```
RRRRRRRRRRR      UUU          UUU  NNN         NNN     000000000     FFFFFFFFFFFFFFF   FFFFFFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN         NNN     000000000     FFFFFFFFFFFFFFF   FFFFFFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN         NNN     000000000     FFFFFFFFFFFFFFF   FFFFFFFFFFFFFFF
RRR          RRR UUU          UUU  NNN         NNN  000          000 FFF               FFF
RRR          RRR UUU          UUU  NNN         NNN  000          000 FFF               FFF
RRR          RRR UUU          UUU  NNN         NNN  000          000 FFF               FFF
RRR          RRR UUU          UUU  NNNNNN      NNN  000          000 FFF               FFF
RRR          RRR UUU          UUU  NNNNNN      NNN  000          000 FFF               FFF
RRR          RRR UUU          UUU  NNNNNN      NNN  000          000 FFF               FFF
RRRRRRRRRRR      UUU          UUU  NNN   NNN   NNN  000          000 FFFFFFFFFFF       FFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN   NNN   NNN  000          000 FFFFFFFFFFF       FFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN   NNN   NNN  000          000 FFFFFFFFFFF       FFFFFFFFFFF
RRR   RRR        UUU          UUU  NNN       NNNNNN 000          000 FFF               FFF
RRR   RRR        UUU          UUU  NNN       NNNNNN 000          000 FFF               FFF
RRR   RRR        UUU          UUU  NNN       NNNNNN 000          000 FFF               FFF
RRR      RRR     UUU          UUU  NNN          NNN 000          000 FFF               FFF
RRR      RRR     UUU          UUU  NNN          NNN 000          000 FFF               FFF
RRR      RRR     UUU          UUU  NNN          NNN 000          000 FFF               FFF
RRR         RRR  UUUUUUUUUUUUUUU  NNN          NNN    000000000     FFF               FFF
RRR         RRR  UUUUUUUUUUUUUUU  NNN          NNN    000000000     FFF               FFF
RRR         RRR  UUUUUUUUUUUUUUU  NNN          NNN    000000000     FFF               FFF
RRR         RRR  UUUUUUUUUUUUUUU  NNN          NNN    000000000     FFF               FFF
RRR         RRR  UUUUUUUUUUUUUUU  NNN          NNN    000000000     FFF               FFF
```

```
_$2

Sym
---
NDX
NDX
NUM
NUM
OPE
OUT
PAC
PAC
PAC
PAC
PAC
PAC
PAC
PAD
PAG
PAG
PAG
PAG
PAG
PAG
PAG
PER
PUT
RCO
RIN

RLI
RNO
RNO
RTY
SAV
STR
STR
STR
STR
STR
STR

STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
```

```
PPPPPPPP      000000      000000    LL
PPPPPPPP      000000      000000    LL
PP      PP  00      00  00      00  LL
PP      PP  00      00  00      00  LL
PP      PP  00      00  00      00  LL
PP      PP  00      00  00      00  LL
PPPPPPPP    00      00  00      00  LL
PPPPPPPP    00      00  00      00  LL
PP          00      00  00      00  LL
PP          00      00  00      00  LL
PP          00      00  00      00  LL                ....
PP          00      00  00      00  LL                ....
PP            000000      000000    LLLLLLLLLL        ....
PP            000000      000000    LLLLLLLLLL        ....


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

```
;     1          0001  0 %TITLE 'Dynamic memory management routines'
;     2          0002  0 MODULE POOL (IDENT = 'V04-000'
;     3          0003  0                 %BLISS32[,ADDRESSING_MODE(EXTERNAL=LONG_RELATIVE,NONEXTERNAL=LONG_RELATIVE)]
;     4          0004  0                 ) =
;     5          0005  1 BEGIN
;     6          0006  1 !
;     7          0007  1 !**********************************************************************
;     8          0008  1 !*                                                                    *
;     9          0009  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                           *
;    10          0010  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.            *
;    11          0011  1 !*  ALL RIGHTS RESERVED.                                             *
;    12          0012  1 !*                                                                    *
;    13          0013  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    14          0014  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    15          0015  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
;    16          0016  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
;    17          0017  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
;    18          0018  1 !*  TRANSFERRED.                                                       *
;    19          0019  1 !*                                                                    *
;    20          0020  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
;    21          0021  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
;    22          0022  1 !*  CORPORATION.                                                       *
;    23          0023  1 !*                                                                    *
;    24          0024  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    25          0025  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.            *
;    26          0026  1 !*                                                                    *
;    27          0027  1 !*                                                                    *
;    28          0028  1 !**********************************************************************
;    29          0029  1 !
;    30          0030  1 !++
;    31          0031  1 ! FACILITY:
;    32          0032  1 !   DSR (Digital Standard RUNOFF) /DSRPLUS DSRINDEX/INDEX Utility
;    33          0033  1 !
;    34          0034  1 ! ABSTRACT: Pooling for dynamic memory is supported.
;    35          0035  1 !
;    36          0036  1 ! ENVIRONMENT:   Transportable
;    37          0037  1 !
;    38          0038  1 ! AUTHOR:        R.W. Friday
;    39          0039  1 !
;    40          0040  1 ! CREATION DATE: January, 1979
;    41          0041  1 !
;    42          0042  1 ! MODIFIED BY:
;    43          0043  1 !       002     JPK00024        23-May-1983
;    44          0044  1 !               Modified lowercasing algorithm in PERMUTE. Now lowercase only
;    45          0045  1 !               if word contains only 1 letter or if second letter in word is
;    46          0046  1 !               lowercase. Picked up modules PAGMRG and POOL from DSR/DSRPLUS
;    47          0047  1 !               since they are no longer used by DSR/DSRPLUS.
;    48          0048  1 !
;    49          0049  1 !--
;    50          0050  1
;    51          0051  1 !
;    52          0052  1 ! TABLE OF CONTENTS:
;    53          0053  1 !
;    54          0054  1
;    55          0055  1 FORWARD ROUTINE
;    56          0056  1     CPOOL : NOVALUE,            ! Free all memory in a pool.
;    57          0057  1     FPOOL : NOVALUE,            ! Free a pool and all storage.
```

```
;    58          0058  1     GPOOL : NOVALUE,            ! Get and initialize a pool
;    59          0059  1     IPOOL : NOVALUE,            ! Initialize an area as a pool control area.
;    60          0060  1     XPOOL;                      ! Extend a pool.
;    61          0061  1
;    62          0062  1 !
;    63          0063  1 ! INCLUDE FILES:
;    64          0064  1 !
;    65          0065  1
;    66          0066  1 LIBRARY 'NXPORT:XPORT';         ! XPORT macros
;    67          0067  1 REQUIRE 'REQ:DMDEFS';           ! POOL_..., PAD_..., & GET_SEG_ADDR def's
;    68          0148  1
;    69          0149  1 !
;    70          0150  1 ! MACROS:
;    71          0151  1 !
;    72          0152  1
;    73          0153  1 !
;    74          0154  1 ! EQUATED SYMBOLS:
;    75          0155  1 !
;    76          0156  1
;    77          0157  1 LITERAL
;    78          0158  1     TRUE = 1,
;    79          0159  1     FALSE = 0;
;    80          0160  1
;    81          0161  1 !
;    82          0162  1 ! OWN STORAGE:
;    83          0163  1 !
;    84          0164  1
;    85          0165  1 !
;    86          0166  1 ! EXTERNAL REFERENCES:
;    87          0167  1 !
```

```
;    89          0168  1 %SBTTL 'CPOOL -- Free all memory in a pool'
;    90          0169  1 GLOBAL ROUTINE cpool (area) : NOVALUE =
;    91          0170  1
;    92          0171  1 !++
;    93          0172  1 ! FUNCTIONAL DESCRIPTION:
;    94          0173  1 !
;    95          0174  1 !       CPOOL deallocates all memory allocated to a pool.
;    96          0175  1 !
;    97          0176  1 ! FORMAL PARAMETERS:
;    98          0177  1 !
;    99          0178  1 !       AREA is the address of the pool.
;   100          0179  1 !
;   101          0180  1 ! IMPLICIT INPUTS:
;   102          0181  1 !
;   103          0182  1 !       NONE
;   104          0183  1 !
;   105          0184  1 ! IMPLICIT OUTPUTS:
;   106          0185  1 !
;   107          0186  1 !       NONE
;   108          0187  1 !
;   109          0188  1 ! ROUTINE VALUE:
;   110          0189  1 ! COMPLETION CODES:
;   111          0190  1 !
;   112          0191  1 !       NONE
;   113          0192  1 !
;   114          0193  1 ! SIDE EFFECTS:
;   115          0194  1 !
;   116          0195  1 !       NONE
;   117          0196  1 !
;   118          0197  1 !--
;   119          0198  1
;   120          0199  2     BEGIN
;   121          0200  2
;   122          0201  2     BIND                                        !Point to start of segment list.
;   123          0202  2         SEGMENT_LIST = ..AREA + POOL_CNTRL_SIZE*%UPVAL;
;   124          0203  2
;   125          0204  2     BIND                                        !AREA is the address of the pool.
;   126          0205  2         XAREA = ..AREA : POOL;
;   127          0206  2
;   128          0207  2     !
;   129          0208  2     ! Don't try to clear an empty pool.
;   130          0209  2     !
;   131          0210  2     IF ..AREA EQL 0 THEN RETURN;
;   132          0211  2
;   133          0212  2     INCR I FROM 0 TO .XAREA [POOL_ACT_PADS] - 1 DO
;   134          0213  3         BEGIN
;   135          0214  3         BIND                                    !Point to a descriptor.
;   136          0215  3             SEGMENT = SEGMENT_LIST + (.I*PAD_CNTRL_SIZE*%UPVAL) : PAD;
;   137          0216  3
;   138          0217  3                                                 !Free this chunk of memory.
;   139        P 0218  3         $XPO_FREE_MEM ( BINARY_DATA =
;   140        P 0219  3                         (.SEGMENT[PAD_SIZE],          !Size of segment
;   141          0220  3                          .SEGMENT[PAD_ADDRESS]) );    !Start of segment.
;   142          0221  2         END;
;   143          0222  2
;   144          0223  1     END;                        !End of CPOOL
```

```
                                                        .TITLE  POOL Dynamic memory management routines
                                                        .IDENT  \V04-000\

                                                        .EXTRN  XPO$FREE_MEM, XPO$FAILURE

                                                        .PSECT  $CODE$,NOWRT,2

                                                001C 00000      .ENTRY  CPOOL, Save R2,R3,R4                    ; 0169
                                          5E            08  C2 00002      SUBL2   #8, SP
                                    54    04    BC      0C  C1 00005      ADDL3   #12, @AREA, R4                  ; 0202
                                          53    04    BC  D0 0000A      MOVL    @AREA, R3                       ; 0205
                                                34  13 0000E      BEQL    3$                              ; 0210
                                          52        01  CE 00010      MNEGL   #1, I                           ; 0212
                                                2A  11 00013      BRB     2$
                                          50      6442  7E 00015 1$:  MOVAQ   (R4)[I], R0                     ; 0215
                                    6E    60        04  A5 00019      MULW3   #4, (R0), $XPO$DESC             ; 0220
                                    02    AE        02  90 0001D      MOVB    #2, $XPO$DESC+2
                                    03    AE        02  90 00021      MOVB    #2, $XPO$DESC+3
                                    04    AF    04  A0  D0 00025      MOVL    4(R0), $XPO$DESC+4
                                    00000000G   EF  9F 0002A      PUSHAB  XPO$FAILURE
                                                7E  7C 00030      CLRQ    -(SP)
                                          7E        02  CE 00032      MNEGL   #2, -(SP)
                                                10  AE  9F 00035      PUSHAB  $XPO$DESC
                              00000000G   EF        05  FB 00038      CALLS   #5, XPO$FREE_MEM
                                    D1    52    04  A3  F2 0003F 2$:  AOBLSS  4(R3), I, 1$                   ; 0212
                                                04  00044 3$:  RET                                         ; 0223

; Routine Size:  69 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  146          0224  1 %SBTTL 'FPOOL -- Free a pool and all storage'
;  147          0225  1 GLOBAL ROUTINE fpool (area) : NOVALUE =
;  148          0226  1
;  149          0227  1 !++
;  150          0228  1 ! FUNCTIONAL DESCRIPTION:
;  151          0229  1 !
;  152          0230  1 !       Free an entire storage pool.  All the areas pointed to by the active
;  153          0231  1 !       PADs are freed.  Then the pool area itself is freed.
;  154          0232  1 !
;  155          0233  1 ! FORMAL PARAMETERS:
;  156          0234  1 !
;  157          0235  1 !       AREA is the address of the POOL
;  158          0236  1 !
;  159          0237  1 ! IMPLICIT INPUTS:
;  160          0238  1 !
;  161          0239  1 !       NONE
;  162          0240  1 !
;  163          0241  1 ! IMPLICIT OUTPUTS:
;  164          0242  1 !
;  165          0243  1 !       NONE
;  166          0244  1 !
;  167          0245  1 ! ROUTINE VALUE:
;  168          0246  1 ! COMPLETION CODES:
;  169          0247  1 !
;  170          0248  1 !       NONE
;  171          0249  1 !
;  172          0250  1 ! SIDE EFFECTS:
;  173          0251  1 !
;  174          0252  1 !       NONE
;  175          0253  1 !
;  176          0254  1 !--
;  177          0255  1
;  178          0256  2     BEGIN
;  179          0257  2
;  180          0258  2     BIND                            !AREA is the address of the pool.
;  181          0259  2         XAREA = ..AREA : POOL;
;  182          0260  2
;  183          0261  2     CPOOL (.AREA);                  !First, deallocate all associated storate.
;  184          0262  2
;  185          0263  2     !
;  186          0264  2     ! Don't try to free the pool if it's empty.
;  187          0265  2     !
;  188          0266  2     IF ..AREA NEQ 0
;  189          0267  2     THEN
;  190          0268  2         $XPO_FREE_MEM ( BINARY_DATA = (.XAREA [POOL_ACT_SIZE], ..AREA));
;  191          0269  2
;  192          0270  2     .AREA = 0;                      !Clear pool pointer holder.
;  193          0271  1     END;                            !End of FPOOL


                                    0004 00000          .ENTRY  FPOOL, Save R2                          ; 0225
                              5E            08  C2 00002          SUBL2   #8, SP                          ;
                              52      04    BC  D0 00005          MOVL    @AREA, R2                       ; 0259
                                      04    AC  DD 00009          PUSHL   AREA                            ; 0261
```

```
                          AB    AF                 01   FB 0000C          CALLS   #1, CPOOL                                    ;
                                             04    BC   D5 00010          TSTL    @AREA                                        ; 0266
                                                   27   13 00013          BEQL    1$                                           ;
                     6E         08    A2          04   A5 00015          MULW3   #4, 8(R2), $XPO$DESC                         ; 0268
                                02    AE          02   90 0001A          MOVB    #2, $XPO$DESC+2                              ;
                                03    AE          02   90 0001E          MOVB    #2, $XPO$DESC+3                              ;
                                04    AE          04   BC   D0 00022     MOVL    @AREA, $XPO$DESC+4                           ;
                                          00000000G EF   9F 00027          PUSHAB  XPO$FAILURE                                  ;
                                                    7E   7C 0002D          CLRQ    -(SP)                                        ;
                                          7E        02   CE 0002F          MNEGL   #2, -(SP)                                    ;
                                                04    AE   9F 00032          PUSHAB  $XPO$DESC                                    ;
                              00000000G   EF          05   FB 00035          CALLS   #5, XPO$FREE_MEM                             ;
                                                04    BC   D4 0003C 1$:      CLRL    @AREA                                        ; 0270
                                                         04 0003F          RET                                                  ; 0271

; Routine Size:  64 bytes,    Routine Base:  $CODE$ + 0045
```

```
;  195     0272  1 %SBTTL 'GPOOL -- Get and initialize a pool'
;  196     0273  1 GLOBAL ROUTINE gpool (area, count) : NOVALUE =
;  197     0274  1
;  198     0275  1 !++
;  199     0276  1 ! FUNCTIONAL DESCRIPTION:
;  200     0277  1 !
;  201     0278  1 !       GPOOL generates a pool area and initializes it.
;  202     0279  1 !
;  203     0280  1 ! FORMAL PARAMETERS:
;  204     0281  1 !
;  205     0282  1 !       AREA is set to the address of the generated pool.
;  206     0283  1 !       COUNT is the number of extensions to be supported.
;  207     0284  1 !
;  208     0285  1 ! IMPLICIT INPUTS:
;  209     0286  1 !
;  210     0287  1 !       NONE
;  211     0288  1 !
;  212     0289  1 ! IMPLICIT OUTPUTS:
;  213     0290  1 !
;  214     0291  1 !       NONE
;  215     0292  1 !
;  216     0293  1 ! ROUTINE VALUE:
;  217     0294  1 ! COMPLETION CODES:
;  218     0295  1 !
;  219     0296  1 !       Returns TRUE if generation was successful, otherwise FALSE.
;  220     0297  1 !
;  221     0298  1 ! SIDE EFFECTS:
;  222     0299  1 !
;  223     0300  1 !       NONE
;  224     0301  1 !
;  225     0302  1 !--
;  226     0303  1
;  227     0304  2     BEGIN
;  228     0305  2     LOCAL
;  229     0306  2         SIZE;
;  230     0307  2
;  231     0308  2     SIZE = .COUNT*PAD_CNTRL_SIZE + POOL_CNTRL_SIZE;     !Physical size of pool control area.
;  232     0309  2
;  233     0310  2     $XPO_GET_MEM (FULLWORDS =.SIZE, RESULT = .AREA);    !Allocate pool control area.
;  234     0311  2     IPOOL(.AREA, .SIZE);                                !Initialize pool control area.
;  235     0312  1     END;                            !End of GPOOL


                                                .EXTRN  XPO$ALLOC_MEM

                                    0004 00000  .ENTRY  GPOOL, Save R2                  ; 0273
                              5E    04   C2 00002  SUBL2  #4, SP                       ;
                              52 08 AC   D0 00005  MOVL   COUNT, R2                     ; 0308
                              52    02   C4 00009  MULL2  #2, SIZE                      ;
                              52    03   C0 0000C  ADDL2  #3, SIZE                      ;
                        02020000    8F   DD 0000F  PUSHL  #33685504                     ; 0310
                              04    AE   D4 00015  CLRL   $XPO$DESC+4                   ;
                      00000000G     EF   9F 00018  PUSHAB XPO$FAILURE                   ;
                              7E    7C 0001E  CLRQ   -(SP)                            ;
                              7E    01   CE 00020  MNEGL  #1, -(SP)                    ;
                                 10 AE   9F 00023  PUSHAB $XPO$DESC                     ;
```

```
                 7E              52              02  78 00026          ASHL    #2, SIZE, -(SP)                                 ;
    00000000G    EF                              06  FB 0002A          CALLS   #6, XPO$ALLOC_MEM                               ;
                 05                              50  E9 00031          BLBC    $XPO$STATUS, T$                                 ;
          04     BC      04                      AE  D0 00034          MOVL    $XPO$DESC+4, @AREA                              ;
                                                 52  DD 00039 1$:      PUSHL   SIZE                                            ; 0311
                                          04     AC  DD 0003B          PUSHL   AREA                                            ;
    00000000V    EF                              02  FB 0003E          CALLS   #2, IPOOL                                       ;
                                                     04 00045          RET                                                     ; 0312

; Routine Size:  70 bytes,    Routine Base:  $CODE$ + 0085
```

```
;  237          0313  1 %SBTTL 'IPOOL -- Initialize a pool control area'
;  238          0314  1 GLOBAL ROUTINE ipool (area, size) : NOVALUE =
;  239          0315  1
;  240          0316  1 !++
;  241          0317  1 ! FUNCTIONAL DESCRIPTION:
;  242          0318  1 !
;  243          0319  1 !       IPOOL initializes a pool area.
;  244          0320  1 !
;  245          0321  1 ! FORMAL PARAMETERS:
;  246          0322  1 !
;  247          0323  1 !       AREA is the address of the generated pool.
;  248          0324  1 !       SIZE is the number of BPVALs in the area.
;  249          0325  1 !
;  250          0326  1 ! IMPLICIT INPUTS:
;  251          0327  1 !
;  252          0328  1 !       NONE
;  253          0329  1 !
;  254          0330  1 ! IMPLICIT OUTPUTS:
;  255          0331  1 !
;  256          0332  1 !       NONE
;  257          0333  1 !
;  258          0334  1 ! ROUTINE VALUE:
;  259          0335  1 ! COMPLETION CODES:
;  260          0336  1 !
;  261          0337  1 !       NONE
;  262          0338  1 !
;  263          0339  1 ! SIDE EFFECTS:
;  264          0340  1 !
;  265          0341  1 !       NONE
;  266          0342  1 !
;  267          0343  1 !--
;  268          0344  1
;  269          0345  2     BEGIN
;  270          0346  2
;  271          0347  2     BIND
;  272          0348  2         XAREA = ..AREA : POOL;
;  273          0349  2
;  274          0350  2                                                 !Maximum number of areas.
;  275          0351  2     XAREA [POOL_MAX_PADS] = (.SIZE - POOL_CNTRL_SIZE)/PAD_CNTRL_SIZE;
;  276          0352  2     XAREA [POOL_ACT_SIZE] = .SIZE;              !Physical size of pool control area.
;  277          0353  2     XAREA [POOL_ACT_PADS] = 0;                  !No areas yet.
;  278          0354  1     END;                            !End of IPOOL


                                                0000 00000          .ENTRY  IPOOL, Save nothing                             ; 0314
                            50          04  BC  D0 00002          MOVL    @AREA, R0                                       ; 0348
                    51      08          AC  03  C3 00006          SUBL3   #3, SIZE, R1                                    ; 0351
                    60              51      02  C7 0000B          DIVL3   #2, R1, (R0)                                    ;
                            08          A0  08  AC  D0 0000F      MOVL    SIZE, 8(R0)                                     ; 0352
                                        04  A0  D4 00014          CLRL    4(R0)                                           ; 0353
                                                04 00017          RET                                                     ; 0354

; Routine Size:  24 bytes,    Routine Base:  $CODE$ + 00CB
```

```
;  280          0355  1 %SBTTL 'XPOOL -- Extend a pool'
;  281          0356  1 GLOBAL ROUTINE xpool (area,size) =
;  282          0357  1
;  283          0358  1 !++
;  284          0359  1 ! FUNCTIONAL DESCRIPTION:
;  285          0360  1 !
;  286          0361  1 !       XPOOL extends a dynamic storage pool by allocating a
;  287          0362  1 !       block of dynamic storage and chaining it into the pool.
;  288          0363  1 !
;  289          0364  1 ! FORMAL PARAMETERS:
;  290          0365  1 !
;  291          0366  1 !       AREA is the address of the pool to be extended.
;  292          0367  1 !       SIZE is the number of BPVALS needed.
;  293          0368  1 !
;  294          0369  1 ! IMPLICIT INPUTS:
;  295          0370  1 !
;  296          0371  1 !       NONE
;  297          0372  1 !
;  298          0373  1 ! IMPLICIT OUTPUTS:
;  299          0374  1 !
;  300          0375  1 !       NONE
;  301          0376  1 !
;  302          0377  1 ! ROUTINE VALUE:
;  303          0378  1 ! COMPLETION CODES:
;  304          0379  1 !
;  305          0380  1 !       Returns address of start of new segment if one could be
;  306          0381  1 !       could be allocated.  Otherwise returns 0.
;  307          0382  1 !
;  308          0383  1 ! SIDE EFFECTS:
;  309          0384  1 !
;  310          0385  1 !       NONE
;  311          0386  1 !
;  312          0387  1 !--
;  313          0388  1
;  314          0389  2     BEGIN
;  315          0390  2
;  316          0391  2     BIND                            !AREA is the address of the pool.
;  317          0392  2         XAREA = ..AREA : POOL;
;  318          0393  2
;  319          0394  2     IF .XAREA [POOL_ACT_PADS] EQL .XAREA [POOL_MAX_PADS] THEN RETURN 0;
;  320          0395  2
;  321          0396  3     BEGIN
;  322          0397  3     BIND                            !Point to where descriptor gets built.
;  323          0398  3         NEW_PAD = ..AREA + %UPVAL*( (.XAREA [POOL_ACT_PADS] * PAD_CNTRL_SIZE) + POOL_CNTRL_SIZE ) : PAD;
;  324          0399  3
;  325          0400  3                                     !Allocate the segment.
;  326          0401  3     $XPO_GET_MEM ( FULLWORDS = .SIZE, RESULT = NEW_PAD [PAD_ADDRESS]);
;  327          0402  3     NEW_PAD [PAD_SIZE] = .SIZE; !Remember physical size of segment.
;  328          0403  3                                     !Update PAD count
;  329          0404  3     XAREA [POOL_ACT_PADS] = .XAREA [POOL_ACT_PADS] + 1;
;  330          0405  3
;  331          0406  3                                     !Tell user where the segment is.
;  332          0407  3     RETURN .NEW_PAD [ PAD_ADDRESS];
;  333          0408  2     END;
;  334          0409  2
;  335          0410  1     END;                            !End of XPOOL
```

```
                                                 000C 00000          .ENTRY  XPOOL, Save R2,R3                               ; 0356
                                              5E    08   C2 00002    SUBL2   #8, SP                                          ;
                                              52    04   BC D0 00005  MOVL    @AREA, R2                                       ; 0392
                                              50    04   A2 D0 00009  MOVL    4(R2), R0                                       ; 0394
                                              62         50 D1 0000D  CMPL    R0, (R2)                                        ;
                                                         3D 13 00010  BEQL    2$                                              ;
                                              53    0C A240 7E 00012  MOVAQ   12(R2)[R0], R3                                  ; 0398
                                              6E 02020000   8F D0 00017  MOVL    #33685504, $XPO$DESC                        ; 0401
                                                    04   AE D4 0001E  CLRL    $XPO$DESC+4                                     ;
                                                 00000000G EF 9F 00021  PUSHAB  XPO$FAILURE                                     ;
                                                         7E 7C 00027  CLRQ    -(SP)                                           ;
                                              7E         01 CE 00029  MNEGL   #1, -(SP)                                       ;
                                                    10   AE 9F 0002C  PUSHAB  $XPO$DESC                                      ;
                                        7E          08   AC 02 78 0002F  ASHL    #2, SIZE, -(SP)                             ;
                                           00000000G EF    06 FB 00034  CALLS   #6, XPO$ALLOC_MEM                             ;
                                                    05         50 E9 0003B  BLBC    $XPO$STATUS, 1$                         ;
                                                    04   A3 04 AE D0 0003E  MOVL    $XPO$DESC+4, 4(R3)                      ;
                                                    63   08 AC D0 00043 1$:  MOVL    SIZE, (R3)                              ; 0402
                                                         04 A2 D6 00047  INCL    4(R2)                                       ; 0404
                                              50         04 A3 D0 0004A  MOVL    4(R3), R0                                   ; 0407
                                                            04 0004E  RET                                                    ;
                                                         50 D4 0004F 2$:  CLRL    R0                                         ; 0410
                                                            04 00051  RET                                                    ;

; Routine Size:  82 bytes,    Routine Base:  $CODE$ + 00E3

;  336          0411  1
;  337          0412  1 END                                 !End of module
;  338          0413  0 ELUDOM



;                                    PSECT SUMMARY
;
;      Name                        Bytes                           Attributes
;
;  $CODE$                            309  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                                  Library Statistics
;                                                 -------- Symbols --------      Pages       Processing
;      File                                       Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]XPORT.L32;1                  590        21         3       252          00:00.2
```

```
;                                       COMMAND QUALIFIERS

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

; Size:         309 code + 0 data bytes
; Run Time:       00:09.0
; Elapsed Time:   00:17.5
; Lines/CPU Min:   2747
; Lexemes/CPU-Min: 43124
; Memory Used:  63 pages
; Compilation Complete
```

PAGE
LIS

PERMUTE
LIS

PAGCMB
LIS

PANPA
LIS

RCS
LIS

POOL
LIS

RINIT
LIS

PERIOD
LIS

PUS
LIS

PUTNDX
LIS

PARSEP
LIS

PUTTOC
LIS

REQUIR
LIS

PAGMRG
LIS

PARAG
LIS

REPEAT
LIS