

.....

```

NN      NN  DDDDDDDD  XX      XX  000000  UU      UU  TTTTTTTTTT
NN      NN  DDDDDDDD  XX      XX  000000  UU      UU  TTTTTTTTTT
NN      NN  DD      DD  XX      XX  00      00  UU      UU  TT
NN      NN  DD      DD  XX      XX  00      00  UU      UU  TT
NNNN    NN  DD      DD  XX      XX  00      00  UU      UU  TT
NNNN    NN  DD      DD  XX      XX  00      00  UU      UU  TT
NN      NN  DD      DD  XX      XX  00      00  UU      UU  TT
NN      NN  DD      DD  XX      XX  00      00  UU      UU  TT
NN      NN  DD      DD  XX      XX  00      00  UU      UU  TT
NN      NN  DD      DD  XX      XX  00      00  UU      UU  TT
NN      NN  DD      DD  XX      XX  00      00  UU      UU  TT
NN      NN  DDDDDDDD  XX      XX  000000  UUUUUUUUUU  TT
NN      NN  DDDDDDDD  XX      XX  000000  UUUUUUUUUU  TT

```

```

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

```

```
0001 0 %TITLE 'NDXOUT -- Sort and store index entries'
0002 0 MODULE NDXOUT (IDENT = 'V04-000'
0003 0          %BLISS32 [, ADDRESSING_MODE (EXTERNAL = LONG_RELATIVE, NONEXTERNAL = LONG_RELATIVE)]
0004 0          ) =
0005 1 BEGIN
0006 1
0007 1
0008 1 *****
0009 1 *
0010 1 *   COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0011 1 *   DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0012 1 *   ALL RIGHTS RESERVED.
0013 1 *
0014 1 *   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0015 1 *   ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0016 1 *   INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0017 1 *   COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0018 1 *   OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0019 1 *   TRANSFERRED.
0020 1 *
0021 1 *   THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0022 1 *   AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0023 1 *   CORPORATION.
0024 1 *
0025 1 *   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0026 1 *   SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0027 1 *
0028 1 *****
0029 1 *****
0030 1
0031 1 ++
0032 1 FACILITY:
0033 1   DSR (Digital Standard RUNOFF) /DSKPLUS DSRINDEX/INDEX Utility
0034 1
0035 1 ABSTRACT:
0036 1   The routines contained in this module sort and store
0037 1   index entries. This module is part of INDEX and was
0038 1   adopted from the TCX module XOUT.
0039 1
0040 1
0041 1 ENVIRONMENT: Transportable
0042 1
0043 1 AUTHOR: JPK
0044 1
0045 1 MODIFIED BY:
0046 1
0047 1   007 JPK00018 09-Mar-1983
0048 1   Modified INDEX to handle new BRN format.
0049 1   Modified NDXOUT to handle specifiable levels on SORT= string.
0050 1   Modified NDXFMT to output new RUNOFF prologue.
0051 1   Modified NDXPAG to output new TMS prologue and RUNOFF epilogue.
0052 1
0053 1   006 JPK00015 04-Feb-1983
0054 1   Cleaned up module names, modified revision history to
0055 1   conform with established standards. Updated copyright dates.
0056 1
0057 1   005 JPK00012 24-Jan-1983
```

NDXOUT
V04-000

NDXOUT -- Sort and store index entries

K 11
16-Sep-1984 01:04:24 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:07:15 [RUNOFF.SRC]NDXOUT.BLI;1

Page 2
(1)

:	58	0058	1	:	Modified NDXVMSMSG.MSG to define error messages for both
:	59	0059	1	:	DSRINDEX and INDEX.
:	60	0060	1	:	Added require of NDXVMSREQ.R32 to NDXOUT, NDXFMT, NDXDAT,
:	61	0061	1	:	INDEX, NDXMSG, NDXXTN, NDXTMS, NDXVMS and NDXPAG for BLISS32.
:	62	0062	1	:	Since this file defines the error message literals,
:	63	0063	1	:	the EXTERNAL REFERENCES for the error message literals
:	64	0064	1	:	have been removed.
:	65	0065	1	:	
:	66	0066	1	004	JP00010 24-Jan-1983
:	67	0067	1	:	Removed routines GETDAT and UPDDAT from NDXDAT - they
:	68	0068	1	:	performed no useful function. Removed references to these
:	69	0069	1	:	routines from NDXOUT, NDXFMT, and NDXMSG.
:	70	0070	1	:	Removed reference to XPOOL in NDXOUT - not used.
:	71	0071	1	:	
:	72	0072	1	003	JP00009 24-Jan-1983
:	73	0073	1	:	Modified to enhance performance. The sort buckets have each
:	74	0074	1	:	been divided into 27 sub-buckets; 1 for each letter and 1
:	75	0075	1	:	for non-alphas. Removed reference to BUCKET from INDEX.
:	76	0076	1	:	Definition of the structure was added to NDXPOL. References
:	77	0077	1	:	to BUCKET were changed in modules NDXOUT, NDXINI, NDXFMT
:	78	0078	1	:	and NDXDAT.
:	79	0079	1	:	
:	80	0080	1	002	JP00004 24-Sep-1982
:	81	0081	1	:	Modified NDXOUT, NDXMSG, NDXFMT, and NDXDAT for TOPS-20.
:	82	0082	1	:	Strings stored in the index pool use the first fullword
:	83	0083	1	:	for their length. References to these strings were incorrect.
:	84	0084	1	:	
:	85	0085	1	:-	

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
Declarations

L 11
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 3
(2)

```
: 87      0086 1 %SBTTL 'Declarations'
: 88      0087 1
: 89      0088 1 | TABLE OF CONTENTS:
: 90      0089 1 |
: 91      0090 1 |
: 92      0091 1 FORWARD ROUTINE
: 93      0092 1     XOUT      : NOVALUE,
: 94      0093 1     SORT_AS   : NOVALUE,
: 95      0094 1     FIND_POS   : NOVALUE,
: 96      0095 1     FIND_BUCKET,
: 97      0096 1     INSERT_INX : NOVALUE,
: 98      0097 1     INSERT_REF,
: 99      0098 1     ENTRY_CMP,
: 100     0099 1     STRG_CMP,
: 101     0100 1     CHRCMP      : NOVALUE;
: 102     0101 1
: 103     0102 1 |
: 104     0103 1 | INCLUDE FILES:
: 105     0104 1 |
: 106     0105 1 |
: 107     0106 1 LIBRARY 'NXPORT:XPORT';
: 108     0107 1
: 109     0108 1 SWITCHES LIST (REQUIRE);
: 110     0109 1
: 111     0110 1 REQUIRE 'REQ:NDXCLI';
```

```
| Put away index item
| Build sort string
| Locate position for insertion
| Locate bucket for insertion
| Insert index item into list
| Insert page reference into list
| Compare new entry with current entry
| Compare two strings
| Compare two characters in internal format
```

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
Declarations

M 11
16-Sep-1984 01:04:24
15-Sep-1984 22:53:19

VAX-11 Bliss-32 V4.0-742
_S255\$DUA28:[RUNOFF.SRC]NDXCL:REQ;1
Page 4
(1)

IDENT = 0V04-00004

```
*****
*
*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*  ALL RIGHTS RESERVED.
*
*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
*  TRANSFERRED.
*
*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
*  CORPORATION.
*
*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
*****
```

```
++
FACILITY:
  DSR (Digital Standard RUNOFF) /DSRPLUS DSRINDEX/INDEX Utility
```

ABSTRACT: INDEX command line definitions

ENVIRONMENT: Transportable

AUTHOR: JPK

CREATION DATE: January 1982

MODIFIED BY:

004 JPK00015 C4-Feb-1983
Cleaned up module names, modified revision history to
conform with established standards. Updated copyright dates.

003 JPK00011 24-Jan-1983
Changed CMDBLK [NDX\$G_LEVEL] to CMDBLK [NDX\$H_LEVEL]
Changed CMDBLK [NDX\$H_FORMAT] to CMDBLK [NDX\$H_LAYOUT]
Changed CMDBLK [NDX\$V_TMS11] and CMDBLK [NDX\$V_TEX] to CMDBLK [NDX\$H_FORMAT]
Changed comparisons of (.CHRS12 EQLA CHRS2A) to
(.CMDBLK [NDX\$H_FORMAT] EQL TMS11 A).
Definitions were changed in NDXCLI and references to the
effected fields were changed in NDXPAG, NDXFMT, INDEX, NDXVMS
and NDXCLIDMP.

002 RER00002 20-Jan-1983
Modified VMS command line interface module NDXVMS:
- changed /FORMAT qualifier to /LAYOUT.

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
Declarations

N 11
16-Sep-1984 01:04:24
15-Sep-1984 22:53:19

VAX-11 Bliss-32 V4.0-742
_S255SDUA28:[RUNOFF.SRC]NDXCLI.REQ;1 Page 5
(1)

: R0168 1
: R0169 1
: R0170 1
: R0171 1
: R0172 1
: R0173 1
: R0174 1
: R0175 1
: R0176 1
: R0177 1

:
:
:
:
:
:
:
:
:--

- changed use of /RESERVE and /REQUIRE for DSRPLUS.
- added code for new DSRPLUS qualifiers /FORMAT and
/TELLTALE HEADINGS.
Added fields to NDXCLI for new qualifiers: NDX\$V_TELLTALE
and NDX\$V_TEX.
Conditionalized output of NDX\$V_PAGE_MERGE in NDXCLIDMP to
account for different DSR and DSRPLUS default values.

NE
VC

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
Declarations

B 12

16-Sep-1984 01:04:24
15-Sep-1984 22:53:19

VAX-11 Bliss-32 V4.0-742
_S255SDUA28:[RUNOFF.SRC]NDXCLI.REQ;1

Page 6
(2)

ND
V04

```

R0178 1  |
R0179 1  |
R0180 1  |
R0181 1  |
R0182 1  |
R0183 1  |
R0184 1  |
R0185 1  |
R0186 1  |
R0187 1  |
R0188 1  |
R0189 1  |
R0190 1  |
R0191 1  |
R0192 1  |
R0193 1  |
R0194 1  |
R0195 1  |
R0196 1  |
R0197 1  |
R0198 1  |
R0199 1  |
R0200 1  |
R0201 1  |
R0202 1  |
R0203 1  |
R0204 1  |
R0205 1  |
R0206 1  |
R0207 1  |
R0208 1  |
R0209 1  |
R0210 1  |
R0211 1  |
R0212 1  |
R0213 1  |
R0214 1  |
R0215 1  |
R0216 1  |
R0217 1  |
R0218 1  |
R0219 1  |
R0220 1  |
R0221 1  |
R0222 1  |
R0223 1  |
R0224 1  |
R0225 1  |
R0226 1  |
R0227 1  |
R0228 1  |
R0229 1  |
R0230 1  |
R0231 1  |
R0232 1  |
R0233 1  |
R0234 1  |

|
| NDXCMD_FIELDS
|
$FIELD ndxcmd_fields =
| SET
|
| NDXSV_OPTIONS = [$INTEGER], | Command option indicators:
|
| $OVERLAY (NDXSV_OPTIONS)
|
| NDXSV_INPUT_CONCAT = [$BIT], | Input file concatenated to previous
| NDXSV_OUTPUT = [$BIT], | Generate output file
| NDXSV_REQUIRE = [$BIT], | Require file specified
| NDXSV_PAGES = [$BIT], | Include page references in index
| NDXSV_OVERRIDE = [$BIT], | Override master index information
| NDXSV_STANDARD_PAGE = [$BIT], | Generate standard page numbers
| NDXSV_CONTINUATION = [$BIT], | Generate continuation headings
| NDXSV_GUIDE = [$BIT], | Generate guide headings
| NDXSV_WORD_SORT = [$BIT], | Sort entries word by word
| NDXSV_LOG = [$BIT], | Generate /LOG message
| NDXSV_MASTER = [$BIT], | Generate a master index
| NDXSV_PAGE_MERGE = [$BIT], | Merge adjacent page references
| NDXSV_TELLTALE = [$BIT], | Generate telltale headings
|
| $CONTINUE
|
| NDXSH_FORMAT = [$SHORT_INTEGER], | Output format: DSR, TMS, TEX
| NDXSH_LAYOUT = [$SHORT_INTEGER], | Output layout type
| NDXSH_NONALPHA = [$SHORT_INTEGER], | Treatment of leading nonalphas during sort
| NDXSH_LEVEL = [$SHORT_INTEGER], | Deepest level to include in index
| NDXSG_COLUMN_WID = [$INTEGER], | Column width
| NDXSG_GUTTER_WID = [$INTEGER], | Gutter width
| NDXSG_LINES_PAGE = [$INTEGER], | Lines per page
| NDXSG_RESERVE_LINES = [$INTEGER], | Number of lines to reserve when requiring a file
| NDXSG_SEPARATE_WIDTH = [$INTEGER], | Width of reference portion of entry
| NDXST_MASTER_BOOK = [$DESCRIPTOR(DYNAMIC)], | Book name descriptor for Master indexing
| NDXST_INPUT_FILE = [$DESCRIPTOR(DYNAMIC)], | Input file name descriptor
| NDXST_OUTPUT_FILE = [$DESCRIPTOR(DYNAMIC)], | Output file name descriptor
| NDXST_REQUIRE_FILE = [$DESCRIPTOR(DYNAMIC)], | Require file name descriptor
| NDXST_RELATED_FILE = [$DESCRIPTOR(DYNAMIC)], | Related file name descriptor is saved here
| | by NDXINP for later use by MAKNDX
|
| NDXST_COMMAND_LINE = [$DESCRIPTOR(DYNAMIC)] | Copy of entire command line
|
| TES;
|
| End of NDXCMD_FIELDS
|
LITERAL
| NDXCMD$K_LENGTH = $FIELD_SET_SIZE;
|
MACRO
| $NDXCMD = BLOCK [NDXCMD$K_LENGTH] FIELD (NDXCMD_FIELDS) %;
|
$LITERAL
| DSR = $DISTINCT, | Output formats (NDXSH_FORMAT)
| TMS11_A = $DISTINCT, | Runoff
| | TMS=A
```



```

R0235 1      TMS11_E          = $DISTINCT;    ! TMS=E
R0236 1      TEX              = $DISTINCT;    ! TEX
R0237 1
R0238 1      $LITERAL         ! Output layouts (NDX$H_LAYOUT)
R0239 1      TWO_COLUMN       = $DISTINCT;    ! Normal two column format
R0240 1      ONE_COLUMN        = $DISTINCT;    ! Normal one column format
R0241 1      SEPARATE          = $DISTINCT;    ! Separate reference format
R0242 1      GALLEY           = $DISTINCT;    ! TMS11 Galley format
R0243 1
R0244 1      $LITERAL         ! Treatment of leading nonalphas during sort (NDX$H_NONALPHA)
R0245 1      BEFORE            = $DISTINCT;    ! Leading nonalphas sort before alphas
R0246 1      AFTER             = $DISTINCT;    ! Leading nonalphas sort after alphas
R0247 1      IGNORE            = $DISTINCT;    ! Leading nonalphas are ignored
R0248 1
R0249 1      !
R0250 1      !-- End of NDXCLI.REQ

```

:	112
:	113

```
0251 1
0252 1 REQUIRE 'REQ:NDXXPL';
```

D 12
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 BLISS-32 V4.0-742
[RUNOFF.SRC]INDXOUT.BLI;1

Page 8
(2)

ND
VO

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
Declarations

E 12
16-Sep-1984 01:04:24
15-Sep-1984 22:53:35

VAX-11 Bliss-32 V4.0-742
_S255\$DUA28:[RUNOFF.SRC]NDXXPL.REQ;1 Page 9
(1)

ND
V0

Version: 'V04-000'

*
* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
* ALL RIGHTS RESERVED.
*

* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
* TRANSFERRED.
*

* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
* CORPORATION.
*

* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*

++
FACILITY: DSR (Digital Standard RUNOFF) / DSRPLUS

ABSTRACT:
This file contains definitions of data structures used to support
the extended indexing features of the DSRPLUS INDEX program.

ENVIRONMENT: Transportable BLISS

AUTHOR: J.P. Kellerman

CREATION DATE: January 1982

MODIFIED BY:

002 KAD00002 Keith Dawson 07-Mar-1983
Global edit of all modules. Updated module names, idents,
copyright dates. Changed require files to BLISS library.

--
! Extended INDEX attributes block.

\$FIELD XPL_FIELDS =
SET

XPLSV_OPTIONS = [\$INTEGER], ! Attributes options

\$OVERLAY (XPLSV_OPTIONS)

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
Declarations

F 12
16-Sep-1984 01:04:24 VAX-11 Bliss-32 V4.0-742 Page 10
15-Sep-1984 22:53:35 _\$255\$DUA28:[RUNOFF.SRC]NDXXPL.REQ;1 (1)

```

R0310 1      XPLSV_VALID      = [ $BIT ],      ! Attributes block contains valid information.
R0311 1      XPLSV_BOLD       = [ $BIT ],      ! Bold page reference.
R0312 1      XPLSV_UNDERLINE  = [ $BIT ],      ! Underlined page reference.
R0313 1      XPLSV_BEGIN      = [ $BIT ],      ! Begin page range.
R0314 1      XPLSV_END        = [ $BIT ],      ! End page range.
R0315 1      XPLSV_MASTER     = [ $BIT ],      ! Master index entry.
R0316 1      XPLSV_PERMUTE    = [ $BIT ],      ! Permute index entry.
R0317 1      XPLSV_NOERMUTE   = [ $BIT ],      ! Set if permute explicitly forbidden.
R0318 1      XPLSV_SORT       = [ $BIT ],      ! Set if SORT string present.
R0319 1      XPLSV_APPEND     = [ $BIT ],      ! Set if append string present.
R0320 1
R0321 1      $CONTINUE
R0322 1
R0323 1      XPLST_SORT       = [ $DESCRIPTOR(DYNAMIC) ], ! SORT string.
R0324 1      XPLST_APPEND    = [ $DESCRIPTOR(DYNAMIC) ], ! APPEND string.
R0325 1
R0326 1      TES;
R0327 1
R0328 1      LITERAL
R0329 1      XPLSK_LENGTH = $FIELD_SET_SIZE;
R0330 1
R0331 1      MACRO
R0332 1      $XPL_BLOCK = BLOCK [XPLSK_LENGTH] FIELD (XPL_FIELDS) %;
R0333 1
R0334 1      !
R0335 1      ! Macros for INDEX_ATTRIBUTES flags
R0336 1
R0337 1      MACRO
R0338 1      XPLUSV_VALID      = 0, 0, 1, 0 %,      ! Set if attributes data is valid.
R0339 1      XPLUSV_BOLD       = 0, 1, 1, 0 %,      ! Set if page reference is bolded.
R0340 1      XPLUSV_UNDERLINE = 0, 2, 1, 0 %,      ! Set if page reference is underlined.
R0341 1      XPLUSV_BEGIN      = 0, 3, 1, 0 %,      ! Set if entry begins a page range.
R0342 1      XPLUSV_END        = 0, 4, 1, 0 %,      ! Set if entry ends a page range.
R0343 1      XPLUSV_MASTER     = 0, 5, 1, 0 %,      ! Set if master index entry only.
R0344 1      XPLUSV_PERMUTE    = 0, 6, 1, 0 %,      ! Set if entry is to be permuted.
R0345 1      XPLUSV_NOERMUTE   = 0, 7, 1, 0 %,      ! Set if permute is explicitly forbidden.
R0346 1      XPLUSV_SORT       = 0, 8, 1, 0 %,      ! Set if entry contains a SORT string.
R0347 1      XPLUSV_APPEND     = 0, 9, 1, 0 %,      ! Set if entry contains an APPEND string.
R0348 1
R0349 1      !
R0349 1      End of NDXXPL.REQ
```

: 114
: 115

```
0350 1
0351 1 REQUIRE 'REQ:NDXPOL';
```

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 11
(2)

ND
VO

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
Declarations

H 12
16-Sep-1984 01:04:24
15-Sep-1984 22:53:26

VAX-11 Bliss-32 V4.0-742
_S255\$DUA28:[RUNOFF.SRC]NDXPOL.REQ;1 Page 12
(1)

ND
VO

Version: 'V04-000'

*
* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
* ALL RIGHTS RESERVED.
*

* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
* TRANSFERRED.
*

* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
* CORPORATION.
*

* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*

++
FACILITY:
DSR (Digital Standard RUNOFF) /DSRPLUS DSRINDEX/INDEX Utility

ABSTRACT:
This file contains literals and macros defining the data structures
found in the internal index pool

ENVIRONMENT: Transportable

AUTHOR: JPK

CREATION DATE: January 1982

MODIFIED BY:

003 JPK00015 04-Feb-1983
Cleaned up module names, modified revision history to
conform with established standards. Updated copyright dates.

002 JPK00009 24-Jan-1983
Modified to enhance performance. The sort buckets have each
been divided into 27 sub-buckets; 1 for each letter and 1
for non-alphas. Removed reference to BUCKET from INDEX.
Definition of the structure was added to NDXPOL. References
to BUCKET were changed in modules NDXOUT, NDXINI, NDXFMT
and NDXDAT.

--


```

: R0409 1 ! Index entry
: R0410 1
: R0411 1 $FIELD XE_FIELDS =
: R0412 1 SET
: R0413 1
: R0414 1 XESA_PREV = [$ADDRESS], ! Link to previous item
: R0415 1 XESA_NEXT = [$ADDRESS], ! Link to next item
: R0416 1 XESA_SUBX = [$ADDRESS], ! Sub index pointer
: R0417 1 XESA_REF = [$ADDRESS], ! Reference pointer
: R0418 1 XESA_TEXT = [$ADDRESS], ! Pointer to text of index item
: R0419 1 XESA_SORT_AS = [$ADDRESS], ! Pointer to SORT_AS string
: R0420 1 XESH_SUBC = [$SHORT_INTEGER], ! Sub index level
: R0421 1
: R0422 1 XESV_FLAGS = [$SHORT_INTEGER], ! Entry flags
: R0423 1
: R0424 1 $OVERLAY (XESV_FLAGS)
: R0425 1
: R0426 1 XESV_BARS = [$BIT], ! Change bar flag
: R0427 1
: R0428 1 $CONTINUE
: R0429 1
: R0430 1 XESA_BOOK_LIST = [$ADDRESS] ! Master index book name list
: R0431 1
: R0432 1 $ALIGN (FULLWORD)
: R0433 1
: R0434 1 TES;
: R0435 1
: R0436 1 LITERAL
: R0437 1 XESK_LENGTH = $FIELD_SET_SIZE;
: R0438 1
: R0439 1 MACRO
: R0440 1 $XE_BLOCK = BLOCK [XESK_LENGTH] FIELD (XE_FIELDS) %;
: R0441 1
: R0442 1 ! End of Index entry
: R0443 1
: R0444 1
: R0445 1 ! Reference entry
: R0446 1
: R0447 1 $FIELD XX_FIELDS =
: R0448 1 SET
: R0449 1
: R0450 1 XXSA_LINK = [$ADDRESS], ! Link to additional entries
: R0451 1 XXSA_APPEND = [$ADDRESS], ! APPEND text pointer
: R0452 1 XXSH_PAGE = [$SHORT_INTEGER], ! Transaction number
: R0453 1
: R0454 1 XXSV_FLAGS = [$SHORT_INTEGER], ! Display attributes
: R0455 1
: R0456 1 $OVERLAY (XXSV_FLAGS)
: R0457 1
: R0458 1 XXSV_BOLD = [$BIT], ! Bold page reference
: R0459 1 XXSV_UNDERLINE = [$BIT], ! Underline page reference
: R0460 1 XXSV_BEGIN = [$BIT], ! Begin page range
: R0461 1 XXSV_END = [$BIT], ! End page range
: R0462 1
: R0463 1 $CONTINUE
: R0464 1
: R0465 1 XXSA_BOOK = [$ADDRESS] ! Master index book name
```

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
Declarations

J 12
16-Sep-1984 01:04:24
15-Sep-1984 22:53:26

VAX-11 Bliss-32 V4.0-742
_S255SDUA28:[RUNOFF.SRC]NDXPOL.REQ;1 Page 14
(1)

ND
VO

```

: R0466 1
: R0467 1      $ALIGN (FULLWORD)
: R0468 1
: R0469 1      TES;
: R0470 1
: R0471 1      LITERAL
: R0472 1      XX$K_LENGTH = $FIELD_SET_SIZE;
: R0473 1
: R0474 1      MACRO
: R0475 1      $XX_BLOCK = BLOCK [XX$K_LENGTH] FIELD (XX_FIELDS) %;
: R0476 1
: R0477 1      ! End of Reference entry
: R0478 1
: R0479 1
: R0480 1      ! Master index book reference entry
: R0481 1
: R0482 1      $FIELD XM_FIELDS =
: R0483 1      SET
: R0484 1
: R0485 1      XMSA_LINK      = [$ADDRESS],      ! Link to additional entries
: R0486 1      XMSA_BOOK      = [$ADDRESS]        ! Pointer to book name
: R0487 1
: R0488 1      TES;
: R0489 1
: R0490 1      LITERAL
: R0491 1      XMSK_LENGTH = $FIELD_SET_SIZE;
: R0492 1
: R0493 1      MACRO
: R0494 1      $XM_BLOCK = BLOCK [XMSK_LENGTH] FIELD (XM_FIELDS) %;
: R0495 1
: R0496 1      ! End of Master index book reference entry
: R0497 1
: R0498 1
: R0499 1      ! Current Entry
: R0500 1
: R0501 1      $FIELD C_FIELDS =
: R0502 1      SET
: R0503 1
: R0504 1      CSA_CURR      = [$ADDRESS],      ! Pointer to current cell
: R0505 1      CSA_PREV      = [$ADDRESS],      ! Pointer to previous cell
: R0506 1      CSA_HEAD      = [$ADDRESS],      ! Pointer to head of chain
: R0507 1
: R0508 1      $ALIGN (FULLWORD)
: R0509 1
: R0510 1      CSV_FLAGS      = [$INTEGER],      ! Current cell flags
: R0511 1
: R0512 1      $OVERLAY (CSV_FLAGS)
: R0513 1
: R0514 1      CSV_IDNS      = [$BIT]           ! Identical string flag
: R0515 1
: R0516 1      $CONTINUE
: R0517 1
: R0518 1      TES;
: R0519 1
: R0520 1      LITERAL
: R0521 1      CSK_LENGTH = $FIELD_SET_SIZE;
: R0522 1
```


NDXOUT
V04-000

NDXOUT -- Sort and store index entries
Declarations

K 12
16-Sep-1984 01:04:24
15-Sep-1984 22:53:26

VAX-11 Bliss-32 V4.0-742
_S255\$DUA28:[RUNOFF.SRC]NDXPOL.REQ;1 Page 15
(1)

ND
VO

```

: R0523 1 MACRO
: R0524 1   SC_BLOCK = BLOCK [C$K_LENGTH] FIELD (C_FIELDS) %;
: R0525 1
: R0526 1   ! End of current entry
: R0527 1
: R0528 1
: R0529 1
: R0530 1   ! Dummy datasets
: R0531 1
: R0532 1 LITERAL
: R0533 1   DS_X_ENTRY = XESK_LENGTH,
: R0534 1   DS_XX_ENTRY = XXSK_LENGTH,
: R0535 1   DS_XM_ENTRY = XMSK_LENGTH,
: R0536 1   DS_X_STRING = 0;
: R0537 1
: R0538 1
: R0539 1   ! Structure definition for bucket array.
: R0540 1
: R0541 1       Buckets are arranged so that each row represents the first letter of
: R0542 1       the string and each column represents the second letter of the string.
: R0543 1
: R0544 1       This approach is used only for master indexes as no performance
: R0545 1       improvement is realised until about 10 input files have been processed.
: R0546 1
: R0547 1       Indexes which are not master indexes use only the first element of
: R0548 1       each row, i.e., [0, 0] ... [26, 0].
: R0549 1
: R0550 1       The only exception is for nonalphabetic characters which use only
: R0551 1       element [0, 0]. Elements [0, 1] ... [0, 26] are not used since mapping
: R0552 1       all nonalphabetic into one row loses the sort order of the first
: R0553 1       character in the string. For nonalphabetic to work correctly in a two
: R0554 1       dimensional bucket scheme, the array would have to be at least 127 x 127
: R0555 1
: R0556 1           0   1           .   26
: R0557 1       0  **  not used  :  .
: R0558 1       1  A?  AA           AZ
: R0559 1       :
: R0560 1       :
: R0561 1       :
: R0562 1       26  Z?  ZA  .  .  .  ZZ
: R0563 1
: R0564 1 STRUCTURE
: R0565 1   $BUCKET_ARRAY [ROW_IDX, COL_IDX; M, N] =
: R0566 1   [M * N * %UPVAL] ($BUCKET_ARRAY + (ROW_IDX * N + COL_IDX) * %UPVAL);
: R0567 1
: R0568 1   !-- End of NDXPOL.REQ
```

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
Declarations

: 116
: 117

0569 1
0570 1 REQUIRE 'REQ:LETTER';

L 12
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 16
(2)

ND
VO

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
Declarations

M 12
16-Sep-1984 01:04:24
15-Sep-1984 22:52:49

VAX-11 Bliss-32 V4.0-742
_S255SDUA28:[RUNOFF.SRC]LETTER.REQ;1 Page 17
(1)

Version: 'V04-000'

*
* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
* ALL RIGHTS RESERVED.
*

* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
* TRANSFERRED.
*

* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
* CORPORATION.
*

* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*

++
FACILITY: DSR (Digital Standard RUNOFF) / DSRPLUS

ABSTRACT:
Macros to test if a character is an appropriately flavored letter,
and macros to convert between upper and lower case.

ENVIRONMENT: Transportable BLISS

AUTHOR: Rich Friday

CREATION DATE: 1978

MODIFIED BY:

002 KAD00002 Keith Dawson 07-Mar-1983
Global edit of all modules. Updated module names, idents,
copyright dates. Changed require files to BLISS library.

--
MACRO

upper letter (khar) = ! See if upper case letter
(khar GEQ %C'A' AND khar LEQ %C'Z')

%,

lower letter (khar) = ! See if lower case letter
(khar GEQ %C'a' AND khar LEQ %C'z')

%,

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
Declarations

N 12
16-Sep-1984 01:04:24
15-Sep-1984 22:52:49

VAX-11 Bliss-32 V4.0-742
_S255\$DUA28:[RUNOFF.SRC]LETTER.REQ;1 Page 18
(1)

```
: MR0628 1      letter (khar) = ! See if any type of letter
: MR0629 1      (upper_letter (khar) OR lower_letter (khar))
: R0630 1      %;
: R0631 1
: R0632 1      MACRO
: MR0633 1      upper_case (khar) = ! Convert to upper case
: MR0634 1      (khar + %C'A' - %C'a')
: R0635 1      %,
: R0636 1
: MR0637 1      lower_case (khar) = ! Convert to lower case
: MR0638 1      (khar + %C'a' - %C'A')
: R0639 1      %;
: R0640 1
: R0641 1      !
:                               End of LETTER.REQ
```

ND
VO

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
Declarations

B 13
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 19
(2)

NDX
V04

```
: 118
: 119
: 120
: 121
: 122

0642 1
L 0643 1 %IF %BLISS (BLISS32)
0644 1 %THEN
0645 1
0646 1 REQUIRE 'REQ:NDXVMSREQ';
```

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
Declarations

C 13
16-Sep-1984 01:04:24
15-Sep-1984 22:53:32

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXVMSREQ.R32;1

Page 20
(1)

ND
VO

Version: 'V04-000'

*
* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
* ALL RIGHTS RESERVED.
*

* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
* TRANSFERRED.
*

* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
* CORPORATION.
*

* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*

++
FACILITY:
DSR (Digital Standard RUNOFF) /DSRPLUS DSRINDEX/INDEX Utility

ABSTRACT:
This file contains external references to the error message numbers
for DSRINDEX/INDEX.

New messages must be defined in NDXVMSMSG.MSG and referenced here:
both in the MACRO section (for DSRINDEX) and the EXTERNAL LITERAL
section (for INDEX)

ENVIRONMENT: VAX/VMS User Mode

AUTHOR: JPK

CREATION DATE: 01-Feb-1983

MODIFIED BY:

004 JPK00022 30-Mar-1983
Modified NDXVMS, NDXFMT, NDXPAG, NDXVMSMSG and NDXVMSREQ
to generate TEX output. Added module NDXTEX.

003 JPK00021 28-Mar-1983
Modified NDXT20 to include E2.0 functionality.
Modified NDXCLIDMP, NDXFMT, NDXPAG, NDXVRS to require RNODEF
for BLISS36 and to remove any conditional require based on
DSRPLUS_DEF.

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
Declarations

D 13
16-Sep-1984 01:04:24
15-Sep-1984 22:53:32

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXVMSREQ.R32;1

Page 21
(1)

: R0704 1
: R0705 1
: R0706 1
: R0707 1
: R0708 1
: R0709 1
: R0710 1

002 JPK00010 04-Feb-1983
Cleaned up module names, modified revision history to
conform with established standards. Updated copyright dates.
--
REQUIRE 'REQ:RNODEF';

ND
VO

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
Declarations

E 13
16-Sep-1984 01:04:24
15-Sep-1984 22:54:08

VAX-11 Bliss-32 V4.0-742
_S255SDUA28:[RUNOFF.SRC]RNODEF.REQ;1 Page 22
(1)

ND
VO

Version: 'V04-000'

*
* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
* ALL RIGHTS RESERVED.
*

* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
* TRANSFERRED.
*

* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
* CORPORATION.
*

* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*

++
FACILITY: DSR (Digital Standard RUNOFF) / DSRPLUS

ABSTRACT:
Converts BLISS/VARIANT values into useful names.

ENVIRONMENT: Transportable BLISS

AUTHOR: Rich Friday

CREATION DATE: 1978

MODIFIED BY:

016 KAD00016 Ray Marshall 19-Mar-1984
Added GERMAN, FRENCH, & ITALIAN.

015 KAD00015 Keith Dawson 18-Apr-1983
Made the LN01 conditional the default for vanilla DSR --
its value is 0 (no variant supplied).

014 KAD00014 Keith Dawson 22-Mar-1983
Asserted the LN01 conditional when DSRPLUS is asserted.

013 KAD00013 Keith Dawson 20-Mar-1983
Removed all references to .BIX and .BTC files.

012 KAD00012 Keith Dawson 07-Mar-1983
Global edit of all modules. Updated module names, idents,
copyright dates. Changed require files to BLISS library.

NDXOUT
VC4-000

NDXOUT -- Sort and store index entries
Declarations

F 13

16-Sep-1984 01:04:24

15-Sep-1984 22:54:08

VAX-11 Bliss-32 V4.0-742

_\$255\$DUA28:[RUNOFF.SRC]RNODEF.REQ;1

Page 23

(1)

R0768 1
R0769 1
R0770 1
R0771 1
R0772 1
R0773 1
R0774 1
R0775 1
R0776 1
R0777 1
R0778 1
R0779 1
R0780 1
R0781 1
R0782 1
R0783 1
R0784 1
R0785 1
R0786 1
R0787 1
R0788 1
R0789 1
R0790 1
R0791 1
R0792 1
R0793 1
R0794 1
R0795 1
R0796 1
R0797 1
R0798 1
R0799 1
R0800 1
R0801 1
R0802 1
R0803 1
R0804 2
R0805 1
R0806 1
R0807 1
R0808 1
R0809 1
R0810 1
R0811 1
R0812 1
R0813 1
R0814 2
R0815 1
R0816 1
R0817 1
R0818 1
R0819 1
R0820 1
R0821 1
R0822 1
R0823 1
R0824 1

--

++

DEFINITION OF /VARIANT BITS

The bit assignments are as follows:

Bit	Weight	Meaning
--	0	If no /VARIANT is supplied (as for vanilla DSR), compile with LN01 support. LN01 support is also implied by the DSRPLUS variant.
0	1	CLEAR = Unassigned SET = Unassigned
1	2	CLEAR = Normal compile SET = Compile for DSRPLUS
4-6	16	CLEAR = English (American) version SET = 16 = German (Austrian) 32 = French 48 = Italian

--

This variable (LN01) controls whether or not to compile an LN01-flavored DSR. It is asserted by default, and also whenever DSRPLUS is asserted.

Modules utilizing LN01 are:

DOOPTS NOUT

COMPILETIME

ln01 =

((%VARIANT EQL 0) OR %VARIANT/2)

;

This variable (DSRPLUS) controls compilation for the DSRPLUS program.

All modules utilize DSRPLUS.

COMPILETIME

dsrplus =

(%VARIANT/2)

;

This variable (FLIP) controls compilation of FLIP features of DSRPLUS. It assures that FLIP features are compiled only on VMS systems.

Modules utilizing FLIP are many and various.

COMPILETIME

flip =

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
Declarations

G 13
16-Sep-1984 01:04:24
15-Sep-1984 22:54:08

VAX-11 Bliss-32 V4.0-742
_S255\$DUA28:[RUNOFF.SRC]RNODEF.REQ;1 Page 24
(1)

ND
VO

```
: R0825 2      ( %VARIANT/2 AND %BLISS(BLISS32) )
: R0826 1      ;
: R0827 1
: R0828 1
: R0829 1      -----
: R0830 1      4-6    16    CLEAR = English (American) version
: R0831 1      SET    =    16 = German (Austrian)
: R0832 1      32 = French
: R0833 1      48 = Italian
: R0834 1      COMPILETIME
: R0835 1      German = ( %VARIANT/16 AND NOT %VARIANT/32 AND NOT %VARIANT/64 ) ;
: R0836 1      COMPILETIME
: R0837 1      French = ( NOT %VARIANT/16 AND %VARIANT/32 AND NOT %VARIANT/64 ) ;
: R0838 1      COMPILETIME
: R0839 1      Italian = ( %VARIANT/16 AND %VARIANT/32 AND NOT %VARIANT/64 ) ;
: R0840 1      -----
:                               End of RNODEF.REQ
```

```

R0841 1
L R0842 1
R0843 1
R0844 1
R0845 1
R0846 1
R0847 1
R0848 1
R0849 1
R0850 1
R0851 1
R0852 1
R0853 1
R0854 1
R0855 1
R0856 1
R0857 1
R0858 1
R0859 1
R0860 1
R0861 1
R0862 1
R0863 1
R0864 1
R0865 1
R0866 1
R0867 1
R0868 1
R0869 1
R0870 1
R0871 1
R0872 1
R0873 1
R0874 1
R0875 1
R0876 1
R0877 1
R0878 1
R0879 1
R0880 1
R0881 1
R0882 1
R0883 1
R0884 1
R0885 1
R0886 1
R0887 1
R0888 1
R0889 1
R0890 1
R0891 1
R0892 1
R0893 1
R0894 1
R0895 1
R0896 1
R0897 1

XIF NOT DSRPLUS
XTHEN

MACRO
INDEX$_BADLOGIC = DSRINDEX$_BADLOGIC X,
INDEX$_BADVALUE = DSRINDEX$_BADVALUE X,
INDEX$_INSVIRMEM = DSRINDEX$_INSVIRMEM X,
INDEX$_LINELENG = DSRINDEX$_LINELENG X,
INDEX$_NOREF = DSRINDEX$_NOREF X,
INDEX$_OPENIN = DSRINDEX$_OPENIN X,
INDEX$_OPENOUT = DSRINDEX$_OPENOUT X,
INDEX$_TOOMANY = DSRINDEX$_TOOMANY X,
INDEX$_VALERR = DSRINDEX$_VALERR X,
INDEX$_CANTBAL = DSRINDEX$_CANTBAL X,
INDEX$_CLOSEQUOT = DSRINDEX$_CLOSEQUOT X,
INDEX$_CONFQUAL = DSRINDEX$_CONFQUAL X,
INDEX$_CTRLCHAR = DSRINDEX$_CTRLCHAR X,
INDEX$_DOESNTFIT = DSRINDEX$_DOESNTFIT X,
INDEX$_DUPBEGIN = DSRINDEX$_DUPBEGIN X,
INDEX$_EMPTYIN = DSRINDEX$_EMPTYIN X,
INDEX$_IGNORED = DSRINDEX$_IGNORED X,
INDEX$_INVINPUT = DSRINDEX$_INVINPUT X,
INDEX$_INVRECORD = DSRINDEX$_INVRECORD X,
INDEX$_LASTCONT = DSRINDEX$_LASTCONT X,
INDEX$_NOBEGIN = DSRINDEX$_NOBEGIN X,
INDEX$_NOEND = DSRINDEX$_NOEND X,
INDEX$_NOINDEX = DSRINDEX$_NOINDEX X,
INDEX$_NOLIST = DSRINDEX$_NOLIST X,
INDEX$_OVERSTRK = DSRINDEX$_OVERSTRK X,
INDEX$_SKIPPED = DSRINDEX$_SKIPPED X,
INDEX$_SYNTAX = DSRINDEX$_SYNTAX X,
INDEX$_TEXTFILE = DSRINDEX$_TEXTFILE X,
INDEX$_TOODEEP = DSRINDEX$_TOODEEP X,
INDEX$_TOOFEW = DSRINDEX$_TOOFEW X,
INDEX$_TRUNCATED = DSRINDEX$_TRUNCATED X,
INDEX$_COMPLETE = DSRINDEX$_COMPLETE X,
INDEX$_CREATED = DSRINDEX$_CREATED X,
INDEX$_IDENT = DSRINDEX$_IDENT X,
INDEX$_PROCFILE = DSRINDEX$_PROCFILE X,
INDEX$_TEXT = DSRINDEX$_TEXT X,
INDEX$_TEXTD = DSRINDEX$_TEXTD X,
INDEX$_TMS11 = DSRINDEX$_TMS11 X;

XFI

EXTERNAL LITERAL
INDEX$_BADLOGIC, ! <internal logic error detected>
INDEX$_BADVALUE, ! <'!AS' is an invalid keyword value>
INDEX$_INSVIRMEM, ! <insufficient virtual memory>
INDEX$_LINELENG, ! <maximum line length is 120>
INDEX$_NOREF, ! <page reference not found>
INDEX$_OPENIN, ! <error opening '!AS' for input>
INDEX$_OPENOUT, ! <error opening '!AS' for output>
INDEX$_TOOMANY, ! <too many values supplied>
INDEX$_VALERR, ! <specified value is out of legal range>
INDEX$_CANTBAL, ! <can't balance last page>
```

```
: R0898 1 INDEX$_CLOSEQUOTE, <missing close quote>
: R0899 1 INDEX$_CONFQUAL, <conflicting qualifiers>
: R0900 1 INDEX$_CTRLCHAR, <the following line contains control characters - ignored>
: R0901 1 INDEX$_DOESNTFIT, <'!AD' will not fit at the current indentation level>
: R0902 1 INDEX$_DUPBEGIN, <duplicate .XPLUS (BEGIN) - inserted as .XPLUS (>>
: R0903 1 INDEX$_EMPTYIN, <empty input file '!AS'>
: R0904 1 INDEX$_IGNORED, <'!AS' ignored>
: R0905 1 INDEX$_INVINPUT, <invalid input file format in file '!AS'>
: R0906 1 INDEX$_INVRECORD, <invalid record type in file '!AS'>
: R0907 1 INDEX$_LASTCONT, <can't generate continuation heading on last page>
: R0908 1 INDEX$_NOBEGIN, <.XPLUS (END) with no .XPLUS (BEGIN) - inserted as .XPLUS (>>
: R0909 1 INDEX$_NOEND, <.XPLUS (BEGIN) has no corresponding .XPLUS (END)>
: R0910 1 INDEX$_NOINDEX, <no index information in file '!AS'>
: R0911 1 INDEX$_NOLIST, <parameter list not allowed>
: R0912 1 INDEX$_OVERSTRK, <the following line contains an overstrike sequence>
: R0913 1 INDEX$_SKIPPED, <!UL reference!%S inside page range - ignored>
: R0914 1 INDEX$_SYNTAX, <error parsing '!AS'>
: R0915 1 INDEX$_TEXTFILE, <error processing line !UL of TEX character file '!AS'>
: R0916 1 INDEX$_TOODEEP, <maximum subindex depth exceeded>
: R0917 1 INDEX$_TOOFEW, <not enough values supplied>
: R0918 1 INDEX$_TRUNCATED, <string too long - truncated>
: R0919 1 INDEX$_COMPLETE, <processing complete '!AS'>
: R0920 1 INDEX$_CREATED, <'!AS' created>
: R0921 1 INDEX$_IDENT, <INDEX version !AD>
: R0922 1 INDEX$_PROCFILE, <processing file '!AS'>
: R0923 1 INDEX$_TEXT, <!AS>
: R0924 1 INDEX$_TEXTD, <entry text: '!AD'>
: R0925 1 INDEX$_TMS11, <output file full - continuing with file '!AS'>
: R0926 1
```

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
Declarations

J 13
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI:1

Page 27
(2)

ND
VO

```
: 123      0927 1
: 124      0928 1 %FI
: 125      0929 1
: 126      0930 1 SWITCHES LIST (NOREQUIRE);
: 127      0931 1
: 128      0932 1
: 129      0933 1
: 130      0934 1
: 131      0935 1
: 132      0936 1 MACRO
: 133      0937 1     REPEAT = WHILE 1 DO %;
: 134      0938 1
: 135      0939 1
: 136      0940 1
: 137      0941 1
: 138      0942 1
: 139      0943 1 LITERAL
: 140      0944 1     TRUE = 1;
: 141      0945 1     FALSE = 0;
: 142      0946 1
: 143      0947 1
: 144      0948 1
: 145      0949 1
: 146      0950 1
: 147      0951 1 OWN
: 148      0952 1     CELL      : $C_BLOCK,      ! Current call characteristics
: 149      0953 1     SORT_STR  : VECTOR [CH$ALLOCATION (1200)], ! Build sort string here
: 150      0954 1     SORT_PTR,   ! Pointer to sort string
: 151      0955 1     SORT_LEN,   ! Length of sort string
: 152      0956 1     USER_SORT_LEN, ! Length of user specified sort string
: 153      0957 1     USER_SORT_PTR; ! Pointer to user specified sort string
: 154      0958 1
: 155      0959 1
: 156      0960 1
: 157      0961 1
: 158      0962 1
: 159      0963 1 EXTERNAL LITERAL
: 160      0964 1     TAB      : UNSIGNED (8),      ! TAB character
: 161      0965 1     RINTES   : UNSIGNED (8);      ! Special escape character
: 162      0966 1
: 163      0967 1 EXTERNAL
: 164      0968 1     CMDBLK : $NDXCMD,      ! Command line information block
: 165      0969 1     XPLBLK : $XPL_BLOCK,    ! Extended indexing information block
: 166      0970 1     BUCKET : $BUCKET_ARRAY [27, 27], ! Hashing buckets (first character of entry)
: 167      0971 1     BOOKID;  ! Address of master index book ident string
: 168      0972 1
: 169      0973 1 EXTERNAL ROUTINE
: 170      0974 1     SAVDAT,      ! Place data in work storage
: 171      0975 1     DMPENT : NOVALUE;
```

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
XOUT -- Put away index item

K 13
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.ELI;1

Page 28
(3)

ND
VO

```
: 173 0976 1 %SBTTL 'XOUT -- Put away index item'
: 174 0977 1
: 175 0978 1 GLOBAL ROUTINE XOUT (ENTRY_LENGTH, ENTRY_PTR, XTN, BAR_FLAG) : NOVALJE =
: 176 0979 1
: 177 0980 1  !+
: 178 0981 1  FUNCTIONAL DESCRIPTION:
: 179 0982 1
: 180 0983 1      Place an index or sub-index item into the index master storage
: 181 0984 1      list in alphabetical order.
: 182 0985 1
: 183 0986 1  FORMAL PARAMETERS:
: 184 0987 1
: 185 0988 1      ENTRY_LENGTH      - Length of index entry text
: 186 0989 1      ENTRY_PTR        - CH$PTR to index entry text
: 187 0990 1      XTN              - Transaction number
: 188 0991 1      BAR_FLAG         - Change bar flag
: 189 0992 1
: 190 0993 1  IMPLICIT INPUTS:
: 191 0994 1
: 192 0995 1      CMDBLK          - Command line information block
: 193 0996 1      XPLBLK          - Extended indexing attributes block
: 194 0997 1      CELL            - Information about current position in list
: 195 0998 1
: 196 0999 1  IMPLICIT OUTPUTS:
: 197 1000 1
: 198 1001 1      None
: 199 1002 1
: 200 1003 1  ROUTINE VALUE:
: 201 1004 1  COMPLETION CODES:
: 202 1005 1
: 203 1006 1      NONE
: 204 1007 1
: 205 1008 1  SIDE EFFECTS:
: 206 1009 1
: 207 1010 1      Master index is built.
: 208 1011 1
: 209 1012 1  --
: 210 1013 1
: 211 1014 2  BEGIN
: 212 1015 2
: 213 1016 2  LOCAL
: 214 1017 2      INT_HL,
: 215 1018 2      LAST_NB,
: 216 1019 2      STG_PTR,
: 217 1020 2      SUBX_STG,
: 218 1021 2      SUBX_CNT;
: 219 1022 2
: 220 1023 2  !
: 221 1024 2  ! Is this trip necessary?
: 222 1025 2  !
: 223 1026 2  IF .ENTRY_LENGTH EQL 0 THEN RETURN;
: 224 1027 2
: 225 1028 2  !
: 226 1029 2  ! Initialization
: 227 1030 2
: 228 1031 3  BEGIN
: 229 1032 3  MAP
```



```
: 230      1033      3      CELL : VECTOR [CSK_LENGTH];
: 231      1034      3
: 232      1035      3      INCR I FROM 0 TO CSK_LENGTH - 1 DO CELL [.I] = 0;
: 233      1036      2      END;
: 234      1037      2
: 235      1038      2      SUBX_STG = .ENTRY_PTR;          ! Get address of index string
: 236      1039      2      INT_HL = .ENTRY_LENGTH;      ! Get length of index string.
: 237      1040      2
: 238      1041      2      STG_PTR = .SUBX_STG;
: 239      1042      2      LAST_NB = .SUBX_STG;
: 240      1043      2      SUBX_CNT = 0;
: 241      1044      2
: 242      1045      2      IF .XPLBLK [XPLSV_VALID]
: 243      1046      2      THEN
: 244      1047      3      BEGIN
: 245      1048      3          ! Attributes block is valid. Initialize user specified sort parameters.
: 246      1049      3
: 247      1050      3
: 248      1051      3      BIND
: 249      1052      3          SORT_STR = XPLBLK [XPLST_SORT] : $STR_DESCRIPTOR ();
: 250      1053      3
: 251      1054      3      USER_SORT_LEN = .SORT_STR [STR$H_LENGTH];
: 252      1055      3      USER_SORT_PTR = .SORT_STR [STR$A_POINTER];
: 253      1056      3      END
: 254      1057      2      ELSE
: 255      1058      2          USER_SORT_LEN = 0;
: 256      1059      2
: 257      1060      2      !
: 258      1061      2      ! Scan the entire character string
: 259      1062      2
: 260      1063      2      INCR I FROM 1 TO .INT_HL DO
: 261      1064      3      BEGIN
: 262      1065      3
: 263      1066      3      LOCAL
: 264      1067      3          CHARACTER;
: 265      1068      3
: 266      1069      3      CHARACTER = CH$RCHAR_A (STG_PTR);
: 267      1070      3
: 268      1071      3      !
: 269      1072      3      ! Look for special handling
: 270      1073      3
: 271      1074      3      IF .CHARACTER EQL RINTES
: 272      1075      3      THEN
: 273      1076      4      BEGIN
: 274      1077      4
: 275      1078      4          ! Interpret escape sequences.
: 276      1079      4
: 277      1080      4      CHARACTER = CH$RCHAR_A (STG_PTR);
: 278      1081      4      I = .I + 1;
: 279      1082      4
: 280      1083      4      IF .CHARACTER EQL %C'J'
: 281      1084      4      THEN
: 282      1085      5      BEGIN
: 283      1086      5
: 284      1087      5          ! Set up sub-index
: 285      1088      5
: 286      1089      5      LOCAL
```

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
XOUT -- Put away index item

M 13
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 30
(3)

ND
VO

```

287      1090      5      T_PTR : REF $XE_BLOCK;
288      1091      5
289      1092      5      CH$RCHAR_A (STG_PTR);
290      1093      5      I = .I + 1;
291      1094      5
292      1095      5
293      1096      5      Set up sort string
294      1097      5
295      1098      5      SORT_AS (.SUBX_STG, CH$DIFF (.LAST_NB, .SUBX_STG), .SUBX_CNT);
296      1099      5
297      1100      5
298      1101      5      Look for entry
299      1102      5
300      1103      5      FIND_POS (.SUBX_STG, CH$DIFF (.LAST_NB, .SUBX_STG), .SUBX_CNT, FALSE, 0);
301      1104      5
302      1105      5
303      1106      5      Enter it if it is not already there
304      1107      5
305      1108      5      IF NOT .CELL[C$V_IDNS]
306      1109      5      THEN
307      1110      5          INSERT_INX (.SUBX_STG, CH$DIFF (.LAST_NB, .SUBX_STG), .SUBX_CNT, 0, .BAR_FLAG, .ENTRY_LE
308      1111      5
309      1112      5
310      1113      5      Clear out sort string
311      1114      5
312      1115      5      SORT_LEN = 0;
313      1116      5      SORT_PTR = 0;
314      1117      5
315      1118      5
316      1119      5      Skip over text
317      1120      5
318      1121      5      SUBX_STG = .STG_PTR;
319      1122      5      LAST_NB = .SUBX_STG;
320      1123      5      SUBX_CNT = .SUBX_CNT + 1;
321      1124      5      CELL[C$V_IDNS] = FALSE;
322      1125      5
323      1126      5
324      1127      5      Is there a sub-index list?
325      1128      5
326      1129      5      T_PTR = .CELL [C$A_CURR];
327      1130      5      IF .T_PTR [X$A_SUBX] EQL 0
328      1131      5      THEN
329      1132      5
330      1133      5          Insert end of sub-index list
331      1134      5
332      1135      5          INSERT_INX (0, 0, .SUBX_CNT, 0, .BAR_FLAG, .ENTRY_LENGTH, .ENTRY_PTR)
333      1136      5      ELSE
334      1137      6          BEGIN
335      1138      6
336      1139      6          Set pointer to head of sub list
337      1140      6
338      1141      6          CELL [C$A_PREV] = .CELL [C$A_CURR];
339      1142      6          CELL [C$A_CURR] = .T_PTR [X$A_SUBX]
340      1143      6          END
341      1144      5      END
342      1145      4      ELSE
343      1146      4          LAST_NB = .STG_PTR
```



```

      .TITLE  NDXOUT NDXOUT -- Sort and store index entries
      .IDENT  \V04-000\

      .PSECT  $OWNS$,NOEXE,2

00000 CELL:   .BLKB   16
00010 SORT_STR:
              .BLKB   1200
004C0 SORT_PTR:
              .BLKB    4
004C4 SORT_LEN:
              .BLKB    4
004C8 USER_SORT_LEN:
              .BLKB    4
004CC USER_SORT_PTR:
              .BLKB    4

      .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

```

```
NDXOUT -- Sort and store index entries
XOUT -- Put away index item
```

Page 32
(3)

NDX
V04

				.EXTRN	DSRINDEX\$_OVERSTRK	
				.EXTRN	DSRINDEX\$_SKIPPED	
				.EXTRN	DSRINDEX\$_SYNTAX	
				.EXTRN	DSRINDEX\$_TEXTFILE	
				.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	TAB, RINTES, CMDBLK	
				.EXTRN	XPLBLK, BUCKET, BOOKID	
				.EXTRN	SAVDAT, DMPENT	
				.PSECT	\$CODE\$,NOWRT,2	
F9				.ENTRY	XOUT, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	: 0978
	5B	00000000G	EF 9E 00002	MOVAB	XPLBLK, R11	:
	5A	00000000'	EF 9E 00009	MOVAB	CELL, R10	:
	56	04	AC D0 00010	MOVL	ENTRY_LENGTH, R6	: 1026
			01 12 00014	BNEQ	1\$	:
			04 00016	RET		:
			50 D4 00017	CLRL	I	: 1035
		6A40	D4 00019	CLRL	CELL[I]	:
			03 F3 0001C	ACBLEQ	#3, I, 2\$	:
	50		AC D0 00020	MOVL	ENTRY_PTR, SUBX_STG	: 1038
	54	08	D0 00024	MOVL	R6, INT_HL	: 1039
	59		D0 00027	MOVL	SUBX_STG, STG_PTR	: 1041
	57		D0 0002A	MOVL	SUBX_STG, LAST_NB	: 1042
	52		D4 0002D	CLRL	SUBX_CNT	: 1043
			E9 0002F	BLBC	XPLBK, 3\$	: 1045
	OE		AB 3C 00032	MOVZWL	SORT_STR, USER_SORT_LEN	: 1054
04CB	CA	04	AB D0 00038	MOVL	SORT_STR+4, USER_SORT_PTR	: 1055
04CC	CA	08	04 11 0003E	BRB	4\$	: 1045
		04CB	CA D4 00040	CLRL	USER_SORT_LEN	: 1058
			58 D4 00044	CLRL	I	: 1063
		0099	31 00046	BRW	11\$	:
	50		87 9A 00049	MOVZBL	(STG_PTR)+, CHARACTER	: 1069
00000000G	8F		50 D1 0004C	CML	CHARACTER, #RINTES	: 1074
			03 13 00053	BEQL	6\$	:
		0082	31 00055	BRW	9\$	:
	50		87 9A 00058	MOVZBL	(STG_PTR)+, CHARACTER	: 1080
			58 D6 0005B	INCL	I	: 1081
0000004A	8F		50 D1 0005D	CML	CHARACTER, #74	: 1083
			79 12 00064	BNEQ	10\$	:
			57 D6 00066	INCL	STG_PTR	: 1092
			58 D6 00068	INCL	I	: 1093
			55 DD 0006A	PUSHL	SUBX_CNT	: 1098
	52		54 C3 0006C	SUBL3	SUBX_STG, LAST_NB, R3	:
			53 DD 00070	PUSHL	R3	:
			54 DD 00072	PUSHL	SUBX_STG	:
00000000V	EF		03 FB 00074	CALLS	#3, SORT_AS	:
			7E 7C 0007B	CLRQ	-(SP)	: 1103
			28 BB 0007D	PUSHR	#^M<R3,R5>	:

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
XOUT -- Put away index item

C 14
16-Sep-1984 01:04:24 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:07:15 [RUNOFF.SRC]NDXOUT.BLI;1

Page 33
(3)

NDX
V04

00000000V	EF		54	DD	0007F	PUSHL	SUBX STG	
	15		55	FB	00081	CALLS	#5, FIND_POS	
		0C	AA	E8	00088	BLBS	CELL+12, 7\$	1108
		08	AC	DD	0008C	PUSHL	ENTRY_PTR	1110
			56	DD	0008F	PUSHL	R6	
		10	AC	DD	00091	PUSHL	BAR_FLAG	
			7E	D4	00094	CLRL	-(SP)	
			28	BB	00096	PUSHR	#^M<R3,R5>	
00000000V	EF		54	DD	00098	PUSHL	SUBX STG	
		04C0	07	FB	0009A	CALLS	#7, INSERT_INX	
			CA	7C	000A1	CLRQ	SORT_PTR	1116
	54		57	DD	000A5	MOVL	STG_PTR, SUBX_STG	1121
	52		54	DD	000A8	MOVL	SUBX_STG, LAST_NB	1122
			55	D6	000AB	INCL	SUBX_CNT	1123
0C	AA		01	8A	000AD	BICB2	#1, CELL+12	1124
	50		6A	DD	000B1	MOVL	CELL, T_PTR	1129
		08	A0	D5	000B4	TSTL	8(T_PTR)	1130
			17	12	000B7	BNEQ	8\$	
		08	AC	DD	000B9	PUSHL	ENTRY_PTR	1135
			56	DD	000BC	PUSHL	R6	
		10	AC	DD	000BE	PUSHL	BAR_FLAG	
			7E	D4	000C1	CLRL	-(SP)	
			55	DD	000C3	PUSHL	SUBX_CNT	
			7E	7C	000C5	CLRQ	-(SP)	
00000000V	EF		07	FB	000C7	CALLS	#7, INSERT_INX	
			12	11	000CE	BRB	11\$	
04	AA		6A	DD	000D0	MOVL	CELL, CELL+4	1141
	6A	08	A0	DD	000D4	MOVL	8(T_PTR), CELL	1142
			08	11	000D8	BRB	11\$	1130
	20		50	D1	000DA	CMPL	CHARACTER, #32	1149
			03	13	000DD	BEQL	11\$	
FF61	58		57	DD	000DF	MOVL	STG_PTR, LAST_NB	
			59	F1	000E2	ACBL	INT_HL, #1, 1, 5\$	1074
			55	DD	000E8	PUSHL	SUBX_CNT	1156
			54	C2	000EA	SUBL2	SUBX_STG, R2	
			52	DD	000ED	PUSHL	R2	
			54	DD	000EF	PUSHL	SUBX_STG	
00000000V	EF		03	FB	000F1	CALLS	#3, SORT_AS	
		0C	AC	DD	000F8	PUSHL	XTN	1157
			01	DD	000FB	PUSHL	#1	
			24	BB	000FD	PUSHR	#^M<R2,R5>	
			54	DD	000FF	PUSHL	SUBX STG	
00000000V	EF		05	FB	00101	CALLS	#5, FIND_POS	
		08	AC	DD	00108	PUSHL	ENTRY_PTR	1158
			56	DD	0010B	PUSHL	R6	
	7E	0C	AC	7D	0010D	MOVQ	XTN, -(SP)	
			24	BB	00111	PUSHR	#^M<R2,R5>	
			54	DD	00113	PUSHL	SUBX STG	
00000000V	EF		07	FB	00115	CALLS	#7, INSERT_INX	
			04	00	0011C	RET		1160

; Routine Size: 285 bytes, Routine Base: \$CODE\$ + 0000

```
359 1161 1 XSBTTL 'SORT_AS -- Build sort string'
360 1162 1 ROUTINE SORT_AS (I_PTR, I_LEN, LEVEL) : NOVALUE =
361 1163 1 ++
362 1164 1
363 1165 1 FUNCTIONAL DESCRIPTION:
364 1166 1
365 1167 1     This routine builds the sort string used to position the index
366 1168 1     entry in the index.
367 1169 1
368 1170 1     If the user specified a sort string then that string is used.
369 1171 1
370 1172 1     If /SOPT=LETTER was specified on the command line, a sort string
371 1173 1     is built from the input string.
372 1174 1
373 1175 1     If /SORT=NONALPHA=IGNORE was specified on the command line, a
374 1176 1     sort string is built if the first character in the input string
375 1177 1     is not alphabetic.
376 1178 1
377 1179 1     Otherwise no string is built and the entry is positioned
378 1180 1     according to the text of the entry.
379 1181 1
380 1182 1 FORMAL PARAMETERS:
381 1183 1
382 1184 1     I_LEN   - Length of input string
383 1185 1     I_PTR   - Pointer to input string
384 1186 1     LEVEL   - Subindex level
385 1187 1
386 1188 1 IMPLICIT INPUTS:
387 1189 1
388 1190 1     XPLBLK   - Extended index attributes block
389 1191 1     CMDBLK   - Command line information block
390 1192 1     USER_SORT_LEN - Length of user specified sort string if any
391 1193 1     USER_SORT_PTR - Pointer to user specified sort string if any
392 1194 1
393 1195 1 IMPLICIT OUTPUTS:
394 1196 1
395 1197 1     USER_SORT_LEN - Length of remainder of user specified sort string if any
396 1198 1     USER_SORT_PTR - Pointer to remainder of user specified sort string if any
397 1199 1     SORT_PTR     - Points to the sort string if any
398 1200 1     SORT_LEN     - Is the length of the sort string
399 1201 1
400 1202 1 ROUTINE VALUE:
401 1203 1 COMPLETION CODES:
402 1204 1
403 1205 1     None
404 1206 1
405 1207 1 SIDE EFFECTS:
406 1208 1
407 1209 1     None
408 1210 1
409 1211 2 -- BEGIN
410 1212 2
411 1213 2     LOCAL
412 1214 2     LEN,
413 1215 2     PTR;
414 1216 2
415 1217 2     SORT_LEN = 0;
```

```
416 1218 2 SORT_PTR = CH$PTR (SORT_STR);
417 1219 2
418 1220 2 IF .USER_SORT_LEN NEQ 0
419 1221 2 THEN
420 1222 3 BEGIN
421 1223 3
422 1224 3     User specified a sort string.
423 1225 3     Get the next segment.
424 1226 3
425 1227 3 SORT_PTR = .USER_SORT_PTR;
426 1228 3
427 1229 3 WHILE .USER_SORT_LEN GTR 0 DO
428 1230 4 BEGIN
429 1231 4
430 1232 4     LOCAL
431 1233 4     CH;
432 1234 4
433 1235 4     CH = CH$RCHAR_A (USER_SORT_PTR);           ! Get next character
434 1236 4     USER_SORT_LEN = .USER_SORT_LEN - 1;       ! One less character in string
435 1237 4
436 1238 4     IF .CH EQL RINTES
437 1239 4     THEN
438 1240 5 BEGIN                                     ! RUNOFF escape sequence
439 1241 5
440 1242 5     CH = CH$RCHAR_A (USER_SORT_PTR);
441 1243 5     CH$RCHAR_A (USER_SORT_PTR);
442 1244 5     USER_SORT_LEN = .USER_SORT_LEN - 2;
443 1245 5
444 1246 5     IF .CH EQL %C'J' THEN EXITLOOP;           ! Subindex sequence signals end of string
445 1247 5
446 1248 5     SORT_LEN = .SORT_LEN + 3;                 ! 3 more characters in sort string
447 1249 5     END
448 1250 4     ELSE
449 1251 4     SORT_LEN = .SORT_LEN + 1;                 ! 1 more character in sort string
450 1252 4
451 1253 4     END;
452 1254 3
453 1255 3 RETURN;
454 1256 2 END;
455 1257 2
456 1258 2 LEN = .I_LEN;
457 1259 2 PTR = .I_PTR;
458 1260 2
459 1261 2 SELECTONE .CMDBLK [NDX$H_NONALPHA] OF
460 1262 2 SET
461 1263 2
462 1264 2 [IGNORE]:
463 1265 3 BEGIN
464 1266 3
465 1267 3     Ignore leading non-alphas
466 1268 3
467 1269 3     LOCAL
468 1270 3     SCAN_PTR,
469 1271 3     FIRST_PTR,
470 1272 3     FIRST_LEN;
471 1273 3
472 1274 3     FIRST_PTR = .PTR;
```

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
SORT_AS -- Build sort string

F 14
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 36
(4)

```

: 473 1275 3 FIRST_LEN = 0;
: 474 1276 3 SCAN_PTR = .PTR;
: 475 1277 3
: 476 1278 3 DECR I FROM .LEN TO 1 DO
: 477 1279 4 BEGIN
: 478 1280 4 LOCAL
: 479 1281 4 CH;
: 480 1282 4
: 481 1283 4 CH = CH$RCHAR_A (SCAN_PTR);
: 482 1284 4
: 483 1285 4 IF .CH EQL RINTES
: 484 1286 4 THEN
: 485 1287 5 BEGIN
: 486 1288 5
: 487 1289 5 RUNOFF escape sequence
: 488 1290 5
: 489 1291 5 IF .FIRST_LEN EQL 0
: 490 1292 5 THEN
: 491 1293 6 BEGIN
: 492 1294 6
: 493 1295 6 Save pointer and length if first escape sequence seen
: 494 1296 6
: 495 1297 6 FIRST_LEN = .I;
: 496 1298 6 FIRST_PTR = CH$PLUS (.SCAN_PTR, -1);
: 497 1299 5 END;
: 498 1300 5
: 499 1301 5 CH$RCHAR_A (SCAN_PTR); ! Skip rest of sequence
: 500 1302 5 CH$RCHAR_A (SCAN_PTR);
: 501 1303 5 I = .I - 2; ! Decrement length remaining
: 502 1304 5 END
: 503 1305 4 ELSE
: 504 1306 5 BEGIN
: 505 1307 5 IF LETTER (.CH)
: 506 1308 5 THEN
: 507 1309 6 BEGIN
: 508 1310 6
: 509 1311 6 Alphabetic character
: 510 1312 6
: 511 1313 6 IF .FIRST_LEN EQL 0
: 512 1314 6 THEN
: 513 1315 7 BEGIN
: 514 1316 7
: 515 1317 7 No RUNOFF escape sequence was seen.
: 516 1318 7 Save pointer and length.
: 517 1319 7
: 518 1320 7 FIRST_LEN = .I;
: 519 1321 7 FIRST_PTR = CH$PLUS (.SCAN_PTR, -1);
: 520 1322 6 END;
: 521 1323 6
: 522 1324 6 EXITLOOP;
: 523 1325 6 END
: 524 1326 5 ELSE
: 525 1327 5 FIRST_LEN = 0;
: 526 1328 4 END;
: 527 1329 3 END;
: 528 1330 3
: 529 1331 3 IF .FIRST_LEN NEQ 0
```


NDXOUT
V04-000

NDXOUT -- Sort and store index entries
SORT_AS -- Build sort string

G 14
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 37
(4)

NDX
V04

```

530      1332 3      THEN
531      1333 4      BEGIN
532      1334 4      |
533      1335 4      |   Found an alphabetic sequence
534      1336 4      |
535      1337 4      |   LEN = .FIRST_LEN;
536      1338 4      |   PTR = .FIRST_PTR;
537      1339 3      |   END;
538      1340 2      |
539      1341 2      |   END;
540      1342 2      [AFTER]:
541      1343 3      BEGIN
542      1344 3      |
543      1345 3      |   Put leading nonalphas after
544      1346 3      |
545      1347 3      |   IF .LEVEL NEQ 0
546      1348 3      |   THEN
547      1349 4      |   BEGIN
548      1350 4      |   |
549      1351 4      |   |   Build a sort string for all but top level entries.
550      1352 4      |   |   Top level entries are sorted after by examining the
551      1353 4      |   |   nonalpha bucket last.
552      1354 4      |   |
553      1355 4      |   |   LOCAL
554      1356 4      |   |   SCAN_PTR;
555      1357 4      |   |
556      1358 4      |   |   SCAN_PTR = .PTR;
557      1359 4      |   |
558      1360 4      |   |   INCR I FROM 1 TO .LEN DO
559      1361 5      |   |   BEGIN
560      1362 5      |   |   |   LOCAL
561      1363 5      |   |   |   CH;
562      1364 5      |   |   |
563      1365 5      |   |   |   CH = CH$RCHAR_A (SCAN_PTR);
564      1366 5      |   |   |
565      1367 5      |   |   |   IF .CH EQL RINTES
566      1368 5      |   |   |   THEN
567      1369 6      |   |   |   BEGIN
568      1370 6      |   |   |   |
569      1371 6      |   |   |   |   RUNOFF escape sequence - skip over it
570      1372 6      |   |   |   |
571      1373 6      |   |   |   |   CH$RCHAR_A (SCAN_PTR);
572      1374 6      |   |   |   |   CH$RCHAR_A (SCAN_PTR);
573      1375 6      |   |   |   |   I = .I + 2;
574      1376 6      |   |   |   |   END
575      1377 5      |   |   |   ELSE
576      1378 6      |   |   |   BEGIN
577      1379 6      |   |   |   |
578      1380 6      |   |   |   |   Have first character
579      1381 6      |   |   |   |
580      1382 7      |   |   |   |   IF NOT LETTER (.CH)
581      1383 6      |   |   |   |   THEN
582      1384 7      |   |   |   |   BEGIN
583      1385 7      |   |   |   |   |
584      1386 7      |   |   |   |   |   Leading nonalpha.
585      1387 7      |   |   |   |   |   Make it sort after by building a sort string
586      1388 7      |   |   |   |   |   which starts with 'zzzz'
```

```
587 1389 7      !
588 1390 7      CH$COPY (4, CH$PTR (UPLIT ('zzzz')), .LEN, .PTR, %C' ', .LEN + 4, CH$PTR (SORT_STR))
589 1391 7
590 1392 7      LEN = .LEN + 4;
591 1393 7      PTR = CH$PTR (SORT_STR);
592 1394 6      END;
593 1395 6
594 1396 6      EXITLOOP;
595 1397 5      END;
596 1398 4      END;
597 1399 3      END;
598 1400 2      END;
599 1401 2
600 1402 2      [BEFORE]:
601 1403 2      |
602 1404 2      | Sort nonalphas before.
603 1405 2      | Since this is the default, no action is required.
604 1406 2      |
605 1407 2
606 1408 2
607 1409 2      TES;
608 1410 2
609 1411 2      IF .LEN NEQ .I_LEN
610 1412 2      THEN
611 1413 3          BEGIN
612 1414 3      |
613 1415 3      | A sort string has been built.
614 1416 3      | Save pointer and length of resulting string
615 1417 3      |
616 1418 3      | SORT_LEN = .LEN;
617 1419 3      | SORT_PTR = .PTR;
618 1420 2      | END;
619 1421 2
620 1422 2      IF NOT .CMDBLK [NDX$V_WORD_SORT]
621 1423 2      THEN
622 1424 3          BEGIN
623 1425 3      |
624 1426 3      | Letter by letter sort - remove whitespace.
625 1427 3      |
626 1428 3      | LOCAL
627 1429 3      | RINTES_PTR,
628 1430 3      | RINTES_LEN,
629 1431 3      | SCAN_PTR;
630 1432 3
631 1433 3      RINTES_PTR = 0;
632 1434 3      RINTES_LEN = 0;
633 1435 3      SCAN_PTR = .PTR;
634 1436 3
635 1437 3      SORT_PTR = CH$PTR (SORT_STR);
636 1438 3      SORT_LEN = 0;
637 1439 3
638 1440 3      INCR I FROM 1 TO .LEN DO
639 1441 4          BEGIN
640 1442 4              LOCAL
641 1443 4              CH;
642 1444 4
643 1445 4              CH = CH$RCHAR_A (SCAN_PTR);
```


NDXOUT
V04-000

NDXOUT -- Sort and store index entries
SORT_AS -- Build sort string

I 14
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 39
(4)

```

644 1446 4
645 1447 4
646 1448 4
647 1449 5
648 1450 5
649 1451 5
650 1452 5
651 1453 5
652 1454 5
653 1455 6
654 1456 6
655 1457 6
656 1458 6
657 1459 6
658 1460 6
659 1461 6
660 1462 5
661 1463 5
662 1464 5
663 1465 5
664 1466 5
665 1467 5
666 1468 5
667 1469 5
668 1470 4
669 1471 5
670 1472 6
671 1473 5
672 1474 6
673 1475 6
674 1476 6
675 1477 6
676 1478 6
677 1479 6
678 1480 7
679 1481 7
680 1482 7
681 1483 7
682 1484 7
683 1485 7
684 1486 7
685 1487 7
686 1488 7
687 1489 7
688 1490 6
689 1491 6
690 1492 5
691 1493 6
692 1494 6
693 1495 6
694 1496 6
695 1497 6
696 1498 6
697 1499 6
698 1500 6
699 1501 6
700 1502 5

IF .CH EQL RINTES
THEN
  BEGIN
    : RUNOFF escape sequence.
    IF .RINTES_LEN EQL 0
    THEN
      BEGIN
        : Not a multiple sequence.
        : Save pointer to beginning of output sequence.
        RINTES_LEN = .SORT_LEN;
        RINTES_PTR = .SORT_PTR;
      END;
    CH$WCHAR_A (.CH, SORT_PTR);
    CH$WCHAR_A (CH$RCHAR_A (SCAN_PTR), SORT_PTR);
    CH$WCHAR_A (CH$RCHAR_A (SCAN_PTR), SORT_PTR);
    SORT_LEN = .SORT_LEN + 3;
    I = .I + 2;
  END
ELSE
  BEGIN
    IF (.CH EQL %C' ') OR (.CH EQL TAB) OR (.CH EQL %C'-' )
    THEN
      BEGIN
        : Whitespace.
        IF .RINTES_PTR NEQ 0
        THEN
          BEGIN
            : Whitespace was emphasized.
            : Remove emphasis from output string
            SORT_PTR = .RINTES_PTR;
            SORT_LEN = .RINTES_LEN;

            RINTES_PTR = 0;
            RINTES_LEN = 0;
          END;
        END
      ELSE
        BEGIN
          : Some other character
          CH$WCHAR_A (.CH, SORT_PTR);
          SORT_LEN = .SORT_LEN + 1;

          RINTES_PTR = 0;
          RINTES_LEN = 0;
        END;
      END
    END
  END
```

NDXOUT
V04-000

NDXOUT -- sort and store index entries
SORT_AS -- Build sort string

J 14
16-Sep-1984 01:04:24
14-Sep-1984 17:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 40
(4)

```
: 701      1503  4      END;  
: 702      1504  3      END;  
: 703      1505  3      SORT_PTR = CH$PTR (SORT_STR);  
: 704      1506  3      END;  
: 705      1507  2      END;  
: 706      1508  2      END;  
: 707      1509  1      END;
```

.PSECT \$PLITS\$,NOWRT,NOEXE,2

7A 7A 7A 7A 00000 P.AAA: .ASCII \zzzz\

.PSECT \$CODE\$,NOWRT,2

```
OFFC 00000 SORT_AS: .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 ; 1162  
SE 00000000' 04 C2 00002 SUBL2 #4, SP ; 1217  
00000000' EF D4 00005 CLRL SORT_LEN ; 1218  
00000000' EF 9E 0000B MOVAB SORT_STR, SORT_PTR ; 1220  
00000000' EF D5 00016 TSTL USER_SORT_LEN ; 1227  
00000000' EF D0 0001E BEQL 5$ ; 1229  
00000000' EF D5 00029 1$: TSTL USER_SORT_LEN ; 1235  
01 14 0002F BGTR 2$ ; 1236  
04 00031 RET ; 1238  
50 00000000' FF 9A 00032 2$: MOVZBL @USER_SORT_PTR, CH ; 1242  
00000000' EF D6 00039 INCL USER_SORT_PTR ; 1243  
00000000' EF D7 0003F DECL USER_SORT_LEN ; 1244  
00000000G 8F 50 D1 00045 CMPL CH, #RINTES ; 1246  
00000000' 2D 12 0004C BNEQ 4$ ; 1248  
50 00000000' FF 9A 0004E MOVZBL @USER_SORT_PTR, CH ; 1251  
00000000' EF D6 00055 INCL USER_SORT_PTR ; 1258  
00000000' EF D6 0005B INCL USER_SORT_LEN ; 1259  
00000000' EF 02 C2 00061 SUBL2 #2, USER_SORT_LEN ; 1261  
00000004A 8F 50 D1 00068 CMPL CH, #74 ; 1264  
01 12 0006F BNEQ 3$ ; 1274  
04 00071 RET ; 1275  
00000000' EF 03 C0 00072 3$: ADDL2 #3, SORT_LEN ; 1276  
AE 11 00079 BRB 1$ ; 1278  
00000000' EF D6 0007B 4$: INCL SORT_LEN ; 1283  
A6 11 00081 BRB 1$ ; 1285  
56 08 AC D0 00083 5$: MOVL I_LEN, LEN ; 1274  
57 04 AC D0 00087 MOVL I_PTR, PTR ; 1275  
50 00000000G EF 32 0008B CMTWL CMDBLK+8, R0 ; 1276  
03 50 B1 00092 CMPW R0, #3 ; 1278  
6F 12 00095 BNEQ 14$ ; 1283  
51 57 D0 00097 MOVL PTR, FIRST_PTR ; 1285  
54 D4 0009A CLRL FIRST_LEN ; 1274  
52 57 D0 0009C MOVL PTR, SCAN_PTR ; 1275  
50 01 A6 9E 0009F MOVAB 1(R6), I ; 1276  
52 11 000A3 BRB 12$ ; 1278  
00000000G 53 82 9A 000A5 6$: MOVZBL (SCAN_PTR)+, CH ; 1283  
8F 53 D1 000A8 CMPL CH, #RINTES ; 1285  
13 12 000AF BNEQ 8$ ; 1285
```

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
SORT_AS -- Build sort string

K 14
16-Sep-1984 01:04:24 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:07:15 [RUNOFF.SRC]NDXOUT.BLI;1

Page 41
(4)

		54	D5	000B1	TSTL	FIRST_LEN	1291
		07	12	000B3	BNEQ	7\$	
	54	50	D0	000B5	MOVL	I, FIRST_LEN	1297
	51	A2	9E	000B8	MOVAB	-1(R2), FIRST_PTR	1298
	52	02	C0	000BC	ADDL2	#2, SCAN_PTR	1302
	50	02	C2	000BF	SUBL2	#2, I	1303
		33	11	000C2	BRB	12\$	1285
00000041	8F	53	D1	000C4	CMPL	CH, #65	1307
		09	19	000CB	BLSS	9\$	
0000005A	8F	53	D1	000CD	CMPL	CH, #90	
		12	15	000D4	BLEQ	10\$	
00000061	8F	53	D1	000D6	CMPL	CH, #97	
		16	19	000DD	BLSS	11\$	
0000007A	8F	53	D1	000DF	CMPL	CH, #122	
		0D	14	000E6	BGTR	11\$	
		54	D5	000E8	TSTL	FIRST_LEN	1313
		0E	12	000EA	BNEQ	13\$	
	54	50	D0	000EC	MOVL	I, FIRST_LEN	1320
	51	A2	9E	000EF	MOVAB	-1(R2), FIRST_PTR	1321
		05	11	000F3	BRB	13\$	1309
		54	D4	000F5	CLRL	FIRST_LEN	1327
	AB	50	F5	000F7	SOBGTR	I, 6\$	1278
		54	D5	000FA	TSTL	FIRST_LEN	1331
		56	13	000FC	BEQL	18\$	
	56	54	D0	000FE	MOVL	FIRST_LEN, LEN	1337
	57	51	D0	00101	MOVL	FIRST_PTR, PTR	1338
		7D	11	00104	BRB	21\$	1261
	02	50	B1	00106	CMPW	R0, #2	1342
		7E	12	00109	BNEQ	23\$	
		AC	D5	0010B	TSTL	LEVEL	1347
		79	13	0010E	BEQL	23\$	
	6E	57	D0	00110	MOVL	PTR, SCAN_PTR	1358
	5B	56	D0	00113	MOVL	LEN, R11	1360
		5A	D4	00116	CLRL	I	
		6B	11	00118	BRB	22\$	
	50	BE	9A	0011A	MOVZBL	@SCAN_PTR, CH	1365
		6E	D6	0011E	INCL	SCAN_PTR	
00000000G	8F	50	D1	00120	CMPL	CH, #RINTES	1367
		09	12	00127	BNEQ	16\$	
		6E	D6	00129	INCL	SCAN_PTR	1373
		6E	D6	0012B	INCL	SCAN_PTR	1374
		02	C0	0012D	ADDL2	#2, I	1375
	5A	53	11	00130	BRB	22\$	1367
00000041	8F	50	D1	00132	CMPL	CH, #65	1382
		09	19	00139	BLSS	17\$	
0000005A	8F	50	D1	0013B	CMPL	CH, #90	
		45	15	00142	BLEQ	23\$	
00000061	8F	50	D1	00144	CMPL	CH, #97	
		09	19	0014B	BLSS	19\$	
0000007A	8F	50	D1	0014D	CMPL	CH, #122	
		33	15	00154	BLEQ	23\$	
	59	A6	9E	00156	MOVAB	4(R6), R9	1390
	58	EF	9E	0015A	MOVAB	SORT_STR, R8	
59	20	04	2C	00161	MOVCS	#4, P.AAA, #32, R9, (R8)	
		68		0016A			
		0C	18	0016B	BGEQ	20\$	
	58	04	C0	0016D	ADDL2	#4, R8	

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
SORT_AS -- Build sort string

L 14
16-Sep-1984 01:04:24 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:07:15 [RUNOFF.SRC]NDXOUT.BLI;1

Page 42
(4)

NDX
V04

59	20	59	04	C2	00170	SUBL2	#4, R9		
		67	56	2C	00173	MOVCS	LEN, (PTR), #32, R9, (R8)		
			68		00178				
		56	04	C0	00179	20\$: ADDL2	#4, LEN	1392	
		57	EF	9E	0017C	MOVAB	SORT_STR, PTR	1393	
			04	11	00183	21\$: BRB	23\$	1378	
91		5A	5B	F3	00185	22\$: AOBLEQ	R11, I, 15\$	1360	
	08	AC	56	D1	00189	23\$: CMPL	LEN, I_LEN	1411	
			0E	13	0018D	BEQL	24\$		
		00000000'	EF	56	D0	0018F	MOVL	LEN, SORT_LEN	1418
		00000000'	EF	57	D0	00196	MOVL	PTR, SORT_PTR	1419
		01	EF	E9	0019D	24\$: BLBC	CMDBLK+1, -25\$	1422	
			04	001A4	RET				
			51	7C	001A5	25\$: CLRQ	RINTES_LEN	1434	
		54	57	D0	001A7	MOVL	PTR, SCAN_PTR	1435	
		00000000'	EF	9E	001AA	MOVAB	SORT_STR, -SORT_PTR	1437	
			EF	D4	001B5	CLRL	SORT_LEN	1438	
			53	D4	001BB	CLRL	I	1440	
			4F	11	001BD	BRB	28\$		
		50	84	9A	001BF	26\$: MOVZBL	(SCAN_PTR)+, CH	1445	
		00000000G	8F	50	D1	001C2	CMPL	CH, #RINTES	1447
			45	12	001C9	BNEQ	29\$		
			51	D5	001CB	TSTL	RINTES_LEN	1453	
			0E	12	001CD	BNEQ	27\$		
		51	EF	D0	001CF	MOVL	SORT_LEN, RINTES_LEN	1460	
		52	EF	D0	001D6	MOVL	SORT_PTR, RINTES_PTR	1461	
		00000000'	FF	50	90	001DD	27\$: MOVAB	CH, @SORT_PTR	1464
			EF	D6	001E4	INCL	SORT_PTR		
		00000000'	FF	84	90	001EA	MOVB	(SCAN_PTR)+, @SORT_PTR	1465
			EF	D6	001F1	INCL	SORT_PTR		
		00000000'	FF	84	90	001F7	MOVB	(SCAN_PTR)+, @SORT_PTR	1466
			EF	D6	001FE	INCL	SORT_PTR		
		00000000'	EF	03	C0	00204	ADDL2	#3, SORT_LEN	1467
			53	02	C0	0020B	ADDL2	#2, I	1468
			3C	11	0020E	28\$: BRB	33\$	1447	
		20	50	D1	00210	29\$: CMPL	CH, #32	1472	
			0E	13	00213	BEQL	30\$		
		00000000G	8F	50	D1	00215	CMPL	CH, #TAB	
			05	13	0021C	BEQL	30\$		
		2D	50	D1	0021E	CMPL	CH, #45		
			14	12	00221	BNEQ	31\$		
			52	D5	00223	30\$: TSTL	RINTES_PTR	1478	
			25	13	00225	BEQL	33\$		
		00000000'	EF	52	D0	00227	MOVL	RINTES_PTR, SORT_PTR	1485
		00000000'	EF	51	D0	0022E	MOVL	RINTES_LEN, SORT_LEN	1486
			13	11	00235	BRB	32\$	1488	
		00000000'	FF	50	90	00237	31\$: MOVAB	CH, @SORT_PTR	1497
			EF	D6	0023E	INCL	SORT_PTR		
		00000000'	EF	D6	00244	INCL	SORT_LEN	1498	
			51	7C	0024A	32\$: CLRQ	RINTES_LEN	1501	
FF6D			56	F1	0024C	33\$: ACBL	LEN, #T, I, 26\$	1440	
	53	00000000'	EF	9E	00252	MOVAB	SORT_STR, SORT_PTR	1506	
			04	0025D	RET			1509	

; Routine Size: 606 bytes, Routine Base: \$CODE\$ + 011D

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
FIND_POS -- Locate position for insertion

M 14
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 43
(5)

```

709 1510 1 %SBTTL 'FIND_POS -- Locate position for insertion'
710 1511 1
711 1512 1 ROUTINE FIND_POS (STG_PTR, STG_LEN, SUB_CNT, LAST, XTN) : NOVALUE =
712 1513 1
713 1514 1 ++
714 1515 1 FUNCTIONAL DESCRIPTION:
715 1516 1
716 1517 1     Locate the proper position in the master list for placing a new
717 1518 1     item. Also make sure the item is not a complete duplicate of an
718 1519 1     existing entry.
719 1520 1
720 1521 1 FORMAL PARAMETERS:
721 1522 1
722 1523 1     STG_PTR - Address of input text.
723 1524 1     STG_LEN - Length of input text.
724 1525 1     SUB_CNT - Sub-index level (0 to n)
725 1526 1     LAST   - TRUE if this is the last call to FIND_POS for this entry
726 1527 1     XTN    - Transaction number if LAST = TRUE
727 1528 1
728 1529 1 IMPLICIT INPUTS:
729 1530 1
730 1531 1     CELL           - Characteristics of current position in list
731 1532 1     SORT_LEN      - Length of sort string if any
732 1533 1     SORT_PTR      - Pointer to sort string if any
733 1534 1
734 1535 1 IMPLICIT OUTPUTS:
735 1536 1
736 1537 1     CELL           - set up for insertion
737 1538 1
738 1539 1 ROUTINE VALUE:
739 1540 1 COMPLETION CODES:
740 1541 1
741 1542 1     NONE.
742 1543 1
743 1544 1 SIDE EFFECTS:
744 1545 1
745 1546 1     NONE
746 1547 1
747 1548 1 --
748 1549 1
749 1550 2 BEGIN
750 1551 2
751 1552 2 LOCAL
752 1553 2     LINE_PTR,
753 1554 2     LINE_LEN;
754 1555 2
755 1556 2 IF .SORT_LEN NEQ 0
756 1557 2 THEN
757 1558 2     BEGIN
758 1559 2     :
759 1560 2     : Have a sort string to use
760 1561 2     :
761 1562 2     LINE_PTR = .SORT_PTR;
762 1563 2     LINE_LEN = .SORT_LEN;
763 1564 2     END
764 1565 2 ELSE
765 1566 2 BEGIN
```

```
766 1567      | no sort string - use entry text
767 1568
768 1569
769 1570      | LINE_PTR = .STG_PTR;
770 1571      | LINE_LEN = .STG_LEN;
771 1572      | END;
772 1573
773 1574      |
774 1575      | Skip the bucket positioning for sub-indexes
775 1576
776 1577      | IF .SUB_CNT EQL 0
777 1578      | THEN
778 1579      | BEGIN
779 1580      |
780 1581      | The first character that is not a special sequence determines
781 1582      | the bucket number.
782 1583
783 1584      | LOCAL
784 1585      | BUCKET_NUMBER,
785 1586      | SUB_BUCKET;
786 1587
787 1588      | BUCKET_NUMBER = FIND_BUCKET (LINE_LEN, LINE_PTR);
788 1589
789 1590      | IF .BUCKET_NUMBER NEQ 0
790 1591      | THEN
791 1592      |
792 1593      | Use the second character in the string to determine the
793 1594      | sub-bucket number unless the first character in the string was a
794 1595      | nonalphabetic.
795 1596
796 1597      | SUB_BUCKET = FIND_BUCKET (LINE_LEN, LINE_PTR)
797 1598      | ELSE
798 1599      |
799 1600      | Nonalphabetic characters are always sorted using a single bucket
800 1601      | because the 'squared bucket' algorithm does not work for them.
801 1602
802 1603      | SUB_BUCKET = 0;
803 1604
804 1605      |
805 1606      | Now remember all of the information needed for future use.
806 1607
807 1608      | CELL [CSA_HEAD] = BUCKET [.BUCKET_NUMBER, .SUB_BUCKET];
808 1609      | CELL [CSA_CURR] = .BUCKET [.BUCKET_NUMBER, .SUB_BUCKET];
809 1610      | CELL [CSA_PREV] = 0;
810 1611      | CELL [CSV_IDNS] = FALSE;
811 1612      | END;
812 1613
813 1614      |
814 1615      | Now find the proper position for insertion
815 1616
816 1617      | REPEAT
817 1618      | BEGIN
818 1619      |
819 1620      | LOCAL
820 1621      | CUR_CELL : REF $XE_BLOCK;
821 1622
822 1623      |
```



```

823 1624 3      | Point to data in storage
824 1625 3      |
825 1626 3      | CUR_CELL = .CELL [CSA_CURR];
826 1627 3      |
827 1628 3      |
828 1629 3      | If this is the last item, return current position
829 1630 3      |
830 1631 3      | IF (.CUR_CELL[XESA_NEXT] EQL 0) AND (.SUB_CNT EQL .CUR_CELL [XESH_SUBC]) THEN RETURN;
831 1632 3      |
832 1633 3      |
833 1634 3      | See if we are at the correct position for an insertion
834 1635 3      |
835 1636 3      | IF .SUB_CNT GTR .CUR_CELL [XESH_SUBC] THEN RETURN;
836 1637 3      |
837 1638 3      | IF ENTRY_CMP (.STG_PTR, .STG_LEN, .LAST, .XTN, .SUB_CNT) THEN RETURN;
838 1639 3      |
839 1640 3      |
840 1641 3      | Make sure we still point at original data
841 1642 3      |
842 1643 3      | CUR_CELL = .CELL [CSA_CURR];
843 1644 3      |
844 1645 3      |
845 1646 3      | Advance to next location
846 1647 3      |
847 1648 3      | CELL [CSA_CURR] = .CUR_CELL [XESA_NEXT];
848 1649 3      | END;
849 1650 3      |
850 1651 1      | END;

```

!End of FIND_POS

```

                                007C 00000 FIND_POS:
                                .WORD
56 00000000G EF 9E 00002      MOVAB      Save R2,R3,R4,R5,R6      : 1512
55 00000000V EF 9E 00009      MOVAB      BUCKET, R6
54 00000000' EF 9E 00010      MOVAB      FIND_BUCKET, R5
5E          08 C2 00017      MOVAB      CELL, R4
50          04C4 C4 D0 0001A      SUBL2      #8, SP
04 6E          08 13 0001F      MOVL      SORT_LEN, R0      : 1556
AE          04C0 C4 D0 00021      BEQL      1$
53          0C AC 7D 0002C 1$:  MOVQ      SORT_PTR, LINE_PTR
04 6E          0C AC D0 00030 2$:  MOVL      R0, LINE_LEN      : 1562
53          0C AC D0 00030 2$:  BRB      2$      : 1563
04 6E          0C AC 7D 0002C 1$:  MOVQ      STG_PTR, LINE_PTR      : 1556
53          0C AC D0 00030 2$:  MOVL      SUB_CNT, R3      : 1570
04 6E          0C AC 7D 0002C 1$:  BNEQ      5$      : 1577
53          0C AC D0 00030 2$:  PUSHL      SP      : 1588
04 6E          0C AC 7D 0002C 1$:  PUSHAB     LINE_LEN
53          0C AC D0 00030 2$:  CALLS      #2, FIND_BUCKET
04 6E          0C AC 7D 0002C 1$:  MOVL      R0, BUCKET_NUMBER      : 1590
53          0C AC D0 00030 2$:  BEQL      3$      : 1597
04 6E          0C AC 7D 0002C 1$:  PUSHL      SP
53          0C AC D0 00030 2$:  PUSHAB     LINE_LEN
04 6E          0C AC 7D 0002C 1$:  CALLS      #2, FIND_BUCKET
53          0C AC D0 00030 2$:  MOVL      R0, SUB_BUCKET
04 6E          0C AC 7D 0002C 1$:  BRB      4$
53          0C AC D0 00030 2$:

```

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
FIND_POS -- Locate position for insertion

C 15
16-Sep-1984 01:04:24 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:07:15 [RUNOFF.SRC]NDXOUT.BLI;1

Page 46
(5)

NDX
V04

				51	D4	00050	3\$:	CLRL	SUB_BUCKET	:	1603	
				18	C5	00052	4\$:	MULL3	#27, BUCKET_NUMBER, R0	:	1608	
				51	C0	00056		ADDL2	SUB_BUCKET, R0	:		
				6640	DE	00059		MOVAL	BUCKET[R0], CELL+8	:		
				6640	D0	0005E		MOVL	BUCKET[R0], CELL	:	1609	
				04	A4	D4	00062	CLRL	CELL+4	:	1610	
					01	8A	00065	BICB2	#1, CELL+12	:	1611	
					64	D0	00069	5\$:	MOVL	CELL, CUR_CELL	:	1626
				04	A2	D5	0006C	TSTL	4(CUR_CELL)	:	1631	
					08	12	0006F	BNEQ	6\$	:		
53	18	A2			00	EC	00071	CMPV	#0, #16, 24(CUR_CELL), R3	:		
					25	13	00077	BEQL	7\$	:		
53	18	A2			00	EC	00079	6\$:	CMPV	#0, #16, 24(CUR_CELL), R3	:	1636
					1D	19	0007F	BLSS	7\$	:		
					53	DD	00081	PUSHL	R3	:	1638	
					AC	7D	00083	MOVQ	LAST, -(SP)	:		
					04	AC	7D	00087	MOVQ	STG_PTR, -(SP)	:	
					05	FB	0008B	CALLS	#5, ENTRY_CMP	:		
					50	E8	00092	BLBS	R0, 7\$	:		
					64	D0	00095	MOVL	CELL, CUR_CELL	:	1643	
					04	A2	D0	00098	MOVL	4(CUR_CELL), CELL	:	1648
					CB	11	0009C	BRB	5\$	:	1612	
					04	0009E	7\$:	RET		:	1651	

; Routine Size: 159 bytes, Routine Base: \$CODE\$ + 037B

; R


```
852 1652 1 %SBTTL 'FIND_BUCKET -- Get bucket number'
853 1653 1 ROUTINE FIND_BUCKET (LEN_A, PTR_A) =
854 1654 1 ++
855 1655 1
856 1656 1 FUNCTIONAL DESCRIPTION:
857 1657 1
858 1658 1 This routine is called to determine the bucket number of the first
859 1659 1 character in a string which is not a special sequence
860 1660 1
861 1661 1 FORMAL PARAMETERS:
862 1662 1
863 1663 1 LEN_A - Address of a variable which contains the length of the string
864 1664 1 The value is updated to reflect the number of unscanned
865 1665 1 characters in the string.
866 1666 1 PTR_A - Address of a variable which contains a CH$PTR to the string
867 1667 1 The value is updated to point to the first unscanned
868 1668 1 character in the string.
869 1669 1
870 1670 1 IMPLICIT INPUTS:
871 1671 1
872 1672 1 None
873 1673 1
874 1674 1 IMPLICIT OUTPUTS:
875 1675 1
876 1676 1 None
877 1677 1
878 1678 1 ROUTINE VALUE:
879 1679 1 COMPLETION CODES:
880 1680 1
881 1681 1 Returns a value from 0 to 26 indicating the bucket number.
882 1682 1 (0 = nonalpha, 1 = A, ... 26 = Z)
883 1683 1
884 1684 1 SIDE EFFECTS:
885 1685 1
886 1686 1 None
887 1687 1 --
888 1688 2 BEGIN
889 1689 2
890 1690 2 LOCAL
891 1691 2 CH;
892 1692 2
893 1693 2 BIND
894 1694 2 LEN = .LEN_A,
895 1695 2 PTR = .PTR_A;
896 1696 2
897 1697 2 CH = 0;
898 1698 2
899 1699 2 WHILE .LEN GT 0 DO
900 1700 3 BEGIN
901 1701 3
902 1702 3 Get the first character that is not a special sequence
903 1703 3
904 1704 3 CH = CH$RCHAR A (PTR);
905 1705 3 LEN = .LEN - 1;
906 1706 3
907 1707 3 IF .CH EQL RINTES
908 1708 3 THEN
```

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
FIND_BUCKET -- Get bucket number

E 15
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 48
(6)

```

: 909      1709  4      BEGIN
: 910      1710  4
: 911      1711  4      Skip special sequence
: 912      1712  4
: 913      1713  4      CH$RCHAR_A (PTR);
: 914      1714  4      CH$RCHAR_A (PTR);
: 915      1715  4      LEN = .LEN - 2;
: 916      1716  4      END
: 917      1717  3      ELSE
: 918      1718  4      BEGIN
: 919      1719  4
: 920      1720  4      Some other character
: 921      1721  4
: 922      1722  4      IF LOWER_LETTER (.CH) THEN CH = UPPER_CASE (.CH);
: 923      1723  4
: 924      1724  4      EXITLOOP;
: 925      1725  3      END;
: 926      1726  3
: 927      1727  2      END;
: 928      1728  2
: 929      1729  2
: 930      1730  2      Using the first non-special character, figure out which index
: 931      1731  2      bucket is the right one to look into. Buckets 1 through 26 are
: 932      1732  2      alphabetic, and all other characters belong in bucket 0.
: 933      1733  2
: 934      1734  2      RETURN (IF (.CH GEQ %C'A') AND (.CH LEQ %C'Z') THEN (.CH - %C'A' + 1) ELSE 0);
: 935      1735  1      END;

```

		0004 00000 FIND_BUCKET:				
	51	04	AC D0 00002	.WORD	Save R2	1653
			50 D4 00006	MOVL	LEN_A, R1	1694
			61 D5 00008	CLRL	CH	1697
			35 15 0000A	1\$: TSTL	(R1)	1699
	52	08	BC D0 0000C	BLEQ	3\$	
	50		62 9A 00010	MOVL	@PTR_A, R2	1704
		08	BC D6 00013	MOVZBL	(R2), CH	
			61 D7 00016	INCL	@PTR_A	
			50 D1 00018	DECL	(R1)	1705
00000000G	8F		08 12 0001F	CMPL	CH, #RINTES	1707
		08	BC D6 00021	BNEQ	2\$	
		08	BC D6 00024	INCL	@PTR_A	1713
	61		02 C2 00027	INCL	@PTR_A	1714
			DC 11 0002A	SUBL2	#2, (R1)	1715
00000061	8F		50 D1 0002C	BRB	1\$	1707
			0C 19 00033	2\$: CMPL	CH, #97	1722
0000007A	8F		50 D1 00035	BLSS	3\$	
		03	14 0003C	CMPL	CH, #122	
	50		20 C2 0003E	BGTR	3\$	
00000041	8F		50 D1 00041	SUBL2	#32, CH	
			0E 19 00048	3\$: CMPL	CH, #65	1734
0000005A	8F		50 D1 0004A	BLSS	4\$	
		05	14 00051	CMPL	CH, #90	
				BGTR	4\$	

```
NDXOUT -- Sort and store index entries
FIND_BUCKET -- Get bucket number
```

F 15
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 49
(6)

```

50      CO    A0    9E 00053      MOVAB    -64(R0), R0
          04 00057      RET
          50    D4 00058 4$:      CLRL      R0
          04 0005A      RET

```

; Routine Size: 91 bytes, Routine Base: \$CODES + 041A

NDX
V04

.....

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
INSERT_INX -- Insert index item into list

G 15
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 50
(7)

```

937 1736 1 %SBTTL 'INSERT_INX -- Insert index item into list'
938 1737 1
939 1738 1 ROUTINE INSERT_INX (STRING, LENGTH, SUB_CNT, XTN, BAR, ENT_LEN, ENT_PTR) : NOVALUE =
940 1739 1
941 1740 1 ++
942 1741 1 FUNCTIONAL DESCRIPTION:
943 1742 1
944 1743 1     Insert an index item into the index list.
945 1744 1
946 1745 1 FORMAL PARAMETERS:
947 1746 1
948 1747 1     STRING - CH$PTR to the string associated with the item.
949 1748 1             (zero is allowed).
950 1749 1     LENGTH - Length of the passed string.
951 1750 1     SUB_CNT - Sub-index level of item (0 to n)
952 1751 1     XTN - Transaction number of the page associated with this
953 1752 1             index item.
954 1753 1     BAR - Change bar flag
955 1754 1     ENT_LEN - Length of whole index entry
956 1755 1     ENT_PTR - CH$PTR to whole index entry string
957 1756 1
958 1757 1 IMPLICIT INPUTS:
959 1758 1
960 1759 1     CELL - Information table about current position in list.
961 1760 1     BOOKID - Master index book ident string address
962 1761 1     SORT_LEN - Length of sort string if any
963 1762 1     SORT_PTR - Pointer to sort string if any
964 1763 1
965 1764 1 IMPLICIT OUTPUTS:
966 1765 1
967 1766 1     NONE
968 1767 1
969 1768 1 ROUTINE VALUE:
970 1769 1 COMPLETION CODES:
971 1770 1
972 1771 1     NONE
973 1772 1
974 1773 1 SIDE EFFECTS:
975 1774 1
976 1775 1     NONE
977 1776 1
978 1777 1 --
979 1778 1
980 1779 2 BEGIN
981 1780 2 LOCAL
982 1781 2     XMREF : $XM_BLOCK;
983 1782 2
984 1783 2     |
985 1784 2     | Build book reference entry
986 1785 2
987 1786 2     XMREF [XMSA_LINK] = 0;
988 1787 2     XMREF [XMSA_BOOK] = .BOOKID;
989 1788 2
990 1789 2     |
991 1790 2     | Check for existing entry
992 1791 2
993 1792 2     IF .CELL [CSV_IDNS]
```

```

: 994      1793 2      THEN
: 995      1794 2      BEGIN
: 996      1795 2      :
: 997      1796 2      : Identical string
: 998      1797 2      :
: 999      1798 2      LOCAL
1000      1799 2      XE_TEMP : REF $XE_BLOCK,
1001      1800 2      XM_TEMP : REF $XM_BLOCK;
1002      1801 2      :
1003      1802 2      :
1004      1803 2      : Get current cell
1005      1804 2      :
1006      1805 2      XE_TEMP = .CELL [CSA_CURR];
1007      1806 2      :
1008      1807 2      :
1009      1808 2      : Get first entry in book list chain
1010      1809 2      :
1011      1810 2      XM_TEMP = .XE_TEMP [XESA_BOOK_LIST];
1012      1811 2      :
1013      1812 2      REPEAT
1014      1813 2      BEGIN
1015      1814 2      :
1016      1815 2      : Walk book list chain until we either find a reference to
1017      1816 2      : the current book or until the end of chain
1018      1817 2      :
1019      1818 2      IF .XM_TEMP [XMSA_BOOK] EQL .BOOKID THEN EXITLOOP;
1020      1819 2      :
1021      1820 2      IF .XM_TEMP [XMSA_LINK] EQL 0
1022      1821 2      THEN
1023      1822 2      BEGIN
1024      1823 2      :
1025      1824 2      : End of chain - insert a new book reference
1026      1825 2      :
1027      1826 2      XM_TEMP [XMSA_LINK] = SAVDAT (XMREF, DS_XM_ENTRY, XMSK_LENGTH);
1028      1827 2      EXITLOOP;
1029      1828 2      END
1030      1829 2      ELSE
1031      1830 2      XM_TEMP = .XM_TEMP [XMSA_LINK];
1032      1831 2      END;
1033      1832 2      :
1034      1833 2      IF .XTN NEQ 0
1035      1834 2      THEN
1036      1835 2      BEGIN
1037      1836 2      :
1038      1837 2      : There is a page pointer, so attach it.
1039      1838 2      :
1040      1839 2      LOCAL
1041      1840 2      XX_TEMP : REF $XX_BLOCK;
1042      1841 2      :
1043      1842 2      IF .XE_TEMP [XESA_REF] NEQ 0
1044      1843 2      THEN
1045      1844 2      BEGIN
1046      1845 2      :
1047      1846 2      : Entry has references
1048      1847 2      :
1049      1848 2      LOCAL
1050      1849 2      RANGE_BOOK,
```

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
INSERT_INX -- Insert index item into list

I 15
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 52
(7)

```

: 1051      1850 5      RANGE_ACTIVE;
: 1052      1851 5
: 1053      1852 5      RANGE_ACTIVE = FALSE;          ! Have not seen a BEGIN yet
: 1054      1853 5      RANGE_BOOK = 0;
: 1055      1854 5      XX_TEMP = .XE_TEMP [XESA_REF]; ! Get the start of the chain
: 1056      1855 5
: 1057      1856 5      REPEAT
: 1058      1857 6      BEGIN
: 1059      1858 6
: 1060      1859 6      ! Find the chain end
: 1061      1860 6
: 1062      1861 6      XX_TEMP = .XX_TEMP;
: 1063      1862 6
: 1064      1863 6      IF .XX_TEMP [XX$V_BEGIN]
: 1065      1864 6      THEN
: 1066      1865 7      BEGIN
: 1067      1866 7
: 1068      1867 7      ! Beginning of page range
: 1069      1868 7
: 1070      1869 7      RANGE_ACTIVE = TRUE;
: 1071      1870 7      RANGE_BOOK = .XX_TEMP [XX$A_BOOK]; ! Save book identifier.
: 1072      1871 6      END;
: 1073      1872 6
: 1074      1873 6      IF .XX_TEMP [XX$V_END]
: 1075      1874 6      OR .XX_TEMP [XX$A_BOOK] NEQ .RANGE_BOOK
: 1076      1875 6      THEN
: 1077      1876 6
: 1078      1877 6      ! A range ends when either an END is encountered
: 1079      1878 6      ! or when we switch books
: 1080      1879 6
: 1081      1880 6      RANGE_ACTIVE = FALSE;
: 1082      1881 6
: 1083      1882 6
: 1084      1883 6      ! Check for end of chain
: 1085      1884 6
: 1086      1885 6      IF .XX_TEMP [XX$A_LINK] NEQ 0
: 1087      1886 6      THEN
: 1088      1887 6      XX_TEMP = .XX_TEMP [XX$A_LINK]
: 1089      1888 6
: 1090      1889 6      ELSE
: 1091      1890 6      EXITLOOP;
: 1092      1891 5
: 1093      1892 5      END;
: 1094      1893 5      IF .RANGE_ACTIVE
: 1095      1894 5      THEN
: 1096      1895 6      BEGIN
: 1097      1896 6
: 1098      1897 6      ! Saw a BEGIN with no END
: 1099      1898 6
: 1100      1899 6      IF .XPLBLK [XPL$V_VALID] AND .XPLBLK [XPL$V_BEGIN]
: 1101      1900 6      THEN
: 1102      1901 7      BEGIN
: 1103      1902 7
: 1104      1903 7      ! Have a BEGIN inside a BEGIN
: 1105      1904 7
: 1106      1905 7      ! Signal errors for BLISS32
: 1107      1906 7      XIF XBLISS (BLISS32)
: 1107      1906 7      XTHEN
```


NDXOUT
V04-000

NDXOUT -- Sort and store index entries
INSERT_INX -- Insert index item into list

J 15
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 53
(7)

```

: 1108      1907 7
: 1109      1908 7          SIGNAL (INDEX$_DUPBEGIN);
: 1110      1909 7
: 1111      U 1910 7 %ELSE          ! Use $XPO_PUT_MSG otherwise
: 1112      U 1911 7
: 1113      U 1912 7          $XPO_PUT MSG (SEVERITY = WARNING,
: 1114      U 1913 7          STRING = 'duplicate .XPLUS (BEGIN) -- inserted as .XPLUS (')');
: 1115      U 1914 7
: 1116      1915 7 %FI
: 1117      1916 7
: 1118      1917 7          DMPENT (.ENT_LEN, .ENT_PTR);
: 1119      1918 7          XPLBLK [XPL$V_BEGIN] = FALSE;
: 1120      1919 6          END;
: 1121      1920 6      ELSE
: 1122      1921 5          : Have no unmatched BEGIN's
: 1123      1922 5          :
: 1124      1923 5          IF .XPLBLK [XPL$V_VALID] AND .XPLBLK [XPL$V_END]
: 1125      1924 5          THEN
: 1126      1925 5              BEGIN
: 1127      1926 5              : Have an END with no BEGIN
: 1128      1927 6              :
: 1129      1928 6
: 1130      1929 6
: 1131      1930 6
: 1132      L 1931 6 %IF %BLISS (BLISS32)
: 1133      1932 6 %THEN          ! Signal errors for BLISS32
: 1134      1933 6
: 1135      1934 6          SIGNAL (INDEX$_NOBEGIN);
: 1136      1935 6
: 1137      U 1936 6 %ELSE          ! Use $XPO_PUT_MSG otherwise
: 1138      U 1937 6
: 1139      U 1938 6          $XPO_PUT MSG (SEVERITY = WARNING,
: 1140      U 1939 6          STRING = '.XPLUS (END) with no .XPLUS (BEGIN) -- inserted as .XPLUS (')');
: 1141      U 1940 6
: 1142      1941 6 %FI
: 1143      1942 6
: 1144      1943 6          DMPENT (.ENT_LEN, .ENT_PTR);
: 1145      1944 6          XPLBLK [XPL$V_END] = FALSE;
: 1146      1945 5          END;
: 1147      1946 5
: 1148      1947 5
: 1149      1948 5          : Add new pointer to entry and update it in memory
: 1150      1949 5          :
: 1151      1950 5          XX TEMP [XX$A_LINK] = INSERT_REF (.XTN);
: 1152      1951 5          END
: 1153      1952 4      ELSE
: 1154      1953 5          BEGIN
: 1155      1954 5          :
: 1156      1955 5          : Entry has no references.
: 1157      1956 5          :
: 1158      1957 5          IF .XPLBLK [XPL$V_VALID] AND .XPLBLK [XPL$V_END]
: 1159      1958 5          THEN
: 1160      1959 6              BEGIN
: 1161      1960 6              :
: 1162      1961 6              : Have an END with no BEGIN
: 1163      1962 6              :
: 1164      L 1963 6 %IF %BLISS (BLISS32)
```

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
INSERT_INX -- Insert index item into list

K 15
16-Sep-1984 01:04:24 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:07:15 [RUNOFF.SRC]NDXOUT.BLI;1

Page 54
(7)

```

: 1165      1964  6 XTHEN                                ! Signal errors for BLISS32
: 1166      1965  6
: 1167      1966  6                                SIGNAL (INDEX$_NOBEGIN);
: 1168      1967  6
: 1169      1968  6 XELSE                                ! Use $XPO_PUT_MSG otherwise
: 1170      1969  6
: 1171      1970  6                                $XPO_PUT_MSG (SEVERITY = WARNING,
: 1172      1971  6                                STRING = '.XPLUS (END) with no .XPLUS (BEGIN) -- inserted as .XPLUS ()');
: 1173      1972  6
: 1174      1973  6 XFI
: 1175      1974  6
: 1176      1975  6                                DMPENT (.ENT_LEN, .ENT_PTR);
: 1177      1976  6                                XPLBLK [XPL$_END] = FALSE;
: 1178      1977  5                                END;
: 1179      1978  5
: 1180      1979  5
: 1181      1980  5                                ! Point entry to reference and update it in memory.
: 1182      1981  5
: 1183      1982  5                                XE TEMP [XES$_REF] = INSERT_REF (.XTN);
: 1184      1983  4                                END;
: 1185      1984  4
: 1186      1985  3                                END;
: 1187      1986  3                                END
: 1188      1987  2                                ELSE
: 1189      1988  2                                BEGIN
: 1190      1989  3                                ! String is different, insert new string
: 1191      1990  3
: 1192      1991  3                                LOCAL
: 1193      1992  3                                REF_PTR,
: 1194      1993  3                                LAST_CELL,
: 1195      1994  3                                NEXT_CELL,
: 1196      1995  3                                TEMP : REF $XE_BLOCK,
: 1197      1996  3                                TEMP1,
: 1198      1997  3                                TEMP_CELL : $XE_BLOCK;
: 1199      1998  3
: 1200      1999  3
: 1201      2000  3                                ! Get links to chain
: 1202      2001  3
: 1203      2002  3                                TEMP = .CELL [CSA_CURR];
: 1204      2003  3
: 1205      2004  3                                IF .SUB_CNT EQL .TEMP [XES$_SUBC]
: 1206      2005  3                                THEN
: 1207      2006  3                                BEGIN
: 1208      2007  4                                NEXT_CELL = .CELL [CSA_CURR];
: 1209      2008  4                                LAST_CELL = .TEMP [XES$_PREV]
: 1210      2009  4                                END
: 1211      2010  4                                ELSE
: 1212      2011  3                                BEGIN
: 1213      2012  4                                NEXT_CELL = 0;
: 1214      2013  4                                LAST_CELL = .CELL [CSA_CURR]
: 1215      2014  4                                END;
: 1216      2015  3
: 1217      2016  3
: 1218      2017  3                                IF .XTN NEQ 0
: 1219      2018  3                                THEN
: 1220      2019  4                                BEGIN
: 1221      2020  4                                !

```



```
: 1222      2021      4      | Have a page reference
: 1223      2022      4
: 1224      2023      4      | IF .XPLBLK [XPL$V_VALID] AND .XPLBLK [XPL$V_END]
: 1225      2024      4      | THEN
: 1226      2025      5      | BEGIN
: 1227      2026      5
: 1228      2027      5      | Have an END with no BEGIN
: 1229      2028      5
: 1230      L 2029      5      | IF %BLISS (BLISS32)
: 1231      2030      5      | %THEN
: 1232      2031      5      |                                     ! Signal errors for BLISS32
: 1233      2032      5      |                                     SIGNAL (INDEX$_NOBEGIN);
: 1234      2033      5
: 1235      U 2034      5      | %ELSE
: 1236      2035      5      |                                     ! Use $XPO_PUT_MSG otherwise
: 1237      2036      5      |                                     $XPO_PUT_MSG (SEVERITY = WARNING,
: 1238      2037      5      |                                     STRING = '.XPLUS (END) with no .XPLUS (BEGIN) -- inserted as .XPLUS ()');
: 1239      U 2038      5
: 1240      2039      5      | %FI
: 1241      2040      5
: 1242      2041      5      | DMPENT (.ENT_LEN, .ENT_PTR);
: 1243      2042      5      | XPLBLK [XPL$V_END] = FALSE;
: 1244      2043      4      | END;
: 1245      2044      4
: 1246      2045      4      | REF_PTR = INSERT_REF (.XTN);
: 1247      2046      4      | END
: 1248      2047      3      | ELSE
: 1249      2048      3      | REF_PTR = 0;
: 1250      2049      3
: 1251      2050      3      |
: 1252      2051      3      | | Start to set up new entry
: 1253      2052      3      | |
: 1254      2053      3      | TEMP_CELL [XES$A_PREV] = .LAST_CELL;
: 1255      2054      3      | TEMP_CELL [XES$A_NEXT] = .NEXT_CELL;
: 1256      2055      3      | TEMP_CELL [XES$H_SUBC] = .SUB_CNT;
: 1257      2056      3      | TEMP_CELL [XES$V_BARS] = .BAR;
: 1258      2057      3      | TEMP_CELL [XES$A_REF] = .REF_PTR;
: 1259      2058      3      | TEMP_CELL [XES$A_SUBX] = 0;
: 1260      2059      3      | TEMP_CELL [XES$A_BOOK_LIST] = SAVDAT (XMREF, DS_XM_ENTRY, XMSK_LENGTH);
: 1261      2060      3
: 1262      2061      3      | | Remember text string
: 1263      2062      3      | |
: 1264      2063      3      | | IF .STRING NEQ 0
: 1265      2064      3      | | THEN
: 1266      2065      3      | | TEMP_CELL [XES$A_TEXT] = SAVDAT (.STRING, DS_X_STRING, .LNGTH)
: 1267      2066      3      | | ELSE
: 1268      2067      3      | | TEMP_CELL [XES$A_TEXT] = 0;
: 1269      2068      3
: 1270      2069      3
: 1271      2070      3      | |
: 1272      2071      3      | | Save sort string if there is one
: 1273      2072      3      | |
: 1274      2073      3      | | IF .SORT_LEN NEQ 0
: 1275      2074      3      | | THEN
: 1276      2075      3      | | TEMP_CELL [XES$A_SORT_AS] = SAVDAT (.SORT_PTR, DS_X_STRING, .SORT_LEN)
: 1277      2076      3      | | ELSE
: 1278      2077      3      | | TEMP_CELL [XES$A_SORT_AS] = 0;
```

```

: 1279      2078      3
: 1280      2079      3
: 1281      2080      3      | Now put away the entry proper
: 1282      2081      3
: 1283      2082      3      TEMP1 = SAVDAT (TEMP_CELL, DS_X_ENTRY, XESK_LENGTH);
: 1284      2083      3
: 1285      2084      3
: 1286      2085      3      | Link to previous entry
: 1287      2086      3
: 1288      2087      3      IF .LAST_CELL NEQ 0
: 1289      2088      3      THEN
: 1290      2089      4          BEGIN
: 1291      2090      4              .EMP = .LAST_CELL;
: 1292      2091      4
: 1293      2092      4              IF .SUB_CNT NEQ .TEMP [XESK_SUBC]
: 1294      2093      4              THEN
: 1295      2094      4                  TEMP [XESA_SUBX] = .TEMP1
: 1296      2095      4              ELSE
: 1297      2096      4                  TEMP [XESA_NEXT] = .TEMP1;
: 1298      2097      4
: 1299      2098      4              END
: 1300      2099      3      ELSE
: 1301      2100      3          | Head of List
: 1302      2101      3          |
: 1303      2102      3          | .CELL [CSA_HEAD] = .TEMP1;
: 1304      2103      3
: 1305      2104      3
: 1306      2105      3      | Link to the following cell
: 1307      2106      3
: 1308      2107      3
: 1309      2108      3      IF .NEXT_CELL NEQ 0
: 1310      2109      3      THEN
: 1311      2110      4          BEGIN
: 1312      2111      4              TEMP = .NEXT_CELL;
: 1313      2112      4              TEMP [XESA_PREV] = .TEMP1;
: 1314      2113      3          END;
: 1315      2114      3
: 1316      2115      3
: 1317      2116      3      | Remember where we left off
: 1318      2117      3
: 1319      2118      3      CELL [CSA_CURR] = .TEMP1;
: 1320      2119      3      END
: 1321      2120      3
: 1322      2121      1      END;                                !End of INSERT_INX
```

OFFC 00000 INSERT_INX:

5B 00000000G	8F	D0 00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11
5A 00000000G	EF	9E 00009	MOVL	#DSRINDEX\$ NOBEGIN, R11
59 00000000G	00	9E 00010	MOVAB	DMPENT, R10
58 00000000G	EF	9E 00017	MOVAB	LIB\$SIGNAL, R9
57 00000000G	EF	9E 0001E	MOVAB	SAVDAT, R8
56 00000000G	EF	9E 00025	MOVAB	CELL, R7
			MOVAB	XPLBLK, R6

: 1738

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
INSERT_INX -- Insert index item into list

N 15
16-Sep-1984 01:04:24 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:07:15 [RUNOFF.SRC]NDXOUT.BLI;1

Page 57
(7)

		SE		28	C2	0002C	SUBL2	#40, SP			
			20	AE	D4	0002F	CLRL	XMREF		1786	
24		AE	00000000G	EF	D0	00032	MOVL	BOOKID, XMREF+4		1787	
		50		67	D0	0003A	MOVL	CELL, R0		1805	
		54		10	AC	D0	0003D	MOVL	XTN, R4	1833	
		03		0C	A7	E8	00041	BLBS	CELL+12, 1\$	1792	
				00C6	31	00045	BRW	15\$			
		53		50	D0	00048	1\$:	MOVL	R0, XE_TEMP	1805	
		52		1C	A3	D0	0004B	MOVL	28(XE_TEMP), XM_TEMP	1810	
	00000000G	EF		04	A2	D1	0004F	2\$:	CPL	4(XM_TEMP), BOOKID	1818
					18	13	00057	BEQL	4\$		
					62	D5	00059	TSTL	(XM_TEMP)	1820	
					0F	12	0005B	BNEQ	3\$		
					02	DD	0005D	PUSHL	#2	1826	
					02	DD	0005F	PUSHL	#2		
			28	AE	9F	00061	PUSHAB	XMREF			
		68		03	FB	00064	CALLS	#3, SAVDAT			
		62		50	D0	00067	MOVL	R0, (XM_TEMP)			
				05	11	0006A	BRB	4\$		1822	
		52		62	D0	0006C	3\$:	MOVL	(XM_TEMP), XM_TEMP	1830	
				DE	11	0006F	BRB	2\$		1810	
				54	D5	00071	4\$:	TSTL	R4	1833	
				01	12	00073	BNEQ	5\$			
					04	00075	RET				
55			01	00	EF	00076	5\$:	EXTZV	#0, #1, XPLBLK, R5	1899	
				0C	A3	D5	0007B	TSTL	12(XE_TEMP)	1842	
					6A	13	0007E	BEQL	13\$		
					50	7C	00080	CLRL	RANGE_BOOK	1853	
		52		0C	A3	D0	00082	MOVL	12(XE_TEMP), XX_TEMP	1854	
	07	0A			02	E1	00086	6\$:	BBC	#2, 10(XX_TEMP), 7\$	1863
		51			01	D0	0008B	MOVL	#1, RANGE_ACTIVE	1869	
		50		0C	A2	D0	0008E	MOVL	12(XX_TEMP), RANGE_BOOK	1870	
	06	0A			03	E0	00092	7\$:	BBS	#3, 10(XX_TEMP), 8\$	1873
		50		0C	A2	D1	00097	CPL	12(XX_TEMP), RANGE_BOOK	1874	
					02	13	0009B	BEQL	9\$		
					51	D4	0009D	8\$:	CLRL	RANGE_ACTIVE	1880
					62	D5	0009F	9\$:	TSTL	(XX_TEMP)	1885
					05	13	000A1	BEQL	10\$		
		52			62	D0	000A3	MOVL	(XX_TEMP), XX_TEMP	1887	
					DE	11	000A6	BRB	6\$		
		1C			51	E9	000A8	10\$:	BLBC	RANGE_ACTIVE, 11\$	1893
		2F			55	E9	000AB	BLBC	R5, 12\$	1899	
	28	66			03	E1	000AE	BBC	#3, XPLBLK, 12\$		
			00000000G		8F	DD	000B2	PUSHL	#DSRINDEX\$, DUPBEGIN	1908	
		69			01	FB	000B8	CALLS	#1, LIB\$SIGNAL		
		7E		18	AC	7D	000BB	MOVQ	ENT_LEN, -(SP)	1917	
		6A			02	FB	000BF	CALLS	#2, DMPENT		
		66			08	8A	000C2	BICB2	#8, XPLBLK	1918	
					16	11	000C5	BRB	12\$	1893	
		13			55	E9	000C7	11\$:	BLBC	R5, 12\$	1925
		66			04	E1	000CA	BBC	#4, XPLBLK, 12\$		
	0F				5B	DD	000CE	PUSHL	R11	1934	
		69			01	FB	000D0	CALLS	#1, LIB\$SIGNAL		
		7E		18	AC	7D	000D3	MOVQ	ENT_LEN, -(SP)	1943	
		6A			02	FB	000D7	CALLS	#2, DMPENT		
		66			10	8A	000DA	BICB2	#16, XPLBLK	1944	
					54	DD	000DD	12\$:	PUSHL	R4	1950

B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

			00000000V	EF	01	FB	000DF	CALLS	#1, INSERT_REF	
				62	50	D0	000E6	MOVL	R0, (XX_TEMP)	
				13		04	000E9	RET		1842
			OF	66	55	E9	000EA	13\$: BLBC	R5, 14\$	1957
				69	04	E1	000ED	BBC	#4, XPLBLK, 14\$	
				7E	5B	DD	000F1	PUSHL	R11	1966
				6A	01	FB	000F3	CALLS	#1, LIB\$SIGNAL	
				66	18	AC	7D 000F6	MOVQ	ENT_LEN, -(SP)	1975
					02	FB	000FA	CALLS	#2, DMPENT	
					10	8A	000FD	BICB2	#16, XPLBLK	1976
			00000000V	EF	54	DD	00100	14\$: PUSHL	R4	1982
			OC	A3	01	FB	00102	CALLS	#1, INSERT_REF	
					50	D0	00109	MOVL	R0, 12(XE_TEMP)	
				53		04	0010D	RET		1833
OC	AC	18	A3	10	50	D0	0010E	15\$: MOVL	R0, TEMP	2003
					00	EC	00111	CMPV	#0, #16, 24(TEMP), SUB_CNT	2005
				52	08	12	00118	BNEQ	16\$	
				55	50	D0	0011A	MOVL	R0, NEXT_CELL	2008
					63	D0	0011D	MOVL	(TEMP), LAST_CELL	200
					05	11	00120	BRB	17\$	
				55	52	D4	00122	16\$: CLRL	NEXT_CELL	2013
					50	D0	00124	MOVL	R0, LAST_CELL	2014
					54	D5	00127	17\$: TSTL	R4	2017
				13	21	13	00129	BEQL	19\$	
			OF	66	66	E9	0012B	BLBC	XPLBLK, 18\$	2023
					04	E1	0012E	BBC	#4, XPLBLK, 18\$	
				69	5B	DD	00132	PUSHL	R11	2032
				7E	01	FB	00134	CALLS	#1, LIB\$SIGNAL	
				6A	18	AC	7D 00137	MOVQ	ENT_LEN, -(SP)	2041
				66	02	FB	0013B	CALLS	#2, DMPENT	
					10	8A	0013E	BICB2	#16, XPLBLK	2042
			00000000V	EF	54	DD	00141	18\$: PUSHL	R4	2045
					01	FB	00143	CALLS	#1, INSERT_REF	
					02	11	0014A	BRB	20\$	2017
				6E	50	D4	0014C	19\$: CLRL	REF_PTR	2048
				04	55	D0	0014E	20\$: MOVL	LAST_CELL, TEMP_CELL	2053
				18	52	D0	00151	MOVL	NEXT_CELL, TEMP_CELL+4	2054
1A	AE	01		00	OC	AC	B0 00155	MOVW	SUB_CNT, TEMP_CELL+24	2055
				OC	14	AC	F0 0015A	INSV	BAR, #0, #1, TEMP_CELL+26	2056
					50	D0	00161	MOVL	REF_PTR, TEMP_CELL+12	2057
					08	AE	D4 00165	CLRL	TEMP_CELL+8	2058
					02	DD	00168	PUSHL	#2	2059
					02	DD	0016A	PUSHL	#2	
				68	28	AE	9F 0016C	PUSHAB	XMREF	
				1C	03	FB	0016F	CALLS	#3, SAVDAT	
					50	D0	00172	MOVL	R0, TEMP_CELL+28	
					04	AC	D5 00176	TSTL	STRING	2064
					11	13	00179	BEQL	21\$	
					08	AC	DD 0017B	PUSHL	LNTH	2066
					7E	D4	0017E	CLRL	-(SP)	
					04	AC	DD 00180	PUSHL	STRING	
				68	03	FB	00183	CALLS	#3, SAVDAT	
				10	50	D0	00186	MOVL	R0, TEMP_CELL+16	
					03	11	0018A	BRB	22\$	
					10	AE	D4 0018C	21\$: CLRL	TEMP_CELL+16	2068
				50	04C4	C7	D0 0018F	22\$: MOVL	SORT_LEN, R0	2073
					11	13	00194	BEQL	23\$	

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
INSERT_INX -- Insert index item into list

C 16
16-Sep-1984 01:04:24 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:07:15 [RUNOFF.SRC]NDXOUT.BLI;1

Page 59
(7)

				50	DD	00196	PUSHL	R0	:	2075
				7E	D4	00198	CLRL	-(SP)	:	
		04C0		C7	DD	0019A	PUSHL	SORT_PTR	:	
				03	FB	0019E	CALLS	#3, SAVDAT	:	
	14	68		50	D0	001A1	MOVL	R0, TEMP_CELL+20	:	
		AE		03	11	001A5	BRB	24\$	:	
			14	AE	D4	001A7	CLRL	TEMP_CELL+20	:	2077
				08	D0	001AA	PUSHL	#8	:	2082
				08	DD	001AC	PUSHL	#8	:	
			08	AE	9F	001AE	PUSHAS	TEMP_CELL	:	
		68		03	FB	001B1	CALLS	#3, SAVDAT	:	
				55	D5	001B4	TSTL	LAST_CELL	:	2087
				18	13	001B6	BEQL	26\$	:	
		53		55	D0	001B8	MOVL	LAST_CELL, TEMP	:	2090
0C	AC		18	10	EC	001BB	CMPV	#0, #16, 24(TEMP), SUB_CNT	:	2092
				06	13	001C2	BEQL	25\$	:	
	08	A3		50	D0	001C4	MOVL	TEMP1, 8(TEMP)	:	2094
				0A	11	001C8	BRB	27\$	:	
	04	A3		50	D0	001CA	MOVL	TEMP1, '(TEMP)	:	2096
				04	11	001CE	BRB	27\$	:	2087
	08	B7		50	D0	001D0	MOVL	TEMP1, @CELL+8	:	2103
				52	D5	001D4	TSTL	NEXT_CELL	:	2108
				06	13	001D6	BEQL	28\$	:	
		53		52	D0	001D8	MOVL	NEXT_CELL, TEMP	:	2111
		63		50	D0	001DB	MOVL	TEMP1, (TEMP)	:	2112
		67		50	D0	001DE	MOVL	TEMP1, CELL	:	2118
				04	001E1		RET		:	2121

; Routine Size: 482 bytes, Routine Base: \$CODE\$ + 0475

```
1324 2122 1 %SBTTL 'INSERT_REF -- Insert page reference into list'
1325 2123 1 ROUTINE INSERT_REF (XTN) =
1326 2124 1 ++
1327 2125 1
1328 2126 1 FUNCTIONAL DESCRIPTION:
1329 2127 1
1330 2128 1 This routine inserts a page reference into the indexing pool
1331 2129 1
1332 2130 1 FORMAL PARAMETERS:
1333 2131 1
1334 2132 1 XTN - Transaction number
1335 2133 1
1336 2134 1 IMPLICIT INPUTS:
1337 2135 1
1338 2136 1 XPLBLK - Extended indexing attributes block
1339 2137 1 BOOKID - Master index book ident string address
1340 2138 1
1341 2139 1 IMPLICIT OUTPUTS:
1342 2140 1
1343 2141 1 None
1344 2142 1
1345 2143 1 ROUTINE VALUE:
1346 2144 1 COMPLETION CODES:
1347 2145 1
1348 2146 1 Returns the address of the saved page reference
1349 2147 1
1350 2148 1 SIDE EFFECTS:
1351 2149 1
1352 2150 1 None
1353 2151 1 --
1354 2152 2 BEGIN
1355 2153 2
1356 2154 2 LOCAL
1357 2155 2 REF_CELL : $XX_BLOCK;
1358 2156 2
1359 2157 2 REF_CELL [XX$A_LINK] = 0;
1360 2158 2 REF_CELL [XX$A_BOOK] = .BOOKID;
1361 2159 2 REF_CELL [XX$H_PAGE] = .XTN;
1362 2160 2 REF_CELL [XX$V_FLAGS] = 0;
1363 2161 2 REF_CELL [XX$A_APPEND] = 0;
1364 2162 2
1365 2163 2 IF .XPLBLK [XPL$V_VALID]
1366 2164 2 THEN
1367 2165 2 BEGIN
1368 2166 2
1369 2167 2 Have .XPLUS information
1370 2168 2
1371 2169 2 LOCAL
1372 2170 2 APPEND : REF $STR_DESCRIPTOR ();
1373 2171 2
1374 2172 2 REF_CELL [XX$V_BOLD] = .XPLBLK [XPL$V_BOLD];
1375 2173 2 REF_CELL [XX$V_UNDERLINE] = .XPLBLK [XPL$V_UNDERLINE];
1376 2174 2 REF_CELL [XX$V_BEGIN] = .XPLBLK [XPL$V_BEGIN];
1377 2175 2 REF_CELL [XX$V_END] = .XPLBLK [XPL$V_END];
1378 2176 2
1379 2177 2 APPEND = XPLBLK [XPL$T_APPEND];
1380 2178 2 IF .APPEND [STR$H_LENGTH] NEQ 0
```


NDXOUT
V04-000

NDXOUT -- Sort and store index entries
INSERT_REF -- Insert page reference into list

E 16
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI:1

Page 61
(8)

```
: 1381      2179      3      THEN
: 1382      2180      3      |
: 1383      2181      3      | Have an append string
: 1384      2182      3      |
: 1385      2183      3      | REF_CELL [XX$A_APPEND] = SAVDAT (.APPEND [STR$A_POINTER], DS_X_STRING, .APPEND [STR$H_LENGTH]);
: 1386      2184      3      |
: 1387      2185      2      END;
: 1388      2186      2
: 1389      2187      2      RETURN SAVDAT (REF_CELL, DS_XX_ENTRY, XX$K_LENGTH);
: 1390      2188      1      END;
```

```
                                000C 00000 INSERT_REF:
                                .WORD      Save R2,R3
                                MOVAB      SAVDAT, R3
                                MOVAB      XPLBLK, R2
                                SUBL2      #12, SP
                                CLRL       REF_CELL
                                MOVL        BOORID, REF_CELL+12
                                MOVZWL     XTN, REF_CELL+8
                                CLRL       REF_CELL+4
                                BLBC       XPLBLK, 1$
                                EXTZV      #1, #1, XPLBLK, R0
                                INSV       R0, #0, #1, REF_CELL+10
                                EXTZV      #2, #1, XPLBLK, R0
                                INSV       R0, #1, #1, REF_CELL+10
                                EXTZV      #3, #1, XPLBLK, R0
                                INSV       R0, #2, #1, REF_CELL+10
                                EXTZV      #4, #1, XPLBLK, R0
                                INSV       R0, #3, #1, REF_CELL+10
                                MOVAB      XPLBLK+12, APPEND
                                TSTW       (APPEND)
                                BEQL       1$
                                MOVZWL     (APPEND), -(SP)
                                CLRL       -(SP)
                                PUSHL      4(APPEND)
                                CALLS      #3, SAVDAT
                                MOVL        R0, REF_CELL+4
                                1$:
                                PUSHL      #4
                                PUSHAB     REF_CELL
                                CALLS      #3, SAVDAT
                                RET

                                53 00000000G EF 9E 00002
                                52 00000000G EF 9E 00009
                                5E      0C C2 00010
                                7E D4 00013
                                0C AE 00000000G EF D0 00015
                                08 AE      04 AC 3C 0001D
                                04 AE D4 00022
                                43      62 E9 00025
                                01      01 EF 00028
                                00      50 F0 0002D
                                01      02 EF 00033
                                01      50 F0 00038
                                01      03 EF 0003E
                                02      50 F0 00043
                                01      04 EF 00049
                                03      50 F0 0004E
                                50      0C A2 9E 00054
                                60      0F B5 00058
                                7E      60 3C 0005C
                                04      7E D4 0005F
                                63      04 A0 DD 00061
                                04 AE      03 FB 00064
                                04      50 D0 00067
                                04      04 DD 0006B 1$:
                                04      04 DD 0006D
                                08      AE 9F 0006F
                                63      03 FB 00072
                                04      04 00075
```

; Routine Size: 118 bytes. Routine Base: \$CODE\$ + 0657

NDXOUT
V04-000

NDXOUT -- Sort and store index entries

ENTRY_CMP -- Compare new entry with current ent

F 16

16-Sep-1984 01:04:24

14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742

[RUNOFF.SRC]NDXOUT.BLI;1

Page 62
(9)

```

1392 2189 1 %SBTTL 'ENTRY_CMP -- Compare new entry with current entry'
1393 2190 1 ROUTINE ENTRY_CMP (NEW_PTR, NEW_LEN, LAST, XTN, LEVEL) =
1394 2191 1 ++
1395 2192 1
1396 2193 1 FUNCTIONAL DESCRIPTION:
1397 2194 1
1398 195 1 This routine compares the new entry with the current entry.
1399 2196 1
1400 2197 1 For subindex entries (i.e. LEVEL NEQ 0) performs the following checks:
1401 2198 1
1402 2199 1 If LAST is TRUE, checks to see if new entry is either
1403 2200 1 a .ENTRY or .YPLUS.
1404 2201 1
1405 2202 1 If the new entry is a .Y or .YP, it checks to see if the
1406 2203 1 current entry is either a .X (.XP) or if the current entry
1407 2204 1 has subentries. If so, a value of TRUE is returned indicating
1408 2205 1 that the new entry should be inserted before the current entry.
1409 2206 1
1410 2207 1 If LAST is FALSE or the new entry is not a .Y (.YP) and the
1411 2208 1 current entry is a .Y (.YP), a value of FALSE is returned
1412 2209 1 indicating that the new entry should be inserted after the
1413 2210 1 current entry.
1414 2211 1
1415 2212 1 Otherwise, calls STRG_CMP to see if the new entry should be
1416 2213 1 inserted here. If so, calls STRG_CMP again to see if the
1417 2214 1 new entry is identical to the current entry.
1418 2215 1
1419 2216 1 FORMAL PARAMETERS:
1420 2217 1
1421 2218 1 NEW_PTR - Pointer to new entry text
1422 2219 1 NEW_LEN - Length of new entry
1423 2220 1 LAST - TRUE if XTN should be compared to transaction number
1424 2221 1 associated with CELL.
1425 2222 1 XTN - Transaction number of new entry if LAST is TRUE
1426 2223 1 LEVEL - Subindex level (0 to n)
1427 2224 1
1428 2225 1 IMPLICIT INPUTS:
1429 2226 1
1430 2227 1 CELL - contains pointers to the list item for comparison.
1431 2228 1 SORT_LEN- Length of sort string if any
1432 2229 1 SORT_PTR- pointer to sort string if any
1433 2230 1
1434 2231 1 IMPLICIT OUTPUTS:
1435 2232 1
1436 2233 1 None
1437 2234 1
1438 2235 1 ROUTINE VALUE:
1439 2236 1 COMPLETION CODES:
1440 2237 1
1441 2238 1 TRUE if new entry should be inserted before the current entry,
1442 2239 1 FALSE otherwise.
1443 2240 1
1444 2241 1 SIDE EFFECTS:
1445 2242 1
1446 2243 1 None
1447 2244 1 --
1448 2245 2 BEGIN
```



```

: 1449      2246 2    LOCAL
: 1450      2247 2      CEPTR : REF $XE_BLOCK,
: 1451      2248 2      N_PTR,
: 1452      2249 2      N_LEN,
: 1453      2250 2      C_VEC : REF VECTOR,
: 1454      2251 2      C_PTR,
: 1455      2252 2      C_LEN;
: 1456      2253 2
: 1457      2254 2    CEPTR = .CELL [C$A_CURR];
: 1458      2255 2
: 1459      2256 2    IF .LEVEL NEQ 0
: 1460      2257 2    THEN
: 1461      2258 2      BEGIN
: 1462      2259 2        : Subindex entry.
: 1463      2260 2        : Check to see if we should float a .Y
: 1464      2261 2
: 1465      2262 2      IF .LAST AND (.XTN EQL 0)          ! If at bottom of new entry
: 1466      2263 2      THEN                                ! and new entry is a .Y
: 1467      2264 2        BEGIN
: 1468      2265 2          IF (.CEPTR [XE$A_REF] NEQ 0) OR (.CEPTR [XE$A_SUBX] NEQ 0)
: 1469      2266 2          THEN
: 1470      2267 2            : Current entry is a .X or .XP or has subentries.
: 1471      2268 2            :
: 1472      2269 2            RETURN TRUE;                  ! New entry is before current entry
: 1473      2270 2          ELSE
: 1474      2271 2            :
: 1475      2272 2            END
: 1476      2273 2          ELSE
: 1477      2274 2            BEGIN
: 1478      2275 2              : Not at bottom of entry or not .Y
: 1479      2276 2              :
: 1480      2277 2              IF (.CEPTR [XE$A_REF] EQL 0) AND (.CEPTR [XE$A_SUBX] EQL 0)
: 1481      2278 2              THEN
: 1482      2279 2                : Current entry is a .Y or .YP
: 1483      2280 2                :
: 1484      2281 2                RETURN FALSE;              ! New entry is after current entry
: 1485      2282 2              ELSE
: 1486      2283 2                :
: 1487      2284 2                END;
: 1488      2285 2              END;
: 1489      2286 2            END;
: 1490      2287 2          END;
: 1491      2288 2        END;
: 1492      2289 2      IF .SORT_LEN NEQ 0
: 1493      2290 2      THEN
: 1494      2291 2        BEGIN
: 1495      2292 2          : New entry has a sort string. Use it.
: 1496      2293 2          :
: 1497      2294 2          N_PTR = .SORT_PTR;
: 1498      2295 2          N_LEN = .SORT_LEN;
: 1499      2296 2          END
: 1500      2297 2        ELSE
: 1501      2298 2          BEGIN
: 1502      2299 2            :
: 1503      2300 2            : no sort string is available. use text.
: 1504      2301 2            :
: 1505      2302 2          END
```

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
ENTRY_CMP -- Compare new entry with current ent

H 16
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 54
(9)

```
: 1506      2303      3      N_PTR = .NEW_PTR;  
: 1507      2304      3      N_LEN = .NEW_LEN;  
: 1508      2305      3      END;  
: 1509      2306      2  
: 1510      2307      2      IF .CEPTR [XESA_SORT_AS] NEQ 0  
: 1511      2308      2      THEN  
: 1512      2309      2          :  
: 1513      2310      2          : Current entry has a sort string. Use it.  
: 1514      2311      2          :  
: 1515      2312      2          C_VEC = .CEPTR [XESA_SORT_AS]  
: 1516      2313      2      ELSE  
: 1517      2314      2          :  
: 1518      2315      2          : Current entry has no sort string. Use text of entry.  
: 1519      2316      2          :  
: 1520      2317      2          C_VEC = .CEPTR [XESA_TEXT];  
: 1521      2318      2          :  
: 1522      2319      2          :  
: 1523      2320      2          : Get number of characters in internal sort string  
: 1524      2321      2          : Length is stored as the first fullword of the string  
: 1525      2322      2          :  
: 1526      2323      2          C_LEN = .C_VEC [0];  
: 1527      2324      2          C_PTR = CH$PTR (C_VEC [1]);  
: 1528      2325      2          :  
: 1529      2326      2          :  
: 1530      2327      2          : Check to see if this is the proper insertion point  
: 1531      2328      2          :  
: 1532      2329      2          IF STRG_CMP (.N_LEN, .N_PTR, .C_LEN, .C_PTR)  
: 1533      2330      2          THEN  
: 1534      2331      3              BEGIN  
: 1535      2332      3                  :  
: 1536      2333      3                  : This is almost the spot.  
: 1537      2334      3                  : Check for identical sort strings.  
: 1538      2335      3                  :  
: 1539      2336      3                  IF .CELL [CSV_IDNS]  
: 1540      2337      3                  THEN  
: 1541      2338      4                      BEGIN  
: 1542      2339      4                          :  
: 1543      2340      4                          : Sort strings were identical.  
: 1544      2341      4                          : Compare text strings to determine positioning.  
: 1545      2342      4                          :  
: 1546      2343      4                          CELL [CSV_IDNS] = FALSE;  
: 1547      2344      4                          :  
: 1548      2345      4                          C_VEC = .CEPTR [XESA_TEXT];  
: 1549      2346      4                          C_LEN = .C_VEC [0];  
: 1550      2347      4                          C_PTR = CH$PTR (C_VEC [1]);  
: 1551      2348      4                          :  
: 1552      2349      4                          RETURN STRG_CMP (.NEW_LEN, .NEW_PTR, .C_LEN, .C_PTR);  
: 1553      2350      4                      END  
: 1554      2351      3                  ELSE  
: 1555      2352      3                      :  
: 1556      2353      3                      : Sort strings different.  
: 1557      2354      3                      : This is the correct position for insertion.  
: 1558      2355      3                      :  
: 1559      2356      3                      RETURN TRUE;  
: 1560      2357      3                  END  
: 1561      2358      2      ELSE  
: 1562      2359      2          RETURN FALSE;
```

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
ENTRY_CMP -- Compare new entry with current ent

I 16
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 65
(9)

: 1563
: 1564

2360 2
2361 1 END;

```
00FC 00000 ENTRY_CMP:
57 00000000V EF 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7 : 2190
56 00000000' EF 9E 00009 MOVAB STRG_CMP, R7 :
54 F4 A6 D0 00010 MOVAB CELL+12, R6 :
14 AC D5 00014 MOVL CELL, CEPTR : 2254
1F 13 00017 TSTL LEVEL : 2256
11 0C AC E9 00019 BEQL 2$ :
10 AC D5 0001D BLBC LAST, 1$ : 2263
0C 12 00020 TSTL XTN :
0C A4 D5 00022 BNEQ 1$ :
66 12 00025 TSTL 12(CEPTR) : 2266
08 A4 D5 00027 BNEQ 7$ :
0C 13 0002A TSTL 8(CEPTR) :
5F 11 0002C BEQL 2$ :
0C A4 D5 0002E 1$: BRB 7$ : 2271
05 12 00031 TSTL 12(CEPTR) : 2279
08 A4 D5 00033 BNEQ 2$ :
59 13 00036 TSTL 8(CEPTR) :
50 04B8 C6 D0 00038 2$: BEQL 8$ :
07 13 0003D MOVL SORT_LEN, R0 : 2289
51 04B4 C6 D0 0003F BEQL 3$ :
08 11 00044 MOVL SORT_PTR, N_PTR : 2295
51 04 AC D0 00046 3$: BRB 4$ : 2289
50 08 AC D0 0004A MOVL NEW_PTR, N_PTR : 2303
14 A4 D5 0004E 4$: MOVL NEW_LEN, N_LEN : 2304
06 13 00051 TSTL 20(CEPTR) : 2307
52 14 A4 D0 00053 BEQL 5$ :
04 11 00057 MOVL 20(CEPTR), C_VEC : 2312
52 10 A4 D0 00059 5$: BRB 6$ :
55 62 D0 0005D 6$: MOVL 16(CEPTR), C_VEC : 2317
53 04 A2 9E 00060 MOVL (C_VEC), C_LEN : 2323
53 DD 00064 MOVAB 4(R2), C_PTR : 2324
23 BB 00066 PUSHL C_PTR : 2329
67 04 FB 00068 PUSHR #M<R0,R1,R5> :
23 50 E9 0006B CALLS #4, STRG_CMP :
1C 66 E9 0006E BLBC R0, 8$ :
66 01 8A 00071 BLBC CELL+12, 7$ : 2336
52 10 A4 D0 00074 BICB2 #1, CELL+12 : 2343
55 62 D0 00078 MOVL 16(CEPTR), C_VEC : 2345
53 04 A2 9E 0007B MOVL (C_VEC), C_LEN : 2346
53 DD 0007F MOVAB 4(R2), C_PTR : 2347
55 DD 00081 PUSHL C_PTR : 2349
04 DD 00083 PUSHL C_LEN :
08 AC DD 00086 PUSHL NEW_PTR :
67 04 FB 00089 CALLS #4, STRG_CMP :
04 0008C RET : 2356
50 01 D0 0008D 7$: MOVL #1, R0 :
04 00090 RET : 2359
50 D4 00091 8$: CLRL R0 : 2361
```

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
ENTRY_CMP -- Compare new entry with current ent

J 16
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 66
(9)

04 00093

FET

;

; Routine Size: 148 bytes, Routine Base: \$CODE\$ + 06CD

```
: 1566      2362 1 %SBTTL 'STRG_CMP -- Compare two strings'
: 1567      2363 1
: 1568      2364 1 ROUTINE STRG_CMP (S1_LEN, S1_PTR, S2_LEN, S2_PTR) =
: 1569      2365 1
: 1570      2366 1 ++
: 1571      2367 1 FUNCTIONAL DESCRIPTION:
: 1572      2368 1
: 1573      2369 1     This routine is called to compare two strings.
: 1574      2370 1     It returns TRUE if string 1 should be before string 2.
: 1575      2371 1
: 1576      2372 1     It sets CELL [CSV_IDNS] if the strings are identical.
: 1577      2373 1
: 1578      2374 1 FORMAL PARAMETERS:
: 1579      2375 1
: 1580      2376 1     S1_LEN - Length of string 1
: 1581      2377 1     S1_PTR - Pointer to string 1
: 1582      2378 1     S2_LEN - Length of string 2
: 1583      2379 1     S2_PTR - Pointer to string 2
: 1584      2380 1
: 1585      2381 1 IMPLICIT INPUTS:
: 1586      2382 1
: 1587      2383 1     NONE
: 1588      2384 1
: 1589      2385 1 IMPLICIT OUTPUTS:
: 1590      2386 1
: 1591      2387 1     CELL [CSV_IDNS] - set to true if strings are identical
: 1592      2388 1
: 1593      2389 1 ROUTINE VALUE:
: 1594      2390 1 COMPLETION CODES:
: 1595      2391 1
: 1596      2392 1     TRUE   - String 1 is before string 2
: 1597      2393 1     FALSE  - Otherwise
: 1598      2394 1
: 1599      2395 1 SIDE EFFECTS:
: 1600      2396 1
: 1601      2397 1     NONE
: 1602      2398 1
: 1603      2399 1 --
: 1604      2400 1
: 1605      2401 2 BEGIN
: 1606      2402 2
: 1607      2403 2 LOCAL
: 1608      2404 2 CASECMP,
: 1609      2405 2 CHARCMP,
: 1610      2406 2 COLUMN,
: 1611      2407 2 EMPHCOMP,
: 1612      2408 2 ICASE,
: 1613      2409 2 IEMPH,
: 1614      2410 2 OLDCase,
: 1615      2411 2 OLDEMPH,
: 1616      2412 2 PTR_1,
: 1617      2413 2 REM_1,
: 1618      2414 2 PTR_2,
: 1619      2415 2 REM_2;
: 1620      2416 2
: 1621      2417 2 PTR_1 = .S1_PTR;
: 1622      2418 2 REM_1 = .S1_LEN;
```

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
STRG_CMP -- Compare two strings

L 16
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 68
(10)

```
: 1623      2419  2      PTR_2 = .S2_PTR;  
: 1624      2420  2      REM_2 = .S2_LEN;  
: 1625      2421  2  
: 1626      2422  2      ICASE = 0;          ! No differences in case yet  
: 1627      2423  2      IEMPH = 0;         ! No differences in emphasis yet  
: 1628      2424  2      OLDCase = 0;        ! No differences in case yet  
: 1629      2425  2      OLDEMPH = 0;       ! No differences in emphasis yet  
: 1630      2426  2      COLUMN = 0;        ! No print positions scanned yet.  
: 1631      2427  2  
: 1632      2428  2  
: 1633      2429  2      ! Loop until done with both strings  
: 1634      2430  2  
: 1635      2431  2      REPEAT  
: 1636      2432  3          BEGIN  
: 1637      2433  3              !  
: 1638      2434  3              ! Update count of print columns, so positions of case and emphasis  
: 1639      2435  3              ! differences can be remembered.  
: 1640      2436  3              !  
: 1641      2437  3              COLUMN = .COLUMN + 1;  
: 1642      2438  3              !  
: 1643      2439  3              ! Make sure neither string has run out. If one has, this is  
: 1644      2440  3              ! the place for insertion.  
: 1645      2441  3              !  
: 1646      2442  3              IF (.REM_2 LEQ 0) OR (.REM_1 LEQ 0)  
: 1647      2443  4              THEN  
: 1648      2444  3                  BEGIN  
: 1649      2445  4                      !  
: 1650      2446  4                      ! Check for exact string before leaving  
: 1651      2447  4                      !  
: 1652      2448  4                      IF (.REM_2 LEQ 0) AND (.REM_1 LEQ 0)  
: 1653      2449  5                      THEN  
: 1654      2450  4                          !  
: 1655      2451  4                          ! Both strings have run out. They're identical if  
: 1656      2452  4                          ! there are no case or emphasis differences.  
: 1657      2453  4                          !  
: 1658      2454  4                          CELL [CSV_IDNS] = ((.ICASE EQL 0) AND (.IEMPH EQL 0))  
: 1659      2455  5                      ELSE  
: 1660      2456  4                          !  
: 1661      2457  4                          ! Only one string has run out. The longer of the two strings  
: 1662      2458  4                          ! is "greater" than the shorter string, or conversely, the  
: 1663      2459  4                          ! one that's run out is the "lesser" of the two. Return TRUE  
: 1664      2460  4                          ! if the input string is the "lesser" of the two.  
: 1665      2461  4                          !  
: 1666      2462  4                          RETURN (.REM_1 LEQ 0);  
: 1667      2463  4                      !  
: 1668      2464  4                      IF .OLDEMPH NEQ 0 THEN RETURN (.OLDEMPH EQL 1);  
: 1669      2465  4                      IF .OLDCase NEQ 0 THEN RETURN (.OLDCase EQL -1);  
: 1670      2466  4                      !  
: 1671      2467  4                      RETURN TRUE;  
: 1672      2468  4                      !  
: 1673      2469  4                      END;  
: 1674      2470  3  
: 1675      2471  3      CHRCMP (PTR_1, PTR_2, CASECMP, CHARCMP, EMPHCMP, REM_1, REM_2);  
: 1676      2472  3  
: 1677      2473  3      IF .CHARCMP NEQ 0  
: 1678      2474  3      THEN  
: 1679      2475  3
```


NDXOUT
V04-000

NDXOUT -- Sort and store index entries
STRG_CMP -- Compare two strings

M 16
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 69
(10)

```

: 1680      2476  4      RETURN (.CHARCMP EQL -1)
: 1681      2477  3      ELSE
: 1682      2478  4      BEGIN
: 1683      2479  4      |
: 1684      2480  4      | Remember differences in the string so they can be
: 1685      2481  4      | applied if the string runs out.
: 1686      2482  4      |
: 1687      2483  4      | If there is a difference of cases, the very first place where
: 1688      2484  4      | case differs is the significant case difference. All other
: 1689      2485  4      | positions are secondary.
: 1690      2486  4      |
: 1691      2487  5      IF (.ICASE EQL 0) AND (.CASECMP NEQ 0)
: 1692      2488  4      THEN
: 1693      2489  5      BEGIN
: 1694      2490  5      |
: 1695      2491  5      | Remember column position where difference occurred
: 1696      2492  5      | Remember what the case difference was.
: 1697      2493  5      |
: 1698      2494  5      | ICASE = .COLUMN;
: 1699      2495  5      | OLDCASE = .CASECMP;
: 1700      2496  4      | END;
: 1701      2497  4      |
: 1702      2498  4      |
: 1703      2499  4      | If there is a difference in emphasis, the very first place where
: 1704      2500  4      | emphasis differs is the significant emphasis difference.
: 1705      2501  4      | All other positions are secondary.
: 1706      2502  4      |
: 1707      2503  5      IF (.IEMPH EQL 0) AND (.EMPHCMP NEQ 0)
: 1708      2504  4      THEN
: 1709      2505  5      BEGIN
: 1710      2506  5      |
: 1711      2507  5      | Remember column position where difference occurred.
: 1712      2508  5      | Remember what the difference in emphasis was.
: 1713      2509  5      |
: 1714      2510  5      | IEMPH = .COLUMN;
: 1715      2511  5      | OLDEMPH = .EMPHCMP;
: 1716      2512  4      | END;
: 1717      2513  3      END;
: 1718      2514  3      END
: 1719      2515  3      END
: 1720      2516  3      END;
: 1721      2517  1      !End of STRG_CMP

```

```

                                00FC 0000 STRG_CMP:
                                .WORD      Save R2,R3,R4,R5,R6,R7
                                SUBL2      #20, SP
10      SE      08      14      C2 00002      MOVL      S1_PTR, PTR_1
                                04      AC 00005      PUSHL     S1_LEN
10      AE      10      AC 0000A      MOVL      S2_PTR, PTR_2
                                0C      AC 0000D      PUSHL     S2_LEN
                                54      7C 00015      CLRQ      IEMPH
                                52      7C 00017      CLRQ      OLDEMPH
                                56      D4 00019      CLRL      COLUMN
                                : 2364
                                : 2417
                                : 2418
                                : 2419
                                : 2420
                                : 2423
                                : 2425
                                : 2426

```

00000000'	EF	57	57	56	D6	0001B	1\$:	INCL	COLUMN	2437
		01	50	50	D4	0001D		CLRL	R0	2443
				6E	D5	0001F		TSTL	REM_2	
				04	14	00021		BGTR	2\$	
				50	D6	00023		INCL	R0	
				05	11	00025		BRB	3\$	
				04	AE	D5	00027	2\$:	TSTL	REM_1
				50	14	0002A		BGTR	10\$	
				50	E9	0002C	3\$:	BLBC	R0, 6\$	2449
				04	AE	D5	0002F		TSTL	REM_1
				22	14	00032		BGTR	6\$	
				51	D4	00034		CLRL	R1	2455
				55	D5	00036		TSTL	ICASE	
				02	12	00038		BNEQ	4\$	
				51	D6	0003A		INCL	R1	
				50	D4	0003C	4\$:	CLRL	R0	
				54	D5	0003E		TSTL	TEMPH	
				02	12	00040		BNEQ	5\$	
				50	D6	00042		INCL	R0	
				51	D2	00044	5\$:	MCOML	R1, R7	
				57	8E	00047		BICB3	R7, R0, R7	
				57	F0	0004B		INSV	R7, #0, #1, CELL+12	
				08	11	00054		BRB	7\$	
				50	D4	00056	6\$:	CLRL	R0	2463
				04	AE	D5	00058		TSTL	REM_1
				4B	15	0005B		BLEQ	12\$	
					04	0005D		RET		
				52	D5	0005E	7\$:	TSTL	OLDEMPH	2465
				07	13	00060		BEQL	8\$	
				50	D4	00062		CLRL	R0	
				52	D1	00064		CMPL	OLDEMPH, #1	
				3D	11	00067		BRB	11\$	
				53	D5	00069	8\$:	TSTL	OLDCASE	2467
				0B	13	0006B		BEQL	9\$	
				50	D4	0006D		CLRL	R0	
				53	D1	0006F		CMPL	OLDCASE, #-1	
				2E	11	00076		BRB	11\$	
				50						
				01	D0	00078	9\$:	MOVL	#1, R0	2469
					04	0007B		RET		
				5E	DD	0007C	10\$:	PUSHL	SP	2472
				08	AE	9F	0007E	PUSHAB	REM_1	
				10	AE	9F	00081	PUSHAB	EMPRCMP	
				18	AE	9F	00084	PUSHAB	CHARCMP	
				20	AE	9F	00087	PUSHAB	CASECMP	
				28	AE	9F	0008A	PUSHAB	PTR_2	
				30	AE	9F	0008D	PUSHAB	PTR_1	
					07	FB	00090	CALLS	#7, CHRCMP	
				0C	AE	D5	00097	TSTL	CHARCMP	2474
					0F	13	0009A	BEQL	13\$	
				50	D4	0009C		CLRL	R0	2476
				0C	AE	D1	0009E	CMPL	CHARCMP, #-1	
				26	12	000A6	11\$:	BNEQ	16\$	
				50	D6	000AB	12\$:	INCL	R0	
					04	000AA		RET		
				55	D5	000AB	13\$:	TSTL	ICASE	2487
				0C	12	000AD		BNEQ	14\$	
				10	AE	D5	000AF	TSTL	CASECMP	

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
STRG_CMP -- Compare two strings

C 1
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 71
(10)

ND
VO

55	07	13	000B2	BEQL	14\$	:		:
53	56	D0	000B4	MOVL	COLUMN, ICASE	:	2494	:
	10	AE	D0 000B7	MOVL	CASECMP, OLDCASE	:	2495	:
	54	D5	000BB 14\$:	TSTL	IEMPH	:	2503	:
	0C	12	000BD	BNEQ	15\$	:		:
	08	AE	D5 000BF	TSTL	EMPHCMP	:		:
54	07	13	000C2	BEQL	15\$	:		:
52	56	D0	000C4	MOVL	COLUMN, IEMPH	:	2510	:
	08	AE	D0 000C7	MOVL	EMPHCMP, OLDEMPH	:	2511	:
	FF4D	31	000CB 15\$:	BRW	1\$	:	2426	:
	04	000CE	16\$:	RET		:	2517	:

; Routine Size: 207 bytes, Routine Base: \$CODE\$ + 0761

```
1723 2518 1 %SBTTL 'CHRCMP -- Compare two characters in internal format'
1724 2519 1
1725 2520 1 ROUTINE CHRCMP (XA, XB, CASECMP, CHARCMP, EMPHCMP, REMAINDER_A, REMAINDER_B) : NOVALUE =
1726 2521 1
1727 2522 1 ++
1728 2523 1 FUNCTIONAL DESCRIPTION:
1729 2524 1
1730 2525 1 CHRCMP compares two characters in RUNOFF internal format (i.e.,
1731 2526 1 as generated by SCANT).
1732 2527 1
1733 2528 1 Basically, the comparison is done lexically, with the change
1734 2529 1 that the characters which are not letters are lexically smaller
1735 2530 1 than any letters.
1736 2531 1
1737 2532 1 It takes overstriking, underlining, and bolding into account
1738 2533 1 when doing the comparison. If two characters are identical
1739 2534 1 except for their emphasis, the comparison is such that the
1740 2535 1 character with the most emphasis is lexically smallest.
1741 2536 1
1742 2537 1 Bolding is considered to emphasize more than underlining, but
1743 2538 1 less than both underlining and bolding together. Underlining
1744 2539 1 emphasizes more than overstriking; but note that the
1745 2540 1 overstriking sequence is NOT taken into account in the
1746 2541 1 comparison. Upper case emphasizes more than lower case. Emphasis
1747 2542 1 is always less significant than "naked character" differences.
1748 2543 1
1749 2544 1 FORMAL PARAMETERS:
1750 2545 1
1751 2546 1 XA and XB are CH$PTRs to the characters to be compared.
1752 2547 1
1753 2548 1 CHARCMP - Returned as if computed by subtracting the internal
1754 2549 1 representations of the characters, except that letters
1755 2550 1 are "greater" than all other characters.
1756 2551 1
1757 2552 1 CASECMP - Returned as if computed by subtracting the
1758 2553 1 "upper/lower caseness" of the characters. Upper case
1759 2554 1 has "value" 0, lower case 1.
1760 2555 1 By definition, characters other than letters are in upper case.
1761 2556 1
1762 2557 1 EMPHCMP - Returned as SIGN(emphasis of A - emphasis of B) where
1763 2558 1 each emphasis type requires one bit. Overstriking has
1764 2559 1 the value 1, underlining has the value 2 and bolding
1765 2560 1 the value 4.
1766 2561 1
1767 2562 1 REMAINDER_A - the number of characters scanned in XA is subtracted from it.
1768 2563 1 REMAINDER_B - the number of characters scanned in XB is subtracted from it.
1769 2564 1
1770 2565 1 IMPLICIT INPUTS:
1771 2566 1
1772 2567 1 The arrangement of the internal representation is implicit
1773 2568 1 in the algorithm. Basically it assumes that the "naked"
1774 2569 1 character comes after the escape (=emphasis) sequences.
1775 2570 1
1776 2571 1 IMPLICIT OUTPUTS:
1777 2572 1
1778 2573 1 NONE
1779 2574 1
```

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
CHRCMP -- Compare two characters in internal fo

E 1
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 73
(11)

ND
VO

```
: 1780 2575 1 | ROUTINE VALUE:
: 1781 2576 1 |
: 1782 2577 1 |     The result is returned as if it could be computed by
: 1783 2578 1 |     SIGN (.A - .B);
: 1784 2579 1 |
: 1785 2580 1 | SIDE EFFECTS:
: 1786 2581 1 |
: 1787 2582 1 |     NONE
: 1788 2583 1 |
: 1789 2584 1 | --
: 1790 2585 1 |
: 1791 2586 2 | BEGIN
: 1792 2587 2 |
: 1793 2588 2 | BIND
: 1794 2589 2 |     PTR_A = .XA;
: 1795 2590 2 |     PTR_B = .XB;
: 1796 2591 2 |
: 1797 2592 2 | LOCAL
: 1798 2593 2 |     CA,
: 1799 2594 2 |     CB,
: 1800 2595 2 |     RA,
: 1801 2596 2 |     RB;
: 1802 2597 2 |
: 1803 2598 2 | RA = 0;
: 1804 2599 2 | RB = 0;
: 1805 2600 2 | .CASECMP = 0;
: 1806 2601 2 | .CHARCMP = 0;
: 1807 2602 2 | .EMPHCMP = 0;
: 1808 2603 2 |
: 1809 2604 2 | REPEAT
: 1810 2605 3 | BEGIN
: 1811 2606 3 |     CA = CH$RCHAR A (PTR_A);
: 1812 2607 3 |     .REMAINDER_A = ..REMAINDER_A - 1;      ! Subtract off scanned character
: 1813 2608 3 |
: 1814 2609 3 | IF .CA EQL RINTES
: 1815 2610 3 | THEN
: 1816 2611 4 |     BEGIN
: 1817 2612 4 |     |
: 1818 2613 4 |     | Interpret escape sequence.
: 1819 2614 4 |     |
: 1820 2615 4 |     CA = CH$RCHAR A (PTR_A);
: 1821 2616 4 |     .REMAINDER_A = ..REMAINDER_A - 2;      ! Subtract off scanned characters.
: 1822 2617 4 |
: 1823 2618 4 | SELECTONE .CA OF
: 1824 2619 4 | SET
: 1825 2620 4 |
: 1826 2621 4 | [%C'B']:
: 1827 2622 4 | |
: 1828 2623 4 | | Emphasis value of bolding.
: 1829 2624 4 | |
: 1830 2625 4 | | RA = .RA OR 4;
: 1831 2626 4 |
: 1832 2627 4 | [%C'U']:
: 1833 2628 4 | |
: 1834 2629 4 | | Emphasis value for underlining.
: 1835 2630 4 | |
: 1836 2631 4 | | RA = .RA OR 2;
```

```
: 1837 2632 4
: 1838 2633 4      [%C'C']:
: 1839 2634 4      |
: 1840 2635 4      | Emphasis value for overstriking.
: 1841 2636 4      |
: 1842 2637 4      | RA = .RA OR 1;
: 1843 2638 4      |
: 1844 2639 4      | [OTHERWISE]:
: 1845 2640 4      |
: 1846 2641 4      | | Non-emphasis value (do nothing)
: 1847 2642 4      | |
: 1848 2643 4      | |
: 1849 2644 4      | |
: 1850 2645 4      | TES;
: 1851 2646 4      |
: 1852 2647 4      | CH$RCHAR_A (PTR_A);
: 1853 2648 4      | END
: 1854 2649 3      | ELSE
: 1855 2650 4      | BEGIN
: 1856 2651 4      |
: 1857 2652 5      | IF UPPER_LETTER (.CA)
: 1858 2653 4      | THEN
: 1859 2654 5      | CA = LOWER_CASE (.CA)
: 1860 2655 4      | ELSE
: 1861 2656 4      | .CASECMP = 1;
: 1862 2657 4      | EXITLOOP
: 1863 2658 4      | END
: 1864 2659 2      | END;
: 1865 2660 2      |
: 1866 2661 2      | | Scan second character.
: 1867 2662 2      | |
: 1868 2663 2      | | REPEAT
: 1869 2664 2      | | BEGIN
: 1870 2665 3      | | CB = CH$RCHAR_A (PTR_B);
: 1871 2666 3      | | .REMAINDER_B = ..REMAINDER_B - 1;      ! Subtract off scanned character
: 1872 2667 3      | |
: 1873 2668 3      | | IF .CB EQL RINTES
: 1874 2669 3      | | THEN
: 1875 2670 3      | | BEGIN
: 1876 2671 4      | | |
: 1877 2672 4      | | | Interpret escape sequence.
: 1878 2673 4      | | |
: 1879 2674 4      | | |
: 1880 2675 4      | | | CB = CH$RCHAR_A (PTR_B);
: 1881 2676 4      | | | .REMAINDER_B = ..REMAINDER_B - 2;      ! Subtract off scanned characters
: 1882 2677 4      | | | SELECTONE .CB OF
: 1883 2678 4      | | | SET
: 1884 2679 4      | | |
: 1885 2680 4      | | | [%C'B']:
: 1886 2681 4      | | | |
: 1887 2682 4      | | | | Emphasis value for bolding.
: 1888 2683 4      | | | |
: 1889 2684 4      | | | | RB = .RB OR 4;
: 1890 2685 4      | | | |
: 1891 2686 4      | | | [%C'U']:
: 1892 2687 4      | | | |
: 1893 2688 4      | | | | Emphasis value for underlining
```

```
1894 2689 4
1895 2690 4      RB = .RB OR 2;
1896 2691 4
1897 2692 4      [%C'O']:
1898 2693 4          |
1899 2694 4          | Emphasis value for overstriking
1900 2695 4
1901 2696 4      RB = .RB OR 1;
1902 2697 4
1903 2698 4      [OTHERWISE]:
1904 2699 4          |
1905 2700 4          | Non-emphasis value (do nothing)
1906 2701 4
1907 2702 4
1908 2703 4
1909 2704 4      TES;
1910 2705 4
1911 2706 4      CH$RCHAR_A (PTR_B);
1912 2707 4      END
1913 2708 3      ELSE
1914 2709 4          BEGIN
1915 2710 4
1916 2711 5          IF UPPER_LETTER (.CB)
1917 2712 4          THEN
1918 2713 5              CB = LOWER_CASE (.CB)
1919 2714 4          ELSE
1920 2715 4              .CASECMP = ..CASECMP - 1;
1921 2716 4          EXITLOOP
1922 2717 4          END
1923 2718 2      END;
1924 2719 2
1925 2720 2      |
1926 2721 2      | At this point, the 'naked' characters are in CA and CB.
1927 2722 2      | Decoded emphasis escape sequences are in RA and RB.
1928 2723 2
1929 2724 2      | 'Subtract' emphasis to get relationship.
1930 2725 2
1931 2726 2      .EMPHCMP = SIGN (.RA - .RB);
1932 2727 2
1933 2728 2      |
1934 2729 2      | Compare the 'naked' part of the characters and
1935 2730 2      | return the relationship.
1936 2731 2
1937 2732 3      IF LOWER_LETTER (.CA)
1938 2733 2      THEN
1939 2734 3          IF LOWER_LETTER (.CB)
1940 2735 2          THEN
1941 2736 2              .CHARCMP = SIGN (.CA - .CB)
1942 2737 2          ELSE
1943 2738 2              .CHARCMP = 1
1944 2739 2
1945 2740 2      ELSE
1946 2741 3          IF LOWER_LETTER (.CB)
1947 2742 2          THEN
1948 2743 2              .CHARCMP = -1
1949 2744 2          ELSE
1950 2745 2              .CHARCMP = SIGN (.CA - .CB);
```

! First character is lower case
! Second character is lower case
! return relationship between characters
! Second character is upper case
! hence second character is 'largest'.
! First character is upper case
! Second character is lower case
! hence first character is 'largest'
! Second character is upper case
! return relationship between characters

ND
VO

!End of CHRCMP

PC	Op	OpC	OpD	OpI	OpR	OpS	OpT	OpV	OpW	OpX	OpY	OpZ	OpAA	OpAB	OpAC	OpAD	OpAE	OpAF	OpAG	OpAH	OpAI	OpAJ	OpAK	OpAL	OpAM	OpAN	OpAO	OpAP	OpAQ	OpAR	OpAS	OpAT	OpAU	OpAV	OpAW	OpAX	OpAY	OpAZ	OpBA	OpBB	OpBC	OpBD	OpBE	OpBF	OpBG	OpBH	OpBI	OpBJ	OpBK	OpBL	OpBM	OpBN	OpBO	OpBP	OpBQ	OpBR	OpBS	OpBT	OpBU	OpBV	OpBW	OpBX	OpBY	OpBZ	OpCA	OpCB	OpCC	OpCD	OpCE	OpCF	OpCG	OpCH	OpCI	OpCJ	OpCK	OpCL	OpCM	OpCN	OpCO	OpCP	OpCQ	OpCR	OpCS	OpCT	OpCU	OpCV	OpCW	OpCX	OpCY	OpCZ	OpDA	OpDB	OpDC	OpDD	OpDE	OpDF	OpDG	OpDH	OpDI	OpDJ	OpDK	OpDL	OpDM	OpDN	OpDO	OpDP	OpDQ	OpDR	OpDS	OpDT	OpDU	OpDV	OpDW	OpDX	OpDY	OpDZ	OpEA	OpEB	OpEC	OpED	OpEE	OpEF	OpEG	OpEH	OpEI	OpEJ	OpEK	OpEL	OpEM	OpEN	OpEO	OpEP	OpEQ	OpER	OpES	OpET	OpEU	OpEV	OpEW	OpEX	OpEY	OpEZ	OpFA	OpFB	OpFC	OpFD	OpFE	OpFF	OpFG	OpFH	OpFI	OpFJ	OpFK	OpFL	OpFM	OpFN	OpFO	OpFP	OpFQ	OpFR	OpFS	OpFT	OpFU	OpFV	OpFW	OpFX	OpFY	OpFZ	OpGA	OpGB	OpGC	OpGD	OpGE	OpGF	OpGG	OpGH	OpGI	OpGJ	OpGK	OpGL	OpGM	OpGN	OpGO	OpGP	OpGQ	OpGR	OpGS	OpGT	OpGU	OpGV	OpGW	OpGX	OpGY	OpGZ	OpHA	OpHB	OpHC	OpHD	OpHE	OpHF	OpHG	OpHH	OpHI	OpHJ	OpHK	OpHL	OpHM	OpHN	OpHO	OpHP	OpHQ	OpHR	OpHS	OpHT	OpHU	OpHV	OpHW	OpHX	OpHY	OpHZ	OpIA	OpIB	OpIC	OpID	OpIE	OpIF	OpIG	OpIH	OpII	OpIJ	OpIK	OpIL	OpIM	OpIN	OpIO	OpIP	OpIQ	OpIR	OpIS	OpIT	OpIU	OpIV	OpIW	OpIX	OpIY	OpIZ	OpJA	OpJB	OpJC	OpJD	OpJE	OpJF	OpJG	OpJH	OpJI	OpJJ	OpJK	OpJL	OpJM	OpJN	OpJO	OpJP	OpJQ	OpJR	OpJS	OpJT	OpJU	OpJV	OpJW	OpJX	OpJY	OpJZ	OpKA	OpKB	OpKC	OpKD	OpKE	OpKF	OpKG	OpKH	OpKI	OpKJ	OpKK	OpKL	OpKM	OpKN	OpKO	OpKP	OpKQ	OpKR	OpKS	OpKT	OpKU	OpKV	OpKW	OpKX	OpKY	OpKZ	OpLA	OpLB	OpLC	OpLD	OpLE	OpLF	OpLG	OpLH	OpLI	OpLJ	OpLK	OpLL	OpLM	OpLN	OpLO	OpLP	OpLQ	OpLR	OpLS	OpLT	OpLU	OpLV	OpLW	OpLX	OpLY	OpLZ	OpMA	OpMB	OpMC	OpMD	OpME	OpMF	OpMG	OpMH	OpMI	OpMJ	OpMK	OpML	OpMM	OpMN	OpMO	OpMP	OpMQ	OpMR	OpMS	OpMT	OpMU	OpMV	OpMW	OpMX	OpMY	OpMZ	OpNA	OpNB	OpNC	OpND	OpNE	OpNF	OpNG	OpNH	OpNI	OpNJ	OpNK	OpNL	OpNM	OpNN	OpNO	OpNP	OpNQ	OpNR	OpNS	OpNT	OpNU	OpNV	OpNW	OpNX	OpNY	OpNZ	OpOA	OpOB	OpOC	OpOD	OpOE	OpOF	OpOG	OpOH	OpOI	OpOJ	OpOK	OpOL	OpOM	OpON	OpOO	OpOP	OpOQ	OpOR	OpOS	OpOT	OpOU	OpOV	OpOW	OpOX	OpOY	OpOZ	OpPA	OpPB	OpPC	OpPD	OpPE	OpPF	OpPG	OpPH	OpPI	OpPJ	OpPK	OpPL	OpPM	OpPN	OpPO	OpPP	OpPQ	OpPR	OpPS	OpPT	OpPU	OpPV	OpPW	OpPX	OpPY	OpPZ	OpQA	OpQB	OpQC	OpQD	OpQE	OpQF	OpQG	OpQH	OpQI	OpQJ	OpQK	OpQL	OpQM	OpQN	OpQO	OpQP	OpQQ	OpQR	OpQS	OpQT	OpQU	OpQV	OpQW	OpQX	OpQY	OpQZ	OpRA	OpRB	OpRC	OpRD	OpRE	OpRF	OpRG	OpRH	OpRI	OpRJ	OpRK	OpRL	OpRM	OpRN	OpRO	OpRP	OpRQ	OpRR	OpRS	OpRT	OpRU	OpRV	OpRW	OpRX	OpRY	OpRZ	OpSA	OpSB	OpSC	OpSD	OpSE	OpSF	OpSG	OpSH	OpSI	OpSJ	OpSK	OpSL	OpSM	OpSN	OpSO	OpSP	OpSQ	OpSR	OpSS	OpST	OpSU	OpSV	OpSW
----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

NDXOUT
V04-000

NDXOUT -- Sort and store index entries

CHRCMP -- Compare two characters in internal fo

I 1
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 77
(11)

ND
VO

	51		04	88	000A7	BISB2	#4	RB	2684
			1A	11	000AA	BRB	10\$		
	00000055	8F	50	D1	000AC	8\$:	CMPL	CB, #85	2686
			05	12	000B3	BNEQ	9\$		
	51		02	88	000B5	BISB2	#2	RB	2690
			0C	11	000B8	BRB	10\$		
	0000004F	8F	50	D1	000BA	9\$:	CMPL	CB, #79	2692
			03	12	000C1	BNEQ	10\$		
	51		01	88	000C3	BISB2	#1	RB	2696
			BC	D6	000C6	10\$:	INCL	@XB	2706
			B3	11	000C9	BRB	7\$		2669
	00000041	8F	50	D1	000CB	11\$:	CMPL	CB, #65	2711
			0E	19	000D2	BLSS	12\$		
	0000005A	8F	50	D1	000D4	CMPL	CB, #90		
			05	14	000DB	BGTR	12\$		
	50		20	C0	000DD	ADDL2	#32, CB		2713
			03	11	000E0	BRB	13\$		
			BC	D7	000E2	12\$:	DECL	@CASECMP	2715
	51		54	C2	000E5	13\$:	SUBL2	RA, R1	2726
			51	D5	000E8	TSTL	R1		
51	51		51	DC	000EA	MOVPSL	R1		
			02	EF	000EC	EXTZV	#2, #2, R1, R1		
		14	BC	FF	A1	9E	000F1	MOVAB	-1(R1), @EMPHCMP
	00000061	8F	52	D1	000F6	CMPL	CA, #97		2732
			1F	19	000FD	BLSS	15\$		
	0000007A	8F	52	D1	000FF	CMPL	CA, #122		
			16	14	00106	BGTR	15\$		
	00000061	8F	50	D1	00108	CMPL	CB, #97		2734
			09	19	0010F	BLSS	14\$		
	0000007A	8F	50	D1	00111	CMPL	CB, #122		
			1A	15	00118	BLEQ	16\$		
	63		01	D0	0011A	14\$:	MOVL	#1, (R3)	2738
				04	0011D	RET			2734
	00000061	8F	50	D1	0011E	15\$:	CMPL	CB, #97	2741
			0D	19	00125	BLSS	16\$		
	0000007A	8F	50	D1	00127	CMPL	CB, #122		
			04	14	0012E	BGTR	16\$		
	63		01	CE	00130	MNEGL	#1, (R3)		2743
				04	00133	RET			
	50		52	C2	00134	16\$:	SUBL2	CA, R0	2745
			50	D5	00137	TSTL	R0		
			50	DC	00139	MOVPSL	R0		
50	50		02	EF	0013B	EXTZV	#2, #2, R0, R0		
			63	FF	A0	9E	00140	MOVAB	-1(R0), (R3)
				04	00144	RET			2747

: Routine Size: 325 bytes, Routine Base: \$CODE\$ + 0830

: 1953 2748 1 END !End of module
: 1954 2749 0 ELUDOM

.EXTRN LIB\$SIGNAL

NDXOUT
V04-000

NDXOUT -- Sort and store index entries
CHRCMP -- Compare two characters in internal fo

J 1
16-Sep-1984 01:04:24
14-Sep-1984 13:07:15

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]NDXOUT.BLI;1

Page 78
(11)

PSECT SUMMARY

Name	Bytes	Attributes
\$OWNS	1232	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODES	2421	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$SPLITS	4	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]XPORT.L32;1	590	42	7	252	00:00.1

COMMAND QUALIFIERS

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

: Size: 2421 code + 1236 data bytes
: Run Time: 00:59.8
: Elapsed Time: 01:59.5
: Lines/CPU Min: 2760
: Lexemes/CPU-Min: 38622
: Memory Used: 232 pages
: Compilation Complete

0344 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

0345 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

