from the message.
0000    104 ;
0000    105 ;$BYTE  value          matches a single byte from the message.
0000    106 ;
0000    107 ;$LOOK  [value]        Examine a byte but do not advance the pointers.
0000    108 ;                     Optionally, examine a byte with the specified value.
0000    109 ;
0000    110 ;$EOM                  matches the end of the message.
0000    111 ;
0000    112 ;$NULL                 this transition is always successful (equivalent to
0000    113 ;                     the TPARSE TPA$_LAMDA transition).
0000    114 ;
```

```
0000    115 ;$SBEXP state           recursively calls NPARSE to parse a subexpression
0000    116 ;                       from the message.
0000    117 ;
0000    118 ;$ERROR code            forces a failure from NPARSE at the top level and loads
0000    119 ;                       the error code into R0.  All error codes should
0000    120 ;                       be even numbers suitable for testing with BLBC or BLBS.
0000    121 ;
0000    122 ;$EXTZV fld             matches a value of a specified field of bits within
0000    123 ;                       a byte and optionally advance the pointers.  The
0000    124 ;                       'fld' argument has the following form:
0000    125 ;
0000    126 ;                               <value,position,size[,NPA$_ADVANCE]>.
0000    127 ;
0000    128 ;                               where:
0000    129 ;
0000    130 ;                               value = field value to match
0000    131 ;                               position = bit position within the byte
0000    132 ;                               size = field width (0<size<9)
0000    133 ;
0000    134 ;                       The field value (not the entire byte) is stored in
0000    135 ;                       NPA$B_BYTE.
0000    136 ;
0000    137 ;$MASK  bitmask          matches a byte with any of the specified bits set.
0000    138 ;
0000    139 ;$MATCH number           matches a sequence of bytes, 'number' in length.
0000    140 ;
0000    141 ;
0000    142 ;
0000    143 ;                                 Calling Conventions
0000    144 ;
0000    145 ;NPARSE is invoked in the following manner:
0000    146 ;
0000    147 ;       CALLS   #2,NMA$NPARSE
0000    148 ;
0000    149 ;When NPARSE is invoked, the following arguments must be set up:
0000    150 ;
0000    151 ;       P1 - pointer to NPARSE argument block
0000    152 ;       P2 - label of the starting state in the NPARSE table
0000    153 ;
0000    154 ;The NPARSE argument block has the following format:
0000    155 ;
0000    156 ;
0000    157 ;                       +---------+---------+---------+---------+
0000    158 ;NPA$L_COUNT             !                 NPA$K_COUNT0          !
0000    159 ;                       +---------+---------+---------+---------+
0000    160 ;NPA$L_MSGCNT           !       bytes remaining to be parsed    !
0000    161 ;                       +---------+---------+---------+---------+
0000    162 ;NPA$L_MSGPTR           !       pointer to unparsed message     !
0000    163 ;                       +---------+---------+---------+---------+
0000    164 ;NPA$L_OPTIONS          !                 *reserved*            !
0000    165 ;                       +---------+---------+---------+---------+
0000    166 ;NPA$L_FLDCNT           !       count of matched field          !
0000    167 ;                       +---------+---------+---------+---------+
0000    168 ;NPA$L_FLDPTR           !       pointer to matched field        !
0000    169 ;NPA$B_BYTE             +---------+---------+---------+---------+
0000    170 ;NPA$W_WORD             !                 unsigned value        !
0000    171 ;NPA$L_LONG             +---------+---------+---------+---------+
```

```
0000    172 ;NPA$L_NUMBER   !                  signed value                 !
0000    173 ;               +---------+---------+---------+---------+
0000    174 ;NPA$L_PARAM    !       action routine parameter               !
0000    175 ;               +---------+---------+---------+---------+
0000    176 ;
0000    177 ;
0000    178 ;On return from NPARSE, R0 contains the status code.
0000    179 ;
0000    180 ;Action routines are called with the argument pointer (AP) pointing to
0000    181 ;the NPARSE argument block.  Transitions may be rejected by returning R0
0000    182 ;with the low bit clear.
0000    183 ;
0000    184 ;
0000    185 ;Example of a state table to parse a message consisting of:
0000    186 ;
0000    187 ;       1. Image-8 field
0000    188 ;       2. Parameter field (parameters 100, 101 and 102 are allowed)
0000    189 ;          The parameters may be repeated any number of times.
0000    190 ;
0000    191 ;       IMSG$   MSGTAB
0000    192 ;
0000    193 ;       FIELD$
0000    194 ;       $IMAGE  8
0000    195 ;
0000    196 ;       FIELD$  PARM
0000    197 ;       $SBEXP  PARAM,PARM
0000    198 ;       $EOM    ,NPA$_EXIT
0000    199 ;       $ERROR  NML$_STS_FOR
0000    200 ;
0000    201 ;       FIELD$  PARAM
0000    202 ;       $PARAM  100,NPA$_EXIT,,1,PARNUM
0000    203 ;       $PARAM  101,NPA$_EXIT,,2,PARNUM
0000    204 ;       $PARAM  102,NPA$_EXIT,,4,PARNUM
0000    205 ;
0000    206 ;       FIELD$
0000    207 ;
```

```
                     0000   209          .SBTTL  Declarations
                     0000   210
                     0000   211 ;
                     0000   212 ; INCLUDE FILES:
                     0000   213 ;
                     0000   214
                     0000   215 $NPADEF          ; Define NPARSE argument block offsets
                     0000   216 NPADF$           ; Define NPARSE function and matching codes
                     0000   217
                     0000   218 ;
                     0000   219 ; MACROS:
                     0000   220 ;
                     0000   221
                     0000   222 ;
                     0000   223 ; EQUATED SYMBOLS:
                     0000   224 ;
                     0000   225
                     0000   226          .SBTTL  ****DEBUG****
                     0000   227
00000001             0000   228          $$DEBUG$$       = 1
                     0000   229
                00000000    230          .PSECT  $OWN$,NOEXE,RD,WRT
                     0000   231 ;
                     0000   232 ; OWN STORAGE:
                     0000   233 ;
                     0000   234
                     0000   235 NPA$GL_LOGMASK::
00000000             0000   236          .LONG   0               ; Logging mask (1=yes)
                     0004   237
```

```
                     0004   239          .SBTTL  NMA$NPARSE - MAIN ROUTINE BODY
                 00000000   240          .PSECT  $CODE$,EXE,RD,NOWRT
                     0000   241 ;++
                     0000   242 ; FUNCTIONAL DESCRIPTION:
                     0000   243 ;
                     0000   244 ;
                     0000   245 ; FORMAL PARAMETERS:
                     0000   246 ;
                     0000   247 ;       P1 - Pointer to NPARSE argument block
                     0000   248 ;       P2 - Initial state pointer
                     0000   249 ;
                     0000   250 ; IMPLICIT INPUTS:
                     0000   251 ;
                     0000   252 ;       NPARSE argument block must contain message length (NPA$L_MSGCNT)
                     0000   253 ;       and message pointer (NPA$L_MSGPTR).
                     0000   254 ;
                     0000   255 ; IMPLICIT OUTPUTS:
                     0000   256 ;
                     0000   257 ;       NPARSE argument block contains length of unparsed message (NPA$L_MSGCNT)
                     0000   258 ;       and pointer to unparsed part of message (NPA$L_MSGPTR).
                     0000   259 ;
                     0000   260 ; ROUTINE VALUE:
                     0000   261 ; COMPLETION CODES:
                     0000   262 ;
                     0000   263 ;       R0 - Success or error status code
                     0000   264 ;
                     0000   265 ; SIDE EFFECTS:
                     0000   266 ;
                     0000   267 ;       NONE
                     0000   268 ;
                     0000   269 ;--
                     0000   270
                 07FC 0000   271          .ENTRY  NMA$NPARSE,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10>
          5A 04 AC    D0 0002   272          MOVL    4(AP),R10                       ; Get argument block pointer
          52 04 AA    D0 0006   273          MOVL    NPA$L_MSGCNT(R10),R2            ; Get length argument
          53 08 AA    D0 000A   274          MOVL    NPA$L_MSGPTR(R10),R3            ; Get buffer pointer argument
          55 08 AC    D0 000E   275          MOVL    8(AP),R5                        ; Get initial state argument
                         0012   276 .IF     DEFINED $$DEBUG$$
             04 AA    DD 0012   277          PUSHL   NPA$L_MSGCNT(R10)               ; Store initial message count
   0000009A'EF    DF 0015   278          PUSHAL  OUT_INIT                        ; Store pointer to FAO control string
000002D5'EF    02 FB 001B   279          CALLS   #2,NPA$$DEBUG_OUT               ; Call debug output routine
                         0022   280 .ENDC
0000003E'EF    00 FB 0022   281          CALLS   #0,NPA$$SUB_EXPR                ; Begin message processing
                         0029   282 .IF     DEFINED $$DEBUG$$
             01    BB 0029   283          PUSHR   #^M<R0>                         ; Save register
             04 AA    DD 002B   284          PUSHL   NPA$L_MSGCNT(R10)               ; Store remaining message length
   000000C6'EF    DF 002E   285          PUSHAL  OUT_LEFT                        ; Store pointer to FAO control string
000002D5'EF    02 FB 0034   286          CALLS   #2,NPA$$DEBUG_OUT               ; Call debug output routine
             01    BA 003B   287          POPR    #^M<R0>                         ; Restore register
                         003D   288 .ENDC
                     04 003D   289          RET
```

```
                                003E    291          .SBTTL  NPA$$SUB_EXPR - SUBEXPRESSION ENTRY POINT
                                003E    292 ;
                                003E    293 ; Subexpression entry point
                                003E    294 ;
                           0040 003E    295          .ENTRY  NPA$$SUB_EXPR,^M<R6>    ;
                                0040    296 ;
                                0040    297 ; Enter new state specified in R5
                                0040    298 ;
FFFFFFFF 8F    55    D1         0040    299 MSTAT:   CMPL    R5,#NPA$_FAIL           ; Fail the parse ?
                     2A    13   0047    300          BEQL    SYNERR                  ; If EQL, yes
                  10 AA    D4   0049    301          CLRL    NPA$L_FLDCNT(R10)       ; Clear any old field
                  18 AA    D4   004C    302          CLRL    NPA$L_LONG(R10)         ;  ...
                  1C AA    D4   004F    303          CLRL    NPA$L_NUMBER(R10)       ;  ...
               14 AA       53   D0   0052    304          MOVL    R3,NPA$L_FLDPTR(R10)    ; Initialize field pointer
                  56       85   3C   0056    305          MOVZWL  (R5)+,R6        ; Get type
                                0059    306          CASE    R6,<-                   ; Dispatch
                                0059    307                  MATCH_IMAGE,-           ; 00 - image field
                                0059    308                  MATCH_WORD,-            ; 01 - word (unsigned)
                                0059    309                  MATCH_MASK,-            ; 02 - match masked byte value
                                0059    310                  MATCH_BYTE,-            ; 03 - byte (unsigned)
                                0059    311                  MATCH_EOM,-             ; 04 - match end of message
                                0059    312                  MATCH_ALL,-             ; 05 - always match
                                0059    313                  MATCH_SBEXP,-           ; 06 - match a subexpression
                                0059    314                  MATCH_ERROR,-           ; 07 - exit from parse with error
                                0059    315                  MATCH_STRING,-          ; 08 - match byte string
                                0059    316                  EXAMINE_BYTE,-          ; 09 - examine a byte
                                0059    317                  MATCH_EXTZV-            ; 10 - match extracted byte value
                                0059    318                  >                       ;
                                0073    319 SYNERR:
                  50    01    CA   0073    320          BICL    #1,R0                   ; Indicate parse failure
                                0076    321 .IF      DEFINED $$DEBUG$$
                        01    BB   0076    322          PUSHR   #^M<R0>                 ; Save register
                        50    DD   0078    323          PUSHL   R0                      ; Store exit status value
                00000088'EF   DF   007A    324          PUSHAL  OUT_FAIL                ; Store pointer to FAO control string
           000002D5'EF        02   FB   0080    325          CALLS   #2,NPA$$DEBUG_OUT       ; Call debug output routine
                        01    BA   0087    326          POPR    #^M<R0>                 ; Restore register
                                0089    327 .ENDC
                        16    11   0089    328          BRB     EXIT                    ;
                                008B    329 SUCCESS:
                                008B    330 .IF      DEFINED $$DEBUG$$
                00000108'EF   DF   008B    331          PUSHAL  OUT_SUCC                ; Store output string address
                00000007'8F   DD   0091    332          PUSHL   #OUT_SUCC_LEN           ;   and length
           00000311'EF        02   FB   0097    333          CALLS   #2,NPA$$DEBUG_OUT2      ; Call debug output routine
                                009E    334 .ENDC
                  50    01    C8   009E    335          BISL    #1,R0                   ; Indicate successful parse
                                00A1    336 EXIT:
                        04    00A1    337          RET                             ;
```

```
                                00A2   339          .SBTTL  MATCH_IMAGE - MATCH ON IMAGE MODE FIELD
                                00A2   340 ;+
                                00A2   341 ; **-MATCH_IMAGE-MATCH ON IMAGE MODE FIELD
                                00A2   342 ;
                                00A2   343 ;       AN IMAGE MODE FIELD CONSISTS OF A SINGLE BYTE GIVING THE LENGTH OF
                                00A2   344 ;       THE FIELD FOLLOWED BY THAT NUMBER OF BYTES.
                                00A2   345 ;-
                                00A2   346 ; INPUTS:
                                00A2   347 ;       R2 - # OF BYTES REMAINING TO BE SCANNED
                                00A2   348 ;       R3 - POINTER TO UNSCANNED PART OF MESSAGE
                                00A2   349 ;       R5 - POINTER TO ?
                                00A2   350 ;
                                00A2   351 MATCH_IMAGE:
                          52    D5 00A2   352          TSTL    R2                      ; Any bytes left in message ?
                          26    15 00A4   353          BLEQ    10$                     ; If LEQ, count is not present
                       56 63    9A 00A6   354          MOVZBL  (R3),R6                 ; Get length of image field
                    02 A5 56    D1 00A9   355          CMPL    R6,2(R5)                ; Is it within specified range ?
                          1D    1A 00AD   356          BGTRU   10$                     ; If GTRU, no
                          56    D6 00AF   357          INCL    R6                      ; Take into account the count byte
                       52 56    D1 00B1   358          CMPL    R6,R2                   ; Enough characters in message ?
                          16    1A 00B4   359          BGTRU   10$                     ; If GTRU, no
                    10 AA 56    D0 00B6   360          MOVL    R6,NPA$L_FLDCNT(R10)    ; Set size of matched field
                                00BA   361 .IF      DEFINED $$DEBUG$$
                          53    DD 00BA   362          PUSHL   R3                      ; Store pointer to image field
                00000000'EF     DF 00BC   363          PUSHAL  OUT_IMAGE               ; Store pointer to FAO control string
            000002D5'EF   02    FB 00C2   364          CALLS   #2,NPA$$DEBUG_OUT       ; Call debug output routine
                                00C9   365 .ENDC
                        01A2    31 00C9   366          BRW     UPDTRN                  ; Update pointers and do transition
                        0150    31 00CC   367 10$:     BRW     NXTRAN                  ; Reject the transition
```

```
                     00CF    369          .SBTTL  MATCH_WORD - MATCH ON PARAMETER VALUE IN MESSAGE
                     00CF    370 ;+
                     00CF    371 ; **-MATCH_WORD-MATCH ON PARAMETER VALUE IN MESSAGE
                     00CF    372 ;
                     00CF    373 ;       A word value occupies two consecutive message bytes.
                     00CF    374 ;       Match the specified word value (unsigned).
                     00CF    375 ;-
                     00CF    376 ; INPUTS:
                     00CF    377 ;       R2 - # OF BYTES REMAINING TO BE SCANNED
                     00CF    378 ;       R3 - POINTER TO UNSCANNED PART OF MESSAGE
                     00CF    379 ;
                     00CF    380 MATCH_WORD:
          02   52    D1  00CF    381          CMPL    R2,#2                   ; Two bytes left in message ?
               22    19  00D2    382          BLSS    10$                     ; If LSS, no
       18 AA   63    3C  00D4    383          MOVZWL  (R3),NPA$W_WORD(R10)    ; Get word value for action routines
 18 AA    02 A5    B1  00D8    384          CMPW    2(R5),NPA$W_WORD(R10)   ; Does it match ?
               17    12  00DD    385          BNEQU   10$                     ; If NEQU, no
       10 AA   02    D0  00DF    386          MOVL    #2,NPA$L_FLDCNT(R10)    ; Set length of matched field
                     00E3    387 .IF     DEFINED $$DEBUG$$
            18 AA    DD  00E3    388          PUSHL   NPA$W_WORD(R10)         ; Store word value
    00000011'EF      DF  00E6    389          PUSHAL  OUT_WORD                ; Store pointer to FAO control string
000002D5'EF      02    FB  00EC    390          CALLS   #2,NPA$$DEBUG_OUT       ; Call debug output routine
                     00F3    391 .ENDC
              0178   31  00F3    392          BRW     UPDTRN                  ; Update pointers and do transition
              0126   31  00F6    393 10$:     BRW     NXTRAN                  ; Reject the transition
```

```
                     00F9   395          .SBTTL  MATCH_MASK - MATCH ON MASKED BYTE VALUE
                     00F9   396 ;+
                     00F9   397 ; **-MATCH_MASK-MATCH ON BYTE VALUE IN MESSAGE
                     00F9   398 ;
                     00F9   399 ;       MATCH ON THE NEXT AVAILABLE BYTE IN THE MESSAGE.
                     00F9   400 ;-
                     00F9   401
                     00F9   402 MATCH_MASK:
           52   D5   00F9   403          TSTL    R2                          ; Any bytes left in message ?
           22   15   00FB   404          BLEQ    20$                         ; If LEQ,  no
     18 AA   63   9A   00FD   405 10$:    MOVZBL  (R3),NPA$B_BYTE(R10)        ; Get next byte from message
18 AA   02 A5   93   0101   406          BITB    2(R5),NPA$B_BYTE(R10)       ; Any mask bits set ?
           17   13   0106   407          BEQLU   20$                         ; If EQLU, no
     10 AA   01   D0   0108   408          MOVL    #1,NPA$L_FLDCNT(R10)        ; Set length of matched field
                     010C   409 .IF       DEFINED $$DEBUG$$
        18 AA   DD   010C   410          PUSHL   NPA$B_BYTE(R10)             ; Store byte value
  00000055'EF   DF   010F   411          PUSHAL  OUT_MATCH                   ; Store pointer to FAO control string
000002D5'EF   02   FB   0115   412          CALLS   #2,NPA$$DEBUG_OUT           ; Call debug output routine
                     011C   413 .ENDC
         014F   31   011C   414          BRW     UPDTRN                      ; Update pointers and do transition
         00FD   31   011F   415 20$:     BRW     NXTRAN                      ; Reject the transition
```

```
                                0122   417          .SBTTL  EXAMINE_BYTE - LOOK AT A BYTE WITHOUT ADVANCING POINTERS
                                0122   418
                                0122   419 ;+
                                0122   420 ; **-EXAMINE_BYTE-LOOK AT A BYTE WITHOUT ADVANCING MESSAGE POINTERS
                                0122   421 ;-
                                0122   422 ;
                                0122   423 EXAMINE_BYTE:
                          52    D5  0122   424          TSTL    R2                          ; Any bytes left in message ?
                          27    15  0124   425          BLEQ    20$                         ; If LEQ, no
                       18 AA    63  9A 0126   426          MOVZBL  (R3),NPA$B_BYTE(R10)        ; Get next byte
                          65    01  B3 012A   427          BITW    #NPA$M_EXT,(R5)             ; Extension present ?
                          0B    13  012D   428          BEQLU   10$                         ; If EQLU, no
                    18 AA   02    A5  91 012F   429          CMPB    2(R5),NPA$B_BYTE(R10)       ; Do they match ?
                          17    12  0134   430          BNEQU   20$                         ; If NEQU, no
                       10 AA    01    D0  0136   431          MOVL    #1,NPA$L_FLDCNT(R10)        ; Set length of matched field
                                0139   432 10$:
                                0139   433 .IF     DEFINED $$DEBUG$$
                             18 AA    DD  0139   434          PUSHL   NPA$B_BYTE(R10)             ; Store byte value
                   00000033'EF    DF  013C   435          PUSHAL  OUT_LOOK                    ; Store pointer to FAO control string
              000002D5'EF    02    FB  0142   436          CALLS   #2,NPA$$DEBUG_OUT           ; Call debug output routine
                                0149   437 .ENDC
                         0131   31  0149   438          BRW     DOTRAN                      ; Do the transition
                         00CF   31  014C   439 20$:      BRW     NXTRAN                      ; Reject the transition
```

```
                     0150   441          .SBTTL  MATCH_EXTZV - LOOK AT EXTRACTED FIELD VALUE
                     0150   442 ;+
                     0150   443 ; **-EXAMINE_BYTE-LOOK AT BYTE WITH SPECIFIED FIELD
                     0150   444 ;-
                     0150   445 ;
                     0150   446 MATCH_EXTZV:
            52   D5  0150   447          TSTL    R2                              ; Any bytes left in message ?
            38   15  0152   448          BLEQ    20$                             ; If LEQ, no
     18 AA  02 A5  9A  0154   449          MOVZBL  2(R5),NPA$B_BYTE(R10)           ; Get field value
        56  03 A5  9A  0159   450          MOVZBL  3(R5),R6                        ; Get bit position
        57  04 A5  9A  015D   451          MOVZBL  4(R5),R7                        ; Get field size (width)
        58     63  9A  0161   452          MOVZBL  (R3),R8                         ; Get next byte in message
18 AA   58   57   56  ED  0164   453          CMPZV   R6,R7,R8,NPA$B_BYTE(R10) ; Match ?
            20   12  016A   454          BNEQU   20$                             ; If NEQU, no
     10 AA  01   D0  016C   455          MOVL    #1,NPA$L_FLDCNT(R10)            ; Set field count
                     0170   456 .IF       DEFINED $$DEBUG$$
           02 A5   DD  0170   457          PUSHL   2(R5)                           ; Store field information
   00000044'EF   DF  0173   458          PUSHAL  OUT_EXTZV                       ; Store pointer to FAO control string
000002D5'EF   02   FB  0179   459          CALLS   #2,NPA$$DEBUG_OUT               ; Call debug output routine
                     0180   460 .ENDC
        05 A5  01   93  0180   461          BITB    #NPA$_ADVANCE,5(R5)             ; Advance pointers ?
               03   13  0184   462          BEQL    10$                             ; If EQL, no
             00E5   31  0186   463          BRW     UPDTRN                          ; Update pointers and do transition
             00F2   31  0189   464 10$:      BRW     DOTRAN                          ; Perform the transition
             0090   31  018C   465 20$:      BRW     NXTRAN                          ; Reject the transition
```

```
                     018F   467          .SBTTL  MATCH_BYTE - MATCH ON SPECIFIED BYTE VALUE
                     018F   468 ;+
                     018F   469 ; **-MATCH_BYTE-MATCH ON SPECIFIED BYTE VALUE
                     018F   470 ;
                     018F   471 ;       MATCH ON THE NEXT BYTE IN THE MESSAGE IF IT HAS A SPECIFIED VALUE.
                     018F   472 ;-
                     018F   473 ; INPUTS:
                     018F   474 ;       R2 - # OF BYTES REMAINING TO BE SCANNED
                     018F   475 ;       R3 - POINTER TO UNSCANNED PART OF MESSAGE
                     018F   476 ;
                     018F   477 MATCH_BYTE:
           52   D5   018F   478          TSTL    R2                          ; Any bytes left in message ?
           22   15   0191   479          BLEQ    10$                         ; If LEQ, no
     18 AA   63   9A   0193   480          MOVZBL  (R3),NPA$B_BYTE(R10)        ; Get next byte from message
18 AA   02 A5   91   0197   481          CMPB    2(R5),NPA$B_BYTE(R10)       ; Does it match specified byte ?
           17   12   019C   482          BNEQU   10$                         ; If NEQU, no
     10 AA   01   D0   019E   483          MOVL    #1,NPA$L_FLDCNT(R10)        ; Set length of matched field
                     01A2   484 .IF       DEFINED $$DEBUG$$
        18 AA   DD   01A2   485          PUSHL   NPA$B_BYTE(R10)             ; Store parameter value
00000022'EF   DF   01A5   486          PUSHAL  OUT_BYTE                    ; Store pointer to FAO control string
000002D5'EF   02   FB   01AB   487          CALLS   #2,NPA$$DEBUG_OUT           ; Call debug output routine
                     01B2   488 .ENDC
           00B9   31   01B2   489          BRW     UPDTRN                      ; Update pointers and do transition
             68   11   01B5   490 10$:     BRB     NXTRAN                      ; Reject the transition
```

```
                     01B7   492          .SBTTL  MATCH_SBEXP - MATCH ON SUBEXPRESSION
                     01B7   493 ;+
                     01B7   494 ; **-MATCH_SBEXP-MATCH ON SUBEXPRESSION
                     01B7   495 ;
                     01B7   496 ;       CALL THE PARSER RECURSIVELY TO MATCH ON A SPECIFIED
                     01B7   497 ;       SUBEXPRESSION.
                     01B7   498 ;-
                     01B7   499 ; INPUTS:
                     01B7   500 ;       R2 - # OF BYTES REMAINING TO BE SCANNED
                     01B7   501 ;       R3 - POINTER TO UNSCANNED PART OF MESSAGE
                     01B7   502 ;       R5 - POINTER TO CURRENT STATE
                     01B7   503 ;
                     01B7   504 MATCH_SBEXP:
           53    DD  01B7   505          PUSHL   R3                      ; Save registers
           55    DD  01B9   506          PUSHL   R5                      ;  ...
     55    02 A5    DO  01BB   507          MOVL    2(R5),R5                ; Get starting state for subexpression
                     01BF   508 .IF       DEFINED $$DEBUG$$
     00000103'EF    DF  01BF   509          PUSHAL  OUT_SBEXP               ; Store pointer to output string
     00000005'8F    DD  01C5   510          PUSHL   #OUT_SBEXP_LEN          ;   and length
00000311'EF    02    FB  01CB   511          CALLS   #2,NPA$$DEBUG_OUT2      ; Call debug output routine
                     01D2   512 .ENDC
    FE67 CF    00    FB  01D2   513          CALLS   #0,NPA$$SUB_EXPR        ; Try to parse subexpression
           20    BA  01D7   514          POPR    #^M<R5>                 ; Restore state pointer
     14 AA    6E    DO  01D9   515          MOVL    (SP),NPA$L_FLDPTR(R10)  ; Set pointer to matched subexpression
 10 AA    53    8E    C3  01DD   516          SUBL3   (SP)+,R3,NPA$L_FLDCNT(R10) ; Length of matched subexpression
           3A 50    E9  01E2   517          BLBC    R0,NXTRAN               ; If LBC, reject the transition
            0096    31  01E5   518          BRW     DOTRAN                  ; Perform the transition
```

```
                     01E8   520          .SBTTL  MATCH_STRING - MATCH ON SEQUENCE OF BYTES
                     01E8   521 ;+
                     01E8   522 ; **-MATCH_STRING-MATCH ON SEQUENCE OF BYTES
                     01E8   523 ;
                     01E8   524 ;       MATCH ON A SPECIFIED NUMBER OF BYTES IN THE MESSAGE.
                     01E8   525 ;-
                     01E8   526 ; INPUTS:
                     01E8   527 ;       R2 - # OF BYTES REMAINING TO BE SCANNED
                     01E8   528 ;       R3 - POINTER TO UNSCANNED PART OF MESSAGE
                     01E8   529 ;       R5 - POINTER TO FLAG BYTE OF CURRENT STATE
                     01E8   530 ;
                     01E8   531 MATCH_STRING:
          56   02 A5   D0   01E8   532          MOVL    2(R5),R6                ; Get number of bytes to match
               52   56   D1   01EC   533          CMPL    R6,R2                   ; Enough characters left in message ?
                    2E   14   01EF   534          BGTR    NXTRAN                  ; If GTR, No - try next transition
          10 AA    56   D0   01F1   535          MOVL    R6,NPA$L_FLDCNT(R10)    ; Set up length of field
                     01F5   536 .IF     DEFINED $$DEBUG$$
                    56   DD   01F5   537          PUSHL   R6                      ; Store byte string pointer
        00000055'EF   DF   01F7   538          PUSHAL  OUT_MATCH               ; Store pointer to FAO control string
    000002D5'EF   02   FB   01FD   539          CALLS   #2,NPA$$DEBUG_OUT       ; Call debug output routine
                     0204   540 .ENDC
                    68   11   0204   541          BRB     UPDTRN                  ; Update pointers and perform transition
```

```
                     0206   543          .SBTTL  MATCH_EOM - CHECK FOR END OF MESSAGE
                     0206   544 ;+
                     0206   545 ; **-MATCH_EOM-CHECK FOR END OF MESSAGE
                     0206   546 ;
                     0206   547 ;       CHECK THAT WE HAVE PARSED UP TO THE END OF THE MESSAGE
                     0206   548 ;-
                     0206   549 ; INPUTS:
                     0206   550 ;       R2 - # OF BYTES REMAINING TO BE PARSED
                     0206   551 ;       R3 - POINTER TO UNSCANNED PART OF MESSAGE
                     0206   552 ;
                     0206   553 MATCH_EOM:
                  52 D5  0206   554          TSTL    R2                      ; Any bytes remaining to be parsed ?
                  15 12  0208   555          BNEQ    NXTRAN                  ; If NEQ, yes
                         020A   556 .IF       DEFINED $$DEBUG$$
      000000F1'EF  DF  020A   557          PUSHAL  OUT_EOM                 ; Store output string address
      0000000E'8F  DD  0210   558          PUSHL   #OUT_EOM_LEN            ;   and length
00000311'EF  02  FB  0216   559          CALLS   #2,NPA$$DEBUG_OUT2      ; Call debug output routine
                         021D   560 .ENDC
                  5F 11  021D   561          BRB     DOTRAN                  ;
```

```
                     021F   563          .SBTTL  NXTRAN - TRY NEXT ENTRY IN TRANSITION TABLE
                     021F   564 ;+
                     021F   565 ; **-NXTRAN-TRY NEXT ENTRY IN TRANSITION TABLE
                     021F   566 ;
                     021F   567 ;       THE CURRENT TRANSITION HAS FAILED, TRY TO MATCH ON THE NEXT
                     021F   568 ;       ENTRY IN THE TRANSITION TABLE.
                     021F   569 ;-
                     021F   570 ; INPUTS:
                     021F   571 ;       R2 - # OF BYTES REMAINING UNSCANNED
                     021F   572 ;       R3 - POINTER TO UNSCANNED PORTION OF MESSAGE
                     021F   573 ;       R5 - POINTER TO FLAG BYTE OF TRANSITION
                     021F   574 ;
     52   10 AA   C0  021F   575 NXTRAN: ADDL2   NPA$L_FLDCNT(R10),R2     ; Return any field matched
     53   10 AA   C2  0223   576         SUBL2   NPA$L_FLDCNT(R10),R3     ;  for re-parsing
        56   85   32  0227   577         CVTWL   (R5)+,R6                 ; Get flags
             03   18  022A   578         BGEQ    10$                      ; If GEQ, no
           FE44   31  022C   579         BRW     SYNERR                   ; Last transition in state
        57   06   D0  022F   580 10$:     MOVL    #6,R7                    ; Count of flag bits to test
          02 56   E9  0232   581 20$:     BLBC    R6,30$                   ; If LBC, entry not present
             85   D5  0235   582         TSTL    (R5)+                    ; Skip entry in table
56   56    FF 8F   9C  0237   583 30$:     ROTL    #-1,R6,R6                ; Check next flag bit
          F3 57   F5  023C   584         SOBGTR  R7,20$                   ; Loop
            FDFE   31  023F   585         BRW     MSTAT                    ;   and try next transition
```

```
                     0242   587          .SBTTL  MATCH_ERROR - LEAVE PARSER WITH ERROR
                     0242   588 ;+
                     0242   589 ; **-MATCH_ERROR-LEAVE PARSER WITH ERROR
                     0242   590 ;
                     0242   591 ;       EXIT FROM THE PARSER AT THE TOP LEVEL WITH AN ERROR CODE.
                     0242   592 ;-
                     0242   593 ; INPUTS:
                     0242   594 ;       R2 - # OF BYTES REMAINING TO BE SCANNED
                     0242   595 ;       R3 - POINTER TO UNSCANNED PART OF MESSAGE
                     0242   596 ;       R5 - POINTER TO CURRENT STATE
                     0242   597 ;
                     0242   598 ; OUTPUTS:
                     0242   599 ;       R0 - ERROR CODE FROM STATE TABLE
                     0242   600 ;
                     0242   601 MATCH_ERROR:
          50   02 A5   D0   0242   602          MOVL    2(R5),R0                ; Pick up error code
                     0246   603 .IF     DEFINED $$DEBUG$$
                01   BB   0246   604          PUSHR   #^M<R0>                 ; Save register
                50   DD   0248   605          PUSHL   R0                      ; Store error value
      00000077'EF   DF   024A   606          PUSHAL  OUT_ERROR               ; Store pointer to FAO control string
 000002D5'EF   02   FB   0250   607          CALLS   #2,NPA$$DEBUG_OUT       ; Call debug output routine
                01   BA   0257   608          POPR    #^M<R0>                 ; Restore register
                     0259   609 .ENDC
                23   11   0259   610          BRB     DOTRAN                  ; Perform the transition
```

```
                     025B   612          .SBTTL  MATCH_ALL - PERFORM NULL STATE TRANSITION
                     025B   613          .SBTTL  DOTRAN - PERFORM THE STATE TRANSITION
                     025B   614          .SBTTL  UPDTRN - UPDATE COUNT AND POINTERS
                     025B   615 ;+
                     025B   616 ; **-UPDTRN-UPDATE COUNT AND POINTERS
                     025B   617 ;
                     025B   618 ;       UPDATE THE COUNT AND POINTER FOR THE CURRENT MATCHING TRANSITION
                     025B   619 ;       BEFORE PERFORMING THE TRANSITION.
                     025B   620 ;
                     025B   621 ; **-DOTRAN-PERFORM THE STATE TRANSITION
                     025B   622 ;
                     025B   623 ;       A SUCCESSFUL MATCH HAS OCCURRED ON A SUB-FIELD IN THE MESSAGE,
                     025B   624 ;       PERFORM THE SPECIFIED STATE TRANSITION.
                     025B   625 ;-
                     025B   626 ; INPUTS:
                     025B   627 ;       R2 - # OF CHARACTERS REMAINING TO BE SCANNED
                     025B   628 ;       R3 - POINTER TO MESSAGE REMAINING TO BE SCANNED
                     025B   629 ;       R5 - POINTER TO FLAG WORD IN CURRENT TRANSITION
                     025B   630 ;
                     025B   631 MATCH_ALL:
                     025B   632 .IF      DEFINED $$DEBUG$$
      000000FF'EF    DF     025B   633          PUSHAL  OUT_NULL                ; Store string pointer
      00000004'8F    DD     0261   634          PUSHL   #OUT_NULL_LEN           ; Store string length
00000311'EF    02     FB     0267   635          CALLS   #2,NPA$$DEBUG_OUT2      ; Call debug output routine
                     026E   636 .ENDC
          52    10 AA   C2     026E   637 UPDTRN: SUBL2   NPA$L_FLDCNT(R10),R2    ; Reduce count of bytes remaining
          04 AA   52     D0     0272   638          MOVL    R2,NPA$L_MSGCNT(R10)    ; ...
          53    10 AA   C0     0276   639          ADDL2   NPA$L_FLDCNT(R10),R3    ; Update message pointer
          08 AA   53     D0     027A   640          MOVL    R3,NPA$L_MSGPTR(R10)    ; ...
                  55     DD     027E   641 DOTRAN: PUSHL   R5                      ; Save current state table address
             56   85     32     0280   642          CVTWL   (R5)+,R6                ; Get flags
             56   01     D3     0283   643          BITL    #NPA$M_EXT,R6           ; Extension present ?
                  02     13     0286   644          BEQLU   10$                     ; If EQLU, no
                  85     D5     0288   645          TSTL    (R5)+                   ; Skip extension field in entry
            20 AA   D4     028A   646 10$:    CLRL    NPA$L_PARAM(R10)        ; Zero action routine parameter
          56    02     D3     028D   647          BITL    #NPA$M_PARAM,R6         ; Action routine parameter present ?
                04     13     0290   648          BEQLU   20$                     ; If EQLU, no
       20 AA    85     D0     0292   649          MOVL    (R5)+,NPA$L_PARAM(R10)  ; Get action routine parameter
          56    04     D3     0296   650 20$:    BITL    #NPA$M_ACTION,R6        ; Action routine address present ?
                1A     13     0299   651          BEQLU   40$                     ; If EQLU, no
             0041 8F     BB     029B   652          PUSHR   #^M<R0,R6>              ; Save action flags and status
             50   01     D0     029F   653          MOVL    #1,R0                   ; Set default return status to success
             95   6A     FA     02A2   654          CALLG   (R10),@(R5)+            ; Call action routine
                09   50   E8     02A5   655          BLBS    R0,30$                  ; If LBS, then success
             0041 8F     BA     02A8   656          POPR    #^M<R0,R6>              ; Restore registers
                  20     BA     02AC   657          POPR    #^M<R5>                 ; ...
                FF6E     31     02AE   658          BRW     NXTRAN                  ; Reject the transition
             0041 8F     BA     02B1   659 30$:    POPR    #^M<R0,R6>              ; Restore flags and status
                  8E     D5     02B5   660 40$:    TSTL    (SP)+                   ; Clean up stack
             56   10     D3     02B7   661          BITL    #NPA$M_MASK,R6          ; Mask present ?
                  09     13     02BA   662          BEQLU   50$                     ; If EQLU, no
             57   85     D0     02BC   663          MOVL    (R5)+,R7                ; Get bit mask to be set
             58   85     D0     02BF   664          MOVL    (R5)+,R8                ;  and address
             68   57     C8     02C2   665          BISL    R7,(R8)                 ; Set the mask bits
             56   08     D3     02C5   666 50$:    BITL    #NPA$M_STATE,R6         ; Explicit transition present ?
                  08     13     02C8   667          BEQLU   60$                     ; If EQLU, no
             55   85     D0     02CA   668          MOVL    (R5)+,R5                ; Get pointer to next state
```

```
                03   12  02CD   669          BNEQ    60$                     ; If EQL, then stop
            FDB9   31   02CF   670          BRW     SUCCESS                 ; Successful parse
            FD6B   31   02D2   671 60$:     BRW     MSTAT                   ;
```

```
                                02D5   673 .IF     DEFINED $$DEBUG$$
                                02D5   674          .SBTTL  NPARSE DEBUGGING DATA
                                02D5   675
                                02D5   676          .SAVE
                            00000000   677          .PSECT  NPA$DEBUG$PURE,NOEXE,NOWRT,BYTE
                                0000   678
                                0000   679 OUT_IMAGE:
20 65 67 61 6D 49 00000008'010E0000'   0000   680          .ASCID  "Image !AC"
                          43 41 21     000E
                                0011   681 OUT_WORD:
20 20 64 72 6F 57 00000019'010E0000'   0011   682          .ASCID  "Word  !XW"
                          57 58 21     001F
                                0022   683 OUT_BYTE:
20 20 65 74 79 42 0000002A'010E0000'   0022   684          .ASCID  "Byte  !XB"
                          42 58 21     0030
                                0033   685 OUT_LOOK:
20 20 6B 6F 6F 4C 0000003B'010E0000'   0033   686          .ASCID  "Look  !XB"
                          42 58 21     0041
                                0044   687 OUT_EXTZV:
20 76 7A 74 78 45 0000004C'010E0000'   0044   688          .ASCID  "Extzv !XL"
                          4C 58 21     0052
                                0055   689 OUT_MATCH:
20 68 63 74 61 4D 0000005D'010E0000'   0055   690          .ASCID  "Match !XB"
                          42 58 21     0063
                                0066   691 OUT_MASK:
20 20 6B 73 61 4D 0000006E'010E0000'   0066   692          .ASCID  "Mask  !XB"
                          42 58 21     0074
                                0077   693 OUT_ERROR:
20 72 6F 72 72 45 0000007F'010E0000'   0077   694          .ASCID  "Error !XL"
                          4C 58 21     0085
                                0088   695 OUT_FAIL:
64 65 6C 69 61 46 00000090'010E0000'   0088   696          .ASCID  "Failed !XL"
                       4C 58 21 20     0096
                                009A   697 OUT_INIT:
6E 69 73 72 61 50 000000A2'010E0000'   009A   698          .ASCID  "Parsing message (Length = !UL bytes)"
4C 28 20 65 67 61 73 73 65 6D 20 67    00A8
20 4C 55 21 20 3D 20 68 74 67 6E 65    00B4
                  29 73 65 74 79 62    00C0
                                00C6   699 OUT_LEFT:
73 72 61 70 6E 55 000000CE'010E0000'   00C6   700          .ASCID  "Unparsed message length = !UL bytes"
6C 20 65 67 61 73 73 65 6D 20 64 65    00D4
20 4C 55 21 20 3D 20 68 74 67 6E 65    00E0
                     73 65 74 79 62    00EC
                                00F1   701 OUT_EOM:
61 73 73 65 6D 20 66 6F 20 64 6E 45    00F1   702          .ASCII  "End of message"
                                65 67  00FD
                          0000000E     00FF   703 OUT_EOM_LEN     = .-OUT_EOM
                                00FF   704 OUT_NULL:
                             6C 6C 75 4E 00FF   705          .ASCII  "Null"
                          00000004     0103   706 OUT_NULL_LEN    = .-OUT_NULL
                                0103   707
                                0103   708 OUT_SBEXP:
                          70 78 65 62 53 0103   709          .ASCII  "Sbexp"
                          00000005     0108   710 OUT_SBEXP_LEN   = .-OUT_SBEXP
                                0108   711 OUT_SUCC:
                       73 73 65 63 63 75 53 0108   712          .ASCII  "Success"
                          00000007     010F   713 OUT_SUCC_LEN    = .-OUT_SUCC
```

```
                    010F   714
                00000000   715          .PSECT  NPA$DEBUG$IMP,NOEXE,WRT,BYTE
                    0000   716 ;
                    0000   717 ; FAO Output Buffer
                    0000   718 ;
                    0000   719 FAODESC:
00000050        0000   720          .LONG   80
00000008'       0004   721          .ADDRESS FAOBUF
                    0008   722 FAOBUF:
00000058        0008   723          .BLKB   80
                    0058   724 OUT_DSC:
0000005A        0058   725 FAOLEN:  .BLKW   1
    0000        005A   726          .WORD   0
00000000'       005C   727          .ADDRESS 0
                    0060   728
                000002D5   729          .RESTORE
```

```
                                02D5   731          .SBTTL  NPA$$DEBUG_OUT
                                02D5   732          .SBTTL  NPA$$DEBUG_OUT2
                                02D5   733
                                02D5   734 ;
                                02D5   735 ; Entry point for argument/string output
                                02D5   736 ;
                          0000  02D5   737          .ENTRY  NPA$$DEBUG_OUT,^M<>
                00000000'EF   D5  02D7   738          TSTL    NPA$GL_LOGMASK          ; Flag set if logging enabled
                           31   13  02DD   739          BEQL    10$                     ; Logging is disabled
                                02DF   740          $FAO_S  CTRSTR=@4(AP),-         ;
                                02DF   741                  OUTLEN=FAOLEN,-         ;
                                02DF   742                  OUTBUF=FAODESC,-        ;
                                02DF   743                  P1=8(AP)                ;
0000005C'EF     00000008'EF   DE  02F8   744          MOVAL   FAOBUF,OUT_DSC+4        ;
                00000058'EF   DF  0303   745          PUSHAL  OUT_DSC                 ;
             00000000'GF      01  FB  0309   746          CALLS   #1,G^LIB$PUT_OUTPUT     ; Print string
                              04  0310   747 10$:       RET                             ;
                                0311   748 ;
                                0311   749 ; Entry point for string output
                                0311   750 ;
                          0000  0311   751          .ENTRY  NPA$$DEBUG_OUT2,^M<>
                00000000'EF   D5  0313   752          TSTL    NPA$GL_LOGMASK          ; Flag set if logging disabled
                           1D   13  0319   753          BEQL    10$                     ; Logging disabled
          00000058'EF     04 AC   D0  031B   754          MOVL    4(AP),OUT_DSC           ;
          0000005C'EF     08 AC   D0  0323   755          MOVL    8(AP),OUT_DSC+4         ;
                00000058'EF   DF  032B   756          PUSHAL  OUT_DSC                 ;
             00000000'GF      01  FB  0331   757          CALLS   #1,G^LIB$PUT_OUTPUT     ; Print the string
                              04  0338   758 10$:       RET                             ;
                                0339   759 .ENDC
                                0339   760
                                0339   761          .END
```

```
$$DEBUG$$       =  00000001
$$T2            =  00000004
DOTRAN             0000027E  R      03
EXAMINE_BYTE       00000122  R      03
EXIT               000000A1  R      03
FAOBUF             00000008  R      05
FAODESC            00000000  R      05
FAOLEN             00000058  R      05
LIB$PUT_OUTPUT     ********    X    03
MATCH_ACL          0000025B  R      03
MATCH_BYTE         0000018F  R      03
MATCH_EOM          00000206  R      03
MATCH_ERROR        00000242  R      03
MATCH_EXTZV        00000150  R      03
MATCH_IMAGE        000000A2  R      03
MATCH_MASK         000000F9  R      03
MATCH_SBEXP        000001B7  R      03
MATCH_STRING       000001E8  R      03
MATCH_WORD         000000CF  R      03
MSTAT              00000040  R      03
NMA$NPARSE         00000000  RG     03
NPA$$DEBUG_OUT     000002D5  RG     03
NPA$$DEBUG_OUT2    00000311  RG     03
NPA$$SUB_EXPR      0000003E  RG     03
NPA$B_BYTE         00000018
NPA$C_LENGTH0      00000024
NPA$G_LOGMASK      00000000  RG     02
NPA$K_LENGTH0      00000024
NPA$L_COUNT        00000000
NPA$L_FLDCNT       00000010
NPA$L_FLDPTR       00000014
NPA$L_LONG         00000018
NPA$L_MSGCNT       00000004
NPA$L_MSGPTR       00000008
NPA$L_NUMBER       0000001C
NPA$L_OPTIONS      0000000C
NPA$L_PARAM        00000020
NPA$M_ACTION    =  00000004
NPA$M_EXT       =  00000001
NPA$M_LAST      =  00008000
NPA$M_MASK      =  00000010
NPA$M_MSKADR    =  00000020
NPA$M_OFFSET    =  00000040
NPA$M_PARAM     =  00000002
NPA$M_STATE     =  00000008
NPA$W_WORD         00000018
NPA$_ADVANCE    =  00000001
NPA$_BYTE       =  00000003
NPA$_EOM        =  00000004
NPA$_ERROR      =  00000007
NPA$_EXIT       =  00000000
NPA$_EXTZV      =  0000000A
NPA$_FAIL       =  FFFFFFFF
NPA$_IMAGE      =  00000000
NPA$_LOOK       =  00000009
NPA$_MASK       =  00000002
NPA$_MATCH      =  00000008
NPA$_NULL       =  00000005
NPA$_SBEXP      =  00000006
NPA$_WORD       =  00000001
NXTRAN             0000021F  R      03
OUT_BYTE           00000022  R      04
OUT_DSC            00000058  R      05
OUT_EOM            000000F1  R      04
OUT_EOM_LEN     =  0000000E
OUT_ERROR          00000077  R      04
OUT_EXTZV          00000044  R      04
OUT_FAIL           00000088  R      04
OUT_IMAGE          00000000  R      04
OUT_INIT           0000009A  R      04
OUT_LEFT           000000C6  R      04
OUT_LOOK           00000033  R      04
OUT_MASK           00000066  R      04
OUT_MATCH          00000055  R      04
OUT_NULL           000000FF  R      04
OUT_NULL_LEN    =  00000004
OUT_SBEXP          00000103  R      04
OUT_SBEXP_LEN   =  00000005
OUT_SUCC           00000108  R      04
OUT_SUCC_LEN    =  00000007
OUT_WORD           00000011  R      04
SUCCESS            0000008B  R      03
SYNERR             00000073  R      03
SYS$FAO            ********    X    03
UPDTRN             0000026E  R      03
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000024 (36.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
OWN	00000004 (4.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
$CODE$	00000339 (825.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE
NPA$DEBUG$PURE	0000010F (271.)	04 (4.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC BYTE
NPA$DEBUG$IMP	00000060 (96.)	05 (5.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.09	00:00:01.06
Command processing	121	00:00:00.76	00:00:04.49
Pass 1	181	00:00:02.93	00:00:10.87
Symbol table sort	0	00:00:00.09	00:00:00.25
Pass 2	145	00:00:01.69	00:00:03.90
Symbol table output	8	00:00:00.08	00:00:00.09
Psect synopsis output	2	00:00:00.03	00:00:00.07
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	488	00:00:05.69	00:00:20.74

The working set limit was 1500 pages.
15720 bytes (31 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 86 non-local and 22 local symbols.
761 source lines were read in Pass 1, producing 33 object records in Pass 2.
13 pages of virtual memory were used to define 11 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SHRLIB]NMALIBRY.MLB;1	0
_255DUA28:[SYS.OBJ]LIB.MLB;1	1
_255DUA28:[NML.OBJ]NMLLIB.MLB;1	2
_255DUA28:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	8

136 GETS were required to define 8 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:NPARSE/OBJ=OBJ$:NPARSE MSRC$:NPARSE/UPDATE=(ENH$:NPARSE)+LIB$:NMLLIB/LIB+EXECML$/LIB+SHRLIB$:NMALIBRY/LIB
```

NPARSE
LIS

REQUEST
MAP

OPCOM

REPLY
MAP

NMLXFER
LIS

OPCCRASH
MAP

NMLV2STA
LIS

OPCOMLIB
B32

ALLOCATE
LIS

NMLZERO
LIS

OPCOMDEF
SDL

OPCOM
MAP

OPCDEFTMP
SDL

CANCEL
LIS