```
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
```

FILEID**NDXDAT

```
NN       NN  DDDDDDDD    XX       XX  DDDDDDDD      AAAAAA    TTTTTTTTTT
NN       NN  DDDDDDDD    XX       XX  DDDDDDDD      AAAAAA    TTTTTTTTTT
NN       NN  DD      DD  XX       XX  DD      DD   AA      AA       TT
NN       NN  DD      DD  XX       XX  DD      DD   AA      AA       TT
NNNN     NN  DD      DD    XX  XX    DD      DD   AA      AA       TT
NNNN     NN  DD      DD    XX  XX    DD      DD   AA      AA       TT
NN  NN   NN  DD      DD      XX      DD      DD   AA      AA       TT
NN  NN   NN  DD      DD      XX      DD      DD   AA      AA       TT
NN    NNNN  DD      DD    XX  XX    DD      DD   AAAAAAAAAA       TT
NN    NNNN  DD      DD    XX  XX    DD      DD   AAAAAAAAAA       TT
NN       NN  DD      DD  XX       XX  DD      DD   AA      AA       TT          ....
NN       NN  DD      DD  XX       XX  DD      DD   AA      AA       TT          ....
NN       NN  DDDDDDDD    XX       XX  DDDDDDDD    AA      AA       TT          ....
NN       NN  DDDDDDDD    XX       XX  DDDDDDDD    AA      AA       TT          ....


LL             IIIIII      SSSSSSSS
LL             IIIIII      SSSSSS^S
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 'NDXDAT - Index pool data manipulation routines'
;  2     0002  0 MODULE NDXDAT (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 !*                                                                           *
; 10     0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
; 11     0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
; 12     0012  1 !*  ALL RIGHTS RESERVED.                                                     *
; 13     0013  1 !*                                                                           *
; 14     0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
; 15     0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
; 16     0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
; 17     0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
; 18     0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
; 19     0019  1 !*  TRANSFERRED.                                                             *
; 20     0020  1 !*                                                                           *
; 21     0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
; 22     0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
; 23     0023  1 !*  CORPORATION.                                                             *
; 24     0024  1 !*                                                                           *
; 25     0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
; 26     0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
; 27     0027  1 !*                                                                           *
; 28     0028  1 !*                                                                           *
; 29     0029  1 !*****************************************************************************
; 30     0030  1 !
; 31     0031  1
; 32     0032  1 !++
; 33     0033  1 ! FACILITY:
; 34     0034  1 !   DSR (Digital Standard RUNOFF) /DSRPLUS DSRINDEX/INDEX Utility
; 35     0035  1 !
; 36     0036  1 ! ABSTRACT:
; 37     0037  1 !   This module contains routines used to manipulate the binary index pool
; 38     0038  1 !
; 39     0039  1 ! ENVIRONMENT:   Transportable
; 40     0040  1 !
; 41     0041  1 ! AUTHOR:        JPK
; 42     0042  1 !
; 43     0043  1 ! CREATION DATE: February 1982
; 44     0044  1 !
; 45     0045  1 ! MODIFIED BY:
; 46     0046  1 !
; 47     0047  1 !       006     JPK00015        04-Feb-1983
; 48     0048  1 !               Cleaned up module names, modified revision history to
; 49     0049  1 !               conform with established standards. Updated copyright dates.
; 50     0050  1 !
; 51     0051  1 !       005     JPK00012        24-Jan-1983
; 52     0052  1 !               Modified NDXVMSMSG.MSG to define error messages for both
; 53     0053  1 !               DSRINDEX and INDEX.
; 54     0054  1 !               Added require of NDXVMSREQ.R32 to NDXOUT, NDXFMT, NDXDAT,
; 55     0055  1 !               INDEX, NDXMSG, NDXXTN, NDXTMS, NDXVMS and NDXPAG for BLISS32.
; 56     0056  1 !               Since this file defines the error message literals,
; 57     0057  1 !               the EXTERNAL REFERENCEs for the error message literals
```

```
;    58          0058  1 !                 have been removed.
;    59          0059  1 !
;    60          0060  1 !         004     JPK00010        24-Jan-1983
;    61          0061  1 !                 Removed routines GETDAT and UPDDAT from NDXDAT - they
;    62          0062  1 !                 performed no useful function. Removed references to these
;    63          0063  1 !                 routines from NDXOUT, NDXFMT, and NDXMSG.
;    64          0064  1 !                 Removed reference to XPOOL in NDXOUT - not used.
;    65          0065  1 !
;    66          0066  1 !         003     JPK00009        24-Jan-1983
;    67          0067  1 !                 Modified to enhance performance. The sort buckets have each
;    68          0068  1 !                 been divided into 27 sub-buckets; 1 for each letter and 1
;    69          0069  1 !                 for non-alphas. Removed reference to BUCKET from INDEX.
;    70          0070  1 !                 Definition of the structure was added to NDXPOL. References
;    71          0071  1 !                 to BUCKET were changed in modules NDXOUT, NDXINI, NDXFMT
;    72          0072  1 !                 and NDXDAT.
;    73          0073  1 !
;    74          0074  1 !         002     JPK00004        24-Sep-1982
;    75          0075  1 !                 Modified NDXOUT, NDXMSG, NDXFMT, and NDXDAT for TOPS-20.
;    76          0076  1 !                 Strings stored in the index pool use the first fullword
;    77          0077  1 !                 for their length. References to these strings were incorrect.
;    78          0078  1 !
;    79          0079  1 !--
```

```
;    81          0080  1 !
;    82          0081  1 ! TABLE OF CONTENTS:
;    83          0082  1 !
;    84          0083  1 FORWARD ROUTINE
;    85          0084  1     XINIT  : NOVALUE,          ! Initialize index pool
;    86          0085  1     SAVDAT,                    ! Store data in pool
;    87          0086  1     XMEM   : NOVALUE,          ! Verify availability of pool space
;    88          0087  1     GENTRY;                    ! Get index entry from pool
;    89          0088  1
;    90          0089  1 !
;    91          0090  1 ! INCLUDE FILES:
;    92          0091  1 !
;    93          0092  1 LIBRARY 'NXPORT:XPORT';
;    94          0093  1
;    95          0094  1 SWITCHES LIST (REQUIRE);
;    96          0095  1
;    97          0096  1 REQUIRE 'REQ:NDXPOL';
```

```
;  R0097  1      !
;  R0098  1      ! Version:      'V04-000'
;  R0099  1      !
;  R0100  1      !****************************************************************************
;  R0101  1      !*                                                                          *
;  R0102  1      !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                *
;  R0103  1      !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
;  R0104  1      !*  ALL RIGHTS RESERVED.                                                    *
;  R0105  1      !*                                                                          *
;  R0106  1      !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;  R0107  1      !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;  R0108  1      !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;  R0109  1      !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;  R0110  1      !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;  R0111  1      !*  TRANSFERRED.                                                            *
;  R0112  1      !*                                                                          *
;  R0113  1      !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;  R0114  1      !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;  R0115  1      !*  CORPORATION.                                                            *
;  R0116  1      !*                                                                          *
;  R0117  1      !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;  R0118  1      !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
;  R0119  1      !*                                                                          *
;  R0120  1      !*                                                                          *
;  R0121  1      !****************************************************************************
;  R0122  1      !
;  R0123  1
;  R0124  1      !++
;  R0125  1      ! FACILITY:
;  R0126  1      !   DSR (Digital Standard RUNOFF) /DSRPLUS DSRINDEX/INDEX Utility
;  R0127  1      !
;  R0128  1      ! ABSTRACT:
;  R0129  1      !   This file contains literals and macros defining the data structures
;  R0130  1      !   found in the internal index pool
;  R0131  1      !
;  R0132  1      ! ENVIRONMENT:   Transportable
;  R0133  1      !
;  R0134  1      ! AUTHOR:        JPK
;  R0135  1      !
;  R0136  1      ! CREATION DATE: January 1982
;  R0137  1      !
;  R0138  1      ! MODIFIED BY:
;  R0139  1      !
;  R0140  1      !       003     JPK00015        04-Feb-1983
;  R0141  1      !               Cleaned up module names, modified revision history to
;  R0142  1      !               conform with  stablished standards. Updated copyright dates.
;  R0143  1      !
;  R0144  1      !       002     JPK00009        24-Jan-1983
;  R0145  1      !               Modified to enhance performance. The sort buckets have each
;  R0146  1      !               been divided into 27 sub-buckets; 1 for each letter and 1
;  R0147  1      !               for non-alphas. Removed reference to BUCKET from INDEX.
;  R0148  1      !               Definition of the structure was added to NDXPOL. References
;  R0149  1      !               to BUCKET were changed in modules NDXOUT, NDXINI, NDXFMT
;  R0150  1      !               and NDXDAT.
;  R0151  1      !
;  R0152  1      !--
;  R0153  1
```

```
;  R0154  1     ! Index entry
;  R0155  1
;  R0156  1     $FIELD XE_FIELDS =
;  R0157  1         SET
;  R0158  1
;  R0159  1         XE$A_PREV           = [$ADDRESS],          ! Link to previous item
;  R0160  1         XE$A_NEXT           = [$ADDRESS],          ! Link to next item
;  R0161  1         XE$A_SUBX           = [$ADDRESS],          ! Sub index pointer
;  R0162  1         XE$A_REF            = [$ADDRESS],          ! Reference pointer
;  R0163  1         XE$A_TEXT           = [$ADDRESS],          ! Pointer to text of index item
;  R0164  1         XE$A_SORT_AS        = [$ADDRESS],          ! Pointer to SORT_AS string
;  R0165  1         XE$H_SUBC           = [$SHORT_INTEGER],    ! Sub index level
;  R0166  1
;  R0167  1         XE$V_FLAGS          = [$SHORT_INTEGER],    ! Entry flags
;  R0168  1
;  R0169  1             $OVERLAY (XE$V_FLAGS)
;  R0170  1
;  R0171  1             XE$V_BARS               = [$BIT],      ! Change bar flag
;  R0172  1
;  R0173  1             $CONTINUE
;  R0174  1
;  R0175  1         XE$A_BOOK_LIST      = [$ADDRESS]           ! Master index book name list
;  R0176  1
;  R0177  1         $ALIGN (FULLWORD)
;  R0178  1
;  R0179  1         TES;
;  R0180  1
;  R0181  1     LITERAL
;  R0182  1         XE$K_LENGTH = $FIELD_SET_SIZE;
;  R0183  1
;  R0184  1     MACRO
;  R0185  1         $XE_BLOCK = BLOCK [XE$K_LENGTH] FIELD (XE_FIELDS) %;
;  R0186  1
;  R0187  1     ! End of Index entry
;  R0188  1
;  R0189  1
;  R0190  1     ! Reference entry
;  R0191  1
;  R0192  1     $FIELD XX_FIELDS =
;  R0193  1         SET
;  R0194  1
;  R0195  1         XX$A_LINK           = [$ADDRESS],          ! Link to additional entries
;  R0196  1         XX$A_APPEND         = [$ADDRESS],          ! APPEND text pointer
;  R0197  1         XX$H_PAGE           = [$SHORT_INTEGER],    ! Transaction number
;  R0198  1
;  R0199  1         XX$V_FLAGS          = [$SHORT_INTEGER],    ! Display attributes
;  R0200  1
;  R0201  1             $OVERLAY (XX$V_FLAGS)
;  R0202  1
;  R0203  1             XX$V_BOLD       = [$BIT],              ! Bold page reference
;  R0204  1             XX$V_UNDERLINE  = [$BIT],              ! Underline page reference
;  R0205  1             XX$V_BEGIN      = [$BIT],              ! Begin page range
;  R0206  1             XX$V_END        = [$BIT],              ! End page range
;  R0207  1
;  R0208  1             $CONTINUE
;  R0209  1
;  R0210  1         XX$A_BOOK           = [$ADDRESS]           ! Master index book name
```

```
;  R0211  1
;  R0212  1          $ALIGN (FULLWORD)
;  R0213  1
;  R0214  1          TES;
;  R0215  1
;  R0216  1      LITERAL
;  R0217  1          XX$K_LENGTH = $FIELD_SET_SIZE;
;  R0218  1
;  R0219  1      MACRO
;  R0220  1          $XX_BLOCK = BLOCK [XX$K_LENGTH] FIELD (XX_FIELDS) %;
;  R0221  1
;  R0222  1      ! End of Reference entry
;  R0223  1
;  R0224  1
;  R0225  1      ! Master index book reference entry
;  R0226  1
;  R0227  1      $FIELD XM_FIELDS =
;  R0228  1          SET
;  R0229  1
;  R0230  1          XM$A_LINK           = [$ADDRESS],           ! Link to additional entries
;  R0231  1          XM$A_BOOK           = [$ADDRESS]            ! Pointer to book name
;  R0232  1
;  R0233  1          TES;
;  R0234  1
;  R0235  1      LITERAL
;  R0236  1          XM$K_LENGTH = $FIELD_SET_SIZE;
;  R0237  1
;  R0238  1      MACRO
;  R0239  1          $XM_BLOCK = BLOCK [XM$K_LENGTH] FIELD (XM_FIELDS) %;
;  R0240  1
;  R0241  1      ! End of Master index book reference entry
;  R0242  1
;  R0243  1
;  R0244  1      ! Current Entry
;  R0245  1
;  R0246  1      $FIELD C_FIELDS =
;  R0247  1          SET
;  R0248  1
;  R0249  1          C$A_CURR            = [$ADDRESS],           ! Pointer to current cell
;  R0250  1          C$A_PREV            = [$ADDRESS],           ! Pointer to previous cell
;  R0251  1          C$A_HEAD            = [$ADDRESS],           ! Pointer to head of chain
;  R0252  1
;  R0253  1          $ALIGN (FULLWORD)
;  R0254  1
;  R0255  1          C$V_FLAGS           = [$INTEGER],           ! Current cell flags
;  R0256  1
;  R0257  1              $OVERLAY (C$V_FLAGS)
;  R0258  1
;  R0259  1              C$V_IDNS        = [$BIT]                ! Identical string flag
;  R0260  1
;  R0261  1              $CONTINUE
;  R0262  1
;  R0263  1          TES;
;  R0264  1
;  R0265  1      LITERAL
;  R0266  1          C$K_LENGTH = $FIELD_SET_SIZE;
;  R0267  1
```

```
;  R0268  1     MACRO
;  R0269  1         $C_BLOCK = BLOCK [C$K_LENGTH] FIELD (C_FIELDS) %;
;  R0270  1
;  R0271  1     ! End of current entry
;  R0272  1
;  R0273  1
;  R0274  1     !
;  R0275  1     ! Dummy datasets
;  R0276  1     !
;  R0277  1     LITERAL
;  R0278  1         DS_X_ENTRY  = XE$K_LENGTH,
;  R0279  1         DS_XX_ENTRY = XX$K_LENGTH,
;  R0280  1         DS_XM_ENTRY = XM$K_LENGTH,
;  R0281  1         DS_X_STRING = 0;
;  R0282  1
;  R0283  1     !
;  R0284  1     ! Structure definition for bucket array.
;  R0285  1     !
;  R0286  1     !       Buckets are arranged so that each row represents the first letter of
;  R0287  1     !       the string and each column represents the second letter of the string.
;  R0288  1     !
;  R0289  1     !       This approach is used only for master indexes as no performance
;  R0290  1     !       improvement is realised until about 10 input files have been processed.
;  R0291  1     !
;  R0292  1     !       Indexes which are not master indexes use only the first element of
;  R0293  1     !       each row, i.e., [0, 0] ... [26, 0].
;  R0294  1     !
;  R0295  1     !       The only exception is for nonalphabetic characters which use only
;  R0296  1     !       element [0, 0]. Elements [0, 1] ... [0, 26] are not used since mapping
;  R0297  1     !       all nonalphabetics into one row loses the sort order of the first
;  R0298  1     !       character in the string. For nonalphabetics to work correctly in a two
;  R0299  1     !       dimensional bucket scheme, the array would have to be at least 127 x 127
;  R0300  1     !
;  R0301  1     !            0   1   .   .   .  26
;  R0302  1     !          0 **  not used  .  .  .
;  R0303  1     !          1 A?  AA            AZ
;  R0304  1     !          .                   .
;  R0305  1     !          .                   .
;  R0306  1     !          .
;  R0307  1     !         26 Z?  ZA  .   .   .  ZZ
;  R0308  1     !
;  R0309  1     STRUCTURE
;  R0310  1         $BUCKET_ARRAY [ROW_IDX, COL_IDX; M, N] =
;  R0311  1             [M * N * %UPVA[] ($BUCKET_ARRAY + (ROW_IDX * N + COL_IDX) * %UPVAL);
;  R0312  1
;  R0313  1     !--     End of NDXPOL.REQ
```

```
;    98          0314  1
;    99        L 0315  1 %IF %BLISS (BLISS32)
;   100          0316  1 %THEN
;   101          0317  1
;   102          0318  1 REQUIRE 'REQ:NDXVMSREQ';
```

```
;  R0319  1
;  R0320  1        Version:      'V04-000'
;  R0321  1
;  R0322  1        ****************************************************************************
;  R0323  1        *                                                                          *
;  R0324  1        *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;  R0325  1        *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
;  R0326  1        *  ALL RIGHTS RESERVED.                                                    *
;  R0327  1        *                                                                          *
;  R0328  1        *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;  R0329  1        *  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;  R0330  1        *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;  R0331  1        *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;  R0332  1        *  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;  R0333  1        *  TRANSFERRED.                                                            *
;  R0334  1        *                                                                          *
;  R0335  1        *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;  R0336  1        *  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;  R0337  1        *  CORPORATION.                                                            *
;  R0338  1        *                                                                          *
;  R0339  1        *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;  R0340  1        *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
;  R0341  1        *                                                                          *
;  R0342  1        *                                                                          *
;  R0343  1        ****************************************************************************
;  R0344  1
;  R0345  1
;  R0346  1        ++
;  R0347  1        FACILITY:
;  R0348  1          DSR (Digital Standard RUNOFF) /DSRPLUS DSRINDEX/INDEX Utility
;  R0349  1
;  R0350  1        ABSTRACT:
;  R0351  1              This file contains external references to the error message numbers
;  R0352  1              for DSRINDEX/INDEX.
;  R0353  1
;  R0354  1              New messages must be defined in NDXVMSMSG.MSG and referenced here:
;  R0355  1              both in the MACRO section (for DSRINDEX) and the EXTERNAL LITERAL
;  R0356  1              section (for INDEX)
;  R0357  1
;  R0358  1        ENVIRONMENT:    VAX/VMS User Mode
;  R0359  1
;  R0360  1        AUTHOR:         JPK
;  R0361  1
;  R0362  1        CREATION DATE: 01-Feb-1983
;  R0363  1
;  R0364  1        MODIFIED BY:
;  R0365  1
;  R0366  1              004     JPK00022        30-Mar-1983
;  R0367  1                      Modified NDXVMS, NDXFMT, NDXPAG, NDXVMSMSG and NDXVMSREQ
;  R0368  1                      to generate TEX output. Added module NDXTEX.
;  R0369  1
;  R0370  1              003     JPK00021        28-Mar-1983
;  R0371  1                      Modified NDXT20 to include E2.0 functionality.
;  R0372  1                      Modified NDXCLIDMP, NDXFMT, NDXPAG, NDXVRS to require RNODEF
;  R0373  1                      for BLISS36 and to remove any conditional require based on
;  R0374  1                      DSRPLUS_DEF.
;  R0375  1
```

```
;  R0376  1     !       002     JPK00010        04-Feb-1983
;  R0377  1     !               Cleaned up module names, modified revision history to
;  R0378  1     !               conform with established standards. Updated copyright dates.
;  R0379  1     !
;  R0380  1     !--
;  R0381  1
;  R0382  1     REQUIRE 'REQ:RNODEF';
```

```
;  R0383  1        !
;  R0384  1        ! Version:      'V04-000'
;  R0385  1        !
;  R0386  1        !****************************************************************************
;  R0387  1        !*                                                                          *
;  R0388  1        !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;  R0389  1        !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
;  R0390  1        !*  ALL RIGHTS RESERVED.                                                    *
;  R0391  1        !*                                                                          *
;  R0392  1        !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;  R0393  1        !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;  R0394  1        !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;  R0395  1        !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;  R0396  1        !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;  R0397  1        !*  TRANSFERRED.                                                            *
;  R0398  1        !*                                                                          *
;  R0399  1        !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;  R0400  1        !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;  R0401  1        !*  CORPORATION.                                                            *
;  R0402  1        !*                                                                          *
;  R0403  1        !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;  R0404  1        !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
;  R0405  1        !*                                                                          *
;  R0406  1        !*                                                                          *
;  R0407  1        !****************************************************************************
;  R0408  1        !
;  R0409  1
;  R0410  1        !++
;  R0411  1        ! FACILITY:      DSR (Digital Standard RUNOFF) / DSRPLUS
;  R0412  1        !
;  R0413  1        ! ABSTRACT:
;  R0414  1        !       Converts BLISS/VARIANT values into useful names.
;  R0415  1        !
;  R0416  1        ! ENVIRONMENT:   Transportable BLISS
;  R0417  1        !
;  R0418  1        ! AUTHOR:        Rich Friday
;  R0419  1        !
;  R0420  1        ! CREATION DATE: 1978
;  R0421  1        !
;  R0422  1        ! MODIFIED BY:
;  R0423  1        !
;  R0424  1        !       016     KAD00016        Ray Marshall    19-Mar-1984
;  R0425  1        !               Added GERMAN, FRENCH, & ITALIAN.
;  R0426  1        !
;  R0427  1        !       015     KAD00015        Keith Dawson    18-Apr-1983
;  R0428  1        !               Made the LN01 conditional the default for vanilla DSR --
;  R0429  1        !                 its value is 0 (no variant supplied).
;  R0430  1        !
;  R0431  1        !       014     KAD00014        Keith Dawson    22-Mar-1983
;  R0432  1        !               Asserted the LN01 conditional when DSRPLUS is asserted.
;  R0433  1        !
;  R0434  1        !       013     KAD00013        Keith Dawson    20-Mar-1983
;  R0435  1        !               Removed all references to .BIX and .BTC files.
;  R0436  1        !
;  R0437  1        !       012     KAD00012        Keith Dawson    07-Mar-1983
;  R0438  1        !               Global edit of all modules. Updated module names, idents,
;  R0439  1        !               copyright dates. Changed require files to BLISS library.
```

```
;  R0440  1      !
;  R0441  1      !--
;  R0442  1
;  R0443  1      !++
;  R0444  1      !       D E F I N I T I O N   O F  /VARIANT  B I T S
;  R0445  1      !
;  R0446  1      !       The bit assignments are as follows:
;  R0447  1      !
;  R0448  1      !       Bit  Weight   Meaning
;  R0449  1      !-------------------------------------------------------------------------
;  R0450  1      !       --      0     If no /VARIANT is supplied (as for vanilla DSR),
;  R0451  1      !                     compile with LN01 support. LN01 support is also
;  R0452  1      !                     implied by the DSRPLUS variant.
;  R0453  1      !
;  R0454  1      !        0      1     CLEAR =   Unassigned
;  R0455  1      !                     SET   =   Unassigned
;  R0456  1      !
;  R0457  1      !        1      2     CLEAR =   Normal compile
;  R0458  1      !                     SET   =   Compile for DSRPLUS
;  R0459  1      !
;  R0460  1      !       4-6    16     CLEAR =   English (American) version
;  R0461  1      !                     SET   =   16 = German (Austrian)
;  R0462  1      !                               32 = French
;  R0463  1      !                               48 = Italian
;  R0464  1      !--
;  R0465  1
;  R0466  1      !-------------------------------------------------------------------------
;  R0467  1      !   This variable (LN01) controls whether or not to compile an LN01-flavored
;  R0468  1      !   DSR. It is asserted by default, and also whenever DSRPLUS is asserted.
;  R0469  1      !
;  R0470  1      !   Modules utilizing LN01 are:
;  R0471  1      !
;  R0472  1      !       DOOPTS  NOUT
;  R0473  1
;  R0474  1      COMPILETIME
;  R0475  1          ln01 =
;  R0476  2              ( (%VARIANT EQL 0) OR  %VARIANT/2 )
;  R0477  1          ;
;  R0478  1
;  R0479  1      !-------------------------------------------------------------------------
;  R0480  1      !   This variable (DSRPLUS) controls compilation for the DSRPLUS program.
;  R0481  1      !
;  R0482  1      !   All modules utilize DSRPLUS.
;  R0483  1
;  R0484  1      COMPILETIME
;  R0485  1          dsrplus =
;  R0486  2              ( %VARIANT/2 )
;  R0487  1          ;
;  R0488  1
;  R0489  1      !-------------------------------------------------------------------------
;  R0490  1      !   This variable (FLIP) controls compilation of FLIP features of DSRPLUS.
;  R0491  1      !   It assures that FLIP features are compiled only on VMS systems.
;  R0492  1      !
;  R0493  1      !   Modules utilizing FLIP are many and various.
;  R0494  1
;  R0495  1      COMPILETIME
;  R0496  1          flip =
```

```
;  R0497  2              ( %VARIANT/2 AND %BLISS(BLISS32) )
;  R0498  1          ;
;  R0499  1
;  R0500  1      !-----------------------------------------------------------------------
;  R0501  1      !       4-6    16     CLEAR =   English (American) version
;  R0502  1      !                     SET   =   16 = German (Austrian)
;  R0503  1      !                               32 = French
;  R0504  1      !                               48 = Italian
;  R0505  1      COMPILETIME
;  R0506  1          German  = ( %VARIANT/16 AND NOT %VARIANT/32 AND NOT %VARIANT/64 ) ;
;  R0507  1      COMPILETIME
;  R0508  1          French  = ( NOT %VARIANT/16 AND %VARIANT/32 AND NOT %VARIANT/64 ) ;
;  R0509  1      COMPILETIME
;  R0510  1          Italian = ( %VARIANT/16 AND %VARIANT/32 AND NOT %VARIANT/64 ) ;
;  R0511  1      !-----------------------------------------------------------------------
;  R0512  1      !                       End of RNODEF.REQ
```

```
;  R0513  1
; LR0514  1      %IF NOT DSRPLUS
;  R0515  1      %THEN
;  R0516  1
;  R0517  1      MACRO
;  R0518  1           INDEX$_BADLOGIC   = DSRINDEX$_BADLOGIC   %,
;  R0519  1           INDEX$_BADVALUE   = DSRINDEX$_BADVALUE   %,
;  R0520  1           INDEX$_INSVIRMEM  = DSRINDEX$_INSVIRMEM  %,
;  R0521  1           INDEX$_LINELENG   = DSRINDEX$_LINELENG   %,
;  R0522  1           INDEX$_NOREF      = DSRINDEX$_NOREF      %,
;  R0523  1           INDEX$_OPENIN     = DSRINDEX$_OPENIN     %,
;  R0524  1           INDEX$_OPENOUT    = DSRINDEX$_OPENOUT    %,
;  R0525  1           INDEX$_TOOMANY    = DSRINDEX$_TOOMANY    %,
;  R0526  1           INDEX$_VALERR     = DSRINDEX$_VALERR     %,
;  R0527  1           INDEX$_CANTBAL    = DSRINDEX$_CANTBAL    %,
;  R0528  1           INDEX$_CLOSEQUOT  = DSRINDEX$_CLOSEQUOT  %,
;  R0529  1           INDEX$_CONFQUAL   = DSRINDEX$_CONFQUAL   %,
;  R0530  1           INDEX$_CTRLCHAR   = DSRINDEX$_CTRLCHAR   %,
;  R0531  1           INDEX$_DOESNTFIT  = DSRINDEX$_DOESNTFIT  %,
;  R0532  1           INDEX$_DUPBEGIN   = DSRINDEX$_DUPBEGIN   %,
;  R0533  1           INDEX$_EMPTYIN    = DSRINDEX$_EMPTYIN    %,
;  R0534  1           INDEX$_IGNORED    = DSRINDEX$_IGNORED    %,
;  R0535  1           INDEX$_INVINPUT   = DSRINDEX$_INVINPUT   %,
;  R0536  1           INDEX$_INVRECORD  = DSRINDEX$_INVRECORD  %,
;  R0537  1           INDEX$_LASTCONT   = DSRINDEX$_LASTCONT   %,
;  R0538  1           INDEX$_NOBEGIN    = DSRINDEX$_NOBEGIN    %,
;  R0539  1           INDEX$_NOEND      = DSRINDEX$_NOEND      %,
;  R0540  1           INDEX$_NOINDEX    = DSRINDEX$_NOINDEX    %,
;  R0541  1           INDEX$_NOLIST     = DSRINDEX$_NOLIST     %,
;  R0542  1           INDEX$_OVERSTRK   = DSRINDEX$_OVERSTRK   %,
;  R0543  1           INDEX$_SKIPPED    = DSRINDEX$_SKIPPED    %,
;  R0544  1           INDEX$_SYNTAX     = DSRINDEX$_SYNTAX     %,
;  R0545  1           INDEX$_TEXFILE    = DSRINDEX$_TEXFILE    %,
;  R0546  1           INDEX$_TOODEEP    = DSRINDEX$_TOODEEP    %,
;  R0547  1           INDEX$_TOOFEW     = DSRINDEX$_TOOFEW     %,
;  R0548  1           INDEX$_TRUNCATED  = DSRINDEX$_TRUNCATED  %,
;  R0549  1           INDEX$_COMPLETE   = DSRINDEX$_COMPLETE   %,
;  R0550  1           INDEX$_CREATED    = DSRINDEX$_CREATED    %,
;  R0551  1           INDEX$_IDENT      = DSRINDEX$_IDENT      %,
;  R0552  1           INDEX$_PROCFILE   = DSRINDEX$_PROCFILE   %,
;  R0553  1           INDEX$_TEXT       = DSRINDEX$_TEXT       %,
;  R0554  1           INDEX$_TEXTD      = DSRINDEX$_TEXTD      %,
;  R0555  1           INDEX$_TMS11      = DSRINDEX$_TMS11      %;
;  R0556  1
;  R0557  1      %FI
;  R0558  1
;  R0559  1      EXTERNAL LITERAL
;  R0560  1           INDEX$_BADLOGIC,     ! <internal logic error detected>
;  R0561  1           INDEX$_BADVALUE,     ! <'!AS' is an invalid keyword value>
;  R0562  1           INDEX$_INSVIRMEM,    ! <insufficient virtual memory>
;  R0563  1           INDEX$_LINELENG,     ! <maximum line length is 120>
;  R0564  1           INDEX$_NOREF,        ! <page reference not found>
;  R0565  1           INDEX$_OPENIN,       ! <error opening '!AS' for input>
;  R0566  1           INDEX$_OPENOUT,      ! <error opening '!AS' for output>
;  R0567  1           INDEX$_TOOMANY,      ! <too many values supplied>
;  R0568  1           INDEX$_VALERR,       ! <specified value is out of legal range>
;  R0569  1           INDEX$_CANTBAL,      ! <can't balance last page>
```

```
;  R0570  1          INDEX$_CLOSEQUOT,     ! <missing close quote>
;  R0571  1          INDEX$_CONFQUAL,      ! <conflicting qualifiers>
;  R0572  1          INDEX$_CTRLCHAR,      ! <the following line contains control characters - ignored>
;  R0573  1          INDEX$_DOESNTFIT,     ! <'!AD' will not fit at the current indentation level>
;  R0574  1          INDEX$_DUPBEGIN,      ! <duplicate .XPLUS (BEGIN) - inserted as .XPLUS ()>
;  R0575  1          INDEX$_EMPTYIN,       ! <empty input file '!AS'>
;  R0576  1          INDEX$_IGNORED,       ! <'!AS' ignored>
;  R0577  1          INDEX$_INVINPUT,      ! <invalid input file format in file '!AS'>
;  R0578  1          INDEX$_INVRECORD,     ! <invalid record type in file '!AS'>
;  R0579  1          INDEX$_LASTCONT,      ! <can't generate continuation heading on last page>
;  R0580  1          INDEX$_NOBEGIN,       ! <.XPLUS (END) with no .XPLUS (BEGIN) - inserted as .XPLUS ()>
;  R0581  1          INDEX$_NOEND,         ! <.XPLUS (BEGIN) has no corresponding .XPLUS (END)>
;  R0582  1          INDEX$_NOINDEX,       ! <no index information in file '!AS'>
;  R0583  1          INDEX$_NOLIST,        ! <parameter list not allowed>
;  R0584  1          INDEX$_OVERSTRK,      ! <the following line contains an overstrike sequence>
;  R0585  1          INDEX$_SKIPPED,       ! <!UL reference!%S inside page range - ignored>
;  R0586  1          INDEX$_SYNTAX,        ! <error parsing '!AS'>
;  R0587  1          INDEX$_TEXFILE,       ! <error processing line !UL of TEX character file '!AS'>
;  R0588  1          INDEX$_TOODEEP,       ! <maximum subindex depth exceeded>
;  R0589  1          INDEX$_TOOFEW,        ! <not enough values supplied>
;  R0590  1          INDEX$_TRUNCATED,     ! <string too long - truncated>
;  R0591  1          INDEX$_COMPLETE,      ! <processing complete '!AS'>
;  R0592  1          INDEX$_CREATED,       ! <'!AS' created>
;  R0593  1          INDEX$_IDENT,         ! <INDEX version !AD>
;  R0594  1          INDEX$_PROCFILE,      ! <processing file '!AS'>
;  R0595  1          INDEX$_TEXT,          ! <!AS>
;  R0596  1          INDEX$_TEXTD,         ! <entry text: '!AD'>
;  R0597  1          INDEX$_TMS11;         ! <output file full - continuing with file '!AS'>
;  R0598  1
```

```
;  103          0599  1
;  104          0600  1 %FI
;  105          0601  1
;  106          0602  1 SWITCHES LIST (NOREQUIRE);
;  107          0603  1
;  108          0604  1 !
;  109          0605  1 ! MACROS:
;  110          0606  1 !
;  111          0607  1 !
;  112          0608  1 ! EQUATED SYMBOLS·
;  113          0609  1 !
;  114          0610  1
;  115          0611  1 LITERAL
;  116          0612  1     TRUE = 1,
;  117          0613  1     FALSE = 0;
;  118          0614  1
;  119          0615  1 !
;  120          0616  1 ! OWN STORAGE:
;  121          0617  1 !
;  122          0618  1 !
;  123          0619  1 ! EXTERNAL REFERENCES:
;  124          0620  1 !
;  125          0621  1
;  126          0622  1 EXTERNAL LITERAL
;  127          0623  1     MAXLST;                                 ! Maximum subindex depth
;  128          0624  1
;  129          0625  1 EXTERNAL
;  130          0626  1     NDXPOL,                                 ! Address of indexing pool
;  131          0627  1     NDXSGE,                                 ! End of current segment.
;  132          0628  1     NDXSGF,
;  133          0629  1     BUCKET : $BUCKET_ARRAY [27, 27],
;  134          0630  1     LSTPTR : REF $XE_BLOCK,
;  135          0631  1     INDLVL,                                 ! Index level
;  136          0632  1     LSTSTK : VECTOR;                        ! Temporary entry stack
;  137          0633  1
;  138          0634  1 EXTERNAL ROUTINE
;  139          0635  1     ENTMSG : NOVALUE,
;  140          0636  1     GPOOL,
;  141          0637  1     XPOOL;
```

```
;  143          0638  1 %SBTTL 'XINIT -- Initialize index processor'
;  144          0639  1
;  145          0640  1 GLOBAL ROUTINE XINIT : NOVALUE =
;  146          0641  1
;  147          0642  1 !++
;  148          0643  1 ! FUNCTIONAL DESCRIPTION:
;  149          0644  1 !
;  150          0645  1 !       Initialize the index processor.
;  151          0646  1 !
;  152          0647  1 ! FORMAL PARAMETERS:
;  153          0648  1 !
;  154          0649  1 !       NONE
;  155          0650  1 !
;  156          0651  1 ! IMPLICIT INPUTS:
;  157          0652  1 !
;  158          0653  1 !       NONE
;  159          0654  1 !
;  160          0655  1 ! IMPLICIT OUTPUTS:
;  161          0656  1 !
;  162          0657  1 !       NDXSGE          - marker at end of work storage
;  163          0658  1 !
;  164          0659  1 ! ROUTINE VALUE:
;  165          0660  1 ! COMPLETION CODES:
;  166          0661  1 !
;  167          0662  1 !       NONE
;  168          0663  1 !
;  169          0664  1 ! SIDE EFFECTS:
;  170          0665  1 !
;  171          0666  1 !       One null entry is placed in each bucket.
;  172          0667  1 !
;  173          0668  1 !--
;  174          0669  1
;  175          0670  2     BEGIN
;  176          0671  2
;  177          0672  2     LOCAL
;  178          0673  2         TEMP_BUF : $XE_BLOCK;
;  179          0674  2
;  180          0675  2     GPOOL (NDXPOL, 1000);       ! Generate a pool with space for 1000 segments.
;  181          0676  2     NDXSGE = 0;                 ! Pool is empty, but allocated.
;  182          0677  2     NDXSGF = 0;                 ! ...
;  183          0678  2
;  184          0679  2     !
;  185          0680  2     ! Fill in constant parts of bucket entry
;  186          0681  2     !
;  187          0682  2     TEMP_BUF [XE$A_PREV] = 0;
;  188          0683  2     TEMP_BUF [XE$A_NEXT] = 0;
;  189          0684  2     TEMP_BUF [XE$A_SUBX] = 0;
;  190          0685  2     TEMP_BUF [XE$H_SUBC] = 0;
;  191          0686  2     TEMP_BUF [XE$V_BARS] = FALSE;
;  192          0687  2     TEMP_BUF [XE$A_REF] = 0;
;  193          0688  2     TEMP_BUF [XE$A_TEXT] = 0;
;  194          0689  2
;  195          0690  2     !
;  196          0691  2     ! Now place one entry in each bucket
;  197          0692  2     !
;  198          0693  2     INCR I FROM 0 TO 26 DO
;  199          0694  2         INCR J FROM 0 TO 26 DO
```

```
;  200          0695  2              BUCKET [.I, .J] = SAVDAT (TEMP_BUF, DS_X_ENTRY, XE$K_LENGTH);
;  201          0696  2
;  202          0697  1      END;                            !End of XINIT


;                                                                       .TITLE  NDXDAT NDXDAT - Index pool data manipulation ro
                                                                                utines
                                                                        .IDENT  \V04-000\

                                                                        .EXTRN  DSRINDEX$_BADLOGIC
                                                                        .EXTRN  DSRINDEX$_BADVALUE
                                                                        .EXTRN  DSRINDEX$_INSVIRMEM
                                                                        .EXTRN  DSRINDEX$_LINELENG
                                                                        .EXTRN  DSRINDEX$_NOREF
                                                                        .EXTRN  DSRINDEX$_OPENIN
                                                                        .EXTRN  DSRINDEX$_OPENOUT
                                                                        .EXTRN  DSRINDEX$_TOOMANY
                                                                        .EXTRN  DSRINDEX$_VALERR
                                                                        .EXTRN  DSRINDEX$_CANTBAL
                                                                        .EXTRN  DSRINDEX$_CLOSEQUOT
                                                                        .EXTRN  DSRINDEX$_CONFQUAL
                                                                        .EXTRN  DSRINDEX$_CTRLCHAR
                                                                        .EXTRN  DSRINDEX$_DOESNTFIT
                                                                        .EXTRN  DSRINDEX$_DUPBEGIN
                                                                        .EXTRN  DSRINDEX$_EMPTYIN
                                                                        .EXTRN  DSRINDEX$_IGNORED
                                                                        .EXTRN  DSRINDEX$_INVINPUT
                                                                        .EXTRN  DSRINDEX$_INVRECORD
                                                                        .EXTRN  DSRINDEX$_LASTCONT
                                                                        .EXTRN  DSRINDEX$_NOBEGIN
                                                                        .EXTRN  DSRINDEX$_NOEND
                                                                        .EXTRN  DSRINDEX$_NOINDEX
                                                                        .EXTRN  DSRINDEX$_NOLIST
                                                                        .EXTRN  DSRINDEX$_OVERSTRK
                                                                        .EXTRN  DSRINDEX$_SKIPPED
                                                                        .EXTRN  DSRINDEX$_SYNTAX
                                                                        .EXTRN  DSRINDEX$_TEXFILE
                                                                        .EXTRN  DSRINDEX$_TOODEEP
                                                                        .EXTRN  DSRINDEX$_TOOFEW
                                                                        .EXTRN  DSRINDEX$_TRUNCATED
                                                                        .EXTRN  DSRINDEX$_COMPLETE
                                                                        .EXTRN  DSRINDEX$_CREATED
                                                                        .EXTRN  DSRINDEX$_IDENT
                                                                        .EXTRN  DSRINDEX$_PROCFILE
                                                                        .EXTRN  DSRINDEX$_TEXT, DSRINDEX$_TEXTD
                                                                        .EXTRN  DSRINDEX$_TMS11
                                                                        .EXTRN  MAXLST, NDXPOL, NDXSGE
                                                                        .EXTRN  NDXSGF, BUCKET, LSTPTR
                                                                        .EXTRN  INDLVL, LSTSTK, ENTMSG
                                                                        .EXTRN  GPOOL, XPOOL

                                                                        .PSECT  $CODE$,NOWRT,2

                                                003C 00000              .ENTRY  XINIT, Save R2,R3,R4,R5                 ; 0640
5E                                            20  C2 00002              SUBL2   #32, SP                                 ;
7E                                  03E8      8F  3C 00005              MOVZWL  #1000, -(SP)                            ; 0675
   00000000G                                  EF  9F 0000A              PUSHAB  NDXPOL                                  ;
```

```
                          00000000G  EF                 02  FB 00010          CALLS   #2, GPOOL                                ;
                                                00000000G  EF  D4 00017          CLRL    NDXSGE                                   ; 0676
                                                00000000G  EF  D4 0001D          CLRL    NDXSGF                                   ; 0677
                                                           6E  7C 00023          CLRQ    TEMP_BUF                                 ; 0682
      18   AE              11                    00         00  F0 00025          INSV    #0, #0, #17, TEMP_BUF+26                 ; 0686
                                                      08   AE  7C 0002B          CLRQ    TEMP_BUF+8                               ; 0684
                                                      10   AE  D4 0002E          CLRL    TEMP_BUF+16                              ; 0688
                                                           52  D4 00031          CLRL    I                                        ; 0693
                          55                    52         1B  C5 00033 1$:      MULL3   #27, I, R5                               ; 0695
                                                           54  D4 00037          CLRL    J                                        ;
                          53                    55         54  C1 00039 2$:      ADDL3   J, R5, R3                                ;
                                                           08  DD 0003D          PUSHL   #8                                       ;
                                                           08  DD 0003F          PUSHL   #8                                       ;
                                                      08   AE  9F 00041          PUSHAB  TEMP_BUF                                 ;
                          00000000V  EF                    03  FB 00044          CALLS   #3, SAVDAT                               ;
                          00000000GEF43                    50  D0 0004B          MOVL    R0, BUCKET[R3]                           ;
                          E2                    54         1A  F3 00053          AOBLEQ  #26, J, 2$                               ;
                          D8                    52         1A  F3 00057          AOBLEQ  #26, I, 1$                               ; 0694
                                                           04  0005B          RET                                              ; 0697

; Routine Size:  92 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  204     0698  1 %SBTTL 'SAVDAT -- Place data in working storage'
;  205     0699  1
;  206     0700  1 GLOBAL ROUTINE SAVDAT (ADR, DATASET, SIZE) =
;  207     0701  1
;  208     0702  1 !++
;  209     0703  1 ! FUNCTIONAL DESCRIPTION:
;  210     0704  1 !
;  211     0705  1 !       Place data item into work space.
;  212     0706  1 !       Note that in this version all indexing information is
;  213     0707  1 !       saved in core.  This routine could be replaced with
;  214     0708  1 !       an interface to a work-file system on -11s.
;  215     0709  1 !
;  216     0710  1 ! FORMAL PARAMETERS:
;  217     0711  1 !
;  218     0712  1 !       ADR     - Address of data to be saved (CH$PTR if DATASET is DS_X_STRING)
;  219     0713  1 !       DATASET - Name of dataset where data will be stored
;  220     0714  1 !       SIZE    - Number of characters (used only by DS_X_STRING)
;  221     0715  1 !
;  222     0716  1 ! IMPLICIT INPUTS:
;  223     0717  1 !
;  224     0718  1 !       NONE
;  225     0719  1 !
;  226     0720  1 ! IMPLICIT OUTPUTS:
;  227     0721  1 !
;  228     0722  1 !       NONE
;  229     0723  1 !
;  230     0724  1 ! ROUTINE VALUE:
;  231     0725  1 ! COMPLETION CODES:
;  232     0726  1 !
;  233     0727  1 !       Work space ADDRESS of data is returned.
;  234     0728  1 !
;  235     0729  1 ! SIDE EFFECTS:
;  236     0730  1 !
;  237     0731  1 !       Work space pointers are updated
;  238     0732  1 !
;  239     0733  1 !--
;  240     0734  1
;  241     0735  2     BEGIN
;  242     0736  2
;  243     0737  2     MAP
;  244     0738  2         ADR : REF VECTOR;
;  245     0739  2
;  246     0740  2     LOCAL
;  247     0741  2         WSADDR : REF VECTOR;
;  248     0742  2
;  249     0743  2     !
;  250     0744  2     ! Handle strings differently from fixed length data
;  251     0745  2     !
;  252     0746  2     IF .DATASET NEQ DS_X_STRING
;  253     0747  2     THEN
;  254     0748  3         BEGIN
;  255     0749  3         !
;  256     0750  3         ! Fixed length data
;  257     0751  3         !
;  258     0752  3         ! Make sure there's sufficient dynamic memory to accommodate this entry
;  259     0753  3         !
;  260     0754  3         XMEM (.DATASET);
```

```
;  261          0755  3
;  262          0756  3          !
;  263          0757  3          ! Point to where data will be placed (at end of current segment).
;  264          0758  3          !
;  265          0759  3          WSADDR = .NDXSGE;
;  266          0760  3
;  267          0761  3          !
;  268          0762  3          ! Advance over allocated storage
;  269          0763  3          !
;  270          0764  3          NDXSGE = .NDXSGE + .DATASET * %UPVAL;    ! Point to next free memory in segment
;  271          0765  3          NDXSGF = .NDXSGF - .DATASET;             ! Count off memory used.
;  272          0766  3
;  273          0767  3          INCR I FROM 0 TO .DATASET - 1 DO
;  274          0768  3              WSADDR [.I] = .ADR [.I];
;  275          0769  3
;  276          0770  3          END
;  277          0771  2      ELSE
;  278          0772  3          BEGIN
;  279          0773  3          !
;  280          0774  3          ! Strings - variable length data
;  281          0775  3          !
;  282          0776  3          LOCAL
;  283          0777  3              SRC_PTR,
;  284          0778  3              WRD_PTR;
;  285          0779  3
;  286          0780  3          !
;  287          0781  3          ! Ensure there's sufficient memory to accommodate this.
;  288          0782  3          !
;  289          0783  3          XMEM (CH$ALLOCATION (.SIZE) + 1);
;  290          0784  3
;  291          0785  3          !
;  292          0786  3          ! Point to where data will be placed (at end of current segment).
;  293          0787  3          !
;  294          0788  3          WSADDR = .NDXSGE;
;  295          0789  3
;  296          0790  3          !
;  297          0791  3          ! Advance over allocated storage
;  298          0792  3          !
;  299          0793  3          NDXSGE = .NDXSGE + (CH$ALLOCATION (.SIZE) + 1) * %UPVAL;     ! Point past used memory.
;  300          0794  3          NDXSGF = .NDXSGF - (CH$ALLOCATION (.SIZE) + 1);              ! Count off used memory.
;  301          0795  3
;  302          0796  3          !
;  303          0797  3          ! Set up string pointers
;  304          0798  3          !
;  305          0799  3          SRC_PTR = .ADR;                          ! ADR is a CH$PTR to string
;  306          0800  3          WRD_PTR = CH$PTR (WSADDR [1]);           ! String starts after 1st fullword
;  307          0801  3
;  308          0802  3          !
;  309          0803  3          ! Save length as the first fullword of string
;  310          0804  3          !
;  311          0805  3          WSADDR [0] = .SIZE;
;  312          0806  3
;  313          0807  3          !
;  314          0808  3          ! Store the string
;  315          0809  3          !
;  316          0810  3          INCR I FROM 1 TO .SIZE DO
;  317          0811  3              CH$WCHAR_A (CH$RCHAR_A (SRC_PTR), WRD_PTR);
```

```
;  318          0812  3
;  319          0813  2          END;
;  320          0814  2
;  321          0815  2      RETURN .WSADDR;
;  322          0816  1      END;                                .End of SAVDAT



                                                00FC 00000          .ENTRY  SAVDAT, Save R2,R3,R4,R5,R6,R7          ; 0700
                                    57 00000000V    EF   9E 00002   MOVAB   XMEM, R7                                ;
                                    56 00000000G    EF   9E 00009   MOVAB   NDXSGF, R6                              ;
                                    55 00000000G    EF   9E 00010   MOVAB   NDXSGE, R5                              ;
                                    53        08    AC   D0 00017   MOVL    DATASET, R3                             ; 0746
                                                    20   13 0001B   BEQL    3$                                      ;
                                                    53   DD 0001D   PUSHL   R3                                      ; 0754
                                    67              01   FB 0001F   CALLS   #1, XMEM                                ;
                                    52              65   D0 00022   MOVL    NDXSGE, WSADDR                          ; 0759
                                    75          9543     DE 00025   MOVAL   @NDXSGE[R3], NDXSGE                     ; 0764
                                    66              53   C2 00029   SUBL2   R3, NDXSGF                             ; 0765
                                    50              01   CE 0002C   MNEGL   #1, I                                  ; 0768
                                                    06   11 0002F   BRB     2$                                      ;
                                  6240        04 BC40   D0 00031 1$:  MOVL    @ADR[I], (WSADDR)[I]                    ;
                              F6    50              53   F2 00037 2$:  AOBLSS  R3, I, 1$                           ;
                                                    36   11 0003B   BRB     6$                                      ; 0746
                              53        0C    AC      03   C1 0003D 3$:  ADDL3   #3, SIZE, R3                       ; 0783
                                    53              04   C6 00042   DIVL2   #4, R3                                  ;
                                    54        01    A3   9E 00045   MOVAB   1(R3), R4                               ;
                                                    54   DD 00049   PUSHL   R4                                      ;
                                    67              01   FB 0004B   CALLS   #1, XMEM                                ;
                                    52              65   D0 0004E   MOVL    NDXSGE, WSADDR                          ; 0788
                                    75          9543     DE 00051   MOVAL   @NDXSGE[R3], NDXSGE                     ; 0793
                                    65              04   C0 00055   ADDL2   #4, NDXSGE                             ;
                                    66              54   C2 00058   SUBL2   R4, NDXSGF                             ; 0794
                                    53        04    AC   D0 0005B   MOVL    ADR, SRC_PTR                            ; 0799
                                    50        04    A2   9E 0005F   MOVAB   4(R2), WRD_PTR                         ; 0800
                                    62        0C    AC   D0 00063   MOVL    SIZE, (WSADDR)                          ; 0805
                                                    51   D4 00067   CLRL    I                                       ; 0811
                                                    03   11 00069   BRB     5$                                      ;
                                    80              83   90 0006B 4$:  MOVB    (SRC_PTR)+, (WRD_PTR)+                  ;
                                F8  51        0C    AC   F3 0006E 5$:  AOBLEQ  SIZE, I, 4$                         ;
                                    50              52   D0 00073 6$:  MOVL    WSADDR, R0                              ; 0815
                                                    04   00076        RET                                          ; 0816

; Routine Size:  119 bytes,    Routine Base:  $CODE$ + 005C
```

```
;  324          0817  1 %SBTTL 'XMEM -- Verify availability of dynamic memory'
;  325          0818  1 ROUTINE XMEM (AMOUNT) : NOVALUE =
;  326          0819  1 !++
;  327          0820  1 !
;  328          0821  1 ! FUNCTIONAL DESCRIPTION:
;  329          0822  1 !
;  330          0823  1 !       This routine verifies that the requested amount of memory is
;  331          0824  1 !       available. If not enough memory is available, it attempts to
;  332          0825  1 !       allocate more. If this fails, a fatal message is issued
;  333          0826  1 !       forcing program termination.
;  334          0827  1 !
;  335          0828  1 ! FORMAL PARAMETERS:
;  336          0829  1 !
;  337          0830  1 !       AMOUNT  - Amount of memory desired
;  338          0831  1 !
;  339          0832  1 ! IMPLICIT INPUTS:
;  340          0833  1 !
;  341          0834  1 !       NDXSGE  - End of current segment
;  342          0835  1 !       NDXSGF  - Amount of free memory in current segment
;  343          0836  1 !
;  344          0837  1 ! IMPLICIT OUTPUTS:
;  345          0838  1 !
;  346          0839  1 !       NDXSGE  - End of new segment if more memory is allocated
;  347          0840  1 !       NDXSGF  - Amount of free memory if more is alloacted
;  348          0841  1 !
;  349          0842  1 ! ROUTINE VALUE:
;  350          0843  1 ! COMPLETION CODES:
;  351          0844  1 !
;  352          0845  1 !       None
;  353          0846  1 !
;  354          0847  1 ! SIDE EFFECTS:
;  355          0848  1 !
;  356          0849  1 !       None
;  357          0850  1 !--
;  358          0851  2     BEGIN
;  359          0852  2
;  360          0853  2     IF .NDXSGF LSS .AMOUNT
;  361          0854  2     THEN
;  362          0855  3         BEGIN
;  363          0856  3         !
;  364          0857  3         ! Not enough space in current segment, so get a new one.
;  365          0858  3         !
;  366          0859  3         NDXSGE = XPOOL (NDXPOL, 1000);  ! Try to get about 1k.
;  367          0860  3
;  368          0861  3         IF .NDXSGE NEQ 0
;  369          0862  3         THEN
;  370          0863  3             !
;  371          0864  3             ! The requested amount was available.
;  372          0865  3             ! Save size of segment
;  373          0866  3             !
;  374          0867  3             NDXSGF = 1000
;  375          0868  3         ELSE
;  376          0869  3             !
;  377          0870  3             ! The requested amount could not be allocated (pool full)
;  378          0871  3             !
;  379          0872  3
;  380        L 0873  3 %IF %BLISS (BLISS32)
```

```
;  381          0874  3 %THEN                                          ! Signal error for BLISS32
;  382          0875  3
;  383          0876  3             SIGNAL_STOP (INDEX$_INSVIRMEM);
;  384          0877  3
;  385        U 0878  3 %ELSE                                          ! Use $XPO_PUT_MSG otherwise
;  386        U 0879  3
;  387        U 0880  3             $XPO_PUT_MSG (SEVERITY = FATAL,
;  388        U 0881  3                 STRING = 'can''t extend index pool.');
;  389        U 0882  3
;  390          0883  3 %FI
;  391          0884  3
;  392          0885  2         END;
;  393          0886  1     END;


                                        0004 00000 XMEM:    .WORD   Save R2                                  ; 0818
                          52 00000000G  EF   9E 00002       MOVAB   NDXSGF, R2
                       04 AC            62   D1 00009       CMPL    NDXSGF, AMOUNT                          ; 0853
                                        2E   18 0000D       BGEQ    2$
                          7E     03E8    8F   3C 0000F       MOVZWL  #1000, -(SP)                            ; 0859
                             00000000G  EF   9F 00014       PUSHAB  NDXPOL
                 00000000G  EF          02   FB 0001A       CALLS   #2, XPOOL
                 00000000G  EF          50   D0 00021       MOVL    R0, NDXSGE
                                        06   13 00028       BEQL    1$                                      ; 0861
                          62     03E8    8F   3C 0002A       MOVZWL  #1000, NDXSGF                           ; 0867
                                        04 0002F           RET
                             00000000G  8F   DD 00030 1$:    PUSHL   #DSRINDEX$_INSVIRMEM                    ; 0876
                 00000000G  00          01   FB 00036       CALLS   #1, LIB$STOP
                                        04 0003D 2$:       RET                                             ; 0886

; Routine Size:  62 bytes,    Routine Base:  $CODE$ + 00D3
```

```
;  395          0887  1 %SBTTL 'GENTRY -- from working storage'
;  396          0888  1 GLOBAL ROUTINE GENTRY (B_ROW, B_COL, NO_SUBX) =
;  397          0889  1
;  398          0890  1 !++
;  399          0891  1 ! FUNCTIONAL DESCRIPTION:
;  400          0892  1 !
;  401          0893  1 !       Get a data entry from the working storage space.
;  402          0894  1 !
;  403          0895  1 ! FORMAL PARAMETERS:
;  404          0896  1 !
;  405          0897  1 !       B_ROW   - bucket ROW
;  406          0898  1 !       B_COL   - bucket COLumn
;  407          0899  1 !       NO_SUBX - If true, ignore sub-index while advancing
;  408          0900  1 !
;  409          0901  1 ! IMPLICIT INPUTS:
;  410          0902  1 !
;  411          0903  1 !       INDLVL  - subindex level
;  412          0904  1 !       LSTPTR - contains current position in list
;  413          0905  1 !
;  414          0906  1 ! IMPLICIT OUTPUTS:
;  415          0907  1 !
;  416          0908  1 !       INDLVL   - indentation level
;  417          0909  1 !       LSTPTR - updated position in list.
;  418          0910  1 !       LSTSTK   - temporary stack for saving sub-index lists
;  419          0911  1 !
;  420          0912  1 ! ROUTINE VALUE:
;  421          0913  1 ! COMPLETION CODES:
;  422          0914  1 !
;  423          0915  1 !       TRUE    - entry found
;  424          0916  1 !       FALSE   - No more entries left in list
;  425          0917  1 !
;  426          0918  1 ! SIDE EFFECTS:
;  427          0919  1 !
;  428          0920  1 !       Each time called, the pointer is advanced along the chain of entries
;  429          0921  1 !       in the list.
;  430          0922  1 !
;  431          0923  1 !--
;  432          0924  1
;  433          0925  2     BEGIN
;  434          0926  2
;  435          0927  2     LOCAL
;  436          0928  2         L_PTR : REF $XE_BLOCK;
;  437          0929  2
;  438          0930  2     IF .LSTPTR EQL 0
;  439          0931  2     THEN
;  440          0932  3         BEGIN
;  441          0933  3         !
;  442          0934  3         ! Head of list processing
;  443          0935  3         !
;  444          0936  3         LSTPTR = .BUCKET [.B_ROW, .B_COL];
;  445          0937  3
;  446          0938  3         IF .LSTPTR [XE$A_TEXT] EQL 0 THEN RETURN FALSE;
;  447          0939  3         END
;  448          0940  2     ELSE
;  449          0941  3         BEGIN
;  450          0942  3         !
;  451          0943  3         ! Processing for all other list elements
```

```
;  452          0944  3          !
;  453          0945  3          L_PTR = .LSTPTR;
;  454          0946  3
;  455          0947  3          !
;  456          0948  3          ! Look for a sub-list entry
;  457          0949  3          !
;  458          0950  4          IF (.L_PTR [XE$A_SUBX] NEQ 0) AND (NOT .NO_SUBX)
;  459          0951  3          THEN
;  460          0952  4              BEGIN
;  461          0953  4              IF .INDLVL LSS MAXLST
;  462          0954  4              THEN
;  463          0955  5                  BEGIN
;  464          0956  5                  !
;  465          0957  5                  ! Remember where we left off by placing top of sub-list on the stack
;  466          0958  5                  !
;  467          0959  5                  LSTSTK [.INDLVL] = .LSTPTR;
;  468          0960  5
;  469          0961  5                  !
;  470          0962  5                  ! Point to sub-list
;  471          0963  5                  !
;  472          0964  5                  LSTPTR = .L_PTR [XE$A_SUBX];
;  473          0965  5
;  474          0966  5                  !
;  475          0967  5                  ! Keep track of indentation level
;  476          0968  5                  !
;  477          0969  5                  INDLVL = .INDLVL + 1;
;  478          0970  5
;  479          0971  5                  RETURN TRUE;
;  480          0972  5                  END
;  481          0973  4              ELSE
;  482          0974  5                  BEGIN
;  483          0975  5                  !
;  484          0976  5                  ! Index entry is too deep
;  485          0977  5                  !
;  486          0978  5
;  487        L 0979  5 %IF %BLISS (BLISS32)
;  488          0980  5 %THEN                                           ! Signal error for BLISS32
;  489          0981  5
;  490          0982  5                  SIGNAL (INDEX$_TOODEEP);
;  491          0983  5
;  492        U 0984  5 %ELSE                                           ! Use $XPO_PUT_MSG otherwise
;  493        U 0985  5
;  494        U 0986  5                  $XPO_PUT_MSG (SEVERITY = WARNING, STRING = 'Maximum subindex depth exceeded');
;  495        U 0987  5
;  496          0988  5 %FI
;  497          0989  5
;  498          0990  5                  ENTMSG ();                     ! Dump entry
;  499          0991  4                  END;
;  500          0992  3              END;
;  501          0993  3
;  502          0994  3          !
;  503          0995  3          ! Check for end of list
;  504          0996  3          !
;  505          0997  3          IF .L_PTR [XE$A_NEXT] NEQ 0
;  506          0998  3          THEN
;  507          0999  3              LSTPTR = .L_PTR [XE$A_NEXT]
;  508          1000  3          ELSE
```

```
;  509          1001  3              RETURN FALSE;
;  510          1002  3
;  511          1003  3          !
;  512          1004  3          ! Look out for entries on a sub-list
;  513          1005  3          !
;  514          1006  3          IF .LSTPTR [XE$A_NEXT] EQL 0
;  515          1007  3          THEN
;  516          1008  4              BEGIN
;  517          1009  4              !
;  518          1010  4              ! See if we need to unstack a sub-list
;  519          1011  4              !
;  520          1012  4              IF .INDLVL EQL 0
;  521          1013  4              THEN
;  522          1014  4                  !
;  523          1015  4                  ! Nothing to unstack, quit
;  524          1016  4                  !
;  525          1017  4                  RETURN FALSE
;  526          1018  4              ELSE
;  527          1019  5                  BEGIN
;  528          1020  5                  !
;  529          1021  5                  ! Unstack branch of tree
;  530          1022  5                  !
;  531          1023  5                  INDLVL = .INDLVL - 1;
;  532          1024  5                  LSTPTR = .LSTSTK [.INDLVL];
;  533          1025  5                  RETURN GENTRY (.B_ROW, .B_COL, TRUE);
;  534          1026  4                  END;
;  535          1027  3              END;
;  536          1028  2          END;
;  537          1029  2
;  538          1030  2      RETURN TRUE;
;  539          1031  1      END;                            !End of GENTRY


                                        003C 00000          .ENTRY  GENTRY, Save R2,R3,R4,R5                ; 0888
                              55 00000000G  EF   9E 00002   MOVAB   LSTSTK, R5                              ;
                              54 00000000G  EF   9E 00009   MOVAB   INDLVL, R4                              ;
                              53 00000000G  EF   9E 00010   MOVAB   LSTPTR, R3                              ;
                              51            63   D0 00017   MOVL    LSTPTR, R1                              ; 0930
                                            1B   12 0001A   BNEQ    1$                                      ;
                        50    04    AC      1B   C5 0001C   MULL3   #27, B_ROW, R0                         ; 0936
                              50          08  AC   C0 00021   ADDL2   B_COL, R0                             ;
                              63 00000000GEF40   D0 00025   MOVL    BUCKET[R0], LSTPTR                      ;
                              50            63   D0 0002D   MOVL    LSTPTR, R0                              ; 0938
                                      10    A0   D5 00030   TSTL    16(R0)                                  ;
                                            64   12 00033   BNEQ    4$                                      ;
                                            66   11 00035   BRB     5$                                      ;
                              52            51   D0 00037 1$: MOVL    R1, L_PTR                           ; 0945
                                      08    A2   D5 0003A   TSTL    8(L_PTR)                                ; 0950
                                            30   13 0003D   BEQL    3$                                      ;
                              2C          0C  AC   E8 0003F   BLBS    NO_SUBX, 3$                             ;
                              50            64   D0 00043   MOVL    INDLVL, R0                              ; 0953
                  00000000G   8F            50   D1 00046   CMPL    R0, #MAXLST                             ;
                                            0C   18 0004D   BGEQ    2$                                      ;
                        6540                51   D0 0004F   MOVL    R1, LSTSTK[R0]                          ; 0959
```

```
                                63        08    A2  D0 00053          MOVL    8(L_PTR), LSTPTR                ; 0964
                                64        D6 00057                   INCL    INDLVL                          ; 0969
                                3E        11 00059                   BRB     4$                              ; 0971
                             00000000G    8F  DD 0005B 2$:           PUSHL   #DSRINDEX$_TOODEEP              ; 0982
                    00000000G   00        01  FB 00061              CALLS   #1, LIB$SIGNAL
                    00000000G   EF              00  FB 00068          CALLS   #0, ENTMSG                      ; 0990
                                      04    A2  D5 0006F 3$:          TSTL    4(L_PTR)                        ; 0997
                                29        13 00072                   BEQL    5$
                                63        04  A2  D0 00074          MOVL    4(L_PTR), LSTPTR                ; 0999
                                50        63  D0 00078              MOVL    LSTPTR, R0                      ; 1006
                                      04    A0  D5 0007B              TSTL    4(R0)
                                19        12 0007E                   BNEQ    4$
                                64        D5 00080                   TSTL    INDLVL                          ; 1012
                                19        13 00082                   BEQL    5$
                                64        D7 00084                   DECL    INDLVL                          ; 1023
                                50        64  D0 00086              MOVL    INDLVL, R0                      ; 1024
                                63          6540  D0 00089          MOVL    LSTSTK[R0], LSTPTR
                                01        DD 0008D                   PUSHL   #1                              ; 1025
                                7E        04  AC  7D 0008F          MOVQ    B_ROW, -(SP)
                          FF68  CF              03  FB 00093          CALLS   #3, GENTRY
                                                04 00098            RET                                     ; 1019
                                50        01  D0 00099 4$:          MOVL    #1, R0                          ; 1030
                                                04 0009C            RET
                                50        D4 0009D 5$:              CLRL    R0                              ; 1031
                                                04 0009F            RET

; Routine Size:  160 bytes,    Routine Base:  $CODE$ + 0111

;  540          1032  1
;  541          1033  1 END                                ! End of module
;  542          1034  0 ELUDOM


                                                                    .EXTRN  LIB$SIGNAL, LIB$STOP

;                                  PSECT SUMMARY

;      Name                      Bytes                          Attributes

;  $CODE$                          433  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        29         4       252          00:00.1
```

```
;                                       COMMAND QUALIFIERS

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

; Size:         433 code + 0 data bytes
; Run Time:        00:16.6
; Elapsed Time:    00:33.5
; Lines/CPU Min:    3744
; Lexemes/CPU-Min: 42955
; Memory Used:  100 pages
; Compilation Complete
```